NFL Play-by-Play Pipeline: Generating and Validating Natural Language Descriptions

Winter 2025

Marissa Fluharty

Abstract

Predicting the outcome of plays in the NFL is a difficult task, but one that is doable using player, game, and play-by-play statistics. However, taking that prediction and generating an explanation for coaches to use as a tool is not something as explored. This projects main method aims to tackle this issue through the training and evaluation of a T5 model. The model's main task is providing a stylized play-by-description while predicting the outcome of a play. After evaluating the model using multiple evaluation metrics, including a BERT model to ensure accuracy, this T5 model had relatively good performance. Further study could be done using different pre-trained models or different features in the input to the T5 model.

Introduction (5 points) This project's main goal is to solve the difficulty of play calling within the NFL, making use of the enormous amount of data that each play accumulates. Play calling in the NFL is a hardened skill gained through experience, but even so, sometimes experienced coaches call the wrong play.

With the model created in this project, a predicted outcome can be given to the coach based on a play they would like to make. Using this, coaches can compare the pros and cons of different plays and ultimately make their decision. This problem can help eliminate some of the variables that coaches have to consider when making play calls, so that they can focus on the more nuanced human element of coaching that a model can not so easily measure.

Research has been done surrounding NFL game outcome prediction, as well as analyzing play-by- plays to count points of games, but focusing on play-specific outcomes can be a useful tool for coaches in real time. The importance of this project lies in the generative nature of it. Using the play-by-play data and prediction results to generate human-understood text is essential to the usability of this as a tool.

The model that will be used to help achieve this goal is a fine-tuned T5 model which will be able to provide a play-by-play style outcome based on the inputted natural language information.

The main contributions from this project have been the knowledge gained about text generation and the nuances and struggles that come with it. In addition, data transformation and labeling were a big component of this project due to the fact that a natural language input did not exist for the data.

To address the fact that this dataset is widely available, as discussed with the professor, the addition of the BERT model was added to add an extra layer of innovation. Though, none of the research I could find used a T5 model, the innovation for the problem is in the way I define the inputs to the T5 model and how the T5 model is evaluated using the BERT model.

Data (10 points) The dataset used for the entire pipeline process comes from a python library called nfl_data_py. This library contains many datasets that combine the NFL data for play-by-play, rosters, draft picks and even more. For this project the play-by-play dataset is being used. The library allows selection of the data by year and since two models are being trained and the most recent data is the most relevant, the last three years of data was selected. The spread of the data for the three years selected, 2022 - 2024, can be seen in the table below:

The selected features have changed slightly since the project update assignment as the generative models became more understood. Previously, the prediction and natural language generation were separate, but now the prediction is done through the natural language interpretation

Table 1: Dataset Spread by Year and Play Type

Season	Play Type	Percentage	Count
2022	run	∼42.4	15037
2022	pass	∼57.6	20393
			35430
2023	run	∼41.8	14877
2023	pass	∼58.2	20723
			35600
2024	run	∼42.9	15044
2024	pass	∼57.1	20006
			35050
Total:		106,080	

and evaluated with a classification model.

The next step was a large time consuming portion of both creating a natural language input and annotating the data with labels based on specific play outcomes given. An ind depth table of what each of the columns used throughout the data pipeline of this table can be found in the Supplemental Material Table 10. The tables below specifies what columns were used to construct the inputs and outputs for the T5 and BERT models:

Table 2: T5 Model Inputs and Outputs

In	posteam defteam passer_player_name offense_personnel defense_personnel qtr down play_type ydstogo yardline_100 quarter_seconds_remaining no_huddle shotgun
Out	desc passer_player_name

Table 3: BERT Model Inputs and Outputs

In	desc passer_player_name
Out	first_down touchdown yards_gained ydstogo fumble_lost interception down incomplete_pass

For the T5 model, the input variables were converted to natural language by using 1 of around 100 templates that vary the natural language. Potentially even more templates could have been used to make the model even more robust, but with limited time 100 templates was enough to start. The output variables were only the play-by-play description already given in the dataset that details the outcome of the given play. In addition the passer_player_name, the quarterback name, was needed in order to eliminate all the names from the description besides the QB's name. This was done in order to stop the model from learning patterns from names. I initially trained the model without this step and the model would attempt to guess player names and jersey numbers which were factually inaccurate.

For the BERT model used in evaluating the T5 data, the inputs and outputs were slightly different as seen in Table 3. The inputs for the BERT model are essentially the goal output of the T5 model since the BERT model's main purpose is to evaluate the performance of T5 model. The output for the BERT model is a label based on the given columns listed in the table.

The dataset was split so that 50% of the data went to each corresponding model. The data for each model was split 80/10/10 for the train, validation, and test sets respectively. Additionally the 10% allocated for the test set for BERT was split again so that there were two test sets with about 5% of the data. This was done in order to allow one test set to evaluate the performance of the BERT model itself and to let the other test set be used to evaluate the performance of the T5 descriptions using the BERT model. Below is the table listing the count values for each split of the data:

Table 4: Split of the Dataset for Each Model

Model	Split	Percentage	Count
T5	train	∼80	26450
	val	∼10	3306
	test	∼10	3307
			33063
BERT	train	∼80	26451
	val	∼10	3306
	test1	∼5	1653
	test2	∼5	1654
			33064

The following table shows the labels for the

BERT model that were created mutually exclusively, and their distribution within the half of the data for the BERT model.

Table 5: Class Distribution for BERT Model

Label Name	Percentage	Count
yardage_gained	∼34.9	11524
first_down	∼25.4	8414
incomplete_pass	∼12.4	4109
failed_third_ down_conversion	∼11.9	3945
yardage_lost	∼6.3	2076
touchdown	∼3.8	1265
no_gain	∼3.5	1145
turnover	∼1.8	586

These were all of the preprocessing and data splitting steps taken to start to train and evaluate each of these models. Finally, to illustrate an idea of what the each data entry looks like, below, in Table 6 are a few examples. Each data entry has an input_text: the template version of the information before a play begins used as the input for the T5 model, a target_text: the play-by-play description used as the target output for the T5 model and the input for the BERT model, and an outcome_label: the class label described in Table 5 used as the target output for the BERT model. Below is Table 6:

Table 6: Data Point Examples

input_text	**target_text**	**outcome_label**
CAR gets in formation at NO's 66 yardline. 2nd and 5, 11 min left in 2Q. B.Mayfield behind 1 RB, 2 TE, 2 WR. NO: 4 DL, 3 LB, 4 DB. No huddle.	6-B.Mayfield pass incomplete short right to [a player].	incomplete_ pass
CIN at own 39. 2nd and 2, 2 min in 3Q. 1 RB, 2 TE, 2 WR. TEN: 2 DL, 4 LB, 5 DB. Shotgun.	28-J.Mixon up the middle to TEN 37 for 2 yards.	first_ down

Related Work (10 point) In a research paper focused on NFL game outcomes, they studied 9 different machine learning algorithms on 43 classifiers, only achieving an accuracy of 67% with Naive Bayes.[2] This is different from my project as I am focusing on specific plays, but it does give me an indication of where I can expect the accuracy to be for my BERT model. I believe my approach will be better since it incorporates text generation so the outcome can be more interpretable for humans. As well as the fact that my project may be able to perform better by focusing on specific plays rather than the intricacy of a whole entire game.

In another research paper focus on NFL game outcomes, the focus was on how to use Deep Learning to predict games.[3] The deep learning focus of this research is more closely aligned to the NLP focus of my project, as deep learning can and is used with NLP. Again their problem is different in the focus of whole games vs. individual plays, like the previous paper. However, it does have a closer connection to NLP. I believe my approach will be better because of the increased research and use of deep learning since the research was done (2018). Since my project will focus on individual plays, I believe it will have a wider use case then this paper's findings.

A research paper focusing on counting the total points earned by each team in NBA and NFL games using play-by-play results.[4] This paper used GPT-4 to analyze and two different methods of doing so: divide and conquer, and chain of thought. My problem is different from theirs in that I am using the play-by-plays to predict the outcome of the plays, where as they are using it to count the points in a game. However, the large NLP focus of play-by-play data makes this paper the closest one to my project. I believe that my approach will be better because I will be using it to generate a play-byplay description of the outcomes of plays which coaches can then use for on the spot decision making.

Although American football and European football (or soccer) rely on different rules, they are still sport systems that can be boiled down to statistics for prediction. That is what TacticalGPT: Uncovering the Potential of LLMs for Predicting Tactical Decisions in Professional Football did with its study on using TacticalGPT for generating output for Who, What, and Where type prompts. [5] This paper is different in content, but is more similar to what I am trying to generate with responses. However, my problem is much more niche, while theirs is generalized, such as questions like "Where is the

player on the pitch?". Instead, my problem takes in specific inputs about a play and generates an outcome and explanation for the outcome.

My project is more focused on predicting the outcome of play-by-play data and generating an explanation to go along with it, there are other ways to provide a larger NLP component. Within a paper entitled Generating Factually Consistent Sport Highlights Narrations, highlight narrations are generated and explored using a Natural Language Inference model. [6] Again this project is focused on soccer, but I believe it is still applicable in this scenario as any sport would be. This paper is most similar to the outcomes I expect my T5 model to create which are play-by-play like outputs. The difference in these projects mainly rely on the sport being focused on and the model methods being used. What sets my project apart will be the ability to evaluate the T5 model using a BERT model so that when using the T5 model, the accuracy of plays is still maintained.

Methods (50 points) Within this section I will discuss the design decisions made throughout the pipeline process, from data import to the model outputs. The data preprocessing has already been described in the data section, but within this section I will describe the decisions made with more detail. The figure below highlights the data preprocessing done up until the point the data is split:

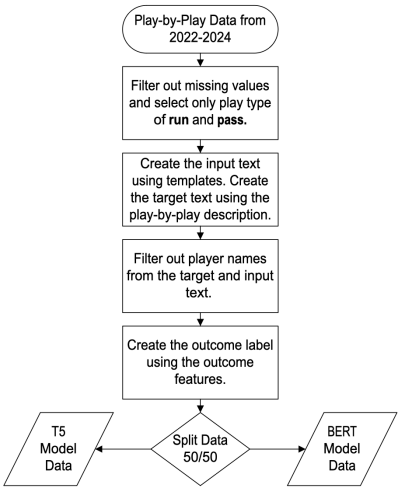

Figure 1: Data Preprocessing

The decision to focus only on run and pass plays was mainly made to narrow the focus of the project. The majority of the plays run in football are these two types of plays, and though there is focus on designing the other special plays within football most of the attention is on run and pass plays.

To create the input, I designed around 100 templates that were chosen at random for each data point. These incorporated all of the variables described in the input section of Table 2. Many templates were needed in order to replicate the varying nature of human language, but it was a tedious task to change how the information was presented. Some of the templates are just slight variations of others. To had more variability, if a data point holds true for the features no_huddle and shotgun, these have separate template sentences then the other variables since they are not always true. This allows for even more variation as there are around 10-15 templates for each of those variables. This increases the variability for each input statement.

Filtering out other player names was an essential step to ensure accuracy for the BERT model. When the T5 model outputs its predicted outcome, originally with these names, it guessed both names and jersey numbers making factually incorrect statements. To fix this issue, removing all names besides the quarterback was essential in ensuring accuracy.

Finally, the outcome label was created for the BERT model based on the play-by-play description given in the dataset. The input features given in Table 3 were used to construct this label, each label is mutually exclusive and was chosen to divide the most common outcomes in football. The data entry is filtered through a series of if/elif statements to ensure that only one label is used. One important distinction to note is that yardage_gained is the last elif statement ran. This means that this label specifies a play in which yardage is gained but there is no first down or touchdown. Another note is that turnover includes both interceptions and fumbles in an attempt to try and make the class sizes closer in distribution.

Once the data was split, they were each sent to their respective pipelines. Within these pipelines the data was split further, following the standard split of 80% of the data for training, and 10% for validation and testing. The models then follow the normal pipeline for natural language processing, first tokenizing the data, training and finally evaluating. The T5 model uses the t5-small tokenizer and model, whereas the BERT model uses BERT base uncased for sequence classification. The T5

pipeline can be seen in Figure 2 which is below.

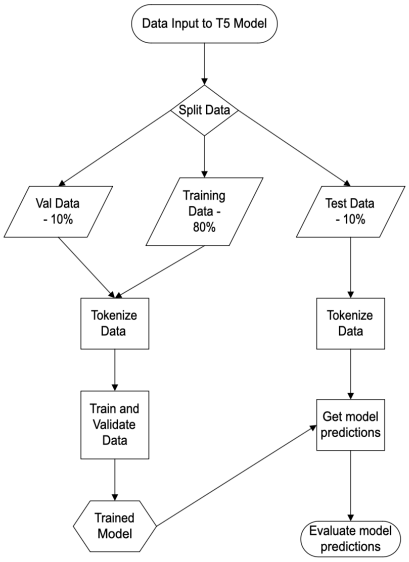

Figure 2: Model Pipeline for T5 Model

The biggest difference between the two pipelines is that the BERT model has two testing sets, its given 10% being split in half. With one half the model will be evaluated for its individual performance, and with the other it will be used to validate the accuracy of the T5 model. The BERT model will use the target text generated by the T5 model in order to classify the target text with an outcome label. If this label matches the outcome label assigned to the data entry then the T5 model output will be considered good. This relies on the BERT model itself being accurate, which is one of the biggest struggles from this project. The BERT pipeline can be seen in Figure 3 which is below:

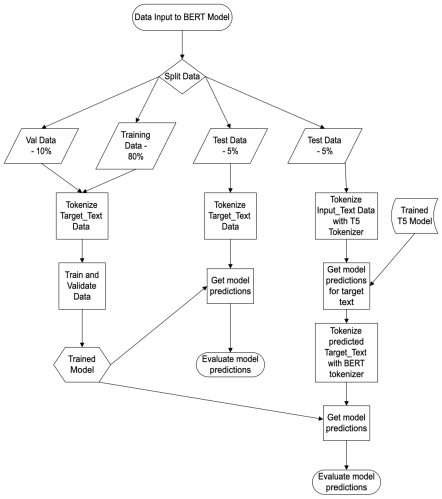

Figure 3: Model Pipeline for BERT Model

The libraries used for the whole process were torch, numpy, pandas, sklearn, transformers, datasets, evaluate, wandb, random, re, and nfl_data_py. In addition to this I used a couple different resources to further understand how both BERT and T5 models worked. [references]

The decision to use the T5 model instead of the GPT3 model I hoped to use when planning this project, was mainly due to time and cost constraints. The T5 model trained much faster on my computer and the encoder-decoder was a similar structure to what I was used to working with. This allowed me to move along with the project faster, as the initial data transformation took up a large portion of my time. The structure of this model made more sense for the way my data ended up being structured, as well, having a rigidly structured play-by-play output. This model was also discussed in class in a later lecture and caught my attention, so I wanted to learn even more about it. [7-8]

The decision to use the BERT model in addition to other evaluation metrics for the T5 model was due to the fact that I could not confirm accuracy on my own just by reading through the thousands of data entries. Since BERT is a bidirectional encoder it would be able to detect the accuracy of whether or not the T5 target text output was still giving the correct general outcome answer. The BLEU score and ROGUE alone could not explain accuracy. [9-10]

Evaluation and Results (30 points) For the T5 model the data was evaluated using the BLEU and ROUGE score. [11] The BLEU score measures the n-grams and precision of the generated text where a score of 0.4 or higher is considered very good. The ROUGE score has three different measures: ROUGE-1, ROUGE-2, and ROUGE-L. The ROUGE-1 measures the overlap of the unigrams between the target text and the generated text, with a score of 0.4 or higher considered good. ROUGE-2 is similar to ROUGE-1 in that it measures the overlap of the bigrams between the target text and generated text, where a score above 0.2 is considered good. Finally, the ROUGE-L measures the length for the longest common subsequence between the target text and the generated text. A score of 0.3 or higher is considered good for the ROUGE-L.

For the BERT model the data was evaluated using the F1 score and the accuracy. [12] Finally

when using the BERT model to check the validity of T5 generated text, the F1 score and accuracy were looked at. The accuracy measures the count of correctly classified labels over the entire set, a value higher than 0.8 in this metric is considered good for multi-class classification. The F1 score measures the balance between the model's precision and recall, where a value above 0.7 is considered good for multi-class classification.

Starting with the T5 model, two baselines were constructed to compare against for the evaluation metrics. The first baseline I used was shuffling the input_text's corresponding target_text. This allowed me to see whether or not the generated text was highly related to the input text enough. This baseline should yield bad ROGUE scores as they match unigrams and bigrams. The second baseline I used was a templated output to see if the model's generated responses outperform a template based on the output values.

For the BERT model, again, two baselines were constructed to compare the model performance against pure randomness or using the majority class label. The first baseline, focused on randomness, assigns a random class label from the eight given classes to the target text. The second baseline assigns the majority class label, yardage_gained, to all inputs. The crossover of using BERT to predict T5 will be compared against these BERT baselines.

After performing baseline analysis, I trained the models and measured their performance in the respective metrics for each model. All of the performance metrics can be found in Table 7 and 8 below.

Table 7: Evaluation Metrics for T5 Model

Method	BLEU	R-1	R-2	R-L
Baseline One	0.0897	0.2870	0.0907	0.2766
Baseline Two	0.0000	0.1182	0.0040	0.1159
Trained T5 Model	0.3225	0.5244	0.3092	0.5160

Based on the above metrics, the T5 model was more successful in attaining better evaluation metrics than the BERT model. No matter the training parameters, the accuracy for the BERT model could not get much higher. Beyond evaluating using baselines I also tested different weight decay values for both models. These can be found in Table 9 below.

Table 8: Evaluation Metrics for BERT Model

Method	Accuracy	F1 Score
Baseline One	0.1264	0.1026
Baseline Two	0.3303	0.0621
Trained BERT Model	0.7732	0.7680
Trained BERT Model with T5 Input	0.2703	0.0751

Table 9: Evaluation Metrics for Changing Hyperparameters

Model	WD Value	Metric	Value
T5	0.01	BLEU	0.3123
T5	0.0001	BLEU	0.3083
BERT	0.01	F1	0.7779
BERT	0.0001	F1	0.7683

Based on the table above I decided on a weight decay of 0.01 for the BERT model and a weight decay of 0.01 for the T5 model. These hyperparameters were used to get the trained model results in Tables 7 and 8.

Discussion (20 points) The T5 model's evaluation metrics, as well as the baseline results, can be seen in Table 7 in the previous section. The BLEU score for both the baselines is far short of the 0.3 threshold for a good score, however the trained model does achieve a score above 0.3. This establishes that the trained model works well when compared to the other methods attempted in the baseline. The performance of the model may be artificially lowered because of the penalty the BLEU score gives for short responses. However, it might be an indication of overfitting since the model still performs pretty well even with this penalty.

The ROUGE-1 scores for the baselines are pretty high, higher than what might be expected for a baseline. However, since this score measures the overlap of unigrams it is not surprising that this metric is pretty high since many of the words needed for the generation are given in the input. The ROUGE-1 score for the final trained

model is above the 0.4 threshold which indicates good performance for the model. Again, this is not unexpected because this score just measures the unigram values.

The ROUGE-2 scores for the baselines are very, very low which is to be expected given the baselines chosen. The overlap of two words between the target and input text would be way lower when shuffling the features corresponding target text and when just using a vague explanation. The ROUGE-2 score for the final trained model is above the 0.2 threshold which indicates good performance by the model. This is expected because the model learned the same formatting structure as the target text and is likely to learn the output format. However, this may be artificially inflated due to the terms used like "a player" and the names of the QB which automatically give the generated text better performance as long as it generates them.

Finally for the T5 model, the ROUGE-L scores for the first baseline is not too bad, only 0.03 away from 0.3. However, the second baseline performs much worse. Since the this score measures the longest common subsequence it makes since that the first baseline would perform better as the "generated text" used still holds the same structure as the target text. The second baseline follows a completely different structure and so performs worse here. The trained model performs very well with this metric which could be an indication of overfitting, or an indication that the generated text learned the output format very well. Since it is a structured output the high ROUGE-L score could be because the generated output follows that structure very closely.

Overall the T5 model baselines perform much worse than the trained model, all below the thresholds considered good for each of the metrics. Whereas the final trained T5 model performs well above the thresholds for these metrics, giving it good performance overall.

The BERT model's evaluation metrics, as well as the baseline results can be seen in Table 8 in the previous section. The accuracy for the first baseline is much lower than the threshold 0.8. This is expected because a random class was chosen so an accuracy of about 0.125 would reflect the random chance of getting one class out of eight which is around where the baseline performs. The second baseline performs twice as well for accuracy, but this is artificially inflated because the majority class label was chosen for all test values. The majority class takes up about 34.9 percent of the data so an accuracy around there is what would be expected. The final trained BERT model achieves a higher accuracy then the baselines, however the accuracy is not greater than 0.8. Though the metric is close to being in the good range it falls just short. This is potentially expected because of the imbalanced class sizes within the outcome labels. The largest class accounts for 34% of the data whereas the smallest only holds 1.8% of the data. Because of this I still think the model performs relatively well given the data that it does have access too.

The final metric is the F1 score for the BERT model and baselines. Both of the baselines performed poorly within the F1 score, and the second baseline performed worse in F1 than the first despite its higher accuracy. This is unsurprising because the F1 will be penalized even more when one class is chosen for every feature text given, than if just any class was chosen like for the first baseline. The F1 score for the final model is above the threshold which indicates good performance which is slightly unexpected given the low accuracy score. However, I believe the higher F1 score may be due to the fact that the model is learning the classification task. The better performance for F1 also indicates that even though the model may not be insanely accurate it performs better with recall and precision.

The last performance to look at is the final row of Table 8, which is how accurate the T5 model is at predicting outcome based on the BERT model. Both the accuracy and the F1 score are insanely low here, but noteworthy that both metrics perform better than at least one of the baselines. I believe that the bad performance here is mainly due to the lower performance of the BERT model in general. Applying the BERT model to evaluate the T5 generated text may be too much for an already weak performing model which is to be expected. However, the accuracy of the T5 model is at least higher than random guessing class labels at about 0.125, which indicates that there could be room for improvement.

Overall, the main methods I used for the T5 model were much more successful than those used for the BERT model. The process is not yet to the point where it would be satisfactory to an end user, some next steps discussed in later sections may

help improve the models to get to that point. The performance of almost all of the models was way better with respect to the baseline which shows that they were learning something for the task. This was expected because play by play data has been used before for text generation, however with GPT models. I believe that my approach to the T5 model worked well, but could have been improved with even more input templates or by using more of the feature columns given in the dataset. The approach for the BERT model I believe is a good idea, however it did not work well within my current set up. This could be due to the way I labeled the data, but is most likely due to the large class imbalance. An improvement may be to augment the data to artificially inflate the smaller classes, or to redefine the labels in a way that provides a more even distribution.

Conclusion (5 points) The main goal of this project was to solve the difficulty of play calling by providing a natural language response to inputted play data. Using a T5 model to generate these predictions based on a select few feature input columns and evaluating this model using a BERT classification for accuracy of the generated data. I believe that this goal of the project was successful in that natural language responses in the form of play-by-play analysis were able to be generated with high accuracy via the BLEU and ROUGE scores. Where the project can be improved upon is in refining the use of the BERT model to define the accuracy of the T5 model. This portion of the project could potentially be used to give a more simple outcome other than an explanation via play-by-play. But in its current form the accuracy and F1 score are not high enough for that to be feasible.

The conclusions drawn in the previous section show that there is room for improvement upon this project and that with more fine-tuning, additional input templates, and improved classes for the BERT the performance can improve. The evaluation and discussion show that there is viability for a solution like the one proposed in this paper, however more research needs to be done into the best methods. The methods described here could be improved upon or taken another direction by fine-tuning different models. Whatever direction taken, this paper proves that there is a reason to look further into the topic and provide better data-fueled tools to the coaches and anyone else within the NFL.

Other Things We Tried (10 point) A lot of the time that I spent on this project that did not end up being utilized were in the early preprocessing steps. Because I wanted to focus on predicting play call data, I only had one half of the natural language data I needed. I spent a lot of time trying to decide what metrics within the dataset were the most useful, and how to represent the column values given, most of which were floats, strings, and boolean, as a natural language representation for the model. Ultimately I settled on the template format but I did train iterations of the T5 model that were not effective using only one template or just a few. I finally realized I needed to use even more templates because this was the only thing the model was seeing.

I also initially attempted to use the data directly as the categorical values just listed in a string, such as (Down: 3, Quarter:2, ...). However, this method wasn't reflective of my goal to use this model technique for coaches to get the outcome of a play as I wanted the input to be reflective of what they or commentators might say.

Within both of these processes I tried to set up the data for fine-tuning with GPT-3, however I was unable to get the model to fine-tune, constantly getting errors. [13-14] Since I wasn't able to debug past these issues I had to shift my focus to a T5 model, and added the BERT model for additional design, complexity, and performance improvement. I also believed it would be an innovative way to see how well the T5 model was performing without having to use human evaluation.

What You Would Have Done Differently or Next (10 point) One of the biggest things I would have done differently for this project was finding a way to widen the scope even more. Due to time constraints I had to focus on specific play types and the features I chose, but with more time I would have loved to explore how different features given in the original input text to the T5 model would have effected performance. Based on the related works I looked at, and my original idea for the project some general next steps would be to use the GPT3 model I originally wanted to use. I think that overall my project went well but I was originally a bit too ambitious with my idea, especially never working with generative tools before this class. This hurt me in the long run as I had

to aggressively narrow the scope of the project to something that was more attainable.

Acknowledgments I discussed potential focus areas with my parents and friends and ended up focusing on the NFL. However, the problem formulation and the NLP focus was my own idea. The other references I have provided past the research papers, [7-12], were used to help make decisions on evaluation metrics and how to use T5 and BERT models which I didn't have much experience with. This is in addition to minor debugging using StackOverflow which is not listed in the references section.

References

[1] nflverse, "nfl_data_py," *GitHub*, [Online]. Available: https://github.com/nflverse/nfl_data_py.

[2] R. Beal, T. J. Norman, and S. D. Ramchurn, "A critical comparison of machine learning classifiers to predict match outcomes in the NFL," *International Journal of Computer Science in Sport*, vol. 19, no. 2, pp. 36–50, Dec. 2020, doi: 10.2478/ijcss-2020-0009.

[3] P. Bosch, "Predicting the winner of NFL games using Machine and Deep Learning," *Vrije Universiteit, Amsterdam*, 2018.

[4] Y. Hu, K. Song, S. Cho, X. Wang, H. Foroosh, D. Yu, and F. Liu, "Can Large Language Models do Analytical Reasoning?," *arXiv preprint arXiv:2403.04031*, 2024.

[5] M. Clark, J. Hodge, and D. Horton, "TacticalGPT: Uncovering the Potential of LLMs for Predicting Tactical Decisions in Professional Football," *StatsBomb*, Oct. 2023. [Online]. Available: https://statsbomb.com/wp-content/uploads/2023/10/TacticalGPT-Uncovering-the-Potential-of-LLMs-for-Predicting-Tactical-Decisions-in-Professional-Football.pdf.

[6] Z. Zhang, A. Anantharaman, and B. J. Yuan, "ExplainMyMove: Towards Real-Time Natural Language Explanations of Agent Behavior in Multi-Agent Sports Games," *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, Oct. 2023. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3606038.3616157.

[7] NLPlanet, "A Full Guide to Finetuning T5 for Text2Text and Building a Demo with Streamlit," *Medium*, Sep. 2022. [Online]. Available: https://medium.com/nlplanet/a-full-guide-to-finetuning-t5-for-text2text-and-building-a-demo-with-streamlit-c72009631887.

[8] LearnOpenCV, "Fine-Tuning T5," *LearnOpenCV*, Feb. 2023. [Online]. Available: https://learnopencv.com/fine-tuning-t5/.

[9] TensorFlow, "Fine-tune BERT," *TensorFlow*, [Online]. Available: https://www.tensorflow.org/tfmodels/nlp/fine_tune_bert.

[10] La Javaness, "Multiclass and Multilabel Text Classification in One BERT Model," *Medium*, [Online]. Available: https://lajavaness.medium.com/multiclass-and-multilabel-text-classification-in-one-bert-model-95c54aab59dc.

[11] Google Cloud, "Automated Machine Learning - BLEU Score Evaluation," *Google Cloud*, 2023. [Online]. Available: https://cloud.google.com/translate/docs/advanced/automl-evaluate. [Accessed: Mar. 27, 2023].

[12] KDnuggets, "Micro, Macro, and Weighted Averages of F1 Score: Clearly Explained," *KDnuggets*, Jan. 2023. [Online]. Available: https://www.kdnuggets.com/2023/01/micro-macro-weighted-averages-f1-score-clearly-explained.html.

[13] DataCamp, "Fine-Tuning GPT-3 Using the OpenAI API and Python," *DataCamp*, Apr. 2023. [Online]. Available: https://www.datacamp.com/tutorial/fine-tuning-gpt-3-using-the-open-ai-api-and-python.

[14] A. Author, "Multi-label Classification: Do Hamming Loss and Subset Accuracy Tell the Whole Story?," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2011.07805.

A Supplemental Material

Name	Type	Desc.
posteam	string	Three letter team name for the current offense.
defteam	string	Three letter team name for the current defense.
qtr	float	Indicates current quarter.
down	float	Indicates current down.
play_type	string	Indicates the play type the offense wants to run.
ydstogo	float	Yards needed for a first down.
yardline_100	float	Yards needed for a touchdown.
offense_ personnel	string	Describes the offense positions on the field.
defense_ personnel	string	Describes the defense positions on the field.
passer_ player_name	string	The QB name, if it is a pass play type.
quarter_ seconds_ remaining	float	The seconds left in the current quarter.
no_huddle	bool	If there was no huddle before the start of the play.
shotgun	bool	Whether the QB is in shot gun.
desc	string	The play by play description.
first_down	bool	If there was a first down.
touchdown	bool	If there was a touchdown.
yards_gained	float	The number of yards gained from the play ran.
fumble_lost	bool	If there was a fumble that the offense lost.
interception	bool	If there was an interception.
incomplete_ pass	bool	If there was incomplete pass.

Table 10: Data needed for preprocessing.