Game Over: It's Time to Critique the Critics

Matthew Wong

School of Information, UC Berkeley

Abstract

Several NLP technologies are applied and compared on a novel dataset. Baseline classification is improved upon using different strategies, with a focus on word embeddings and iterative model design.

Introduction

In a bustling, information overloaded society like ours, everyone relies on the word of an expert to help make their important decisions. Whether it be by listening to the CDC regarding the latest Coronavirus updates, or to the local food critic about where to eat dinner, our lives are heavily influenced by the experiences of others. However, for consumers it is not always the case that a critic's views align fully with their own, a problem that is exacerbated by sponsors who may give critics compensation in exchange for a biased review. This paper aims to leverage modern Natural Language Processing techniques to help consumers make informed choices. In particular, I am interested in finding a pattern between critics' and users' reviews of video games as well as the difference between the two groups ratings on the same product. A model that can predict when a critic's language might not resonate with users would be indispensable as it could help consumers see whether a critic's opinion can really be trusted.

Approach: Intuitively, there should be a correlation between the language that a critic uses in their writing and the score that they assign a product. The model architectures proposed in this paper should be able to discern these patterns and learn ultimately my target variable: Whether the average critic's score of a particular video game is less than or greater than the average score given by users. I created a total of 5 different models to investigate which architecture and embeddings would produce the best predictions and minimized the binary cross-entropy loss for each of these models for the prediction task. I then performed error analysis by reviewing incorrect predictions as well as the models' confusion matrices.

Methods

Setup: Data collection, cleaning, baseline model creation and training was done using TensorFlow 2.4.1 run locally on a Dell XPS 15 9550. Subsequent model creation and training was done on Google Colab with Cloud Tensor Processing Units (TPUs)

Data: As a passionate video game enthusiast I was quite motivated to use NLP on something related to my pastime. As there was not a premade available dataset that sufficed for my experiments, I decided to scrape my reviews and other data from Metacritic.com using the BeautifulSoup Python package. I chose this source due to its complete records and scraped over 20 years' worth of titles from the webpage. This was a fairly lengthy process totaling over 51 hours of scraping, which produced a dataset with 16,139 individual video game title records. For evaluation I did a 70/30 train/test split on the data.

Features: Along with the title and Metacritic.com URL, I collected release

date, genre, developer, and more importantly every critic and users' reviews and score. Because many of the games had differing numbers of reviews (due to popularity or even infamy) I decided to randomly select 7 critic and user reviews, if available, and concatenate them into one representative review from each group. I binarized my target variable by assigning a positive label to examples where the critic had a higher or equal score than the user and zero otherwise.

Baseline Model: As an initial measurement of and foray into the data I chose to create a basic LSTM (Long Short Term Memory) based classifier. For convenience I decided to use pretrained GLoVe (Pennington et. al., 2014) vectors to encode both critic and user reviews. I chose 100 dimensional GloVe vectors since they would encode more information than the smaller dimensional ones and not require as much computational power as the 300 dimensional vectors. These embeddings were then fed into the LSTM layers, concatenated, and then fed to a series of fully connected layers and finally to a sigmoid activated output layer.

Next Iteration, DistilBERT: Due to constraints on my setup, I was drawn to create word embeddings using DistilBERT due to its speed, ease of use, and importantly its size compared to the full BERT transformer (Sanh et. al., 2019). The model I produced was similar to the baseline model but now included a pooling layer and a dense layer with fewer neurons. Critically different between the baseline model and the future iterations were the inputs which now included the attention masks for both critic and user reviews. In addition, the reviews were truncated to 512 tokens. For the sake of completeness, I decided to create two DistilBERT models, one with a CNN layer and the other with a Bidirectional LSTM layer.

Next Iteration, BERT: While the DistilBERT transformer can retain 97% of the language understanding capabilities as BERT, I still wanted the models to learn as much as they could from the data. By redesigning the data pipeline, I was able to create a model that used the full BERT stack (Devlin et. al., 2018) to create the context-based embeddings for critic and user reviews. Like the previous iteration, I created both a CNN model and a bidirectional LSTM model.

Model Training and Tuning: Optimizer learning rate, epoch, and batch size were varied to find the optimal model. In order for the model to train without resource management errors, I kept batch size between 32 and 64.

Results

The trained models were evaluated on a test set composed of a shuffled 30% sample of the total dataset. The two metrics of interest, Binary Cross-Entropy and Accuracy for each of the models is shown in Table 1. Compared with our baseline test accuracy of 56%, both the CNN and BiLSTM models with DistilBERT and BERT layers outperform the baseline at max **~14%!** This is in line with the performance boost transformers have historically brought to others in the data science community (Adhikari et. al., 2019) (Maslej-Krešňáková et. al., 2020). The best model in the array was the BERT CNN which was surprising since I had expected an LSTM to have better performance on long sequences. Both the CNN and LSTM architectures had generally similar accuracies and losses so the former

Model	Architecture	batch_size	epochs	Learn_rate	Train_loss	Train_acc	Test_loss	Test_acc
DistilBERT	LSTM	64	5	1.00E-04	0.6882	0.5501	0.6833	0.5713
DistilBERT	LSTM	64	5	1.00E-05	0.594	0.694	0.6617	0.6923
DistilBERT	LSTM	64	5	1.00E-06	0.6686	0.6001	0.6637	0.6086
DistilBERT	LSTM	64	10	1.00E-04	0.6869	0.5416	0.6646	0.6122
DistilBERT	LSTM	64	10	1.00E-05	0.2871	0.8893	0.8036	0.6639
DistilBERT	LSTM	64	10	1.00E-06	0.6268	0.6571	0.6347	0.6485
BERT	LSTM	32	5	1.00E-05	0.3695	0.8431	0.6839	0.691
BERT	CNN	32	5	1.00E-05	0.4221	0.8121	0.6145	0.7047
BERT	LSTM	32	10	1.00E-05	0.0327	0.9924	1.2912	0.6862
BERT	CNN	32	10	1.00E-05	0.0742	0.9788	1.0727	0.6701
DistilBERT	CNN	64	10	1.00E-04	0.147	0.9572	1.1793	0.6526
DistilBERT	CNN	64	10	1.00E-05	0.3364	0.8676	0.7363	0.6594
DistilBERT	CNN	64	10	1.00E-06	0.0444	0.9911	1.7554	0.6378
DistilBERT	CNN	64	5	1.00E-04	0.4957	0.7754	0.6704	0.6202
DistilBERT	CNN	64	5	1.00E-05	0.6196	0.6597	0.6281	0.6496
DistilBERT	CNN	64	5	1.00E-06	0.697	0.543	0.6872	0.5893
Baseline	LSTM	64	10	1.00E-05	0.6791	0.5629	0.6832	0.5655
Baseline	LSTM	64	5	1.00E-05	0.6807	0.5652	0.6928	0.5673

Fig 3. Test Loss and Accuracy for Models

point could still hold true with further testing.

A common issue with the models was overfitting to the training data. For the baseline LSTM model, initial architectures allowed for an insignificant training loss and 100% training accuracy but resulted in abysmal test metrics (Less than 50%) which led me to iterate to the current baseline architecture. Our results point to a lower epoch number as optimal as while the training loss and accuracy is better for the ten-epoch models, the test loss is much worse and the accuracy is only a few percent better than five-epoch models. I found that most epoch numbers over ten had diminishing returns on performance, which could possibly just be because people all have different writing styles. The BERT based models had hundreds of millions of trainable parameters, compared to the few hundred thousand trainable parameters in the baselines, which meant that even though Google's Cloud TPU was expediting the training process, it took around thirteen – twenty min to train, compared to the two to three minutes for the baseline model to train. However, thirteen minutes is of course preferable to the estimated two and half hours per epoch training time on my local machine. In addition, a learning rate of $1e^{-5}$ yielded the best performance as higher values caused erratic behavior and lower values took too long to converge to an optimal solution.

Error Analysis: Focusing on the best performing model, we can see where it has some issues. First, we can test the model on some new reviews for a fictional game. I created the following critic and user reviews:

Critic: "Ubisoft pulled out all the stops for the latest release of Assassin's Creed."

User: "I really liked this game. Was slightly disappointed by the graphics but overall, still super impressed."

BERT CNN outputted a value of .4882 when predicting on the toy example above. I changed a few of the words and it seems that mentioning a company really affects how the model makes its decision. (I replaced Ubisoft with Sony and the output became .70!). Is this evidence that Sony may be compensating Metacritic reviewers for higher scores? If we examine some specific examples from the data maybe a pattern will appear. I chose 2 examples that the model misclassified.

Example 1: Midnight Club Street Racing – False Negative

Critic: While offering a decent helping of fun, the bland textures and ubiquitous gameplay make for a somewhat unexciting PS start. The collision physics make flipping pedestrians over your car seem realistic, as well as barbarically satisfying. The greatest shame about Midnight Club Street Racing however is just how repetitive it can become. An extremely fun arcade style racer. A nice breakaway from the norm of arcade racing games, and is definitely worth a look for its impressive visuals and addictive but sometimes a little on the frustrating side gameplay. The sense of speed, the detailed graphics, and most of all, the fun makes this a great addition to the PS lineup. The graphics are very impressive, the sense of speed is downright frightening especially when played from the cockpit of the car and the graphics are an amazing amalgam of detail and functionality.

User: Where the great Midnight Club series began. It’s now available on the PS Store. For a game that was released in, this game looks great. Races can, however, be a pain to beat, and this seem to carry over in each iteration of MC. If you want to see where it all began, or just want to collect all MC games, then buy this. I should warn you though, that customization wise, it isn’t deep as the later games.

The model incorrectly predicts that the user will rate the game higher than the critic. This may be because the critic review is much more detailed and includes richer language (both positive and negative) which was a source of confusion for the model. In addition, the length of the user review was much shorter which the model might have associated with a higher score.

Example 2: Men of Valor- False Positive

Critic: An Oliver Stone directed theme park ride. Dec, p. The battlefield is stained with annoying blemishes, but the game still performs courageously. Dec, p. Certainly the best of the current crop of Vietnam era shooters, and for the record it easily beats Vietcong and Shellshock s attempts. It tells a decent enough story using the tried and tested diary and letters formula which forwards the plot quite nicely. And don t listen to talk about the games hidden agenda. Who cares if its pro American, this is gaming, not art house political cinema, we re here to kick ass and chew bubble gum, and we had a hell of a lot of fun doing it! Generally disappointing. It s a decent game, but just nowhere near as enjoyable or impressive as the many new FPS recruits who have recently landed on the retail battlefield. Multiplayer matches blow it all to hell in the best way. Dec, p. Easily the best Vietnam game to date has done a great job of capturing the atmosphere of the war.

User: I expected this title to rule. But the game simply doesn t seem finished. It has choppy ugly graphics the ai is retarded and the environments make you feel like you re in a mouse maze. It just doesn t feel like a forest when you are just moving down a tunnel with painted green walls. This game is a poor man s Medal of Honor, and that game definitely had room for improvement. I consider myself a war buff, but regret wasting my time with this rushed title. Alright, i like vietnam games and i got conflict vietnam as well... knowing that men of valor will rule over the conflict game and so on...

Both reviews seem to be disappointed in the game and in fact the critic gave it three less points on average than the user. However, the critic review *is* a bit more positive than the user which is most likely why the label was predicted positive. The user review from above was truncated to about a third of the original length. This means that the model probably scored it lower since there were simply more negative words.

While I did sample the same number of reviews (when possible) the fact is that there are vastly more critic reviews in the data

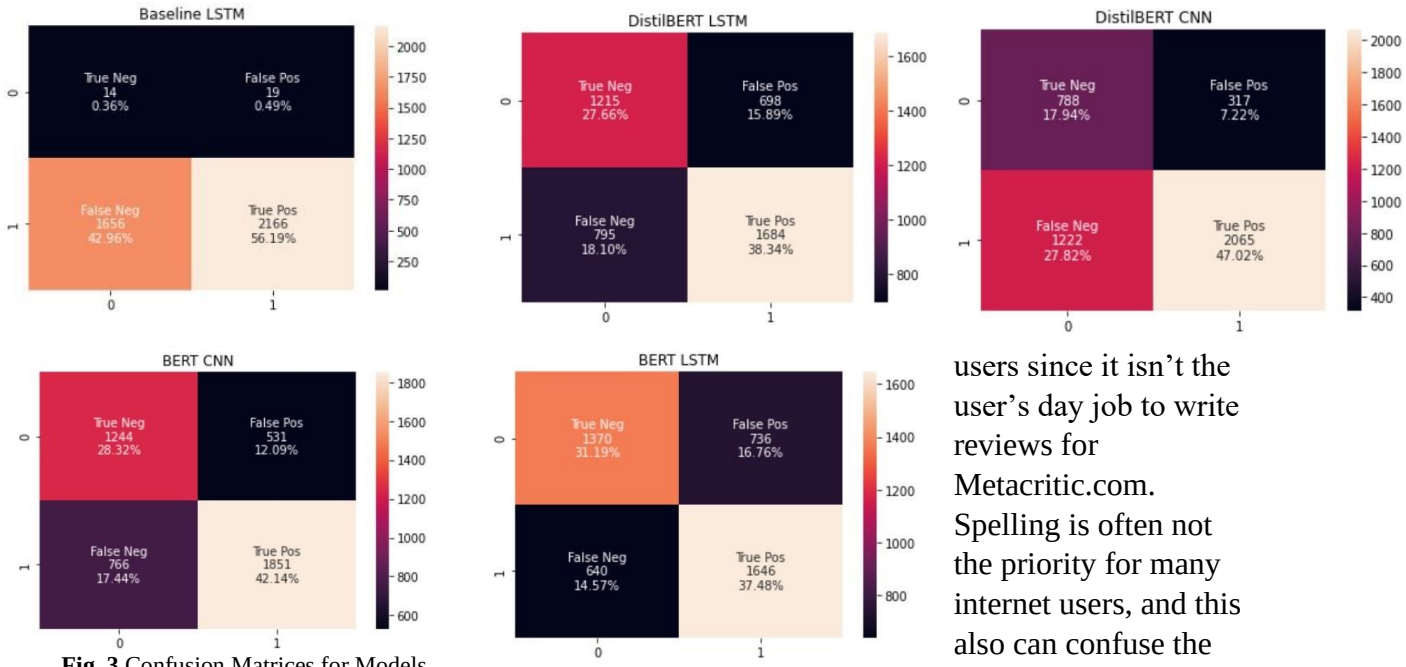


Fig. 3 Confusion Matrices for Models

than user reviews. In addition, the quality of critic reviews is most likely better than the users since it necessarily isn't the user's day job to write reviews for Metacritic.com. (otherwise, they would be a critic!) Spelling is often not the priority for many internet users, and this also can confuse the models since misspelled words will not be properly encoded.

In Fig. 3, we can see the confusion matrices associated with each model. The BERT and DistilBERT models did a reasonably good job at predicting true negatives and positives while the Baseline LSTM almost always predicted the positive label. The CNN architectures had more False Negative predictions than their LSTM counterparts, minus the baseline, which may point to a slight bias in favor of a higher user score.

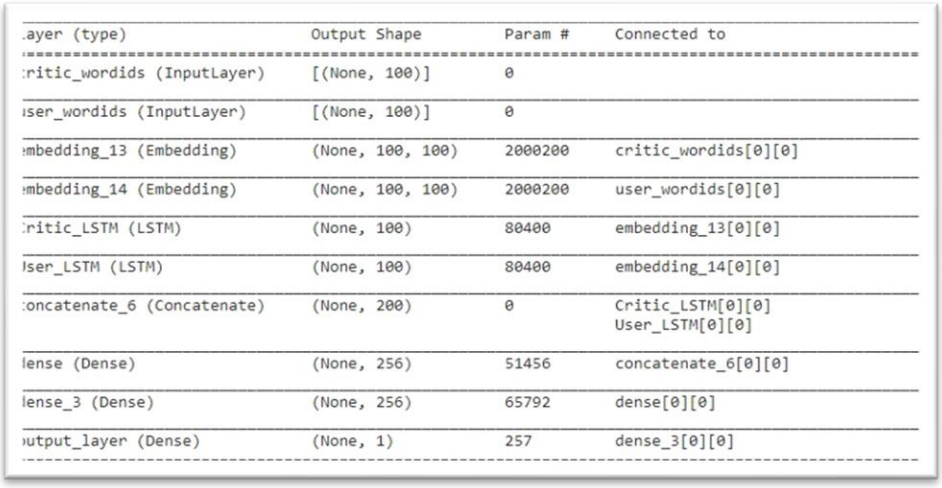
While I did sample the same number of reviews (when possible) the fact is that there are vastly more critic reviews in the data than user reviews. In addition, the quality of critic reviews is most likely better than the users since it isn't the user's day job to write reviews for Metacritic.com. Spelling is often not the priority for many internet users, and this also can confuse the models since misspelled words will not be properly encoded.

Conclusion and Future Work

While an almost 15 percent increase in accuracy from the baseline is a decent result, there are some future areas of research that could push that improvement even further. Scraping data from more websites in order to have more balance between the number of critic and user reviews would help the model understand more about the writing patterns in each population. Further hyperparameter tuning would also boost model performance and even the introduction of other models could prove to be a fruitful endeavor. As for the present, this model accomplishes the goal of detecting a patter between video game critic and user reviews, which could easily be extended to other areas in the retail industry so long as there are critics and users. It is a time in this country where it's more important than ever to know if the experts you listen to are speaking their truth, or

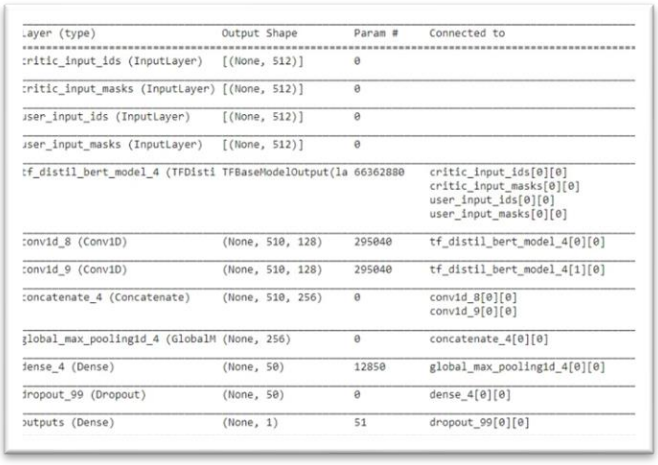
simply as a mouth for some other voice. My hope is that NLP can help bring out the truth in today's information packed age and make the world a more honest place.

Figures

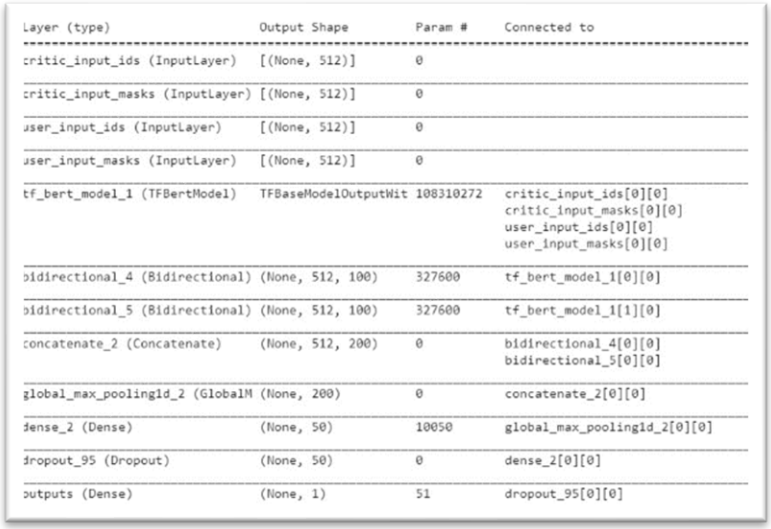


ayer (type)	Output Shape	Param #	Connected to
ritic_wordids (InputLayer)	[(None, 100)]	0	
ser_wordids (InputLayer)	[(None, 100)]	0	
mbedding_13 (Embedding)	(None, 100, 100)	2000200	critic_wordids[0][0]
mbedding_14 (Embedding)	(None, 100, 100)	2000200	user_wordids[0][0]
ritic_LSTM (LSTM)	(None, 100)	80400	embedding_13[0][0]
ser_LSTM (LSTM)	(None, 100)	80400	embedding_14[0][0]
oncatenate_6 (Concatenate)	(None, 200)	0	Critic_LSTM[0][0] User_LSTM[0][0]
ense (Dense)	(None, 256)	51456	concatenate_6[0][0]
ense_3 (Dense)	(None, 256)	65792	dense[0][0]
utput_layer (Dense)	(None, 1)	257	dense_3[0][0]

Fig 1. Model Structure for Baseline LSTM



ayer (type)	Output Shape	Param #	Connected to
ritic_input_ids (InputLayer)	[(None, 512)]	0	
ritic_input_masks (InputLayer)	[(None, 512)]	0	
ser_input_ids (InputLayer)	[(None, 512)]	0	
ser_input_masks (InputLayer)	[(None, 512)]	0	
f_distil_bert_model_4 (TFDisti	TFBaseModelOutput(la	66362880	critic_input_ids[0][0] critic_input_masks[0][0] user_input_ids[0][0] user_input_masks[0][0]
onv1d_8 (Conv1D)	(None, 510, 128)	295040	tf_distil_bert_model_4[0][0]
onv1d_9 (Conv1D)	(None, 510, 128)	295040	tf_distil_bert_model_4[1][0]
oncatenate_4 (Concatenate)	(None, 510, 256)	0	conv1d_8[0][0] conv1d_9[0][0]
lobal_max_pooling1d_4 (GlobalM	(None, 256)	0	concatenate_4[0][0]
ense_4 (Dense)	(None, 50)	12850	global_max_pooling1d_4[0][0]
ropout_99 (Dropout)	(None, 50)	0	dense_4[0][0]
utputs (Dense)	(None, 1)	51	dropout_99[0][0]



Layer (type)	Output Shape	Param #	Connected to
critic_input_ids (InputLayer)	[(None, 512)]	0	
critic_input_masks (InputLayer)	[(None, 512)]	0	
user_input_ids (InputLayer)	[(None, 512)]	0	
user_input_masks (InputLayer)	[(None, 512)]	0	
tf_bert_model_1 (TFBertModel)	TFBaseModelOutputWit	108310272	critic_input_ids[0][0] critic_input_masks[0][0] user_input_ids[0][0] user_input_masks[0][0]
bidirectional_4 (Bidirectional)	(None, 512, 100)	327600	tf_bert_model_1[0][0]
bidirectional_5 (Bidirectional)	(None, 512, 100)	327600	tf_bert_model_1[1][0]
concatenate_2 (Concatenate)	(None, 512, 200)	0	bidirectional_4[0][0] bidirectional_5[0][0]
global_max_pooling1d_2 (GlobalM	(None, 200)	0	concatenate_2[0][0]
dense_2 (Dense)	(None, 50)	10050	global_max_pooling1d_2[0][0]
dropout_95 (Dropout)	(None, 50)	0	dense_2[0][0]
outputs (Dense)	(None, 1)	51	dropout_95[0][0]

Fig. 2 Model Structure for DistilBERT CNN and BERT LSTM

References

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019a. Docbert: Bert for document classification. ArXiv, abs/1904.08398.

Maslej-Krešňáková, V.; Sarnovský, M.; Butka, P.; Machová, K. Comparison of Deep Learning Models and Various Text Pre-Processing Techniques for the Toxic Comments Classification. *Appl. Sci.* **2020**, *10*, 8631. https://doi.org/10.3390/app10238631

V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,” arXiv preprint arXiv:1910.01108, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pretraining of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.