CSE0440 ARTIFICIAL INTELLIGENCE

TERM PROJECT

ALEYNA ERKİŞİ - 1600002059

PROJECT TOPIC

Sentiment Analysis of Steam Game Reviews

LINK TO THE DATASET

- Steam Review Dataset (from dataset search - google)
- Steam Review Dataset (from source)

PROJECT INFO

The dataset contains over 6.4 million publicly available reviews in English from Steam Reviews portion of Steam. Each review is described by review text, the id of game it belongs to, review sentiment (positive or negative) and a number of users who tough review was helpful. My project will focus on sentiment analysis on this dataset similar to what we have learnt in Lab07.

DATA DISCOVERY

Dataset contains 4 categories. GameID that contains the unique id set by Steam Store. Review that contains the text of the review itself. Review Sentiment that displays whether the review is positive or negative. Lastly whether the review is helpful or not is shown in the Helpful row.

	GameID	Review	ReviewSentiment	Helpful
0	10	Ruined my life.	1	0
1	10	This will be more of a "my experience with th...	1	1
2	10	This game saved my virginity.	1	0
3	10	• Do you like original games? • Do you like ga...	1	0
4	10	Easy to learn, hard to master.	1	1
5	10	No r8 revolver, 10/10 will play again.	1	1
6	10	Still better than Call of Duty: Ghosts...	1	1
7	10	cant buy skins, cases, keys, stickers - gaben ...	1	1
8	10	Counter-Strike: Ok, after 9 years of unlimited...	1	1
9	10	Every server is spanish or french. I can now f...	1	0
10	10	Fire in the Hole Simulator 1999	1	0
11	10	I never played a better first person shooter. ...	1	1
12	10	WARNING : DO NOT buy this game, if you do, you...	1	1
13	10	CS is one of the greatest and undermined games...	1	1
14	10	It's... aight.	1	1
15	10	I met a lot of people who slept with my mother...	1	1
16	10	Ruined my life. 10/10	1	1
17	10	When you crouch, you lift your feet in the air...	1	1
18	10	Best shooter for 10 years. Steam evolved becau...	1	0
19	10	One of the best FPS Games, the 1.6 Version wil...	1	1

Figure 1: Data Sample from the Dataset

```
df.describe()
```

	GameID	ReviewSentiment	Helpful
count	6.417106e+06	6.417106e+06	6.417106e+06
mean	2.274695e+05	6.394992e-01	1.472446e-01
std	1.260451e+05	7.687918e-01	3.543496e-01
min	1.000000e+01	-1.000000e+00	0.000000e+00
25%	2.018100e+05	1.000000e+00	0.000000e+00
50%	2.391600e+05	1.000000e+00	0.000000e+00
75%	3.056200e+05	1.000000e+00	0.000000e+00
max	5.653400e+05	1.000000e+00	1.000000e+00

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6417106 entries, 0 to 6417105
Data columns (total 4 columns):
 #   Column           Dtype
---  ------           -----
 0   GameID           int64
 1   Review           object
 2   ReviewSentiment  int64
 3   Helpful          int64
dtypes: int64(3), object(1)
memory usage: 195.8+ MB
```

Figure 2: Dataset Info

```
print("Number of games : {0}".format(len(df.GameID.unique())))
```

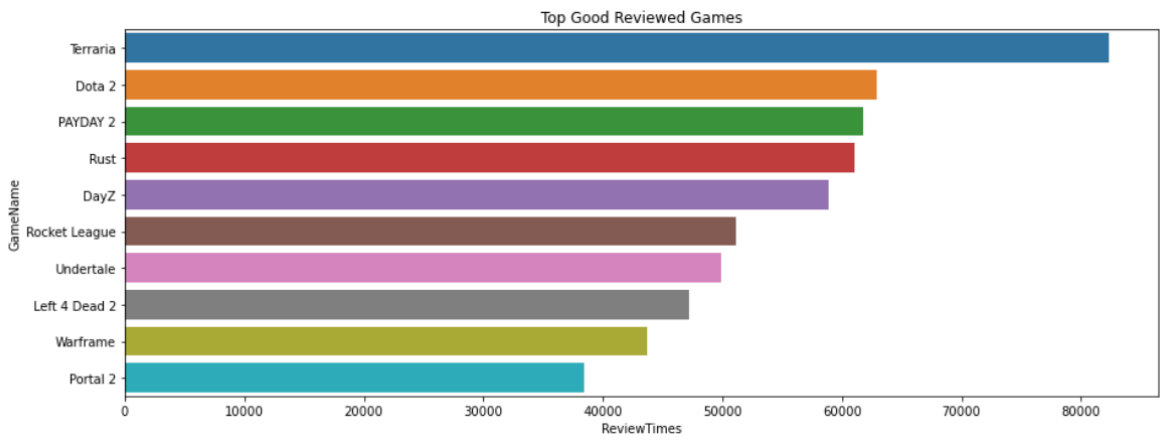
```
Number of games : 9972
```

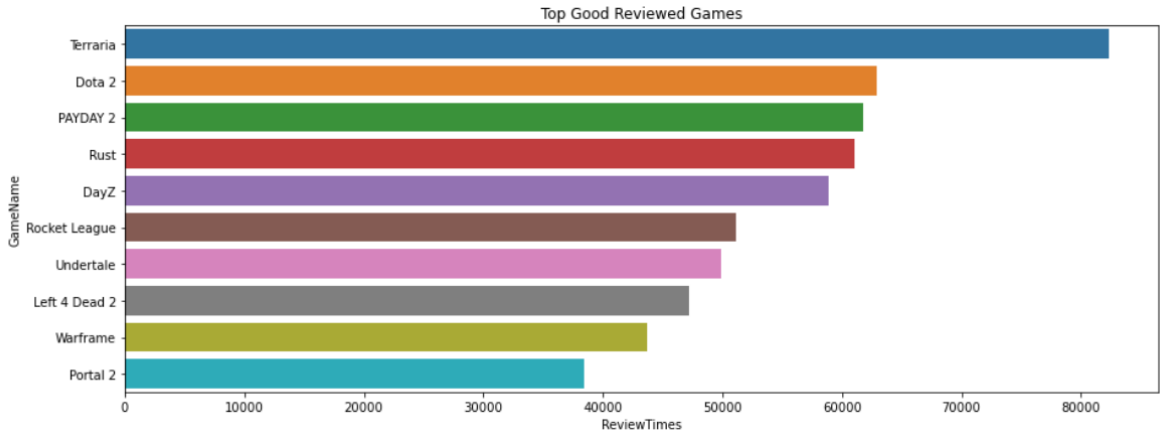
```
print("Number of positive reviews : {0}".format(len(df.loc[df['ReviewSentiment'] == 1])))
print("Number of negative reviews : {0}".format(len(df.loc[df['ReviewSentiment'] == -1])))
print("Number of reviews : {0}".format(len(df.Review)))
```

```
Number of positive reviews : 5260420
Number of negative reviews : 1156686
Number of reviews : 6417106
```

Figure 3: Some Numbers About The Dataset

TOP REVIEWED GAMES





DATA PREPARATION

- Replacing the -1 values with 0 (Binary replacement) (Normalization)
- Converting the data type to np.str to use transform
- Splitting the data 33% test and 67% train.

```
df['ReviewSentiment'] = df['ReviewSentiment'].replace(-1, 0)
```

```
reviews = df['Review'].apply(lambda x: np.str_(x)).values
labels = df['ReviewSentiment'].values
```

```
train_data, test_data, y_train, y_test = train_test_split(reviews,
                                                          labels,
                                                          test_size=0.33,
                                                          )
```

BAG OF WORDS (BOW)

- Creating the BOW with Count Vectorizer

The dataset contains 5M+ reviews. 67% of it is in the train_data. If I try to create the vectorizer without the max_features argument, it will either take forever or I will get a memory error. Latter is more likely to happen. So, with the help of max_features argument, which only considers the features with highest term frequency, I only take the most frequent thousand words. Then, first, I used fit() on my vectorizer and to create a BOW representation, I used the transform() method later.

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
vectorizer = CountVectorizer(max_features=1000)
vectorizer.fit(train_data)
```

```
CountVectorizer(max_features=1000)
```

```
vectorizer.vocabulary_
```

```
{'hate': 407,
 'this': 867,
 'game': 360,
 'ending': 276,
 'is': 459,
 'great': 387,
 'cant': 141,
 'wait': 932,
 'for': 344,
 'the': 855,
 'full': 354,
 'based': 86,
 'it': 463,
 'just': 472,
 'well': 951,
 'worth': 981,
 'price': 686,
 'so': 795,
 'many': 537,
 'and': 54
```

MODELS

-LAYERS COMPARISON

A. Model 1: 1 input – 1 output

```
input_dim = X_train.shape[1]

model = Sequential()
model.add(Dense(128,activation='relu',input_dim=input_dim))
model.add(Dense(1,activation='sigmoid'))

model.summary()
```

B. Model 2: 5 input – 1 output using dropout as regularization technique

```
input_dim = X_train.shape[1]

model2 = Sequential()
model2.add(Dense(128,activation='relu',input_dim=input_dim))
model2.add(Dense(128,activation='relu'))
model2.add(Dropout(0.1))
model2.add(Dense(16,activation='elu'))
model2.add(Dense(1,activation='sigmoid'))

model2.summary()
```

C. Model 3: 4 input – 1 output using dropout and l2 as regularization technique

```
model3 = Sequential()
model3.add(Dense(128,kernel_regularizer=keras.regularizers.l2(0.001),activation='relu',input_dim=input_dim))
model3.add(Dropout(0.5))
model3.add(Dense(128,activation='relu',kernel_regularizer=keras.regularizers.l2(0.001)))
model3.add(Dropout(0.5))
model3.add(Dense(1,activation='sigmoid'))

model2.summary()
```

-TRAINING COMPARISON

A. Model 1: 10 epochs

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

```
model_history = model.fit(X_train, y_train, epochs=10,validation_data=(X_test,y_test))
```

B. Model 2: 5 epochs

```
model2.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
model2_history = model2.fit(X_train, y_train, epochs=5,validation_data=(X_test,y_test))
```

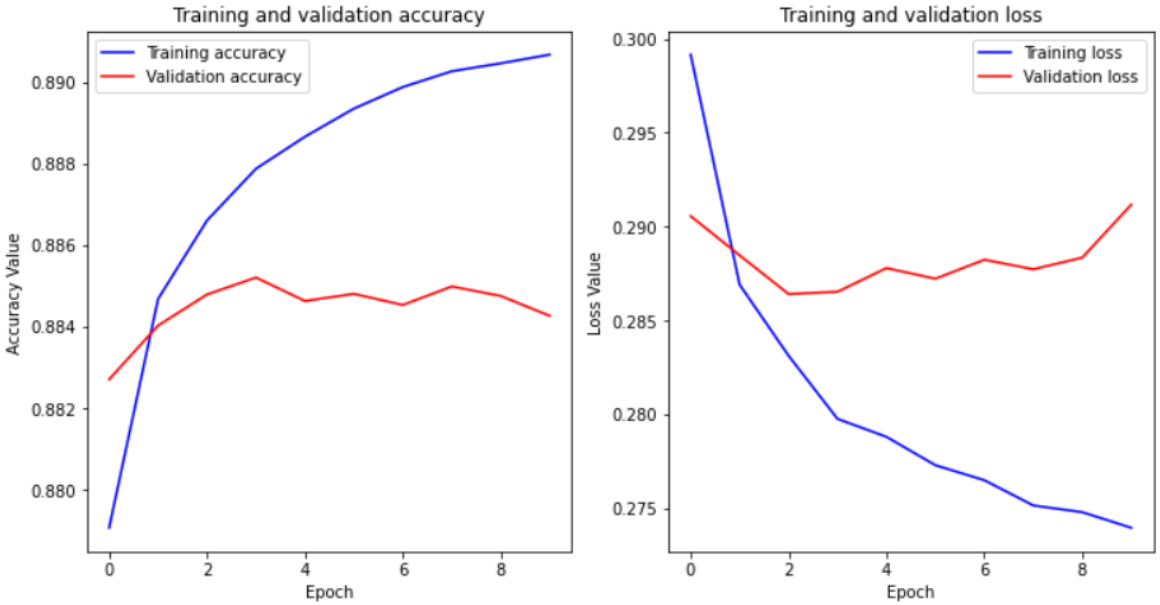
C. Model 3: 5 epochs

```
model3.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

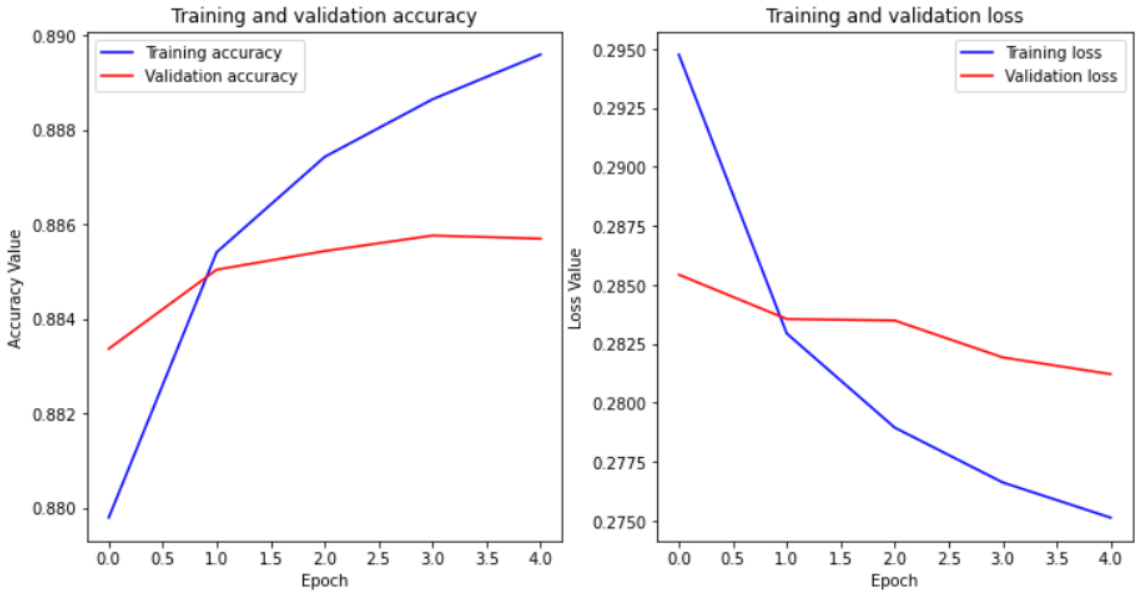
```
model3_history = model3.fit(X_train, y_train, epochs=5,validation_data=(X_test,y_test))
```

-LOSS and ACCURACY COMPARISON

A. Model 1: Training accuracy: 0.8922 - Test accuracy: 0.8843



B. Model 2: Training accuracy: 0.8920- Test accuracy: 0.8857



C. Model 3: Training accuracy: 0.8624- Test accuracy: 0.8627

CONCLUSION

I trained 3 models in the project and Model 1 as you can see stopped learning at some point and started memorizing. This is caused by overfitting. Thus in the Model 2 I used Dropout as regularization technique. But as you can see in the graphs, it was not enough and again after some point model does not learn. In the Model 3 I used l2 regularizer addition to dropout. This time even though my accuracy slightly decreased my model kept learning and it turned out to be the best model out of these 3.