

Automated Event Recognition for Football Commentary Generation

Maliang Zheng¹, Daniel Kudenko¹

Abstract. The enjoyment of many games can be enhanced by in-game commentaries. In this paper, we are focusing on the automatic generation of commentaries for football games, using Championship Manager as a case study. The basis of our approach is a real-time mapping of game states to commentary concepts, such as “dangerous situation for team A”. While in some cases it is feasible to provide such a mapping by hand-coding it, in some cases it is not straight-forward because the meaning of the concepts can’t be easily formalized. In these cases, we propose to use inductive learning techniques that learn such a mapping from annotated game traces.

1 INTRODUCTION

Watching game action on the screen can be made more exciting by providing additional commentary with the pictures. This is the case for many game genres. For example, in-game battle reports that comment on the current situation in real-time strategy games and highlight especially heroic actions by individual troops clearly can enrich the game experience and make it more immersive.

Even though passionate commentary is a desiring merit for the games, generating the automated commentary has to confront enormous technical challenge. In football domain, ROCCO[1], from DFKI, and MIKE[2], from ETL, are the two of such attempts in generating the live commentary from simulator data. However, ROCCO focuses on incrementally recognizing pitch events with combining the elementary predicates, while MIKE emphasizes that cooperating with the six analysis modules to generate a range of remarks.

In this paper, we present our work on providing in-game commentary in real-time for simulated football games within Championship Manager, a highly popular title developed by Beautiful Games Studios (Eidos). In this game, the player steps into the role of a football team manager, purchasing and selling players, as well as overseeing their training progress. The football matches themselves are simulated based on player statistics. The player only watches the simulation and can’t interfere.

To generate the commentary, we first collected commentary concepts, such as “dangerous attack” from football news reports and other sources. We then attempted to manually implement direct mappings from game state to the set of commentary concepts (i.e., an *event recognition* mechanism). While for some concepts we were able to do so, for other concepts it turned out to be infeasible. Instead, we used an inductive learner to map hand-annotated game states selected from trace data, to generate the mapping function.

In this paper, we discuss our approach and present the empirical results on a few case studies, which show the successful application of machine learning to commentary concept generation. While our research has been focused on

football and Championship Manager, the methods can be easily transferred to other games, as long as game trace data is readily available.

The paper is structured as follows. We first present the Championship Manager game, and the structure of the game traces. We then discuss our general approach, followed by examples of hand-coded commentary mappings and inductively learned mappings. We finish the paper with a summary and an outlook to future work.

2 CHAMPIONSHIP MANAGER

This section briefly summarizes our experimental domain, the Championship Manager game (CM²) 2008, presenting both the trace data specification and the game log simulation aspects. The data extracted from each simulated football match are packed by a file (Figure1) that specifies the respective *spatial information* and *action description* of moving objects (i.e. *players, officials, and ball*). The pitch coordinate has its origin at centre mark, the x axis runs across the pitch from top to bottom, y axis points up to the sky, and z axis runs from left goal to right goal. Therefore, for every *sample period* (stated in the file’s header part), the moving object’s *position* is indicated in three-dimensional space³ as a decimal fraction with sign. In addition, the player’s *facing angle* is recorded and measured in degrees, which starts from the z axis and clockwise increases in the xz plane.

```
-----General Information
Version:1
Sample period:0.1■■■■

-----Position & orientation samples for referee (Index:0)
(Index, Position, Facing angle)
0, (50, 0, 0), 270
1, (50, 0, 0), 270
2, (50, 0, 0), 270
3, (50, 0, 0), 270
:
:
69232, (45, 0, 0), 110
69233, (45, 0, 0), 110
69234, (45, 0, 0), 110■■■■

-----Action type for player:(Squad type:Home, Index:0)
(Time, ActionType)
(0.001, eActionTypeWalking1)
(1.067073, eActionTypeWalking1)
(2.202635, eActionTypeWalking1)
(3.257802, eActionTypeWalking1)
:
:
```

Figure 1. Sample Data File Segment

¹Department of Computer Science, University of York, York YO105DD, UK, kudenko@cs.york.ac.uk

²CM2008 is a football-management simulation belongs to Edios Interactive Limited. <http://www.championshipmanager.co.uk/>

³ CM2008 only provides a two-dimensional simulation, so, the value for the y-axis always equals 0.

The samples capturing the above spatial information are recorded sequentially and at regular time points from the start of the match until its end. On the other hand, action descriptions are sampled only at the occurrence of a defined event. Each record thus logs the event *time* and related *action type*. The CM offers more than 200 action types for describing the pitch events. Actions with prefix ‘eActionType’ are dedicated to identify the agents’ (i.e. officials and players) elementary events, such as, eActionTypeLinesmanSignalCorner, eActionTypeShotStraight-JoggingRightFoot, and so on. Other actions that are prefixed with ‘ePersonBallAction’ characterize the ball’s actions, which include kick, drop, dribble kick, head, and pick up.

Similar to a real football match, there are four officials operating on the field and two teams competing to get the ball into the opposing goal. Each team consists of a maximum of eleven players, but without limiting the number of substitutions during the play. A number of common occurrences (e.g. corner ball) are also implemented. The existing commentary system of CM appears to be sensitive and precise in describing the play-by-play events, which almost make up the bulk of commentary. The evaluation of play, like, ‘good save’, fails to be mentioned, however, there seem to be a few of generic statements about the match, such as, ‘Man Utd’s passing was superb’ as long as Man Utd keep the ball for more than 4 passes without the opponent getting a touch.

3 GENERAL APPROACH

The main process of event recognition comprises the collection of commentary concepts and the mapping between trace data and target concepts. In order to enrich the communicable information and minimize the potential ambiguity, the concepts are rigorously selected from multiple sources, such as, the live text commentary offered by Sky Sports, the BBC live in EURO 2008, the Laws of the Game published by FIFA, etc..

Since the football events vary from each other in terms of recognition difficulty, we divided them into two groups. One group collects the events/concepts that are recognizable solely by hand-coded rules, and the other group requires more sophisticated approaches (in our case inductive learning). To make a proper partition, every candidate event must be firstly redefined as a set of attribute-value pairs, in which the attributes are expected to expressively summarize the major characteristics of the events and vary between commentary concepts (details are presented in section 5). Consequently, for those events that are constantly dominated by certain attribute(s), it is unnecessary to seek advanced methods, but simply evaluate the related value(s) of attribute(s) (see the following section for the example), while the rest are passed to machine learning process. Therefore, the attribute definition and its value calculation are two decisive factors in event recognition, especially, for the mappings which are hand-coded. The game state attributes to be used by machine learning to generate the mapping (classifier) are difficult to be decided optimally at a glance, because the attribute selection procedure must take its consideration all applicable attributes with various combinations, and the performance variance in applying different machine learning techniques on the same attribute set may even increase the complexity.

Our approach copes with this uncertainty in attribute selection. Relying on a good domain understanding, all event-related attributes are declared at first. Meanwhile, the classified

instances are randomly sampled to constitute a dataset in reasonable size, and whose values for predefined attributes are thereby calculated. All selected attributes are combined in various forms, but excluding those that are redundant in attribute semantics. This results in a batch of training files that are different in the properties of instances. The Weka⁴ workbench sequentially operates on these files according to three representative machine learning techniques [3, 4, 5]: C4.5, Naive Bayes, and K-Nearest Neighbor.

C4.5 is a typical Decision Tree method, which forms a general approximation of the target function by a decision tree when training examples are given, and all instances are classified by sorting them down from the root to some leaf node. Thus, the final model is expressed in understandable rules with low computation consumption for classification process. However, the training process could be computationally expensive as the amount of training examples must proportion to the complexity of problem. Otherwise, the trees are prone to errors.

Rather than explicitly search through the space of possible hypotheses, Naive Bayes, manipulates probabilities for hypotheses by estimating the frequency of various data combinations within the training examples. Although its effectiveness has been approved by many practical trials, there are two potential factors may affect its performance: 1) the assumption, the values of the attributes are conditionally independent given the target value, may be overly restrictive for some cases. 2) The computational cost on finding the optimal hypothesis could be raised when candidate domain is large.

In distance weighted K-Nearest Neighbor method, the global approximation never being performed until a new instance must be classified. Since the target function is estimated locally and differently for each new query instance, it is robust to noisy training data and effective when plenty training data are available. Nevertheless, the cost of classifying new instances can be high due to local approximation is taken place at classification time. Even worse, this mechanism might take some irrelevant attributes into account when comparing the instances and misclassify the similarity.

After obtaining the classifier model of each learning technique, cross-validation [3] and the standard error computation [6] are applied to estimate the accuracy of the learned hypotheses over unseen instances. Unless the classifier achieves an acceptable error rate, the attribute set is refined. This improvement strategy focuses on removing irrelevant attributes, or alternatively adjusts the training dataset (e.g. collect more training instances). Finally, the model with the highest statistical accuracy is accepted, and the process proceeds to *practical refinement*. All instances that fail to be recognized are classified and added to the training dataset, as the supplements, for subsequently rebuilding the classifier.

Figure 2 shows a screenshot of our commentary system. Once a game log file has been loaded, the simulation components in top right of screen are accordingly initialized: two groups of points on the pitch stand for the competing teams in blue and red colours, four of yellow points denotes the officials, and another point symbolizes the football. As long as the match proceeds, the recognized events (i.e. the commentary) will instantly pop up at

⁴ Weka is a data mining software contains tools for the classification task, available at <http://www.cs.waikato.ac.nz/ml/weka/>

left part of the screen. The match status data (e.g. substitutions) and system control options are also shown on the screen.



Figure 2. Screen Shot of the Football Commentary System

4 HAND-CODED COMMENTARY CONCEPTS

This section uses two examples to illustrate how the mapping between trace data and target concepts is implemented by hand-coding. One of the elementary concepts in football commentary is the *ball controller*. The relevant attributes to decide whether a player A controls the ball are:

- A is acting on the ball
- The last ball action was issued by A & A is still in ball's zone (circle region centred at ball's location with radius at 1.8 meter)

Because the decision only depends on either of these two attributes to be true, the state-to-concept mapping can be generated by hand-coding. The first attribute is determined by the ball actor at a specific moment, and the information can be retrieved from the ball action section of source file. However, the second attribute's value requires further data processing: a queue is used to store the playing history of the ball actor, and it is checked whether the distance between player A and the ball is not greater than the given radius threshold.

Another commentary concept, "player A is the *offside offender*", can be defined as:

- Line referee has raised the flag for an offside offence
- Player A's teammate has kicked the ball
- When the ball was kicked, player A is standing at offside position

Similar to the former case, this concept can be announced when all three attributes are evaluated as true. The value for the first attribute can directly refer to the referee action section of the source file. The second attribute is decided by the squad type of the most recent ball actor, which is stored in the queue. For the third attribute, the process firstly sorts the players' record by their coordinate value in z-axis team by team. Then, it only has to compare the largest (smallest) value of the attacking team with the first two largest (smallest) values of the defending team at the kicking time in order to detect the offside.

5 CASE STUDY FOR CONCEPT LEARNING

In this section we present a case study⁵ of constructing the state-to-commentary mapping with machine learning. The first case concerns learning role of the player, e.g. left back. Because the locations are different even if the players are taking the same role, it is difficult to tackle it by hand-coding a mapping from attributes, such as player's location. When the process begins, possible relevant attributes are chosen, as shown in Table 1.

| Index | Attributes | Rationale |
|-------|---|--|
| 1 | The distances between player's position and the mean line of the squad (z-axis) | The recognition of player's role usually has to refer to his relative position in the team. However, because the number of team players is variable in a match, it is not adaptable to describe their mutual relationship with a fixed number of attributes. These attributes are therefore used as an alternative measures to summarize the relative spatial relationship between player and the whole squad. |
| 2 | The distances between player's position and the mean line of the squad (x-axis) | |
| 3 | The variance of the player's positions in z-axis | Measures may be useful to describe the width of the team's standing in both horizontal and vertical directions. When players are standing wide, the distance between player's position and the mean line may be longer. |
| 4 | The variance of the player's positions in x-axis | |
| 5 | Player's coordinate X | The locations capture individual player's absolute spatial information, which may be considered together with the relative ones. |
| 6 | Player's coordinate Z | |
| 7 | Squad Side | To identify the sign difference of the position value in terms of opposite squad side |

Table 1. Attribute Set for Player Role

The training instances are manually picked out while watching the visual simulation. Meanwhile, the corresponding attribute values are also computed. As we can see from Table 2, the distribution of classes of training instances reflects the fact that the 4-4-2 formation is popular among all the teams in this game.

| Role | GoalKeeper | RightBack | LeftBack | CentreBack | CentreMidfield | LeftMidfield | RightMidfield | CentreForward | Sweeper |
|----------------------|------------|-----------|----------|------------|----------------|--------------|---------------|---------------|---------|
| Item | | | | | | | | | |
| Proportion in Sample | 9% | 9% | 9% | 18% | 18.18% | 9% | 9% | 12.12% | 6% |

Table 2. Classes Distribution in Sample Dataset (size:60)

We applied the C4.5 decision tree learning algorithm to the datasets which are summarized by different attribute combinations, the results are illustrated in Table 3. The first row,

⁵ The case study only uses CM2008's output data for sample generation, and the events to be recognized are not part of the simulator

the Average Error Rate (AER), is obtained by performing the 10-fold cross validation. After taking the Confidence Interval into account, the Estimated Error (EE) reveals that the first model built from attribute combination (1,2,3,4)⁶ outperforms others.

| Combinations Statistics | 1,2,3,4 | 1,2,5,6 | 5,6,7 | 1,2,3,4,5,6 | 1,2,3,4,5,6,7 |
|----------------------------|---------------|---------------|---------------|---------------|---------------|
| Average Error Rate | 24.24 | 27.27 | 25.75 | 30.30 | 31.81 |
| Standard Error | [-0.45,0.45] | [-0.51,0.51] | [-0.93,0.93] | [-0.75,0.75] | [-1.02,1.02] |
| Confidence Interval | [23.78,24.45] | [26.76,27.78] | [24.82,26.68] | [29.55,31.05] | [30.79,32.83] |

Table 3. Statistics Data for C4.5 (size:60)

One characteristic shared by the first and the third combinations is that they are free of the ambiguity in the value of coordinate location (i.e. the attributes in first one only represent the relative distance, and the third combination uses attribute 7 to identify the squad side). Within the coordinate system, the positions of teams in both sides are symmetric according to the origin (centre mark). As a consequence, all calculations must anyhow guarantee the consistency of the sign of values. However, after observing the decision tree, we noticed that attribute 7 is redundant with relation to attribute 6 (i.e. $z > 0$ is coherent to side=right and vice versa). Instead of indicating this side difference by a variable, the succeeding experiments attempt to automatically map all the positions of right-side-squad players into its opposing side prior to building the model.

Table 4 shows the results after redoing the experiment. The second model tends to be much more accurate in prediction, and the worst error rate is significantly reduced to 10.88%. This experiment proves the importance in terms of coordinate consistency.

| Combination Statistics | 1,2,3,4 | 1,2,5,6 | 1,2,3,4,5,6 |
|----------------------------|---------------|--------------|--------------|
| Average Error Rate | 25 | 10.17 | 11.36 |
| Standard Error | [-1.15,1.15] | [-0.71,0.71] | [-0.34,0.34] |
| Confidence Interval | [23.85,26.15] | [9.46,10.88] | [11.02,11.7] |

Table 4. Statistics Data for C4.5 (size:60)

Although training on the same datasets, the performance for the learning models could differ significantly as the algorithm is

changed from C4.5 to K-Nearest Neighbor (KNN). Table 5 presents the results for the models (the best classifier of each attribute combination obtained from intensive trails) that follow the latter algorithm, as well as their settings for the K. The results show that the second model is far more accurate than the others. Comparing the corresponding attribute sets, the attribute 3 and attribute 4 could introduce a large bias that leads to the classification fault, since the “closet” is defined by the difference of corresponding attribute values of two instances, however, the values of attribute 3 and attribute 4 are highly duplicated.

| Combinations Statistics | 1,2,3,4 | 1,2,5,6 | 1,2,3,4,5,6 |
|----------------------------|---------------|---------------|--------------|
| K | 14 | 1 | 5 |
| Average Error Rate | 70.45 | 15.25 | 59.09 |
| Standard Error | [-3.37,3.37] | [-1.57,1.57] | [-3.71,3.71] |
| Confidence Interval | [67.08,73.82] | [13.68,16.82] | [55.38,62.8] |

Table 5. Statistics Data for KNN (size:60)

Figure 3 shows the screenshot of classifier errors generated by Weka. The horizontal and vertical axes represent the labelled class and variance in x-axis (attribute 4) respectively. The plots are grouped into three groups, and the “X” denotes correct prediction and the “□” specifies the false classification. When the vote starts, the closet neighbors are always found in the identical group and dominate the classification if this class has comparatively larger amount. In this example, at least one more instance is marked by either turquoise “X” or green “□”, especially for the group in the middle. This output is also consistent with the one we have got by using C4.5 (See Table 4).

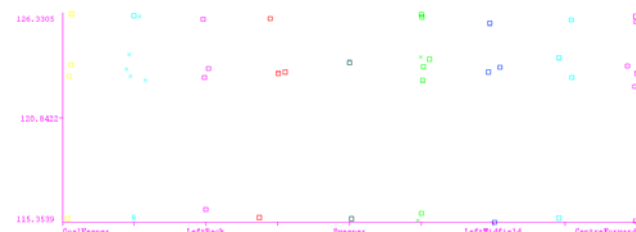


Figure 3. Plot visualization – KNN (attribute set: 1,2,3,4)

The models approximated by Naive Bayes (NB) are relatively weaker on instance classification, when comparing their statistical data (in Table 6) with the former ones.

| Combinations Statistics | 1,2,3,4 | 1,2,5,6 | 1,2,3,4,5,6 |
|----------------------------|---------------|---------------|---------------|
| Average Error Rate | 29.54 | 20.33 | 20.45 |
| Standard Error | [-1.33,1.33] | [-1.29,1.29] | [-0.9, 0.9] |
| Confidence Interval | [28.21,30.87] | [19.04,21.62] | [19.55,21.35] |

Table 6. Statistics Data for NB (size:60)

⁶ The attribute index details please refer to Table 1

Because the estimated error rate satisfies the application requirement, the second model of Table 4 is eventually adopted into this commentary system. Figure 4 reveals that the player roles in this match are all classified correctly. However, the model may misclassify particular roles especially at some occasions. The proposed solution for lowering such occurrence rate is continuously extending and refining the training dataset according to these misclassifications.



Figure 4. Player Role Classification

The ball is frequently passed in each football match. Human observers are able to identify the possible path of the pass and recognize its candidate receivers as soon as the ball is kicked. Sometimes, this assumption directly derives the commentary. Such as, “player X runs to the ball”. However, this is still a big challenge for the computers. Since such *ball transfer path* recognition more or less relies on certain background knowledge and a global view of the match, we tackled this problem with a machine learning approach. In addition, the understanding on the ball transfer path can be used to address another difficulty in identifying the player’s kicking intention for either passing the ball or a direct shot at the goal.

Because a football match is a highly flexible domain, the number of players may vary unpredictably, and makes it difficult fixing this flexible event into the machine learning process. Maintaining a decision structure that includes the mutual relationships between ball holder and all other players cannot be allowed, because the classifier features must be fixed from the beginning. Instead of deciding which teammate this ball will be passed to, every teammate must predict whether the ball is going

to be passed to him. Consequently, each player should estimate the potential ball passing from the fixed set of attributes (Table 7). One of the drawbacks of this design is the lack of global information, which may lead to two or more players being recognized as the ball receiver in some specific cases.

| Index | Attributes | Rationale |
|-------|---|--|
| 1 | Sender’s generic pitch position | This attribute aims to manually transform the numeric attribute values into the nominal items by dividing the whole pitch into several partitions. Because the ball kick action may happen anywhere in the pitch, which might results the attribute values widely different from instance to instance, we do not use the player’s absolute coordinate location to identify the spatial information |
| 2 | Receiver’s generic pitch position | |
| 3 | Sender’s face angle | The player’s face angle may be considered together with his kicking skill for deciding the possible ball transfer path |
| 4 | Sender’s action | This attribute value is given in the data file to notice the corresponding player actions. Different kicking events may represent different power and direct the ball’s flying angle |
| 5 | Receiver’s action | |
| 6 | The angle between the player and the ball holder | This is another spatial measures for recording the relative positional relationship between player and the ball holder |
| 7 | The risk of interception | This is a factor for ball passer usually has to evaluate |
| 8 | The shortest distance between receiver and defender | This is another factor concerning how easy the receiver can touch the ball from the ball holder’s perception |

Table 7. Attribute Set for Ball Transfer Path

The decision on attribute combinations is made according to their semantic coherence. For example, the combination (1,2,3,6)⁷ tries to capture the absolute spatial relations between ball sender and the ball receiver through attribute 1 and attribute 2, while the sender’s face angel may be combined with the relative angle between the player and the ball holder to show their relative positional relationship. As long as the training instances and attribute combinations are determined, the first group of models adhere to the C4.5 algorithm can be achieved. Due to the poor ranking in performance (in Table 8), the initially defined attribute combination set can simply remove the third (1,2,3,4) and the forth (1,2,3,6) options from the first iteration.

| Statistics Combination | Average Error Rate | Standard Error | Confidence Interval |
|---------------------------|--------------------|----------------|---------------------|
| 1,2,3,4,5,6,7,8 | 21.88 | [-1.18,1.18] | [20.7,23.06] |
| 1,2,3,6,7,8 | 21.88 | [-1.18,1.18] | [20.7,23.06] |
| 1,2,3,4 | 31.25 | [-0.35,0.35] | [30.9,31.6] |
| 1,2,3,6 | 31.25 | [-0.35,0.35] | [30.9,31.6] |
| 3,4,5,6,7,8 | 20.31 | [-2.2,2.2] | [18.11,22.51] |
| 5,6,7,8 | 21.88 | [-1.57,1.57] | [20.31,23.45] |

Table 8. Statistics Data for C4.5 (size:60)

⁷ The attribute index details please refer to Table 7

Because the confidence intervals for the models in Table 8 overlap with each other, advance refinement would be better to attempt to filter both attribute and training instances. However, only the last model's performance is improved among the results shown in Table 9

| Statistics Combination | Average Error Rate | Standard Error | Confidence Interval |
|---|--------------------|----------------|---------------------|
| 1,2,3,4,5,6,7,8 (ClassOrder) | 21.87 | [-1.18,1.18] | [20.6,23.05] |
| 1,2,3,6,7,8 | 20.31 | [-1.18,1.18] | [19.13,21.49] |
| 3,4,5,6,7,8 | 25 | [-2.2,2.2] | [22.8,27.2] |
| 5,6,7,8 (Resample) | 18.75 | [-0.71,0.71] | [18.04,19.41] |

Table 9. Statistics Data for C4.5 with Filter(size:60)

Similar statistical analysis is conducted on the models based on KNN and NB as well. Nevertheless, neither of the solutions could significantly boost the prediction accuracy. The probable reason is the size of dataset is extremely small compared to the large variety in attribute values. Therefore, the fourth model of Table 9 could be temporarily added into the system as a compromise of lacking a sufficiently large training dataset.

In most occasions, the choice of teammate for ball passing can be very flexible, even a human observer may fail to make a precise prediction. However, for some specific situations, like there is only a player standing in open space, the recognizer nevertheless can make the effective estimation.

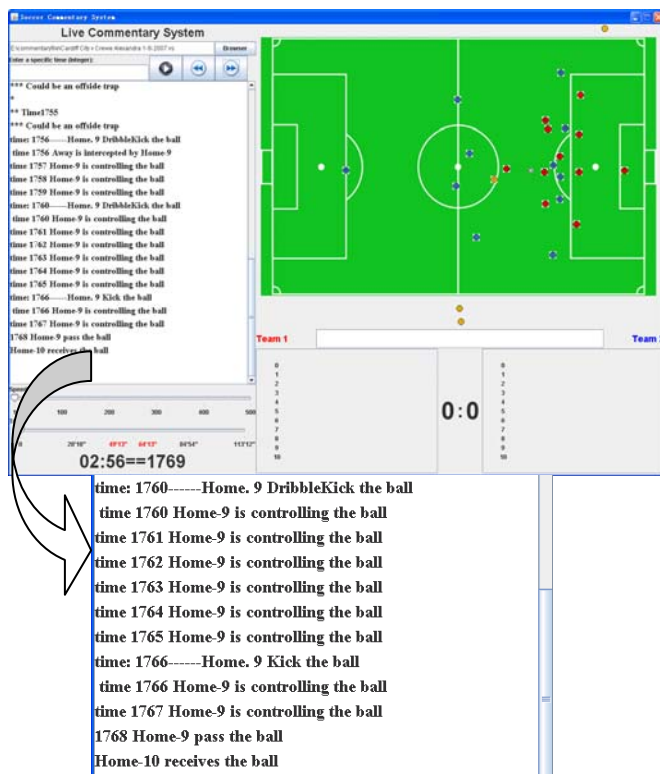


Figure 5. Ball Transfer Path Classification

The Figure 5 shows an example of correct prediction: the ball is predicted as to be passed to the centre forward when playing the counter attack.

Offside trap is very often mentioned in the real commentary, and it is also the precondition for deciding if the attacker successfully beats the trap. The reasonable solution for the computer to understand this event is that the system must have some sorts of background knowledge on its occurrence occasion, and have ability to decide if the defending team probably has the motives for making the trap. It is a challenge work for traditional approach to take all the possible factors into account, nonetheless, may be solvable by machine learning. All attributes are thought to be relevant are listed in Table 12.

| Index | Attributes | Rationale |
|-------|--|--|
| 1 | The mean value of defending line (in z-axis) | This attribute is used for occasion description. We believe there may be some implicit relationships between the relative distance from d-line to their goal and trapping the attacker into offside position. For example, when the team's d-line is high, the defenders could play the trap to against the counter attacks and generally cutting down on conceding goals. |
| 2 | The distance between the ball location to the mean of defending line (in z-axis) | This figure also works for providing the classifier more information about the match situation. For example, if the ball is already in the defending third, the back's forward motion may be less probably for playing the trap but blocking the attack |
| 3 | The distance between forward, and the back who is standing closest to the goal | This is a critical attribute for the final classification. If the distance is too far or forward has already stood at the offside position, it does not make sense to for backs to launch the trap. |
| 4 | The face angle deviation of players who is standing on the defending line | This attribute is used for human intention prediction. As playing this trap requires all backs move forward together, they must communicate with each somehow for cooperation. |

Table 12. Attribute Set for Offside Trap

Similar to the former cases, we pick out all sensible combinations of attributes. The Table13 illustrates the future error rate prediction for each of the models derived from the combinations. Most of model's average error rates are around 20%, but the fifth one's (combination 1,2⁸)reaches at 45.61%, which is definitely less accurate than others. Due to the overlapping in confidence intervals, it is not safe to assert whether the first, the third, or the seventh is the most accurate model.

| Statistics Combination | Average Error Rate | Standard Error | Confidence Interval |
|---------------------------|--------------------|----------------|---------------------|
| 1,2,3,4 | 19.30 | [-1.69,1.69] | [17.61,20.99] |
| 2,3,4 | 22.81 | [-1.48,1.48] | [21.33,24.29] |

⁸ The attribute index details please refer to Table 12

| | | | |
|--------------|-------|--------------|---------------|
| 1,3,4 | 19.30 | [-1.69,1.69] | [17.61,20.99] |
| 3,4 | 24.56 | [-1.31,1.31] | [23.25,25.87] |
| 1,2 | 45.61 | [-1.35,1.35] | [44.26,46.96] |
| 1,2,4 | 21.05 | [-1.93,1.93] | [19.12,22.98] |
| 1,2,3 | 19.29 | [-0.62,0.62] | [18.67,19.91] |
| 2,3 | 36.84 | [-2.02,2.02] | [34.82,38.86] |

Table 13. Statistics Data for C4.5 (size:60)

Since the attribute space for this event is small, and the error rates are at a similar level, the refinement process should focus on the dataset. From the last table's result, the further experiments are restricted only on those combinations that probably can derive the best performance, therefore, the second (2,3,4), fourth (3,4), fifth (1,2), sixth (1,2,4), and eighth (2,3) combinations must be firstly removed. Consequently, by generating a subsample with random replacement, the results presented in Table 14 reveal that the models of the first and the second combinations perform equally best, the predicted error rates are between 10.95% and 13.97%.

| | | | |
|---------------------------|--------------------|----------------|---------------------|
| Statistics Combination | Average Error Rate | Standard Error | Confidence Interval |
| 1,2,3,4 | 12.28 | [-1.69,1.69] | [10.59,13.97] |
| 1,3,4 | 12.28 | [-1.69,1.69] | [10.59,13.97] |
| 1,2,4 | 29.82 | [-1.93,1.93] | [27.89,31.75] |
| 1,2,3 | 17.54 | [-0.62,0.62] | [16.92,18.16] |

Table 14. Statistics Data for C4.5 (size:60)

From Table 15, a more accurate model is obtained from using the second combination (2,3,4) compared with the best result we have got from Table 14. The possible reason is the combination of attribute2, attribute3, and attribute4 is deterministic for the final classification decision, thus, taking them as the calculation factors for defining the neighbour can be comparatively effective.

| | | | |
|---------------------------|--------------------|----------------|---------------------|
| Statistics Combination | Average Error Rate | Standard Error | Confidence Interval |
| 1,2,3,4 | 8.77 | [-1.32,1.32] | [7.45,10.09] |
| 2,3,4 | 7.01 | [-1.09,1.09] | [5.92,8.1] |
| 1,3,4 | 10.52 | [-1.58,1.58] | [8.94,12.1] |
| 3,4 | 10.52 | [-2,2] | [8.52,12.52] |
| 1,2 | 21.05 | [-2.16,2.16] | [18.89,23.21] |
| 1,2,4 | 35.08 | [-1.12,1.12] | [33.96,36.2] |
| 1,2,3 | 40.35 | [-0.93,0.93] | [39.42,41.28] |
| 2,3 | 42.1 | [-2.63,2.63] | [39.47,44.73] |

Table 15. Statistics Data for KNN (K=1, size:60)

Table 16 summarizes the best results of NB. The best accuracy is achieved with the second attribute combination (2,3,4), which is also the best set for KNN. In contrast to the poor performance in C4.5, the fourth combination (3,4) is much

more improved. Consequently, we move on to the performance evaluation.

| | | | |
|---------------------------|--------------------|----------------|---------------------|
| Statistics Combination | Average Error Rate | Standard Error | Confidence Interval |
| 1,2,3,4 | 10.52 | [-0.96,0.96] | [9.56,11.48] |
| 2,3,4 | 8.77 | [-1.16,1.16] | [7.61,9.93] |
| 1,3,4 | 10.52 | [-0.96,0.96] | [9.56,11.48] |
| 3,4 | 8.77 | [-1.16,1.16] | [7.61,9.93] |
| 1,2 (after filter) | 36.84 | [-1.1,1.1] | [35.74,37.94] |
| 1,2,4 | 17.54 | [-1.32,1.32] | [16.22,18.86] |
| 1,2,3 | 17.54 | [-0.38,0.38] | [17.16,17.92] |
| 2,3 (after filter) | 19.29 | [-0.7,0.7] | [18.59,19.99] |

Table 16. Statistics Data for NB (size:60)

In all cases, except of some faults, the model generated by the NB is found out to be more accurate than the KNN, which proves its effectiveness in classification in this domain. In Figure 6, the text box shows that at 6004 (time index), the system recognizes the backs (in blue) are running forward maybe for playing the offside trap. And the visualization reveals this prediction at 6008 (i.e. the backs keep moving up from 6004 to 6008).

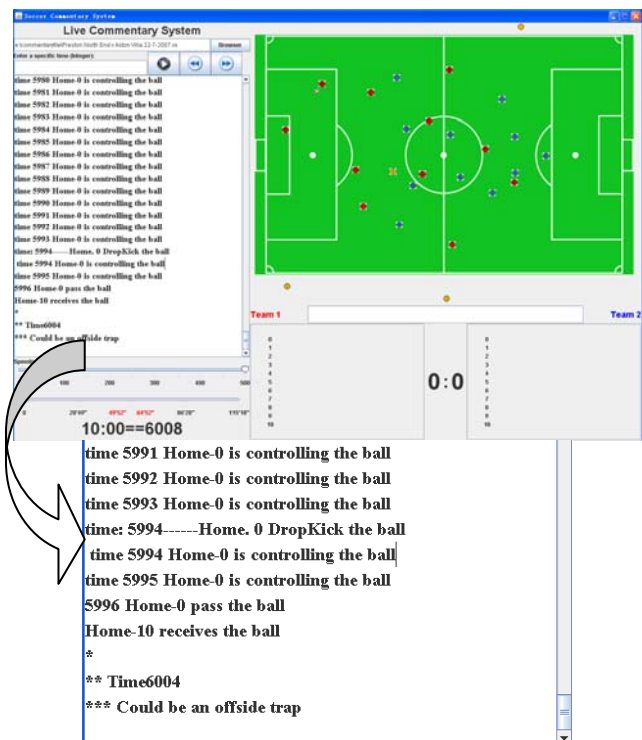


Figure 6. Offside Trap Classification

6 RELATED WORK

To date, we are aware of two other similar attempts at producing the automated football commentary systems in the domain of RoboCup⁹: ROCCO and MIKE. One feature of ROCCO is that it organizes the concepts in hierarchical way, which supports incremental event reorganization. Thus, elementary event and state predicates, which only happen at a single timepoint, are utilized to define those high-level event concepts. Much different from ROCCO, MIKE solves the difficulty by adopting three dedicated agents. One is for simple events, one is for pass work, and another is for shots and goals. Even though the events in both methods are identified by matching predefined propositional models (pattern) [7], the analyzer in ROCCO outperforms the other by the variety of its recognizable events. Intentional concepts, such as, offside trap, is inferred in ROCCO's commentary, while MIKE merely commentates on the ones that have taken place. However, besides the endeavour in event-based analysis, three other agents of MIKE enable it to perform additional state-based analysis. The system is able to automatically comment on the player's action, say, good passes, which benefited from interpreting the knowledge in a mathematical model. For the part of discourse planning and natural language generation, MIKE not only considers the importance of candidate events, but also tends to simulate interruption and abbreviation when generating the template based commentary.

According to the former survey, many strategies have been involved to create the pitch event recognizer but none of them can thoroughly solve the problem. One of the main difficulties is guaranteeing the output accuracy. In both systems, the success of event recognition definitely depends on the correctness and completeness of predefined conditions that are crafted manually. Therefore, their performance may be degraded dramatically when handling tricky occurrences by those static rules [6]. For instance, traditional solutions are occasionally misclassifying a player's kick action as a shot when he merely intends to pass the ball to the teammate who is nearer to the goal. Another weakness concerns the efficiency when generating the model for a large population of recognizable events. Both problems are alleviated by our proposed approach.

7 CONCLUSIONS & OUTLOOK

In this paper, we have described an approach to in-game commentary generation, which is based on the mapping of states to commentary concepts. We showed that while some concepts can be produced by hand-coded mappings, other concepts require a more sophisticated approach. Specifically, we propose the application of inductive learning, and the results of our case studies show the feasibility of this approach.

In order to deploy our approach to a real game, more concept categories need to be tackled, and integrated into the commentary generation system. Also, the sophistication of the text generation can be improved (so far commentary text is based on a small set of simple templates).

In future work, we also plan to extend the approach to other game genres, such as real-time strategy.

ACKNOWLEDGEMENTS

We thank Beautiful Games Studios for providing the data sets used in our study, and specifically Alex Whittaker for his assistance.

REFERENCES

- [1] D. Voelz, E. Andr e, G. Herzog, and T. Rist. Rocco: A RoboCup Soccer Commentator System. In M. Asada and H. Kitano, editor, *RoboCup-98: Robot Soccer World Cup II*, Springer. ISBN 3-540-66320-7 (1998)
- [2] K. Tanaka-Ishii, I. Noda, I. Frank, H. Nakashima, K. Hasida, and H. Matsubara. Mike: An automatic commentary system for soccer. In *Proceedings of ICMAS-98*.
- [3] T. Mitchell. *Machine Learning*. McGraw-Hill Series in Computer Science. ISBN 0-07-042807-7 (1997).
- [4] J. Quanlan. *C4.5: programs for machine learning*. Morgan Kaufmann series in machine learning. ISBN 1-55860-238-0 (1993).
- [5] I. Witten and E. Frank. *Data Mining: practical machine learning tools and techniques*, 2nd Edition. Morgan Kaufmann. ISBN 0120884070 (2005).
- [6] J. Gosling. *Introductory Statistics*. Pascal Press. ISBN 1864410159 (1995).
- [7] E. Andr e, G. Herzog, and T. Rist. Multimedia Presentation of Interpreted Visual Data. In P. Mc Kevitt, editor, *Proc. Of AAAI-94 Workshop on "Integration of Natural Language and Vision Processing"*, pages 74-82, Seattle, WA, (1994). Also available as Report no. 103, SFB 314-Project VITRA, Universit t des Saarlandes, Saarbr cken, Germany.

⁹ RoboCupTM is an international joint project uses soccer game as a central topic of research. More information will be available at: <http://www.robocup.org/>