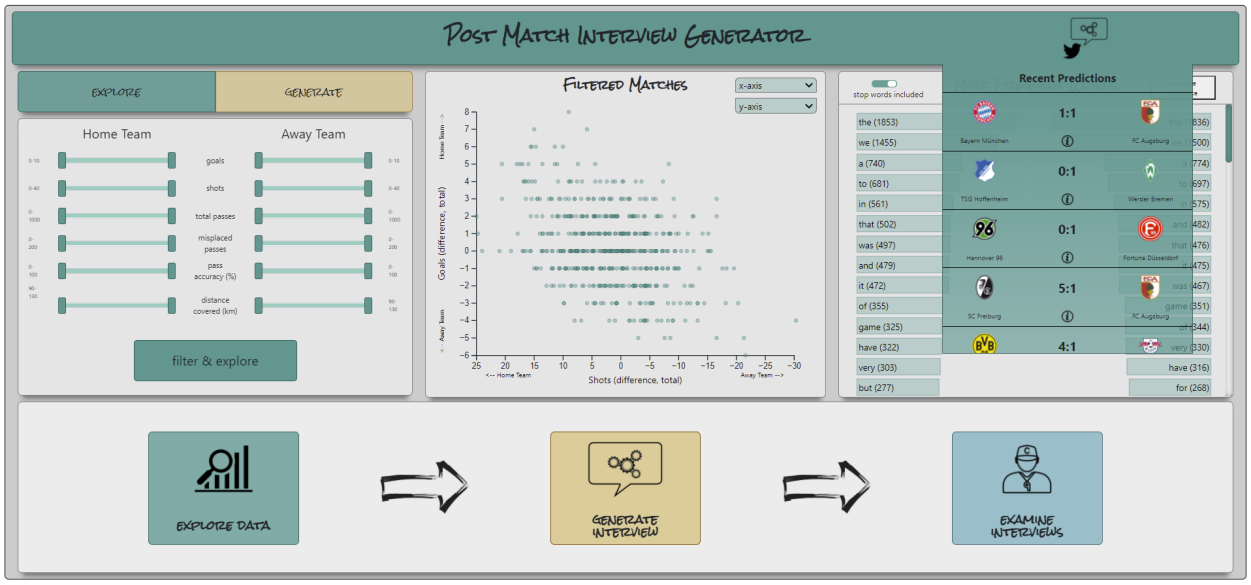

Figure 1: Latest beta version of the dashboard. The figure shows the explore view, where users can filter matches to explore interviews and most frequent words used for these filter settings. This enables the user to get a better understanding of the words that, given a particular score line, the model 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 and, by that, contribute to the research field of computational text generation. 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

Generally speaking, our project tries to provide an example of conditional text generation in narrow use-cases that can be supported with qualitative data. This models that we plan to develop could be used in other domains with comparable conditions.

More specifically, one use case for this project is for young or inexperienced coaches who aren't sure of how to deal with the press after matches. When losing or even when winning, if the coaches don't provide strong interviews, they might become unpopular with the press. Over time, this could lead the press to put a lot of pressure on the coach when the team loses, leading teams to sack their coaches after some bad results.

As part of this project we aim to create a web app. This 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.

Another use case for this project is for soccer video games. In many of these video games, the game provides post-match interviews. Currently, there are only a few fixed possibilities for the interviews and they repeat after each match. Training a model on real interviews can help generate more realistic and more varied interviews for these games.

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 [1]. 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.

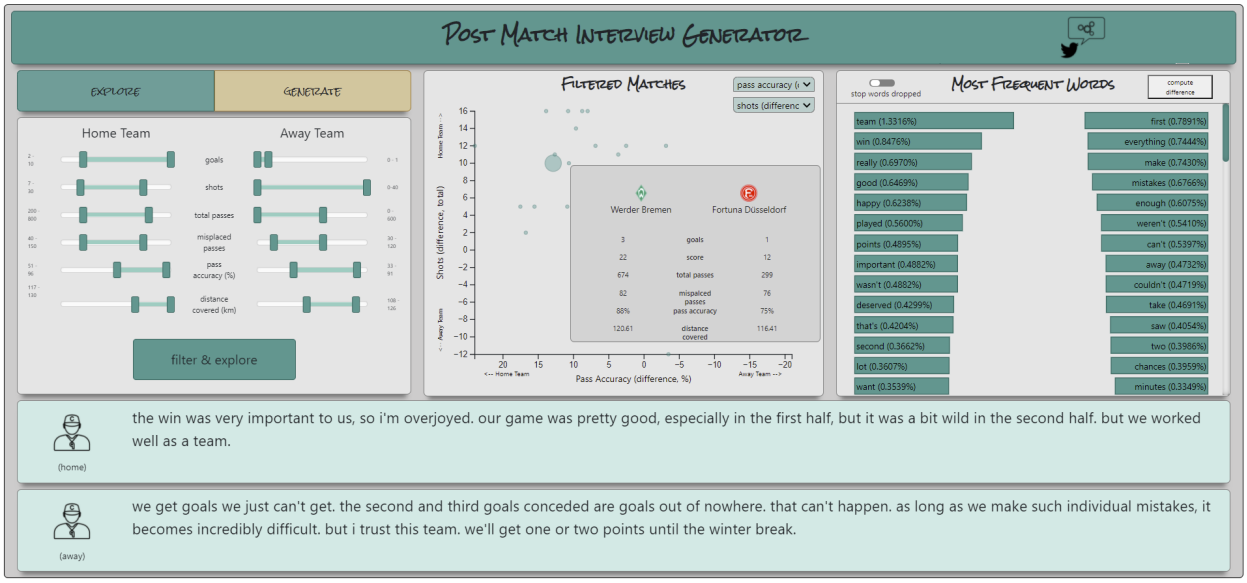

Figure 3: The figure above shows the dashboard's explore view with the filter module (left), the customizable scatter plot encoding all matches (center) as well as the bar chart component (right), that shows the most frequent words for the currently filtered matches. When hovering over a match in the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

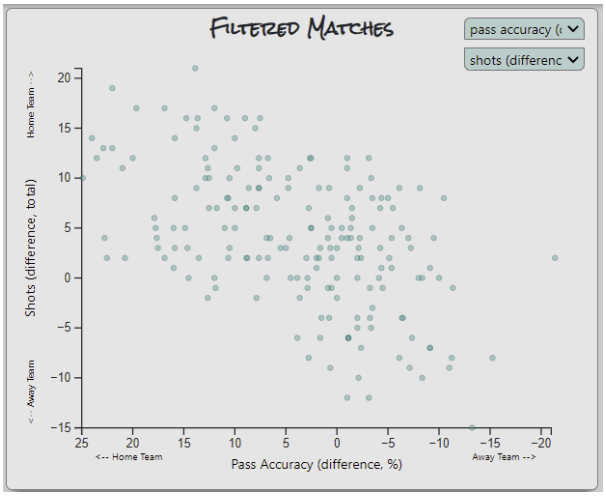

Figure 4: Dashboard Explore View - Scatter Plot — showing filtered matches for 3:1 home victories. Users can customize input data axis to develop a sense both for the data set as well as for the type of interviews that result from these stats. Hovering will display both the match stats as well as the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

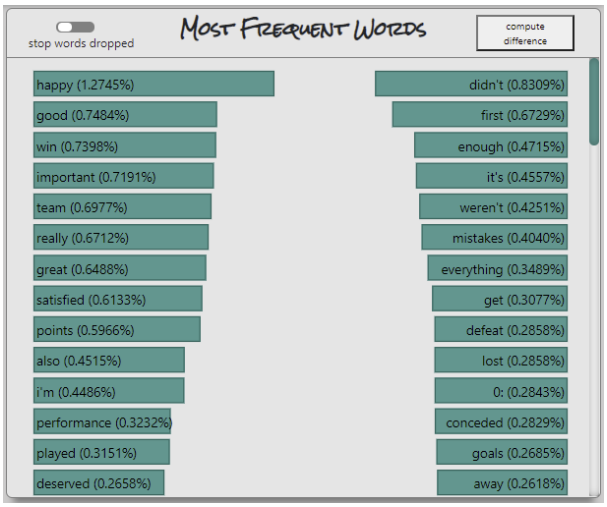

Figure 5: Dashboard Explore View - Most Frequent Words Bar Chart — showing the most frequent words in the current set of filtered matches. Users can drop stop words and calculate the difference in relative word frequency between home and away interviews. This allows to get a sense of the vocabulary a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

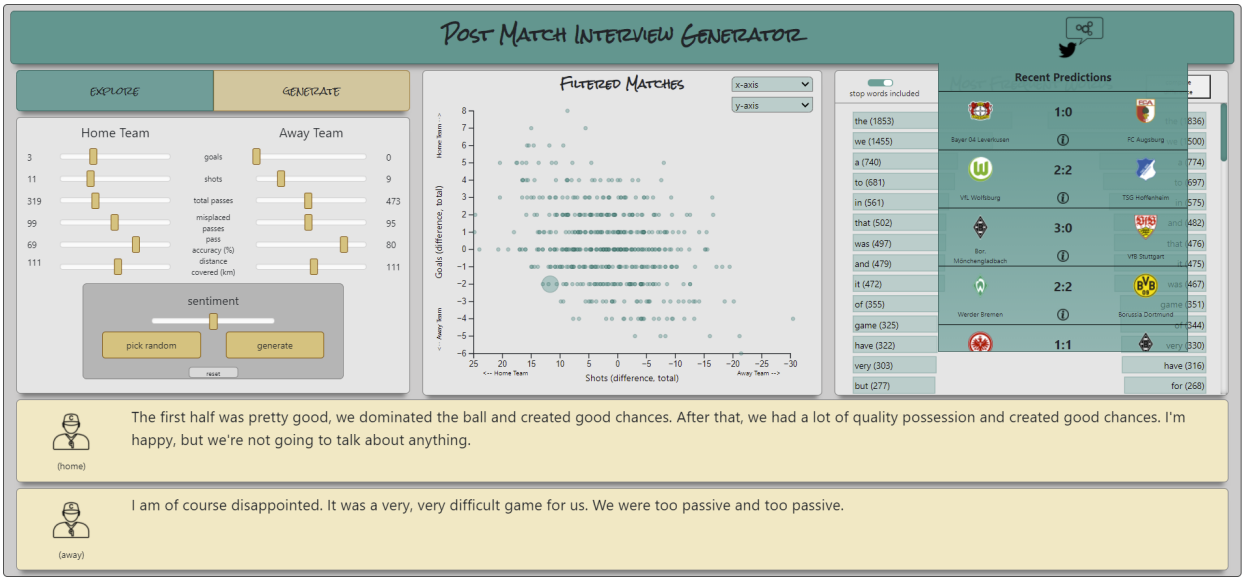

Figure 6: The figure above shows the dashboard's generator view with component for the generator settings on the left side and the generated texts for the home and away coach on the bottom.

- **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 don't

4.2 Next Steps: Wrapper GPT-2 Model

For this model, we train a GPT-2 model that can wrap interview statements around some given keywords [2]. We also train another model such as an FFNN, that given the match inputs (statistics and other parameters), gives back the probabilities for each word to be in a statement about this match. Then, to generate an interview for a match, we feed the match inputs to the second model to get the word probabilities. Then, we sample from the top words. Then, we feed these words to the GPT-2 model to wrap a statement around these words.

4.3 Main Goal: Conditional GPT-2 Model

Our main goal is to train a conditional GPT-2 Model. First, we fine tune a GPT-2 model on the interviews. Then, we add one more layer to the model. This layer takes as an input:

- The outputs from the GPT-2 model, which are the logits of each possible next word.
- The match inputs, after possibly going though a Dense layer.

The output of this final layer would be the logits of every possible next word. The idea of this model is that it will take the logits coming from the GPT-2 model and then modify them according to the match statistics. Then, to generate an interview for a match, we feed the match inputs and an initial word to the model. Then, we start choosing every following word by sampling from the words with the top logits, and them feeding the model with the chosen words, the same way a regular GPT-2 model generates text.

4.4 More Exploration: Seq-2-Seq Model

A disadvantage for the previous models and the data we have is that they only deal with the final results of the matches. The coaches could talk about specific events that happened in the match, such as an important last minute goal or about some missed chances early in the game.

To explore more, we could obtain data about the events of every match. Then, we train a seq2seq model that is "translating" match events to English interview statements. We are not sure how such a model would perform or how large our data needs to be. But we might explore with it.

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.
In addition to that, we can do some additional non-direct tests, such as:

- Do a sentiment analysis on the generated sentence, and compare it to the sentiment score given in the match inputs.
- Using the match inputs, find the matches with the closest inputs, and compare the frequency of the words in the interviews for those matches and the generated interview.

6 Next Steps

- make use of betting odds to reflect expectations. - scrape half-time scores. - replace words like the name of the home team or the final score line with tokens and replace these tokens again after the result has been generated. - gather more data (include crowd sourcing on our platform) - manually check the translations.

References

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

[2] Ivan Lai. Conditional text generation by fine tuning gpt-2. `https://towardsdatascience.com/conditional-text-generation-by-fine-tuning-gpt-2-11c1a9fc639d`, January 2021.