UNIVERSITÄT BASEL

Crowdsourcing Annotations for Video Retrieval in Sports Videos

Bachelor Thesis

Natural Science Faculty of the University of Basel
Department of Mathematics and Computer Science
Databases and Information Systems (DBIS) Group
http://dbis.cs.unibas.ch

Examiner: Prof. Dr. Heiko Schuldt
Supervisor: Ivan Giangreco, MSc
Ihab Al Kabary, MSc

Fabio Sulser
fabio.sulser@stud.unibas.ch
2010-750-248

05.06.2014

Acknowledgments

I would like to express my thanks to Prof. Dr. Heiko Schuldt for giving me the chance to work on this interesting project.
A special thank goes to Ivan Giangreco and Ihab Al Kabary for supporting me during the whole project and supervising my work.
I also would like to thank all my fellow student who supported me if I had some questions and gave me some new ideas to solve a problem.
Last but not least I also would like to thank all my friends and family who have been supporting me over the whole duration of my bachelor studies.

Abstract

Nowadays, in many sports, a detailed game analysis becomes increasingly important when evaluating individual and team performance. To use existing analytic tools there is a need for data. Especially in sports such as soccer, a great controversy currently exists regarding the use of sensor solutions to track players. However, so far these solutions are expensive, not always available and are sometimes inaccurate or produce wrong information. On many occasions, only the video data is at hand, which for professional games is manually annotated by experts in this field of sports. Unfortunately, this data are also expensive to get.

This work shows another possibility to generate metadata of sport games besides using the conventional systems that uses experts, camera systems or GPS-tracking. This approach uses the Microworkers online platform, a crowd-sourcing system with his massive amount of people who are not experts in this field, to collect annotations in sports games. For this purpose, Microworkers [1], a platform that enables employers to create paid campaigns, is used for crowdsourcing. In this campaign, workers have to detect and annotate events in a video snippet of a soccer game. To get good results from crowd users, an understandable and easy to use web interface has been built that microworkers use for entering their task. To ensure a high-quality data, the sequence is annotated by multiple users and the results calculated out of the data collected using two different algorithms, the unweighted pair group method with arithmetic mean (UPGMA) known from bioinformatics, and DBSCAN, a well-known clustering algorithm.

The annotation includes the action, the corresponding team, the position and, the time the event occurred.

To increase the quality of the results, a rating system is applied that determines the reliability of the data entered by the microworker.

The system was evaluated using a dataset from a Manchester City - Bolton Wanderers game and the data collected through microworkers was tested to the ground truth. By creating a pretest where user are checked against the ground truth we could heavily increase the quality of the data. The results of our calculations are very promising depending on the difficulty of the sequence.

Table of Contents

1 Introduction

1.1 Motivation

In sports nowadays it does not suffices to analyze only intuitive performance. A player is increasingly rated by his physical skills and his accuracy. Therefore coaches and responsible persons in a club want to get these data to be able to analyze the game in a modern and simple way.

Different methods are possible to obtain more data. One way is to hire a company to analyze the game and enter the data. A second way is to automatically extract data using a pattern recognition system, and a third way is to give every player a GPS chip and put one into the ball, which will capture their positional information.

Due to the rise of the Internet in the past 20 years and the rise of crowdsourcing in the last five to ten years, we want to build a system using these modern features to provide a less expansive way to get metadata in soccer games regarding interactions in soccer games. To use a crowdsourcing-system make very much sense, because sports, and especially soccer, is one of the most common languages understood by a large number of people, regardless of age or the culture.

Soccer analyzing software like *SportSense* [2] use these kind of metadata for analysis and retrieval purposes. To the best of our knowledge, automatic systems based on image extraction from high-resolution cameras [3–5] are at the moment not reliable enough and technical systems for players are currently not allowed in soccer games. This is why manually annotated data still gives the best results for sport games metadata.

Therefore, some companies are specialized[1,2] to annotate metadata for soccer games and for other sports.

In this project, a system is built to generate manually annotated metadata, using microtasks. As described by Surowiecki [6] a group of untrained people can be even as good as the experts in many application domains.

[1] http://www.optasportspro.com
[2] http://www.sport-universal.com

1.2 Crowdsourcing

The term *crowdsourcing* was coined by Jeff Howie in 2006, when he wrote the article "The Rise of crowdsourcing" [7], which referred to crowdsourcing as a new model for outsourcing jobs that have been performed interoffice.
The word crowdsourcing is a combination of the two words *crowd* and *outsourcing*. Outsourcing describes the process of assigning certain subtasks to a third-party provider, with the main idea to reduce costs and increase the quality by letting experts do the work. In contrast to outsourcing work, employers are not able to choose the worker in the crowdsourcing approach. Employers are only able to chose the providers of a crowdsourcing platform, who themselves also do not know their workers.
These systems are able to solve a big variety of problems that cannot be solved efficiently by algorithms and computers, such as text- and image recognition, analyzing and categorizing video content or verification problems [7]. These problems can be solved by a huge number of persons participating on crowdsourcing platforms, independent of location, ethnic background or gender. Crowdsourcing is a new way of labor division; different to classical job distribution employers on crowdsourcing websites do not exactly know who their employers are. The employers have a huge amount of different workers available, where the workers are scattered over the whole world, with the small condition that they have Internet.
The process of crowdsourcing can be described in three steps. First, crowdsourcing can only be used for short tasks, which is why complex tasks must be divided into easy and short tasks. Second the dependencies between tasks must be to be managed and the quality of a task must be controlled [8, 9].
In other studies, we can see that crowdsourcing works well for a particular kind of task. Kittur [10] defines a good task as being a task that

- is fast to complete
- has low barriers to enter
- is objective and verifiable
- does not need big expertise
- can be broken up into subtasks

Advantages and Disadvantages

A big advantage of crowdsourcing, in contrast to conventional systems, is that they are much cheaper and people are only paid if their work is accepted. By outsourcing the work with crowdsourcing employers only have to pay the worker for the accepted finished task. This means they do not generate any fixed salaries. The pool of users is generally huge. Thereby, employers are able to create restrictions for users participating in their work if the system is able to handle restrictions.
Without any geographical restrictions, the diversification of workers can be seen as an advantage or a disadvantage for employers, depending on their task and their motivation to receive information. The takeaway results provide results from over the whole world, which

can be beneficial as they are independent of ethic, financial, or other conditions, but can be a disadvantage if employers only want to know how costumers in their neighborhood or their country react.

1.3 Microworkers

Microworkers [1] is a crowdsourcing platform, where each user can be a worker and an employer at the same time. Microworkers was created in 2009.

1.3.1 Campaign

The job offer in Microworkers is segmented into predefined categories. Each category contains different subcategories. The distribution of campaigns in the different categories on Microworkers in 2011 is illustrated in Figure 1.1.

Category	**Percentage of jobs**	**Percentage of reward**
Sign up	6.59	6.06
Click or Search	2.69	1.73
Bookmark a page (digg, Delicious, Buzz,..)	5.67	4.21
Youtube	1.04	0.64
Facebook	1.74	1.78
Twitter	0.25	0.31
Voting & Rating (photo, video, article)	1.84	1.11
Yahoo Answers	0.10	0.11
Surveys	0.00	0.00
Forums	0.63	0.62
Download, Install	0.13	0.41
Comment on Other Blogs	0.63	0.61
Write a review online (Service, Product)	0.07	0.21
Write an Article	0.07	0.21
Classifieds posting (Craigslist, Kijiji, etc.	0.12	0.29
Blog/Website Owners	0.95	3.38
Leads	0.33	2.47
Other	77.17	75.75

Table 1.1: Distribution of categories on Microworkers (reprinted from [11]).

As we have seen on Microworkers, the average campaign complexity is low. There are many campaigns where users just have to sign up or submit their email address.

1.3.2 Employer system

As an employer, users are able to create a campaign by filling out an online form. From this online form, employers are able to select the geographical location of participating users by continent or to allow all continents and exclude some explicit countries.
In the second section of the online form, Microworkers offers the campaign founder to declare the corresponding category by choosing one of the categories listed in Figure 1.1.
Employers also need to enter a campaign title, a description that explains what to do and especially where workers can do the work and a description what workers have to enter as proof for their work.
Then, there is also some information employers must fill out. They need to set a value for the maximum time and a value for the number of tasks they want to have finished. At least, the amount workers will earn will have to be set. Microworkers always enters the recommended value that they think is appropriate to the category. As an employer users still have the possibility to change this value.
At the end of the online form, the rating system needs to be selected. Microworkers provides two systems for participants. The "Do not verify or auto rate", which is the default system and was the only one available at the start of Microworkers in 2009. Employers are really free to create any system to prove the worker finishes the task.
The VCODE system, on the other hand, is a system that is provided by PHP GET parameters from Microworkers. The system automatically checks if a user entered the code provided by the campaign and worker ID provided by Microworkers and a secret key every employer receives. VCODE system has two subsystems. The first one will automatically rate entered tokens as satisfied and pay the workers. The second subsystem needs employer to manually annotate every task whether it is accepted or not.

1.4 Related Work

Crowdsourcing has been used in many works for generating information through crowd systems. There are different kinds of systems. Some are based on financial attraction, others more on social tribute. The probably most famous system known worldwide based on a social tribute is Wikipedia[3]. For building systems based on financial approaches some companies are well known such as Amazon Mechanical Turk[4] or Clickworker[5]. One of the most famous tasks in the crowdsourcing approach is reCaptcha [12], where humans have to enter a sequence of distorted characters.

To the best of our knowledge, this work is one of the first to use crowdsourcing for annotating a soccer game. The INRIA investigated the idea of generating soccer data through real-time crowdsourcing [13]. Their approach was to get users entering data on their smartphone while watching a soccer game, with the incentive that users want to help improving a system. Therefore they built a basic system where users have been able to enter motion paths of a player by finger-tracking. Second, they wanted to know which players information they entered, and last, the workers had to enter the event type.
To test their system, they used two groups of user types. Users of the first group do not or rarely watch soccer games and the second group comprises people who often watch soccer games. They compared the entered data by DBSCAN [14] and a Student's t-test.

There has also been an approach of generating video annotations in sports videos by using Mechanical Turk[15]. In this work, the main approach is not to annotate actions in a sports video, the MTurk users must track either a player, the referee or the ball. Users have to track a single object in a video sequence. This approach is not very similar to the common way of getting video information, which based on object tracking. Whereas, this approach is to create metadata on events, which are interactions between players and the ball, players with each other or the ball with a line, as a goal, for example.

1.5 Contributions

In this thesis, the SportSense Crowdapp is introduced. It is an online application to annotate actions in soccer videos using the crowdsourcing platform Microworkers. This approach is through the use of crowds to create annotations that are as good as the data of experts. The final dataset is calculated with collected data from Microworkers with two different algorithms. First, the unweighted pair group method with arithmetic mean (UPGMA) [16] is used. Second, the DBSCAN algorithm is applied.
To evaluate the system, the data from Manchester City against Bolton Wanderers is used. The calculation results are very promising, depending on the difficulty of the sequence.

[3] http://www.wikipedia.org/
[4] https://www.mturk.com/mturk/welcome
[5] http://www.clickworker.com/

2
User Interface

To improve the quality, of one must build understandable and easy tasks as described in *"Crowdsourcing, Collaboration and Creativity"* [10].

2.1 Preliminary User Interface

For preliminary system tests, the system was first evaluated using a simple layout that only considers

- The team
- The event
- The point in time

The first interface in Figure 2.1 does not yet allow entering the positional information.

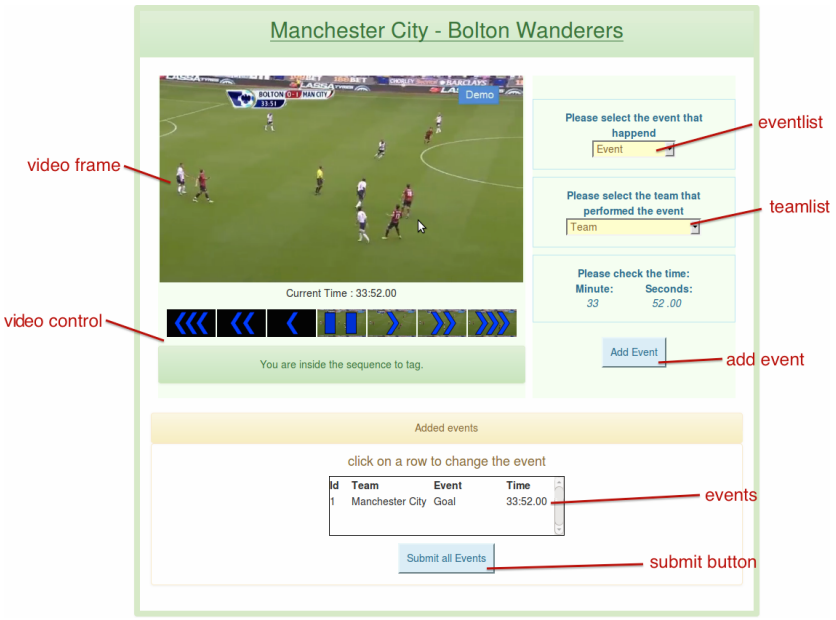


Figure 2.1: User interface that allows data entry

To move through the video, workers have the possibility to use the keyboard, where a push on the spacebar plays and stops the video, and navigation can be done by using the left (backward) and right (forward) arrow keys to navigate through the movie. Another possibility to change the position in the video file is to use the row under the video player. With this extension, the video can be controlled. By clicking on the images on the right side of the pause symbol 2.2 one can go forward. The number of arrows indicates how fast forward and backward a user goes by clicking, where every arrow is equated by a tenth second.

Figure 2.2: Pause symbol used in the movie controller

The background of the navigation symbols shows the video frames at the corresponding time. Black background images indicate that the user is outside his task.
If the worker navigates the video file to a position where he sees an action he to chose the appropriate event from the drop-down list. The third item workers need to select is the team appertaining to the selected event. By clicking on the list, users will see both team names and the corresponding shirt color. After selecting all items, clicking the "Add Event" button will submit the event to the list containing all entered events.
Those steps have to be repeated for the whole scene until all actions in the task's temporal sequence are added to the list. By clicking on a row in the events list, the video will jump to the entered time and the drop-down lists will show the entered values. Workers now have the possibility to change any information of this data set or delete the event. If they do not want to do either of these two manipulations they can cancel the editing mode.
After finishing his task determined by the sequence end, the worker must click the submit button to enter the events in the database and receive his token.
The evaluation of this system by using Microworkers is made in Section 5.1.

2.2 Second Layout

Based on the results of the first test using the preliminary user interface, the layout was adjusted.

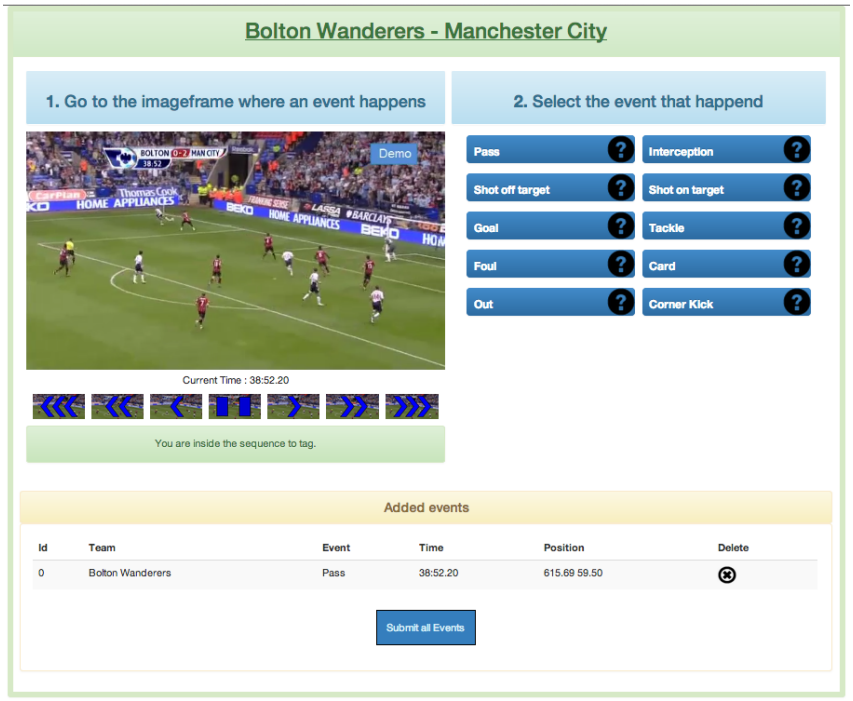


Figure 2.3: Second layout at starting position of video snippet

To guide the user in the choices, some titles with numbers were added to all steps a user has to enter. The input menu is split into different views, where the user first has to select the event by clicking the corresponding button. To help users select the right event, every event button contains a help text which shows up when hovering over the question mark in it. After selecting the event the event view will be replaced by a similar view where the user have to select the appropriate team. After selecting the team, the team frame also will be hidden and a new view will become visible. In this view, workers will see a soccer field where they have to enter the position where the event happens, by clicking on the appropriate position on the field.

After setting the position, participants must click on an "Add Event" button, similar the first layout, to enter the data. To encourage users to enter all events in the sequence and not stop after adding the first one, they will be asked to check the rest of the sequence for more events and to add them, because preliminary tests have shown that users often add just one event and then stop, rather than to check the sequence for more and to add them. In addition, the possibility to edit an event was removed; instead, a delete row containing a delete icon was added to the events list, where users can delete a wrong event by selecting the appropriate icon in the table.

3
Concept

This chapter will examine how the system is built. A closer look will describe the concept behind the data model, the rating system, the data integration, and how the system connects with Microworkers.
Microworkers offers the possibility to build a campaign as a developer, where each campaign is built from a specified number of tasks.

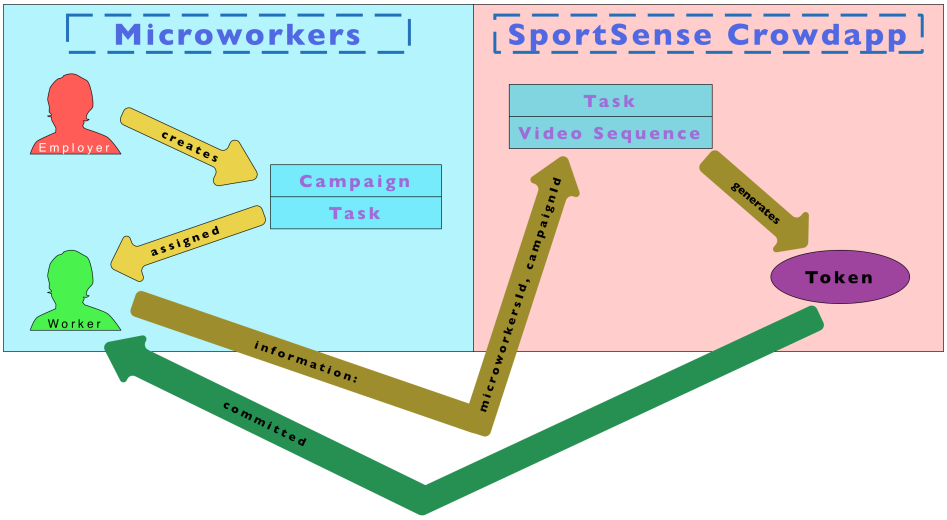


Figure 3.1: Interaction between Microworkers and SportSense Crowdapp

Figure 3.1 shows the interaction between the Microworkers platform and the SportSense Crowdapp. As employer on Microworkers, users have the possibility to create campaigns containing a specific number of subtasks.
Every task on Microworkers is assigned to a worker. This worker opens the Crowdapp, where he will receive a task with a specific video sequence. With the information provided by Microworkers, his token can be generated and, after finishing the task, shown to the worker. This token is the key that he must enter in Microworkers to complete the task and finally also receive payment. A detailed explanation what specifications we made for creating the campaign can be read in Section 4.

3.1 Data Model

In this thesis, a system has been built with which new campaigns can be created easily with other tasks, and which also has the potential to be extended to other sports. The following figure describes the data model that underlies this system.

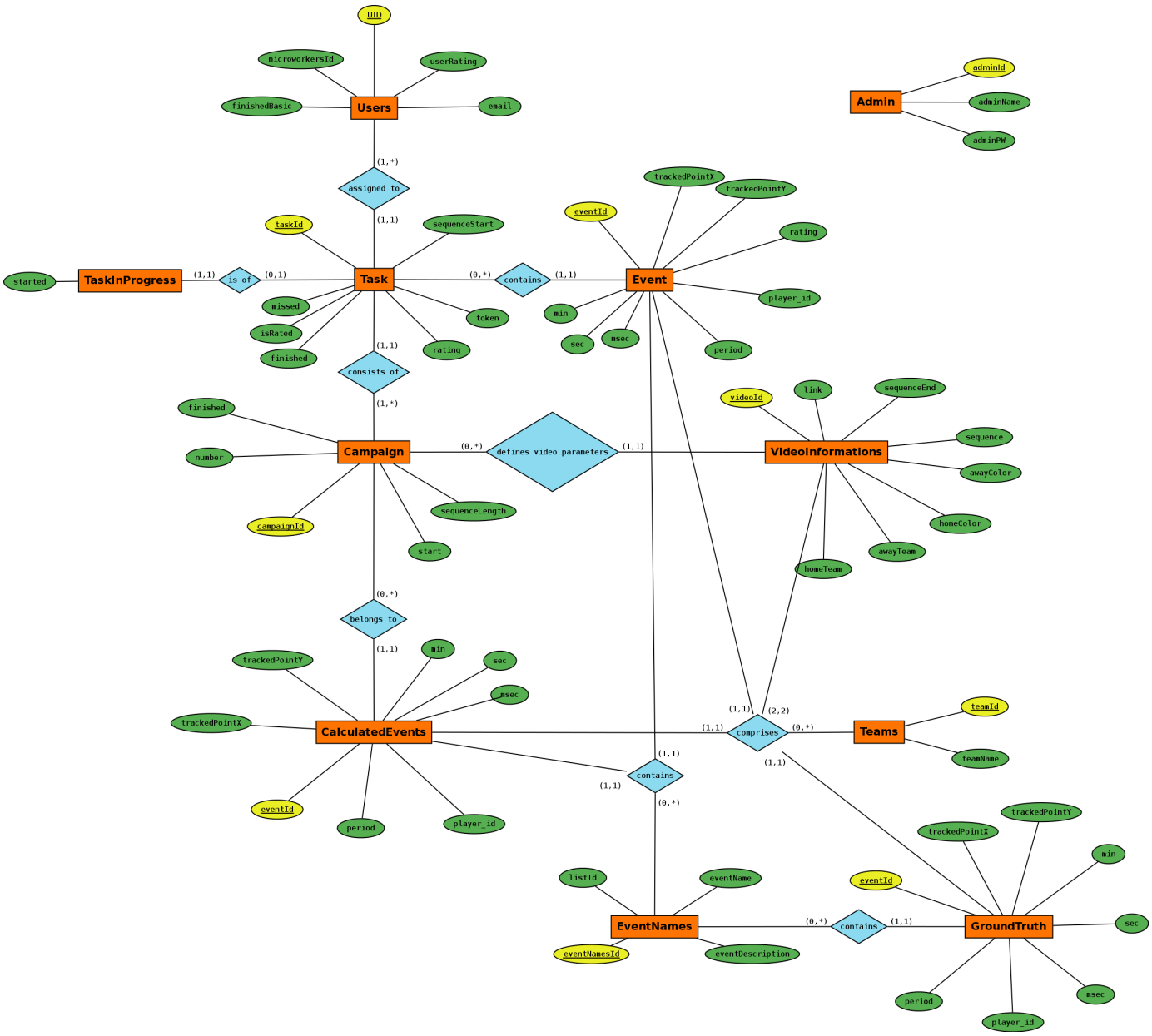


Figure 3.2: System data model

Admin

The *Admin*-entity contains an identifier (*AdminId*), which holds the ID stored as a session value. For the login system, this table also holds a username (*AdminName*) and an password (*AdminPw*), which is encrypted by SHA-512.

Teams

The entity "Team" stores the name of the team ($teamName$).

VideoInformations

For each team ($Teams$), the entity $VideoInformations$ stores information on the team that plays at home ($homeTeam$) and the team that plays away ($awayTeam$). To help to identify the teams in the video, the color of the home team ($homeColor$) and the away team's color ($awayColor$) are also stored. The information where a the video files is stored in the file system is saved in a relative path ($link$). To avoid creating campaigns with starting or ending values outside the video length, the $VideoInformations$ entity also stores the time the video starts ($sequence$) and the time the video file ends ($sequenceEnd$).

Campaign

Every tuple in the campaign refers to a video $VideoInformations$, which provides the information of the video for this campaign. Every campaign starts at a position ($start$) in the video and has a specified number of tasks ($number$) to be done. To get the information on how long every temporal sequence of a task need to be, the entity $Campaign$ contains a value for this task's length ($sequenceLength$).

Users

Every Microworkers user visiting the SportSense Crowdapp gets registered with a unique identifier (UID); the system of Microworkers also provides us their used identifier ($microworkersId$). To improve the system with a rating method, every user has a rating ($userRating$). To detect if the user has already been evaluated in a pretest to determine the confidence of his answers, this information is stored for every user ($finishedBasic$).

Task

Every existing campaign ($Campaign$) consists at least of one task. The entity $Task$ stores the informations of those tasks. If a user ($Users$) finished a task, the values of the entity $Task$ will be updated. Thereby, the number of missed tasks ($missed$) will be calculated and the task will be marked as finished ($finished$). In addition, the token generated ($token$) by the values provided by Microworkers is stored in this entity. A task also comprises a rating ($rating$) as calculated in Section 3.2, and a value to determine if the task is already rated ($isRated$).

Event, CalculatedEvents and GroundTruth

The three entities Event, CalculatedEvents and GroundTruth are very similar to each other. They all store some event information attributes, like the position data for x ($trackedPointX$) and y ($trackedPointY$), the attributes for the minute (min), second (sec) and milliseconds ($msec$), which together make up the time the event happens. All of these entities also contains the ID of the teams ($team\_id$) that performed the event ($type\_id$).

As compared to CalculatedEvents and GroundTruth, the Events entity also contains an attribute holding the event's rating (*rating*).

EventNames

In the *EventNames* entity, all captured events of the corresponding task are saved, containing the values described in the Table 3.1. All events have a unique identifier with the primary key. This key's values consist of the values of Manchester City's dataset. To group them more intuitive, an attribute to order (*ListId*) them on the system was added. Every event also holds a name (*eventsName*) and a description (*eventDescription*).

TaskInProgress

This entity helps the system to determine whether a task has started or not. The starting time of a task (*started*) is saved. After a specific time, these tasks are deleted in the entity TaskInProgress if they are not finished with the use of an event scheduler.

3.2 Rating System

Because it is not known how good the data of user inputs are, a method to establishing the correctness of data has to be determined. Therefore, all users are first evaluated using a ground truth to establish a first reliability rating.
After this first evaluation, users are allowed to work on real data. The data entered in these real tasks will be used to calculated a new basis out of all entered events with respect to the corresponding user's rating. For all users who have participated in the campaign, the rating is calculated with the new basis.
To increase the quality of the data gathered from the Microworkers, a rating system is introduced to check the quality of their given answers.

3.2.1 Events

To specify the events that should be captured, the existing dataset serves as an orientation.

Event name	**Event description**
Pass	When the player in possession kicks the ball to a teammate. Passes can be long or short but must remain within the field of play.
Pass - Offside	A player is in an offside position if: - He in the opposition half. - He in front of the ball. - Have fewer than two opposition players between himself and the goal line when the ball is played to him by a teammate. The goalkeeper counts as an opposing player in this instance.
Take on	A take on is an interception where the intercepting player keeps the ball.
Foul	A foul is an unfair act by a player, which is deemed by the referee.
Out	When the whole of the ball goes over the whole of the sideline or goal line.
Corner Kick	A kick taken by the attacking team that is earned when the defending team puts the ball out of play behind the goal line.
Tackle	When a player without the ball dispossesses an opponent with it.
Interception	Intercepting the ball is winning possession of the ball as it is being passed from one opposition player to another.
Clearance	When a defending player clears the ball out of danger. The purpose of a clearance is to relieve the pressure being exerted by the attacking team.
Shot off target	When an attacking player kicks the ball towards the goal in an effort to score and the ball misses the goal.
Post	When the referee chooses to allow play to continue if an offense has been made against the team in possession of the ball, he enforces the advantage rule. He does this if the attacking team would be unnecessarily penalized if play were to be stopped.
Shot on target	When an attacking player kicks the ball towards goal in an effort to score and the ball flies in direction of the target.
Goal	A goal is scored when the entire ball crosses the whole of the goal line between the goalposts and the crossbar.
Card	Used by a referee to enforce discipline in a match. A yellow card is a warning, and a player collecting two of these will be given a red card and sent from the field of play. A straight red card can be produced by a referee for a bad offense such as serious foul play or a professional foul.
Pass - Ball recovery	When the player in possession kicks the ball to a teammate and the other team retains it.
Pass - Ball touch	When the player in possession kicks the ball to a teammate by touching it only once.

Table 3.1: Events used in the system with the appropriate description, (adapted from[6])

[6] http://worldsoccer.about.com

To reduce the difficulty level, not all of those events are used here. The events used in the system are marked with a gray background. This was done because similar events, for example, an interception and a clearance, are to close to each other and will produce too many errors, especially by lay people evaluating football.

3.2.2 Position

The input field to let users enter position data looks like Figure 3.3 below.

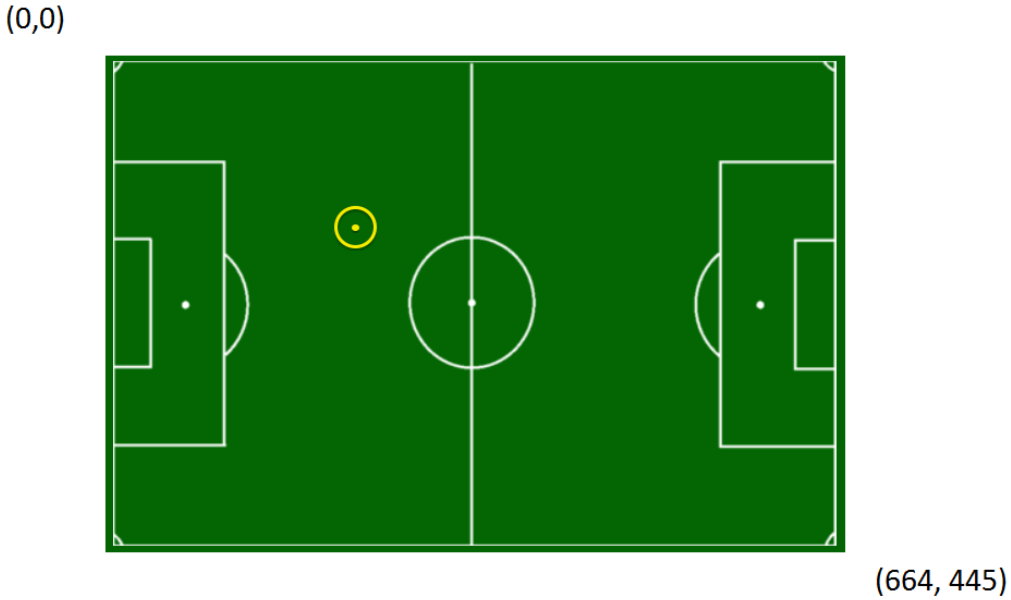


Figure 3.3: Soccer field used in SportSense

Where the x-coordinate is 664 and the y-coordinate 445 units long. The acceptable value to rate the users will be 40 units, which is in the diagonal distance

$$\frac{40}{\sqrt{664^2 + 445^2}} * 100 = 5\%$$

Considering that the size of a soccer field is about 90×45 meters, 5% of the diagonal length would be about five meters long.
The yellow point and circle indicates where a user set the position. His radius is 20 units, which is at the same time the $\delta_{position}$, which is accepted and rated with 0.

3.2.3 Userrating

All users start with an initial rating of 0. If a task is finished the rating of the corresponding user will be rated with his old rating plus the new one. This entails that a user's rating is:

$$R_{User} = \sum_{n} R_{Task} \tag{3.1}$$

where n is the number of tasks per user.

3.2.4 Rating of the task

With the rating of all events R_{Event} in a task, the rating for the task will be calculated by

$$R_{Task} = \sum_{n} R_{Event} \tag{3.2}$$

where n is the number of events per task.

3.2.5 Rating of the event

If a user's rating δ_{user} is smaller than -1, he will receive a warning. If the user's rating is smaller than $\delta_{user} \leq -1.5$, the user will not be able to do any more tasks in the SportSense Crowdapp.
Every input will be rated using a ground truth at the beginning. This rating of the event is defined as the sum of the rating of all dimensions of an event, as we show in Equation 3.3.

This section defines all values of the basis without a tilde ($\backsim$) and all user-entered data points with a tilde.

$$\begin{gathered} R_{Event} = \frac{R_T + R_P + R_E + R_t}{4} \\ R_{Event} \in [0,1] \end{gathered} \tag{3.3}$$

where R_T is the rating of time, R_P is the rating of the position, R_E is the rating of the event and, R_t is the rating of the team. The parts of the total rating will be discussed in the following.

Rating of the team $\mathbf{R}_t$

To get the value for the team R_t, we compare the entered team's ID against the basis team identifier.

$$R_t = \begin{cases} 1 & \text{if team} = \widetilde{team} \\ 0 & \text{if team} = \widetilde{team} \end{cases} \tag{3.4}$$

Rating of the event $\mathbf{R}_E$

To evaluate if the user entered the right or wrong event, we compare his event ID with the basis events identifiers.

$$R_E = \begin{cases} 1 & \text{if event } = \widetilde{event} \\ 0 & \text{if event } = \widetilde{event} \end{cases} \tag{3.5}$$

Rating of the position $\mathbf{R}_P$

A measure for the distance of the user data is given by position X and position Y. The rating on the position data is done by calculating the Euclidean distance (L2) between the point set by the user and the point given in the ground truth.

$$D_{points} = \sqrt{(X - \widetilde{X})^2 + (Y - \widetilde{Y})^2} \tag{3.6}$$

If the distance between the basis($\widetilde{X}$, $\widetilde{Y}$) and the data entered by the user (X, Y) $D_{position}$ is less than the threshold value $\delta_{position} = 20$, the rating is squared to reward users, adding a small positional difference.

$$R_P = (\frac{D_{points}}{\delta_{position}})^2 \tag{3.7}$$

if the distance t is bigger than $\delta_{position}$ the rating of the distance is calculated the following linear function:

$$R_P = \frac{D_{points}}{\delta_{position}} \tag{3.8}$$

With Equations 3.7 and 3.8, the rating for the distances will result in:

$$R_P = \begin{cases} \frac{D_{points}}{\delta_{position}} & D_{points} > \delta_{position} \\ (\frac{D_{points}}{\delta_{position}})^2 & D_{points} \leq \delta_{position} \end{cases} \tag{3.9}$$

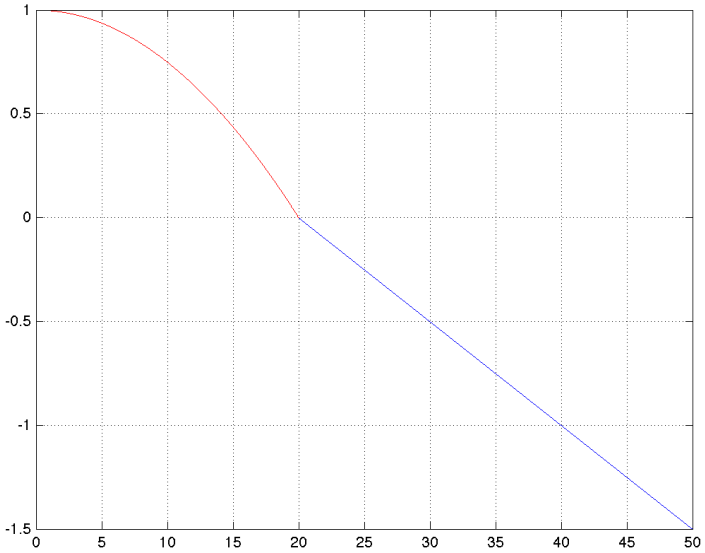


Figure 3.4: Distance calculation of position

Figure 3.4 displays the function R_P graphically. With this differentiation of the distance calculation, the rating of position of points with a small distance to the ground truth points

can be kept small, because it is almost impossible to enter the exactly same position twice, and all points with a bigger position distance than $\delta_{position}$ are rated linear.

Rating of the time $\mathbf{R}_T$

To calculate the rating for the difference in the time compared to the ground truth, sharp boundaries are set for the accepted values. This gives:

$$R_T = \begin{cases} 1 & if D_{time} = 0.0s \\ 0 & if D_{time} = 0.5s \\ -1 & if D_{time} = 1.0s \end{cases} \tag{3.10}$$

Where the distance between the times is calculated by:

$$D_{time} = |time| - |\widetilde{time}|$$

To calculate the rating of the time for a point against the ground truth point, a linear function:

$$R_T = 1 - 2 * D_{time} \tag{3.11}$$

3.3 Data Integration

By using the user's rating as calculated by the pretest in Section 3.2.3, a new basis is calculated with the entered data. This section presents two different methods to calculate the basis. In Section 3.3.2 we first introduce an integration and cleaning method based on the unweighted pair group method with arithmetic mean (UPGMA) algorithm. Section 3.3.3 presents a density-based clustering algorithm.

3.3.1 Distance calculation

For both algorithms, the difference between two entered events needs to be calculated. The mathematical approach to calculate the difference between two data-points in the system will be described. Each data-point is considered as a vector in a five-dimensional space, with the dimensions time, position of X and Y, event and team. For each vector, the following distance functions are defined.
The difference between two entries ($E1$, $E2$) is calculated by the arithmetic mean over all attributes of the points:

$$D_{E1,E2} = \frac{D_P + D_T + D_t + D_E}{4} \tag{3.12}$$

where D_T is the difference of the time, D_P is the difference of the position, D_E is the difference of the events, and D_t is the difference of the team. Those parts of our difference calculation are described hereafter.

Where the respective distances are calculated by:

Difference in positions $\mathbf{D}_P$

$$D_P = \frac{\sqrt{(X_{E1} - X_{E2})^2 + (Y_{E2} - Y_{E2})^2}}{\delta_{position}}$$

Difference in time $\mathbf{D}_T$

$$D_T = |time_{E1} - time_{E2}|$$

Difference in teams $\mathbf{D}_t$

$$D_t = \begin{cases} 1 & \text{if team}_{E1} = \text{team}_{E2} \\ 0 & \text{if team}_{E1} = \text{team}_{E2} \end{cases}$$

Difference in events $\mathbf{D}_E$

$$D_E = \begin{cases} 1 & \text{if team}_{E1} = \text{team}_{E2} \\ 0 & \text{if team}_{E1} \neq \text{team}_{E2} \end{cases}$$

With this difference calculation, two exact same events will end up with

$$E1 \equiv E2 \qquad \Rightarrow \qquad D_{E1,E2} = 0$$

3.3.2 Unweighted pair group method with arithmetic mean

The first way to calculate the bases uses the ideas of unweighted pair group method with arithmetic mean (UPGMA) [16]. The basic approach of the algorithm is to join the two clusters with the smallest distance to a new one, until only one cluster is left, which will be the root of the three. The distance between all points is calculated by the distance calculation between two points in this project, as described in Section 3.3.1.

UPGMA, is a hierarchical clustering algorithm starting at the bottom for creating phylogenetic trees, was initially designed for the use in protein electrophoresis. The requirements of the algorithm is the distance between each pair of the data set.

The algorithm starts with an unresolved tree and joins the two closest points until only one point is left.

This approach does not want to join all points until only one single point is left. A value is set for the maximum distance two points are allowed to have to join them. With this change, this will be stopped if the distance between two clusters is bigger than the specified threshold.

The new point is calculated by two points $E1$ and $E2$, as in the following equations. All variables with a circumflex(ˆ) belong to the second event and all variables without one belong to the first event.

The variable R is the rating of the point, t stands for the team, T for the time, and X and Y for the corresponding position of each event.

new positionX:

$$positionX_{E1,E2} = \frac{X * R + \hat{X} * \hat{R}}{R + \hat{R}}$$

new positionY:

$$positionY_{E1,E2} = \frac{Y * R + \hat{Y} * \hat{R}}{R + \hat{R}}$$

new time:

$$time_{E1,E2} = \frac{T * R + \hat{T} * \hat{R}}{R + \hat{R}}$$

new team:

$$team_{E1,E2} = \begin{cases} t & \text{if } R \geq \hat{R} \\ \hat{t} & \text{if } R < \hat{R} \end{cases}$$

new event:

$$event_{E1,E2} = \begin{cases} E & \text{if } R \geq \hat{R} \\ \hat{E} & \text{if } R < \hat{R} \end{cases}$$

After calculating the new row composed by two events, those events are deleted from the data list and the new event added. For this new event, the distance to all other events is again calculated. This is done until there are no more events with a distance smaller than one.

Algorithm

Listing 3.1 presents the pseudocode, showing the calculation of the UPGMA method. As described below, the distance between all points in the dataset *data* is calculated and stored in a matrix $distanceMatrix$. A counter array *counter* is also created, containing ones for the size of the dataset, because every data entry consists of exactly one point at the beginning. As long as $distanceMatrix$ exhibits values smaller than 1, the closest two data-points are joined to a new point. The closest ones are those ones with the smallest values in the distance matrix. After calculating the new point from those two points with respect to their rating, those two points points are deleted from *data* and the new point is added. The ratings of the two points is also removed and a new entry is added to the rating array containing the sum of the two points ratings. After joining to points, the *counter* also need to update by the amount of the two joined point's rating. At least, the new distance of the two joined points to all others is calculated and the $distanceMatrix$ updated.
In the end, all points existing in the data with a counter bigger than 4 are accepted, which means that all accepted points need to be calculated out of more than four points.

Listing 3.1 Algorithm joining all points similar two each other

```
1: FUNCTION calculatePoints(data, rating)
2:     distMatrix = getDistMatrix(data);
3:     counter = ones array with length of rating;
4:
5:     WHILE distMatrix contains elements ≤1
6:         [index1, index2] = smallest value in distMatrix;
7:
8:         newRow = calculate the new datapoint with the elements ...
               index1, index2;
9:         data[lastRow+1] = newRow;
10:
11:        rating[lastRow+1] = rating[index1] + rating[index2];
12:
13:        counter[lastRow+1] = counter[index1] + counter[index2];
14:
15:        newDist = calculate the distance of the newRow to all others;
16:        distMatrix[lastRow+1] = newDist;
17:
18:    END WHILE
19:
20:    RETURN data = data where appropriate counter≥4
21: END FUNCTION
```

To get the distance matrix, for each row the distance to all other rows in the dataset is calculated. The distance between two rows is calculated as described in Equation 3.12. After calculating all values, an upper triangular matrix is returned.

Listing 3.2 Algorithm to calculate the distance of two points

```
1: FUNCTION getDistMatrix(data)
2:     FOR I = 1 to length of data
3:         FOR J = I + 1 to length of data
4:             Dist = calculate the distance of point I and J;
5:             DistMatrix(I, J) = Dist;
6:         END FOR
7:     END FOR
8:     RETURN DistMatrix;
9: END FUNCTION
```

3.3.2.1 Discussion

The main problem of our UPGMA approach is that it gives more weight to the first point in the list; thus, when joining the two closest events, the event with the corresponding higher rating will prevail. However, this leads to a propagation of errors if the first entered event type or team with the bigger rating was wrong, because the new event joined out of two events gets also a rating that is the sum of the two old ratings.

3.3.3 DBSCAN

The second method is to use a cluster method that starts from an unknown distribution of the data. For this purpose, the DBSCAN algorithm is used. DBSCAN is a density-based algorithm to cluster the data and detect outliers. The idea of the DBSCAN algorithm is that for each point in a neighborhood with distance ϵ there has to be a minimum number of points($MinPoints$), to be accept as a cluster. DBSCAN is able to distinguish between different types of reachability.
In the DBSCAN algorithm, different kinds of points related to a cluster exist [14], as described below:

Directly density reachable

All points with a distance $D(p, q) \leq \epsilon$ are density reachable for point p.

Core and border points

If a point is density reachable, there are two possible states a point can get. A point is a core point if it satisfies that the point itself has more density reachable points than $MinPoints$. $\#D(p, q_i) \leq \epsilon \geq MinPoints$ where i is one element of the data $p_i \neq q$

Density-reachable

A point p is density-reachable from a point q if it there exists a chain of density connected points including the points p and q.

Density-connected

A point p is density-connected with a point q if there exists a point r such that p is density-reachable from r and also q is density reachable from r.

Cluster

A cluster with respect to ϵ and $MinPoints$ is a non-empty subset of data satisfying the conditions

- $\forall p, q$ if $p \in C$ and q is density recheable from p wrt. ϵ and $MinPoints$ than $q \in C$, where C is a cluster.
- $\forall p, q \in C$ the points p and q are density-connected wrt. ϵ and $MinPoints$.

Noise

A point is marked as noise if it do not belong to a cluster where clusters are built with respect to ϵ and $MinPoints$.

In our approach the calculation does not differentiate between density-reachable and density-connected points, because by setting a different ϵ and ϵ_2 for the density-connected points the number of density-connected points can be reduced.

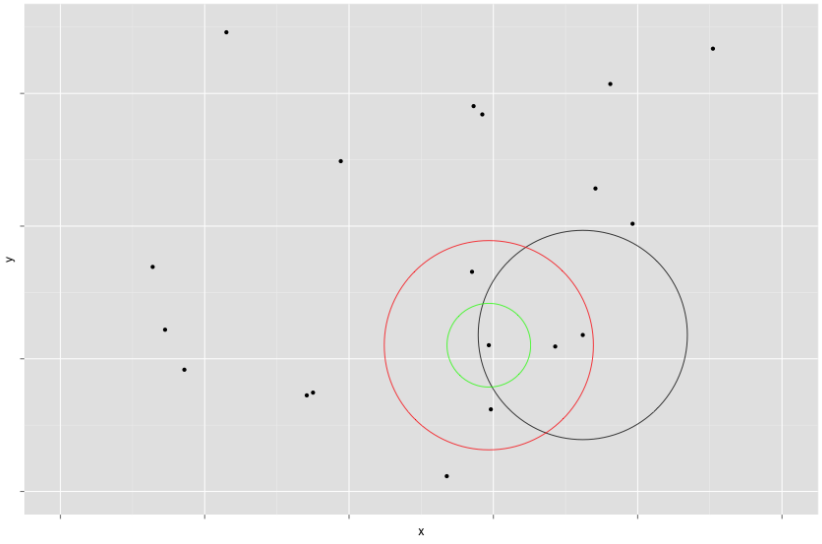


Figure 3.5: Reachability of DBSCAN

Figure 3.5 shows that, with decreasing the ϵ_1 towards ϵ_2 the chain can be brought to stop at some point. When starting with the first point in the black epsilon, one gets three density-reacheable points with it. If one would continue with ϵ_1, as illustrated with the red color, one would create a big chain with two new points points in it. By lowering the setting for ϵ_1 to ϵ_2 for density-reachable points, one can decrease the number of density-connected points. As previously, to calculate the distance of two points the calculation in Section 3.3.1 is used.

Algorithm

The clustering method with the DBSCAN starts by setting the cluster counter to 0. For each row in the dataset, the distance to all other rows in the dataset is calculated, starting starting with the first row. If there are more than the minimum number of points required to build a cluster, the cluster gets expanded and increase the cluster counter by one, to create a new cluster. If the number of indexes with a value smaller than ϵ_1, the data point is an outlier and is tagged as noise by setting its class to -1.
If the number of points similar to it is bigger than one and smaller than $minPoints$, this point is marked as not clustered.

Listing 3.3 Algorithm for the DBSCAN method

```
1: FUNCTION DBSCAN(data, MINPOINTS, EPS1, EPS2)
2:     clusterNumber = 0;
3:     FOR each row in the dataset data
4:         Dist = calculateDistance(row, data);
5:         ind = all Distances > EPS1;
6:         IF size(ind) > MINPOINTS
7:             class(ind) = clusterNumber;
8:             expandCluster(data, ind, clusterNumber, EPS2);
9:             clusterNumber++;
10:        ELSE IF size(ind) == 1
11:            class(i) = -1;
12:        ELSE
13:            class(i) = 0;
14:        END IF
15:    END LOOP
16:    RETURN class;
17: END FUCTION
```

To expand a cluster in Listing 3.3 we use the vector containing all values which belong to this cluster. For each index in the vector we calculate the distance to all other points. The new calculated distances will be compared to ϵ_2. If the number of values smaller than ϵ_2 is bigger than one, those indices are added to the same class. This is done recursively until there are no more points density-connected.

Listing 3.4 Algorithm to expand cluster

```
1: FUNCTION expandCluster(data, ind, clusterNumber, EPS2)
2:     WHILE Ind is not empty
3:         row = data(ind(1));
4:         Remove ind(1);
5:         dist = calculateDistance(row, data);
6:         ind1 = all Distances > EPS2;
7:         IF size(ind1) > 1
```

```
 8:              class(ind1) = clusterNumber;
 9:              expandCluster(data, ind1, clusterNumber, EPS2);
10:          END IF
11:      END WHILE
12: END FUNCTION
```

The function *calculateDistance* in Algorithm 3.4 calculates the distance between one row and all other data rows in the dataset. The distances are calculated as described in Section 3.3.1. After calculating all distances, the arithmetic mean of all distances in all dimensions is calculated.

For each data point, the introduced algorithm calculates its corresponding class. Furthermore, the algorithm also marks the points that are considered as noise. To calculate the base, all points marked as noise are removed. The position and time of the new base is then calculated by the arithmetic mean of all positions and times corresponding to the cluster with respect to the workers rating. The majority of all events in the cluster set the event and team value of each basis.

3.3.3.1 Discussion

The problem of the DBSCAN algorithm is that it also accept density-connected points. Those density-connected points are specially denoted in an array containing the types of a point. This number of density-connected points can be reduced by ϵ_2. The average runtime complexity of DBSCAN is $O(n \log n)$, where n is the data set length. But the complexity is not an disqualifier in our application, because the clustering and calculation of these basis events happens offline in perspective to the user entering the data. A disadvantage of DBSCAN is that it is not deterministic, border points can be reached from multiple clusters, and the calculation of the clusters depends on the chronology of the entered points.

4

Implementation

This project is implemented in PHP 5, which can be run in a web browser. The decision to create the thesis on a browser system was made because users do not have to install any software on their machines and it reduces the time to solve a task. Furthermore, it also allows to make changes to the code without a need for users to re-download it. Finally, PHP is chosen because of his ability to access the UNIX command line tools, which are needed in this project and because of its big community.

The software runs on an Ubuntu server distribution with version 14.04, which has a MySQL database installed. This choice has been made because one can access the distribution and install some frameworks as FFmpeg, Cron and Octave.

To display the video to the users, the HTML5 video element is used, with the controls disabled. The time display video could be set with JavaScript. This has the advantage that the whole video file does not need to be loaded. Getting the current time of the video enables storing the events' temporal information in the database.

Because PHP is inappropriate to perform matrix calculations, a Matlab script is used to calculate the new bases of the points described in Section 3.3.1. The Matlab scripts run from the PHP command line framework using Octave [17], an interpreter language for numerical calculations.

Events

To remove the tasks that have been started and not finished, and are stored in the table of the database system, the MySQL Event Scheduler [18] is used. This event checks every 20 minutes if events older than half an hour exist on database entity holding the tasks started but not finished. If there are some, they will be deleted by the Event Scheduler, which frees up the tasks again for users.

Image handling

To create the images for the video player extension, FFmpeg [19] is used, which is an open source Linux-based multimedia framework able to manipulate multimedia data. This distribution allows creating images of a video file at a specified sequence and in a specified interval.
With FFmpeg, different variables are set as the start and end times of the sequence where FFmpeg should generate images. There is also a variable to set the interval in which the frames should be created, and by setting the images to a smaller size than the videos size the storage space can be reduced. For the purposes of this project, the image size is set to 12 KB.

To store the images created for every task and time, a new folder is used with the token as the name. Taking the token as the folder name ensures that all folder names are unique and the right images can be selected for every task.
To prevent the file system from overloading, Cron [20] is used, which allows one to call a script every half hour, which deletes every folder older than one hour.

Microworkers

For the experiments, an international campaign in the category “search, click and engage” has been created on Microworkers. In the first experiment, the estimated time to solve the task has been set to 10 minutes.

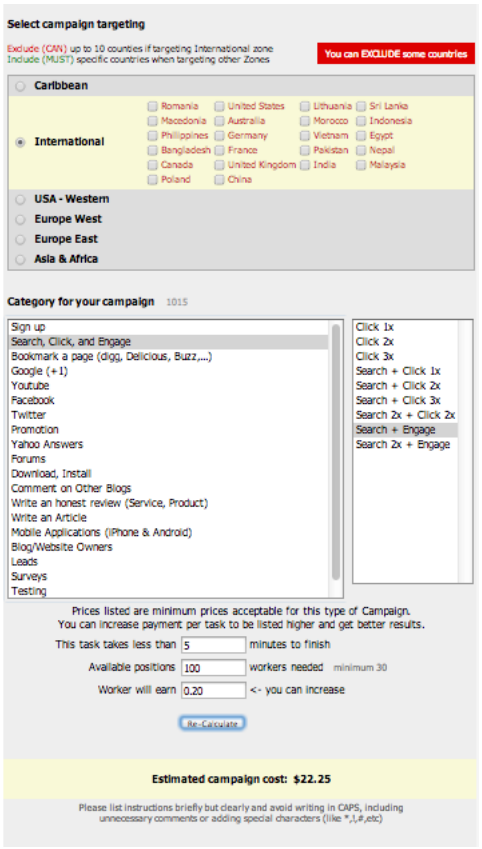


Figure 4.1: First part of Microworkers basic campaign settings

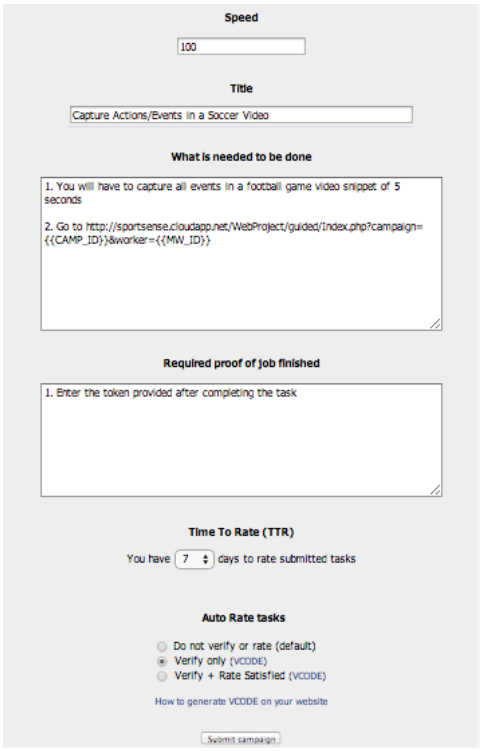


Figure 4.2: Second part of the Microworkers basic campaign settings

Figures 4.1 and 4.2 show how the campaign was created on Microworkers. Based on the results of Google Analytics as shown in Section 5.4, in the final experiments, the time workers need to finish a task was reduced to 5 minutes in further tests.

Interface server database

To connect PHP with the database, PHP Data Objects [21] is used. PDO is an object-oriented interface between the database and PHP. The advantage of PDO is that it does not have to be rewritten to connect to another database engine. The only difference between the database systems is that the parameter that defines the driver must be changed when creating the instance of the constructor.

5 Evaluation

To evaluate our system we run some tests on Microworkers. In this section we introduce the settings and take a look at the results from our tests.

5.1 First Tests

To get a first impression how good the answers from Microworkers are we wanted to create first a basic system to test the workers and to learn where the problems in the system are.

5.1.1 Setting

The first test evaluates the system in a basic setting. For this purpose, we used the preliminary layout as described in Section 2.1. To explain the task in more detail to the users, a tutorial document (Appendix A) was created, which is linked to the campaign description on Microworkers.

In this first system, users only had to select the time, the team, and the event for all events in a 5 second video snippet, where all users got a different sequence. The events have been reduced to:

- Pass
- Foul
- Out
- Corner Kick
- Shot off target
- Shot on target
- Goal
- Card

Data was gathered from 84 users of the Microworkers platform, where each second of the video is annotated by 5 workers.

5.1.2 Results

To evaluate the results of the first test, the data entered by the workers to the video file was compared manually. With those results we decided if a user is accepted and, thus, receives his payment. It took about 20 hours to be finished with all 55 task.

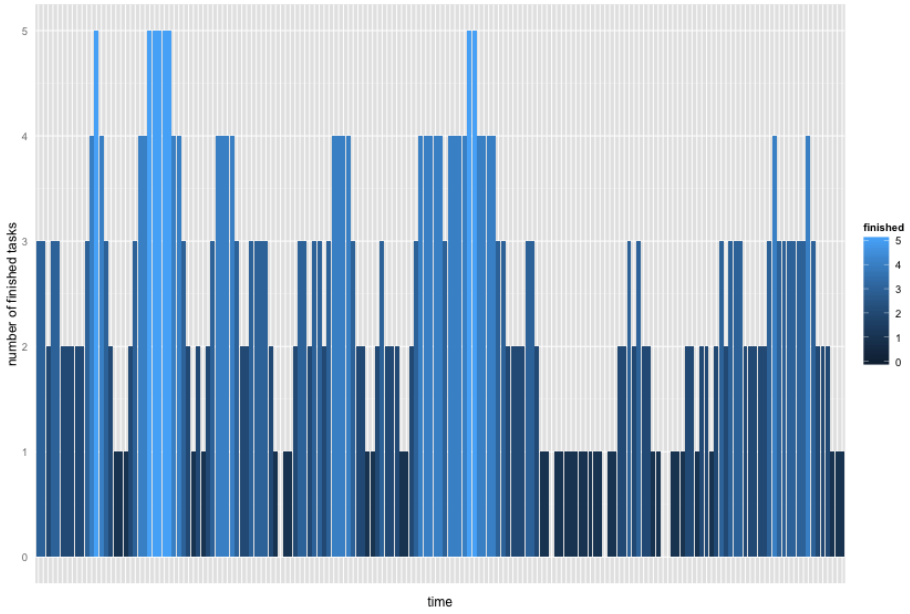


Figure 5.1: Number of finished tasks per second of first test

Figure 5.1 shows there are parts of the sequence that are not annotated, because some users start a task and do not finish it. Of all 169 tasks received, only 84 were finished, which is almost 50%.
Of those 84 finished tasks, 53 were accepted and the other 31 tasks were not; this is about 37% that were not accepted. The tasks were not accepted for various reasons: First, some completed tasks did not contain any events but were marked by the worker as finished. Second, there were tasks that contained completely wrong data, where users missundestood the task and entered just events for almost every second. Third, many workers were just entering one or two events where should be more. Fourth, the data was in some cases very inaccurate regarding the timing of the event.

5.1.3 Conclusion

As a consequence of the first evaluation, changes were made to the system and the user interface:
First, the layout was changed to a more guided version and a prompt was added telling workers to check the sequence for more events, because many users had not entered all events in sequence, mostly only entered the first one.
Second, the system for generating the tasks was changed so that users can be previously assigned to them, because, as can be seen in Figure 5.1, nearly all video seconds do not

consist of the required five user data entries.
Third, the explanation sheet was removed and a tutorial movie explaining the task was added instead.

5.2 Pretest

For further tests, the layout was changed to the second layout approach described in Section 2.2. A PDF file explaining the system was discarded and the instructions were integrated into a tutorial video, which is displayed as an overlay at first when entering the web page.

5.2.1 Setting

The second test introduced a new user interface more guided than the first one, as described in Section 2.2.
Compared to the first test, workers now also have to enter a positional data and some more events were added to the event list.

To explain the layout and the task they have to do, every user will see a tutorial video at the beginning. In this video, the layout and the task of every user are explained.
To get a first rating of every user, all users have to do a pretest against a ground truth, the pretest uses the same sequence for all users.
With the test against the ground truth the rating of each event and every users first task is calculated, as described in Section 3.2.

5.2.2 Results

Out of 709 users who participated in the pretest, 278 scored well enough to be allowed to enter data in the true test.

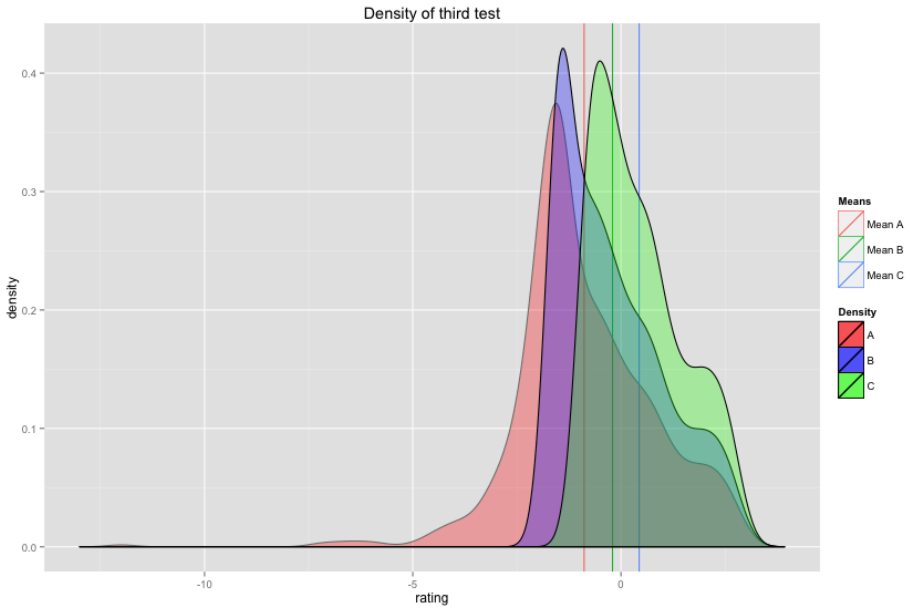


Figure 5.2: Densities of the pretest with a different minimum of rating (mean of A = −0.88, mean of B = −0.20 and the mean of C is 0.44)

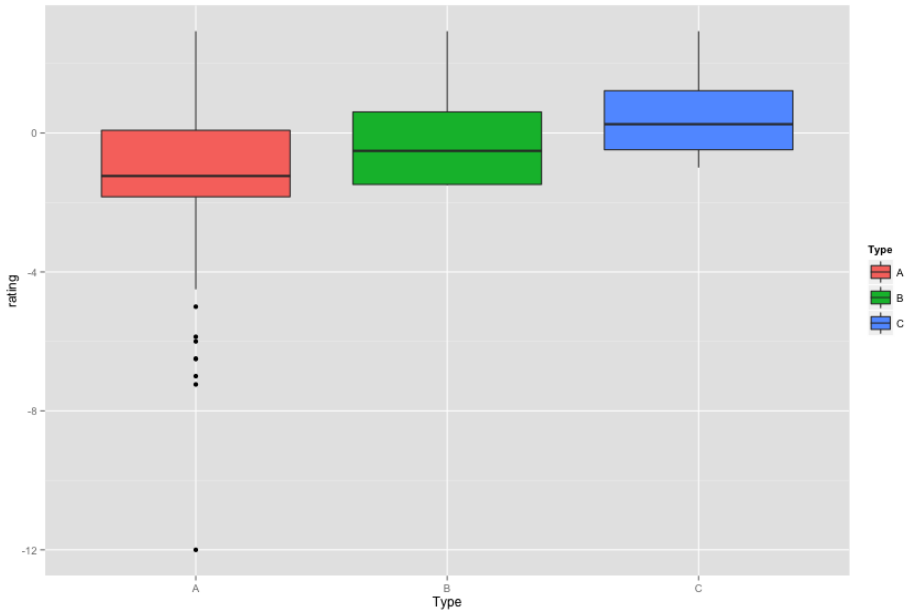


Figure 5.3: Boxplot with task-ratings of pretest

Figures 5.2 and 5.3 show the data of all finished tasks: B shows all tasks of the pretest where the rating is larger than -1.5, which are all accepted tasks. C shows all values larger than -1; users with a pretest rating greater than -1 are accepted to do further tasks.

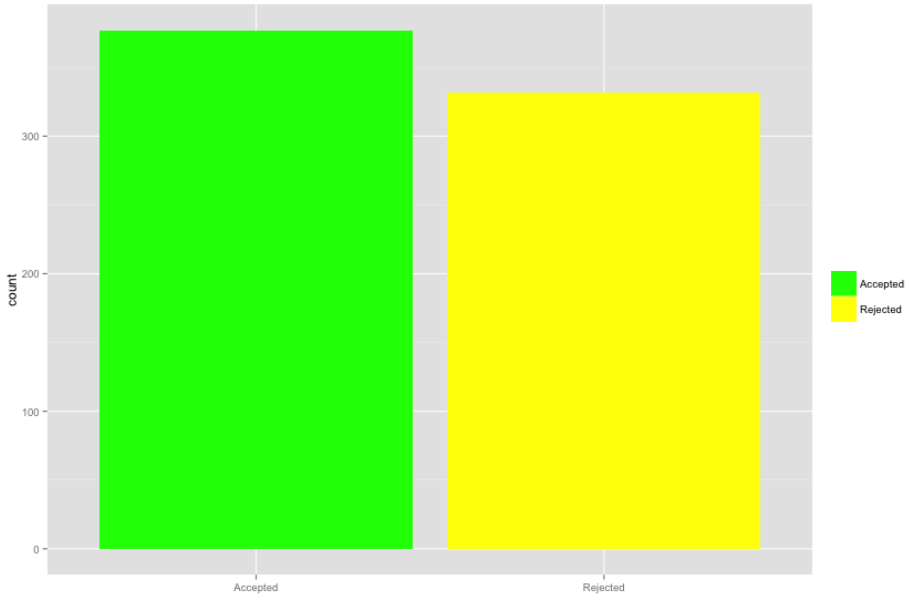


Figure 5.4: Accepted and rejected number of tasks in the pretest

As shown in Figure 5.4, 377 tasks out of 709 were accepted, which is an acceptance rate of 53%. The average rating of those 377 tasks was 0.22 and the average rating of all tasks with

a rating > 0 was 1.17.

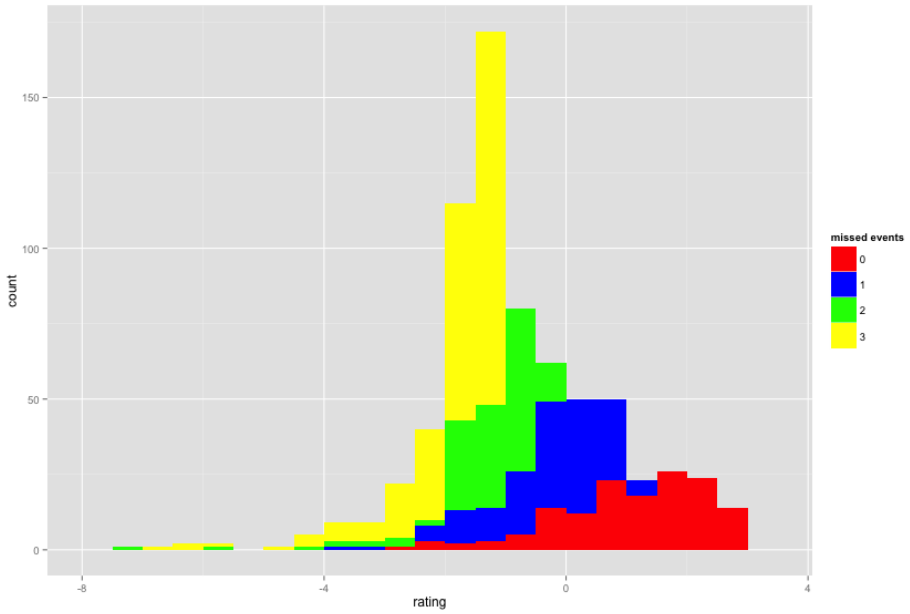


Figure 5.5: Missed events in the pretest

Comparing the missed events with the rating, it can be seen that even if people miss 2 events they can get a rating of 0 which is the average of all accepted event ratings and neither good nor bad. However, even with 0 missed events out of 3, people can get a rating smaller than -1, those people with such a rating and 0 missed events just worked really inexactly. If we look at the number of tasks considering the missed events with a rating ≥ 0, there are 117 tasks with 0 missed events and 70 tasks with missed events $= 1$. Considering all tasks with a rating ≥ 1, only $5,75\%$ have one missed event, all others are without any missed events.

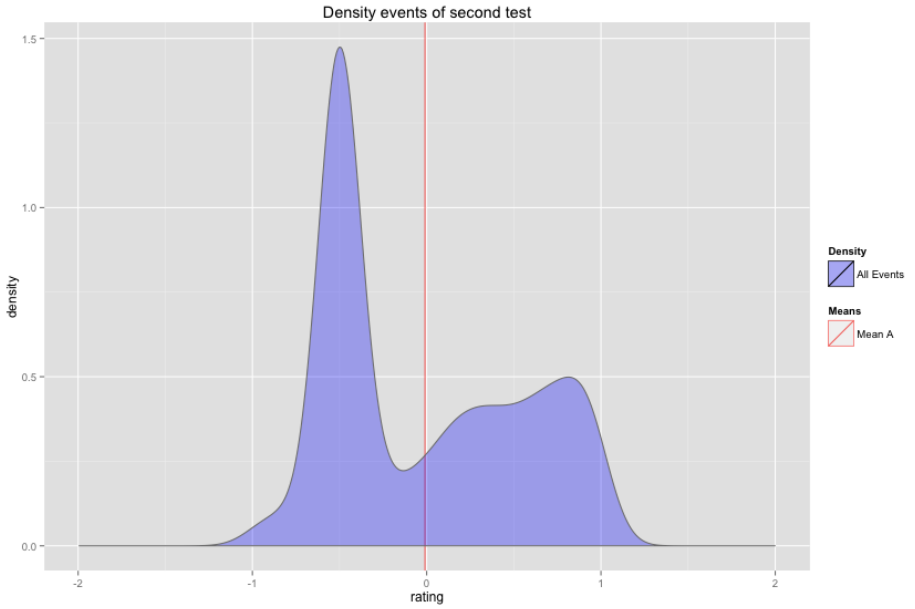


Figure 5.6: Density of all events in the pretest (mean of A = −0.009)

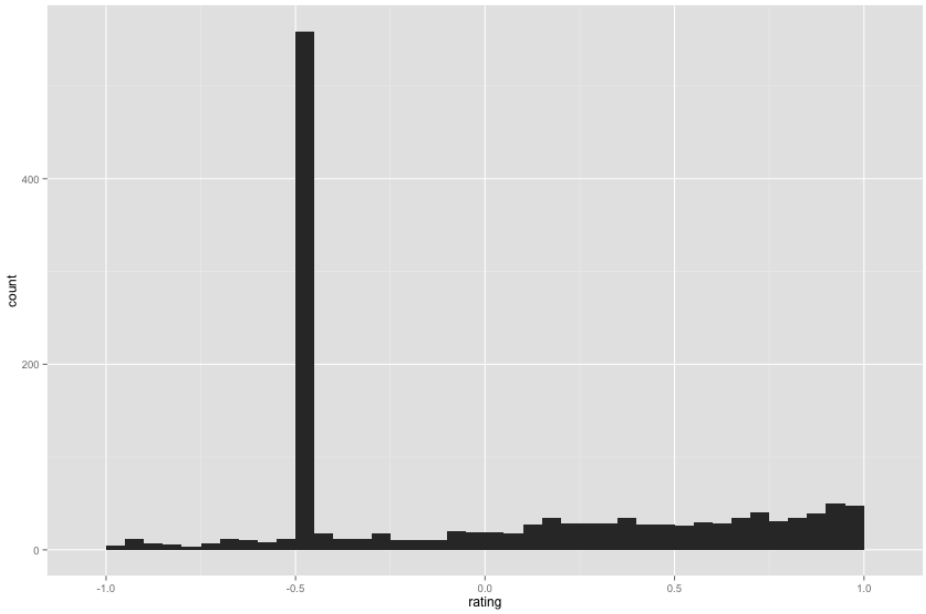


Figure 5.7: Distribution of the rating of pretest

The total number of added events in these 709 tasks was 1406, which means that the average task contained about 1.98 events, whereas it should contain 3 events. Looking at the distribution of the ratings of the pretest, there are 550 added events with a rating of −0.5, which means that the added event did not correspond to any event in this sequence. This means that 39.12% of all added events are wrong.

5.2.3 Conclusion

This second test showed that almost half of the persons had a rating of $\leq$ -1.5. This means that they are not allowed to do further tasks. Those users have not received their token after finishing their first task and therefore they were not paid. According to the introduced rating system, users that score this badly are no longer allowed to participate in the tasks involving the true data and will, thus, not be payed for wrong work.
Figure 5.2 shows that, when excluding users with bad rating from further tasks, the average rating of users in the task increases from -0.88 to 0.44, which is a difference of 1.32.
It is also discovered that there is a correlation between the number of missed events and the rating. All people who missed three events will not be able to do more tasks. In addition, 42 of 143 people with two missed events will also be disqualified from doing more tasks, because their entered values scored too low.

5.3 Evaluation of Data Integration and Cleaning

In this part of the evaluation, the system running the DBSCAN and UPGMA algorithm is tested and the performance output is compared with manually entered data.
For more meaningful results, only sequences of the game without any replay clips or slow motion views were accepted for all the tests.

5.3.1 First test

In the first test, a sequence of 50.5 seconds was chosen, which will result in 101 tasks. The sequence started at minute 33 and 50 seconds and ended with the beginning of the last task at 34:40.

Results

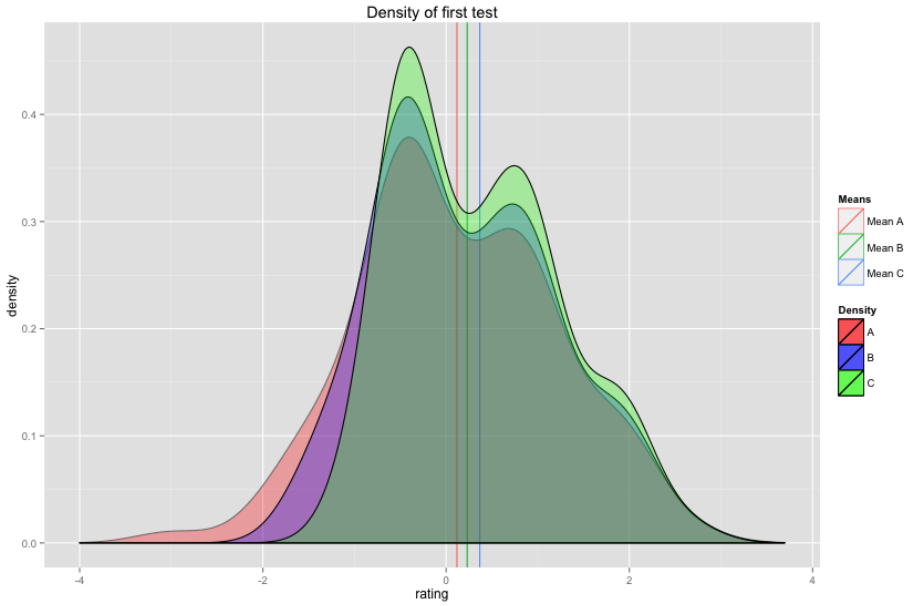


Figure 5.8: Densities of the first with a different minimum of rating (mean of A is 0.12, the mean of B is 0.23 and the mean of C = 0.37)

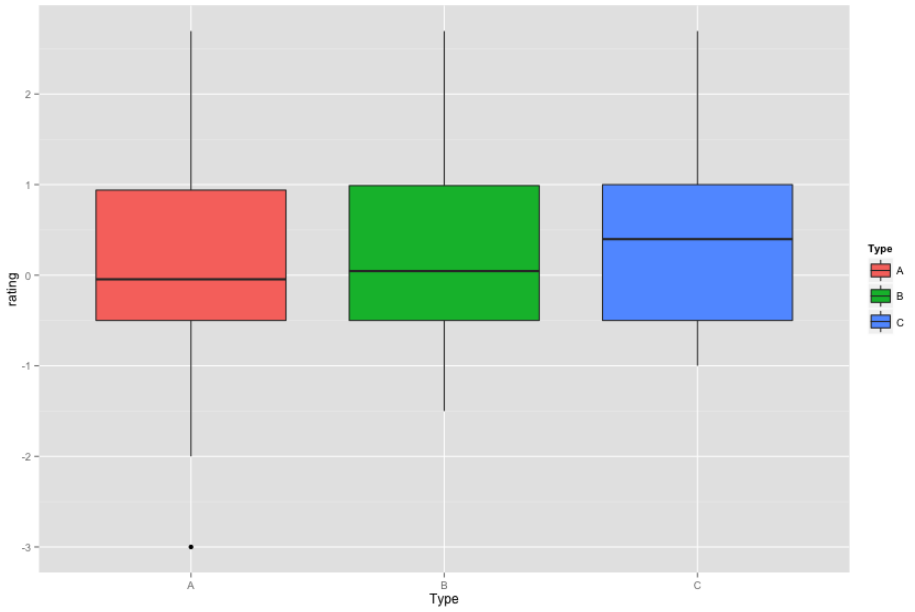


Figure 5.9: Boxplot with task-ratings of first test

As shown in the figures below, all means in this second campaign ≥ 0.

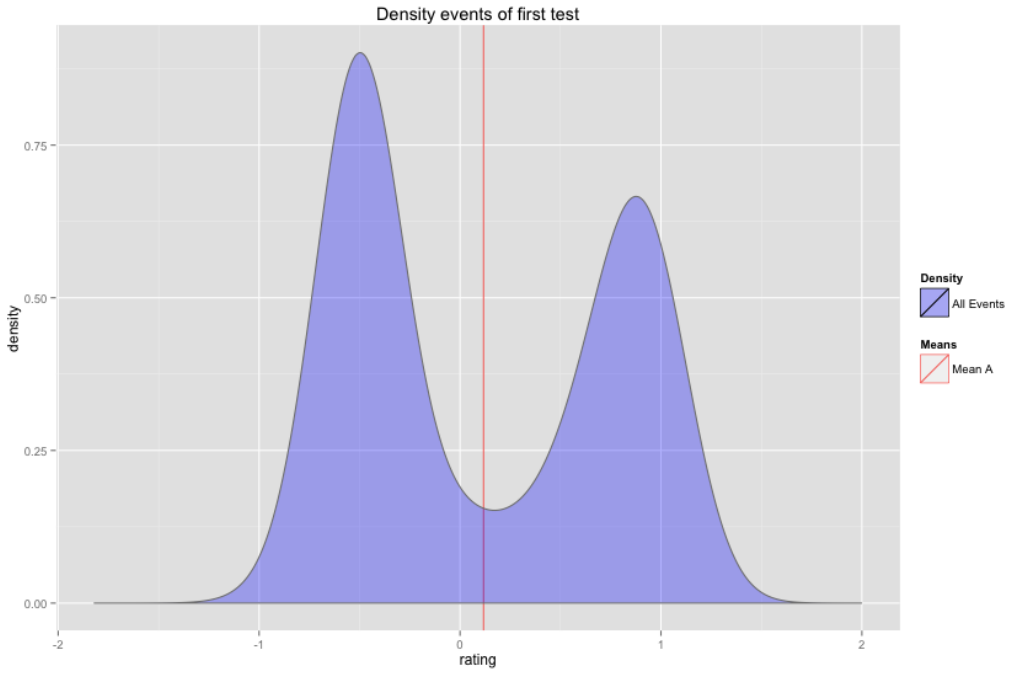


Figure 5.10: Density of all events in the first test (mean of A is 0.12)

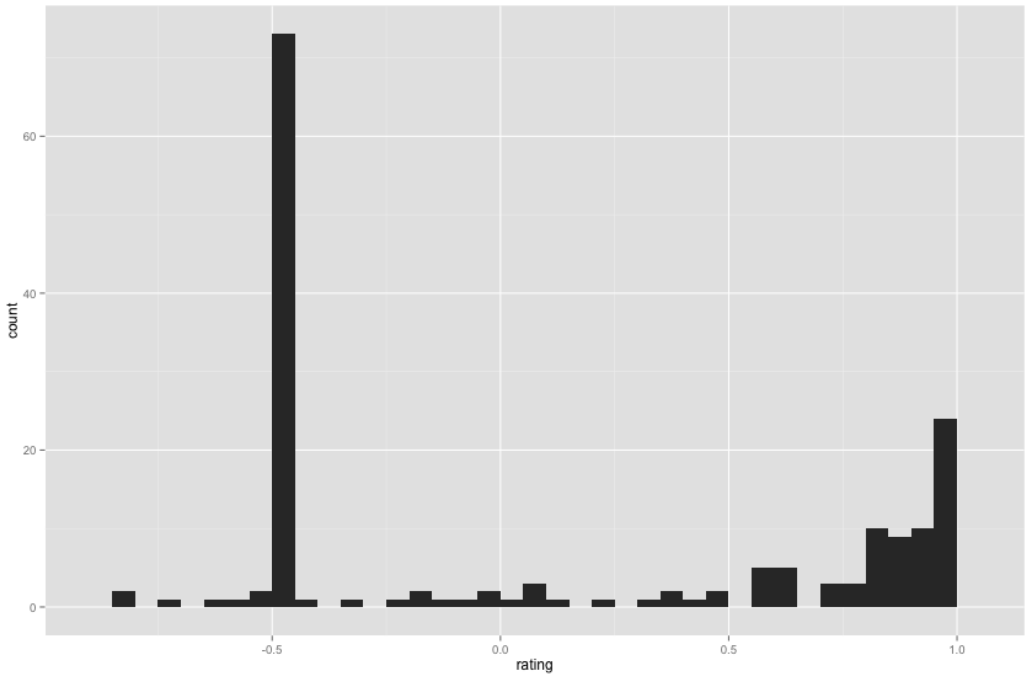


Figure 5.11: Histogram, distribution of event-ratings

In Figures 5.10 and 5.11, the total number of added events in this second campaign is 170, which results in an average of 1.7 events per task. We have 73 events with a rating of -0.5. This are 42.9% events, which cannot be assigned to any calculated events by DBSCAN. Looking at the difference between the points calculated with DBSCAN and the entered points, the following results were obtained.

Difference			
position	**time**	**event**	**team**
12,74153052	0,1	0	0
34,15673433	0,1	0	0
17,17876014	0,1	0	0
5,580197129	0,1	0	0
15,30154567	0,3	0	0
7,011305157	0,2	0	0
61,33984023	0,1	0	0
15,49588978	0,1	0	0
42,52275273	0,2	0	0
29,19160496	0,1	0	0
19,57876911	0,2	0	0
44,82389095	0,1	0	0

Table 5.1: Difference between the calculated events and the ground truth with DBSCAN

The average variance in the time is 0.15; in the position, the average distance is 25.4; and there are no faults in the event and the team's ID. Looking at the red-colored positions, some values had a distance of ≥ 40; these values are not really good, especially the value in the seventh row has a huge difference of 61.34 units, which is 7.6% of the whole field's distance. However, there have been some missing events. The DBSCAN algorithm missed calculating 6 points and instead calculated one wrong point.
For comparing the UPGMA method to the DBSCAN method, the calculated base of UPGMA is given in the table below.

Difference			
position	**time**	**event**	**team**
12,45540044	0,1	0	0
32,64439462	0,1	0	0
61,27385168	0,5	0	0
39,77870033	1,3	0	0
29,58215847	0,4	0	0
14,34155152	0	0	0
36,55601866	0,2	0	0
2,956010825	0	0	0
44,90069599	0,4	0	0
17,27061088	0,5	0	0

Table 5.2: Difference between the calculated events and the ground truth with UPGMA

Table 5.2 shows the distance between the points calculated with UPGMA and the manual data. Calculating the events, UPGMA calculated 10 data points calculated, whereas DBSCAN calculated 12. In addition, 3 wrong events have been calculated. The average positional distance of UPGMA is 36.60 and the average difference in time is 0.74 seconds.

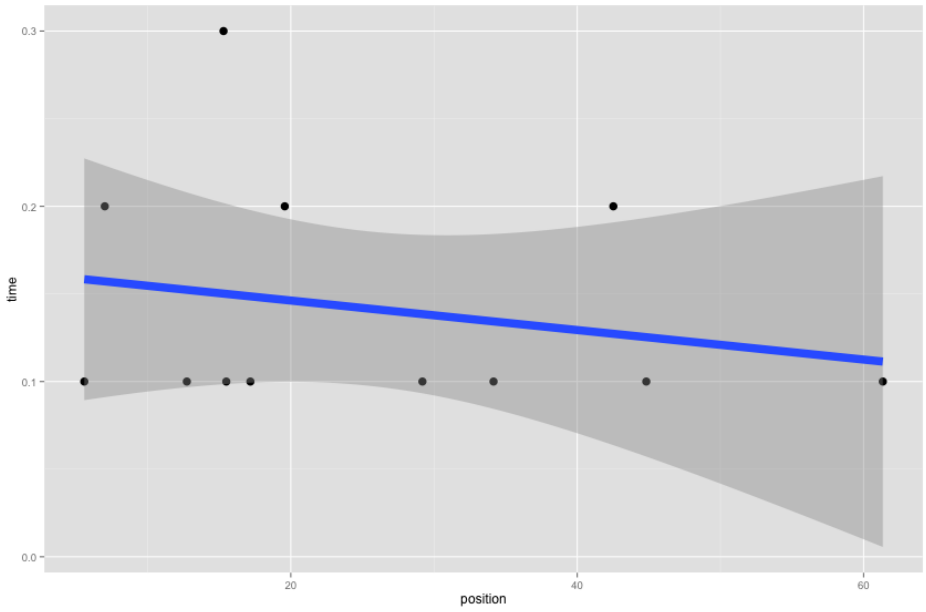


Figure 5.12: Correlation of the position and the time error with DBSCAN

Figure 5.12 shows the distance in the time and the position for each point of the calculated and allocated data-point calculated with DBSCAN. If we look at the blue line we can see that the deviation of the time do not correlate to the deviation of the position.

Conclusion

A comparison of the means with the means of the pretest shows an increase of the mean of all ratings by 1.06 and of the mean of B by 0.5. This is especially significant, because both values, the mean of A and the mean of B could be raised from a negative value to a positive one. With the mean of C, all ratings ≥ -1 increased just by 0.2. However, this is because in the pretest every task had three events that ended up in a maximum rating of 3.0. Whereas, in this task every user got another task and if someone received a sequence without any events, the possible maximum rating is 1.0, also for tasks with one event; tasks with two events leads to a maximum value of 2.0. So the average value is smaller, because there have been 15 events in the sequence of 100 tasks where every task contains a tenth of the whole sequence, so the average event per task is only 1.5, which is also the average maximum rating over all tasks, hereunder we got a big increase of the rating of the users, because the mean of all ratings of section A,B and C increased where at the same time the average maximum rating was half of the maximum rating in the pretest.
A comparison of the two clustering methods against each other shows that DBSCAN produces a better output regarding the number of not calculated events, the number of miscalculated events, and the difference between the calculated events.
If we take a look at the distances between the calculated and our manually entered points we can see that we did not get such a big average distance error; it is only 3.18% of the whole field by calculating the base events by DBSCAN. However, there some missed events, and

those events do have different reasons why they are missing. One point is missing because in the first and last seconds of the analyzed sequence less than 10 task are created. The DBSCAN algorithm accepts clusters with more than four points. Another problem in the sequence is that some points exists that are too close to each other. In this sequence, there are three interceptions, where the ball gets to another player of the intercepting team. Those points have only a difference in the event types ID and will summarize to one single point with DBSCAN.

5.3.2 Second test

The evaluate the stability of the first standard test, a second campaign was created, which contained 150 tasks and was 75 seconds long. The sequence started at minute 6 and 43 seconds and also and did not contain any repetitions or slow motion clips.

Results

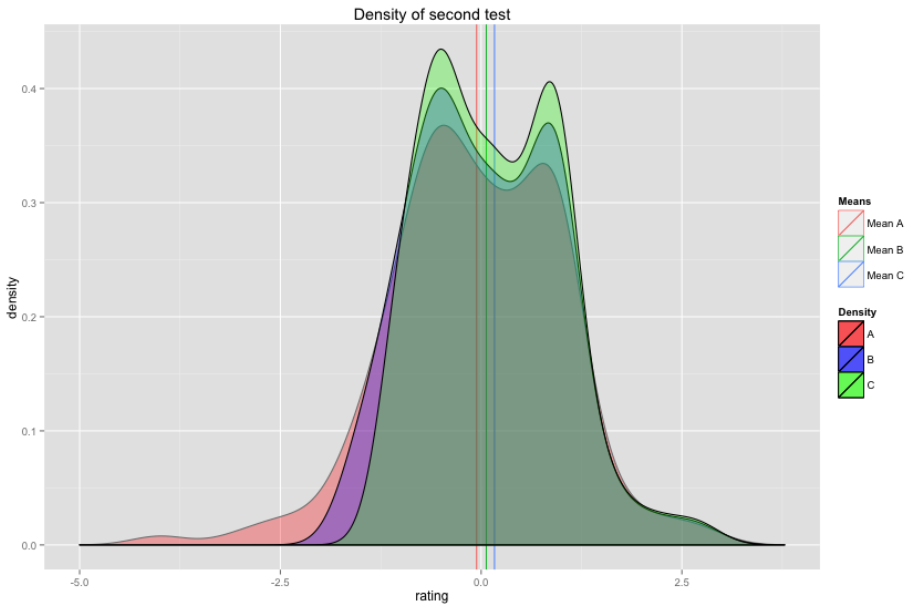


Figure 5.13: Densities of the second with a different minimum of rating (mean of A is -0.06, the mean of B is 0.06 and the mean of C $= 0.17$)

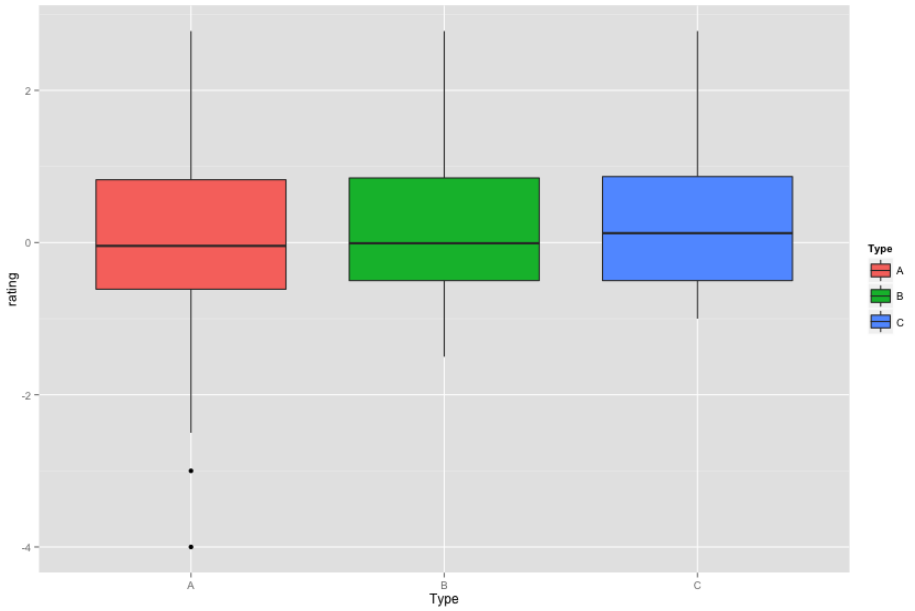


Figure 5.14: Boxplot with task-ratings of second test

As shown in Figure 5.13 after the average rating of all tasks is again negative. In addition, the boxplot in Figure 5.14 does not show a big difference between the three classes A, B and C. This is because only 13 of 150 tasks have a rating less than -1.5 (8%) and not many tasks have a great rating: there are only 23 tasks with a rating ≥ 115 (3%) and only 3 tasks with a rating ≥ 2 (2%).

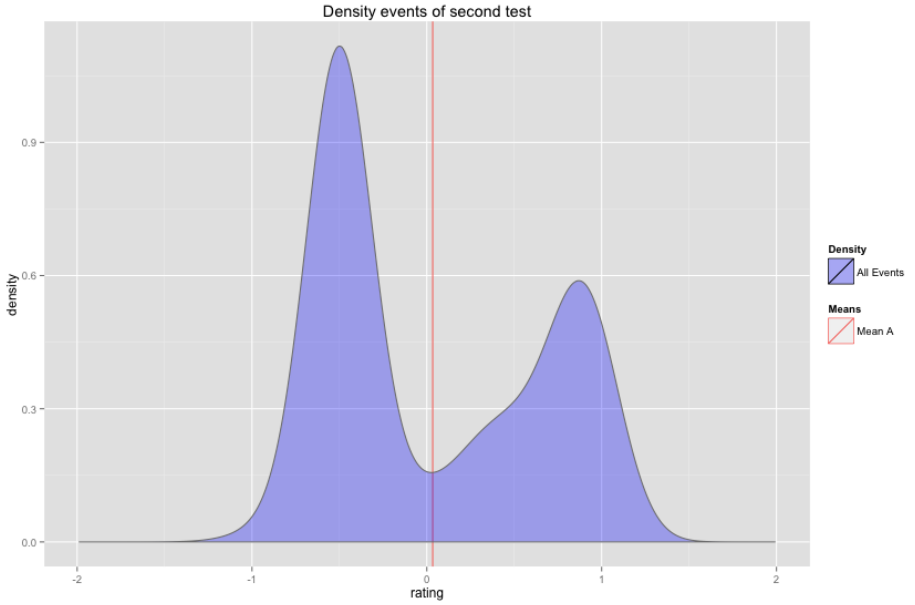


Figure 5.15: Density of all events in the second test (mean of A is 0.034)

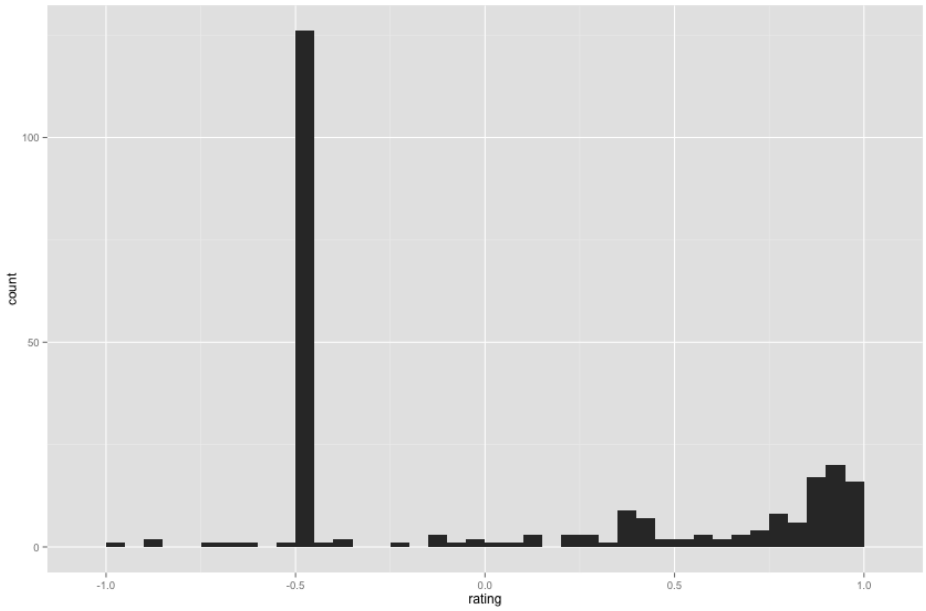


Figure 5.16: Histogram, distribution of event-ratings

49.21% cannot be assigned to any event calculated by DBSCAN. In addition, only 111 events have a rating ≥ 0 and there are 43% events with positive rating. Only 7% are classified as inexact events.

Difference			
position	**time**	**event**	**team**
25,34028327	0,17	0	0
34,30953076	0,05	0	0
19,63206054	0,13	1	1
12,67110102	0,91	0	0
45,67720521	0	0	0
56,70615378	0,27	1	1
26,34557062	0,92	1	0
26,65355513	0,12	0	0
15,68471868	0,08	1	1
18,0454648	0,1	0	0
17,06519557	0,18	0	0
46,75355922	0,08	1	1
15,40143175	0,21	0	0
6,860466456	0,08	0	0
17,72570419	0,6	1	0

Table 5.3: Difference between the calculated events and the ground truth with UPGMA

Table 5.3 shows the difference between the UPGMA calculated events and the manual events. Only 3 events have a big positional distance. The worst thing of this test is that 19 events were not calculated and only 15 were calculated.

Difference			
position	**time**	**event**	**team**
25,34028327	0,17	0	0
40,68001028	0,13	1	0
27,97846143	0,16	0	0
39,41173937	0,19	0	0
17,31048899	0,02	0	0
9,638692287	0,12	0	0
22,65005439	0,72	1	0
47,12557526	0,39	0	0
18,08048698	0,47	1	0
23,87727198	0,33	0	0
18,86387553	0,18	1	1
28,64668393	0,26	0	0
13,13141653	0,19	1	0
59,98198396	0,03	0	0
17,61656323	0,4	0	0
11,83990287	0,15	0	0
25,14386207	0,4	1	0

Table 5.4: Difference between the calculated events and the ground truth with DBSCAN

The comparison of the calculated events of this sequence to the manually added values shows an average position error of 26.31. However, four values are larger than 40 units, with an average distance of 49.26. The average time deviation in the second campaign is 0.254, which is neither bad nor good. A big difference compared to the first task is that that here six wrong events and one wrong team were calculated. Six wrong events are 35.29% wrong events. Another problem in this second campaign is that DBSCAN calculated 17 events, whereas 34 events were added manually, which means that 50% of all tasks were missed.
Sorting the data by the rating, where the biggest rating comes first, results in some other, better values:

Difference			
position	**time**	**event**	**team**
26,00076922	0,17	0	0
26,41113167	0,39	0	0
20,21594482	0,19	0	0
27,31117949	0,25	0	0
5,28924229	0,06	0	0
18,00972584	0,27	1	1
12,99650572	0,39	0	0
39,56039054	0,63	0	0
17,22161575	0,32	0	0
53,87906387	0,37	0	1
16,0350469	0,06	0	0
28,03882688	0,13	0	0
15,43160717	0,09	1	0
19,41012622	0,11	0	0
41,79745686	0,3	0	0
30,81527057	0,43	0	0
15,68963033	0	0	0
6,579004484	0,07	0	0
30,52929413	0,35	1	0

Table 5.5: Difference between the calculated events and the ground truth, with respect to the rating

By sorting those values, the number of events can be decreased, with a difference in the position ≥ 40, as visible in Figure 5.5. The number of not calculated tasks can also be reduced from 17 to 14 and the average position error can be decreased to 23.74 units. The number of wrong teams could be decreased from six to three and the number of wrong teams increased from one to two, by the DBSCAN calculations. Now, just one calculated point exists that does not belong to a manually entered data, whereas, in the by time ordered case it has been two.

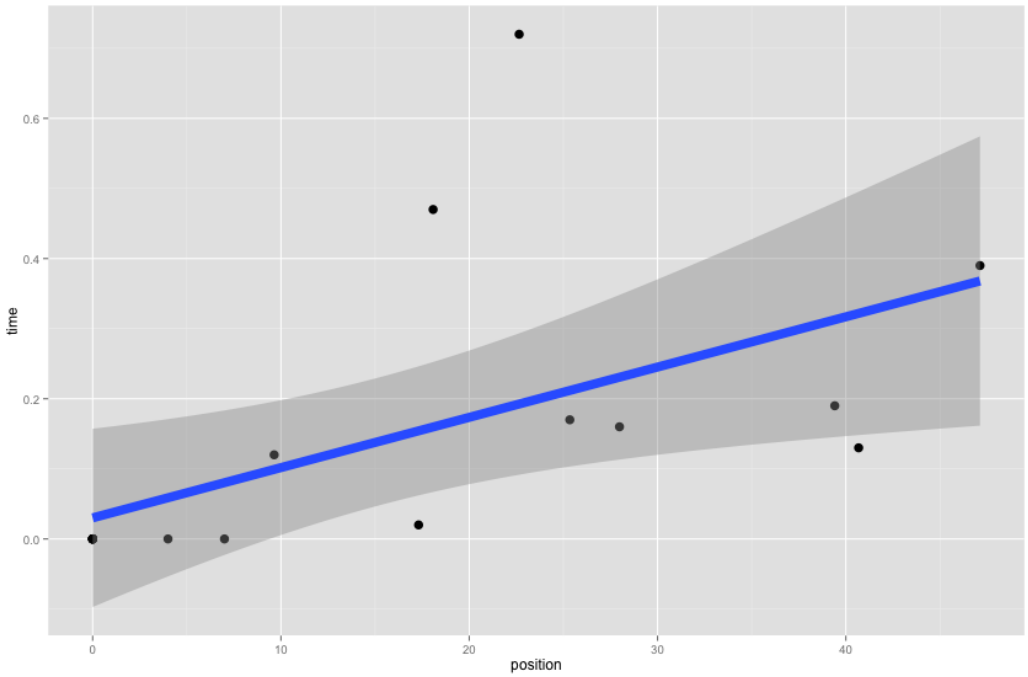


Figure 5.17: Correlation of position and time

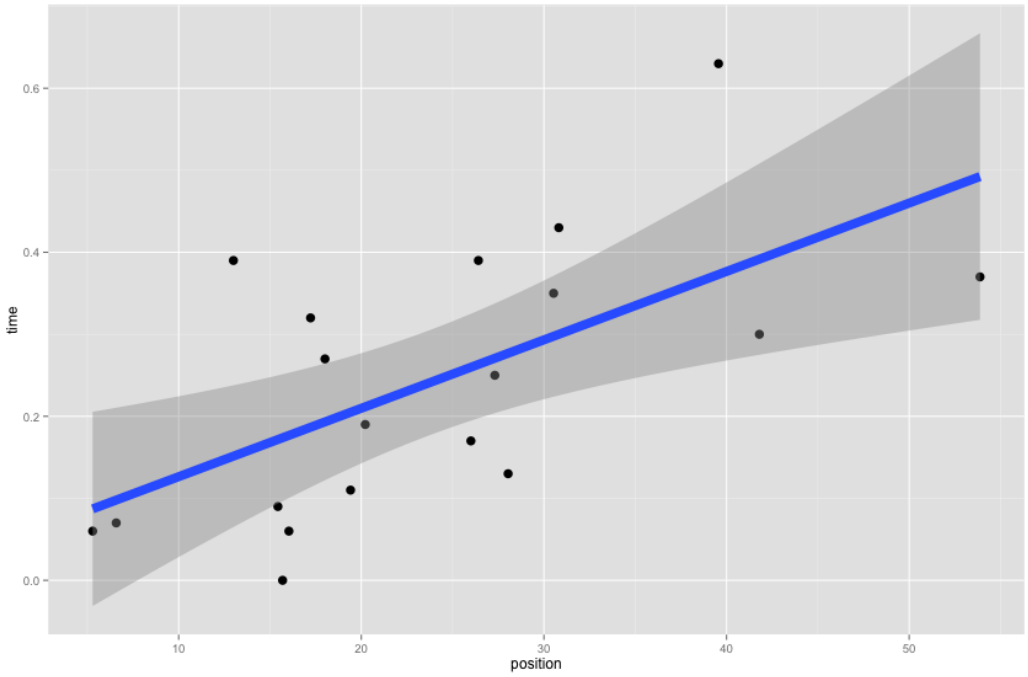


Figure 5.18: Correlation of position and time with ordered method

As shown in Figures 5.18 and 5.17, there is a correlation between the distance of the position and the distance in the time in both DBSCAN methods. In addition, near position values

in this test have close time values, and big position error values have bigger time errors.

Conclusion

In a sequence of close events, the DBSCAN algorithm cannot calculate all events. Changing the parameters was attempted, but having fewer not calculated events will increase the number of wrongly calculated events. In the end, it is better to calculate fewer wrong events, will be interpolated in the SportSense application [2].
Because sorting the values by rating got better results, it is first checked if there are enough points similar to the point with the biggest user rating. With this change, the users can be rated with good entered results. We also compared the calculated events with the UPGMA algorithm. However, calculating the points with the UPGMA will only produce 14 points, which is less than half of all points that should be in this sequence.

5.3.3 Third test

Two take another look at our system we created another campaign of 160 tasks and 80 seconds evaluated of the soccer game.

Results

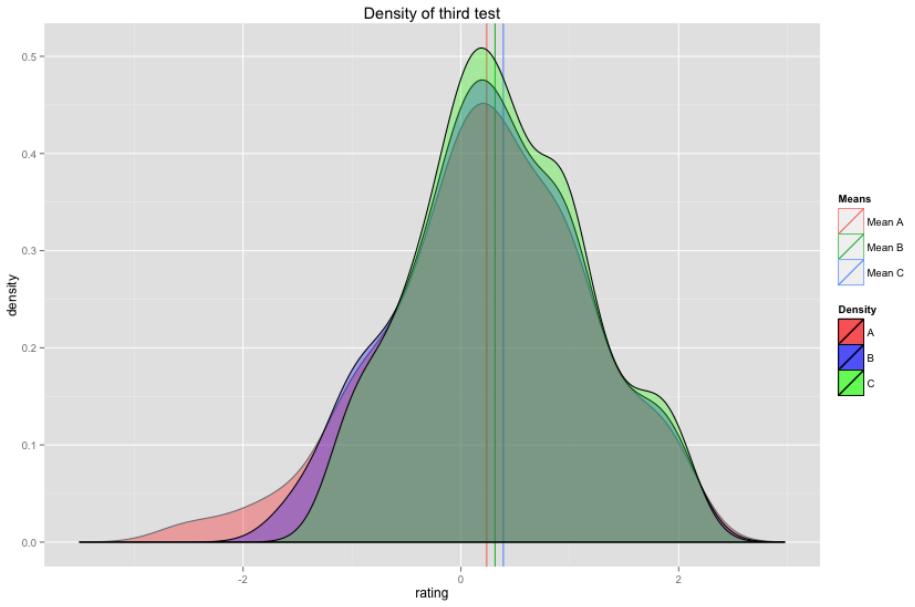


Figure 5.19: Densities of the third with a different minimum of rating (mean of A is 0.24, the mean of B is 0.31 and the mean of C = 0.39)

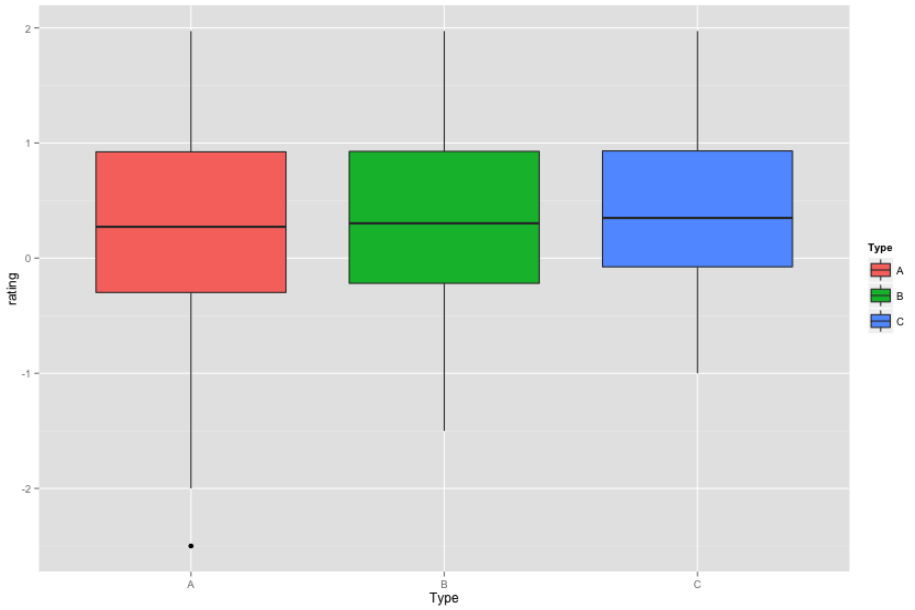


Figure 5.20: Boxplot with task-ratings of third test

In Figures 5.20 and 5.19 show that the upper quartile of all three sections is almost the same. The number of bad tasks is also not big. Of the 160 tasks in this campaign, 98 have a rating bigger than zero, which are 61, 3%, and only 7.5% of the tasks are rated less than zero.

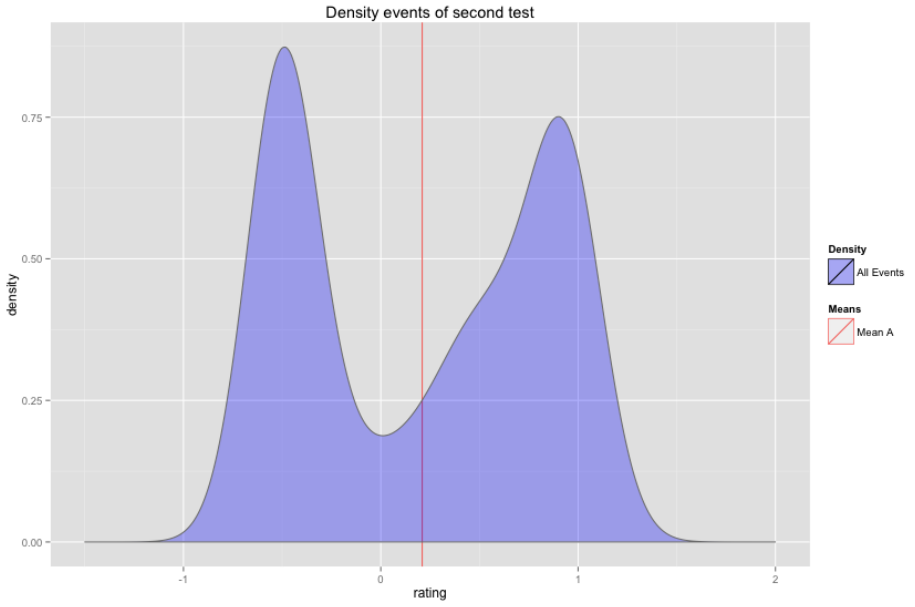


Figure 5.21: Density of all events in the third test (mean of A is 0.034)

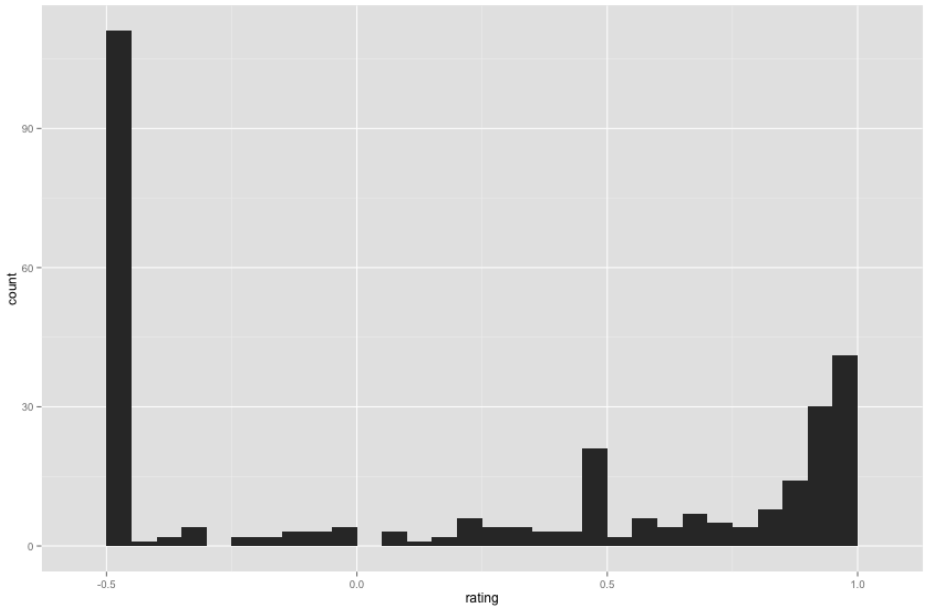


Figure 5.22: Histogram, distribution of event-ratings

In this campaign, the worst task rating is -0.5, which means that there are no inexact tasks in the users' events entered, either they are good or not classified. Of the 300 entered events in this campaign, which makes an average of 1.9 entered events per task, 110 events have been rated by -0.5. These are 35% of entered events, which are not assigned to a calculated event.

Difference			
position	**time**	**event**	**team**
31,53933446	0,024	1	1
11,95378601	0,143	0	0
29,64488234	0,022	0	0
10,41910793	0,033	0	0
94,67647828	0,005	0	0
10,36997589	0,124	0	0
21,30458085	0,099	0	0
47,48483439	0,944	0	0
33,65193326	0,052	0	0
29,08902652	0,205	0	0
47,58417537	0,409	1	0
9,758888922	0,094	0	0
48,20682103	0,899	0	0
44,5881021	0,022	0	0
12,85083055	0,231	0	0

Table 5.6: Difference between the calculated events and the ground truth

Figure 5.6 shows the differences between the calculated events by DBSCAN and the manually entered data. The average distance of all entered events in this test is 32.21 which is not really good. However, the temporal distance does not indicate a big difference between the two related events; the average is only 0.22 seconds. In addition, there are only four missing events, where two of them are in the first two seconds and refer to the calculation problem of DBSCAN that there must be a minimum number of points. There is no event calculated that does not exist in the sequence.

5.4 Demographics

Google Analytics[7] is a framework created by Google that generates the traffic and the traffic's source on a specified website. This tool will be used in the present approach to get an insight of users' home countries and to get an impression how long users stay on SportSense Crowdapp.

5.4.1 Geographic distribution

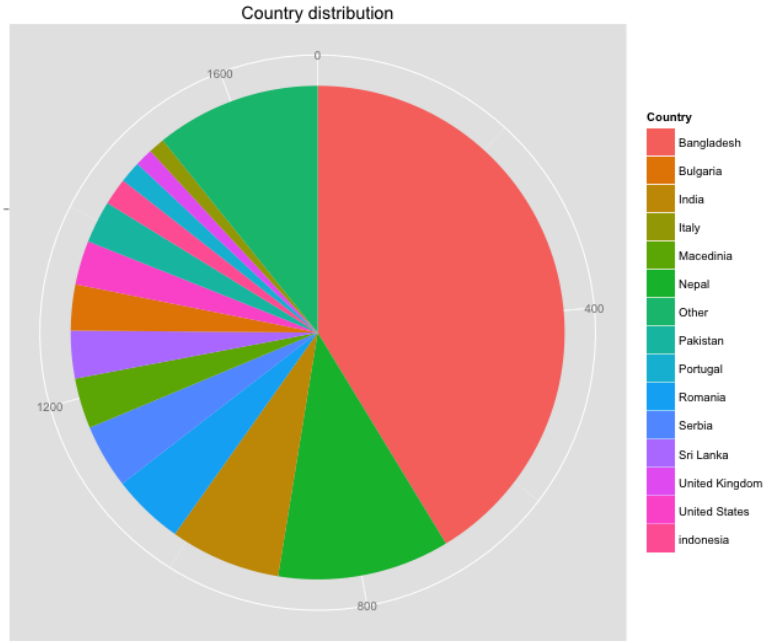


Figure 5.23: Home countries of participators on SportSense Cloudapp

[7] http://www.google.com/intl/de/analytics/

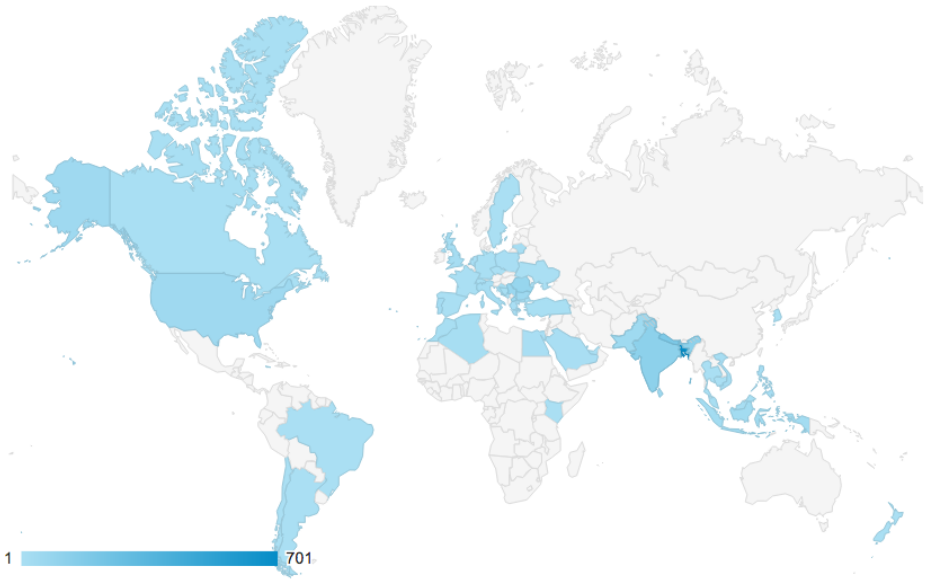


Figure 5.24: Map showing the distribution of our workers

As can be seen in Figure 5.23 and 5.24, a majority (1697) of participants in the campaigns are from Bangladesh. This is about 41%. Bangladesh is followed by Nepal and India. Both countries have over 100 page calls, but together they do not have even half as many page impressions as from Bangladesh.

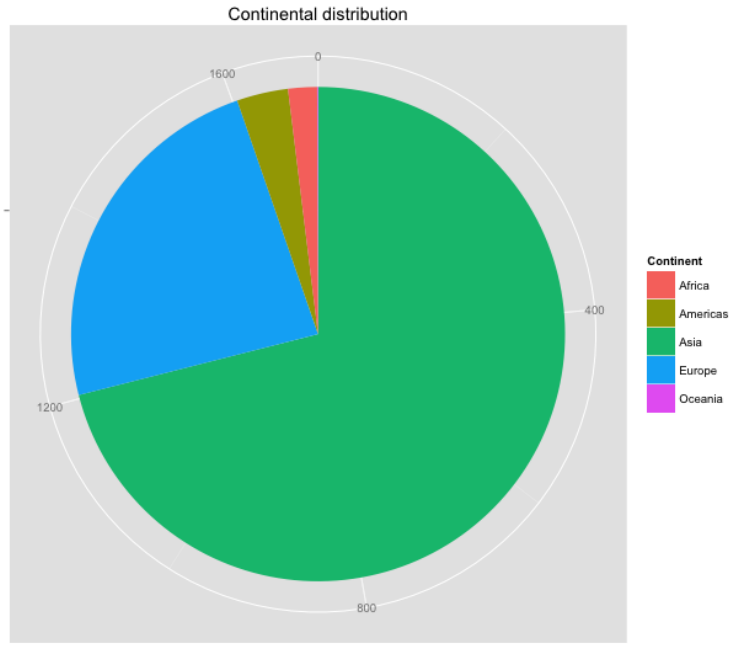


Figure 5.25: Continents of participants on SportSense Crowdapp

Figure 5.25 shows the continental distribution of the workers and indicates that 71% of the

workers are from Asia.

5.4.2 Time

The following examines the time a user needs to finish a task.

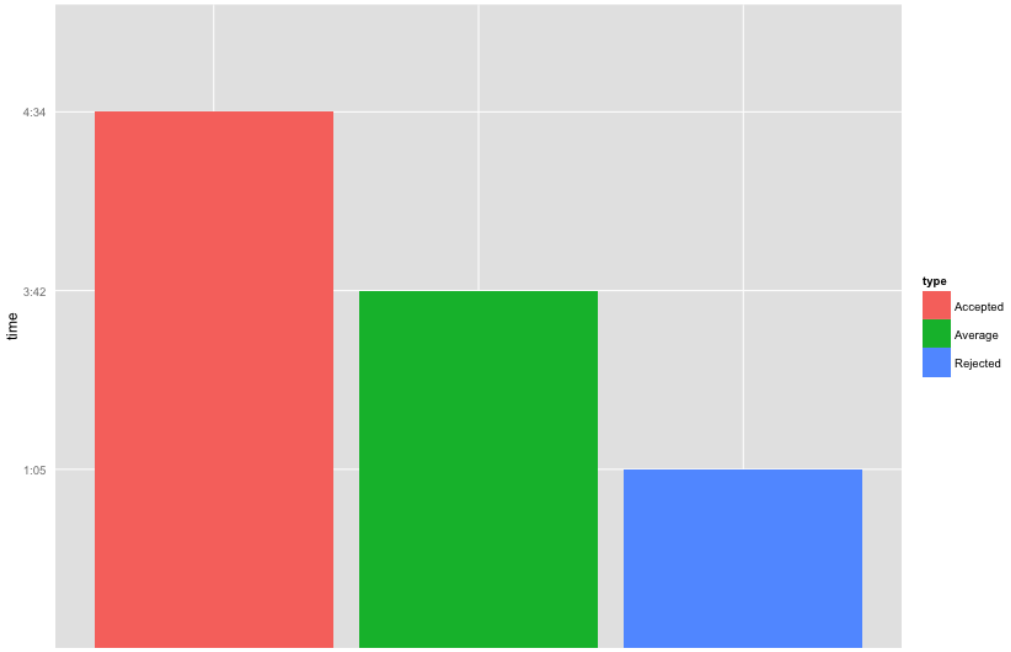


Figure 5.26: Time users need to finish a task

The above figure shows that the average time for accepted tasks is substantially higher than the average time for not accepted tasks. The average time for rejected tasks is 01:05, whereas the average time for accepted tasks is 4:34, which is about a quadruple of the time used by rejected tasks.

5.5 Conclusion

Especially, the pretest 5.2 we have seen that many Microworkers users are not doing their task properly. Many users just finished the task without entering any events, if there have been some. The pretest gave the possibility to eliminate those users from further tasks, because a user who did not enter any data will get a rating of -1.5, which is exactly the limit where users are not qualified to do more tasks.
By creating the tutorial video and playing it in the beginning, the tests also found that a smaller number of users had misunderstood the tasks.
There were also some difficult situations for the users, who were not clear about which event or how many events had to be entered, for example, the situation in a sequence of a player intercepting the ball and passing it at the same time to another player of his team. In this situation, three kinds of user inputs were made. The majority of users analyzing this sequence entered only an interception, the second most users entered only a pass, and there have also been some users entering a pass and an interception.
The calculation of the events in the two tests has shown that the DBSCAN algorithm works better than the UPGMA. However, the DBSCAN also produces mistakes and will not work if too many users enter inexact data, as it is only a statistical approach to calculate the clusters. A big problem DBSCAN has in difficult situations is that it will join two different points to one if they are close enough to each other. As described in the example above the position, the time, and the team passing and intercepting is the same, only the event changed. This little difference cannot be detected as two separate events with the DBSCAN algorithm, because if the parameters are changed to something that it would accept, in this case there would be an overfitting for other points, and this will produce a large number of wrong points in other situations. Thus, false negative events are accepted, and the false positive are kept as small as possible to ensure that there should rather be non-existing events in the data instead of missing events.

6 Discussion

This section, discusses the results of the thesis and explains the project's limitations due to the system: the workers and the provided video.

6.1 Microworkers

In the project, it was seen that Microworkers may not be the best crowdsourcing platform for the current approach.
This has different reasons. First, the system does not allow a user to do a task twice in one campaign, which is a disadvantage for the system, because the tasks in one campaign are not exactly the same, due to the changes in time of the video snippet.
Another issue is that Microworkers does not provide a system to determine by an interface whether to accept a task automatically after all tasks are completed. Each task must be accepted manually, which takes a lot of time, especially considering that one half of a video has about 2820 tasks. The introduced API version by the Microworkers system would fix this problem, but the system was introduced in August 2013 and still not available as a running system.

6.2 Video

It was also seen that the angle of the camera in a video is crucial. To provide good results, the system needs a video that does not contain any repetitions and slow motions clips, because this will lead to doubled data entries or more.
The system also needs a perspective that does not contain a large zoom, because without seeing an edge or more than one line it is hard for workers to set the correct position. Further, the camera view should always show the same perspective of the field, because, for example, a camera on the other side of the field would not be recognized by workers and would lead to entered values mirrored at the central line.

6.3 System

The results on this study have shown that the system is able to generate new events based on a dataset of Microworker users based on a clustering algorithm. With the introduction of a rating system one could disable users entering wrong or too little information from doing more tasks.

The calculated results with our clustering method have been good with respect to some classification problems with close points. Good results were generated in an average sequence of a football game regarding the number of events per second. The rating system could be improved by letting users do more pre-tasks at the beginning or by adding test-tasks after a certain number of tasks.

The system based on the users rating operates quite well. As shown in Figure 5.4, already with the pretest a large number of users could be disqualified from doing bad tasks. One issue is that this may be not enough. By adding a second pretest for every user the number of accepted users could definitely be decreased. With only one pretest we also accept many users adding just one or two of three events, as shown in Figure 5.5. In addition, many users also did their task imprecisely. Those users often get a rating between 0 and -1 and are able to participate more tasks, if they will do a second pretest and get another acceptation. However, with bad results they will be dismissed from further tasks.

Another possibility to further decrease the number of inaccurate workers could also be to scatter test-tasks in the standard tasks rather than only having a pretest. This method is also described in [9]. With this, every user will have to do a test task after a specified number of tasks, which will decide if he still is doing good work or if his quality slackened.

In the test we made we have also seen that the quality of the positions value depends on the position on the field, as also described in [13]. Events near a junction of two lines is simple; more difficult are positions where only two lines are seen; and with positions where only one straight line is seen, it is quite hard to enter the correct position. By entering the data for comparison, it was also seen that there are many actions in a game that cannot clearly be labeled as an event. Another issue is that to get results without duplicates, the video file must not have any slow motions clips or repetitions of scenes. Since every user sees only a five second snippet he may not recognize if this is a repetition or not and repetitions are mostly from views where it is hard to set the right position, as it is sometimes even unknown on which half the action was. Especially if the video contains a camera position behind the goal, users do not know if it was the goal on the left or right side of the field.

6.4 Financial Approach

As the motivation of this work was to find another, cheaper way to get annotations in sport videos, the financial view must also be considered. With the present approach, a task will cost 0.20 Dollars. Assuming that each half lasts about 47 minutes, 45 minutes official time and 2 minutes of stoppage time, we would end up in a total game time of 94 minutes. Due to the video, one needs to create a campaign for each half. This will give us 2 times 47 minutes, where every task is 5 seconds. Because a task always starts 0.5 seconds later than the previous task, this will end up in 5640 tasks. Where each task generates costs of 20

Cents a half would cost 1128 dollars and the total game 2256$.
But that is not the total amount. Because every user must do one pretest at the moment, which also costs 20 cents for all accepted users. The expense for the pretest would reduce with the number of campaign started, because after a certain number of users the number of new users would decrease.
Unfortunately we couldn't find any informations on how much manually annotated data by experts costs for one soccer game.

7

Conclusion and Future Work

7.1 Conclusion

With this thesis, a system was introduced that allows annotating a sport video by using a micro-job platform. The online platform Microworkers was used to collect events by crowdsourcing users and calculating the final events by the clustering algorithm DBSCAN. The evaluation of the tests with Microworkers has shown that it is possible to build a system where only good users can participate in campaigns and use their data. It was shown that calculated data can only be as good as the statistical algorithm to summarize workers entered actions to new events. This approach showed that it was not possible to calculate perfect and complete data, but however, some acceptable results could be generated if the sequence is not too complex.

7.2 Future Work

7.2.1 Administrator system

To make this approach usable for external clients, one would have to extend the administrator system. As a first change, all administrators should have the possibility to upload movies and add the parameters of the soccer game. For this, changes to the database would be necessary. The administrator should also be able to get the calculated events from the database on the administrator system.

7.3 Further Sports

By introducing some small modifications, it would be possible to allow for other sport videos based on a field, in which multiple teams would interact in events.
With some minor modifications in the code and the database, it would be possible to annotate other type of sport in the system, which may be based on a quadratic field, played with a ball, and where multiple teams interact.
With these two more or less small modifications, one can easily extend the system to other ball other ball sports, like rugby, American football, tennis, volleyball, handball, basketball,

field-hockey, cricket, baseball or lacrosse and others.

Bibliography

[1] Microworkers. URL http://www.microworkers.com.

[2] Al Kabary, I. and Schuldt, H. SportSense: Using Motion Queries to Find Scenes in Sports Videos. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, pages 2489–2492. ACM, New York, NY, USA (2013). URL http://doi.acm.org/10.1145/2505515.2508211.

[3] Ballan, L., Bertini, M., Bimbo, A. D., and Nunziati, W. Soccer Players Identification Based on Visual Local Features. In *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, CIVR '07, pages 258–265. ACM, New York, NY, USA (2007). URL http://doi.acm.org/10.1145/1282280.1282321.

[4] Seo, Y., Choi, S., Kim, H., and Hong, K.-S. Where Are the Ball and Players? Soccer Game Analysis with Color Based Tracking and Image Mosaick. In *Proceedings of the 9th International Conference on Image Analysis and Processing-Volume II*, ICIAP '97, pages 196–203. Springer-Verlag, London, UK, UK (1997). URL http://dl.acm.org/citation.cfm?id=646276.686879.

[5] Iwase, S. and Saito, H. Parallel tracking of all soccer players by integrating detected positions in multiple view images. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 4, pages 751–754 Vol.4 (2004).

[6] Surowiecki, J. *The Widsom of Crowds*. Doubleday; Anchor (2005).

[7] Jeff Howie. The Rise of Crowdsourcing. URL http://archive.wired.com/wired/archive/14.06/crowds.html.

[8] Kittur, A., Smus, B., Khamkar, S., and Kraut, R. E. CrowdForge: Crowdsourcing Complex Work. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, pages 43–52. New York, NY, USA (2011).

[9] Liu, Q., Ihler, A., and Steyvers, M. Scoring Workers in Crowdsourcing: How Many Control Questions are Enough? (2013).

[10] Kittur, A. Crowdsourcing, Collaboration and Creativity. *XRDS* (2010).

[11] Hirth, M., Hoßfeld, T., and Tran-Gia, P. Anatomy of a Crowdsourcing Platform - Using the Example of Microworkers.Com (2011).

[12] von Ahn, L., Maurer, B., Mcmillen, C., Abraham, D., and Blum, M. re-CAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science*, (5895):1465–1468 (2008).

[13] Perin, C., Vuillemot, R., and Fekete, J.-D. Real-Time Crowdsourcing of Detailed Soccer Data (2013). URL http://hal.inria.fr/hal-00868775.

[14] Ester, Martin and Kriegel, Hans P. and Sander, Jorg and Xu, Xiaowei. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise (1996).

[15] Vondrick, C., Ramanan, D., and Patterson, D. Efficiently Scaling Up Video Annotation with Crowdsourced Marketplaces. In *Computer Vision – ECCV 2010*, Lecture Notes in Computer Science, pages 610–623. Springer Berlin Heidelberg (2010).

[16] Sokal, R. R. and Michener, C. D. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438 (1958).

[17] Eaton, J. *GNU Octave Manual*. Network Theory Ltd. Bristol, UK (2008). URL http://www.network-theory.co.uk/docs/octave3/index.html.

[18] MySQL Event Scheduler. URL http://dev.mysql.com/doc/refman/5.1/en/events.html.

[19] FFmpeg. URL http://ffmpeg.org/.

[20] Keller, M. S. Take Command: Cron: Job Scheduler. *Linux J.* (1999).

[21] php Data Object. URL http://www.php.net/manual/de/book.pdo.php.

A

Tutorial Document

How-To SportSense

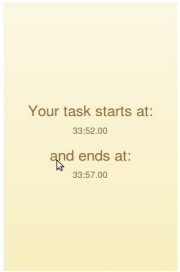
If you accept to do the task, you will see a Website looking like the Picture on the right side.

On this Website we have three sections.
The red one contains the videoplayer, the green one the input elements and the yellow one your entered data.

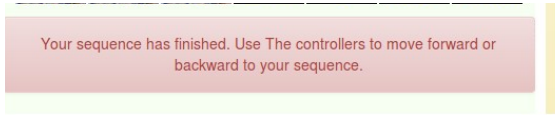
The Video Player

The Video Player got an extension with which you can play through the video.
The blue break icon indicates your actual position in the video.
If the background of the break icon is black, your outside your task.

To get to another position of the video click on the images or use the left and right arrows on your keyboard. If you want to play and stop the video use the spacebar of your keyboard.



If you are outside your sequence a warning will show up under the player and the events will be replaced with the image on the left side.



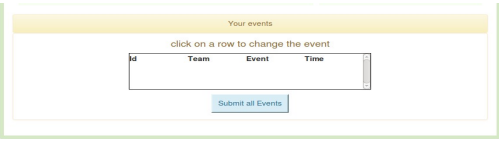
The Input Elements

On the right side of the page you've got two things to chose before adding an event. Select first of all the event which is in the action and as second the team that belongs to the action.
Please think about that a foul, goal, offside, out, etc. is only one if the referee blows his whistle.

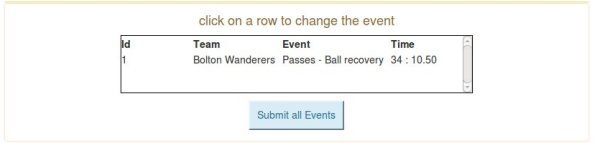
To get the right time, please set the video at the point where the event happens.

List of Events

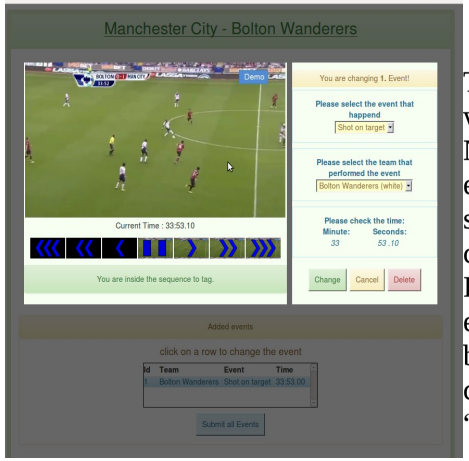
In the last section you will see your captured events. At the beginning there will be an empty list like the image on the right.



If you have added an Event. You will see it in the list as shown below.



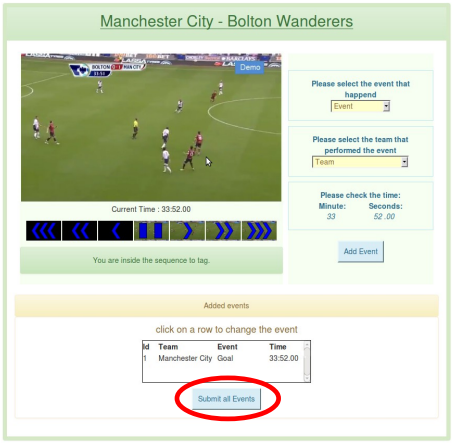
If you think you've added a wrong event you can click on it in the list to change or delete it. The page will change to this:



The selected event in the list will be highlighted with blue. Now you can change the event, team or/and time and submit the changes by clicking the “Change” button. If you want to delete this event click the “Delete” button and if you want to cancel changing click the “cancel” button.

Finishing

If you finished the whole sequence click on the “Submit all events” button which is marked red on the image on the right side.
If there aren't any events in the whole sequence click also the button to receive you're token.



Declaration on Scientific Integrity
Erklärung zur wissenschaftlichen Redlichkeit

includes Declaration on Plagiarism and Fraud
beinhaltet Erklärung zu Plagiat und Betrug

Author — Autor
Fabio Sulser

Matriculation number — Matrikelnummer
2010-750-248

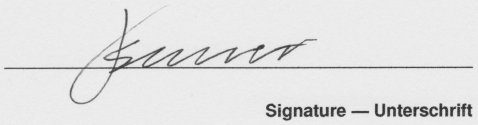
Title of work — Titel der Arbeit
Crowdsourcing Annotations for Video Retrieval in Sports Videos

Type of work — Typ der Arbeit
Bachelor Thesis

Declaration — Erklärung
I hereby declare that this submission is my own work and that I have fully acknowledged the assistance received in completing this work and that it contains no material that has not been formally acknowledged. I have mentioned all source materials used and have cited these in accordance with recognised scientific rules. Hiermit erkläre ich, dass mir bei der

Abfassung dieser Arbeit nur die darin angegebene Hilfe zuteil wurde und dass ich sie nur mit den in der Arbeit angegebenen Hilfsmitteln verfasst habe. Ich habe sämtliche verwendeten Quellen erwähnt und gemäss anerkannten wissenschaftlichen Regeln zitiert.

Basel, 05.06.2014



Signature — Unterschrift

UNIVERSITÄT BASEL

PHILOSOPHISCH-NATURWISSENSCHAFTLICHE FAKULTÄT

Declaration on Scientific Integrity
(including a Declaration on Plagiarism and Fraud)

Bachelor's / Master's Thesis *(Please cross out what does not apply)*

Title of Thesis *(Please print in capital letters)*:
CROWDSOURCING ANNOTATIONS FOR VIDEO RETRIEVAL IN SPORTS VIDEOS

First Name, Surname *(Please print in capital letters)*: FABIO, SULSER

Matriculation No.: 10-750-248

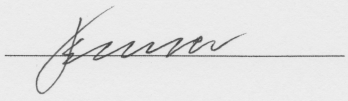
I hereby declare that this submission is my own work and that I have fully acknowledged the assistance received in completing this work and that it contains no material that has not been formally acknowledged.

I have mentioned all source materials used and have cited these in accordance with recognised scientific rules.

In addition to this declaration, I am submitting a separate agreement regarding the publication of or public access to this work.

☐ Yes ■ No

Place, Date: Basel, 05.06.2014

Signature: 

Please enclose a completed and signed copy of this declaration in your Bachelor's or Master's thesis .

UNI
BASEL