Gregory Knapp

CS 472 Final Project Report

1. Problem to Solve

In the course of my work as a Research Assistant in the Harold B. Lee Library, my supervisor proposed the idea of making a digital bookshelf that included both print and eBooks that we had access to and shelved them in call number order. The idea was simple enough but had one main challenge. When print resources are shelved in the library, a cataloger assigns it a call number which dictates the book's location on the shelf. However, eBooks do not always come with call numbers in the metadata, which would make an ordered digital shelf difficult. On top of this, call numbers for a book are not always one-to-one across different libraries; BYU may categorize things slightly different than the University of Utah while both following the general guidelines of the Library of Congress. To help assign call numbers to eBooks to place them on a digital shelf, I decided to look at different deep learning models to find out which performs the best at assigning (classifying) eBooks into call number ranges.

2. Data Analysis

Since my supervisor is the Psychology Librarian, we limited our training data to print items in the Psychology (BF Library of Congress Identification) section. From a previous project I had dataset of 24,749 print items that I used to start building my model. From the master dataset, I extracted just title and author names along with the assigned call number. In order to simplify the classification problem, I used the Library of Congress BF classification categories (found at the Library of Congress website) in order to assign the books to 1 of 53 call number ranges. It would be interesting to see if a model can be trained to generate a more precise call number based on the inputs, however I decided to use the Library of Congress ranges to make it a classification problem.

Initial tests on the dataset were promising, but two issues slowly became clear. Firstly, I did not account for the fact that there were duplicate items in the dataset for books with multiple copies. Secondly, there were not many other useful features in the master dataset that seemed helpful to predict call number aside from title and author name. Fortunately, a cataloging staff member met with me and showed me how I could get a more complete metadata record of each book. Using a web crawling script, I pulled JSON data for each unique book ID in our original dataset from the library website. The result was a dataset with 18,895 items that had more features such as subject headers, publication year, and various note records such as the table of contents, book reviews, and commentary on the item. This duplicate-free dataset became the final dataset for my project that I used in my testing that is discussed in the results section.

3. Technical Approach

To classify books into call number ranges, I knew I would need a model that could handle text inputs. I read a Medium Towards Data Science post that used a LSTM model to classify books into genre based on title which served as my first influence for the problem. I adapted my code from an earlier project in this class and added a 53-node linear output layer to the LSTM model to adapt it for classification and tested it with my dataset. The model trained and learned to create output labels, but only at about a 50% accuracy rate. The model also began overfitting rather quickly and did not seem ideal to handle a classification problem with so many outputs.

Instead of the LSTM model, I turned to the reliable Transformers library for a variety of large language models that could be adapted for classification problems. In this project, I focused on two:

BERT and GPT2, since I was most familiar with them and had experience using them in text-generation problems. Although the models were pre-trained, setting them up for my dataset took more steps than just switching the model out in my LSTM code. I needed to add a classification head to BERT just like the LSTM. GPT2 had a classification version in the Transformers library which I just needed to change the number of outputs to match my categories. Each of the large models had their own tokenizer that I needed to apply when saving my data to my Torch Dataset class. BERT's tokenizer worked like a charm and took my text inputs with no issue. GPT2's tokenizer was much more difficult to work with, as it insisted on me adding end of string characters to the tokenizer and would not train unless it was exactly perfectly set up, which caused me a fair share of grief. It was worth the effort, as both the BERT model and GPT2 models heavily outperformed the LSTM model and reached accuracies in the 70% range.

Even with this improvement, I was still not completely satisfied. Instead of changing the models however, I decided to do more work my dataset. I was using just title and author as inputs, but I began experimenting with the other fields I had available to me. I ran tests that mixed and matched title, author, subject headers, and note fields (as described in my data analysis section) to varying results. The testing was worthwhile as I saw significant improvement from my previous tests, and one clear winner of input data format and model emerged in the end.

4. Results

The following table displays the results of the various tests. BERT consistently outperformed the other two models in all metrics and was clearly the best model for this problem. Using the just the title and author as inputs worked fine, but performance improved when the subject headers were included. When notes were added, accuracy slightly dropped, indicating that the note information was too much text that did not necessarily connect to the classification trends. The best result overall was BERT with just the subject headings, which got the highest test accuracy in only 7 epochs of training, and still had room to continue learning.

Based on these results, it was clear that going forward in my work this summer I can use the BERT model and focus on subject headers in making my classification model. I think with a longer training run it could get above 90% test accuracy, which feels good for a 53-category problem.

Model Name	Epochs	Train Loss	Train Acc.	Val. Loss	Val Acc.	Test Acc.
LSTM	25	1.379	n/a	2.003	54.30%	54%
BERT (Titles)	5	0.692	69.00%	0.88	58.70%	61.20%
GPT2 (Titles)	5	0.856	54.20%	0.986	48.90%	49.90%
BERT (Titles	10	0.224	91.40%	0.758	67.00%	68.30%
GPT2 (Titles)	10	0.555	66.80%	0.914	54.20%	54.60%
BERT (Titles+Subj.)	10	0.121	96.40%	0.365	84.30%	85.40%
GPT2 (Titles+Subj.)	10	0.246	86.30%	0.561	71.10%	70.40%
BERT (Titles+Subj. + Notes)	11	0.111	96.70%	0.392	82.80%	84.60%
BERT (Subj. Only)	7	0.243	91.20%	0.354	84.90%	86.50%

***All tests were run with the 18,895 item dataset, and using a 80/10/10 Train/Val/Test split. The training split was run with a set seed so datasets were identical each time, in order to make a more similar comparison across the different models.**

Time Log and Code Links:

Date	Activity Category	Activity Name	Time Spent on Task (in hours)
Wednesday, February 23, 2022	Reading	Read Literature about LSTM and classification from Medium	2
Wednesday, February 23, 2022	Data Work	Use old library dataset to build 24,000 item set.	3
Thursday, February 24, 2022	Reading	Read Literature about GPT2 classification and BERT classification	3
Thursday, March 3, 2022	Coding	Start building LSTM model using code from earlier in the semester as reference.	3
Thursday, March 3, 2022	Coding	Continue building LSTM model and get it to start training successfully.	2
Thursday, March 10, 2022	Coding	Implement classification models using GPT2 and BERT models from the Transformers library.	4
Friday, March 18, 2022	Coding	Implementing a GPT2 Classification Model for comparison. GPT2 tokenizer was very difficult to work with.	4
Thursday, March 24	Data Work	Create CSV file using webcrawler to parse library website. Various issues came up during the process	4
Thursday, March 29	Data Work	Extracting desired elements form JSON file created by web crawler to make training data variations	3
Friday, March 30	Coding	Refining BERT model code to handle new datasets and add early stopping options for training	2

Hour Totals by Activity Type:

Reading: 5 Hours

Data Work: 10 Hours

Coding: 15 Hours

Total: 30 Hours

Code:

Bert Model: https://colab.research.google.com/drive/1elI9cPAN-JrKwWuvQWaSApGp8PJ8KxuO

GPT2 Model: https://colab.research.google.com/drive/1WHYbxttRMBXl4Pybnrw54Advm1Q4xCKk#scrollTo=ly_81lzPCTT9

LSTM Model: https://colab.research.google.com/drive/1RMFKlSMIBt6Nz2Hr6F-uQ1tMtWbxbXI6