Prediction of the NYT Best Seller Book using an NLP model

Logan Heft (lheft), Samaye Lohan (slohan), Dongjun Shin (djshin25), Youn Kyeong Chang (ychang10)

Github: https://github.com/josh7197/DL_Finalproject_CS2470

Introduction

Books have been inseparable in our daily lives for thousands of years. Despite the increased popularity of digital media, the net revenue of the U.S. book publishing industry is about 25 billion dollars, selling 693.7 million physical books in the U.S. in 2019. [1] However, over the 800 thousand new title books, only 500 of them became New York Times best sellers. [2] The publishing industry profits, like other cultural industries, highly depend on the "hits". The publishers thus may find it helpful at the selection process of the publishing to predict the success of the book in advance. We expect our prediction algorithm would offer a valuable decision process and benefit in strategizing the book promotion. Consumers would also benefit from the model because they will no longer waste time with inferior books. The model predicts best sellers given a title, genre, top author, ratings, number of pages and the year published.

Preprocessing/Data

For this model, we created a dataset on our own using web scraping. Using selenium, we built a web scraper that is able to get information such as title, series, author, rating, genre, publishing company, format and any other information about a book that we desire (as long as it is availableGoodreads). While most of the data was web scraped by our group, we did find another dataset with a list of book titles, and whether or not the book was a New York Times Best Seller. We utilized this found dataset as the titles we were going to build our own dataset from, as well as add the Best Seller binary column to our own dataset. Among the collected dataset, we selected variables of interest which are title, genre, top author, ratings, number of pages, year published and the label, best seller. For the label, we converted it into one-hot encoded vectors using the label encoder and the to_categorical function of the keras.utils. For the features, given that we have textual, numeric and binary variables, we created three different types of inputs. For the first input for textual feature, title, we created a function for removing punctuations, numbers, single character and multiple spaces as well as took GloVe word embeddings. After applying this function to the title, we tokenized the cleaned texts and used paddings to make every text feature have the same length. Lastly, we converted them into numeric form using word embeddings. The second input with three columns was for numerical features, rating, number of pages and year published. We preprocessed them using a standard scaler. The third and the last input consisted of two columns for the one hot encoded binary feature, top authors.

Methodology

We implemented four different models for the textual model, GRU, CNN+GRU, single-headed attention, and multi-headed attention models. Also, we tried to tune the hyperparameters for other layers to get the best accuracy.

1. ***GRU Model***

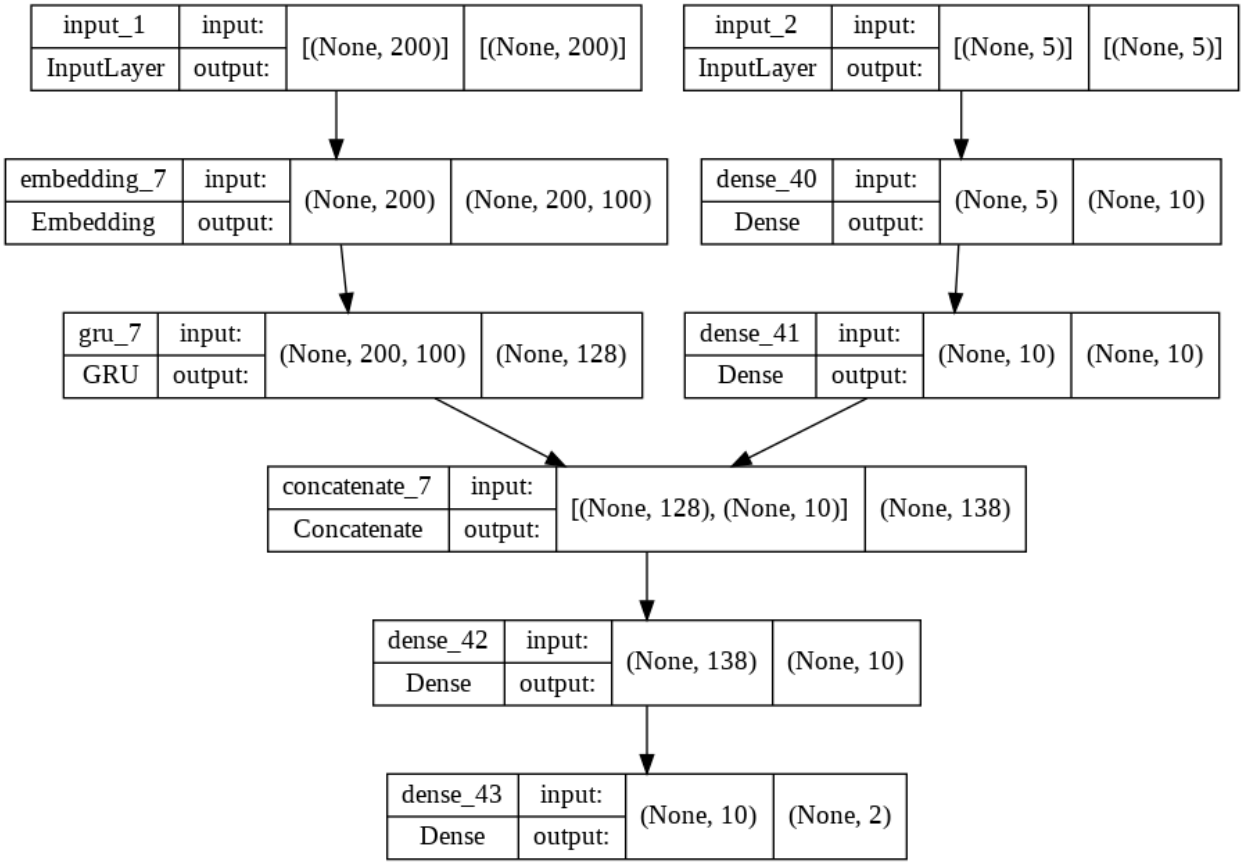

Figure 1: A GRU model

For the first model, we took only a GRU model for the textual input after embedding. This layer was concatenated with the second input which already went through the dense layers. Lastly, this concatenated layer had two additional dense layers.

2. ***CNN + GRU Model***

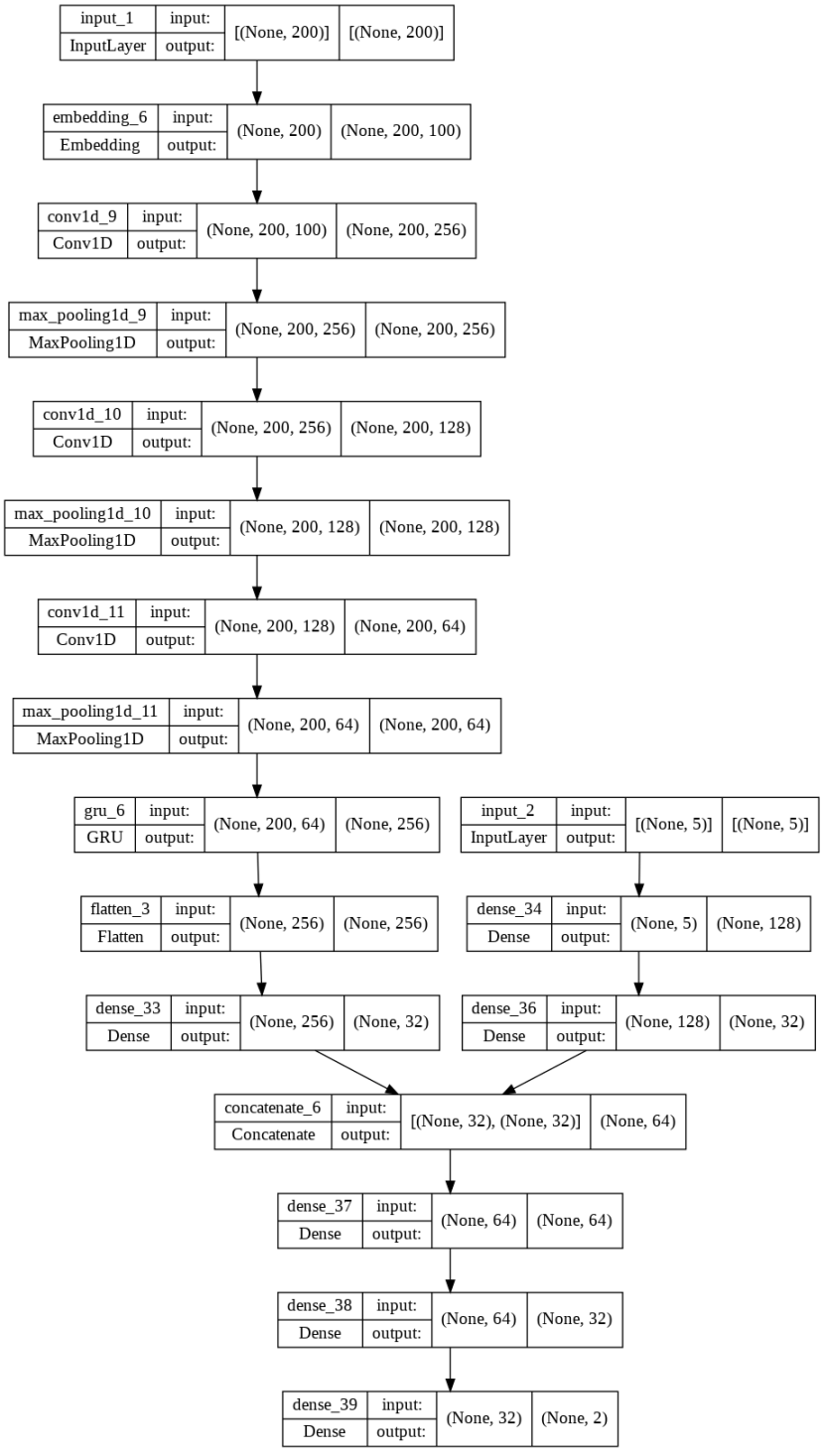

Figure 2: A CNN + GRU model

For the second model, we added convolutional layers to the textual input. Also, to prevent overfitting, we added max pooling layers as well. Similar to the first model, this layer was concatenated with the second input which already went through the dense layers. Lastly, this concatenated layer had two additional dense layers.

3. ***Single-headed Attention model***

We replaced the CNN layers with attention layers because some recent research shows the attention models outperform in natural language processing (NLP) (Sun at el, 2020). Basically, we conducted similar flows to the CNN model. Before we added a single-headed attention layer, we added a bidirectional LSTM layer because we should consider two way implication of NLP, like BERT.

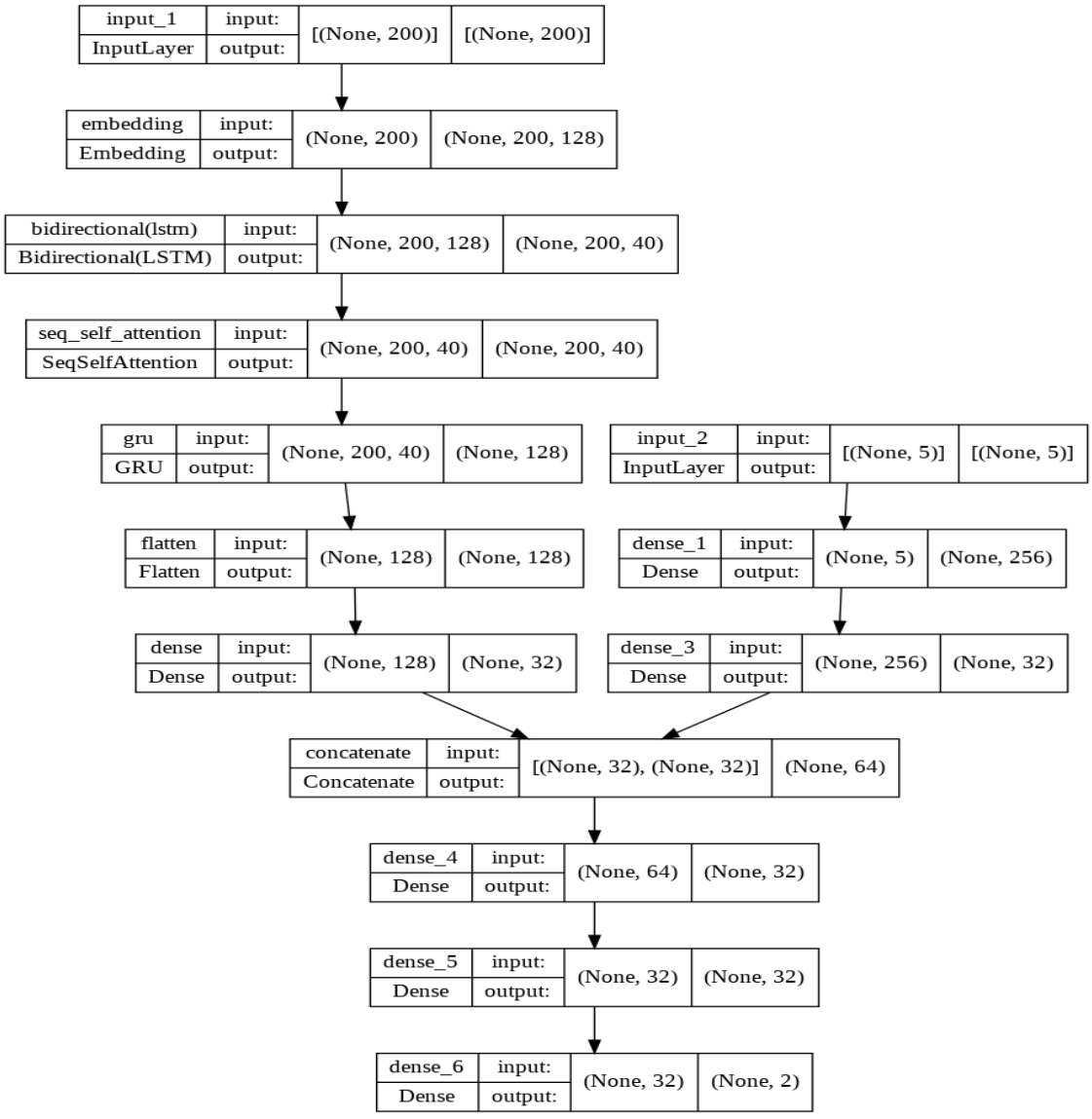

Figure 3: A Single-headed attention model

4. Multi-headed Attention model

We also conducted a multi-headed attention model because a multi-headed attention structure also yields better results. Unlike a single-headed attention model, a multi-headed attention model can figure out two-way word relationships. Therefore, we conduct LSTM with multi headed attention. This is the only difference between our single head attention structure and multi head attention structure.

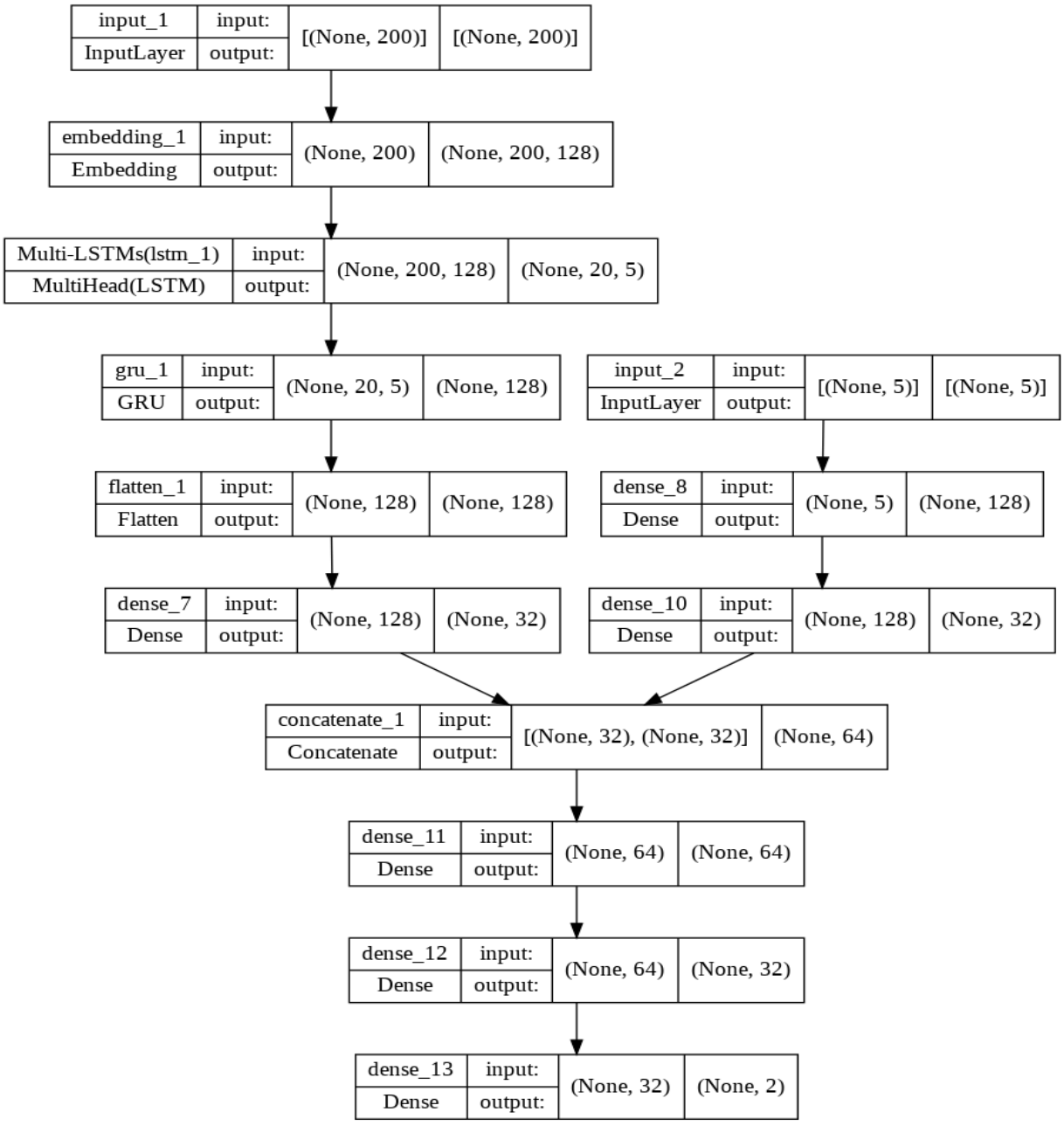

Figure 4: A Multi-headed attention model with “title”(NL variable) and non-ML-variables

Results

Table 1.0: Test Accuracy for each Respective Model Trained

Model	Test Accuracy
GRU	0.73
CNN + GRU	0.81
Single-Headed Attention	0.83
Multi-Headed Attention	0.78

Among our four models, the single-headed attention yielded the best performance in accuracy whereas the GRU model yielded the lowest accuracy.

Challenges

One of the hardest challenges for us was creating the dataset in the first place. While there have been other research studies looking at things like book reviews to predict New York Times Best Sellers, there were no such studies with already published datasets, or working API's to places like Amazon or Goodreads to make data accessible. This meant we would have to create our own dataset, and with nobody in the group having a web scraping background, this was something we needed to figure out ourselves. One of the challenges that we did not realize at first was avoiding Amazon's bot detection. If our program did not wait and operate like a human would, Amazon would flag our bot and temporarily shut down the service to us. Another challenge is to deal with multi-types of the input variables. To overcome the challenge, we applied the submodel at first for each data type and went through the combined model.

Reflection

Our group is extremely content with the way our project ultimately turned out due to various factors ranging from our literary interests to the wide range of possibilities within the NLP domain. Our baseline goal was to start off by implementing a simple recurrent unit model to get a sense of how well it is able to retain information and then make the necessary changes to improve accuracy. Our group wanted to achieve an accuracy of 85% as it was a realistic enough target given the vast nature of the meta information tied to books. Our initial approach was to use the "title" feature as the input in order to predict whether the book is a New York Times bestseller. However, we soon realized that our GRU model yielded a relatively low accuracy, 73%, due to the fact that we were only considering one textual input. To

further improve our model, we concatenated multiple models together with different inputs, textual and integer, in addition to appending max pooling layers to reduce overfitting. With every new variation to the baseline model, our accuracy increased for the most part except for the multi-head attention model which can be attributed to the sparsity of input data available. In other words, pivoting from a single textual input to concatenating multiple models with various input features, allowed us to more accurately depict if a given book is a NYT bestseller or not. One of the biggest limitations in our project was the nature of the dataset as it was limited by the information present in GoodReads and NYT API and if we were to re-do this project, we would have narrowed down our dataset to be representative of a specific subset of books, such genre or age group. The reason for this is that it would help linearize the model's trajectory more clearly and provide a better stream for information retention. In addition, we would have also liked to leverage GCP's computing capabilities to better train the models so that we are not limited to a subset of hyperparameter tuning. Lastly, our group's biggest takeaway from this project is that it is really difficult to generalize certain datasets, especially within literature, that contain an infinite number of features that can be linked to them. When we were conducting web scraping, we actively made decisions on which features to include and exclude based on intuition as to what we believed to be valid predictors for our problem. Therefore, leveraging virtual GPUs and deciding on whether to include all the information about a specific feature or use a more niche dataset such as children's book to predict NYT bestsellers.

References

[1] Toner Buzz: Eye-Popping Book and Reading Statistics.
https://www.tonerbuzz.com/blog/book-and-reading-statistics/ [Online; accessed 11-Apr-2022] (2022)

[2] Statista: U.S. Book Industry/Market—Statistics & Facts.
https://www.statista.com/chart/26572/average-number-of-books-read-by-us-residents-per-year/ [Online; accessed 11-Apr-2022] (2022)

[3] Xiaobing Sun and Wei Lu, "Understanding Attention for Text Classification ", Singapore University of Technology and Design(2020)

[4] Goodfellow, Ian, et al. *Deep Learning*. The MIT Press, 2017.

[5] Krbarounis. "KRBAROUNIS/NYT-Bestsellers-Classification-Project." *GitHub*, https://github.com/krbarounis/NYT-bestsellers-classification-project.