Learning Contextual Event Embeddings to Predict Player Performance in the MLB

Connor Heaton[1] and Prasenjit Mitra[1,2]
(czh5372|pum10)@psu.edu
[1]College of Information Sciences and Technology, Pennsylvania State University
[2]L3S Research Center, Leibniz University Hannover

Baseball
171886

Abstract

Player performance prediction is a key aspect of all facets of the sports industry, whether that be for sportsbooks offering betting lines or for professional teams optimizing their roster. Historically, practitioners predominantly relied upon simple counting statistics as the basis for their projections. In baseball, this may manifest as the number of times a pitcher recorded a strikeout or a batter hit a homerun. While classical machine learning models trained on decades worth of counting statistics describing the entirety of a player's career can provide serviceable predictions, we can build better models by *contextualizing* the events that occur on the field – on what pitch count was the single recorded? Was anybody on base? - drawing parallels to historical transformations in the field of computational linguistics. That is, we see the sports analytics community as currently in the *bag-of-events* era and offer a steppingstone for advancing to the era of *contextual embeddings*. To this end, we train a representation learning model to understand the game of baseball as a sequence of events, and leverage this understanding towards predicting player performance. Trained on only 6 years of data and based on only 10 games worth of play-by-play data, our model can make single-game pitcher strikeout and binary batter has-hit predictions that are competitive with major sports books in the US for the 2021 MLB season. Furthermore, we achieve this performance while making a significantly larger number of predictions – 50% more than the closest sportsbook. All code and data are available at https://github.com/c-heat16/contextual_performance_prediction.

1. Introduction

Player performance prediction is a key aspect in all facets of the sports industry, whether that be for sportsbooks offering betting lines or for professional teams when optimizing their roster. Historically, practitioners predominantly relied upon simple counting statistics, or simple functions derived from these counts, as the basis for their projections. In baseball, this may manifest as counting the number of times a pitcher recorded a strikeout, a walk, an earned run, or an inning pitched. Even the spin rate of a pitch is a counting statistic as it is a *count* of the ball's number of rotations per minute. For batters, perhaps this manifests as counting the number of walks, singles, doubles, or home runs the batter induced from the batter's box or simple aggregate functions such as batting average, on-base percentage (OBP), or slugging (SLG). Then, once these statistics are obtained, existing methods use classical tools of machine learning – *i.e.,* support vector machine (SVM), logistic regression, and/or random forest, among others – to predict future player performance based on these counts [1, 4, 5].

Baseball has a noted history of record keeping, and the most basic counting statistics required for these predictions are available for as far back as 1975. Given decades of training data that describe the entirety of a players' career, the existing performance projection methods achieve

serviceable results, as evidenced by the commercial success of many sports betting companies in the US. However, we believe more can be gained by *contextualizing* the on-field events – what pitch did the pitcher throw, and at what speed, to record the strikeout? What was the launch angle and exit velocity of the ball a batter hit for a homerun? – drawing parallels to historical transformations in the fields of computational linguistics and Natural Language Processing (NLP).

In the early days of computational linguistics, language was described by counting the number of times different words appeared – an approach known as the "bag-of-words" method [13]. Although an *accurate* depiction of the language, the simple counting statistics left a lot to be offered. For example, in what order did the words occur? In what sense? That is, if the word "single" has a count of one, was it used to describe a quantity (a *single* sheet of paper) or an event in baseball (the batter hit a *single*)? Modern, state-of-the-art techniques derive *contextual representations* of language, analyzing the *context* in which words appear, a much more *faithful* description of the language. That is, instead of having a *static* understanding of the word "single" models will *contextualize* the representation of "single" based on the words in the surrounding context. This transition not only improved the efficacy of models performing existing language computation tasks (*i.e.,* spam detection), but allowed for new tasks to be tackled (*i.e.,* language generation tasks).

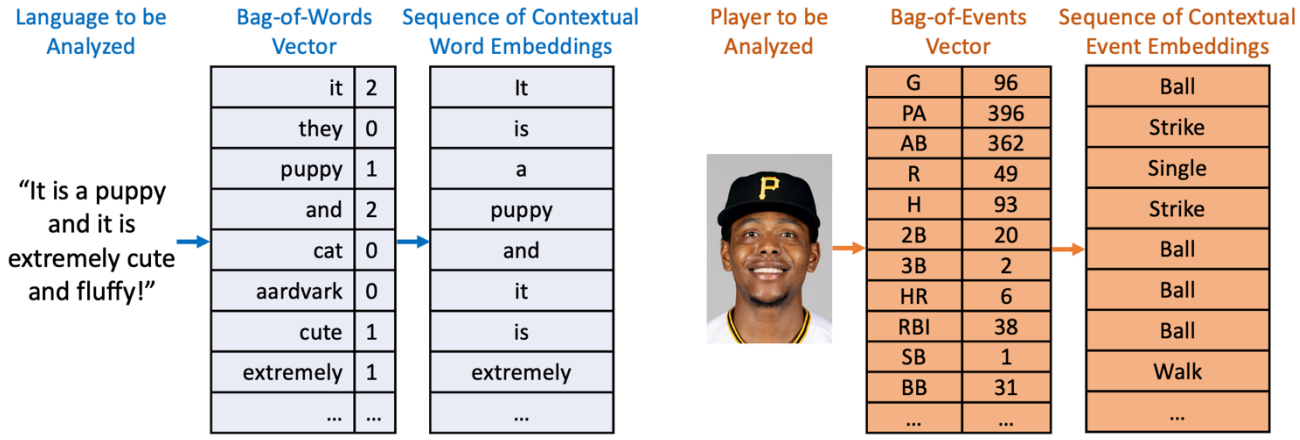

Figure 1. Visualization and comparison of historical transformations in NLP (left) and proposed transformations for the field of sports analytics (right).

We view the sports analytics community as currently in the *bag-of-events* era and offer a steppingstone for advancing the community into the era of *contextual embeddings* (Figure 1). To this end, we train a representation learning model to understand the game of baseball as a sequence of events, integrating sabermetrics, data from in-stadium sensor suites, and the fundamental changes in the game state simultaneously. Once the model understands the game of baseball, we can use this understanding as a starting point for making predictions about future player performance. Not only do we believe this will lead to performance improvements in existing areas of sports analytics, such as pre-game performance prediction, but open the door for new questions to be asked of the game. With this methodology just in its infancy, it can make predictions that are competitive with major sportsbooks in the US.

2. Related Work

Major sportsbooks do not often publicly disclose the way in which they make predictions. So, we cannot discuss them in detail here. We can, however, explore related academic assays towards performance prediction in the MLB. While commercial sportsbook will of course adjust these methods, and they have concerns aside from pure performance prediction, after brief discussions with individuals in the community, we are confident their approaches are not drastically different in kind. We first discuss the general type of data used before discussing the models that learn from said data.

2.1. Data

To the best of our knowledge, major sports books and other prediction-making assays primarily rely upon simple counting statistics as the basis for their projections. Soto Valero [1], for example, explored how data mining tools could be used to predict the outcome of a regular season MLB game. After performing an attribute evaluation and feature selection process, Valero's team-specific models achieved an accuracy of almost 59%, a promising result. Among those relevant to our study (ignoring features such as indicators if a team won their previous game and which team is home), features with the highest importance to the model were related to Batting Average on Balls in Play (BABIP), fielding percentage (FPCT), starting pitcher Earned Run Average (ERA), Slugging (SLG), and On-base Percent (OBP) plus slugging (OPS) – all of which are derived from simple counting statistics.

$$BABIP = \frac{H - HR}{AB - HR - K + SF}$$

Equation 1: Equation to compute BABIP where H is the number of hits, HR the number of home runs, AB the number at-bats, K the number of strikeouts, and SF the number of sacrifice hits during a certain time period.

$$FPCT = \frac{Put\ Outs + Assists}{Put\ Outs + Assists + Errors}$$

Equation 2: Equation to compute fielding percent (FPCT).

$$ERA = 9 \times \frac{ER}{IP}$$

Equation 3: Equation to compute ERA where ER is the number of earned runs and IP the number of innings pitches.

$$SLG = \frac{1B + 2B \times 2 + 3B \times 3 + HR \times 4}{AB}$$

Equation 4: Equation to compute SLG where 1B is the number of singles, 2B the number of doubles, 3B the number of triples, HR the number of home runs, and AB the number of at-bats over a given period.

$$OBP = \frac{H + BB + HBP}{AB + BB + HBP + SF}$$

Equation 5: Equation to compute OBP where H is the number of hits, BB the number of walks, HBP the number of hit by pitch, AB the number of at-bats, and SF the number of sacrifice hits over given period.

$$OPS = OBP + SLG$$

Equation 6: Equation to compute OPS where OBP is computed as described in Equation 5 and SLG as in Equation 4.

Consider the equations to compute these important features presented above in Equations 1-6. As we can see, all these statistics are fundamentally obtained by simply counting the number of times different events occur on the field of play, without regard for *how* they happened. While

metrics can be combined to describe complementary parts of the game, doing so can only reveal so much as all of the metrics are still dependent upon a crude description of the game – event counts.

In a related study, Silver and Huffman [2] explore how machine learning methods can be used to predict the outcome of a plate-appearance (PA) between a given batter and pitcher, finding their approach outperforms existing methods such as log5 [3]. Across a variety of tasks, including the prediction of strikeouts, walks, homeruns, *etc.*, the authors find that the family of features they refer to as "batter_historical" and "pitcher_historical" consistently rank among the most influential for their predictions. Consulting the appendix, we see that the "batter_historical" feature group includes a 365-day moving average of the batter's number of PAs, per-PA rate statistics for singles, doubles, triples, homeruns, walks, groundball double plays, strikeouts, sacrifice hits, and weighted on-base average (wOBA), an adjustment for the parks in which these events occurred, and a binary value indicating if the batter recorded more than 40 PAs. The "pitcher_historical" group contains the same set of features, minus wOBA per-PA. Clearly, these per-PA rate statistics are ultimately derived by simply counting the number of times various events occur, dividing one quantity by another. Again, no care is given to *how* these events occur or their relative order.

Note that even more advanced sabermetrics not mentioned above, such as Wins Above Replacement (WAR), are ultimately derived from simple counting statistics as well. For example, consider a generic formula to compute the WAR of position players given below in Eq 7.

$$\text{WAR} = \frac{\text{RunsCreated} + \text{PositionAdjustment} + \text{LeagueAdjustment} + \text{ReplacementAdjustment}}{\text{RunsPerWin}}$$
$$\text{where RunsCreated} = \text{BattingRuns} + \text{FieldingRuns} + \text{BaserunningRuns}$$

Equation 7: Generic formula to compute the WAR of a position player in the MLB

Each variable of the numerator is computed using what is known as the linear-weights method, a two-step process. For example, consider the $RunsCreated$ value. First, the practitioner must identify the different events that will be used to evaluate position players and determine the number of runs they are worth, on average. The events will likely include strikeouts, singles, triples, home runs, and walks, among others. After the events are identified, it is time to *count* the number of times each event occurred (C_{event}), and the number of runs that were scored (RS), during the time period being evaluated. Once the counts of these events have been obtained, the practitioner will use a linear equation to determine the *run value* associated with each event. That is, they will solve for the coefficients α_{event} in Equation 8 below, where N events are included in the analysis.

$$RS = \alpha_{event1} \cdot C_{event1} + \alpha_{event2} \cdot C_{event2} + \cdots + \alpha_{eventN} \cdot C_{eventN}$$

Equation 8: Example of a linear equation that could be used to solve for the run value of various events, a step towards computing a player's RunsCreated metric.

Once coefficients α_{event} have been identified (*i.e.*, the *value* of each event), the $RunsCreated$ metric for player i can then be computed by solving Equation 9 below, where $C_{event*,i}$ denotes the count of $event$* for player i.

$$RunsCreated_i = \alpha_{event1} \cdot C_{event1,i} + \alpha_{event2} \cdot C_{event2,i} + \cdots + \alpha_{eventN} \cdot C_{eventN,i}$$

Equation 9: An equation that could be used to compute the RunsCreated metric for player i, using the coefficients obtained in Equation 8 above.

As we can see from Equations 7, 9, and 9 above, at the end of the day, the inputs used to compute more advanced sabermetrics such as WAR are simple counting statistics. As a result, the *way* in which these events occur is not reflected in the WAR metric. Consider the following situations. In situation A, batter X hits a dribbler down the third base line and has the speed to beat the throw to first. In situation B, batter Y hits a ball to deep left, and while many other batters would have reached second base, batter Y is unable to do so. For the purposes of computing WAR, both situations are effectively the same – a single was recorded in both cases. Without saying which situation is *better*, we hypothesize including such context will lead to more useful insights.

2.2. Models

Soto Valero explored four different classical machine learning models in their effort to predict the winner of a game: K-nearest neighbors (KNN), the multi-layer perceptron (MLP), the REPTree (a type of decision tree) and the support vector machine (SVM), exploring the efficacy of each as both a classification and regression model [1]. One model was trained per-team. Soto Valero ultimately find that the SVM performs the best, predicting the winner of a regular season game with almost 59% accuracy. In a different attempt to predict the winner of a game, Elfrink et al find that the XGBoost model can achieve an accuracy of almost 56% using their dataset [4].

To predict the result of a PA between two players, Silver and Huffman employ a multi-layer artificial neural network (ANN) [2]. Specifically, they employ a model with two hidden layers, 80 nodes in each layer. When predicting a player's batting average in the subsequent season, Bailey et al explore how Statcast data can be used to improve the forecasting accuracy of the well-known PECOTA algorithm. Bailey et al train a logistic regression model to predict a player's batting average for season $T + 1$ using Statcast data describing their play in season T [5]. Ultimately the authors find that the predictions from the logistic regression model do not surpass PECOTA, but when combined with PECOTA predictions lead to an improvement 56% of the time.

In general, we find that many other approaches for forecasting player performance employ similar types of models. It is important to note that none of the aforementioned models leverage the temporal nature of the game – perhaps as a result of the way data is traditionally made available in this domain, as a set of counts. Given a set of counts, it is impossible to recover the specific sequence of events the counts describe. While other computational methods for analyzing sport do leverage the temporal aspect of the game – such as that of action valuation in hockey or soccer [6, 7] – they do not provide a method for predicting future performance, but rather quantify the quality of previous play. Perhaps quality of previous play could be used as a predictor of future performance, but that is not the intention of the model. Furthermore, in those applications, actions are only valued with respect to their likelihood of scoring a goal. While useful in many cases, it is restricted to a one-dimensional analysis.

3. Methods

Our approach involves two steps. First, we "pre-train" a model to understand the game of baseball as a sequence of events. Next, we leverage this understanding of the game as a starting point and "fine-tune" the model to make predictions about a player's future performance given the sequence of events in which they recently participated. Our dataset, model, and training scheme are described in the pages that follow.

3.1. Dataset

We employ the same methodology employed by Heaton, et al., when constructing our dataset [8], expanding the dataset to include play-by-play data from the 2020 and 2021 seasons. Specifically, the dataset contains season-by-season data from 1995-2021, and play-by-play data from 2015-2021. Summary statistics for the dataset are presented below in Table 1.

Dataset Summary	
# Games	15,743
# PA	1.2M
# Pitches	4.6M
# Batters	2,229
# Pitchers	1,884

Table 1: Summary statistics of the dataset used in this study.

For this study, we are primarily concerned with the play-by-play data as we desire a model that *understands* the game of baseball as a sequence of events. As such, the historical data from 1995-present serves more as a supplement to the play-by-play data, describing the pitcher and batter involved in a PA.

3.2. Model Architecture

We employ the transformer architecture in this study, an architecture that is now ubiquitous in a wide variety of deep learning disciplines, including natural language processing (NLP) and computer vision (CV) [9, 10, 11, 12]. We briefly describe the transformer here but encourage the interested reader to consult the original paper for more details.

A transformer model is a model consisting of one or more transformer layers, and each layer consists primarily of one multi-head attention (MHA) module, which does much of the heavy lifting. Each layer takes as input a sequence of vectors and outputs an updated sequence of vectors of the same shape. In the original implementation, each vector corresponded to a word token. At a high level, each layer operates on a vector-wise basis, updating the representation of each vector based on the vectors in the surrounding context. The MHA module is what enables the model to selectively *tend* to different parts of the context based on the vector being updated.

For example, consider Figure 2 which illustrates the behavior of one *attention head* in a multi-layer transformer-based language model. Lines connecting the right column with the left indicate the model *tending* to those tokens, with darker lines denoting more attention being given to the connected tokens. In this example, we see that, when updating the embedding for the word "it," this *head* pays most attention to the first two words in the sequence, "the" and "animal" – what "it" is in this context. Different *heads* in different layers will learn to perform different functions, this head appearing to learn pronoun resolution. The model will ultimately emit a new embedding for each word token, informed by the context in

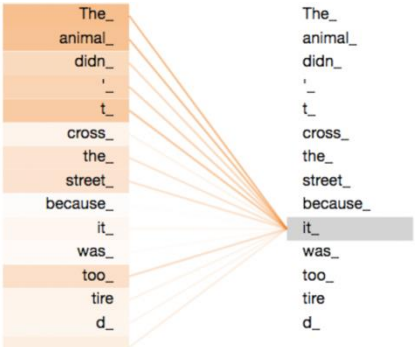

Figure 2: Visualization of attention in a generic transformer-based language model.

which they appear. Our model will utilize attention in the same manner, but each vector in the sequence will correspond to a thrown pitch instead of a word token.

At the beginning of training, the output of the transformer model, like any deep learning model, is not very informative. The model must be trained to emit embeddings that are useful for performing the task at hand. The training process we employ is described below.

3.3. Training Scheme

We employ a two-phase training scheme to train our model. First, we pre-train our model to understand the game of baseball as a sequence of events, and then fine-tune the model to predict future performance, leveraging the knowledge accrued during pre-training. In the sections that follow we describe how data is presented to our model as well as the pre-training and fine-tuning processes.

3.3.1. Data Presentation

As mentioned above, we want our model to understand the game of baseball as a sequence of events, analogous to understanding language as a sequence of words. When working with language, the word present at each timestep can be defined by a discrete token – that word. That word describes completely the *language item* conveyed at that timestep. In our application, however, that is simply not the case – it is not possible to use a discrete token to completely describe the *event* at any given timestep. Sure, you could describe the type of pitch that was thrown or the type of event that resulted from the pitch, but that only tells a small part of the story.

For that reason, we describe every possible aspect of the event using a combination of sources. The primary way in which each event is described is via a *gamestate delta*. The *gamestate* describes to the base occupancy, ball-strike count, number of outs, and score, so the *gamestate delta* describes a change in any of these components. Additionally, we present the model with a description of the type of *pitch* that was thrown and the type of *event* that resulted. These three features are presented as learnable embeddings to the model. Furthermore, we include data collected by Statcast which contains descriptions of pitch speed, rotation, and location as well as batted ball exit velocity, launch angle, and distance. Finally, we give the model a small description of the players involved using traditional sabermetrics. We do not spend much time choosing which sabermetrics to present to the model, giving the model the majority of sabermetrics available via the PyBaseball[1] package and allowing it to learn which metrics are most useful. Specifically, we give the model a description of the pitcher, batter, and their historical matchups segmented to the scopes of the last 15 days, this season, and their career. These sabermetric inputs only account for a very small portion of the description, roughly less than 5% of the numerals in each vector.

Individual pitch descriptions can then be combined to describe at-bats, innings, games, or even multiple games. In our application, we elect to present the model with two very related, but different, sequences of games: one for team-batting and another for pitching. That is, a single model will either be presented with a sequence of N games describing a team's batting, or N games describing an individual pitcher. With the data presentation established, we now train the model.

3.3.2. Pre-training

The first phase of our training is pre-training, where we teach the model to have a general understanding of the game of baseball as a sequence of events. That is, we want the model to learn

[1] https://github.com/jldbc/pybaseball

the general structure and progression of a game in the MLB – that a strikeout can only occur when there are two strikes in the count, the inning ends when three outs are recorded, or that in some situations, an intentional walk is warranted, for example. Fortunately, there is a large amount of existing work into sequential modeling in the field of NLP from which we can build.

Specifically, we adapt the Masked Language Modeling (MLM) scheme proposed by BERT [10] and perform Masked Gamestate Modeling (MGM). That is, when presenting our model with the sequence of events describing the N previous games, described in §3.3.1, we randomly mask 15% of the vectors in the sequence, asking the model to predict the missing gamestate and event. In order to correctly predict the missing entities, the model must inherently learn the general structure and progression of a game in the MLB. That is, based on the surrounding set of events, the model will learn to discern the *instantaneous gamestate* at each timestep, enabling it to make informed predictions about masked entities.

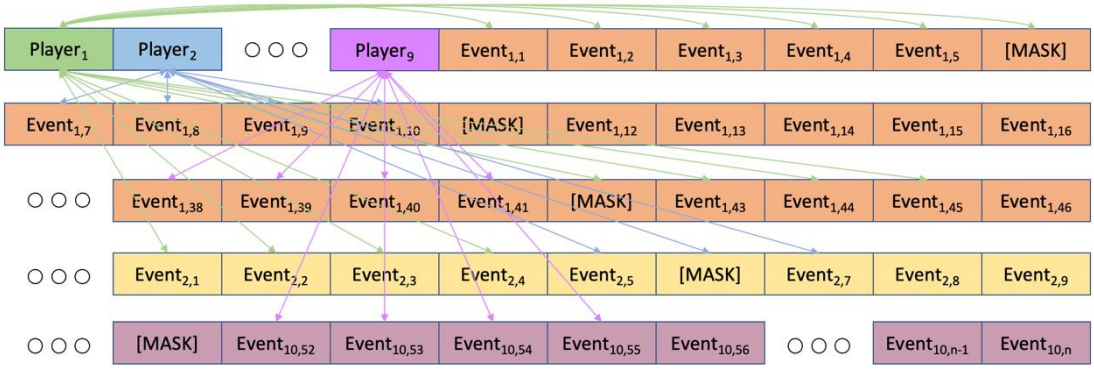

Figure 3: Overview of or MGM pre-training scheme using player embeddings. When presented with a sequence of events describing N games, the model will randomly mask 15% of the events and predict the missing entities. The introduction of player embeddings allows the model to discern a description of how each player has performed in this context, which it can use to make more accurate predictions for masked entities.

MGM helps our model learn about the sequence of events which comprise a game in the MLB, but we are ultimately curious about the players who effectuate said sequence of events. For this reason, we prepend special *player embeddings* to the sequence of events that our model processes. In our application, if we have a sequence of embeddings describing games $[G_{i-N}, G_{i-N-1}, \dots, G_{i-1}]$, we introduce *player embeddings* for the players in the starting lineup in game G_i. When presenting these *player embeddings* to the model, we manipulate the attention mask such that the *player embeddings* can only tend to events in the sequence in which the corresponding player was involved. During pre-training, the model learns to fill the *player embeddings* with signal describing said players that it can use to best predict missing events. A high-level overview of the processing is described in Figure 3.

3.3.3. Fine-tuning

Next, we can use the model's basic understanding of the game of baseball as a sequence of events as a starting point to make predictions about how players will perform in the future, a process known as fine-tuning. Specifically, we would like to make predictions about how a starting pitcher and nine starting batters would perform in a matchup against one another. This amounts to a rather straightforward 4-step process: 1) Identify the starting pitcher and starting batters for (half of) a particular game *G*, 2) Analyze how said players perform in game *G*, obtaining labels *L* for our predictions, 3) Obtain embeddings describing player performance in their past N games using

our model, 4) using these embeddings, predict targets L using a new linear projection layer. This process is described in Figure 4.

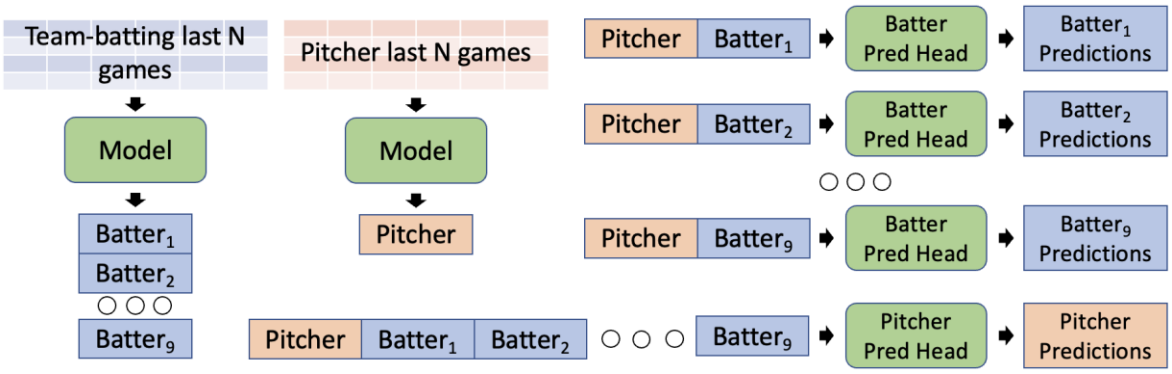

Figure 4: Overview of our fine-tuning process. Once the previous N games for the starting pitcher and batters are identified, they are processed by our model to obtain player embeddings. These embeddings are then used to make performance predictions for the starting pitcher and each batter.

4. Results

We performed the two-step training process described above, and here present results from the fine-tuning phase as we believe that will be of most interest to the audience. Specifically, we pre-trained the model to process a sequence of events from 10 games and fine-tuned it to predict the number of strikeouts, hits, and walks that will be recorded by each batter and the starting pitcher. Performance metrics describing said predictions are described below in Table 2.

	Pitcher			**Batter**		
Target	**MSE**	**MAE**	**R^2**	**MSE**	**MAE**	**R^2**
Strikeouts	5.37	1.85	0.21	0.43	0.57	0.06
Hits	4.62	1.72	0.07	0.42	0.57	0.04
Walks	1.48	0.98	0.04	0.17	0.29	0.02

Table 2: Performance metrics after fine-tuning our model to make predictions about future player performance. The MSE (Mean Squared Error) and MAE (Mean Absolute Error) describe the magnitude of error in the model's predictions, lower values indicating a better prediction. The R^2 metric describes the strength of the model, with values closer to 1 indicating a stronger model.

In analyzing the R^2 scores in Table 2, it clear that the model is significantly better at making predictions about future pitcher performance than future batter performance. This intuitively makes sense – although each type of player is described using up to 10 previous games in which they played, batters and pitchers will participate in a different number of events during that time. Pitchers, for example, will participate in *every* event on the field for which they are pitching. Batters, however, only participate in every 9th AB for their team. Over the course of a single game, the pitcher is likely to participate in roughly 9x as many ABs as a batter – more examples of how they impact the game. Furthermore, previous studies, such as Silver and Huffman, have used a 365-day moving average of statistics to describe batter, a much larger time span than just 10 games. Given out problem formulation, these results make sense.

Analyzing performance metrics from a single model is not wildly informative, however. For this reason, we also compare with predictions from three major sportsbooks in the US. Although they likely make some of the same predictions internally, the predictions the sportsbooks publicize, and

that we can compare against, are those of pitcher single-game strikeouts and binary batter has-hit predictions. While sportsbooks are not *only* concerned with making accurate predictions – they may want to consider the number of bets received on each side of a line, for example – we have been informed by people in the industry that they are a rather faithful reflection of their real predictions. A comparison of predictions from our model and various sportsbooks is presented below in Table 3.

It is important to note, however, the difference in binary batter has-hit predictions in our formulation and that used by the books. In our formulation, we make a prediction as to whether a batter will record a hit against a given *starting pitcher*. The books, on the other hand, make a prediction as to whether a batter will record a hit *in a given game* – arguable an easier task. If a batter has *N* ABs against the *starting pitcher*, they will have at least *N* AB's *over the course of the entire game*. Thus, there are simply more opportunities for batters to reach base and a larger number of individual batters are likely to reach base, increasing the likelihood of selecting a batter that will record a hit, even if at random.

	Pitcher Single-Game Strikeouts				**Batter Has-Hit**
Source	**# Predictions**	**MSE**	**MAE**	**R^2**	**Longest Streak**
Book 1	1,857	4.96	1.80	0.23	8
Book 2	2,957	5.26*	1.85*	0.21*	9
Book 3	3,239*	5.34	1.87	0.18	17
Ours	4,856	5.37	1.85*	0.21*	16*

*Table 3: Accuracy of 2021 projections from various sources. For each metric, the best score is underlined, and the second-best score appears with a * next to it.*

The results presented in Table 3 tell an interesting story. Although our model is not the *best* model on any metric presented, it is certainly competitive. For pitcher strikeouts our MAE is only 0.05 higher, and our R^2 only 0.02 lower, than that of the best sportsbook during the same time period. Our longest streak of correct batter has-hit predictions, 16, is only one day away from the longest streak obtained using a major sportsbook's projections. The biggest difference between our predictions and those of the sportsbooks, however, is in the sheer quantity. For pitcher strikeouts, it appears as though the sportsbooks only offer a line if they are confident in that projection. Our predictions, on the other hand, are made for **every** starting pitcher in 2021 – 50% more predictions than the next closest sportsbook. Our model's projections are competitive with major sportsbooks while reflecting a larger number of predictions.

It is worth highlighting the data utilized to make these predictions. Our model makes predictions based only on 10 games of in-game events from games **played in the MLB**, trained on only six years of data. For starting pitchers, the model is presented with a sequence of events describing their 10 most recent starts. For batters, however, the model is presented with the sequence of events describing how their team's batting performed against the 10 most recent starting pitchers their team faced, deriving embeddings for starting batters based on how they influenced said sequence. If a batter did not start in any of their team's 10 most-recent games, then our model is unable to derive an embedding of their performance. In a similar vein, if a starting pitcher hasn't previously made a start in the MLB, then our model will have **nothing** to base its predictions on for that pitcher. To demonstrate this, we present Figure 5, which describes the number of games used as the basis for prediction for both starting pitchers and batters.

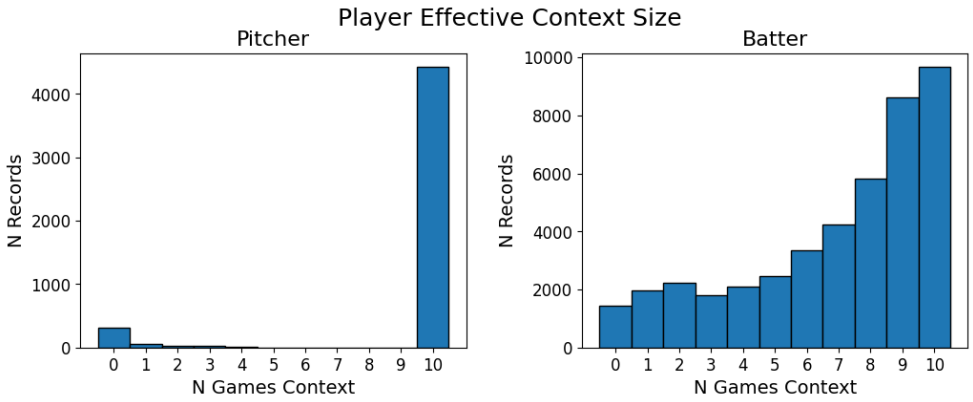

Figure 5: Visualization of the number of games used as basis for predictions for both starting pitchers and batters.

Figure 5 describes two very different situations for starting pitchers and batters. For pitchers, we see that in most cases players are described using a full context of 10 games. However, we see that batter predictions are often made based on significantly fewer than 10 games of effective context. This explains, in part, why our model made worse predictions for batters than pitchers. A potential option would be to include more games of context – if a batter started N games for their team, describing *more* of their team's games will increase the chance said player started in one of the games.

This strategy does not address *new* players, however, like rookies. When a player makes their first start in the MLB, there are no records of in-game events in the MLB in which said player participated – no amount of context will include a description of said player. Before the player made it to the MLB, however, they likely progressed through the minor league systems, and records of how they performed in the minors do exist. Moving forward, leveraging descriptions of how players performed in the minors is a likely path towards improving the performance of our model, and is likely something the major sportsbooks are already doing.

4.1. Embedding Visualization

As mentioned in section 2.1, this work is motivated by the observation that traditional sports analytics methods largely treat all events of the same type (single/double/triple/etc.) as being the same, and the idea that more analytical insights can be gained by *contextualizing* the events that happen in the field of play. Our model is giving quality predictions, but is it truly *contextualizing* the game, or uncovering some statistical phenomenon? To this end, we have our model process randomly selected sequences of events from the 2021 season and extract roughly 300,000 event embeddings for visualization.

Presented below in Figure 6 is a visualization of roughly 12,000 *singles* from the 2021 season, visualized with respect to different aspects of the game. An immediate takeaway from Figure 5 is that our model clearly finds more than one way to describe a single in the game of baseball. To start, the model appears to consider *outs when up* when describing singles, evidenced by the plot in the third column of the first row – in the central mass of embeddings in the bottom half of the plot, there appear rather strong striations for singles induced with zero, one, and two outs already recorded that inning. Additionally, the model includes a notion of runner movement around the basepath when describing events, as seen in the plots in the bottom row of Figure 5. The central

mass contained in the bottom half of the plot seems to mostly describe singles that occurred with the bases empty, while the groupings around the peripheral describe singles that moved players around the basepath, potentially inducing a run scored. Finally, singles that scored one or more run also appear in the same region of the plot in column 1 of row 2.

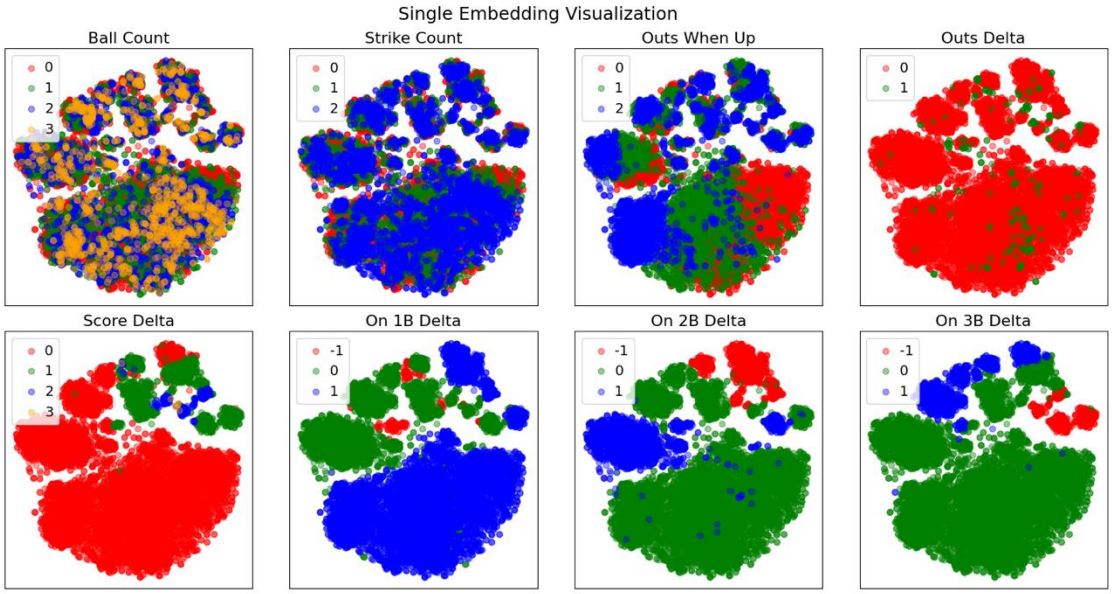

Figure 6: Visualization of singles as seen by the model. The first three plots in row one describe the state of the game before the single was induced, while all other plots describe the change in the game as a result of the single.

For another point of reference, we visualize roughly 20,000 *strikeouts* below in Figure 7. Again, we quickly see that our model finds more than one way to describe a *strikeout* in the MLB. Similar to the visualization of *singles*, we see that the model takes the number of *outs when up* and base occupancy in to account when describing *strikeouts*. In doing so, the model inherently learns to incorporate some notion of *leverage* into its descriptions. Originally, the notion of *leverage* was used to describe how *important* a situation was when a relief pitcher entered the game. Technically, high leverage situations (PA's) are those in which the win probability for the pitching team can change drastically[2]. In many cases except for that of a blowout, situations with multiple runners on base could be considered *high leverage*. As we can see, our model naturally identifies *strikeouts* recorded in *high leverage* situations without any guidance to do so.

[2] https://legacy.baseballprospectus.com/glossary/index.php?mode=viewstat&stat=249

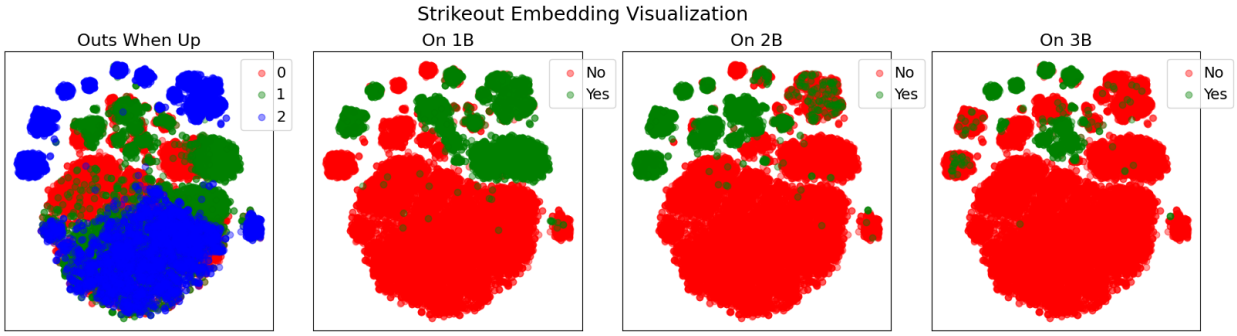

Figure 7: Visualization of strikeouts as seen by the model. The first plot describes the number of outs before the strikeout was recorded, while the remaining plots describe the occupancy of each base.

5. Conclusion

In the pages above we describe our method for *contextualizing* the game of baseball and demonstrate how it can be used for prediction purposes. Our model can make single-game pitcher strikeout and binary batter has-hit predictions that are competitive with three major sportsbooks in the US, all while in a much more restrictive setting. To start, our model is trained using only six years of data to make predictions based **only** on 10 games-worth of play-by-play data **in the MLB** – a small window into a player's career. Sportsbooks, on the other hand, leverage decades of data describing the entirety of a player's career in the MLB, as well as metrics describing their time in the minor league. It is also worth noting that our model yields similar prediction performance to those employed by the sportsbooks but makes these predictions for *at least* 50% more pitchers.

We view this as among the first steps towards *contextualizing* the game of baseball, and ultimately sports in general. Although the sports analytics community has provided many insights using existing approaches, *i.e.,* the "bag-of-events" approach, we have highlighted some of the shortcomings in this family of methods – mainly that all events of the same type are treated the same. As demonstrated in Figures 6 and 7, our model understands events on the field based on the context in which they occur and can leverage this understanding to make predictions about future player performance.

References

[1] Soto Valero, C. (2016). Predicting Win-Loss outcomes in MLB regular season games–A comparative study using data mining methods. *Journal homepage: http://iacss. org/index. php? id*, *15*(2).

[2] Silver, J., & Huffman, T. (2021). Baseball Predictions and Strategies Using Explainable AI. In *The 15th Annual MIT Sloan Sports Analytics Conference*.

[3] Russell, Matthew. "Estimating Baseball Event Probabilities with LOG5." Hi, I'm Matthew, 3 Dec. 2020, https://mruss.dev/log5/.

[4] Elfrink, T. (2018). Predicting the outcomes of MLB games with a machine learning approach. *Vrije Universiteit Amsterdam*.

[5] Bailey, S. R. (2017). Forecasting batting averages in MLB.

[6] Liu, G., & Schulte, O. (2018). Deep reinforcement learning in ice hockey for context-aware player evaluation. *arXiv preprint arXiv:1805.11088*.

[7] Liu, G., Luo, Y., Schulte, O., & Kharrat, T. (2020). Deep soccer analytics: learning an action-value function for evaluating soccer players. *Data Mining and Knowledge Discovery*, *34*(5), 1531-1559.

[8] Heaton, C., & Mitra, P. (2021). Using Machine Learning to Describe How Players Impact the Game in the MLB. In *The 16th Annual MIT Sloan Sports Analytics Conference*.

[9] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

[10] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[11] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877-1901.

[12] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[13] Harris, Z. S. (1954). Distributional structure. *Word*, *10*(2-3), 146-162.