Towards a Focused YouTube Experience

Lakshya
Department of Electrical Engineering
Indian Institute of Technology Ropar
2017eeb1149@iitrpr.ac.in

***Abstract*—Recommendation algorithms for various media content platforms like YouTube are surprisingly good at delivering content that the user would like to watch. This, however, is not the sole purpose of these algorithms. They are designed to keep the user on the platform for as long as possible. This results in numerous unwanted side effects for the user, specifically, leading the user down a proverbial Rabbit Hole through recommendations that are increasingly divergent and irrelevant. This greatly degrades the efficacy of YouTube as a platform to practice productivity. In this project, I propose a machine learning method that can be key in developing solutions that mitigate this effect and enable users to experience a distraction-free and mindful YouTube media consumption.**

***Index Terms*—BERT, Deep Neural Networks, YouTube**

I. Introduction

In my last project [6], I provided quantitative evidence for the existence of the "Rabbit Hole" phenomenon. In this project, I developed a pipeline to utilize a popular general purpose YouTube dataset with a custom web-scrapper to generate my own dataset of YouTube video features. These features will be specific to our task of identifying a distraction indicating shift in a user's watch activity. For this purpose, I also develop a Neural Network based model that appends to the popular BERT[3] with pre-trained weights to classify YouTube videos into generic classes.

II. Data source

Since this project required multiple features for individual youtube videos, I decided to develop my own dataset for this project. The Youtube 8M dataset [8] was used to extract pseudo `videoId` and corresponding `labels`. These pseudo ids were not the real youtube ids for anonymity. The Youtube 8M dataset provided labels for 3862 classes for each video. A total of 11000 pseudo video ids and labels were extracted from this dataset. These pseudo video ids were then processed in accordance with the provided script that was automated to get corresponding real youtube `videoId`. These real `videoId` were then used as input to a custom-built youtube web-scraper that extracted the `title`, `description`, `keywords`, `likeCount`, `dislikeCount`, `viewCount`, and video `length` for the video.

III. Exploratory Data Analysis

The range of 3862 classes was not suitable for this project for multiple reasons. Firstly, most of the videos were classified in few of the categories. Second, I required only few representative classes and thus, such a large number of them would lead to increases computational complexity. I segregated these classes into 25 verticals as described by the authors in [8] using a conversion list provided with the dataset.

$$verticals = \{"Games", "ArtsEntertainment", "Sports", ...\}$$

The features `title`, `description`, `keywords` are in string format, and thus need to be processed. The raw data is made up of multiple languages and decorative unicode characters.

A. String preprocessing

1) Strip escape characters such as `\n \t \r`
2) Remove hyperlinks
3) Compress white spaces
4) Reduce to basic-latin unicode
5) Strip non-punctuation special characters

B. Language Filter

I detect and discard any video entries which consist of non-English text. This is done to increase the quality of the dataset and simplify it for the upcoming BERT model. This is done using the `polyglot` library.

The features `title`, `description`, `keywords` are then concatenated into a single string `keywords`.

C. Scale numerical values

The values of `likeCount`, `dislikeCount`, `viewCount`, and `length` can vary greatly in magnitude. these values are scaled by a scaled down logarithmic function $g(x)$ that maps values from $[0, 10^{10})$ to approximately $[0, 1)$

$$g(x) = 0.1 \log_{10}(x + 1)$$

D. Vectorize Strings

BERT [3] is a state of the art NLP model that is built upon the transformer [2] architecture and can be used for various tasks such as masked language modelling, token classification and sequence classification.

1) Tokenization for BERT: BERT uses the wordpiece fig.1 embeddings [1] which separates words into segmented tokens, thus reducing the size of vocabulary considerably - enabling the realization of powerful models like BERT [3].

The BERT model also takes into consideration the positional embeddings fig.2. This makes BERT the perfect module for multiple NLP tasks by appending custom heads to its output.

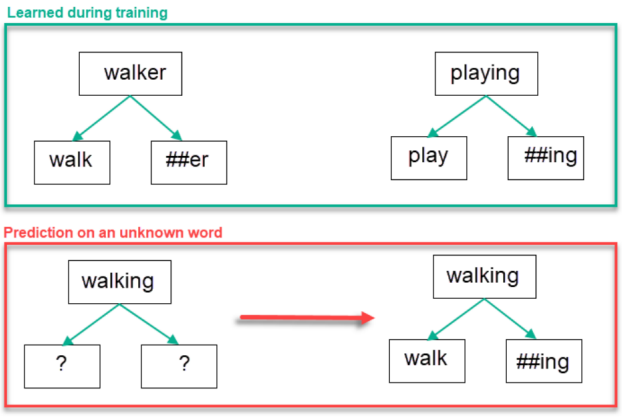

Fig. 1: Wordpiece tokenization of an unseen word during inference. [4]

2) BERT Output Format: The headless BERT model outputs a series of hidden states corresponding to each input token fig.3, including the special token `[CLS]`. The output corresponding to this token is tuned for sequence classification problems. I extract only this output for my purpose. The output vector has the dimension of 768

In combination with the numerical values, the complete processed vector for one video has the dimension 772 fig.4

IV. Architecture and Training

A. DistilBERT

I use the pre-trained light version of BERT called DistilBERT [5] for this project. The weights are initialized from the generalized BERT tasks `'distilbert-base-uncased'`.

B. Classification Head

A simple 2-layer deep neural network is used as the classification head for the DistilBERT model. Regularization techniques were important in the performance of this classifier due to the limited data points at my disposal and high number of features. This scenarios is very prone to overfitting. Multiple configurations of dropout layers, weight regularization factors, learning rates were tested.

C. Over-Sampling

Initial results revealed very poor performance on the test-set. It was found upon deeper inspection that this was due to severe imbalance in the distribution of the dataset. fig.5

To mitigate this, I decided to over-sample the dataset in the following manner:

Let S_i be set of data points with class index i.

Let $M = \max_i |S_i|$.

If, for a hyper parameter k and class i, $|S_i| < \lfloor \frac{M}{k} \rfloor$, then occurrence of S_i is multiplied by $\lfloor \frac{M}{k \times |S_i|} \rfloor$.

D. Architecture and Hyperparameters

After multiple iterations which gave reasonably close results in accuracy, I finally settled on the following architecture fig.6.

- The dropout rate in both dropout layers is 0.4.
- The optimizer is Adam with $lr = 10^{-3}$.
- The intermediate Dense layer has `relu` activation with L1 regularization factor$= 5 \times 10^{-}4$ and L2 regularization factor$= 5 \times 10^{-4}$
- The final layer has `softmax` activation
- batch size used during training was 1024 and epochs were 1500

The tests revealed that a value of $k = 9$ gives the best result for my dataset.

k	best accuracy in %
1	61.637
2	64.008
3	63.793
4	65.732
5	65.732
6	65.732
7	66.379
8	66.379
9	67.672
10	66.163

V. Results

My model achieved an accuracy of 67.67% on the test set and an accuracy of 73.86% on the training set fig.7 for a classification task with 25 classes and a severely imbalanced homegrown dataset. It is important to keep in mind that this classification is completely based on numerical factors such as video length, like count, etc. and string data scraped from the description, title and keywords of the videos. These sources rarely conform to rules of languages and are difficult to extract context from.

VI. Conclusion

In this project, I took another step towards enhancing the productive user experience of the YouTube platform. This project saw development of a complete dataset generation pipeline that is capable of harnessing the 8 million+ videos from the YouTube 8M dataset [8] in tandem with the scraping tools that enhance these resources. In addition, I explored the task of classifying videos with this data into broad predefined classes that can be directly used to make a rough estimate of the productivity of a video. The architecture developed on top of the DistilBERT model [5] can be altered and used to assess the trail of a user's watch history to identify points of distraction. This development has been inspired by my

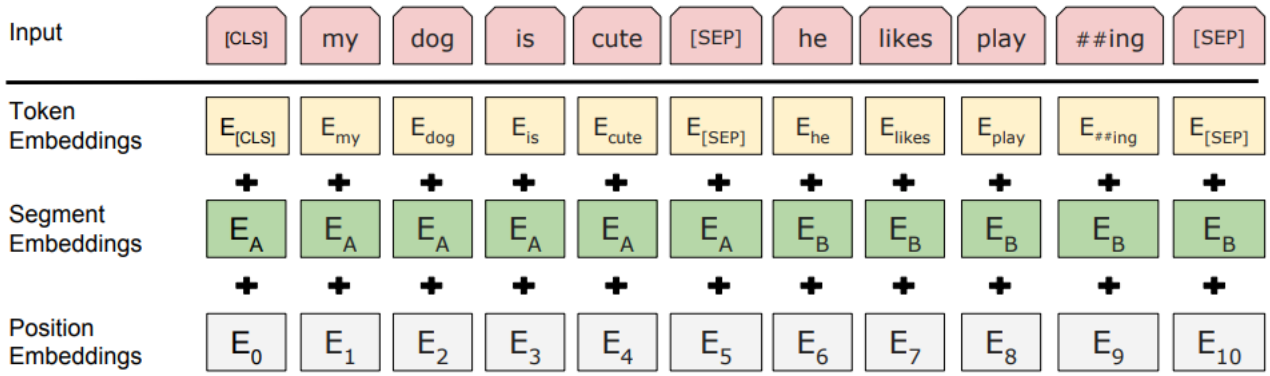

Fig. 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings. [3]

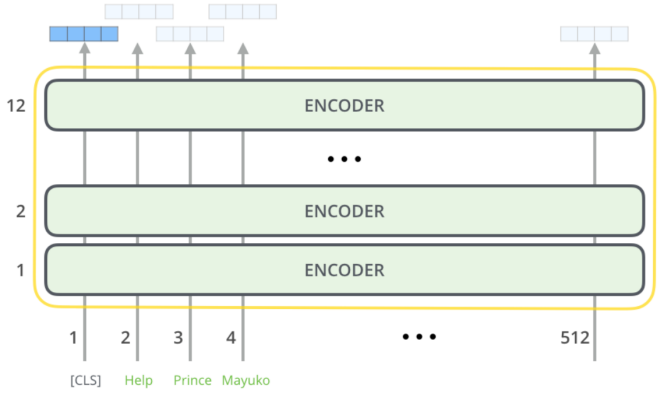

Fig. 3: Each position outputs a vector of size hidden size (768 in BERT Base). For the sentence classification example we've looked at above, we focus on the output of only the first position (that we passed the special [CLS] token to).[7]

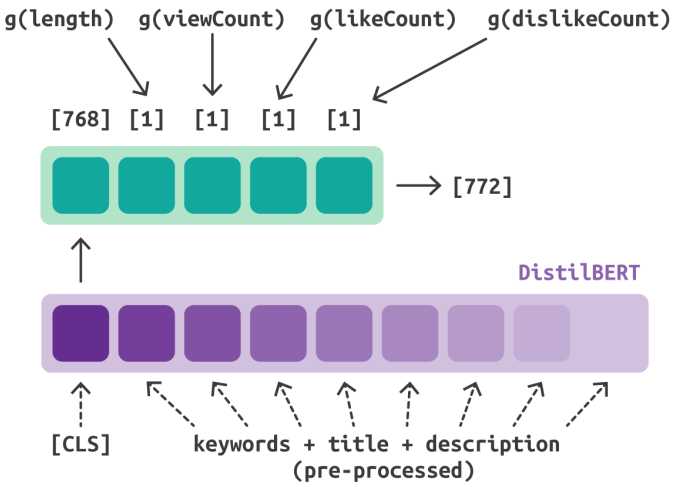

Fig. 4: Pre-processed strings are vectorized using DistilBERT. The output is concatenated with scaled numerical features.

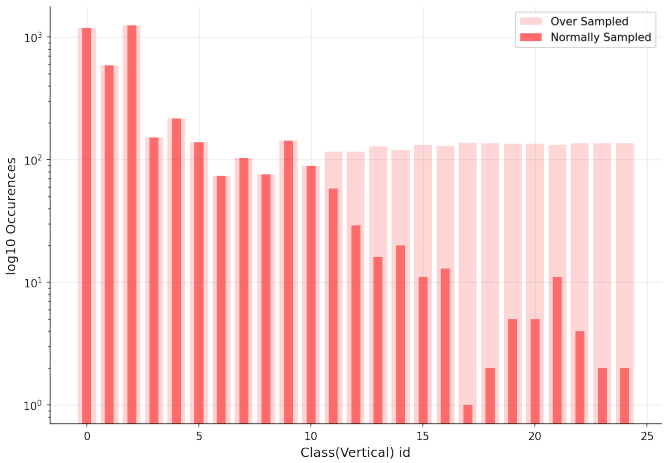

Fig. 5: Distribution of data before and after over-sampling.

Layer (type)	Output Shape	Param #
dropout (Dropout)	(None, 772)	0
dense (Dense)	(None, 64)	49472
dropout_1 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 25)	1625

Total params: 51,097
Trainable params: 51,097
Non-trainable params: 0

Fig. 6: Model summary in TF 2.0.

previous work [6].

The complete code for this project can be found on this link: YouTube-Data-Analysis

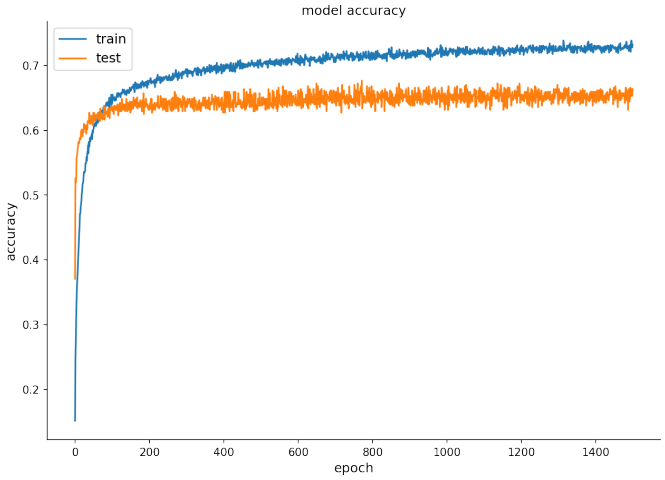

Fig. 7: Training and Testing model accuracy for 1500 epochs in observed best case scenario.

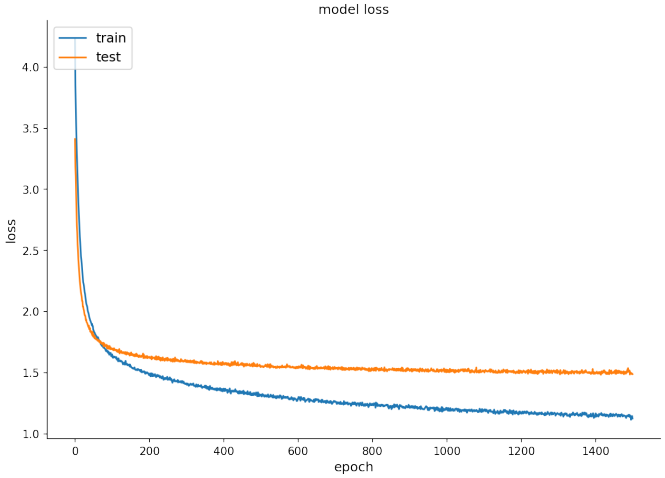

Fig. 8: Training and Testing model loss for 1500 epochs in observed best case scenario.

References

[1] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, *Google's neural machine translation system: Bridging the gap between human and machine translation*, 2016. arXiv: 1609.08144 [cs.CL].

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv:1706.03762 [cs]*, Dec. 2017, arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[3] J. Devlin, M.-W. Chang, and K. Lee, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805 [cs]*, May 2019, arXiv: 1810.04805. [Online]. Available: http://arxiv.org/abs/1810.04805.

[4] A. Ly, B. Uthayasooriyar, and T. Wang, *A survey on natural language processing (nlp) and applications in insurance*, 2020. arXiv: 2010.00462 [stat.ML].

[5] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," *arXiv:1910.01108 [cs]*, Feb. 2020, arXiv: 1910.01108. [Online]. Available: http://arxiv.org/abs/1910.01108.

[6] Lakshya, *heylakshya/youtubeRabbitHole*. Mar. 2021. [Online]. Available: https://github.com/heylakshya/youtubeRabbitHole.

[7] J. Alammar, *The illustrated bert, elmo, and co. (how nlp cracked transfer learning)*. [Online]. Available: http://jalammar.github.io/illustrated-bert/.

[8] *YouTube-8M: A Large and Diverse Labeled Video Dataset for Video Understanding Research*. [Online]. Available: http://research.google.com/youtube8m/index.html.