A Foundation Model for Soccer

Ethan Baron **Daniel Hocevar** **Zach Salehe**

Abstract

We propose a foundation model for soccer, which is able to predict subsequent actions in a soccer match from a given input sequence of actions. As a proof of concept, we train a transformer architecture on three seasons of data from a professional soccer league. We quantitatively and qualitatively compare the performance of this transformer architecture to a baseline Markov model, as well as an MLP model. Additionally, we discuss potential applications of our model and associated ethical considerations.

Our project code can be found at: `https://github.com/danielhocevar/CSC413-Project`

1 Introduction

In recent years, adapting deep learning models trained using self-supervised learning on large datasets has proven effective on a variety of downstream tasks. These large pre-trained models, called "foundation models", have had a profound impact on fields such as natural language processing and computer vision [Bommasani et al., 2022].

We propose to train a foundation model for play-by-play data in soccer based on action data from historical matches. Much like language foundation models are trained to predict the next word given an input context, our soccer foundation model will be trained to predict the subsequent action given an input sequence of actions in a soccer match. Thereby, our model will learn embedding representations for soccer actions that could be used in a variety of downstream tasks. Specifically, our foundation model's embedding representation for actions and sequences of actions could be used as inputs to other models, or as a seed from which to simulate subsequent actions.

There are a variety of interesting use cases for our model. Firstly, the foundation model could be used to generate sequences of future actions given an input sequence, allowing analysts to simulate possible progressions of a match from a given situation. This could allow analysts to analyze tactical decisions within matches. Secondly, the hidden representations of the foundation model can themselves be used as inputs to other models, enabling analysts to refine data-driven player evaluation pipelines.

2 Background

2.1 Deep Learning for Sequence Modeling

One of the largest breakthroughs in recent years with respect to sequence modeling was the introduction of the transformer model [Vaswani et al., 2017]. A transformer is a neural architecture that transforms an input sequence into an output sequence. The driving idea behind transformers is the attention mechanism. At each step in the processing, this mechanism decides which other parts of the input sequence are most important for predicting the subsequent word. Transformers are widely applicable in a variety of downstream tasks. For example, this model could be trained to translate text (sequences of words) from one language to another, or for creating a chat bot.

Transformers come in several different flavours as well. An encoder-only transformer architecture only features encoding layers and omits any sort of decoding process. Encoder-only models are widely useful in natural language understanding and in creating highly valuable text embeddings which can later be applied to a variety of downstream tasks, such as text classification and search. A notable encoder-only model is Google's BERT [Devlin et al., 2019], which features a bidirectional transformer, allowing it to make use of both preceding and succeeding words when analyzing a sequence. Conversely, decoder-only transformer architectures only feature decoding layers, and their main purpose lies in generating new words or sequences based on the inputted representations. Popular decoder-only models include OpenAI's GPT models [Radford et al., 2019]. Encoder-decoder models, as seen in the original transformer paper [Vaswani et al., 2017], feature both encoders and decoders. These models are commonly used for sequence-to-sequence mapping tasks, such as translating text from one language to another.

2.2 Modeling Soccer Actions

Simpson et al. [2022] were the first to apply natural language processing architectures such as RNNs and transformers to predict soccer actions. They train multiple different architectures to predict the coordinates and action type of the subsequent action, given an input sequence of actions. They frame this problem as a multi-objective problem, with a regression component to predict the coordinates of the subsequent action, and a classification component to predict the action type. They found that both transformers and LSTMs offered significant improvements in predictive power over a Markov baseline model. However, they do not consider sequences that include turnovers, where the possession of the ball changes from one team to the other. In contrast to their work, we tokenize actions allowing us to treat the task as a pure classification problem, and we also consider all sequences from a soccer match, including turnovers.

Mendes-Neves et al. [2024] construct a foundation model for soccer actions which they call a Large Action Model. They use a public dataset of actions from soccer matches provided by Wyscout and describe various downstream tasks relying on this foundation model, such as for an in-game win probability model. Their architecture is a multi-layer perceptron trained to predict the next action given an input sequence of the three previous actions. In contrast to their work, we use a deep learning architecture that is more well-suited to the sequential nature of soccer actions, and we also consider longer input sequences.

Magdaci [2021] apply the ideas from `word2vec` [Mikolov et al., 2013] to learn embeddings for soccer actions by leveraging co-occurrences of actions. They tokenize actions into a standardized textual format, specifying the action type, location (in a 5x5 grid), and any other additional information. For example, the token "(1/5,1/5)<dribble>: incomplete" refers to "an unsuccessful dribble on the left side of the defense". The action embeddings are then used to identify similar actions and develop notions of player style. In contrast to this work, we use our trained embeddings as inputs to a generative model and learn representations for sequences of multiple actions.

3 Methods

3.1 Dataset

We use a tabular play-by-play dataset of matches from the 2018-2019, 2019-2020, and 2020-2021 seasons of the FA Women's Super League, widely considered one of the top women's soccer leagues in the world. The data is provided as part of the StatsBomb Open Data repository, a collection of action data for various historical soccer competitions made publicly available by StatsBomb, an industry-leading data provider. Our dataset includes a total of 939920 actions from 326 matches across the three seasons. To ensure that the representations learned by our model can generalize across different teams and seasons, we evaluate our model's performance on a withheld test set containing 10% of the matches. We also use 10% of the matches as a dedicated validation set, which is used for hyperparameter tuning.

For data collection and pre-processing, we leverage the `socceraction` Python package. This package includes functionality to convert the raw data into a representation called SPADL (Soccer Player Action Description Language) [Decroos et al., 2019], a standardized representation which focuses on player-centric actions and a unified set of features. We focus our analysis on the following features:

1. `team`: which team the action is associated with (home or away)
2. `action type`: what type of action occurred (e.g. "pass", "dribble")
3. `x` and `y`: the (x, y) coordinates of the action on the soccer field, in yards

We choose to model actions in a discrete space, which allows us to apply techniques from natural language processing. We must therefore discretize the (x, y) coordinates of the actions. To do so, we split up the soccer field, which has dimensions of 105 × 68 yards, into a hundred equally-sized rectangles. For example, rectangle $(0, 0)$ corresponds to the range $x \in [0, 10.5), y \in [0, 6.8)$ and rectangle $(1, 2)$ corresponds to the range $x \in [10.5, 21), y \in [13.6, 20.4)$. We can then tokenize each action into the following format: "<team>, <action type>, <bin>". For instance, the token "True, dribble, 4, 4" indicates a dribble by the home team in bin $(4, 4)$, which is near the middle of the field.

3.2 Neural Network Architecture

We frame the task of training a soccer action sequence embedding model as an unsupervised learning task, as in Radford et al. [2018]. Specifically, given a corpus of actions $S = a_1, a_2, \ldots, a_n$, the model is tasked with predicting the subsequent action a_t. Using the discretized action space described before, we minimize the cross-entropy loss using the log softmax of the transformer's output logits. This objective is equivalent to maximizing the likelihood of the data as follows:

$$L(S; w) = \sum_{i=k+1}^{n} \log P(a_i | a_{i-k}, \ldots, a_{i-1}; w) \tag{1}$$

Figure 1 illustrates the architecture of our transformer model. Each individual action is first converted to an embedding representation. The sequence of action embeddings are then combined with a positional encoding, and passed into a transformer decoder, following the architecture described by Radford et al. [2018]. This transformer decoder consists of n blocks which learn an embedding representing the sequence of actions. Finally, in order to produce predictions for the learning task described in Equation 1, we pass the transformer-learned representation for the last input token into a token classification head, which is implemented as a single fully-connected layer. This token classification head generates logits for each of the possible tokens, which are then passed through a softmax function to arrive at the predicted probabilities.

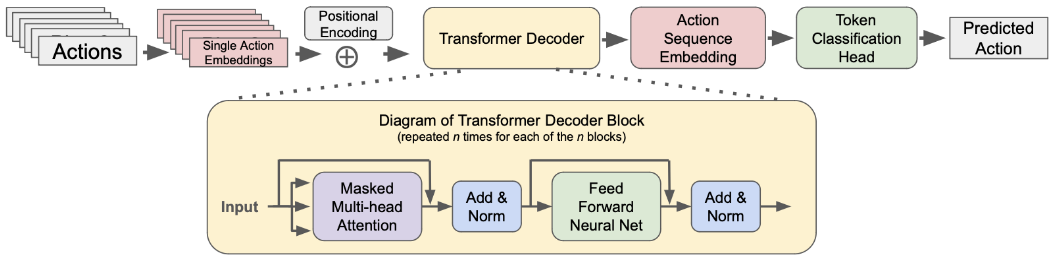

Actions
Single Action Embeddings
Positional Encoding
Transformer Decoder
Action Sequence Embedding
Token Classification Head
Predicted Action
Diagram of Transformer Decoder Block
(repeated n times for each of the n blocks)
Input
Masked Multi-head Attention
Add & Norm
Feed Forward Neural Net
Add & Norm

Figure 1: Model architecture diagram

Hyperparameters We use an action embedding dimension of 50. In each block of the transformer decoder, we use two layers for the feed-forward neural network with a hidden layer size of 500 and ReLu activation. For the token classification head, we simply use a fully-connected linear layer. For the positional encoding, we use sinusoidal functions of various frequencies, as proposed by Vaswani et al. [2017].

Optimization We use an Adam optimizer with a learning rate of 0.0002 and batch size of 32. We train the model for 6 epochs, and omit the use of dropout regularization within our positional encoding module.

Configuration We introduce two different sizes of the model described in Figure 1, each with a different number of decoder layers and attention heads. The specifications for these models are outlined in the table below:

Table 1: Comparison of the two model sizes

Model	Attention Heads	Decoder Blocks	Total Parameters
Small	2	1	365,565
Large	5	4	548,415

3.3 Baseline Models

Markov Model For our first baseline model, we implement a 2-gram language model, which essentially uses the observed pairs of actions in the training data to estimate the probability of each subsequent action from the given previous action. This approach encodes a Markov assumption underlying the data-generating process, which is probably too simple for our domain. However, it is easy to implement, and has reasonable inductive biases, hence it is a good baseline to compare against.

We train the model using observed sequences of actions in the training data $a_1, a_2, \ldots, a_n$. We construct a transition count matrix T, where element (i, j) indicates the number of times in the training data that token j was followed by token i. To avoid probabilities of zero, we introduce Laplace smoothing by adding 0.1 to each element. That is,

$$T_{ij} = 0.1 + \sum_{t=1}^{n-1} \mathbb{I}[a_t = i]\mathbb{I}[a_{t+1} = j]$$

After constructing this transition count matrix, we normalize each row to sum to one, giving us the transition probability matrix P:

$$P_{ij} = \frac{T_{ij}}{\sum_k T_{ik}}$$

Intuitively, this baseline Markov model simply learns to reproduce the observed frequencies of pairs of actions in the training set.

Multilayer Perceptron For our second baseline model, we replace the transformer decoder and token classification head with a simple MLP network, similar to the approach in Mendes-Neves et al. [2024]. That is, each individual action first gets converted into an embedding representation, and the sequence is then positionally encoded. Following this, we concatenate the sequence into a single vector, which is passed through the MLP to predict the following action probabilities.

We use an embedding dimension of 128, along with omitting the use of dropout. Our MLP consists of 4 fully-connected layers, with each layer having 1024 hidden features. We use ReLU activation and apply a softmax activation to the final output. When training, we use an Adam optimizer with a learning rate of 0.0002 and a batch size of 100, which we run for 3 epochs.

4 Results

4.1 Quantitative Results

Table 2: Comparison of quantitative results (TSFR indicates transformer)

	Accuracy				Mean Log Likelihood			
Dataset	Markov	MLP	TF (S)	TF (L)	Markov	MLP	TF (S)	TF (L)
Train	0.420	0.442	0.425	0.429	-2.847	-2.342	-2.469	-2.447
Val	0.417	0.411	0.421	0.422	-3.016	-2.674	-2.587	-2.574
Test	0.424	0.418	0.428	0.429	-2.975	-2.632	-2.549	-2.534

To evaluate the performance of our models quantitatively, we consider two metrics. First, we compute the models' accuracy on each subset of the data, measuring the models' ability to predict the most likely next action from a given situation. Second, we compute the mean log likelihood per example on each subset of the data, measuring how well-calibrated the models' predicted probabilities are. This metric is equivalent to the negative of the mean cross entropy loss per example.

We present the quantitative results of our models in Table 2. We see that both sizes of the transformer architecture achieved slightly higher accuracy than the Markov model for all three datasets. This suggests that the Markov model is adequately identifying the most likely next action given an input sequence. However, we see that the transformer architecture does achieve a significantly higher mean log likelihood than the Markov model, suggesting that the next action probabilities predicted by the transformer model are more well-calibrated. While the MLP also achieves a higher mean log likelihood than the Markov model, it falls short of both the large and small variants of the transformer when it comes to mean log likelihood over the validation and test sets. For this reason, we choose to utilize a variant of the transformer model for the rest of our analysis. In this case, the large variant of the transformer model outperforms the small transformer variant across both validation accuracy and mean log likelihood metrics, indicating its predictions are the most accurate and well-calibrated of all the models we tested.

4.2 Transformer Scaling Laws

Following the approach of Kaplan et al. [2020], we measure how our model's validation accuracy scales depending on the size of the training dataset, the number of actions in the model's context window, and the total number of parameters in the model. The results of this analysis are summarized in Figure 2. These results can be interpreted in order to provide insight into how the model can be extended or improved in order to achieve a better predictive accuracy. For computational reasons, we perform this analysis using the "Small" version of the transformer architecture.

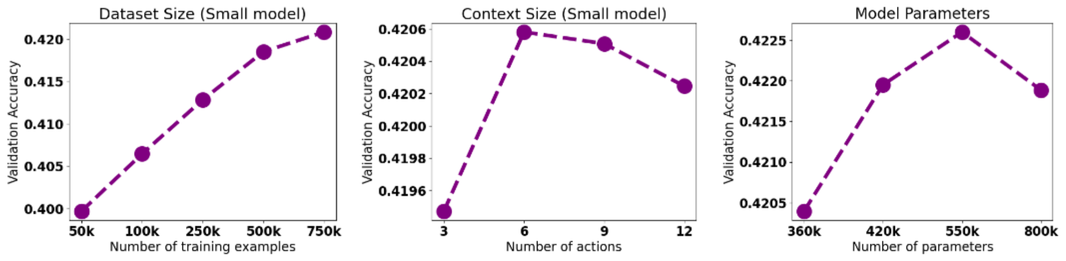


Figure 2: Plots showing how the validation accuracy of the model varies depending on dataset size, context size and number of parameters

Given that the validation accuracy of the model improves as the number of training examples increases, gaining access to a larger dataset than the one we use now would likely improve the accuracy of the model. However, we find that there are diminishing returns to increasing the context window of the model, as well as the number of parameters. Specifically, increasing the context size beyond 9 and the number of parameters beyond 550k does not appear to improve the accuracy of the model. These findings likely indicate that a more complex model is more likely to overfit, making it harder to achieve satisfactory performance on unseen data.

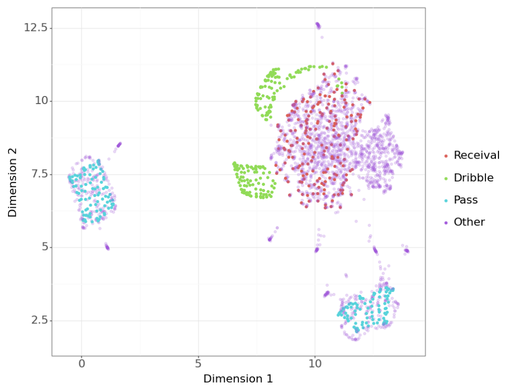
4.3 Visualizing Embeddings

Figure 3 provides a visualization of the learned embeddings for individual plays in out dataset. We use UMAP McInnes et al. [2020] to reduce our 50-dimensional action embeddings into a two-dimensional space for visualization. We see that the model has automatically learned to group similar actions together, such as having distinct clusters for certain action types for each team. Furthermore, the model has learned to represent the geometry of the soccer field, as we see consistent patterns with respect to the bin labels.

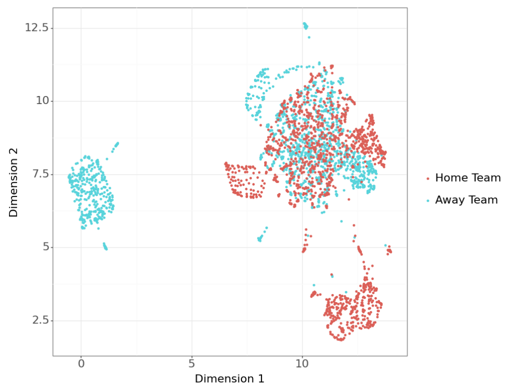
4.4 Example Outputs

Figures 4, 5, and 6 all show visual examples of a 10-play soccer sequence.

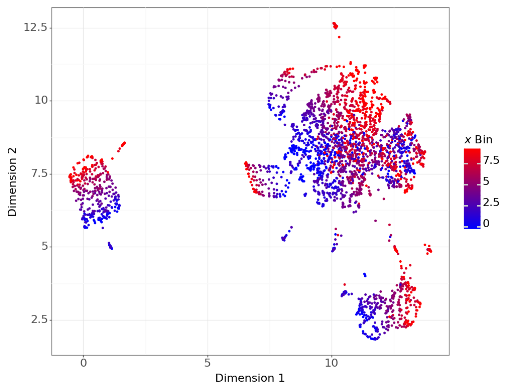
Figures 4 and 5 demonstrate the full ground truth sequence, along with the same sequence, but with the last play being predicted by each of the 3 models, given the previous 9 plays as input. These plots in particular demonstrate a couple of failure cases between each of the models. From inspecting a



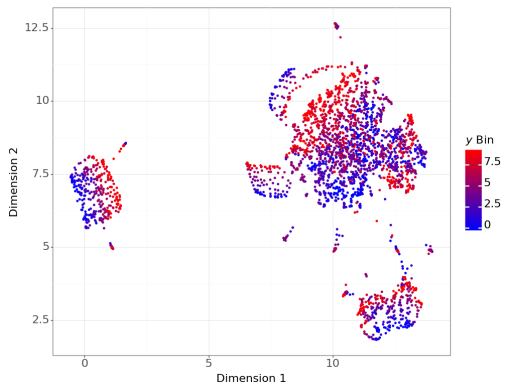
(a) Embeddings by action type



(b) Embeddings by team



(c) Embeddings by x-coordinate



(d) Embeddings by y-coordinate

Figure 3: Visualizations of individual play embeddings

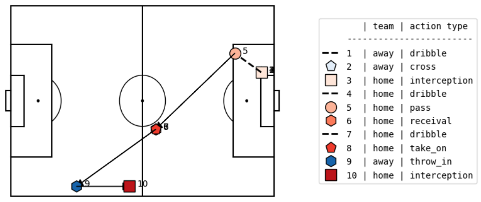
variety of failure cases, it becomes apparent that all 3 models tend to fail on the same sequences way more often than not. In figure 4 for example, the models fail at correctly predicting the occurrence of an interception play. This makes sense, as interceptions are less frequent than other play types, and can occur out of nowhere, leading them to be hard to predict in general. In figure 5, the models are able to recognize that the play following a throw in should be the receival of the throw in. However, none of the models are able to predict where the throw in will be received; understandably so, since in practice this has little to do with the previous actions and more to do with where each player is on the field.

Figure 6 shows two examples of where all models successfully predict the correct subsequent play. For each sequence in the figure, only one plot is shown, as each model's output was identical to the ground truth.

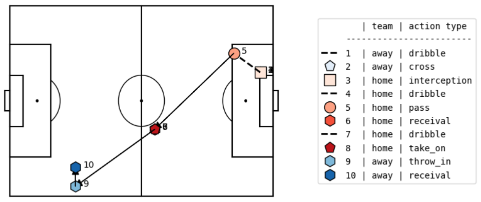
5 Potential Applications

Our foundation model could be used in a variety of interesting applications.

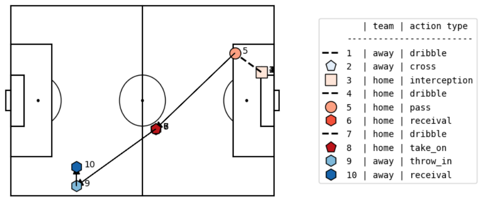
Firstly, the generative nature of our model allows analysts to simulate sequences of soccer actions from a given input sequence. This functionality allows teams to analyze tactical decisions in different situations, and identify the most successful strategies to pursue in a given context. For example, by simulating many sequences of actions following a given input, analysts could determine whether passing the ball to a given area of the pitch is likely to result in more positive outcomes for the team than passing to another area of the pitch.



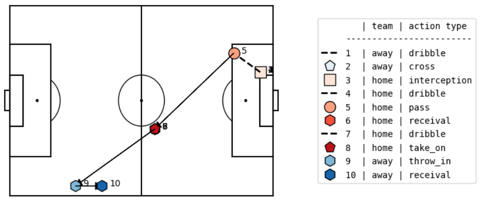
(a) Ground truth



(b) Markov

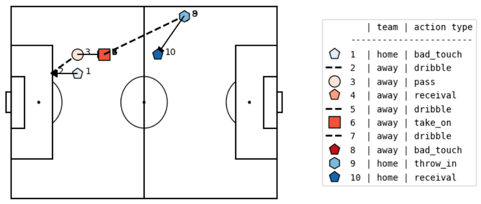


(c) MLP

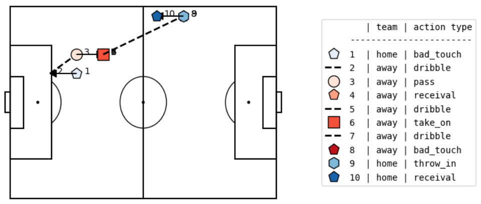


(d) Transformer (large)

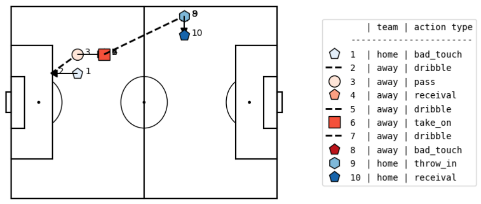
Figure 4: Sequence where models fail



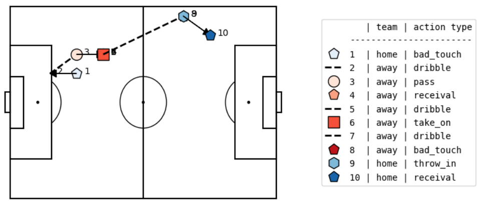
(a) Ground truth



(b) Markov



(c) MLP



(d) Transformer (large)

Figure 5: Another sequence where models fail

Secondly, the sequence embedding learned by our model could be used in several downstream applications. These applications include using the representation of a given state within a soccer match to build an in-game match win probability model, or a continuous indicator for which team has the momentum in a match, as in Mendes-Neves et al. [2024]. Another interesting idea is to include this multivariate state representation as an input to an expected possession value model [Fernández et al., 2020].

Thirdly, the action and sequence embeddings from our model could be used to develop an improved understanding of the unique strength and style of individual soccer players and teams. Just as Magdaci [2021] uses the average embedding of all of a player's actions, we could perform a similar aggregation to arrive at understanding of each player's individual role and preferences. Furthermore, one could

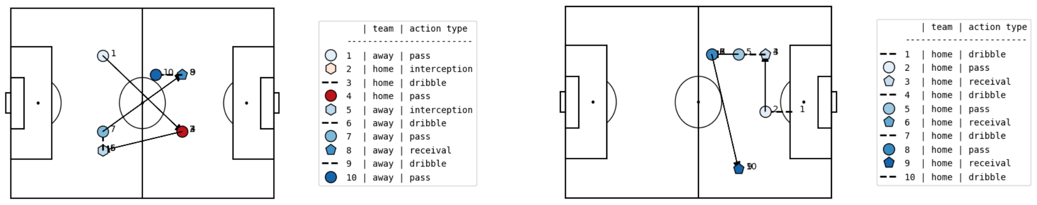


Figure 6: Two sequences where models succeed

even expand our architecture to learn player embeddings as part of the model, automatically extracting useful features that indicate differences in tendencies across different players.

6 Ethical Considerations

We used data from one specific women's professional soccer league. The representations learned by our models may therefore not apply as well to other contexts, such as men's soccer matches or youth competitions. This can therefore affect the performance on downstream tasks which are based on our foundation model. For instance, if our model is used to identify promising young players across different leagues, the analyst should ensure that the representations learned from the model can be applied accurately across these different leagues. Before using our foundation model in these other contexts, it may be useful to fine-tune the model with data from the particular context of interest, or train the model on a more diverse dataset in the first place.

As with any machine learning system, environmental impacts and carbon footprints also come into play. Training large-scale neural network architectures consume a lot of energy, much of which may not be from a renewable source. Additionally, the use of water for cooling and the mining of rare minerals are also topics of concern. We acknowledge these impacts when training our models, which we train on relatively low amounts of compute.

7 Conclusion

We use a transformer decoder architecture to develop a foundation model for soccer actions, capable of predicting the next action following from a given input sequence. We show that this transformer architecture provides more accurate and well-calibrated predictions than a baseline Markov model, as well as an MLP model. Our foundation model offers many interesting directions for future research, including various applications within soccer analytics.

8 Distribution of Labour

Name	Contributions
Ethan	Data preprocessing, Markov baseline model, embedding visualization, writing
Daniel	Model training pipeline, hyperparameter tuning, investigating scaling laws, writing
Zachary	Implement transformer and MLP models, plotting and evaluating results, writing
GitHub Copilot	Initial play tokenization and training loop

Table 3: Distribution of labour

We show the distribution of labour for each team member in Table 3. GitHub Copilot was enabled during these commits:

- `https://github.com/danielhocevar/CSC413-Project/commit/aa0c393938318674ea66e40f1ad18a7e6a069aed`
- `https://github.com/danielhocevar/CSC413-Project/commit/64ec02b963b7ecc5cc15f5af0a0b1db35d62ee95`

References

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding, 2019.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Ian Simpson, Ryan J. Beal, Duncan Locke, and Timothy J. Norman. Seq2event: Learning the language of soccer using transformer-based match event prediction. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 3898–3908, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539138. URL https://doi.org/10.1145/3534678.3539138.

Tiago Mendes-Neves, Luís Meireles, and João Mendes-Moreira. Forecasting events in soccer matches through language, 2024.

Ofir Magdaci. Embedding the language of football using NLP, 2021. URL https://towardsdatascience.com/embedding-the-language-of-football-using-nlp-e52dc153afa6.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013. URL https://api.semanticscholar.org/CorpusID:5959482.

Tom Decroos, Lotte Bransen, Jan Van Haaren, and Jesse Davis. Actions speak louder than goals: Valuing player actions in soccer. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 1851–1861, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330758. URL https://doi.org/10.1145/3292500.3330758.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018. URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

Javier Fernández, Luke Bornn, and Daniel Cervone. A framework for the fine-grained evaluation of the instantaneous expected value of soccer possessions. *Machine Learning*, 110:1389 – 1427, 2020. URL `https://api.semanticscholar.org/CorpusID:227016220`.