Youtube Video Popularity Prediction and Popular Titles Generator

Zihao Zhou, Xuejun Zhang, Hao Liu, Feiyang Jiang

Abstract

Youtube is the world-famous video sharing website platform that allows people to upload all kinds of videos. Youtubers are a group of people who get paid by Youtube if their videos have reached a certain number of views or if the total length of watched videos has passed a threshold. The more views the videos have, the more the Youtubers get paid. They are also supported by companies by advertising. This project intends to use the labeled Youtube Video statistics dataset to predict the popularity scores of given videos by deep learning techniques. This project will also go one step further to analyze the titles of popular videos and apply machine learning algorithms, especially deep learning models, to develop a title generator which will output titles for given words with desired word count. Textual data in the titles are used for both the predictive model and the generator in this project.

Introduction

As youtube becomes the word-famous video sharing platform, people can watch videos created by people around the world. Big traffic drives advertisement revenue and product sales. Youtube channels and Youtubers have become a new type of job in recent decades. Youtubers or channels make revenue from advertising from sponsored companies or Youtube itself. Videos with significant traffic will attract more companies and will generate more revenue. Therefore, this project will analyze the textual titles and make recommendations to the Youtubers when titling their videos.

There is a wide variety of videos with different topics, so there is a big advertising market on Youtube. It is important for the Youtubers to catch people's attention by having appealing video titles with trending words or interesting tags. By doing so, their videos can attract more traffic and therefore they can earn more advertisement revenue from the sponsoring companies. In this project, we have data with given popularity scores with high being popular and low being non-popular. This project intends to apply machine learning algorithms to select the best features to predict the popularity of the Youtube videos from a pool of audience feedback features and Video attributes, so that we can further make recommendations to Youtubers using the title generator.

This project will apply RNN, LSTM, Heteroscedastic uncertainty to fit the training Youtube titles and predict popularity scores for given video titles. In this way, Youtubers can predict the popularity of their video titles before uploading. The team hopes this project will help Youtubers make decisions when making titles to their videos. To go one step further, this project will also help Youtubers come up with title ideas by given topics or key words from the video contents.

Related Works

GloVe

Processing text data requires preprocessing by embedding words into vectors. Global Vectors for Word Representation (GloVe) is one of unsupervised learning algorithms to represent words as vectors. The GloVe training is based on aggregated global word-word co-occurrence statistics from a corpus [1], but noticing here that the corpus could be huge as 13 million words so in this project, a GloVe pretrained dictionary is implemented to embed the words from YouTube videos tags, titles, and description.

Huber

As the whole project takes mainly three natural language processed (NLP) factors, it requires an algorithm with more tolerance to the outliers for smooth loss. Thus, the huber loss in terms of the function below is implemented in this project with target t and prediction p.

$$Huber\ loss(t,p) = \begin{cases} \frac{1}{2}(t-p)^2, & when\ |t-p| \leq\ \partial \\ \partial|t-p| - \frac{\partial^2}{2}, & otherwise \end{cases}$$

Figure 1: Huber Loss Formula [2]

RNN

There are two main neural networks used in this project. The first one is recurrent neural networks (RNN). RNN is different from basic feed-forward neural networks because it needs a loop to be factored in the network's model as a memory state of the neurons. Also its activation outputs from neurons propagate in both directions (from inputs to outputs and from outputs to inputs) as shown in Figure 1. upper part. A represents a full RNN cell that takes the current words as input sequence X_i, and output the current hidden state H_i, then passing this to the next RNN cell for the input sequence. [3]

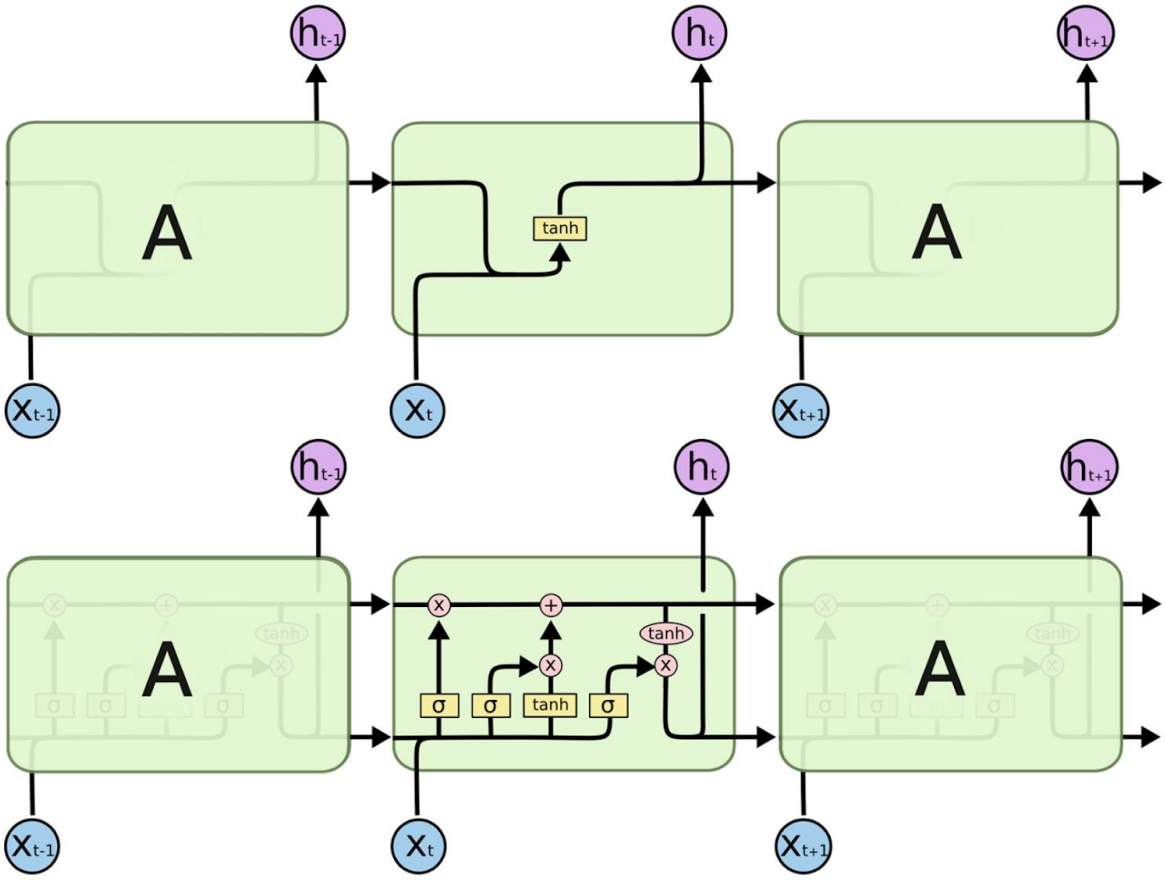

Figure 2: RNN and LSTM model schema [3]

LSTM

Another neural network used in this project is the Long Short Term Memory. LSTM model has been revealed to solve the vanishing gradient problem caused by RNN. As shown in the Figure 2 lower part, while RNN cell only has one internal layer but LSTM has three internal layer acting on the current state h_{t-1} and input X_t. The first layer is the "forget gate", with h_{t-1} and X_t as input and go through the sigmoid layer σ. The second layer is the "update gate" that passes the previous layer to go through the tanh activation layer for performing the element wise multiplication between the results. The third layer is the "output gate" that controls what information to pass on in the next layer.[3]

Homoscedastic and heteroscedastic uncertainty

When it comes to predictive modeling, aleatoric uncertainty rises when, for data in the training set, data points with very similar feature vectors (x) have targets (y) with substantially different values. No matter how good the model is at fitting the average trend, it won't be able to fit every datapoint perfectly when aleatoric uncertainty is present. If this uncertainty stays constant for different x-values, it is called homoscedastic uncertainty; if this uncertainty changes for different x-values in the feature space , it is called heteroscedastic uncertainty.[4]

Dataset

The dataset is from kaggle.com, which was collected using Youtube API. This dataset includes several months (and counting) of data on daily trending YouTube videos. Data is included for Russia, Mexico, South Korea, Japan and India respectively over the same time period. Data includes the video title, channel title, publish time, tags, views, likes and dislikes, description, and comment count. The data also includes a category_id field, which varies between regions.

Methods

Data Preprocessing

The original dataset used for this project is the trending YouTube video statistics from Kaggle website [5]. The whole original dataset is about 500MB and it contains many countries trending YouTube videos features data but only English speaking countries US, UK and Canada are extracted. Any duplicated video data, and any video descriptive features containing non-English content are dropped. Text data are preprocessed with elimination of symbols and special character. After the data cleaning, a 23872 rows * 17 features dataframe is kept. A column representing the popularity of the YouTube video is added. The popularity score is calculated as shown in Figure 3 which is referenced from Stanford researchers. For the first part of the project, video titles, tags and descriptions are used as features to predict the distribution of the popularity score. The data is randomly split into 20000 training data and 3872 test data. The titles, tags and descriptions are tokenized and transformed into numeric sequences, the sequences are then padded to the same length. The second part of the project used title as the feature to build the title generator.

$$Popularity\ Score\ =\ \frac{\#\ of\ comments}{\#\ of\ views}\times(\#\ of\ likes\ -\ 1.5*dislikes)$$

Figure 4: Popularity Score Formula[6]

Model construction

Youtube video popularity score prediction

A standard neural network can only handle homoscedastic uncertainty by introducing bias, which is independent with x-values. In order to deal with heteroscedastic uncertainty, our general approach is to train a model that splits the neural network into two streams, mean and standard deviation, and use them to parameterize a distribution.

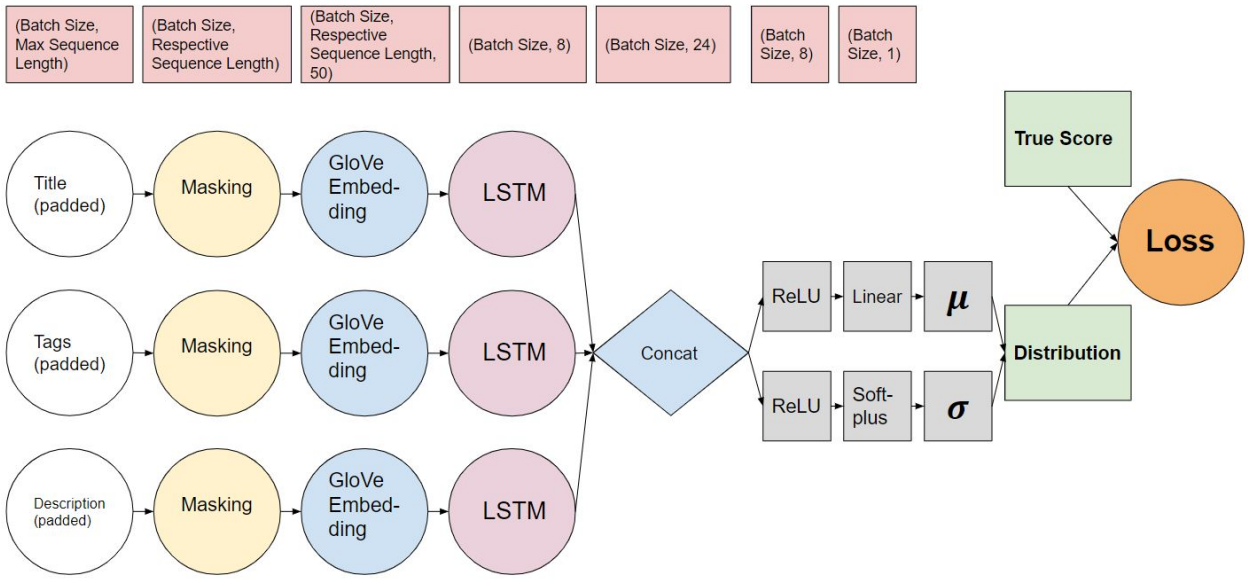

Figure 5: Heteroscedastic Model

As the graph shows, the neural network takes 3 sequence inputs, title, tags and description, and each of them passes through a masking layer to mask zero elements that are used to pad the sequence to the same length. Then, 3 streams of sequences with different length are respectively embedded into 50 dimensional vectors using pre-trained GloVe. After that, the sequences pass through a LSTM layer that outputs a 8 dimensional vector for each sequence and all 3 streams are concatenated into one. Finally, the concat layer is split into two streams for mean and standard deviation, each of which is a 2-layer feed-forward network. Linear function is

used as the output function for mean since the score could be negative when likes are less than 1.5*dislikes; softplus function is used as the output function for standard deviation, since it should be greater than 0. Assuming that the output mean and standard deviation parameterize either a Gaussian or a Laplace (double exponential) distribution, we can then define the loss as the negative log likelihood of the target under the distribution.

$$L(W) = -\frac{1}{N}\sum_k \log p(y_k|\mu = f^W(x_k), \sigma = g^W(x_k))$$

Figure 6: Negative Log Likelihood Formula

In order to make the heteroscedastic model trains easier, we first build a homoscedastic model that is exactly the same as the heteroscedastic model, except that it doesn't have the streams for standard deviation and therefore the loss is also different. In the homoscedastic model, we use huber as our loss. Compared to regular RMSE, it is more tolerant to the outliers with smooth loss.

Then, we use the weights for the LSTM and feed-forward layers of the homoscedastic model as the initial points for the weights for the same layers of the heteroscedastic model. And train for both Gaussian and Laplace distribution assumptions. Below shows the loss for both models.

$$L(W) = \frac{1}{2N}\sum_k \left(\frac{1}{\sigma(x_k; W)^2}|y_k - \mu(x_k; W)|^2 + \log \sigma(x_k; W)^2 + \log 2\pi\right)$$

Figure 7: Loss for Heteroscedastic Model with Gaussian Assumption

$$L(W) = \frac{1}{N}\sum_k \left(\frac{1}{\lambda(x_k; W)}|y_k - \mu(x_k; W)| + \log 2\lambda(x_k; W)\right)$$

Figure 8: Loss for Heteroscedastic Model with Laplace (Double Exponential) Assumption

Title Generator

For the title generator, we mainly used the LSTMs to accomplish our goal. We trained our model to learn the likelihood of occurrence of a word based on the previous sequence of words used in the text datasets. We tried to apply the LSTM models at character level, n-gram level, sentence level and paragraph level to figure out the model's best performance.

We used our cleaned youtube dataset and extracted the title part as the model input. We firstly applied the build-in function from KERAS library to tokenize our input text then padded those sequences and made their lengths equal.

Unlike RNNs, LSTMs have an additional state called 'cell state' through which the network makes adjustments in the information flow. The advantage of this state is that the model can remember or forget the leanings more selectively. We have added a total three layers in the model.

1. Input Layer : Takes the sequence of words as input
2. LSTM Layer : Computes the output using LSTM units. I have added 100 units in the layer, but this number can be fine tuned later.
3. Dropout Layer : A regularisation layer which randomly turns-off the activations of some neurons in the LSTM layer. It helps in preventing overfitting. (Optional Layer)
4. Output Layer : Computes the probability of the best possible next word as output

Finally, with the trained model, we can use our text data to predict the popular word regarding the topic then append those words together to a popular title

Output & Visualization

For the main part of this project which is to predict distribution of YouTube videos popularity score, the model output is the mean and standard deviation of popularity scores of YouTube videos. The metric for the model is the loss value. The Matplotlib package in Python is used to visualize several input factors results. For the second small part of this project which is to build the YouTube videos titles generator, the model output is the generated YouTube titles in string type.

Results

Figure X1 and X2 shows the predicted Gaussian and Laplace distribution for the popularity score of one test data, where the x-axis is the popularity score, y-axis is the negative log likelihood. The red line stands for the true popularity score and the predicted distribution of popularity score is in blue. The deep blue area is the 95% confidence interval of the predicted popularity score distribution. If the red line falls into the deep blue area, it is considered as an

acceptable prediction. As the intersection y-value of the red line and blue area is higher, the better the prediction for that datapoint. Same applies for Figure X3 and Figure X4, which are the predicted Gaussian and Laplace distribution for the popularity score of another test data.

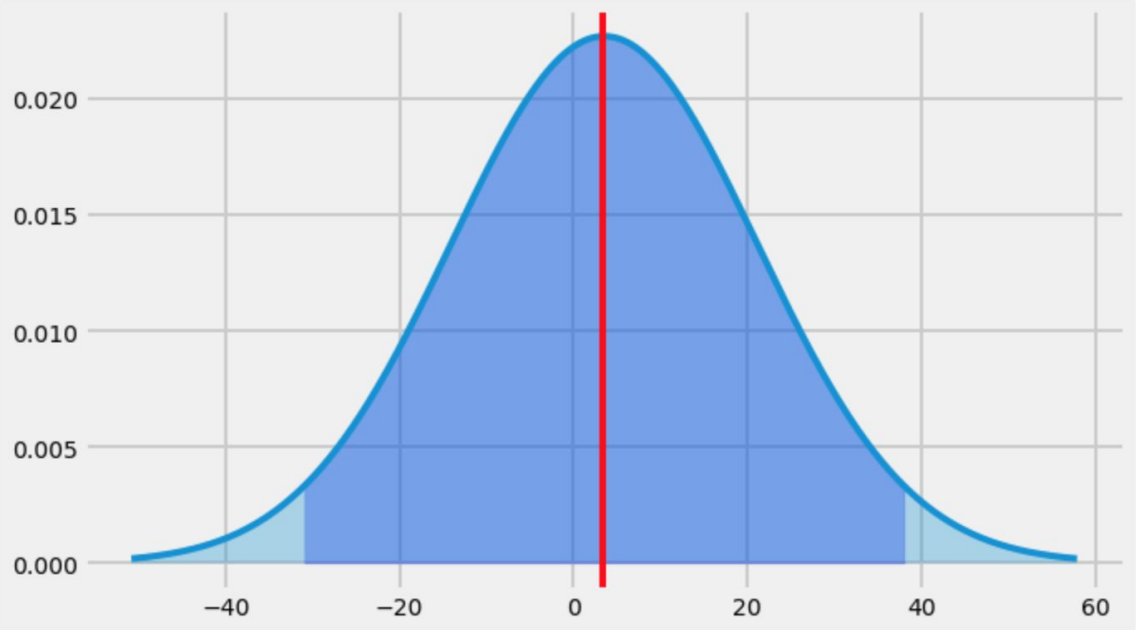

Figure X1: True Popularity Score and Predicted Popularity Score Distribution of test point α under Gaussian Distribution

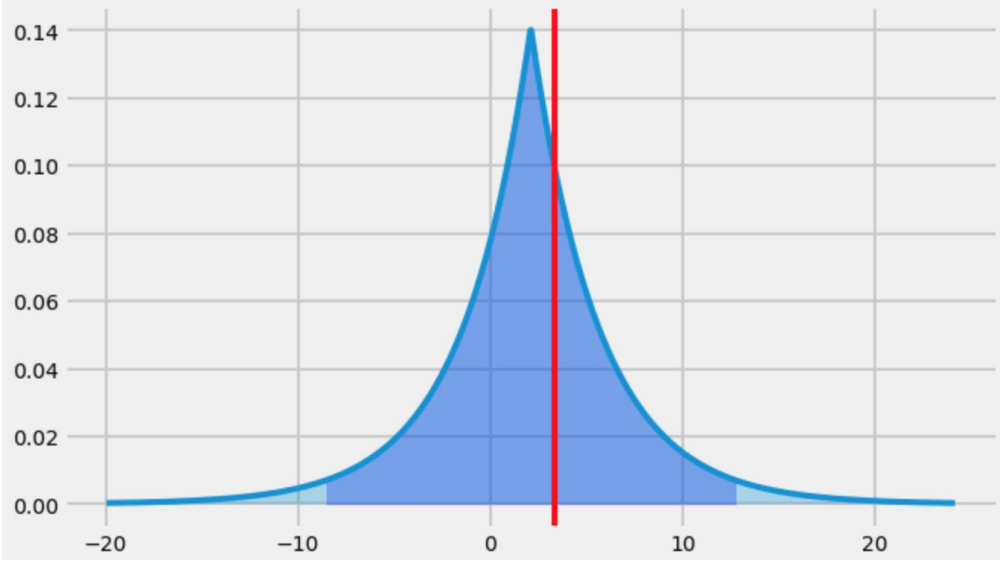

Figure X2: True Popularity Score and Predicted Popularity Score Distribution of test point α under Laplace Distribution

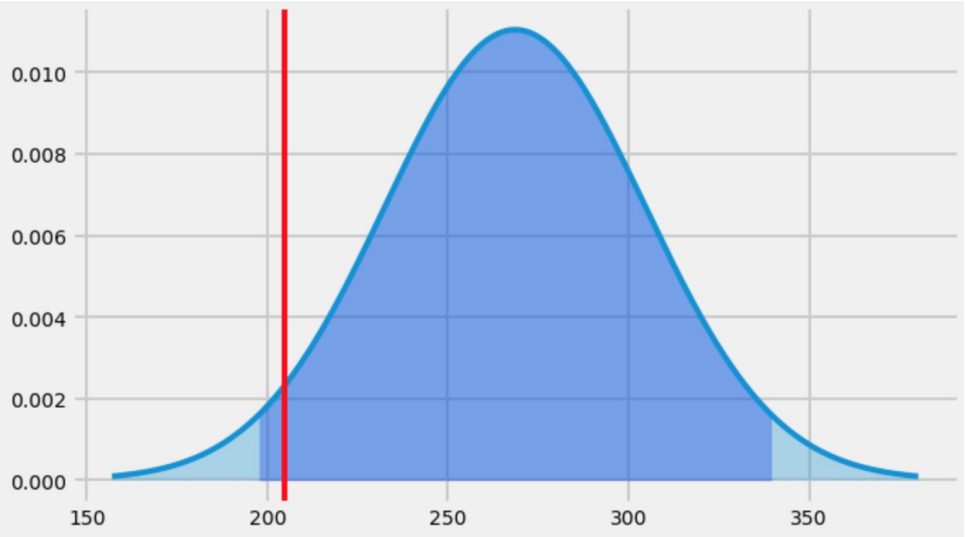

Figure X3: True Popularity Score and Predicted Popularity Score Distribution of test point β under Gaussian Distribution

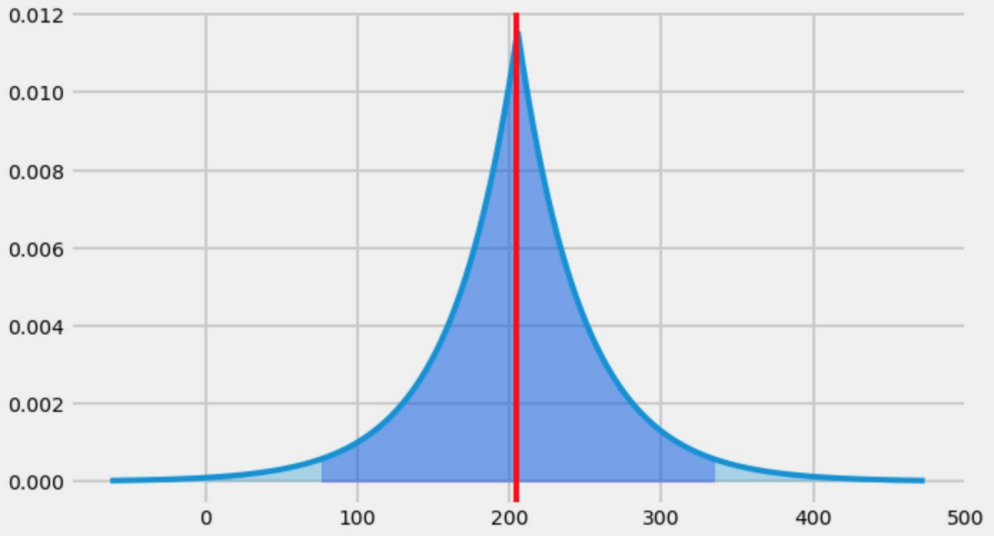

Figure X4: True Popularity Score and Predicted Popularity Score Distribution of test point β under Laplace Distribution

Table 1: Model Metrics Comparison

Model	**Training Loss**	**Test Loss**
Gaussian Assumption	445.7725	742.5018
Laplace Assumption	5.8449	6.6085

```
print (generate_text("drake", 5, model, max_sequence_len))
print (generate_text("Leslie", 8, model, max_sequence_len))
print (generate_text("Australia", 4, model, max_sequence_len))
print (generate_text("Trump", 4, model, max_sequence_len))
print (generate_text("people", 5, model, max_sequence_len))
print (generate_text("nintendo", 6, model, max_sequence_len))
print (generate_text("covid", 3, model, max_sequence_len))
print (generate_text("brunch", 3, model, max_sequence_len))
print (generate_text("noodles", 3, model, max_sequence_len))
```

```
Drake God Plan Official Audio Ft
Leslie Jones The The National Anthem Character 2018 Full
Australia 2018 Golden State Warriors
Trump Is President Donald Trump
People Who Had This Mexico Vs
Nintendo Labo Hands Official Video Ft By
Covid Kim Kardashian West
Brunch The X Anthony
Noodles Open Source Winter
```

Figure 6: YouTube Videos Title Generator Output Example

Conclusion and Discussion

On predicting the Youtube video popularity score, we split the neural network into two streams, mean and standard deviation, and use them to parameterize either a Gaussian or a Laplace (double exponential) distribution. This helps us to quantify the heteroscedastic uncertainty for each data.

At first, the heteroscedastic model drops into a local optimum after a few epochs that outputs a large standard deviation and a mean far from the true value for each data. To address this issue, we built a homoscedastic model that doesn't have the streams for standard deviation but stays exactly the same as the heteroscedastic model for the other parts, and use the weights for the LSTM and feed-forward layers of the homoscedastic model as the initial points for the weights for the same layers of the heteroscedastic model.

Compared to the model with Gaussian distribution, the model with Laplace distribution assumption has a much smaller loss (negative log likelihood) on both training and test data, suggesting the latter is a better model, probably because the Laplace distribution has a heavy tail and therefore more tolerant to the outliers. Moreover, 87.59% of the test data has their true value in the 95% confidence interval of the predicted Laplace distribution, while only 34.91% true value of the test data fall into the 95% confidence interval of the predicted Gasssian distribution, which also shows that the model with Laplace distribution assumption does a better job.

On building the YouTube title generator, the output results do not follow the normal words sequence and grammar but there are still signs showing the relatedness between the words. For example, when the input word is Drake, the output YouTube title is "Drake God Plan Official Audio Ft" The word "Drake" is definitely related to the phrase "God Plan" because it was the rapper's popular record released in 2018 with over 4.3 million stream on Spotify. Hence, it is likely that this YouTube title will attract many viewers. However, the coherence and readability need future work to improve.

Future Work

The next steps for this project would be to explore more models to predict the distribution of the predicted popularity scores and discover metrics to evaluate the performance of the title generator.

The team implemented the heteroscedastic model under both the Gaussian and Laplace Distribution to select the model which yields the best result for the predicted popularity scores. In the future, more deep learning models will be trained and tested to make predictions on the scores to see if there is going to be a lower training loss and testing loss value.

For the title generator, the next steps would be to define key metrics to evaluate the confidence of the titles being generated and give more freedom to the user to customize the recommendation output. For example, for every trial of recommendation, the Youtuber will also have the multiple titles to choose from and with the confidence level respectively.

References

[1] Selivanov, D. (2020, April 18). GloVe Word Embeddings. Retrieved December 16, 2020, from http://text2vec.org/glove.html

[2] Chris. (2019, December 21). About loss and loss functions. Retrieved December 16, 2020, from https://www.machinecurve.com/index.php/2019/10/04/about-loss-and-loss-functions/

[3] Hoffman, G. (2018, January 11). Introduction to LSTMs with TensorFlow. Retrieved December 16, 2020, from https://www.oreilly.com/content/introduction-to-lstms-with-tensorflow/

[4] Reasonable Doubt: Get Onto the Top 35 MNIST Leaderboard by Quantifying Aleatoric Uncertainty. (n.d.). Retrieved December 16, 2020, from https://www.capitalone.com/tech/machine-learning/get-onto-the-top-35-mnist-leaderboard-by-quantifying-aleatoric-uncertainty/

[5] J, M. (2019, June 03). Trending YouTube Video Statistics. Retrieved December 16, 2020, from https://www.kaggle.com/datasnaek/youtube-new

[6] Li, Y., Eng, K., & Zhang, L. (n.d.). YouTube Videos Prediction: Will this video be popular? Retrieved December 05, 2020, from http://cs229.stanford.edu/proj2019aut/data/assignment_308832_raw/26647615.pdf