LSTMMilestone1

Python · Popular Video Games 1980 - 2023

Run
2.7s

Version 1 of 1

Add Tags

In [49]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

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

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

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

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

```
/kaggle/input/popular-video-games-1980-2023/games.csv
```

Importing Libraries

In [50]:
```
#Importing Libraries
import tensorflow as tf
from wordcloud import WordCloud
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.layers import Dense, Dropout
from keras.layers import LSTM
from keras.models import Sequential
from keras.layers import Embedding
from keras.layers import Flatten
from keras.layers import Bidirectional
from keras.callbacks import EarlyStopping
from keras.layers import GlobalAvgPool1D
```

In [51]:
```
train = pd.read_csv("/kaggle/input/popular-video-games-1980-2023/games.csv")
```

In [52]:
```
train.head()
```

Out[52]:

	Unnamed: 0	Title	Release Date	Team	Rating	Times Listed	Number of Reviews	Genres	Summary	Reviews	Plays
0	0	Elden Ring	Feb 25, 2022	['Bandai Namco Entertainment', 'FromSoftware']	4.5	3.9K	3.9K	['Adventure', 'RPG']	Elden Ring is a fantasy, action and open world...	["The first playthrough of elden ring is one o...	17K
1	1	Hades	Dec 10, 2019	['Supergiant Games']	4.3	2.9K	2.9K	['Adventure', 'Brawler', 'Indie', 'RPG']	A rogue-lite hack and slash dungeon crawler in...	['convinced this is a roguelike for people who...	21K
2	2	The Legend of Zelda: Breath of the Wild	Mar 03, 2017	['Nintendo', 'Nintendo EPD Production Group No...	4.4	4.3K	4.3K	['Adventure', 'RPG']	The Legend of Zelda: Breath of the Wild is the...	["This game is the game (that is not CS:GO) th...	30K
3	3	Undertale	Sep 15, 2015	['tobyfox', '8-4']	4.2	3.5K	3.5K	['Adventure', 'Indie', 'RPG', 'Turn Based Stra...	A small child falls into the Underground, wher...	['soundtrack is tied for #1 with nier automata...	28K
4	4	Hollow Knight	Feb 24, 2017	['Team Cherry']	4.4	3K	3K	['Adventure', 'Indie', 'Platform']	A 2D metroidvania with an emphasis on close co...	["this games worldbuilding is incredible, with...	21K

In [53]:
```
train.isnull().sum()
```

Out[53]:

```
Unnamed: 0          0
Title               0
Release Date        0
Team                1
Rating             13
Times Listed        0
Number of Reviews   0
Genres              0
Summary             1
Reviews             0
```

```
Plays                0
Playing              0
Backlogs             0
Wishlist             0
dtype: int64
```

In [54]:
```
train = train.dropna()
```

In [55]:
```
train.drop(['Unnamed: 0', 'Title', 'Release Date', 'Team', 'Times Listed',
       'Number of Reviews', 'Genres', 'Summary','Plays', 'Playing',
       'Backlogs', 'Wishlist'],axis = 1,inplace=True)
```

In [56]:
```
train.isnull().sum()
```

Out[56]:
```
Rating     0
Reviews    0
dtype: int64
```

Tokenizing with NLTK

In [58]:
```
def tokenization(inputs):
    return word_tokenize(inputs) #REFERENCE[1]


train['text_tokenized'] = train['Reviews'].apply(tokenization)
```

In [59]:
```
train.head()
```

Out[59]:

	Rating	Reviews	text_tokenized
0	4.5	["The first playthrough of elden ring is one o...	[[, ``, The, first, playthrough, of, elden, ri...
1	4.3	['convinced this is a roguelike for people who...	[[, 'convinced, this, is, a, roguelike, for, p...
2	4.4	['This game is the game (that is not CS:GO) th...	[[, 'This, game, is, the, game, (, that, is, n...
3	4.2	['soundtrack is tied for #1 with nier automata...	[[, 'soundtrack, is, tied, for, #, 1, with, ni...
4	4.4	["this games worldbuilding is incredible, with...	[[, ``, this, games, worldbuilding, is, incred...

Stopwords Removal

In [60]:
```
stop_words = set(stopwords.words('english'))

def stopwords_remove(inputs):
    return [item for item in inputs if item not in stop_words]

train['text_stop'] = train['text_tokenized'].apply(stopwords_remove)
train.head()
```

Out[60]:

	Rating	Reviews	text_tokenized	text_stop
0	4.5	["The first playthrough of elden ring is one o...	[[, ``, The, first, playthrough, of, elden, ri...	[[, ``, The, first, playthrough, elden, ring, ...
1	4.3	['convinced this is a roguelike for people who...	[[, 'convinced, this, is, a, roguelike, for, p...	[[, 'convinced, roguelike, people, like, genre...
2	4.4	['This game is the game (that is not CS:GO) th...	[[, 'This, game, is, the, game, (, that, is, n...	[[, 'This, game, game, (, CS, :, GO,), I, pla...
3	4.2	['soundtrack is tied for #1 with nier automata...	[[, 'soundtrack, is, tied, for, #, 1, with, ni...	[[, 'soundtrack, tied, #, 1, nier, automata, ...
4	4.4	["this games worldbuilding is incredible, with...	[[, ``, this, games, worldbuilding, is, incred...	[[, ``, games, worldbuilding, incredible, ,, a...

Lemmatization

In [ ]:
```
!unzip /usr/share/nltk_data/corpora/wordnet.zip -d /usr/share/nltk_data/corpora/
```

In [63]:
```
lemmatizer = WordNetLemmatizer()

def lemmatization(inputs):
    return [lemmatizer.lemmatize(word=x, pos='v') for x in inputs]

train['text_lemmatized'] = train['text_stop'].apply(lemmatization)

train.head()
```

Out[63]:

	Rating	Reviews	text_tokenized	text_stop	text_lemmatized
0	4.5	["The first playthrough of elden ring is one o...	[[, ``, The, first, playthrough, of, elden, ri...	[[, ``, The, first, playthrough, elden, ring, ...	[[, ``, The, first, playthrough, elden, ring, ...
1	4.3	['convinced this is a roguelike for people who...	[[, 'convinced, this, is, a, roguelike, for, p...	[[, 'convinced, roguelike, people, like, genre...	[[, 'convinced, roguelike, people, like, genre...
2	4.4	['This game is the game (that is not CS:GO) th...	[[, 'This, game, is, the, game, (, that, is, n...	[[, 'This, game, game, (, CS, :, GO,), I, pla...	[[, 'This, game, game, (, CS, :, GO,), I, pla...
3	4.2	['soundtrack is tied for #1 with nier automata...	[[, 'soundtrack, is, tied, for, #, 1, with, ni...	[[, 'soundtrack, tied, #, 1, nier, automata, ...	[[, 'soundtrack, tie, #, 1, nier, automata, ,....
4	4.4	["this games worldbuilding is incredible, with...	[[, ``, this, games, worldbuilding, is, incred...	[[, ``, games, worldbuilding, incredible, ,, a...	[[, ``, game, worldbuilding, incredible, ,, am...

Joining Tokens into Sentences

In [64]:
```
train['text_cleaned'] = train['text_lemmatized'].str.join(' ')
train.head()
```

Out[64]:

	Rating	Reviews	text_tokenized	text_stop	text_lemmatized	text_cleaned
0	4.5	["The first playthrough of elden ring is one o...	[[, ``, The, first, playthrough, of, elden, ri...	[[, ``, The, first, playthrough, elden, ring, ...	[[, ``, The, first, playthrough, elden, ring, ...	[`` The first playthrough elden ring one best...
1	4.3	['convinced this is a roguelike for people who...	[[, 'convinced, this, is, a, roguelike, for, p...	[[, 'convinced, roguelike, people, like, genre...	[[, 'convinced, roguelike, people, like, genre...	['convinced roguelike people like genre . The...
2	4.4	['This game is the				

2	4.4	game (that is not CS:GO) th...	[[, 'This, game, is, the, game, (, that, is, n...	[[, 'This, game, game, (, CS, :, GO,), I, pla...	[[, 'This, game, game, (, CS, :, GO,), I, pla...	['This game game (CS : GO) I play ever . I ...
3	4.2	['soundtrack is tied for #1 with nier automata...	[[, 'soundtrack, is, tied, for, #, 1, with, ni...	[[, 'soundtrack, tied, #, 1, nier, automata, ...	[[, 'soundtrack, tie, #, 1, nier, automata, .,...	['soundtrack tie # 1 nier automata . super ch...
4	4.4	['this games worldbuilding is incredible, with...	[[, '``, this, games, worldbuilding, is, incred...	[[, '``, games, worldbuilding, incredible, ,, a...	[[, '``, game, worldbuilding, incredible, ,, am...	['`` game worldbuilding incredible , amaze sou...

In [65]:
```
WordCloud = WordCloud(max_words=100,
                      random_state=30,
                      collocations=True).generate(str((train['text_cleaned'])))

plt.figure(figsize=(15, 8))
plt.imshow(WordCloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Tokenizing with Tensorflow

In [66]:
```
num_words = 10000
tokenizer = Tokenizer(num_words=num_words, oov_token='<OOV>')
tokenizer.fit_on_texts(train['text_cleaned'])

word_index = tokenizer.word_index
```

In [67]:
```
Tokenized_train = tokenizer.texts_to_sequences(train['text_cleaned'])
```

Padding

In [69]:
```
maxlen = 40
Padded_train = pad_sequences(Tokenized_train, maxlen=maxlen, padding='pre')
print('Non-padded Version: ', tokenizer.texts_to_sequences([train['text_cleaned'][0]]))
print('Padded Version: ', Padded_train[0])
print('--'*50)
print('Non-padded Version: ', tokenizer.texts_to_sequences([train['text_cleaned'][10]]))
print('Padded Version: ', Padded_train[10])
```

```
Non-padded Version:  [[9, 33, 510, 2010, 1045, 16, 32, 5649, 2, 645, 4671, 227, 104, 241, 51, 5
8, 60, 467, 296, 2281, 37, 227, 1085, 1314, 72, 38, 409, 1736, 1086, 128, 2, 615, 152, 347, 467
2, 36, 2010, 1045, 378, 115, 2, 14, 95, 114, 5650, 14, 43, 133, 24, 36, 51, 153, 646, 14, 3, 75,
2, 277, 244, 26, 9, 1736, 700, 178, 71, 17, 306, 3, 2646, 104, 31, 1584, 2, 18, 294, 47, 3, 42,
7, 5651, 468, 67, 14, 28, 1014, 350, 25, 5652, 1585, 383, 2, 632, 5, 5, 937, 274, 920, 104, 2, 2
44, 825, 783, 17, 172, 1120, 101, 784, 4, 130, 247, 7, 93, 1458, 25, 63, 223, 5653, 4004, 484,
3, 7710, 115, 404, 2, 1121, 314, 16, 4, 202, 175, 80, 76, 11]]
Padded Version:  [   5  937  274  920  104    2  244  825  783   17  172 1120  101  784
    4  130  247    7   93 1458   25   63  223 5653 4004  484    3 7710
  115  404    2 1121  314   16    4  202  175   80   76   11]
--------------------------------------------------------------------------------------------
----
Non-padded Version:  [[5752, 2133, 2655, 398, 29, 2297, 726, 4038, 4697, 4698, 3, 44, 2, 31, 94,
105, 140, 16, 646, 1526, 142, 2, 14, 4, 64, 294, 4, 543, 5753, 5, 5, 9, 346, 2, 5, 5, 9, 291, 1
7, 19, 1055, 1655, 124, 2, 5, 5, 9, 291, 2134, 3153, 2020, 20, 146, 995, 681, 5754, 307, 329, 66
8, 298, 5, 5, 9, 251, 1365, 63, 5, 5, 765, 707, 2298, 2298, 62, 268, 707, 472, 2, 1177, 1464, 27
9, 131, 1120, 2656, 242, 5755, 183, 899, 1527, 1750, 242, 472, 88, 2299, 852, 279, 4, 70, 1594,
1527, 4, 184, 10, 1465, 3519, 2, 3154, 5756, 93, 3154, 2656, 607, 963, 830, 17, 4699, 2656, 230
0, 4, 411, 4700, 727, 513, 2, 5757, 5, 5, 178, 2020, 3155, 5758, 6, 34, 1366, 688, 5759, 4, 63,
1751, 11, 5760, 33, 472, 707, 2, 155, 24, 8, 80, 647, 1223, 7, 3, 964, 4039, 294, 124, 1363, 12
0, 45, 19, 132, 47, 327, 3, 42, 36, 3520, 3521, 3, 275, 8, 1528, 4701, 4, 424, 157, 207, 215, 35
22, 708, 59, 4, 10, 8, 472, 88, 648, 5761, 5762, 120, 34, 2021, 17, 203, 4, 7, 707, 2, 339, 26,
5763, 4040, 177, 2876, 228, 4, 59, 472, 88, 74, 2, 555, 60, 68, 965, 4, 944, 520, 109, 8, 88, 13
6, 124, 109, 234, 50, 387, 197, 34, 792, 325, 269, 257, 472, 16, 351, 783, 50, 407, 71, 387, 31
8, 157, 2022, 243, 327, 1223, 966, 80, 76, 11]]
Padded Version:  [ 555   60   68  965    4  944  520  109    8   88  136  124  109  234
   50  387  197   34  792  325  269  257  472   16  351  783   50  407
   71  387  318  157 2022  243  327 1223  966   80   76   11]
```

Creating the Model

In [70]:
```
model = Sequential()

model.add(Embedding(num_words, 16, input_length=maxlen))
model.add(GlobalAvgPool1D())

tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(50, return_sequences=True, activation='relu'))
model.add(Dropout(0.3))

tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(40, activation='relu', return_sequences=True))
model.add(Dropout(0.3))

tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(40, activation='relu'))
model.add(Dropout(0.3))

model.add(Dense(6, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 40, 16)            160000

 global_average_pooling1d_1   (None, 16)               0
 (GlobalAveragePooling1D)

 dropout_3 (Dropout)         (None, 16)                0

 dropout_4 (Dropout)         (None, 16)                0

 dropout_5 (Dropout)         (None, 16)                0

 dense_1 (Dense)             (None, 6)                 102

=================================================================
Total params: 160,102
Trainable params: 160,102
Non-trainable params: 0
_________________________________________________________________
```

In [72]:

```
from sklearn.model_selection import train_test_split

# Let's say we want to split the data in 80:10:10 for train:valid:test dataset
train_size=0.8

X = Padded_train
y = train["Rating"].values

# In the first step we will split the data in training and remaining dataset
X_train, X_rem, y_train, y_rem = train_test_split(X,y, train_size=0.8,random_state=0)

# Now since we want the valid and test size to be equal (10% each of overall data).
# we have to define valid_size=0.5 (that is 50% of remaining data)
test_size = 0.5
X_valid, X_test, y_valid, y_test = train_test_split(X_rem,y_rem, test_size=0.5,random_state=0)

print(X_train.shape), print(y_train.shape)
print(X_valid.shape), print(y_valid.shape)
print(X_test.shape), print(y_test.shape)
```

```
(1198, 40)
(1198,)
(150, 40)
(150,)
(150, 40)
(150,)
```

Out[72]:

```
(None, None)
```

In [74]:

```
epochs = 20
hist = model.fit(X_train,y_train, epochs=epochs,
                 validation_data=(X_valid, y_valid),
                 )
```

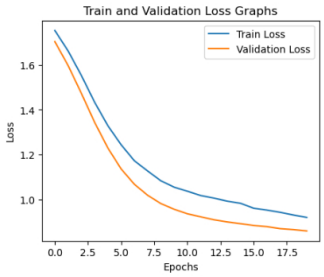
```
Epoch 1/20
38/38 [==============================] - 1s 12ms/step - loss: 1.7526 - accuracy: 0.0376 - val_lo
ss: 1.7031 - val_accuracy: 0.0400
Epoch 2/20
38/38 [==============================] - 0s 5ms/step - loss: 1.6617 - accuracy: 0.0317 - val_los
s: 1.5967 - val_accuracy: 0.0333
Epoch 3/20
38/38 [==============================] - 0s 5ms/step - loss: 1.5525 - accuracy: 0.0284 - val_los
s: 1.4730 - val_accuracy: 0.0333
Epoch 4/20
38/38 [==============================] - 0s 5ms/step - loss: 1.4334 - accuracy: 0.0309 - val_los
s: 1.3439 - val_accuracy: 0.0333
Epoch 5/20
38/38 [==============================] - 0s 5ms/step - loss: 1.3287 - accuracy: 0.0309 - val_los
s: 1.2295 - val_accuracy: 0.0333
Epoch 6/20
38/38 [==============================] - 0s 5ms/step - loss: 1.2440 - accuracy: 0.0259 - val_los
s: 1.1363 - val_accuracy: 0.0333
Epoch 7/20
38/38 [==============================] - 0s 5ms/step - loss: 1.1726 - accuracy: 0.0225 - val_los
s: 1.0680 - val_accuracy: 0.0333
Epoch 8/20
38/38 [==============================] - 0s 5ms/step - loss: 1.1272 - accuracy: 0.0326 - val_los
s: 1.0191 - val_accuracy: 0.0333
Epoch 9/20
38/38 [==============================] - 0s 6ms/step - loss: 1.0835 - accuracy: 0.0384 - val_los
s: 0.9821 - val_accuracy: 0.0333
Epoch 10/20
38/38 [==============================] - 0s 5ms/step - loss: 1.0552 - accuracy: 0.0275 - val_los
s: 0.9569 - val_accuracy: 0.0333
Epoch 11/20
38/38 [==============================] - 0s 5ms/step - loss: 1.0371 - accuracy: 0.0267 - val_los
s: 0.9367 - val_accuracy: 0.0333
Epoch 12/20
38/38 [==============================] - 0s 5ms/step - loss: 1.0181 - accuracy: 0.0359 - val_los
s: 0.9233 - val_accuracy: 0.0333
Epoch 13/20
38/38 [==============================] - 0s 5ms/step - loss: 1.0065 - accuracy: 0.0326 - val_los
s: 0.9100 - val_accuracy: 0.0333
Epoch 14/20
38/38 [==============================] - 0s 5ms/step - loss: 0.9930 - accuracy: 0.0426 - val_los
s: 0.9005 - val_accuracy: 0.0333
Epoch 15/20
38/38 [==============================] - 0s 5ms/step - loss: 0.9835 - accuracy: 0.0359 - val_los
s: 0.8928 - val_accuracy: 0.0333
Epoch 16/20
38/38 [==============================] - 0s 5ms/step - loss: 0.9615 - accuracy: 0.0426 - val_los
s: 0.8848 - val_accuracy: 0.0333
Epoch 17/20
38/38 [==============================] - 0s 5ms/step - loss: 0.9532 - accuracy: 0.0459 - val_los
s: 0.8794 - val_accuracy: 0.0333
Epoch 18/20
38/38 [==============================] - 0s 5ms/step - loss: 0.9431 - accuracy: 0.0409 - val_los
```

```
s: 0.8705 - val_accuracy: 0.0333
Epoch 19/20
38/38 [==============================] - 0s 6ms/step - loss: 0.9312 - accuracy: 0.0518 - val_los
s: 0.8664 - val_accuracy: 0.0333
Epoch 20/20
38/38 [==============================] - 0s 5ms/step - loss: 0.9206 - accuracy: 0.0459 - val_los
s: 0.8605 - val_accuracy: 0.0400
```

Train and Validation Loss Graphs

In [75]:
```
plt.figure(figsize=(5, 4))
plt.plot(hist.history['loss'], label='Train Loss')
plt.plot(hist.history['val_loss'], label='Validation Loss')
plt.title('Train and Validation Loss Graphs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

Out[75]:
```
<matplotlib.legend.Legend at 0x7bef5c767a30>
```



In [ ]:
```
```

Continue exploring

Input
1 file →

Output
0 files →

Logs
2.7 second run - successful →

Comments
0 comments →