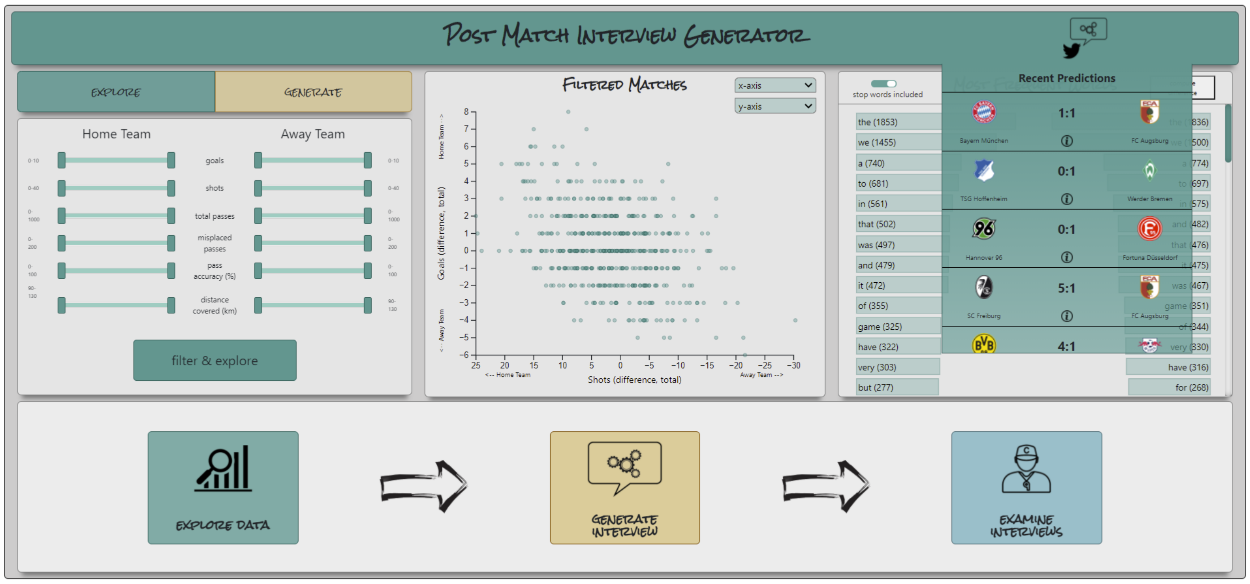


Figure 1: Latest beta version of the dashboard. The figure shows the explore view, where users can filter matches to explore interviews and most frequent words used for these filter settings. This enables the user to get a better understanding of the words that, given a particular score line, the model might pay attention to. Users then move on to generate interviews in the generator view, cf. fig. 6.

Generating Post-Match Interviews for Soccer Games

Harvard CS 109b – Spring 2021

1 Task Description

1.1 Overview

The goal of this project is to predict post-match soccer interviews based on a set of given match stats with the intention to generate meaningful text reflecting these stats ideally in such a way that the generated interview is not distinguishable from an actual interview. What is so particular about the selected type of genre is that the coaches not only use a very distinct jargon but also that those interviews follow rather predictable patterns. A coach's answer always reflects the match statistics in some form - if they have won, a coach says that they're happy with the dominant performance and vice versa if they have lost.

Although text generation is considered one of the hardest tasks in natural language processing, we are hoping to generate reasonable i.e. meaningful post match interviews combining the power of strong, pre-trained language models and the insights of detailed match statistics. Given data about a game, such as the goals scored and shots taken by the home and away teams, and many others, we hope to generate an interview of a few sentences that the team coaches might say after the match.

1.2 Usability and Contributions

Generally speaking, our project tries to provide an example of conditional text generation in narrow use-cases that can be supported with qualitative data.

The main contribution of this project is the model itself that predicts post match interviews given a set of match stats. Just this model by itself could contribute strongly in the field of soccer video games. Many of these games are currently using a set of very few, hard-coded interviews. Our model, that is trained on thousands of real interviews can help to generate more realistic and more varied interviews for these games.

The second contribution of this project is the development of a web-app with a powerful dashboard that will provide detailed insights for domain experts. This includes, young coaches, public relation departments of soccer clubs, and even journalists. The web-app will not only allow coaches to search through thousands of interviews but also to enter match statistics and some further parameters such as sentiment to generate an interview corresponding to these inputs. Young coaches can use this to see some options for what to say, PR departments can advise club officials and journalists can anticipate interviews and prepare questions accordingly.

2 Data

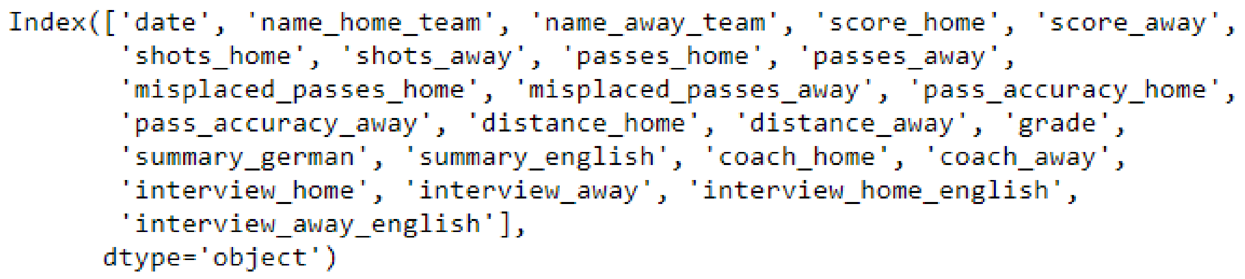
For this project, two different types of data are needed - the statistics of the games as well as the corresponding post-match interviews.

2.1 Data Collection

Since the data needed for this project has not been curated or even gathered, we wrote a few web scraper scripts that would help us to collect the information needed. The scripts can be found both in the supplementary material as well as in our GitHub repository [2]. As many post-match interviews are only available as videos (in particular for the English Premiere League), we resorted to the German website www.kicker.de that as both match statistics as well as the corresponding post-match interviews. That said, all these interviews are in German while we were planning to perform our text generation in English to increase our target audience. Thus, we wrote an additional script to translate all interviews into English using the Google Cloud Translation API. While a random sample-based check showed that the quality of these translations are excellent and grammatically flawless, some of the soccer jargon got lost in translation, which might have an effect on the final model.

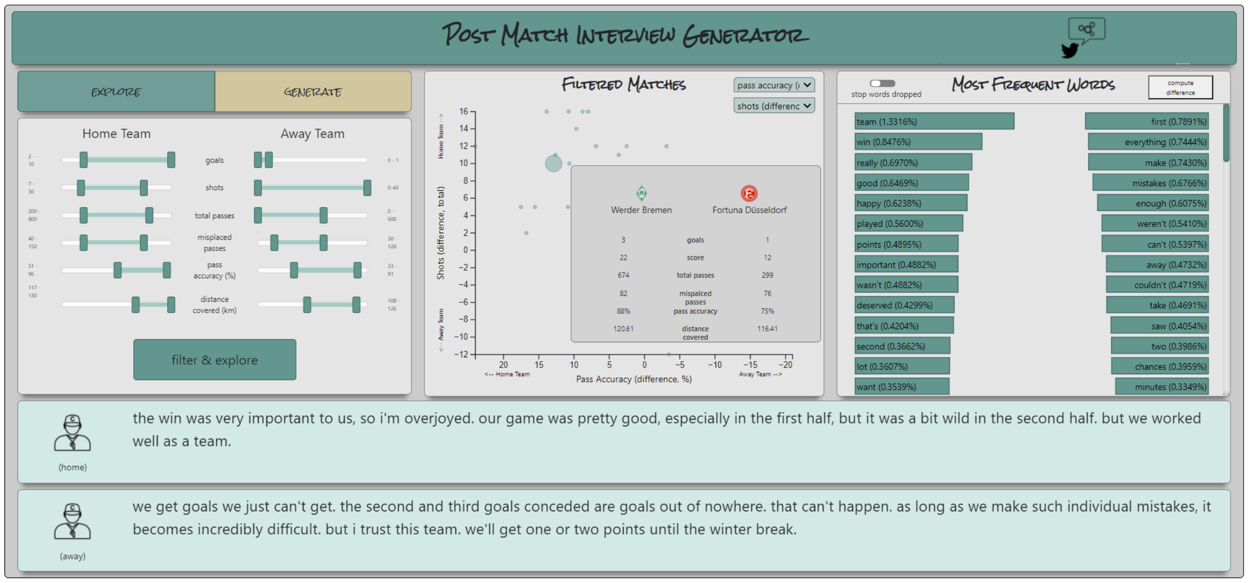
2.2 The Data Available

With these general remarks in mind, let's look at the actual numbers. We have a total of 10 seasons worth of data which is $306 \times 10 = 3060$ matches. For all of these matches we have all the predefined statistical measurements in figure 2. That said, only for 3 of those seasons, we have the full post-match interviews for both home and away coach, which is $306 \times 3 \times 2 = 1836$ interviews.



```
Index(['date', 'name_home_team', 'name_away_team', 'score_home', 'score_away',
       'shots_home', 'shots_away', 'passes_home', 'passes_away',
       'misplaced_passes_home', 'misplaced_passes_away', 'pass_accuracy_home',
       'pass_accuracy_away', 'distance_home', 'distance_away', 'grade',
       'summary_german', 'summary_english', 'coach_home', 'coach_away',
       'interview_home', 'interview_away', 'interview_home_english',
       'interview_away_english'],
      dtype='object')
```

Figure 2: The figure above shows all the columns labels of the final data set, i.e. the statistical information available for each match. All observations have been accumulated using a custom web scraper. The columns *interview_home_english* and *interview_away_english* were generated using a custom translation script based on the Google Cloud Translation API.



Home Team			Away Team
2 - 10	goals		0 - 1
7 - 30	shots		0-40
200 - 800	total passes		0 - 600
40 - 150	misplaced passes		30 - 120
51 - 96	pass accuracy (%)		33 - 91
117 - 130	distance covered (km)		108 - 126

Werder Bremen		Fortuna Düsseldorf
3	goals	1
22	score	12
674	total passes	299
82	mispalced passes	76
88%	pass accuracy	75%
120.61	distance covered	116.41

team (1.3316%)	first (0.7891%)
win (0.8476%)	everything (0.7444%)
really (0.6970%)	make (0.7430%)
good (0.6469%)	mistakes (0.6766%)
happy (0.6238%)	enough (0.6075%)
played (0.5600%)	weren't (0.5410%)
points (0.4895%)	can't (0.5397%)
important (0.4882%)	away (0.4732%)
wasn't (0.4882%)	couldn't (0.4719%)
deserved (0.4299%)	take (0.4691%)
that's (0.4204%)	saw (0.4054%)
second (0.3662%)	two (0.3986%)
lot (0.3607%)	chances (0.3959%)
want (0.3539%)	minutes (0.3349%)

the win was very important to us, so i'm overjoyed. our game was pretty good, especially in the first half, but it was a bit wild in the second half. but we worked well as a team.

we get goals we just can't get. the second and third goals conceded are goals out of nowhere. that can't happen. as long as we make such individual mistakes, it becomes incredibly difficult. but i trust this team. we'll get one or two points until the winter break.

Figure 3: The figure above shows the dashboard's explore view with the filter module (left), the customizable scatter plot encoding all matches (center) as well as the bar chart component (right), that shows the most frequent words for the currently filtered matches. When hovering over a match in the scatter plot, the interviews for the home as well as the away coach will show in the bottom

3 Exploratory Data Analysis

For NLP tasks, it is usually quite hard to perform some meaningful EDA. However, since we wanted to get an overview of our data before building our models and also were hoping to provide some intuitional insights on what our model might ultimately pay attention to, we decided to build an early version of our dashboard, that would help with these tasks. Ultimately, the dashboard will thus, serve two distinct tasks: It will 1) allow domain experts to explore the data and 2) enable users to generate text. The following sections address these two different aspects of the dashboard's architecture.

3.1 The Dashboard's Explore View

Figure 3 shows the dashboard's explore view that consists of three components: an advanced filter module, a customizable scatter plot for the filtered matches, as well as a bar chart component, providing an overview of the most frequent words within the filtered matches. It is this last visualization that provided a lot of meaningful insights: We noticed that, initially, stop words were dominating this frequency charts, which is why we included a possibility to drop stop words from the interviews of the filtered matches. In addition to that, we also included the possibility to compute the difference of word frequencies for a match's opposing coaches, e.g. the difference in relative word frequency a winning coach would use a particular word over the frequency a losing coach would use this very word. Figure **??** shows these differences in relative word frequency for all home wins. While this visualization doesn't work like the transformer's attention mechanism that our GPT-2 based models will use, this provides some insights into which words our model should pay more attention to. In addition, this visualization will, to some extent, also help us to evaluate the generated interviews eventually.
Furthermore, the word frequency bar chart can help to identify some potential problems with regard to our task of building models that can generate meaningful text. To provide an example, we noticed that some meaningful soccer-jargon words appear frequently for both coaches despite a clear and distinct result. If attention-based models predict one of these words as the most likely next word, then our generated text might go on a wrong track, so this is something to be aware and to take into account when developing a suitable model architecture.

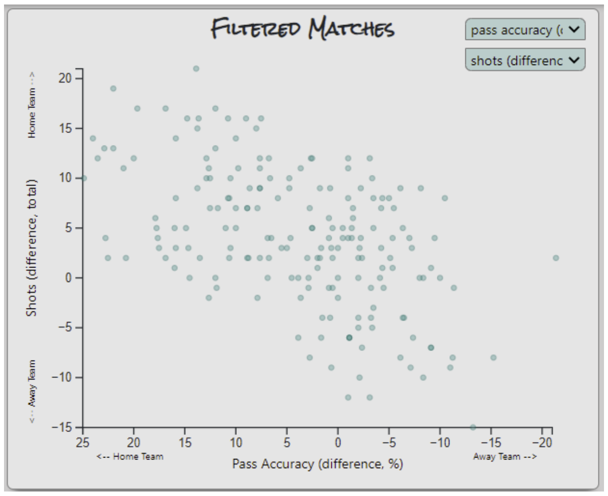


Figure 4: Dashboard Explore View - Scatter Plot — showing filtered matches for 3:1 home victories. Users can customize input data axis to develop a sense both for the data set as well as for the type of interviews that result from these stats. Hovering will display both the match stats as well as the interview in a separate text field. The "Most Frequent Words Bar Chart" 5 is being calculated based on these filtered matches

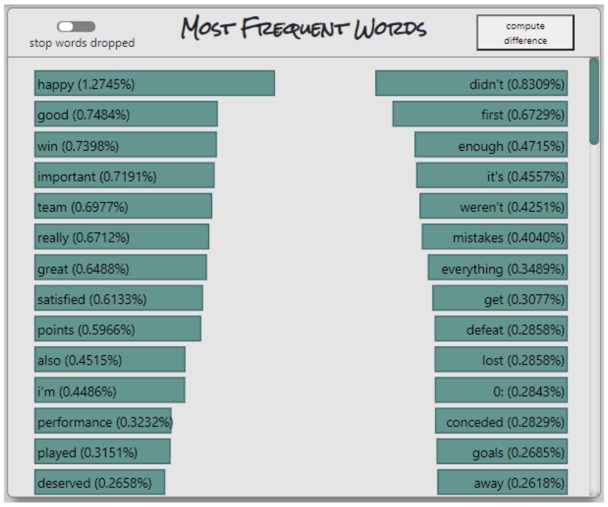


Figure 5: Dashboard Explore View - Most Frequent Words Bar Chart — showing the most frequent words in the current set of filtered matches. Users can drop stop words and calculate the difference in relative word frequency between home and away interviews. This allows to get a sense of the vocabulary a coach is using given a predefined result.

3.2 The Dashboard's Generator View

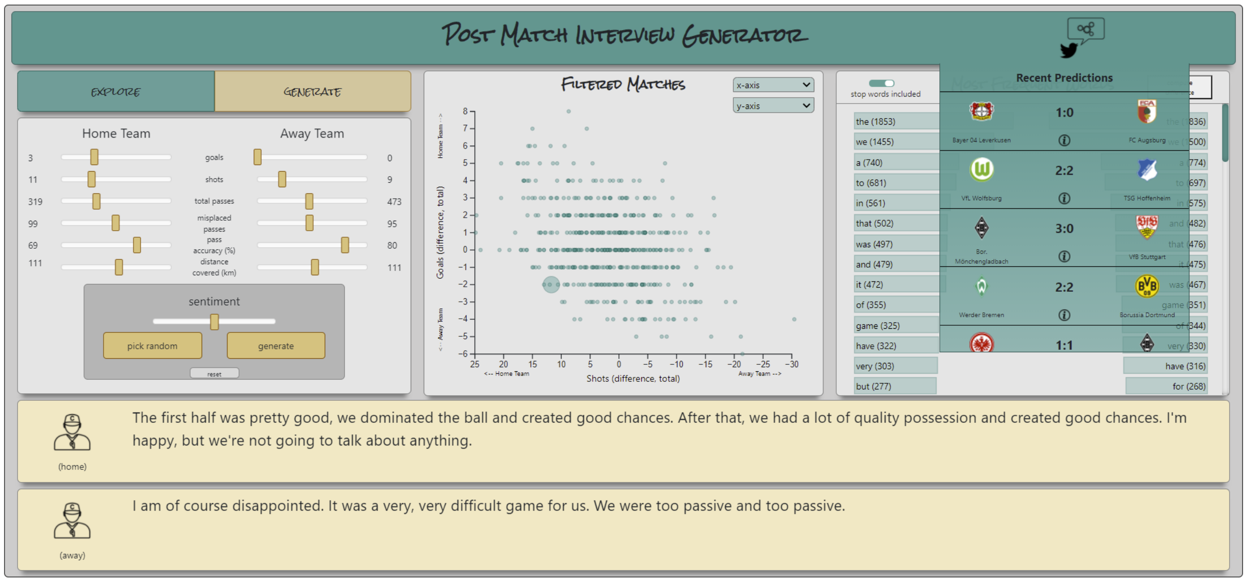
The generator view is the center of the dashboard and the web-app itself. Technically, discussing the generator it s not part of the EDA any, but since it is building directly on the dashboard's explore view, we'll discuss it's features in this section. Figure 6 shows the generator-view which can be accessed by clicking on the *generate* tab next to the explore tab.
The generator allows users to first adjust the sliders (fig. 6, left) to simulate a particular result to then generate both the interview for the home coach as well as the statement of the away coach that would reflect the given match facts.
In the current beta version, the sentiment slider located above the generate button is disabled since we have not performed a sentiment analysis of our scraped data. For the final version of the post match generator, this feature will be included.

4 Modeling

4.1 Baseline Model

For our baseline model, we fine-tune three different GPT-2 models, one for winning, one for drawing, and one for losing coaches, and then for a given match, we generate an interview from the corresponding model. The results sound like good interviews with coherent statements. Check the examples generated from each model:

- **A winning coach:** "Today we're happy with the three points. Overall, it was a deserved victory. We increased the lead to eleven points with a little coaching. I'm very happy with the result and the way the team played. We are very satisfied with this win and are looking forward to the game."
- **A Losing coach:** "Today we found it very difficult. We didn't take advantage of what we got, what we get, we have to put everything that we can on the pitch. It is a deserved victory for Frankfurt."



The first half was pretty good, we dominated the ball and created good chances. After that, we had a lot of quality possession and created good chances. I'm happy, but we're not going to talk about anything.

I am of course disappointed. It was a very, very difficult game for us. We were too passive and too passive.

Figure 6: The figure above shows the dashboard's generator view with component for the generator settings on the left side and the generated texts for the home and away coach on the bottom.

- **A Draw coach** "Today we see a lot of changes and need to improve. We should have won the game for a long time, but the point is worth a lot. We had a very good game with a lot of power and could have won the game for a lot of goals."

Again, we can see that the interviews are consistent with the match result. However, we make the following observations:

- The interviews reference some discussion on the match details. The winning coach said it was a deserved victory and the draw coach said they had deserved the win. But what if in the given match, the teams had done really bad and had gotten the result by luck. We would like our models to capture this case.
- The interviews sometimes mention specific teams, such as "Frankfurt". Such names are unlikely to be consistent with the match we are generating the results for.
- The interviews can mention some season related details, such as the 11 points lead by the winning coach, which might again not be the case for the match we are generating for.

As for the first concern, we aim for our next models to solve this problem. However, for the second concern, we won't address this as we think the user or the application using the plateform can easily switch this to match the team they are generating it for. And for the third concern, we can't address it with the way we generate our data, as it would need the user to provide data about the full season rather then just one match.

So, our main take aways from this is that it is possible to get good interviews, but that we aim to have them be more consistent with the match details.

4.2 Conditional Transformer

To address the concerns mentioned earlier, we want our models to take into consideration all data we have about a match. To do so, we want the data to affect the probabilities of generating words inside the GPT2 model.

Our first approach for this is a conditional GPT 2 model. To build this model, we first fine tune a GPT2 model to all the matches to get it tuned to the languange of the interviews.

Then, we modify the layers of the transformer to have it use the match statistics. The small gpt2 model has some layers of transformers, each one ends with a FFNN. To modify that, after each transformer, we add one more FFNN. This network takes the outputs of the transformer, concatenates it with the match statistics, and then produces new outputs of the same size as the transformer outputs.

To leverage the GPT 2 Model's pretraining, we want to make sure that the added layers don't break the relationships previously found by the layers of transformers. So, we want the models to initially give the same results as the initial GPT 2 model, and then, through fine tuning, it can start finding other relationship with the help of the match statistics. To do so, for all added FFNN layers, we initialize them with an identity matrix for the weights and zeros for the biases. For non square matrices, the identity initializer fills the additional rows with zeros. As a result of this, when we initialize the weights, the FFNNs will be returning exactly their inputs, which means the model will start off with the same results as the GPT2. Then through fine tuning, will start learning the new relationships with the match statistics.

But why are we trying to add the match statistics inbetween the transformer layers? Our intuition is as following:

- For the first few transformer layers, the GPT-2 model tries to understand the previous text in the sentence. But these sentences in the interviews mean different things under different results. Take as an example, the sentence "We played against a strong team". If the team had won the match, then this sentence is showing that the coach is proud of their team and they respect their opponent. But if the team had lost, then this sentence could mean that the coach is finding excuses.
- For the later transformer layers, the GPT-2 model is decoding the information it has into an interview. The match statisics here are helping the model choose the words appropriate for the result.

After training this model, we didn't get good results. Take for example the following text generated using a greedy search for a $2-2$ draw:

- "Today we have to accept defeat. We didn't deserve it today."

The generated text is not at all consistent with the result, and this was true for most examples generated. We think that the main reasons are as following:

- The model is overfitting on the match statistics. We are throwing 14 columns of data on a FFNN with not enough data points. So, for example, it might have generated the defeat statement mentioned earlier based on the number of passes or some other statistics.
- The transformer takes in very small learning rates for training, while usually we would have used a larger learning rate to train an FFNN. In addition, we had to initiailize the match statistics weights with zeros, which is usually bad for FFNNs. These factors might have negatively affect the learning of the model.

We address these concerns in our next approach.

4.3 Categorical GPT 2

As mentioned earlier, the reason the conditional model isn't performing well is that we are throwing a lot of match statistics at it that might not be useful. Instead, let's only put the data that helps.

All of the shots/passes/passes missed only help with one thing: was the team dominanting the match, and so deserved the win or didn't deserve the loss, or were they the weaker side and possibly won by luck or deserved the loss? So, let's actually only use these labels. We create those labels and then feed them into the GPT 2 model as special tokens at the beginning of the text. Then, to generate an interview for a match, we create its labels, feed them as again into the gpt2 model, and have it generate an interview for us. This shouldn't overfit, as we use a small summary of our data. And it only involves using the general structure of GPT 2, so we shouldn't go into any learning problems.

So, given the match statistics, we create two labels:

- **The expected result:** We build a logistical regression model that predict the probability of winning given the statistics of the game (shots, passes...). The probability of win tells us who had more control and dominated the match and who didn't. So, we then divide that into five labels [”dominant loss”, ”regular loss”, ”tie”, ”regular win”, ”dominant win”] corresponding to the probability intervals: [0, 0.2], [0.2, 0.4], [0.4, 0.6], [0.6, 0.8], [0.8, 1].
- **The actual result:** Based on the goal difference, we create the same labels [”dominant loss”, ”regular loss”, ”tie”, ”regular win”, ”dominant win”], based on the goal differences [Loss by more than two goals, loss by one or two goals, tie, win by one or two goals, win by more than two goals].

Then for the GPT 2 model, we create special tokens for these labels and feed them as input ids to the model.

The results look very good. Take the following example generated for an actual result of a regular win, and an expected result of a dominant loss:

- “”It was a very difficult game, we didn't have the punch, we played too slowly and too slowly. I'm very happy that we won the opening game against a very strong opponent.””

We see that the interview is consistent with both the expected and actual result. As the coach acknowledged that it was a difficult game and that they are happy they won. This was the case of many generated examples.

5 Evaluation

Our goal for the models is to generate interviews that work well with the given match inputs (match statistics and other interview parameters). A main challenge of this is that there isn't any clear way to automate testing the outputs. Rather, we have to manually generate a few examples and evaluate how good they are, based on local syntax, coherence through the full interview, and consistency with the match inputs.
The general evaluation is summarised in the following table:

Model:	Local Syntax	Coherence through the full interview	Consistency with the match statistics
Baseline	Looks Good	Looks good	Only consistent with the final result (win/lose/tie)
Conditional Transformer	Looks good	Not coherent	Isn't consistent with the statistics
Categorical GPT 2	Looks good	Looks good	Consistent with the final result and the expected result

6 Final Thoughts, Prospect, and Next Steps

In this project, we built a complete post-match interview generator. We first collected the data for the interviews and matches, created a dashboard to explore the data and generate new interviews, and we created models that can generate new interviews. Additionally, we've provided a supplementary video of our project explaining the architecture of our model and demonstrating the core features of our dashboard.[1]

While our final model is already performing well, we have a variety of ideas to improve future model performance. Below, we're listing the most important next steps that we plan to take in order to enhance model performance.

- Since coaches are often use the half-time score to talk about what was good and bad and what changes were made, we plan to scrape half-time scores for our data to improve text generation for these particular statements.

- Related to that the half-time score lines, we plan to replace score lines in the text with tokens such as [half-time-score] or [final-score] that we could replace again once the text has been generated with the actual scores to provide generated text that is also authentic text when it comes to score lines, which our model is currently lacking.
- In similar fashion, we also plan to replace opponent team names with a token such as [opponent-name] to then replace that token after the text has been generated.
- Furthermore, we plan to use betting odds to reflect prior expectations, which could serve as another label for our model.
- Lastly, we plan to include a page to our dashboard where users can submit stats and interviews to crowd source more data that could help us to increase the training data.

References

[1] Masaoud Haidar and Robert Roessler. Cs109b - post match interview generator. `https://www.youtube.com/watch?v=dyzQFnYt5ig&ab_channel=RobertRoessler`, May 2021.

[2] Masaoud Haidar and Robert Roessler. Post-match generator. github repository. `https://github.com/rob3rtroessler/post-match-predictor`, May 2021.