Technical test

Generative AI recreating football game

Presented by :

Ben Kmicha Rania

Table des matières

Chapitre 1

Game simulation

1.1 Exploratory Data Analysis (EDA)

The notebook data_analysis.ipynb describes the steps taken for data analysis. After conducting a simple data exploration and analyzing the distribution of actions (action occurrences), we note that some actions, such as walk and run, are the most frequently repeated actions during the play session. Additionally, actions like pass, tackle, etc., are considered occasional actions, which is logical for a football match.

Moreover, in examining the distribution of Sequence Lengths per Action and looking at the interquartile range, it is observed that some actions, such as run and cross, last less than other match actions. Also, using this interquartile method, it can be seen that these actions have outliers. However, considering my chosen approach and recognizing that sequence length can vary from one action to another in a real game, I will not remove these abnormal values.

By plotting the mean norm per action, it is evident that Walk and rest actions have the lowest mean norm. This is convenient since the player exerts less effort and lower acceleration values during these two types of actions. Additionally, the shot action has the highest norm value, as the player makes a rapid leg movement during a shot action.

Furthermore, when observing the overlaid sequences for the "Shot" action as an example, the norm values for "shot" preceded by a shot or run action are often higher, indicating more powerthan a shot proceded by a dribble for example, which is realistic. Similarly, for the "walk" action, norm values given a "run" action are higher than norm values when preceded by a "walk" action.

1.2 Different Approaches for Game Recreation

In general, there are models like LSTM-based networks, Recurrent Neural Networks (RNNs) known for temporal dependency, and Generative Adversarial Networks (GANs) that can generate sequences of player movements and actions over time. We could also mention Variational Autoencoders (VAEs) that can capture the underlying distribution of the data and generate new samples. Additionally, some reinforcement learning algorithms can be used to train an agent to make decisions. This kind of approach can be challenging when defining the rewarding function and state representation.

Finally, we can also discuss imitation learning, which involves using imitation learning techniques to make the AI imitate specific playing styles observed in real games.

Given the match data from FOOTBAR company and after conducting a simple EDA, we have identified three approaches. We will specify which one will be used later on.

1.2.1 First approach : TGAN and a classifier

At first glance, we considered utilizing a time series generator model, TGAN [?], designed for multivariate datasets to generate norm sequences with a fixed length. Subsequently, we planned to employ a classifier model to predict the gait label corresponding to each norm sequence.

However, upon closer examination, we identified some potential issues with this approach. Firstly, it would yield only fixed-length sequences, while each gait has a different size. Moreover, using TGAN might hinder our ability to consider causality among different match actions, particularly regarding the relationship between the previous and current action. Additionally, TGAN would not account for the influence of current and previous actions on norm values, potentially resulting in a simulation that lacks realism.

1.2.2 Second approach

The second approach involved employing an LSTM-based neural network for sequence action generation. Subsequently, we intended to utilize a regression model for length prediction, considering actual and previous actions, temporal position (as time step) of the current action as features, and norm length as the target. Finally, we aimed to leverage a time series forecasting model using the autoGluon library.

In practice, each sample in our data represents a time series, and the corresponding label is treated as a time series ID. As gaits are considered entire time series, our norm data is expanded to create concatenated time series. Using the model, we attempted to select the previous label of the target action from the generated sequence and forecast the values based on the predicted length as the time series horizon. Unfortunately, we encountered difficulties in implementing this approach due to the inefficient length of training time series data.

1.2.3 Final approach

Finally, we got the idea of using 3 AI models in order to re-create the football game.

LSTM sequence generator model

The first model is an LSTM-based neural network used to generate player action sequences for game simulation. Initially, we extract all actions as a training dataset from the provided match data after preprocessing. Subsequently, based on a desired play style, we structure the dataset into training sequences. To prepare the input data shape for the LSTM model, we perform one-hot encoding for the training sequences with a fixed length.

As a result, for each example/sequence and each possible starting action in this example/sequence, we create two sequences :

— An input sequence : which contains a subsequence of length sequence _length ; this subsequence range from the note t to the note t+sequence _length1

— An output sequence : which contains the following note to be predicted, the one at position t+sequence_length

The training is therefore performed by giving to the model a set of sequences as input and asking the network to predict each time the action that should come right after this sequence. Coming to the final part, we try to generate a new sequence from the language model, we simply give it as input a random sequence of duration sequence _length and ask the trained network to predict the output (using model.predict).

The output of the network is a vector of probability of dimension n_x which represents the probability of each action to be the next action given as input.

From this vector, we select the action which has the maximum probability.

We then concatenate this new action (its one-hot-encoding representation) at the end of the input sequence. We finally remove the first element of the input sequence to keep its duration constant (sequence _length).

Regression model for norm length prediction

Considering our data analysis and the varied length of each gait, it becomes crucial to incorporate the actual action, the previous action, and the temporal position of the current action as features for predicting the norm length. We opt to use RidgeCV as our model with a grid search method for determining the optimal learning rate value.

Regression model for norm values prediction

In this step, we try to generate the norm values. So according to our analysis, we must take into account this time the temporal relationship feature between values as temporal value index in norm list since it is considered as a time serie. Plus, we need to extract the previous action, the actual action because norms values depends on previous gait and finally the length of the actual gait recognizing that longer actions result in greater variance in norm values. Finally,for each generated action, we try to predict its length based on previously discussed features above and using the second model.Then, we generate the norm values based on the norm length and features extracted using the third regression model RandomForestRegressor. This choice is suitable due to the categorical feature dominance in our data.

The implemented algorithm takes into account the match length, number of games, the style of game by choosing the style actions in arguments and importantly, it mitigates the generation of repetitive and nonsensical actions by incorporating an occurrence penalty. This is particularly crucial given the unbalanced nature of the provided data.

1.2.4 Pre- / post-processing data

In order to have a prepared training data for models part, we have implemented ETL(extraction-transformation-loading) data pipeline. More precisely, we did some transformation to the original match data by :

- Drop "no action" label
- Action encoding :
  - Label encoding for second and third regression model
  - One hot encoding for Lstm generator model
- Filling in missing values :
  - For categorical features, we drop samples where there are missing values
  - For numerical features, We did interpolation for time serie(sample/row)
- Data explosion used for the second approach
- Feature engineering

When we got models' results, we tried to construct a list of dictionaries as needed that corresponds to each match game.