Script analysis for extraction of script elements along with animation scene complexity

Ryan Amundson

https://www.linkedin.com/in/ryanamundson1
ryanamundson1@yahoo.com

DreamWorks Animation

1000 Flower St, Glendale, CA

Northwestern University School of Professional Studies

339 EAST CHICAGO AVENUE, CHICAGO, IL

“Cookies are for closers” - Boss Baby

Abstract

Our goal for this report is to analyze the script for Boss Baby: Family Business and use the complexity breakdown information for the completed movie to categorize complexity and find any script correlations. The script is analyzed by the technical leads for animation, visual effects and lighting to give a general score for the complexity by sequence, which will directly translate to the cost and schedule for the movie. The goal of this report to help improve the process by seeing if the script itself could contain information that will help inform the script breakdowns, by first classifying the basic elements of each sequence, like dialog, scene descriptions and characters by using a bag of n-grams and ensemble based tree learning for classification of each line and then extract complexity information via word vectors and a bidirectional LSTM model for classification using those scene elements.

Introduction

Animated movie production is a complicated process. From initial script to theatrical release for most big budget animated features, it will take on average three years and $100m to go from script to producing the final film. Script breakdowns are an important part of initially determining costs for a 3d animated movie. Each major production department will individually asses the script to come up with a picture of how much each sequence will impact their department. In breaking down the script, the animation department might look at elements like the number of characters in the sequence, or the emotional impact of the scene. The FX department would look at scene descriptions for visual effects elements like fire, explosions, water interactions, all of which could add complexity to the visual representation. In the report we are trying to assess the potential of using the movie script along with the knowledge and information that can added from working in the 3d animation field at DreamWorks to extract some of the elements of complexity directly from the script text using a bi-directional long-term short-term memory (LSTM) model. This could then be used to help inform the schedule early in the planning process and could help limit issues in what it currently a very manual process, as mistakes in resource planning when setting the budgets and schedule timeline can be very costly, as the movie release dates are made far in advance and are not very flexible and a bad release time because of schedule slippage can incur losses in the millions of dollars.

Literature Review

In doing research about this subject, there were many examples of review based sentiment analysis or script sentiment analysis, scene complexity is a slightly different focus, but I think similar methods could be used. However, because this is a unique concept that is more directly related to animated films, there might not be a broad understanding of how shot or sequence complexity is considered when projecting the budget or timeline of an animated feature film.

In “Sentiment Analysis on Movie Scripts and Reviews” (Hutto 2014) the authors used TF-IDF and VADER (Valence Aware Dictionary and sEntiment Reasoner) with NRC (National Research Council Canada affect lexicon) to perform sentiment analysis and then predict the sentiment with a Multinomial Naive Bayes predictor. I would also like to compare the sentiment values across sequences for my analysis, as this could impact animation complexity.

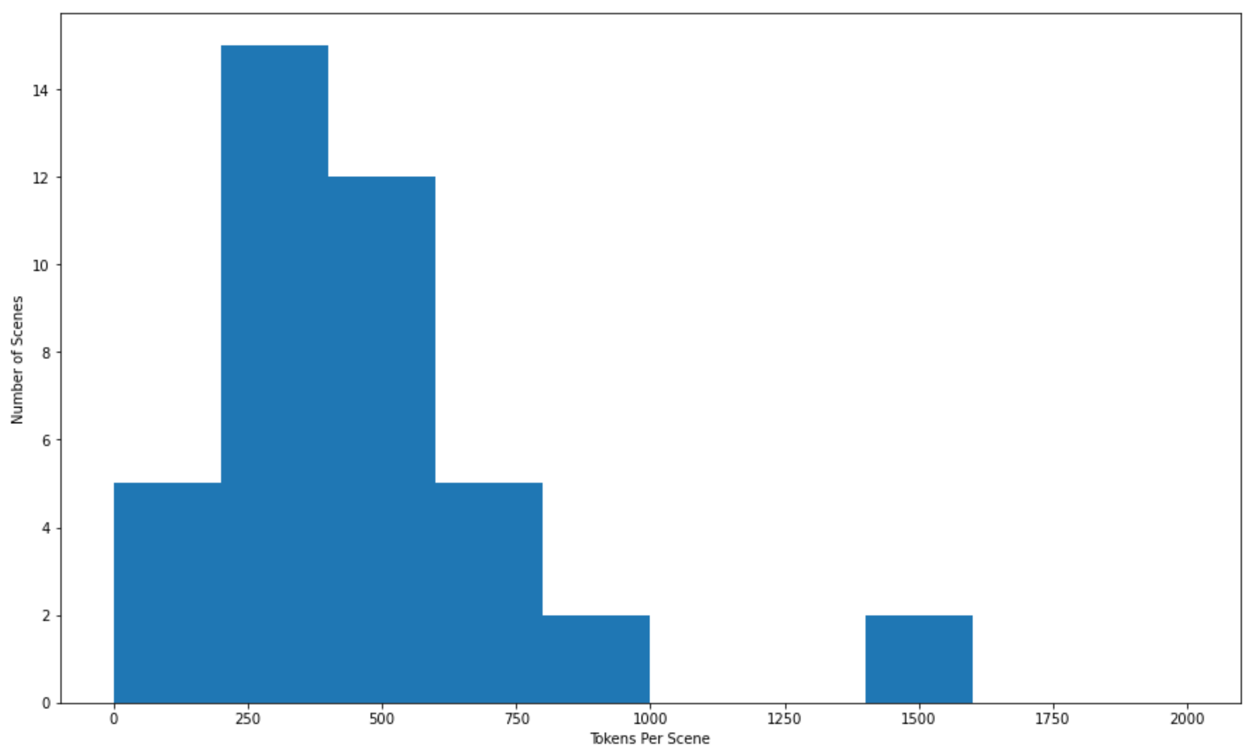
In “Analyzing Movie Scripts as Unstructured Text” (Lee 2017) the authors also use sentiment analysis, but are looking at blocks of text throughout each script. That way they could define graphs of the movie sentiment. The idea was that some movies could be defined as perhaps starting "happy" and ending "sad", thereby maybe being categorized as a tragedy. This is also conceptually interesting to me, and I wonder if major scene beats could be defined in such a way.

Additionally in "Multilayer Network Model of Movie Script" (Mourchid 2019), they logically separated each sequence into different logical components, including settings, characters and dialog. I think breaking up the script into similar components could help with the NLP process. There are some very consistent text characteristics of a script, for instance the speaking character is always uppercased, like "TIM" or "TABITHA" and is on a single line, while in the dialog, names would be expressed as "Tim" or "Tabitha". There are also often a lot of punctuation in dialog lines to express emotional sentiment. I think by maintaining casing, we could train an effective model that can discern these script elements.

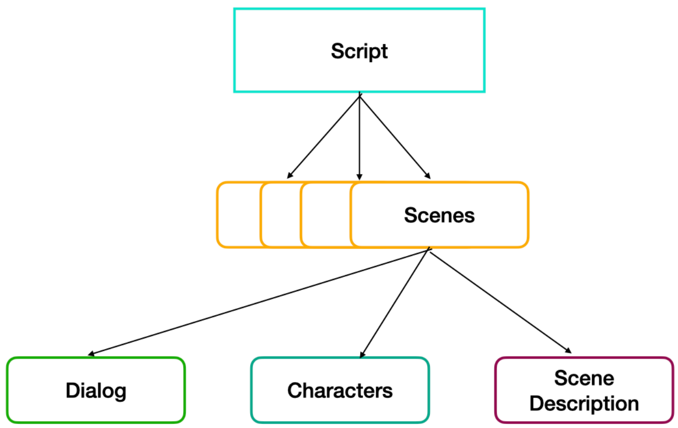
Data

The main source of data I am using is the script itself. I used this movie because the script is publicly available through Universal at: https://awards.universalpictures.com/the-boss-baby-family-business/screenplay/The_Boss_Baby_Family_Business.pdf. This data was partially hand coded to separate the dialog, characters and scene description data. I've additionally added scene complexity data created by the lead animators at DreamWorks, which is included as TABLE1 in the appendix.

The with the script separated into their individual scenes, you can see the number of words in each scene is fairly variable, with most around 250 to 750 words, but with a couple of outliers in the 1500 word range:



Here is the simplified ontological breakdown of the script data:



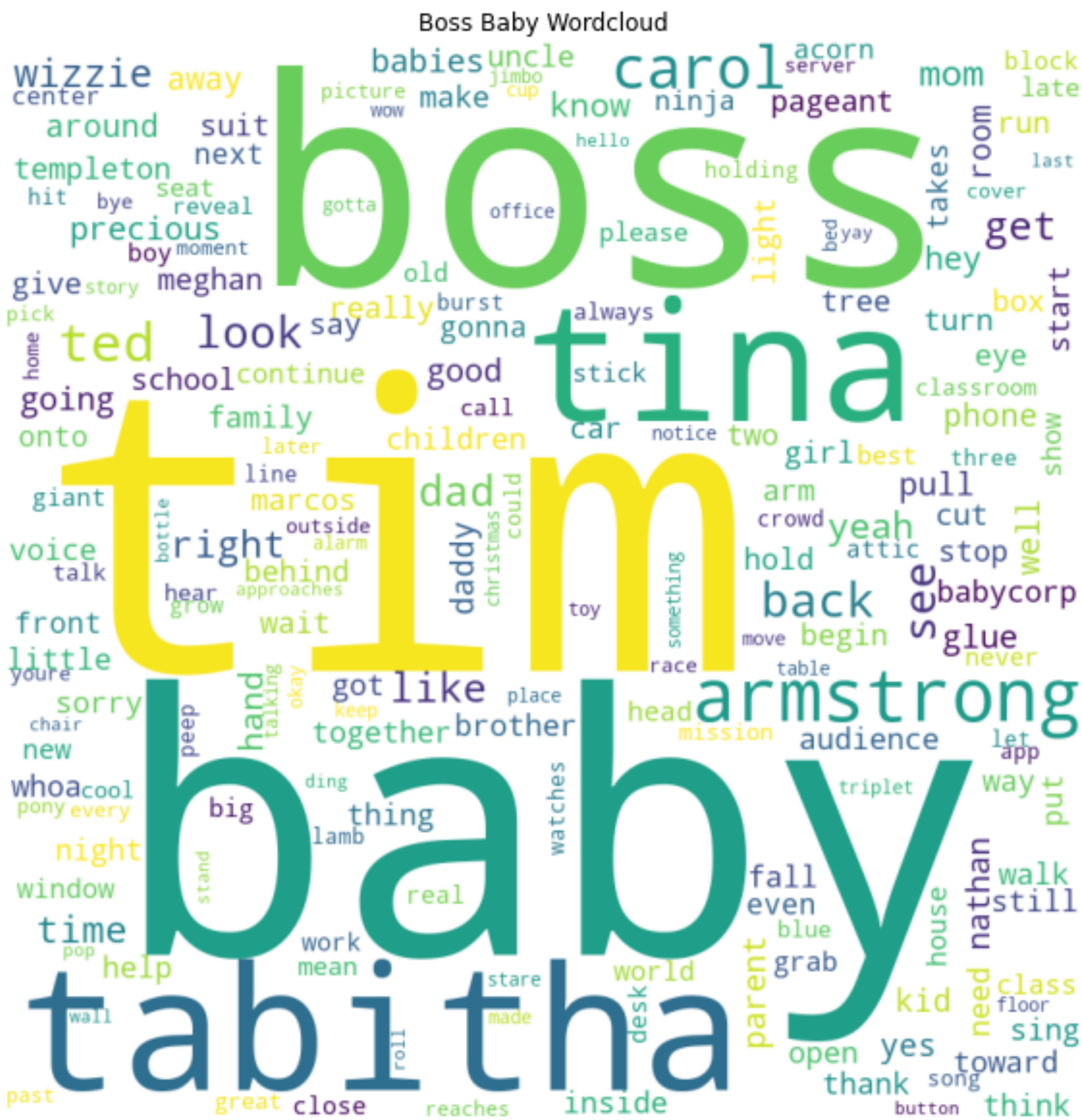
The script is comprised of 41 scenes (or sequences), each of which has some number of either character lines, dialog lines and some scene description lines.

Methods

Because the movie is a scanned script and not rich text data, the first processing that had to be done was to convert the image data into text. I used pytesseract (https://pypi.org/project/pytesseract/) and pdf2image (https://pypi.org/project/pdf2image/) to convert the PDF file to a sequence off images and then used an OCR tool to extract the text from each image to end up with text data.

The text was then split the script text up into it's individual sequences and used the sequence text as the document data. Then the documents were cleaned by splitting on whitespace and removing the punctuation and any two letter or smaller words and stop words and all tokens were then lowercased.

This is the word cloud of the resulting corpus:



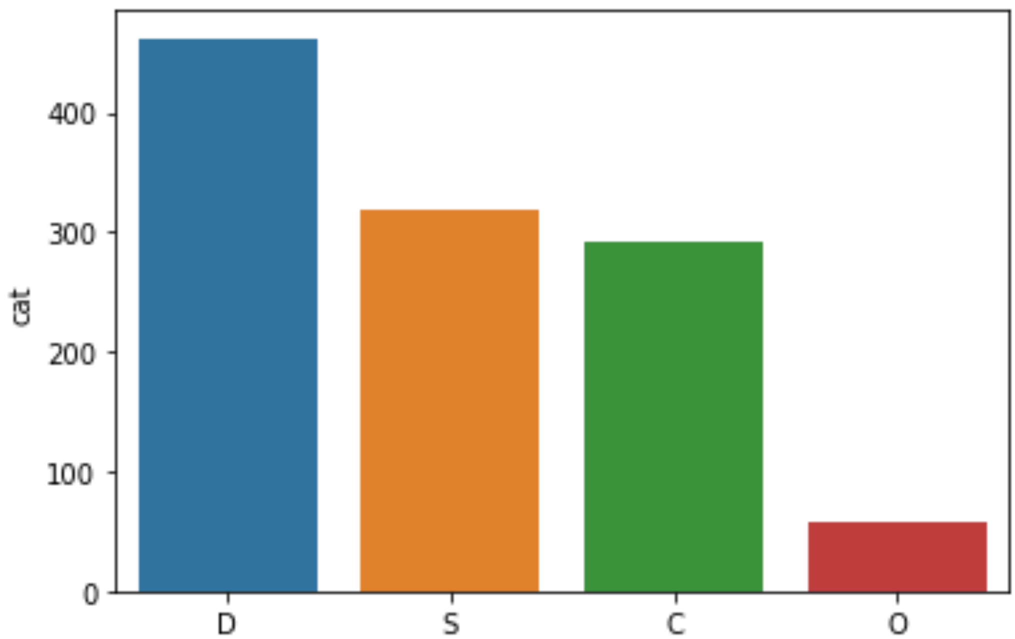
It was important to take into account shorter words because many of the character names and simple dialog for a PG animated movie needs to be taken into account. In the word cloud we can see the prevalence of character names, which makes sense as the main component of a script would be dialog, which is formatted like this in the script:

```
Land on SEVEN-YEAR-OLD TIM counting against a tree as his
parents and BOSS BABY hide in a bush.

                    TIM’S MOM
          Hide, hide, hide!

                    TIM’S DAD
          Quick! Over here!

                    YOUNG TIM
          Four, five, six...

                    TIM V.O. (CONT'D)
          But the years, well they went by so
          fast.

                    TIM’S MOM
          Tim, no peeking!

                    TIM’S DAD
          Yeah, no peeking!

The camera slowly circles the tree.
```

You can see in the script, there are several important components, first there are the scene descriptions and camera movement, like “The camera slowly circles the tree.” The script also contains the dialog, along with the list of characters who are speaking. As each of these are extracted as they contain separate meanings and might contribute differently to the scene complexity. The first step in the analysis will be to classify the lines in the script to the correct components.

Results

Because punctuation, casing and line length are good indicators of the type of line, to determine the type of each line, I trained a XGB gradient boosting model using a bag of n-grams character tokens from 1-4 characters with padding. XGBoost is an ensemble decision tree based learning algorithm that has been shown to be very effective at classification problems, and in 2015 it outperformed other methods for most Kaggle competitions. (Chen 2016). While the bag of n-grams may not help with the semantics of the words because the ordering of words are lost, it is very effective here, because of the regular nature of a movie script. Using a selected a subset of seven sequences and hand-coded each line to a category of either description (S), character (C), dialog (D) and other (O), an XGB model could be used on the resulting tokens. I found that also including the token count helped boost the model prediction accuracy. Here is a breakdown of the categories for the training set:



Mean line length for dialog: 22.539

Mean line length for descriptions: 35.374

Mean line length for characters: 8.205

Mean line length for other: 2.155

Analysis and Interpretation

The first step to the script analysis is to identify the characters, dialog and description text. To accomplish that, we've trained a model using a subset of the sequences that were hand-coded into their categories and a XGB gradient boosting model was trained against the data for multi-class classification. The XGB model outperformed other methods on cross validation.

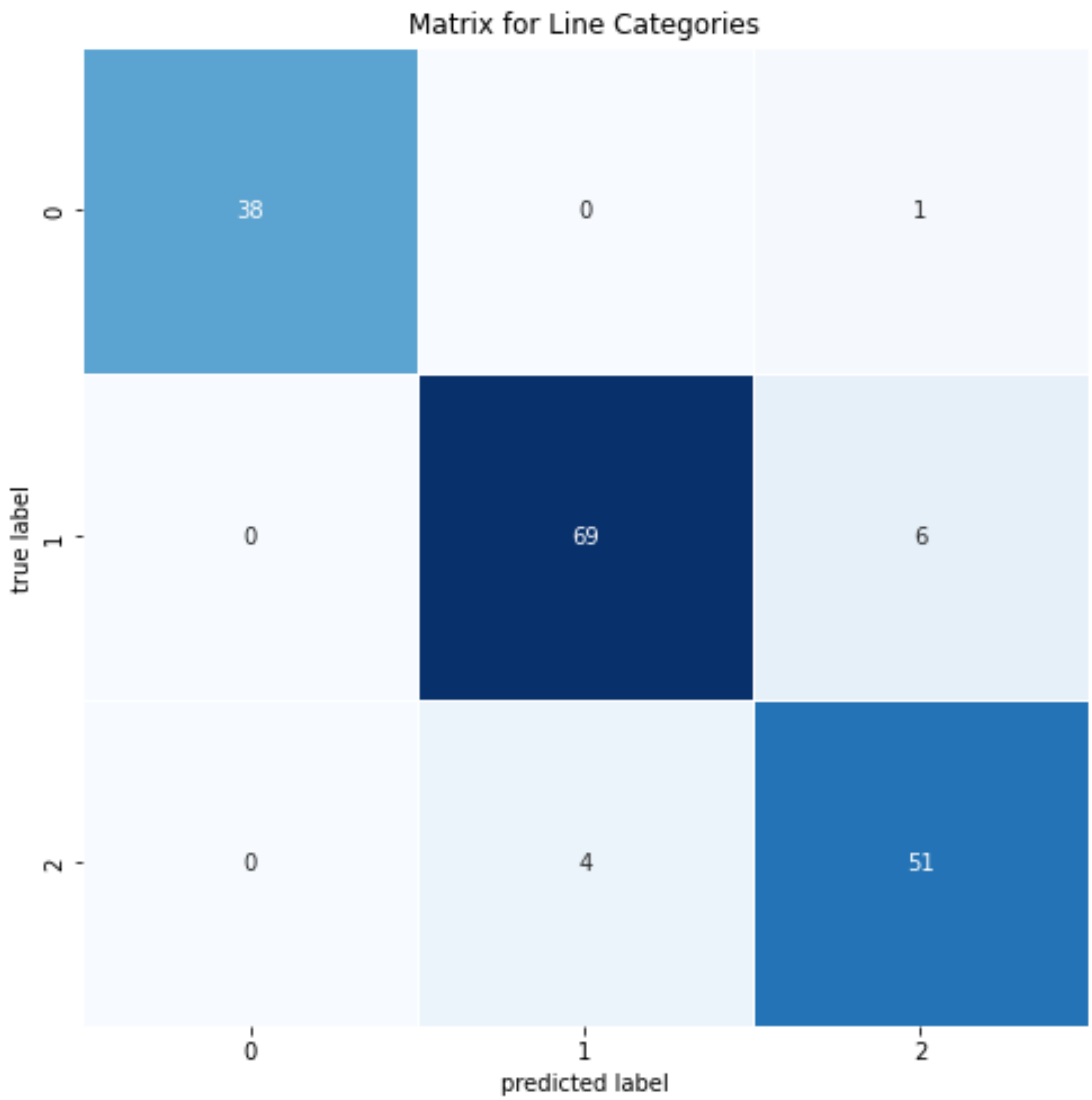
XGBoost cross validation score with a count vectorizer: 0.894

I did a manual validation on a different sequence that was not used for training. For the lines on sequence 2400, this is a report on the accuracy of the predicted line classification:

```
Classification Report
              precision    recall  f1-score   support

           C       1.00      0.97      0.99        39
           D       0.95      0.92      0.93        75
           S       0.88      0.93      0.90        55

    accuracy                           0.93       169
   macro avg       0.94      0.94      0.94       169
weighted avg       0.94      0.93      0.94       169

Accuracy Score: 0.9349
```

So the final accuracy on predicting the correct category for each line was very good at 93.49% accurate.

And the confusion matrix for the validation sequence:

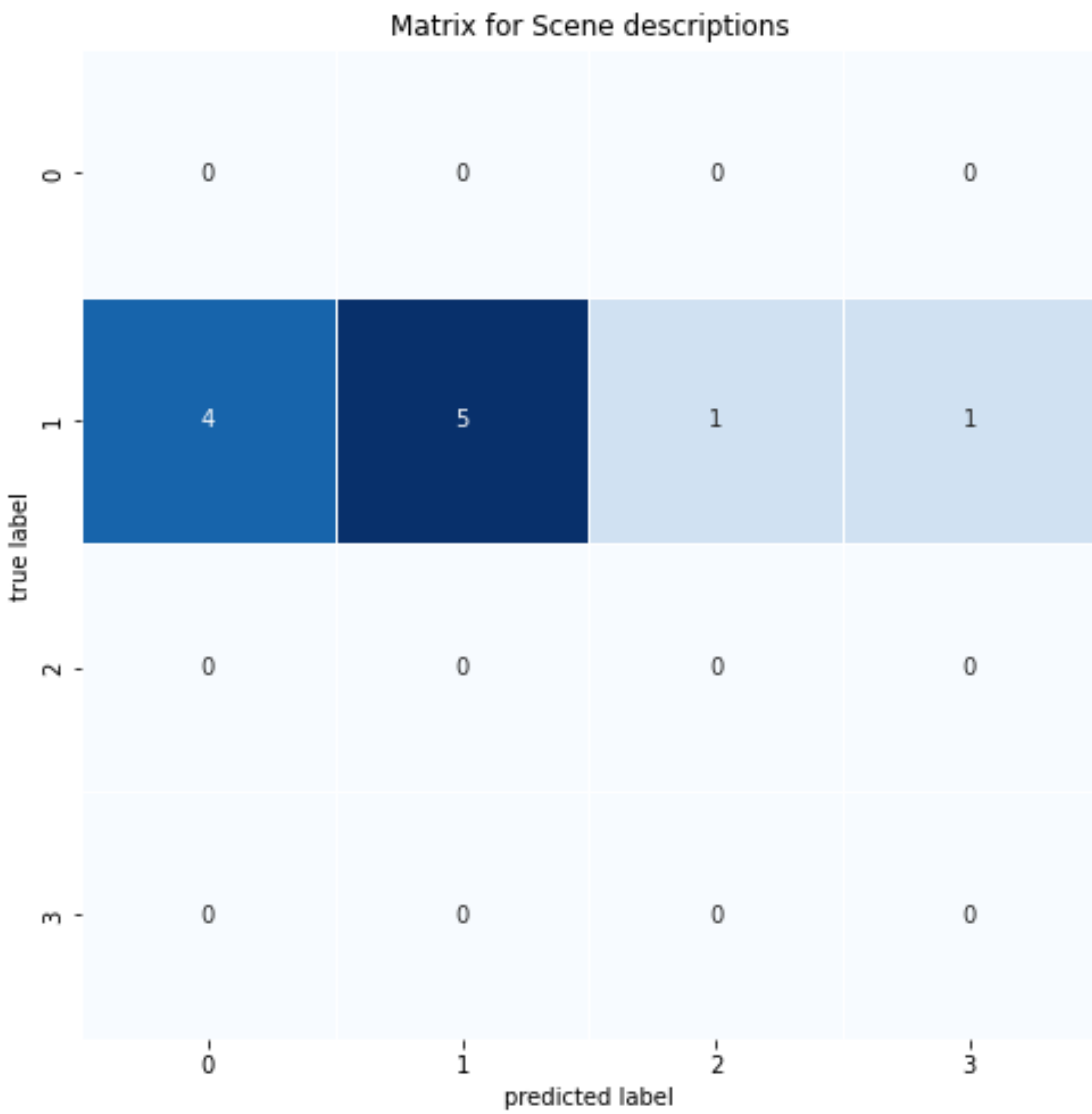


After breaking the the lines into the separate components, I used a bidirectional LSTM network to build categorical model for each of the different components to attempt to predict the proper values for animation complexity. The The long-term, short-term memory (LSTM) network was described by Hochreiter Schmidhuber in 1997 and an effective network for classifying text documents. We are using a bidirectional LSTM model with dropout for regularization. Here is the model summary that was used on all three separate animation components:

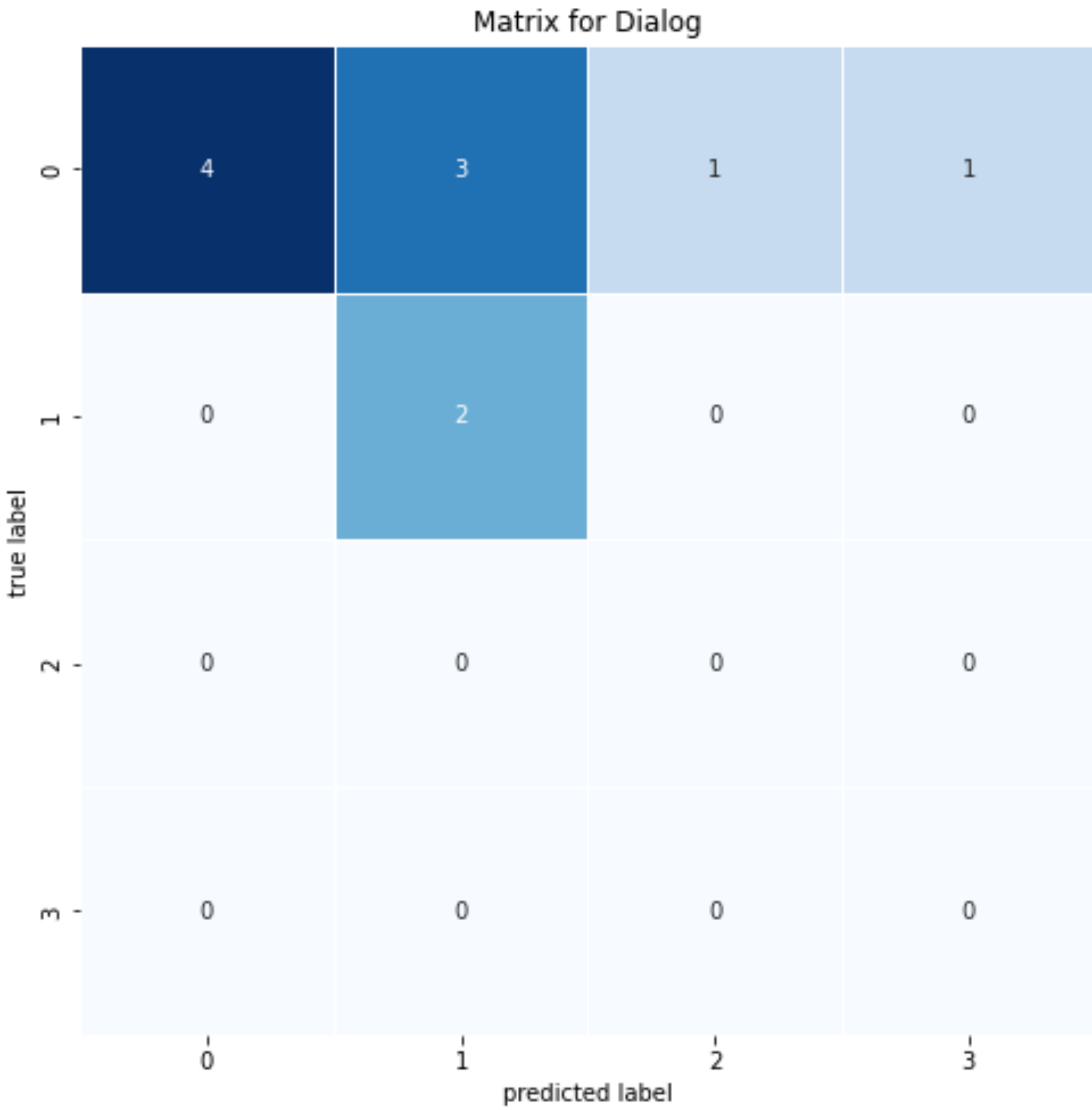
```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, None)]            0

 tf.one_hot_1 (TFOpLambda)   (None, None, 5000)        0

 bidirectional_1 (Bidirectio  (None, 64)               1288448
 nal)

 dropout_1 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 6)                 390

=================================================================
Total params: 1,288,838
Trainable params: 1,288,838
Non-trainable params: 0
_________________________________________________________________
```

This model was then used to predict the categories for the rest of the scenes. The final extracted script was split into 41 sequences and this is the final breakdown of each line type for the scene as output by the model:

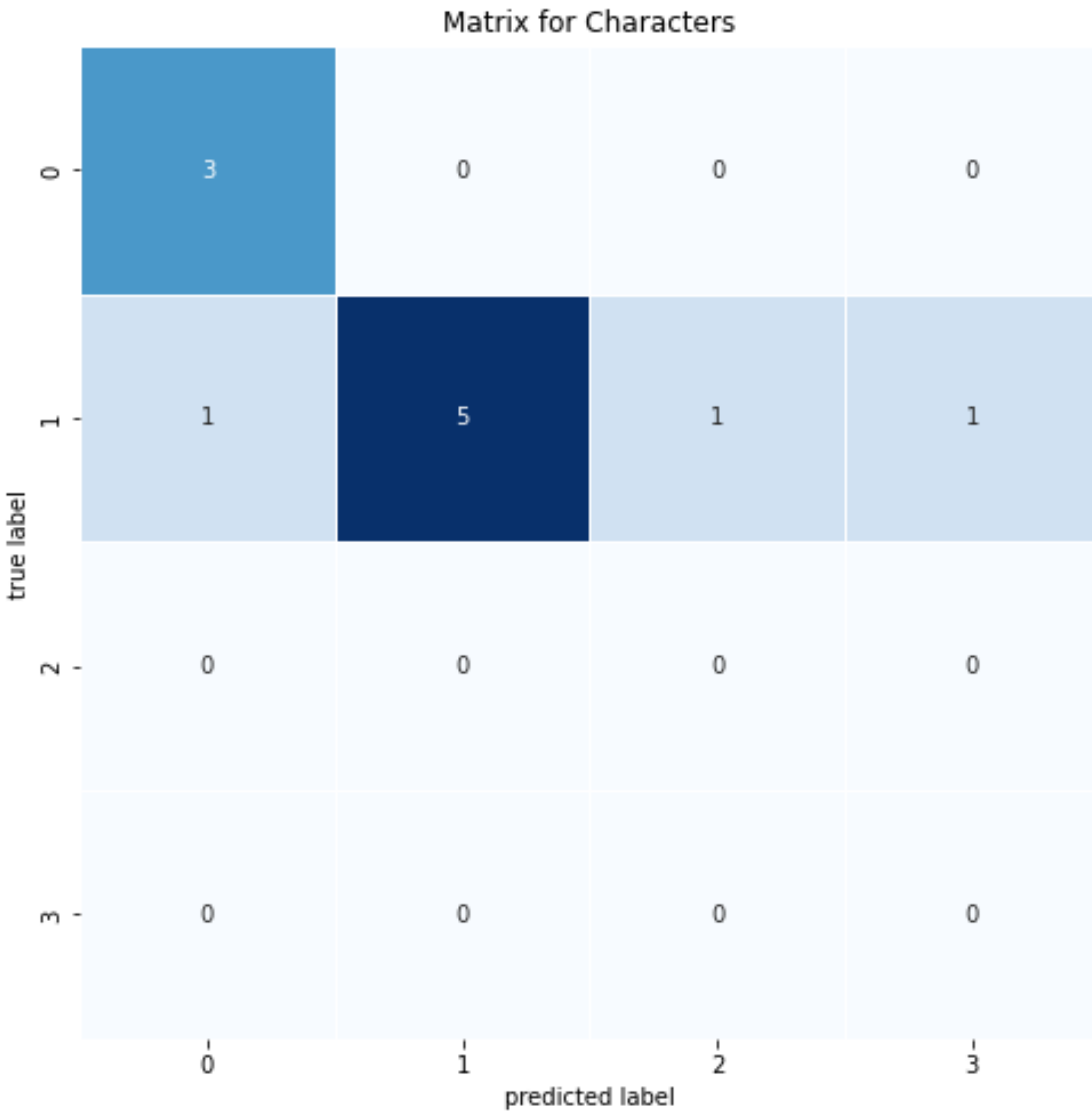
Once the scene were broken out into their components, the LSTM network was trained on each of the scene components. The scene were broken up into 30 scenes for training and a testing set of 11 sequences. For the three models LSTM models, the scene descriptions had an accuracy for 0.4545. Here is the final confusion matrix for the test set using scene descriptions:



The dialog classification had a test accuracy for 0.5455. Here is the final confusion matrix for the test set for the dialog documents:



The character LSTM classification model had a test accuracy for 0.7273. Here is the final confusion matrix for the test set for the character documents:



And the classification report for the character model:

```
Classification Report
              precision    recall  f1-score   support

        CX_2       1.00      0.75      0.86         4
        CX_3       0.62      1.00      0.77         5
        CX_4       0.00      0.00      0.00         1
        CX_5       0.00      0.00      0.00         1

    accuracy                           0.73        11
   macro avg       0.41      0.44      0.41        11
weighted avg       0.65      0.73      0.66        11

Accuracy Score: 0.7273
```

Conclusions

In conclusion, using n-gram tokenization and maintaining the capitalization and punctuation, we we able to create a highly predictive XGB model for categorizing each separate line in a scene into either a character line, scene description or a dialog line. Separating the dialog from the scene descriptions were the biggest source of errors, but there was no good regex or even algorithmic alternatives for determining that correctly.

In using each separate scene component to determine the final animation complexity of the scene, the character names seemed to be the most predictive, which makes sense to me, because the number of different characters and lines is a major part of the expert determination of complexity, but we were unable to obtain much higher accuracy using the other scene elements using bidirectional LSTM models.

Future Directions

Some future directions I would like to take would be to increase the amount of training data using other movie scripts and animation complexity data. We also might be able to improve the accuracy by using all three separate components in one model. Additionally, I would like to experiment more with pre-trained BERT models to try to extract some of the finer details of complexity information. I would also like to blend some numerical data from sentiment analyzers like VADER and NRC and see if that could improve the results.

Appendix/appendices

TABLE 1, complexity matrix from the DreamWorks animation department. The highest complexity given is a 5, meaning most complex, while a 2 is the least complex scene.

Seq	Title	CX
sq400	Tim Time	2
sq500	Not Again	2
sq300	Bedtime	3
sq2150	Tabitha's Room	3
sq2225	Tabitha's Room Part 2	3
sq1950	Setting the Table	2
sq700	Family Reunion	2
sq2100	Family Dinner	5
sq1600	Principal's Office	2
sq1200	Breakfast Scramble	2
sq1300	Pony Express	4
sq2300	Attic Debriefing	2
sq1400	The Center	5
sq1800	Pick Up Line	3
sq1500	School Daze / Dum-Dum Holding Tank	3
sq1550	Bad Boyz	4
sq1700	Recess	2
sq1675	Rehearsal	5+
sq800	Back to Duty	3
sq1100	Back to Bed	2
sq1925	Tim's Mission	3
sq1000	Undercover Brothers	2
sq2400	Ocean's Three	3
sq3100	Girls to the Rescue	2
sq2500	Holiday Pageant	3
sq1900	For Your Eyes Only Pt 1	5
sq1150	Fright Court	2
sq950	Bro Bonding Trip	3
sq1940	For Your Eyes Only Pt 2	2
sq2000	Baby Pep Rally	4
sq2600	Stop the Show	2
sq2750	Caught in the Act	3
sq900	Downsizing	3
sq50	Opening Fantasy	3
sq3300	Server Showdown	3
sq1625	Carol and Tina	2
sq3500	Home for the Holidays	3
sq3400	Show's Over	4
sq3000	Truth is Revealed	3
sq2800	Game Over / Tabitha's Song	2
sq2200	Music is Math	3

TABLE 2, training output for scene description, dialog and character name LSTM models:

Scene Description

```
10/10 [==============================] - 43s 2s/step - loss: 1.3609 - accuracy: 0.2667 - val_loss: 1.3283 - val_accuracy: 0.4545
Epoch 2/200
10/10 [==============================] - 8s 814ms/step - loss: 1.2887 - accuracy: 0.3333 - val_loss: 1.2626 - val_accuracy: 0.4545
Epoch 3/200
10/10 [==============================] - 8s 797ms/step - loss: 1.2082 - accuracy: 0.3667 - val_loss: 1.2017 - val_accuracy: 0.4545
Epoch 4/200
10/10 [==============================] - 8s 783ms/step - loss: 1.1267 - accuracy: 0.4000 - val_loss: 1.1856 - val_accuracy: 0.4545
Epoch 5/200
10/10 [==============================] - 8s 778ms/step - loss: 1.0595 - accuracy: 0.5667 - val_loss: 1.1837 - val_accuracy: 0.4545
Epoch 6/200
10/10 [==============================] - 8s 785ms/step - loss: 0.9850 - accuracy: 0.6667 - val_loss: 1.1762 - val_accuracy: 0.4545
Epoch 7/200
10/10 [==============================] - 8s 778ms/step - loss: 0.9410 - accuracy: 0.6667 - val_loss: 1.1802 - val_accuracy: 0.4545
Epoch 8/200
10/10 [==============================] - 8s 781ms/step - loss: 0.8522 - accuracy: 0.7000 - val_loss: 1.1972 - val_accuracy: 0.3636
Epoch 9/200
10/10 [==============================] - 8s 783ms/step - loss: 0.8200 - accuracy: 0.6333 - val_loss: 1.2449 - val_accuracy: 0.3636
Epoch 10/200
10/10 [==============================] - 8s 784ms/step - loss: 0.9972 - accuracy: 0.6333 - val_loss: 1.2227 - val_accuracy: 0.3636
Epoch 11/200
10/10 [==============================] - 8s 782ms/step - loss: 0.7433 - accuracy: 0.7000 - val_loss: 1.1946 - val_accuracy: 0.3636
2023-03-09 15:10:22.656660: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2023-03-09 15:10:22.745933: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2023-03-09 15:10:22.754218: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
4/4 [==============================] - 12s 2s/step - loss: 1.3283 - accuracy: 0.4545
Test accuracy: 0.455
```

Dialog

```
10/10 [==============================] - 42s 2s/step - loss: 1.3685 - accuracy: 0.2667 - val_loss: 1.3427 - val_accuracy: 0.4545
Epoch 2/200
10/10 [==============================] - 8s 807ms/step - loss: 1.2993 - accuracy: 0.5667 - val_loss: 1.2969 - val_accuracy: 0.4545
Epoch 3/200
```

10/10 [==============================] - 8s 784ms/step - loss: 1.1962 - accuracy: 0.5667 - val_loss: 1.2160 - val_accuracy: 0.4545

Epoch 4/200

10/10 [==============================] - 8s 782ms/step - loss: 1.0965 - accuracy: 0.5667 - val_loss: 1.1947 - val_accuracy: 0.5455

Epoch 5/200

10/10 [==============================] - 8s 781ms/step - loss: 1.0021 - accuracy: 0.6000 - val_loss: 1.1929 - val_accuracy: 0.4545

Epoch 6/200

10/10 [==============================] - 8s 782ms/step - loss: 0.9286 - accuracy: 0.6000 - val_loss: 1.1852 - val_accuracy: 0.4545

Epoch 7/200

10/10 [==============================] - 8s 775ms/step - loss: 0.8119 - accuracy: 0.6000 - val_loss: 1.2197 - val_accuracy: 0.4545

Epoch 8/200

10/10 [==============================] - 8s 777ms/step - loss: 0.7912 - accuracy: 0.6667 - val_loss: 1.2020 - val_accuracy: 0.4545

Epoch 9/200

10/10 [==============================] - 9s 884ms/step - loss: 0.7669 - accuracy: 0.7333 - val_loss: 1.2160 - val_accuracy: 0.4545

Epoch 10/200

10/10 [==============================] - 125s 14s/step - loss: 0.6288 - accuracy: 0.8000 - val_loss: 1.2225 - val_accuracy: 0.4545

Epoch 11/200

10/10 [==============================] - 908s 101s/step - loss: 0.5796 - accuracy: 0.7667 - val_loss: 1.2508 - val_accuracy: 0.3636

Epoch 12/200

10/10 [==============================] - 518s 58s/step - loss: 0.7406 - accuracy: 0.8000 - val_loss: 1.2249 - val_accuracy: 0.4545

Epoch 13/200

10/10 [==============================] - 194s 22s/step - loss: 0.5664 - accuracy: 0.8667 - val_loss: 1.2503 - val_accuracy: 0.4545

Epoch 14/200

10/10 [==============================] - 8s 770ms/step - loss: 0.5093 - accuracy: 0.9000 - val_loss: 1.2386 - val_accuracy: 0.4545

Epoch 15/200

10/10 [==============================] - 8s 768ms/step - loss: 0.4609 - accuracy: 0.9000 - val_loss: 1.3208 - val_accuracy: 0.3636

Epoch 16/200

10/10 [==============================] - 8s 769ms/step - loss: 0.4451 - accuracy: 0.9000 - val_loss: 1.0997 - val_accuracy: 0.5455

2023-03-09 15:42:26.805136: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.

2023-03-09 15:42:26.905661: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.

2023-03-09 15:42:26.916811: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.

4/4 [==============================] - 12s 2s/step - loss: 1.1947 - accuracy: 0.5455

Test accuracy: 0.545

Characters

0/10 [==============================] - 42s 2s/step - loss: 1.3688 - accuracy: 0.4000 - val_loss: 1.3311 - val_accuracy: 0.5455

Epoch 2/200

10/10 [==============================] - 8s 838ms/step - loss: 1.3053 - accuracy: 0.4000 - val_loss: 1.2716 - val_accuracy: 0.5455

Epoch 3/200

10/10 [==============================] - 8s 806ms/step - loss: 1.2255 - accuracy: 0.5000 - val_loss: 1.1790 - val_accuracy: 0.5455

Epoch 4/200

10/10 [==============================] - 8s 795ms/step - loss: 1.1194 - accuracy: 0.5667 - val_loss: 1.1446 - val_accuracy: 0.6364

Epoch 5/200

10/10 [==============================] - 8s 778ms/step - loss: 1.0125 - accuracy: 0.7000 - val_loss: 1.1139 - val_accuracy: 0.6364

Epoch 6/200

10/10 [==============================] - 8s 788ms/step - loss: 0.9351 - accuracy: 0.8000 - val_loss: 1.0709 - val_accuracy: 0.6364

Epoch 7/200

10/10 [==============================] - 8s 792ms/step - loss: 0.7808 - accuracy: 0.8000 - val_loss: 1.0373 - val_accuracy: 0.7273

Epoch 8/200

10/10 [==============================] - 8s 781ms/step - loss: 0.7649 - accuracy: 0.7667 - val_loss: 1.0144 - val_accuracy: 0.6364

Epoch 9/200

10/10 [==============================] - 8s 775ms/step - loss: 0.7365 - accuracy: 0.7333 - val_loss: 1.0415 - val_accuracy: 0.5455

Epoch 10/200

10/10 [==============================] - 8s 783ms/step - loss: 0.6101 - accuracy: 0.7667 - val_loss: 1.0514 - val_accuracy: 0.5455

Epoch 11/200

10/10 [==============================] - 8s 795ms/step - loss: 0.6204 - accuracy: 0.8000 - val_loss: 0.9325 - val_accuracy: 0.7273

Epoch 12/200

10/10 [==============================] - 8s 802ms/step - loss: 0.4997 - accuracy: 0.8000 - val_loss: 0.9193 - val_accuracy: 0.7273

Epoch 13/200

10/10 [==============================] - 8s 794ms/step - loss: 0.5950 - accuracy: 0.7333 - val_loss: 0.8933 - val_accuracy: 0.7273

Epoch 14/200

10/10 [==============================] - 8s 811ms/step - loss: 0.4399 - accuracy: 0.8000 - val_loss: 0.9780 - val_accuracy: 0.6364

Epoch 15/200

10/10 [==============================] - 8s 786ms/step - loss: 0.4143 - accuracy: 0.8333 - val_loss: 1.0221 - val_accuracy: 0.7273

Epoch 16/200

10/10 [==============================] - 8s 775ms/step - loss: 0.4285 - accuracy: 0.8333 - val_loss: 0.9103 - val_accuracy: 0.6364

Epoch 17/200

10/10 [==============================] - 8s 786ms/step - loss: 0.3856 - accuracy: 0.8333 - val_loss: 1.2914 - val_accuracy: 0.5455

2023-03-09 15:45:39.097861: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.

2023-03-09 15:45:39.185332: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.

2023-03-09 15:45:39.193543: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.

4/4 [==============================] - 12s 2s/step - loss: 1.0373 - accuracy: 0.7273

Test accuracy: 0.727

References

Frangidis, Paschalis, Konstantinos Georgiou, and Stefanos Papadopoulos. “Sentiment Analysis on Movie Scripts and Reviews.” SpringerLink. Springer International Publishing, May 29, 2020. https://link.springer.com/chapter/10.1007/978-3-030-49161-1_36.

Hutto, C., and Eric Gilbert. 2014. “VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text”. Proceedings of the International AAAI Conference on Web and Social Media 8 (1):216-25. https://doi.org/10.1609/icwsm.v8i1.14550.

McCullers, Michael. “The Boss Baby: Family Business: Screenplay - Universal Pictures - FYC 2022.” The Boss Baby: Family Business | Screenplay - Universal Pictures - FYC 2022. Accessed January 28, 2023. https://awards.universalpictures.com/the-boss-baby-family-business/screenplay.

Lee, Seong-Ho, Hye-Yeon Yu, and Yun-Gyung Cheong. "Analyzing movie scripts as unstructured text." In 2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService), pp. 249-254. IEEE, 2017.

Mourchid, Y., Renoust, B., Cherifi, H., El Hassouni, M. (2019). Multilayer Network Model of Movie Script. In: Aiello, L., Cherifi, C., Cherifi, H., Lambiotte, R., Lió, P., Rocha, L. (eds) Complex Networks and Their Applications VII. COMPLEX NETWORKS 2018. Studies in Computational Intelligence, vol 812. Springer, Cham. https://doi.org/10.1007/978-3-030-05411-3_62

Le, Quoc V., and Tomas Mikolov. “Distributed Representations of Sentences and Documents.” arXiv.org, May 22, 2014. https://arxiv.org/abs/1405.4053.

Pere, Christophe. “Model Selection in Text Classification.” Medium. Towards Data Science, November 24, 2020. https://towardsdatascience.com/model-selection-in-text-classification-ac13eedf6146.

Chen, Tianqi, and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System.” arXiv.org, June 10, 2016. https://arxiv.org/abs/1603.02754.

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long Short-Term Memory." MIT Press. MIT Press, November 15, 1997. https://direct.mit.edu/neco/article-abstract/9/8/1735/6109/Long-Short-Term-Memory?redirectedFrom=fulltext.

Rogge, Niels. "Transformers-Tutorials/fine_tuning_bert_(and_friends)_for_multi_label_text_classification.Ipynb at Master · Nielsrogge/Transformers-Tutorials." Fine-tuning BERT (and friends) for multi-label text classification. GitHub, November 17, 2021. https://github.com/NielsRogge/Transformers-Tutorials/blob/master/BERT/Fine_tuning_BERT_(and_friends)_for_multi_label_text_classification.ipynb.