Richard Buehling, Robert Hannon
CS505: Introduction to Natural Language Processing
Professor Wayne Snyder
12/23/23

"Attempting Color Commentary Analysis in Sports Broadcasting through Transformer Models: Attempting to Leverage Interview Transcriptions for Play-by-Play Insights"

Abstract

This research investigates our approach to provide color commentary analysis for sports games by leveraging transformer models and interview transcriptions due to the absence of direct datasets for play-by-play commentary and color analysis. With limited text data available and traditional methods such as manual transcription proving to be slow and inaccurate, alternative sources of textual data were explored, leading to the discovery of a substantial collection of sports interview transcriptions. The primary objective was to fine-tune various transformer models to bridge the gap between interview transcriptions and the desired color commentary analysis. An examination of different large language model and transformer architectures was conducted to comprehend the capabilities of each model at trying to handle the task at hand. This exploration provided a deeper understanding of the nuances of different transformer models and the specific tasks each model excels in.

Procedure

We conducted several experiments with different LLM and transformers to figure out the effectiveness of large language models and of transformer models in inferring color commentary insights from interview transcriptions. The project delved into the optimization and fine-tuning techniques required to adapt these models to the unique demands of sports commentary analysis. Evaluation metrics, mostly in the form of perplexity, were used to measure the performance and effectiveness of each model in generating comprehensive color commentary analysis based on the available textual data. We also used the general 'eye-test' for our outputs in evaluating how our model was performing.

We were initially not too ironed out on all the details. A thought we had was using the play-by-play text data from sports games and the associated color commentary to train a language model. Play-by-play data is a very generic and factual statement of what is going on in a game, while color commentary is usually done by a different person simultaneously and is more emotional and not just facts about the game. For example the play-by-play might be "Professor Snyder takes the Three Pointer ... and it is good" while the color commentary might be "Wow that was from deep!". We got approval for this topic and started researching. However, we soon discovered there is almost no data on this topic. There are no transcriptions of any professional baseball, basketball, soccer, etc. games that we could find from any radio or TV sources. The best we could find was one radio call of a 2006 World Series game named “Sports” by Ken Goldsmith, however this did not have any labels on who was speaking. A last resort we thought was potentially transcribing Youtube videos of full games using a premade script or just a Google Chrome Extension, but these all had their own issues and were pretty inaccurate. We would also have to come up with a solution for wrangling this text into different voices, or for coming up with which lines were the play-by-play and which lines were the color commentary. In our search for transcriptions we found https://www.asapsports.com/ which has transcripts of all interviews for almost any sports games. This is labeled for who is asking the question and who is interviewing, and is a sufficient source of data with almost 200,000 interviews.

We planned on just fine tuning the T5 model for this topic, specifically for sequence to sequence input to outputs and question and answering, since we knew that the T5 model was good for fine tuning on sequence to sequence tasks. We planned on just labeling the interviewee and interviewer with tags in our data after creating text files by just copying and pasting the interviews from the site, as there was no better way to extract and export the data. We then planned on using the questions as the input values for training and the answers as the targets for computing loss, however we found that we had to re-adjust our thinking for how we would fine tune our model. We found that this model was not best for this task, and instead worked on the methods below for fine tuning transformers. Although these transformers have already been trained on enormous sums of data, we thought that our data that was specifically transcribed interviews may be able to teach the transformers how to respond to our input in more of a conversational sports specific syntax.

Our initial thought of just copying and pasting all the interviews from the site was extremely naive in how much data we thought we would need. This methodology was inefficient and would never give us enough data to do the training we wanted. Instead we learned a bit about python web scraping and the BeautifulSoup library for requesting page data. We then ‘scraped’ the page for the questions and answers, disregarding the names of the interviewers and interviewees and the labels given to the questions or answers. We then took about 50,000 interviews, each with many questions, to create our dataset which ended up comprising 344,072 questions. We saved this to a CSV for future reference and so that we did not have to re-compute this step which took about 12 hours.

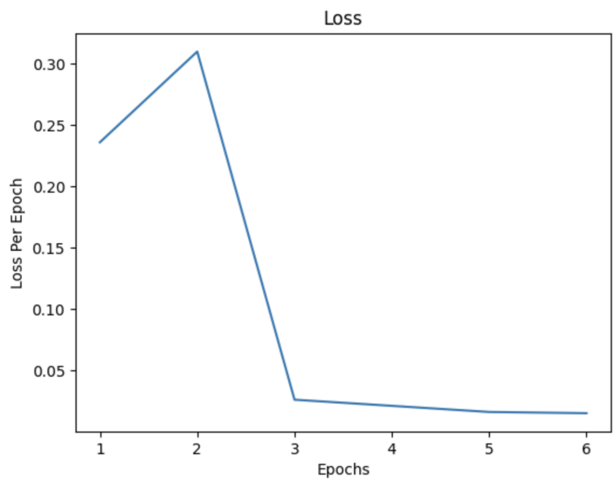
Method: Chat QA with GPT-2

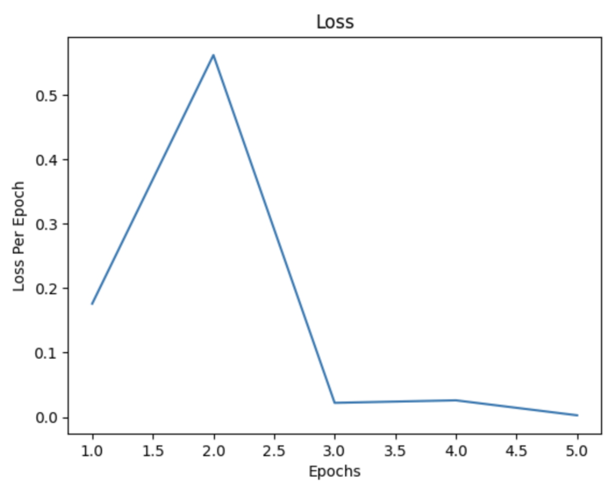
We sought to fine tune the GPT-2 base model specifically to recognize the intricacies of sports specific text. Since it is already trained on such a large corpus of text we had to adjust how it is normally fine tuned to specifically fit our task, although it has already likely been trained on a good amount of sports specific text. We hoped our transcribed interviews would teach our model how to produce outputs in a more conversational syntax. We took the initial GPT-2 Model which required us to change our data by adding starting, ending, and padding tokens to each input for training. We also added a new special '<ans>' token for our answers specifically, since we made our inputs for training: start token + question + <ans> + answer + end token + pad token(s). We then made a dataset using PyTorch of the input embeddings given by the GPT-2 tokenizer and attention mask, which is a list the same length of the input that is 1 at each index besides the pad tokens which are 0. We turned this into a DataLoader and used the recommended optimizer for GPT-2 fine tuning, Adam, to run training on this model. We could then test outputs to this model by passing start token + question/play-by-play + <ans> as inputs to be encoded and passed to our model, and our output would be our desired color commentary. We also could then compare outputs to the regular GPT-2 base model.

Method: BERT for Next Sentence Prediction

BERT has an interesting task that it is able to be fine tuned on which is for next sentence prediction. We knew from BERT's initial training that it was very good at understanding context and relationships between words within a sentence, or for our case, words within questions and associated answers. With more research we discovered that BERT is particularly good at understanding the coherence between sentences, which we thought was perfect for our task. With enough training on sports text data we believed we could train BERT to understand the contextual relationship between sports terms. For our training we figured that our first sentence could be the question and the next sentence could be the answer, even though both of these values were usually more than one sentence long. We specifically imported the model from the transformers library, BertForNextSentencePrediction. BERT is supposed to be trained to be robust and not just spit out exactly what it has been trained on. For this reason the learning rate must be kept relatively small. For next sentence prediction training for some probability the model should be fine tuned on a completely random target that is not the one associated with the input, as to teach BERT what sentences are not associated with each other as well. For this reason our dataset for this training was with probability .2 not the answer associated with the question. We then encoded the questions (inputs) and answers (outputs) with the BERT tokenizer. These encodings were turned into a Dataset and we subsequently had to use a DataLoader with a very small batch size as to not exceed our GPU usage. We ran this through a training loop using the Adam optimizer again and then compared outputs of our model compared to the BERT base model.

We ran two different trainings on half of our dataset for trying to fine tune the BERT base model and compared outputs of our fine tuned model to the base model. For the first training the learning rate we used was .000001, and for the second our learning rate was .00005 with one less epoch to account for our plateauing loss and trying not to let our transformer memorize the training data set. We got significant improvements as seen from the graphs of the second fine tuned model.





However, trying to use this fine tuned model to actually generate text was not successful. Although the outputs made sense grammatically and syntactically, they did not make sense for color commentary. Instead, to test if our inputs were outperforming at least the base model for color commentary we took our inputs that we were originally testing and gave a sample output that we believed would make sense for color commentary. When turning both models back to performing next sentence prediction with the play-by-play input being the first sentence and the color commentary being the second sentence, our fine tuned model was able to predict more accurately than the BERT base model that the color commentary would be a more likely next sentence, which we tested with these inputs and the following code:

```
input_text = "The keeper clears the net"
next_sentence = "He launched that thing down the ice"
```

```
input_text = "The put is being taken from the far side of the green"
next_sentence = "The crowd is going to absolutely roar if she sinks it"
```

```
input_text = "The kick is from fourty two yards"
next_sentence = "No way he makes this one"
```

```
input_text = "The shot is taken from three"
next_sentence = "Wow what a shot"
```

```
print(("Fine Tuned Model" if ((-1*outputs_bert.logits[0][0])/ \
 (outputs_bert.logits[0][1])) < ((-1*outputs.logits[0][0])/ \
  (outputs.logits[0][1])) else "Regular Model"))
```

Our fine tuning tuned down many of the weights of the existing model, so we had to scale the outputs of our testing to compare the bert base model as given from 'outputs_bert' to our model's outputs given by 'outputs'. Our model predicted that our fine tuned model would more likely pick the next sentence, aka the color commentary as input over the original model. Our model however less accurately predicted that the below input would be color commentary for the given play-by-play, which we presume is because the only 'sports specific' text is 'sideline'.

```
input_text = "She is running up the sideline"
next_sentence = "Look at her go"
```

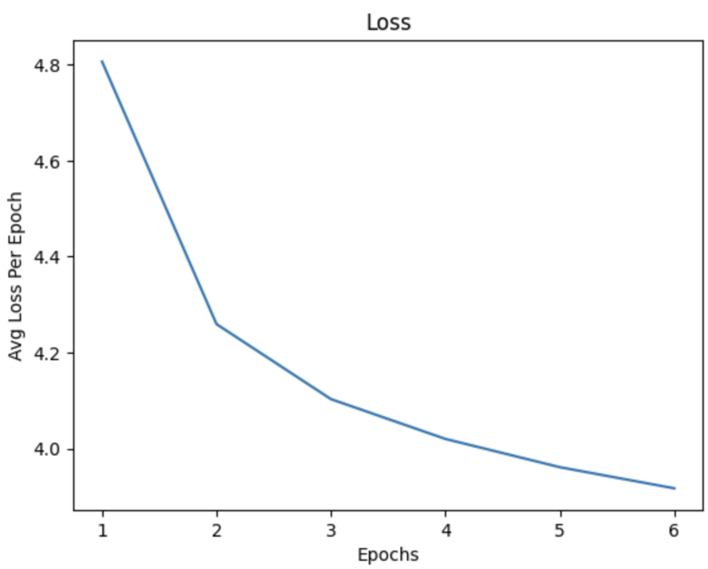
Failed Method: T5 Question and Answering

We researched online for many different question and answer transformer models and found that the T5 model is the preferred method for this task. We wrangled this data into its desired form by turning our questions and answers into their encoded forms before discovering the necessity for 'context', which is a required parameter for question and answer tasks. The context is the information that is surrounding the answer after the question is posed. For example if someone asked "What color is the sky?" the context may be a line from a textbook that says "Blue light is scattered in all directions by the tiny molecules of air in Earth's atmosphere. Blue is scattered more than other colors because it travels as shorter, smaller waves. This is why we see a blue sky most of the time." (NASA) while the answer would just be "blue". In all cases the answer must be within the context. We tried to disregard this and just pass along the question but this did not work. Our idea of question and answering for interviews was not what question and answering tasks are in the context of transformers.

Method: Recurrent Neural Network for Question and Answering

We attempted to expand our understanding of HW 5 for our task. The hardest part of this task was trying to decipher how we would train our long short-term memory architecture to understand the difference between questions and answers. We attempted to do this in a similar way as HW 5, except instead of bracketing our tokenized sentences with <s> tokens we instead bracketed our questions with <q> tags and answers with <a> tags, then adding these tokenized sentences together for training. We used GLoVe embeddings similar to HW 5 for our embedding layer to our architecture and also used a similar architecture with our LSTM and linear layers. We attempted to visualize our results by passing inputs in the shape of <q> + question + </q> + <a>. This gave us our output which we stopped computing once the token was outputted by our model.

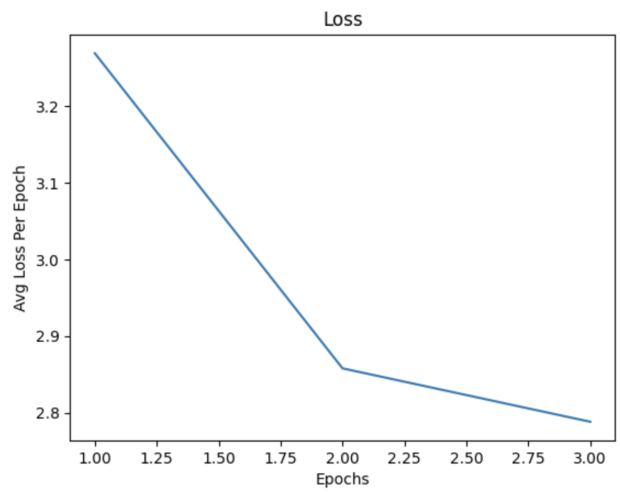
This model did not produce any extremely insightful results from training. We ran training using a quarter subset of the total data, just getting the correlated inputs and targets using sklearn's train_test_split to take 20% of the total question and answers for 68,814. The model built from before was then used and had its hyperparameters adjusted for size of the hidden layer, dropout rate, number of epochs, and learning rate. At first trying to train with 6 epochs, a .0001 learning rate, a hidden size of 256, and a batch size of 64 we got the following results:



The outputs also for the sake of the eye test for most of our tested inputs were repeated outputs of the type similar to:

```
I think it 's a great golf course . It 's a lot of fun . It 's a different golf course . I think it 's going to be a beautiful golf course .
```

Adjusting hyperparameters and running additional training with a hidden size of 512, and 8 epochs led to overfitting of our training set, and exact answers were repeated by our model even for inputs that were not exact questions from the question subset that the model was trained on. We tried training our models on a variety of inputs, some being questions that would be similar to the training questions such as "Does it give you a different perspective having to fight through it this way?” and some being some of the play-by-play commentary we wanted color commentary such as “The shot is good”, but continued getting outputs of the same repeated outputs or exact training data outputs. We learned out model was obviously overfitting and tried a larger learning rate of .005 since our model was training too finely on the training data, along with 3 epochs since our loss was plateauing in the previous models:



This produced much better results. For examples of inputs and outputs:
An example of a question similar to the dataset:

```
start_sequence = ['<q>','How', 'do', 'you', 'feel', 'going', 'into', 'this', '</q>', '<a>']
```

```
I think it 's a great experience .
```

An example of a play-by-play commentary:

```
start_sequence = ['<q>','The', 'kick', 'made', 'it', '</q>', '<a>']
```

```
It 's a lot of fun .
```

An example of a question similar to the dataset that may also be similar to play-by-play commentary:

```
start_sequence = ['<q>','how', 'was', 'the', 'game', '</q>', '<a>']
```

```
I think it 's a great match .
```

These outputs show that the recurrent neural network can be good for picking up nuances in the data, such as that ‘game’ is sometimes used synonymous with ‘match’, however it could not learn to give us color commentary from our play by play data. There is a chance if we used all

344,072 it may have performed better with some fine tuning of hyperparameters, but it could not compete with a transformer model that is trained on countless times the amount of data, with a much more complex architecture than the LSTM. We were still able to train this model to give us some sports syntax and speech, but with no direct training on the task at hand and no way to tell the model exactly what we needed it to do outside the training data we could not get color commentary analysis.

Commentary Generator: A Prototype

In the colab notebook called “GPT2_Deepdive”, the exploration of the capabilities of the fine tuned GPT2 model on the interview data is further explored to create a prototype of a commentary bot. First, the model was tuned to produce more realistic outputs. It was found that a higher repetition penalty, a higher temperature, and sampling techniques were best for generating the most realistic completions. Now, from a visual analysis, it was clear to see that the fine tuned model was generating more conversational text and sports related text. Below is an example of the text generated from the fine tuned model vs the general GPT2 model.

```
Text from Fine-tuned Model:
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
 What?   I think it's a little bit of an adjustment. It was just the first time we were playing in Zhuhai and then you h

Text from Default Pre-trained Model:
What?
The first thing I noticed was that the "I'm a fan of your work" message on my profile page. It's not like you're saying,
```

Now, integrating this into a more realistic commentary chatbot, proved to be a more difficult task. For a prototype, using the GPT4 model through the OpenAI API with custom prompting *in combination* with the fine tuned model proved to be a promising method of accurate commentary definition.

Specifically, the method that was utilized was using GPT4 to take on the role of a sport commentator and then generating arbitrary interview questions. From there the arbitrary questions were then used as a prompt for the GPT2 model and the GPT2 was used to generate realistic responses. Below is an example interaction between the two models:

GPT4 Generated Question:
This season has been a rollercoaster for the Knights. How do you think this victory will impact the team's morale and performance moving forward?

Fine Tuned Model Response:
I mean it is going to be tough because we have so many guys that are playing really well right now but they're not getting too far ahead of themselves or their players in our clubhouse as much as usual which means there can still get some momentum from them on top off. So hopefully

tomorrow we'll see how things go again with those two teams just keep battling each other out here at home field. We've got good pitching. And if anything like today doesn't matter until next year everybody gets healthy then yeah let me say one more game.

Now, comparing the dual-model text generation to one in which only GPT4 generates questions and answers leads to interesting observations. Below is an example of a question and answer generated solely from GPT4:

GPT4 Generated Question:
Rodriguez, you've been phenomenal tonight. What was going through your mind when you took that final shot?

GPT4 Generated Answer:
Rodriguez: Honestly, I was just focused on the basket. I knew I had to make it. I trusted my training, my teammates, and just let it fly. When it went in, it was just pure joy.

One of the first differences between the dual-model generation and the GPT4-only generation is the tonality in the responses. Notice that when the fine-tuned model generates the responses the wording is far more casual and conversational. Now, GPT4 generated responses seem to have a better grasp of context - which is to be expected given the larger context. Perhaps most interestingly though, is the argument that can be made that a lower powerful model, in a specific use case, can generate potentially more realistic responses then a far more powerful model that is not fine-tuned.

Conclusion

The findings of this project showed the potential of transformer models in repurposing interview transcriptions to facilitate play-by-play commentary analysis. The study not only highlights the adaptability of transformer architectures but also shows their applicability in domains with limited specialized textual datasets since they have already been trained on enormous corpuses. Our methodologies for trying to generate comprehensive color commentary analysis in sports broadcasting, shows both the successes of transformers and the limitations of some large language models.

Code

All code can be found on github at https://github.com/robhannon/505_Final_Project

Statement of Contribution

Robert: Did the preprocessing, extraction, and cleaning of the data, along with processing the data as CSVs to be saved on GitHub in a suitable format along with loading the data back into the notebook. Made the initial GPT-2, BERT, T5, and RNN models and did the testing and training for the BERT model and RNN. Drafted up the project plan and the final draft for all parts of the project that I did as described in the preceding sentences.

Richard: Did the deepdive on the GPT2 fine tuned model, including fine tuning the model, using it in combination with GPT4, and implementing rudimentary tests. Contributed the findings to the report and also formatted and styled the report and notebooks.