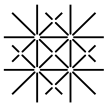


University of Basel

# Evaluating Result Quality of Multimedia Retrieval Systems

Bachelor Thesis

Natural Science Faculty of the University of Basel
Department of Mathematics and Computer Science
Databases and Information Systems Group
https://dbis.dmi.unibas.ch

Examiner: Prof. Dr. Heiko Schuldt
Supervisor: Luca Rossetto, MSc

Sein Coray
s.coray@unibas.ch

19.07.2017

# Acknowledgments

Working completely alone on a thesis without any helping hands would be impossible. Many people helped making this thesis possible. I will not be able to mention everyone who helped me here, still I'm grateful for their help. I want to thank Prof. Dr. Heiko Schuldt for giving me the opportunity to do this thesis. I especially thank my Supervisor Luca Rossetto for always giving advice and all his support throughout this thesis. I would also like to thank my friends at the University and my family for motivating me and all their help, especially for participating in the implemented tool.

# Abstract

To evaluate multimedia retrieval systems, a crowd of people can be asked to evaluate the similarity of query results to get a majority opinion. As this requires breaking complex queries down to simple similarity questions, we need a lot of people to participate in rating similarity. In this thesis we describe how we used two approaches to motivate people to answer questions. The first idea was to connect our implemented tool to Amazon Mechanical Turk to use crowdsourced tasks where answers are given in return for money. The second approach was to use gamification elements so that people answer questions by playing a game. We analyzed how people answered the questions to see if and how well we can evaluate multimedia query results. We inspected if our two approaches worked to gather data and if there are differences in the resulting majority consensus. After some time the approaches both showed promising results presented in this thesis.

# Table of Contents

**Acknowledgments** **ii**

**Abstract** **iii**

**1 Introduction** **1**
1.1 Multimedia Retrieval Systems . . . 1
1.1.1 vitrivr . . . 1
1.2 ArtSimily . . . 2
1.2.1 Microworkers . . . 2
1.2.2 Gamification . . . 2

**2 Related Work** **3**
2.1 Mechanical Turk . . . 3
2.1.1 Validation of user input . . . 4
2.2 Gamification . . . 4

**3 Concepts** **6**
3.1 User Types . . . 6
3.2 Multimedia Query . . . 6
3.3 Questions . . . 7
3.4 Answer Session & Session Validity . . . 7
3.4.1 Special Validators . . . 8
3.5 Result Tuple . . . 9
3.6 Game . . . 9

**4 Implementation** **11**
4.1 Data Storage . . . 11
4.2 Players . . . 11
4.3 Achievements . . . 12
4.4 Microworkers . . . 12
4.5 Admin Interface . . . 13
4.6 Background Processes . . . 14
4.7 Challenges . . . 14
4.7.1 Ground truth . . . 14

4.7.2 Validity . . . . . 15

**5 Process** **17**
5.1 Players . . . . . 17
5.2 Microworkers . . . . . 17
5.3 Feedback & Reactions . . . . . 18
5.3.1 Gamification . . . . . 18
5.3.2 Microworkers . . . . . 19

**6 Evaluation** **20**
6.1 Overall Results . . . . . 20
6.2 Similarity of tuples . . . . . 21
6.2.1 Wide Results . . . . . 21
6.2.2 Clear Results . . . . . 21
6.3 Answer Sessions . . . . . 24
6.3.1 Similarity Answers . . . . . 24
6.3.2 Session Validities . . . . . 24
6.3.3 Session Duration . . . . . 25
6.4 Gamification . . . . . 26
6.4.1 Returning Players . . . . . 27
6.5 Microworkers . . . . . 28
6.5.1 Unique Workers . . . . . 28
6.5.2 Rejected Workers . . . . . 28

**7 Conclusion** **29**
7.1 Conclusions . . . . . 29
7.2 Future Work . . . . . 29
7.2.1 Multimedia . . . . . 29
7.2.2 Gamification . . . . . 30
7.2.3 Microworkers . . . . . 30
7.2.4 Admin Interface . . . . . 30

**Bibliography** **31**

**Appendix A Appendix** **33**
A.1 Mechanical Turk . . . . . 33

**Appendix B Appendix** **36**
B.1 Database Model . . . . . 36

# 1 Introduction

Every year the number of created multimedia data in the world increases. Over social media and other channels huge amounts of images, videos and audio files are exchanged and consumed. To be able to find desired content in this data we need search engines. The classical way this was done in the last years was mostly by having all the files tagged with keywords to be able to find related content. But tagging multimedia is not a trivial process. Another way of searching is to compare multimedia objects directly based on their content. Such search engines are called content-based multimedia retrieval systems.

## 1.1 Multimedia Retrieval Systems

Given the retrieval problem we have a large number of documents and we have a query defining what we want to search from these documents. As a response we want to obtain a list of results which matched to the query. A multimedia retrieval system uses multiple methods called features to describe all the documents. The same features are then applied to the query so it can be compared to the indexed documents. The provided results are then ordered by their relevance to the query input.

### 1.1.1 vitrivr

The multimedia retrieval system which is used in this thesis is vitrivr [9]. The vitrivr stack is built of ADAMpro [4] as database storage, Cineast [8] which is the multimedia retrieval engine and a web frontend. When evaluating the query results, Cineast is the involved part, which has many features for indexing and comparing multimedia objects. To get a good evaluation of a multimedia retrieval system it is required to compare and annotate a huge amount of multimedia objects. This is too much to be completed by a single person. Also to get an as diverse opinion as possible about similarity, the view of many people should be included.

## 1.2 ArtSimily

In this thesis we developed a tool to evaluate query results from such multimedia retrieval systems like vitrivr. We called it ArtSimily. We used two approaches, both of them have in common that the complex evaluation task is broken down into simple questions which can be easily distributed. The first approach consists of using Microworkers where people get paid for answering questions and the second approach is to use gamification elements to motivate people. The goal of this thesis was to use at least microworkers to gather the required data, since we knew that we would be able to gather data in this way.

### 1.2.1 Microworkers

Crowdsourcing of tasks has become more and more popular over the last few years and is widely used in research. In this process a large problem is divided into small chunks of work to be distributed over a crowd of people. Generally, crowdsourcing is not bound to any restriction of location, gender, age, etc. and therefore opens the possibility to use a huge amount of human power. There are multiple large providers for completing and publishing such tasks. In this thesis, the focus is on Amazon Mechanical Turk which is one of the big players in this business.

### 1.2.2 Gamification

Gamifying a task means that game elements (for example achievements or leaderboards) get added to a task. This wraps a potentially boring task into a game which can be played. With such gamification elements we tried to motivate people to play ArtSimily. We asked for feedback from players during the implementation to evaluate how good the gamification elements are performing on the players' motivation.

# 2
# Related Work

## 2.1 Mechanical Turk

Amazon launched Mechanical Turk in 2005 as a service where people can crowd source tasks or solve such tasks. These tasks are called HITs (Human Intelligence Tasks) which are created by a *Requester* and completed by a *Worker*. A requester can set how many workers should complete a HIT and if workers are allowed to complete it multiple times. For each completed HIT the requester pays a small amount of money to the worker.
Mechanical Turk is used in various fields of research. For example, simple tasks which are not solvable by computer algorithms or surveys on a specific topic can be distributed to humans all over the world. Even if the field of use is really large, it is important to make sure that Mechanical Turk is the right instrument to use. Formulating the tasks clearly and understandably is a major factor to get good results. Kittur et al. [5] provided several recommendations regarding creating HITs on Mechanical Turk:

- It is important to have security questions as part of the task as a verification to check if workers are taking the task seriously. They should be informed that their answers will be checked which could also reduce the quota of workers not answering in a meaningful way.
- When less or similar effort is needed to answer well instead of just giving random or useless answers, it can reduce the amount of workers which do not participate correctly.
- To catch most of the suspect responses, they recommend to have more than one way to detect these as there are various possible patterns which could occur in such cases like short task duration or repeated textual answers.

Some research fields heavily started using crowdsourcing to complete tasks to help gather data about a problem which is only solvable by humans. Especially as Buhrmester et al. [1] stated that such data collection can compete with traditional ways regarding the result quality.

### 2.1.1 Validation of user input

As a requester you only want to consider results which were from workers answering seriously. Which means that you do not want workers which are just after the money and clicking randomly on the HITs to get them finished quickly. For this reason Mechanical Turk requires every completed HIT to be approved or rejected by the requester. This can either be done by manually inspecting the responses for every HIT, by providing specific checks (CAPTCHA, etc.) or by analyzing patterns to detect false answers. Zhu and Carterette [11] analyzed behaviors which may occur from such malicious workers. As an outcome they listed some patterns regarding the time line and answers of the task:

**Periodic time patterns** workers are switching between taking shorter and longer amounts of time to answer questions

**Interruption** there is a normal amount of time needed per answer except one single very large time spike

**Periodic answers** the answers oscillate between two ratings consistently

**Fixed answer** all questions were given the exact same answer

Furthermore they also state that it is important to have trap questions (like the above mentioned security questions) to detect bad workers. When they looked at the occurring patterns of the workers which fell trough the trap, 6 out of 8 also showed some of the unusual patterns they described.
Another way to just overcome a few workers which do not answer correctly is by letting multiple workers finish the same task and then sorting out the ones which differ too much from the majority, as done by Urbano et al. [10]. This approach can be useful if the number of tasks is low and it's possible to invest a multiple of the basic costs for the tasks to have every one of them completed multiple times.
For this thesis, we took some of the above described patterns or recommendations to create multiple checks if users are answering well. This way we could automatically accept and reject tasks based on these tests.

## 2.2 Gamification

During the last years the term "Gamification" became widely present in various areas which are not directly related to classical gaming. This method consists of adding specific elements to an application to motivate users to play it as a game although its original actions are not considered to be playful.
To get people motivated and keep them playing such an application needs to have multiple aspects considered. According to Deterding et al. [3] it is possible to differ between multiple levels of abstraction, whereas elements like badges, leaderboards, levels belong to the *Game interface design patterns*, the higher abstracted *Game model* is more dependent on the idea of the game. Considering a gamified research project, the motivation from the game itself often is more difficult to achieve because these tasks are often repetitive and can be boring without any other gamification element. It is simple to add the *Game interface*

*Design patterns* elements to a repetitive task to encourage people to play, but often this is not enough. Codish and Ravid [2] claim that the motivation of players can be improved by having an adaptive gamification which takes the demographics and personality of the user in account to create a personalized experience. This is because some people may find a specific gamification element not playful where others do. So depending on the personality, gender, age and others the game should adapt to maximize the playfulness.

Lux et al. [6] used gamification elements to examine how and how quickly image ranking reaches a consensus. They let people sort images by their relevance to a given criterion and took a look at the resulting overall score for the ranking when considering the answers of all users. As this simple task gets boring they introduced a score which was given after a round of playing. Among other points it considered how fast a user completed the task and how much the user's opinion diverged from the majority.
They were also confronted with the classical cold start problem when using answers from other users to rate new input which leads to the requirement of having certain users give a ground truth at the beginning. This method was also used in ArtSimily by having an admin pruning some image pairs which later were used as security questions.
As an outcome Lux et al. claimed that the majority consensus was reached quite quickly (with a very small number of games). They understand this as a strength of this kind of information retrieval to help build a ranking model. But at this point it has to be considered that their experiment was done with ten participants which most likely always gave correct answers (at least in their opinion). Their model does not consider that people may just randomly give input and not take the task seriously which would increase the distance of answers and could make it difficult to see the correct consensus which is to be determined. The gamification approach was used only in a limited way as the score for a game was the only motivation which was given to keep the users playing. Also it is difficult to analyze the effectiveness of this method with the limited number of participants.
As in ArtSimily we had users where we could not be sure that their answers are meaningful, we expected to need more answers than Lux et al. This is the case, because the majority's opinion might diverge and not be clearly visible on certain pairs with just a small number of answers.

# 3 Concepts

In this chapter we describe the concepts used in ArtSimily. These concepts include how queries are handled, split and distributed. There are two main parts, the front-end accessible by everyone and the back-end which is used by the administrator. Users visiting ArtSimily can simply start playing, they are not required to be logged in. After a certain number of answered questions they get a game score and the game is finished. Microworkers use nearly the same process except that they do not get a score at the end.

## 3.1 User Types

ArtSimily can be used by three different types of users:

**Admin** These users can log in to the admin interface and manage all the queries which are on the system. They are able to see the evaluation progress, import/export queries, add microworkers, etc.

**Player** Users which just use the gamification part of the tool (meaning, they just play with it by giving answers). They are either anonymous or authenticated via a third party provider using OAuth (see Section 4.2).

**Microworker** These are users which are answering questions because they are running sessions from a microworker provider.

## 3.2 Multimedia Query

The whole process starts with having multimedia query results which should get evaluated by ArtSimily. Every query consists of a search input element and a list of results ordered by their rank (given by the multimedia retrieval system). A query $Q$ is defined as follows:

$$Q = \langle q, R \rangle \tag{3.1}$$

Where $q$ is the query input and $R$ is the list of the resulting media objects. So for every search result $r \in R$ the input element gets paired with the result element in the form of a result tuple $RT$ (see Section 3.5):

$$R = \langle r_1, r_2, ..., r_n \rangle \tag{3.2}$$

$$RT = \langle \langle q, r_1 \rangle, \langle q, r_2 \rangle, ..., \langle q, r_n \rangle \rangle \tag{3.3}$$

These input and search elements can be images, sketches, clips, etc.
All multimedia elements (search inputs and results) are saved uniquely as we do not want to evaluate the same combination of the same $q$ and $r$ twice. Throughout this thesis, the term *Media Objects* is used to refer to these elements.

## 3.3 Questions

For the evaluation, users will get questions where they have to evaluate the similarity of two or three given media objects. When having two media objects, the user has to say how similar these two are. If the question consists of three media objects, one of them is the base object and the user needs to answer which of the two other provided objects is more similar to it. Figure 3.1 shows an example of a question on ArtSimily.

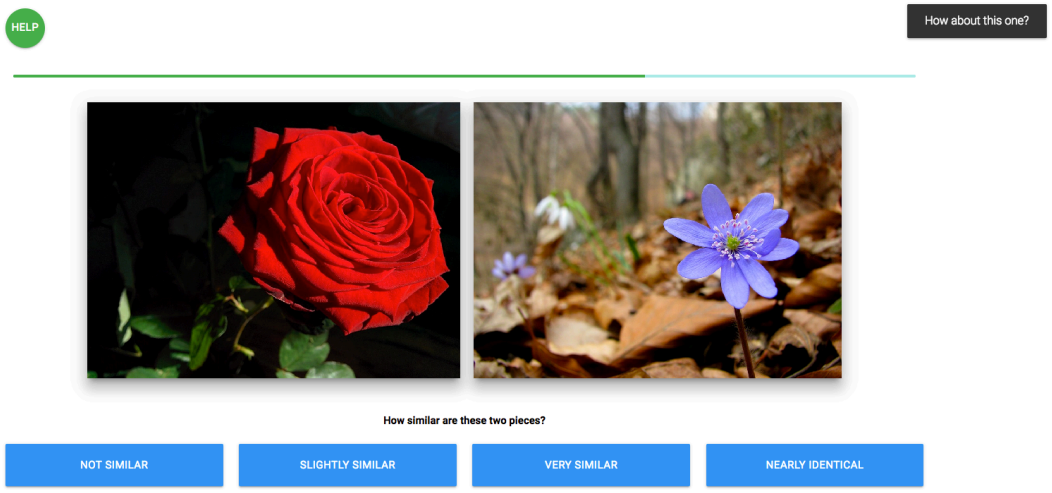



Figure 3.1: Screenshot of question from ArtSimily

## 3.4 Answer Session & Session Validity

As soon as a user (no matter which type) starts viewing questions an *Answer Session* is created. At this point a fixed number of questions (depending on the user type) is generated which belong to this newly created answer session. After all the questions are answered the session is closed and depending on the user type different actions are executed:

- If the session was created by a microworker, a survey code is shown to the user which they can use to confirm they have completed the task (see Section 4.4).

- If the session was created by a player, ArtSimily calculates a score based on their answers and displays it. (see Section 3.6)

To determine the quality of the answers given by the user, a *Session Validity* is calculated. This value is updated after every answered question and some additional checks are performed. To make this possible multiple security questions are included in the session where ArtSimily knows the correct answer quite certainly and therefore can test if the user answers correctly. The session validity influences the similarity calculation for the corresponding result tuples. The session validity $S_k$ for a session $k$ is calculated on all answers $a_0, a_1, ..., a_N$ which have enough values to calculate a Gaussian curve with.

$$a_n \in \{0, 1, 2, 3\} \quad \forall n \; 0 \leq n \leq N \tag{3.4}$$

$$S_k = \max\left(0, \min\left(1, \prod_{i=0}^{n} \left(0.6 + \min\left(p_i(a_i)^3 \cdot mult\left(a_i\right), 1\right)\right)\right)\right) \tag{3.5}$$

Where $p_i$ is the probability density function of the Gaussian curve of the result tuple answered with $a_i$ and

$$mult(a_i) = \left\{ \begin{array}{ll} 2, & \text{for } \sigma_i > 1 \text{ and } |a_i - \mu_i| < 1 \\ 4, & \text{for } 0.5 < \sigma_i \leq 1 \text{ and } |a_i - \mu_i| < 1 \\ 8, & \text{for } 0.25 < \sigma_i \leq 0.5 \text{ and } |a_i - \mu_i| < 0.5 \\ 1, & \text{else} \end{array} \right\} \tag{3.6}$$

$S_k$ is designed to return a significant lower value when security questions are answered incorrectly but also does not punish too severely when the answer is not exactly $\mu$, as this is often not possible because it can be between two answer possibilities.

### 3.4.1 Special Validators

As mentioned by Zhu and Carterette [11] there are some specific patterns which could occur, showing that there is the chance that the user does not answer the questions reasonably. To take this into account for ArtSimily, there are validators which check for specific user behavior at the end of the session. This can result in an additional session validity reduction. Mainly two variants were considered:

**Answer Patterns** If the user always gives the same answer for a given percentage of the questions (at a point where it is very unlikely to happen if they answer correctly) we can assume that the user did not answer in a meaningful way.

**Time Patterns** This applies only to microworker sessions, as normally a microworker goes through a task straight and does not pause for a longer time like a player might do. So if a microworker has a significantly different time gap between two answers compared to the average, we assume that they did not answer well. Also if the average time between answers is too low, we can assume that they did not answer the questions and just clicked through quickly to get the money.

## 3.5 Result Tuple

A *Result Tuple* always consists of one search media object and one result media object. A result tuple can occur multiple times over all queries, but only once per query. Result tuples are always used for questions with two media objects. As soon as a given number of questions with the same result tuple were answered, a Gaussian bell curve is calculated, which represents the current evaluation progress, the actual consensus of all the users and how certain ArtSimily is that this consensus is correct.

$$\mu = \frac{\sum_{k=0}^{n} a_k \cdot S_k}{\sum_{k=0}^{n} S_k} \tag{3.7}$$

$$\sigma = \frac{\sum_{k=0}^{n} (a_k - \mu)}{n+1} \tag{3.8}$$

Boundaries can be set at which point ArtSimily should assume that the value of $\mu$ can be considered as certain. If this stage is not reached, the $\sigma$ value can be used to control how certain the system is that the answer in $\mu$ is the correct one (the smaller $\sigma$, the more certain). Figure 3.2 shows an example of a Gaussian curve from a result tuple.

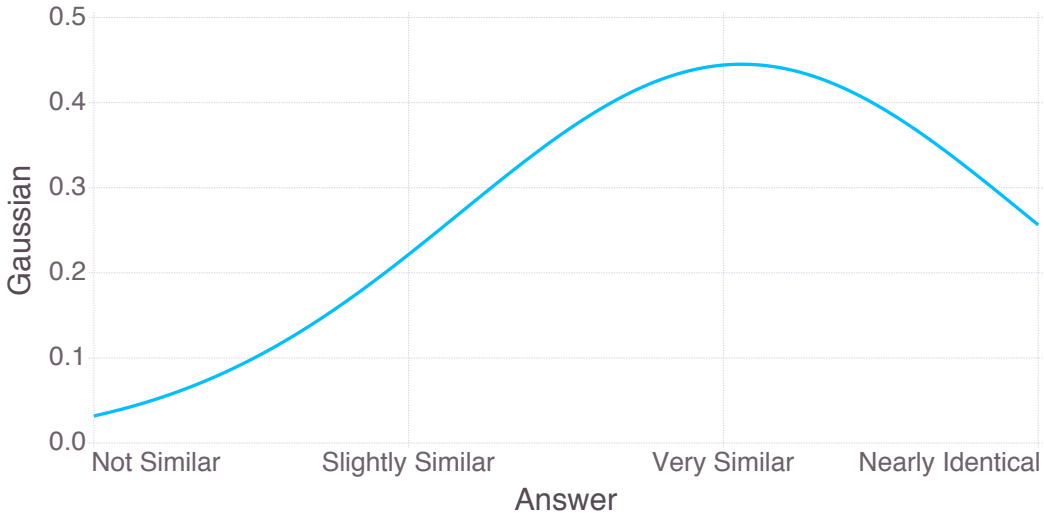



Figure 3.2: Example Gaussian curve with $\sigma = 0.896344$ and $\mu = 2.0586$

## 3.6 Game

When a player starts playing, an answer session is created. When they finish the session, the game will be saved and a score for their game is shown. When calculating the score we need to differentiate between two types of result tuples. All the answers where ArtSimily has enough data to calculate a Gaussian curve influence the game score more than the other answers. This is mainly because we want to give a score based on how good the user's answers are and it is only possible to determine this on questions where ArtSimily approximately knows what the correct answer is. Given $N$ answered result tuples with enough data to calculate the Gaussian and their curve functions $f_0, f_1, ..., f_N$ and answers $A$

$$A = \{a_1, a_2, ..., a_N\} \tag{3.9}$$

$$f_n(a) = \frac{1}{\sigma_n \cdot \sqrt{2\pi}} \cdot exp\left(-\frac{1}{2} \cdot \frac{(a_n - \mu_n)^2}{\sigma_n^2}\right) \tag{3.10}$$

$$F = \{f_1(a), f_2(a), ..., f_N(a)\} \tag{3.11}$$

and $M$ tuples which have not enough data

$$H = \{h_1, h_2, ..., h_M\} \tag{3.12}$$

we calculate the score

$$\text{Score} = \prod_{n=1}^{N} \left(C_{g1} + \frac{n}{N+M} \cdot C_{g2} - \frac{\sigma_n}{3} + \min(5, f_n(a_n))\right) \cdot \prod_{m=1}^{M} \left(C_{n1} + \frac{m}{N+M} \cdot C_{n2}\right) \tag{3.13}$$

where $C_{g1}$, $C_{g2}$, $C_{n1}$, $C_{n2}$ are constants which can be set to adjust the score range a little. $C_{g1}$ and $C_{n1}$ are set a little above one to ensure that for every answered question there is at least a small positive increase of the score. $C_{g2}$ and $C_{n2}$ weigh how much the score is influenced by the position of the answer in the session. The results of functions in $F$ need to be limited to a maximum in case the answers for a tuple were all the same. In such a case $\sigma$ of the curve would be zero and result in a division by zero. As workaround for this, ArtSimily takes a very small $\sigma$ instead of zero which produces fairly high values of $f_n(a)$. Otherwise such a narrow curve would strongly distort the score and validity calculation as all the other values except such big values would have no influence. That is why the outcome of $f_n(a)$ is bounded by five in the score calculation.

# 4 Implementation

This chapter shows some implementation specific decisions and characteristics of ArtSimily. As ArtSimily should be available online and easily accessible, we decided to create a web application which only requires a browser. The complete source code is published on Github[1]. A running instance is available at www.artsimily.com.
The original idea was to start using mostly microworkers for query evaluation and later add the gamification elements. This, because we were not sure if we could get enough users to play and with microworkers it was guaranteed that we would get answers. During implementation we changed this completely as for testing of the session validity it was necessary to be able to answer questions separately anyway, so the gamification part was implemented first. After the session validity was working the first microworker batches were started (see Section 5.2).

## 4.1 Data Storage

Media objects are uniquely identified by their SHA1 checksum and stored in a single directory. If the media object is an image, it is resized on import to avoid long loading times for the webpage. All other data is stored in the SQL database.

## 4.2 Players

Players use the gamification part of ArtSimily. They log in via an external OAuth[2] provider (such as Google, Facebook, etc.), allowing ArtSimily to just have players without any need for own cryptographic authentication (password hashing), session management and email validation. A player can receive achievements and answer questions in sessions. For each player there exists a profile page which is publicly viewable and optionally might contain a profile image from Gravatar[3].

[1] https://github.com/s3inlc/cineast-evaluator
[2] https://oauth.net/2/
[3] https://gravatar.com

## 4.3 Achievements

To motivate the players to play, especially to return regularly to the game, achievements can be unlocked. To use the gamification factor, some of the achievements should be quickly reached at the beginning and the later ones should be reachable with a reasonable effort. There are multiple aspects:

- To keep players playing over a long time, achievements can be received when players return everyday to play.
- As people often want to show what they achieved and compare each other, a public leaderboard is available.
- Players can get achievements by inviting other people to ArtSimily . This helps making it more popular and known to new people.

Achievements can easily be added in the source code and are quite independent from the other parts of the application[4].

## 4.4 Microworkers

Mechanical Turk offers two main approaches to run HITs. One way is to use their WYSIWYG editor to create forms directly on their page which will then be displayed to the workers, filled in, sent and saved by Mechanical Turk. The other way is to use an external page to which Mechanical Turk redirects and then returns back when the HIT is completed. The main disadvantage by using Mechanical Turk directly is that we would have to generate all the sessions before and create all the HITs manually with the WYSIWYG editor. Therefore we decided to go with the external approach[5]:

- The admin creates a batch of tasks on the admin interface of the tool. There a csv file of tokens can be exported.
- This list of tokens is fed into Amazon Mechanical Turk when publishing a batch. Mechanical Turk then will automatically create as many HITs as there are tokens.
- For every token there exists a unique link which a microworker clicks when they accept the HIT.
- After completion the tool gives a code to the microworker which they will then enter into Mechanical Turk to complete. This code is used to check that the microworker finished the session completely and later can be accepted/rejected.

In the background a script regularly checks for completed HITs and for every one of them it checks if the correct code was entered. Depending on the session validity $S_k$ the microworker's session received, it either accepts or rejects the HIT.

[4] https://github.com/s3inlc/cineast-evaluator/wiki/New-Achievements
[5] https://stackoverflow.com/questions/10769152/running-mturk-hits-on-external-website

## 4.5 Admin Interface

On the admin interface, admin users can manage and control the tool.

**Statistics** Global statistics and information about the progress of each query and the result tuples can be viewed.

**Queries** New queries can be uploaded and the corresponding elements can be viewed. In Figure 4.1 a query detail page is shown.

**Pruning** To avoid too many result tuples which have absolutely no similarity, queries can be pruned. This results in having reduced work which needs to be completed by players and microworkers and also increases the number of interesting tuples shown during sessions.

**Results** The current Gaussian curve for all result tuples where it is possible to calculate is visible.

**Microworkers** Microworker batches can be created and the current status for each single microworker is listed as shown in Figure 4.2.

**Answer Questions** Answer sessions can be started and where some of the questions can be answered. These sessions will always reach session validity 1 as the tool assumes that the admin always gives meaningful answers and therefore they can be taken into account strongly.

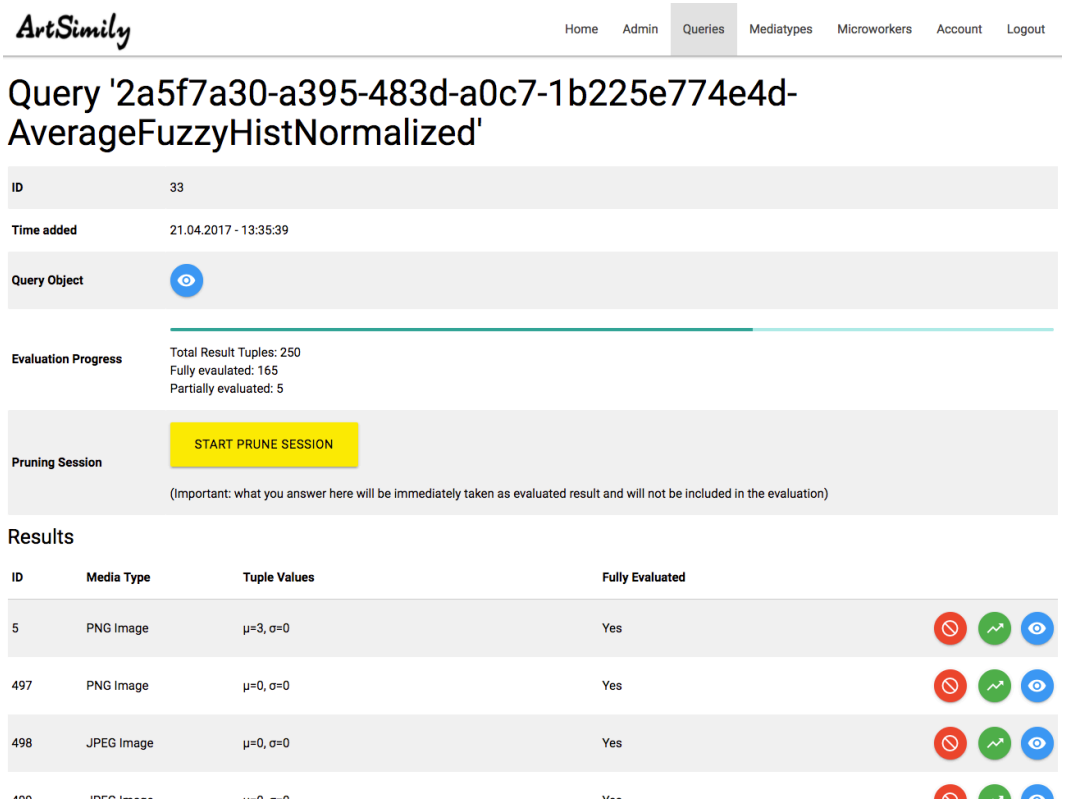



Figure 4.1: Query details page from the ArtSimily admin interface

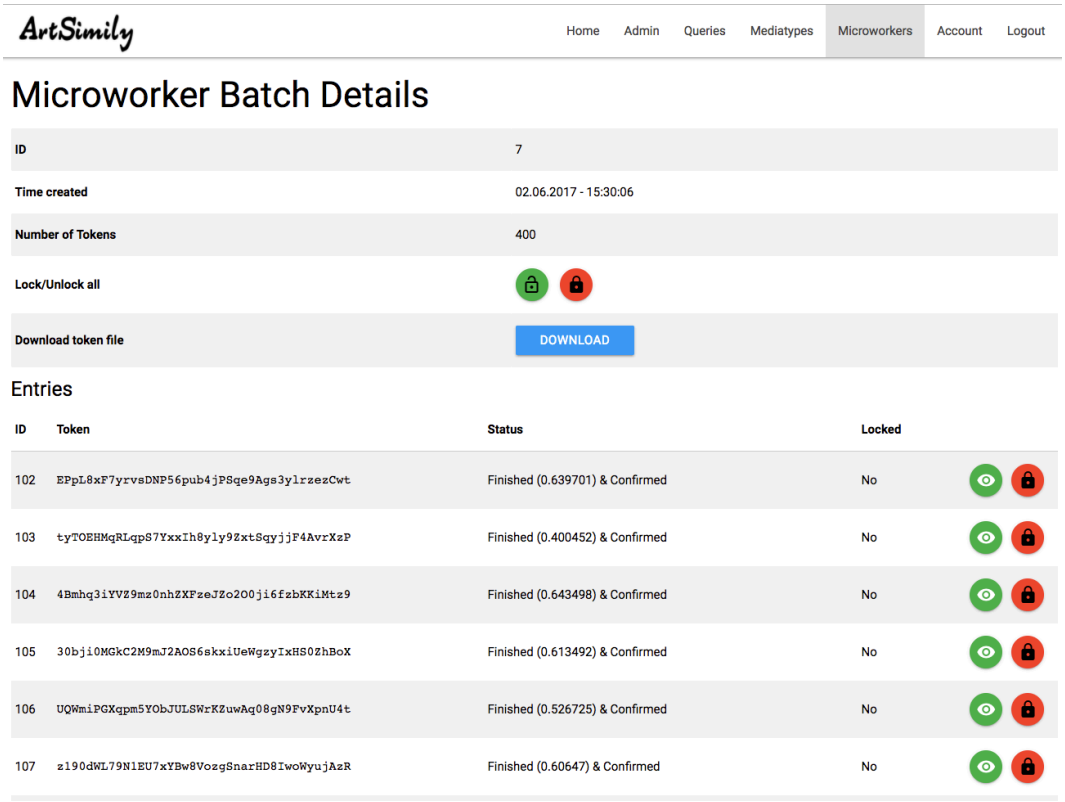



Figure 4.2: Batch details page from the ArtSimily admin interface

## 4.6 Background Processes

Beside the regular checking for completed microworker HITs, a script goes through all result tuples and checks for new answers hourly. It calculates the new Gaussian curve and saves the updated values. If a tuple meets the conditions to be finished, it will not be used in session questions anymore, it is then viewed as *fully evaluated*. This does not necessarily mean that the answer is clear, as it is possible that tuples might occur where no consensus can be found.

## 4.7 Challenges

During the implementation there were some challenges which required us to change some concepts slightly.

### 4.7.1 Ground truth

To determine if a user answers reasonably correctly on a session, ArtSimily must have some result tuples where it already knows the answer. If started completely empty at the beginning, the admin user needed to prune some result tuples and answer questions which then can be used as security questions.

### 4.7.2 Validity

The session validity should be able to separate players/microworkers which do not give meaningful answers from the others who are giving good information. To detect someone who is just clicking through the session as quickly as possible and/or just answers all the questions with the same answer, is fairly easy. But to detect a randomly answering person is challenging as it is expected that in some cases even the random clicker answers the security questions correctly and could get a high validity even though the answers for the other questions might be completely wrong.
Because of this, the session validity gets greatly reduced even if just some of the security questions were answered incorrectly. Even then it's not possible to catch all cases where a random answering person is luckily clicking correctly, but in most cases the validity gets heavily reduced. We simulated 100 randomly answered sessions and looked at the validity[6]. On Figure 4.3 we see that there are some outliers getting a good validity but the majority gets a bad or very bad validity.

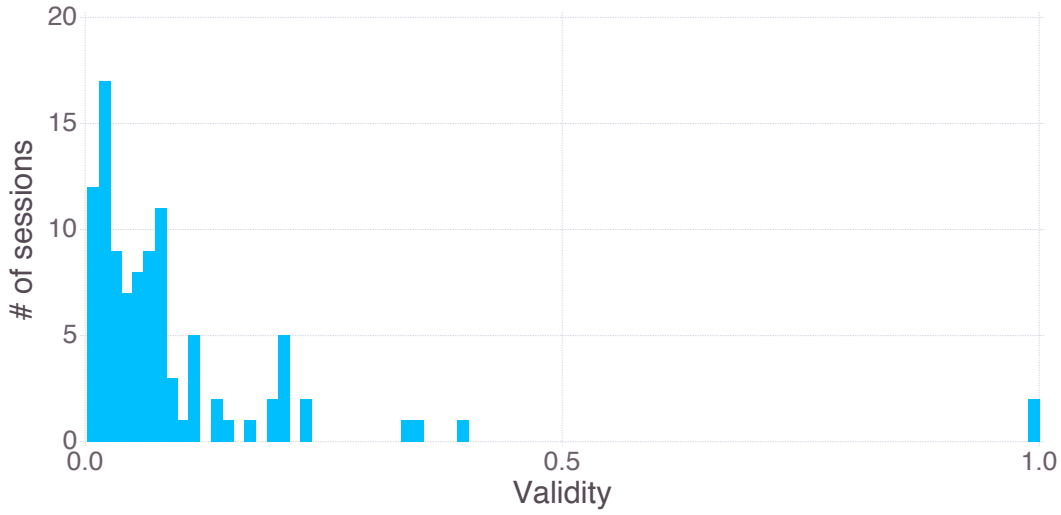



Figure 4.3: Histogram showing the distribution of the validities for 100 randomly answered sessions (mean 0.095).

Getting below these average numbers is quite difficult.
Having bad answers severely reduce the validity would mean that users who answer questions with good intentions could receive low validities. On the other hand we might have wrong answers, which sometimes have a higher validity than near zero. So the Gaussian curve would be slightly affected, but as the correct answers have high validities they are dominating the result. Therefore the influence of the wrong answers is negligible. As the results above show, ArtSimily uses the second approach.
There is a trade-off involved when punishing people for bad answers. Users who answer questions with good intentions might perceive similarity different than most other users. If we were to punish bad answers severely, those users would receive low validities. This is

[6] https://github.com/s3inlc/cineast-evaluator/blob/master/inc/script/validityMassTest.php

exactly what we are trying to avoid since the goal is to see the differences in perceptions. Thus, ArtSimily rewards good answers and answering security questions correctly. As mentioned at the beginning, this means that the average acceptance rate of a random clicker will not be zero, but around 10%.

# 5 Process

Once we had a running version of ArtSimily, we made it publicly accessible under the domain www.artsimily.com. In this chapter we describe our process of spreading ArtSimily and running the microtasks. This also includes the knowledge we gained during this process.

## 5.1 Players

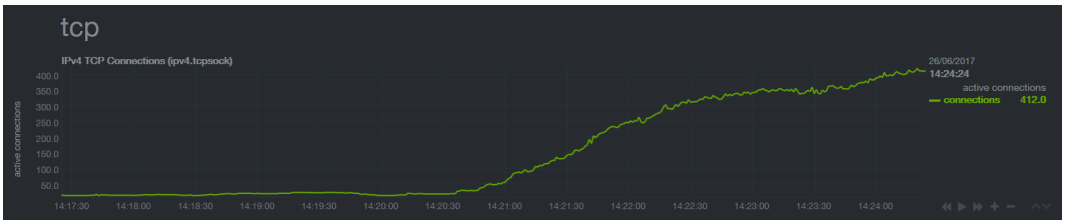
To get people start playing ArtSimily we first tried to motivate persons in our university, friends and family. After we made some smaller changes we also asked specific communities for feedback. Also the affiliate achievements of the game were designed that people spread the word about ArtSimily, where we hoped that players would invite more people.
We created a Twitter account for ArtSimily which was configured to retweet automatically when someone tweets their score[7].

## 5.2 Microworkers

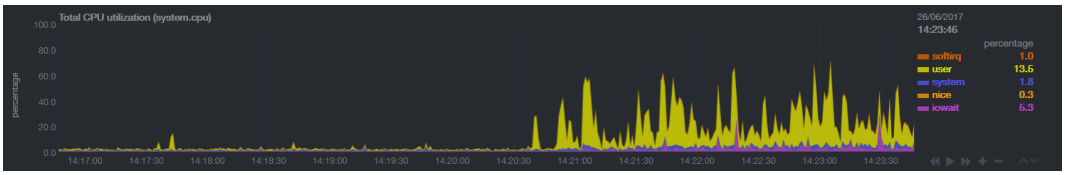
We had a budget of 100$ to run micro tasks on Mechanical Turk. After some research we decided to start with a price of 0.08$ per task, as recommendations were to better start too low than too high as stated by Mason and Suri [7]. Together with the fee for Amazon this gives costs of 0.1$ per task so we could run 1'000 tasks with our budget. On June 2nd we started with a first batch of 100 tasks. We were quite surprised that we had completed half of the batch very quickly. Shortly after this we decided to start a second batch of 400 tasks. After most of the tasks were finished we started running our automatic accepting/rejecting script. At the end we had 20 microworkers which were either not started/finished or we could not assign to any HIT from Mechanical Turk. In total we rejected 47 of the 500 tasks where 0.45 was set as session validity which was at least required to get accepted.
On the 26th and 29th of June we ran the remaining 500 tasks in batches of 200 and 300 tasks respectively. Because of the feedback from the first two batches (see Subsection 5.3.2) we decided to review all tasks which got a validity below the threshold, where we reject

[7] https://twitter.com/@ArtSimily

them, to catch sessions which were answered well, but for some reason got a bad validity. This way we had fewer rejections caused by a bad session validity.
When we started the microworker batches we quickly saw a growing number of connections and a rising load on the server running ArtSimily (see Figure 5.1). We saw that the computationally most expensive part is the database (yellow on Figure 5.1b) .




(a) Increasing number of connections




(b) Load on the server

Figure 5.1: Load and connections on the server after starting a microworker batch

## 5.3 Feedback & Reactions

### 5.3.1 Gamification

We got feedback about the gamification part of ArtSimily from various sources. Some input came directly from people we personally acquired to play. Other feedback was written by Reddit users where we asked for trying it out and give some opinions[8][9].
People were missing a more instant feedback when going through answers. They did not get any feedback before they got the score at the end of the game. This was the reason to add some motivational quotes which appear during the session.
Another problematic point was the score. As it is not communicated how exactly the score is calculated it is difficult for players to know exactly how they could improve the value. One feedback was that a user tested if the score changes based on what he viewed as similar and what not. He was able to say what could be more optimal at a certain point. Based on his experience he stated that in cases he is not sure, he would click *Not Similar* as he received higher scores on the games then. So therefore the consensus of the people was more pessimistic and when answering similar to this consensus the score gets higher.
The second score problem was the variety of scores which a player could receive. Heavily depending on the questions which were asked during a game, the score could get very high or stay quite low, even when the answers were good. This lead to multiple changes of the calculation method, see Section 3.6.

[8] https://www.reddit.com/r/WebGames/comments/6hdoxg/artsimily_a_game_for_science_where_you_look_at/
[9] https://www.reddit.com/r/gamification/comments/6hdvv8/looking_for_feedback_on_a_game_for_science/

### 5.3.2 Microworkers

Although all the batches were completed quite quickly, there were some workers which were not happy with the tasks. After running the batches we received some emails from microworkers. Mainly there were three types of complaints:

- Workers complained about the amount we pay for a task. They said that they really took time to answer the questions and do not agree with the small amount which is paid for such a long task.
- They did not agree with the rejection as they said they answered in a meaningful way.
- Some just sent rude comments.

After the first batch we were also negatively mentioned on Turkopticon[10]. Turkopticon is a website where workers can rate requesters depending on their experience with them. At the time this text was written, four microworkers complained about our microtasks. Two of them also mentioned the low price for such a task, the other two were stating that they got rejected for no reason in their opinion. They were wondering about the automatic validation which rejected their task which resulted that we stated this clearly in later batches. An interesting point is that both complained that they did not get an answer from us, one of them very shortly after the task was running, expecting we would reply immediately despite his email being sent in the late evening in the European time zone. The detailed texts from Turkopticon and Emails are included in the Appendix (see Section A.1).
We learned that the rate of microworkers just clicking the same answer all the time was rather low and most of them took the task seriously. Because of this the rejected ones from the first two batches reacted heavily. To catch such wrong rejections we reviewed all bad sessions before rejecting them definitely in the third and the fourth batch. We also discovered that some microworkers seemed to have problems with the workflow of the HIT and started clicking on the link to the session before they were accepting the hit. Some single users even managed to get out of the microworker process and visited ArtSimily on the normal page and started sessions there. So the work-flow should not be affected by such misbehavior and be able to handle this without the effect that microworkers cannot finish the task when they clicked wrong at some point.

[10] https://turkopticon.ucsd.edu

# 6 Evaluation

In this chapter we show the results which we obtained by running ArtSimily. We discuss the resulting data by giving some examples of evaluated result tuples, we show some information about how much ArtSimily was used and what we learned from this. Also, we compare the results of players and microworkers to determine if there are some specific differences.

## 6.1 Overall Results

The evaluation was conducted on the 3rd of July 2017. This was when the system was running for approximately 2 months. At this point we had the results summary, shown in Table 6.1.
Comparing the Gamification and Microworkers part we see in Table 6.2 that we got different amounts of data.
We have several outcomes here. The number of answered questions from microworkers is slightly larger than from players, but the number of sessions is lower. This is mainly because microworker sessions had approximately 100 questions, where game sessions were around 50

Table 6.1: Global numbers on ArtSimily at the time of evaluation

| **General** | |
|---|---|
| Total imported Queries | 2'305 |
| Total unique Result Tuples | 156'445 |
| Tuples which are fully evaluated | 42'727 |
| Tuples which have at least one answer | 94'822 |
| Number of sessions | 4'409 |
| Pruned Result Tuples | 30'196 |
| Total number of answers | 206'841 |

| **Gamification** | |
|---|---|
| Total players | 50 |
| Total played games | 1'387 |

| **Microworkers** | |
|---|---|
| Total microworkers | 864 |

Table 6.2: Comparing Microworkers to Gamification Players (Anonymous sessions included)

| | Microworkers | Gamification Players |
|---|---|---|
| Number of sessions | 1'375 | 2'775 |
| Answered questions | 101'617 | 75'028 |

and anonymous game sessions only around 20 questions. When we compare the value of sessions of players (from Table 6.1) to the total number of sessions in the gamification part, we see that there were many anonymous sessions. The discrepancy between the microworker sessions and the number of microworkers is because some started sessions which they did not complete and later were completed by another one (see Section 5.2).

## 6.2 Similarity of tuples

The ideal outcome for one result tuple would be that we can see a clear majority consensus and therefore assume this is the correct answer. But we expected that there might be tuples where people do not agree how similar two images are. In the following section, we discuss some examples of such tuples and ones which gave very clear results.

### 6.2.1 Wide Results

Figure 6.1 shows an example where the Gaussian curve is rather wide, even though there were more than 10 answers given. In Figure 6.1c the distribution of the answers is visible. This is an example where it is possible to view similarity in different ways. We assume that some people also looked at the style of the images and therefore these two images are somehow similar.

Figure 6.2 shows another example where it is not possible to say what the majority consensus is. When looking at the distribution of answers in Figure 6.2c, we see that it is nearly evenly distributed. There, people did not have the same opinion on whether such different flowers are very similar or not.

### 6.2.2 Clear Results

Figure 6.3 shows an example where the Gaussian curve got narrow after some answers where it is clearly visible that the majority agreed that they are either *Very Similar* or *Nearly Identical*. When we look at the exact answers in Figure 6.3c we see that the consensus clearly lays between the two answers and there is no answer outside of this. We assume that because of the color or the visible parts of the moon some users did not see it as *Nearly Identical*.

Another example of a good consensus is shown in Figure 6.4. For some reason there were some users saying that these images are not similar as seen in Figure 6.4c. But we see that nearly all said that these images are *Slightly Similar* and did not give them a better similarity rating. We assume that this is because of the different kind of images, the left one is a painting and on the right we have a photo. Furthermore the perspectives of the two images are not the same and therefore they were not rated highly similar.

(a) Images of result tuple


(b) Gaussian curve built by ArtSimily


(c) Answers given

Figure 6.1: Wide Gaussian curve result tuple example 1

(a) Images of result tuple


(b) Gaussian curve built by ArtSimily


(c) Answers given

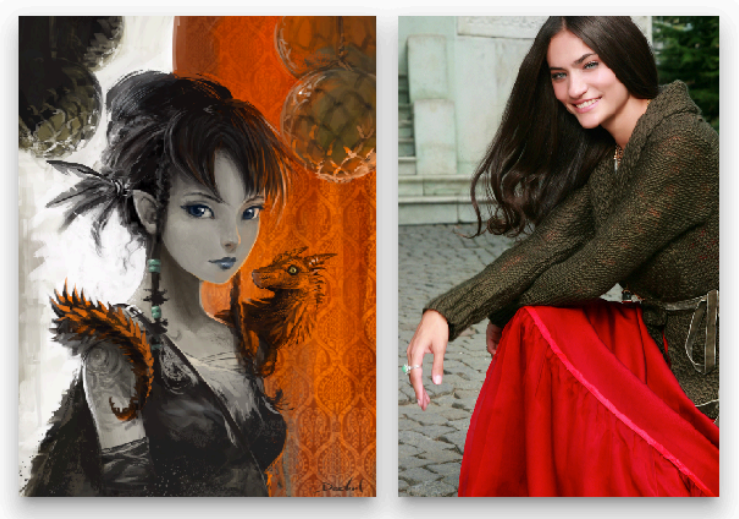
Figure 6.2: Wide Gaussian curve result tuple example 2

(a) Images of result tuple


(b) Gaussian curve built by ArtSimily


(c) Answers given

Figure 6.3: Narrow Gaussian curve result tuple example 1



(a) Images of result tuple


(b) Gaussian curve built by ArtSimily

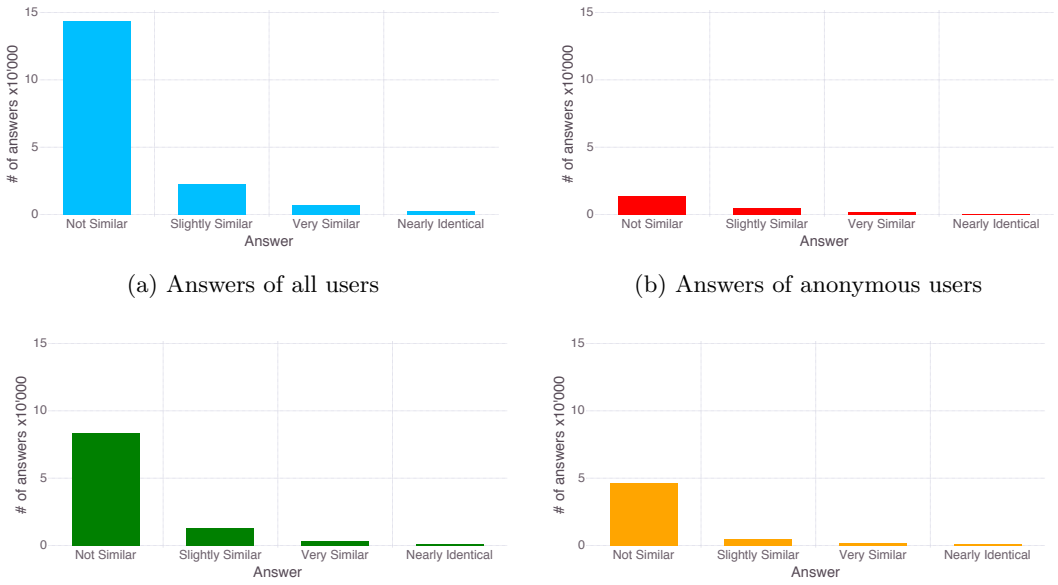

(c) Answers given

Figure 6.4: Narrow Gaussian curve result tuple example 2

## 6.3 Answer Sessions

As we have different user groups giving answers we can compare these and look if we get any difference on certain behaviors. We can always look at these three user groups: Microworkers, players and anonymous players. Sessions from the admin user are always excluded from the following results.

### 6.3.1 Similarity Answers




(a) Answers of all users

(b) Answers of anonymous users

(c) Answers of microworkers

(d) Answers of players

Figure 6.5: All answers given (excluding admin sessions)

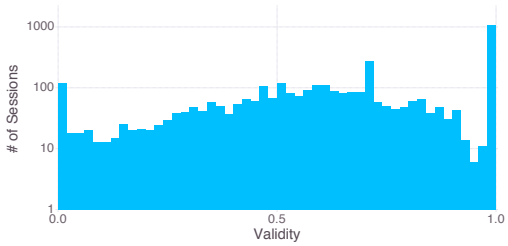
When we look at all the answers which were given by users in Figure 6.5, we see that there is a really strong dominance of *Not Similar* answers. The three user groups have a similar distribution except the anonymous users gave *Slightly Similar* more often as an answer.

### 6.3.2 Session Validities

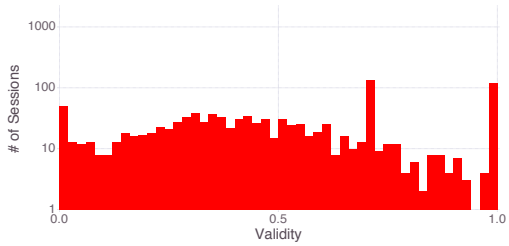
Figure 6.6 shows validity histograms of the different user groups of ArtSimily. Admin sessions were excluded as they got rated with validity 1.0 automatically and are therefore not interesting to take into account. On all four plots we see that there are peaks on 0 and 1. This is because the validity quickly gets near 0 when a user answers security questions incorrectly and it is possible to reach 1 when answering well.

Interesting is that the microworker validities seem to be the best (Figure 6.6c), followed by the players (Figure 6.6d). Often when players or microworkers got bad validities they got close to 0 because their answers were wrong. Between 0 and approximately 0.4 there are only rare occurrences of sessions.

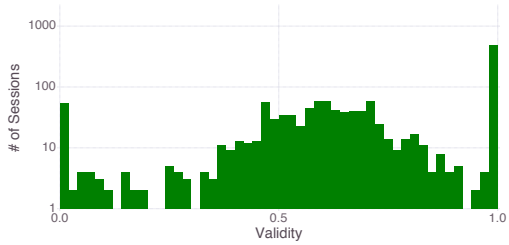
The peak at 0.7 (dominant on Figure 6.6b) is caused by the punishment of certain sessions (because of a pattern or a low average time) which were reduced by 30% even if they answered correctly. This happened on early sessions during development, when there were a lot of non-similar images and people had to correctly click *Not Similar* for the whole session.



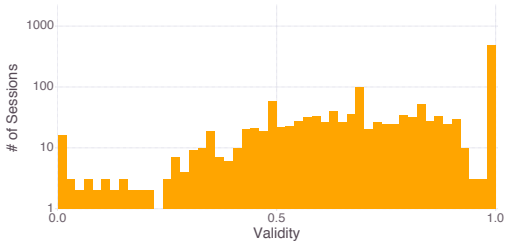

(a) Validities of all user sessions (average 0.6678)




(b) Validities of anonymous sessions (average 0.4958)

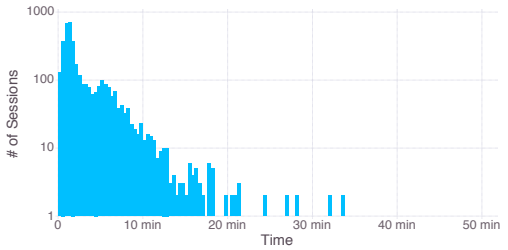



(c) Validities of microworker sessions (average 0.7149)

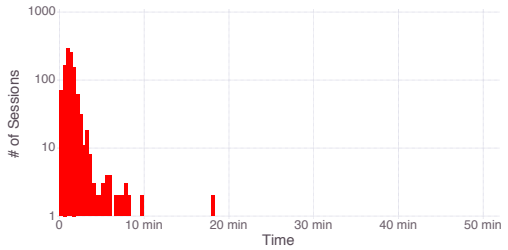



(d) Validities of player sessions (average 0.7575)

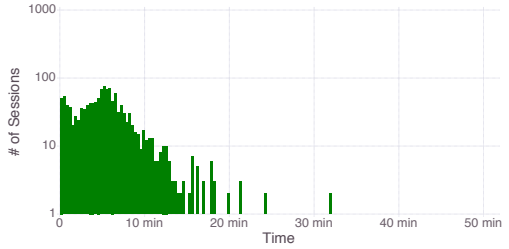
Figure 6.6: Validities of sessions (excluding admin sessions)



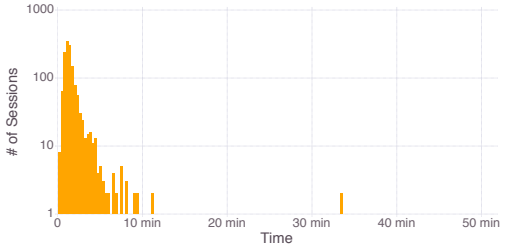

(a) All users




(b) Anonymous users




(c) Microworkers




(d) Players

Figure 6.7: Duration of sessions (excluding admin sessions)

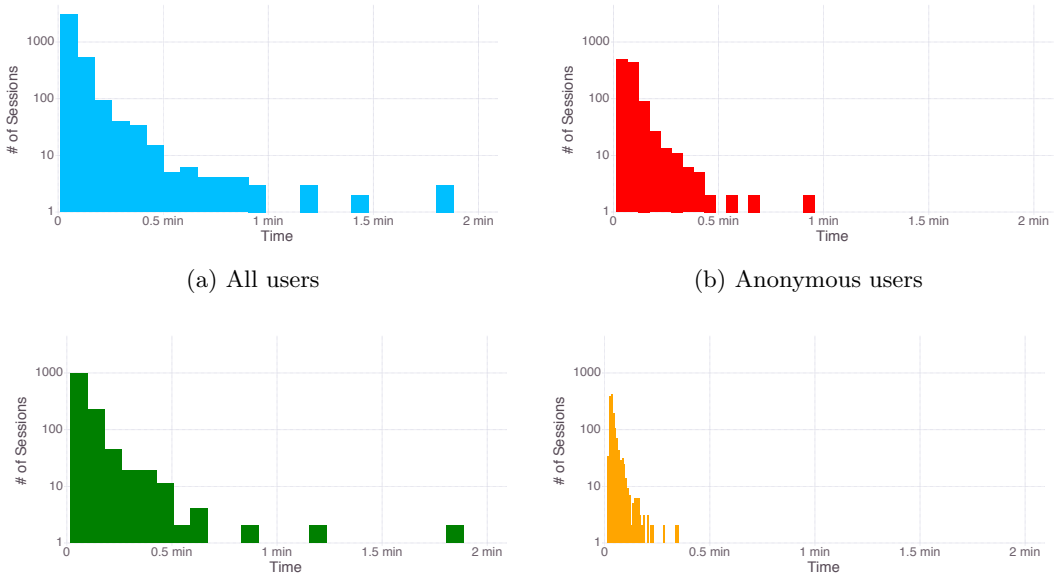
### 6.3.3 Session Duration

Figure 6.7 shows the duration of the answer sessions. Those which took longer than 1 hour were ignored on these histograms as there are sessions which were kept open over multiple days. Microworkers took significantly more time to complete sessions than players or anonymous users. However we need to keep in mind that microworker sessions also were longer than player and anonymous sessions.

Table 6.3: Average time required by Amazon Mechanical Turk on one HIT

| Batch # | Average time |
|---|---|
| 1 | 8.70 min |
| 2 | 7.07 min |
| 3 | 7.72 min |
| 4 | 7.73 min |

In Figure 6.8 we see the time per answer required by the users. Microworkers needed more time per answer than players. We assume that microworkers cared more about giving the correct answer and thought longer about two images than a player who maybe just answered more intuitively.




(a) All users

(b) Anonymous users

(c) Microworkers

(d) Players

Figure 6.8: How much time users required in average to answer (excluding admin sessions)

Regarding the average time the microworkers required to complete HITs (as displayed by Amazon Mechanical Turk) we were surprised that it was as high as shown in Table 6.3. We expected workers to be slightly faster and therefore decided for the amount of the 0.08$ paid per HIT. Having received the average time, it is understandable that some workers complained about the small reward.

## 6.4 Gamification

Even if the players and anonymous users gave a reasonable amount of answers, when we have a look at the results on the gamification part, we need to consider that a lot of answers were given by a small number of players compared to the number of unique microworkers. In Figure 6.9 we see that a lot of players played only a few games and a small fraction played a few hundred games.

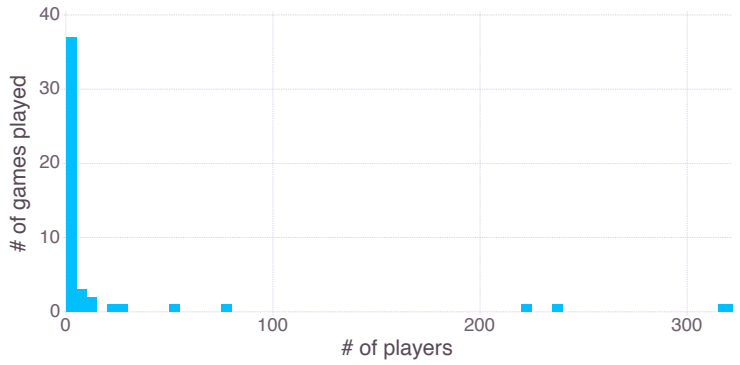



Figure 6.9: Games played by users

### 6.4.1 Returning Players

When looking at the number of times each player played it is clear that the return rate was not as high as we wished. Figure 6.10 shows that the majority of the players just played on a single day. These are just players which registered, played some games and then never returned. When we look at Figure 6.11 we see that at least some not only played one game on these single days, but still players are not very motivated to return to play on more days.

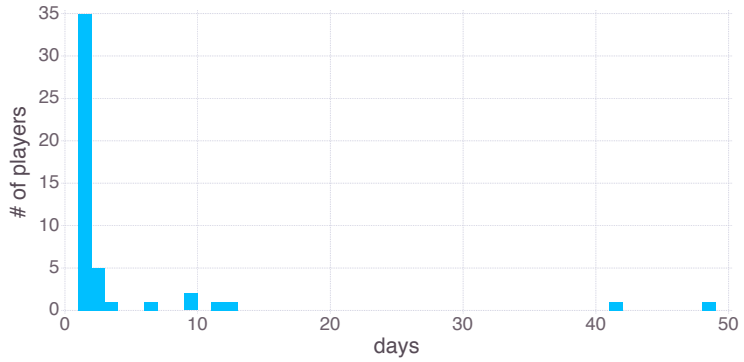



Figure 6.10: Number of days on which users played

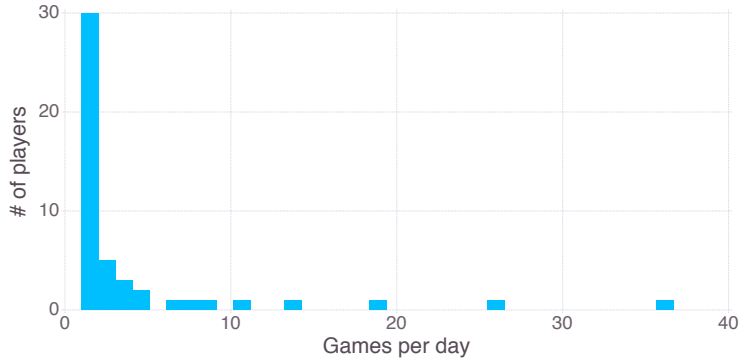


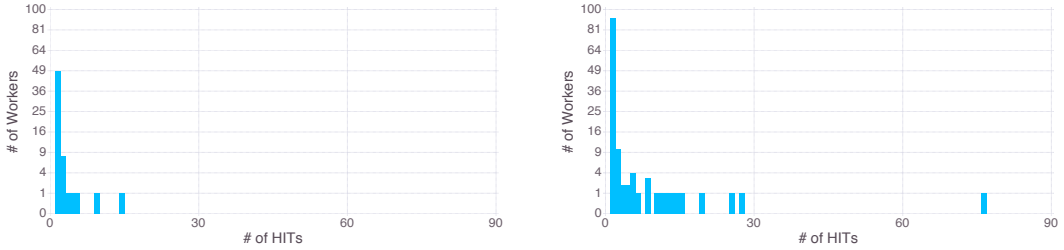

Figure 6.11: Number of games users played per day of playing

## 6.5 Microworkers

In this section we take a look at some specific behaviors of the microworkers from the four batches.

### 6.5.1 Unique Workers

In Figure 6.12 we see that the majority just completed one HIT. Beside some outliers the other workers completed up to 10 HITs. There is an extreme outlier of one single worker who completed over 200 HITs throughout all 4 batches.
We looked closely at some randomly chosen sessions which were completed by this microworker. They managed to get accepted nearly all the sessions, meaning they had a validity of 0.45 or higher. Most of their answers made sense, but there were also some, where it was not clear why they answered like this. For future runs of microworkers it should be changed so that every worker can only do a session once. This way, such outliers would be prevented and it would result in more unique opinions.




(a) Batch 1 (100 HITs)

(b) Batch 2 (400 HITs)

(c) Batch 3 (200 HITs)

(d) Batch 4 (300 HITs)

Figure 6.12: Number of HITs unique workers completed per batch

### 6.5.2 Rejected Workers

When we reviewed microworker sessions on the third and fourth batch, which got a low validity before we automatically rejected them, we saw that there were larger differences on the microworker's perception of similarity. Some workers answered quite optimistically whereas others in a more pessimistic way. In both cases if this tendency was too large, it caused low validities.

# 7 Conclusion

In this chapter we conclude our results from running ArtSimily. Furthermore, we discuss some improvements, changes and additional research which could be done in future work.

## 7.1 Conclusions

ArtSimily was planned to be used to evaluate multimedia query results and as the evaluation has shown we can obtain reasonable results for evaluated queries. Both ways of gathering data (microworkers and gamification) were implemented and used. We hoped that we could attract more players to play ArtSimily. But the elements we used to motivate players did not have as strong an effect as we expected.
On some compared media objects we could clearly see the perception differences of similarity. So in some cases there is no majority consensus and therefore the similarity is not clearly defined, even for a human.
We plan to keep ArtSimily running in the future. It will be used for new evaluations and hopefully improved to get better and more answers from players and microworkers.

## 7.2 Future Work

Even if all the functionalities which are required to make evaluations possible are implemented and working now, there are a lot of future features which can be added. This section describes some of the ideas which can be used to improve ArtSimily.
Generally, some parts of the back-end could be reviewed and improved in performance to overcome speed issues which might occur when a huge amount of new queries would be added.

### 7.2.1 Multimedia

As ArtSimily is designed to evaluate multimedia query results, the usage is not bound to images only. So far it was just used with images, but is ready to be filled with other multimedia objects. Special cases occur when the results are for example clips from a longer

video sequence. In this case the handling of this data has to be discussed and the import procedure needs to be adapted. The same applies to long audio files.
Also the storage might get improved as currently all files are stored in the same folder which results in a directory with a huge number of files in it. This could for example be improved by using subdirectories using the first $n$ chars of the SHA1 checksum.

### 7.2.2 Gamification

The evaluation of ArtSimily has shown that the gamification needs to be improved. Players were interested to play, but it was hard to make them return regularly. The direct gaming experience needs to be improved, so players get better immediate feedback during a game. Some testers also stated that the session might be a bit too long. The solution here would be to make the session length progressively changing depending on how much the user plays. This would require to also have the score calculation changed as it is currently highly dependent on the length of the session.

### 7.2.3 Microworkers

If required, ArtSimily could be extended to support other microworker providers than Amazon Mechanical Turk. It is also recommended that the microworker handling process gets slightly changed to limit to one HIT per microworker. Further, these changes could be used to make the usage more bulletproof for any mishandling of the HIT (for example users clicked on the link without accepting the HIT).
For the session validity it might reduce the number of HITs which would become rejected because of a general other view of similarity, by checking if the difference to the expected answer is similar over the whole session. For example when users have a generally more optimistic opinion on similarity, they always give a slightly higher answer for the similarity and therefore would get a bad session validity. Such a user should still get a good session validity as they gave good answers, but with the current session validity this would be an edge case where they won't pass the automated HIT validation.

### 7.2.4 Admin Interface

On the admin interface there are several points which can be improved.

**Answer Histogram** When reviewing an answer session (especially the microworker ones) it could help to have a histogram of all the answers displayed. This way it is visible very quickly if the worker for example just always answered the same.

**Query Evaluation** At the point where queries are fully evaluated, additional information could be displayed. For example the discounted cumulative gain.

**Query Priorities** In the backend ArtSimily is ready to handle different priorities for queries to force specific queries getting evaluated more quickly, it is currently just not possible to set on the admin interface.

# Bibliography


[1] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5, 2011.

[2] David Codish and Gilad Ravid. Adaptive approach for gamification optimization. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 609–610. IEEE Computer Society, 2014.

[3] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: defining gamification. In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, pages 9–15. ACM, 2011.

[4] Ivan Giangreco and Heiko Schuldt. Adampro: Database support for big multimedia retrieval. *Datenbank-Spektrum*, 16(1):17–26, 2016.

[5] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 453–456. ACM, 2008.

[6] Mathias Lux, Mario Guggenberger, and Michael Riegler. Picturesort: gamification of image ranking. In *Proceedings of the First International Workshop on Gamification for Information Retrieval*, pages 57–60. ACM, 2014.

[7] Winter Mason and Siddharth Suri. Conducting behavioral research on amazons mechanical turk. *Behavior research methods*, 44(1):1–23, 2012.

[8] Luca Rossetto, Ivan Giangreco, and Heiko Schuldt. Cineast: a multi-feature sketch-based video retrieval engine. In *Multimedia (ISM), 2014 IEEE International Symposium on*, pages 18–23. IEEE, 2014.

[9] Luca Rossetto, Ivan Giangreco, Claudiu Tanase, and Heiko Schuldt. vitrivr: A flexible retrieval stack supporting multiple query modes for searching in multimedia collections. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 1183–1186. ACM, 2016.

[10] Julián Urbano, Jorge Morato, Mónica Marrero, and Diego Martín. Crowdsourcing preference judgments for evaluation of music similarity tasks. In *ACM SIGIR workshop on crowdsourcing for search evaluation*, pages 9–16, 2010.

[11] Dongqing Zhu and Ben Carterette. An analysis of assessor behavior in crowdsourced preference judgments. In *SIGIR 2010 workshop on crowdsourcing for search evaluation*, pages 17–20, 2010.

# A
# Appendix

## A.1 Mechanical Turk

Figures and Texts in this section show the reactions we received from some microworkers.

Text A.1: Email reaction from *ca***ants@yahoo.com*

```
sir ,

images counting high. payment is low. increase amount then we do
regularly. pls increase amount.
```

Text A.2: Email reaction from *ch***ron@ace.tamut.edu*

```
I'm curious how they didn't pass automated validation. To my
knowledge there wasn't a specific definition of similarities, as
such I picked things based on what I though was similar. ex:
does the background have similar colorization, are there similar
objects, etc. Each of these things I weighed and added up whether
I thought each image held some similarity to the other image
shown.

I can redo the task if you would like with some idea of what
similar means in the case of this task.

Thank you for your time,
Christopher
```

Text A.3: Email reaction from *af***18@gmail.com*

```
How is that possible.? I entered the right survey code after
completion. Pls reverse my rejecyed hit.:(
```

Text A.4: Email reaction from *wo***tjs@gmail.com*

```
Hi I have just received a rejection from you regarding this hit.

Can I have an explanation to how it was rejected please.

I took my time on this hit, and found it fun and interesting to
participate in, so very surprised about this. If I had rushed
through it fair enough! but I did not! I took the hit seriously!
I gave my honest opinion on each image!

Would highly appreciate a response. As highly confused.

Thanks.
```

Text A.5: Email reaction from *af***idy@yahoo.com*

```
My work was rejected because it "did not pass automated
validation" — what does that mean?

Rejections like this can seriously harm workers' accounts.

For 8 cents I looked at and gave honest impressions on what seemed
like 100 sets of images! Can this be appealed or reversed?

If not, I'm afraid I'll have to leave a very negative review for
others on Turkopticon.
```

Text A.6: Email reaction from *rr***57@gmail.com*

```
piss off
```

Text A.7: Email reaction from *jd***ce2@yahoo.com*

```
What? this is ridiculous I worked hard at that sounds to me like
you are cheating people out of money. I will be reporting you
```

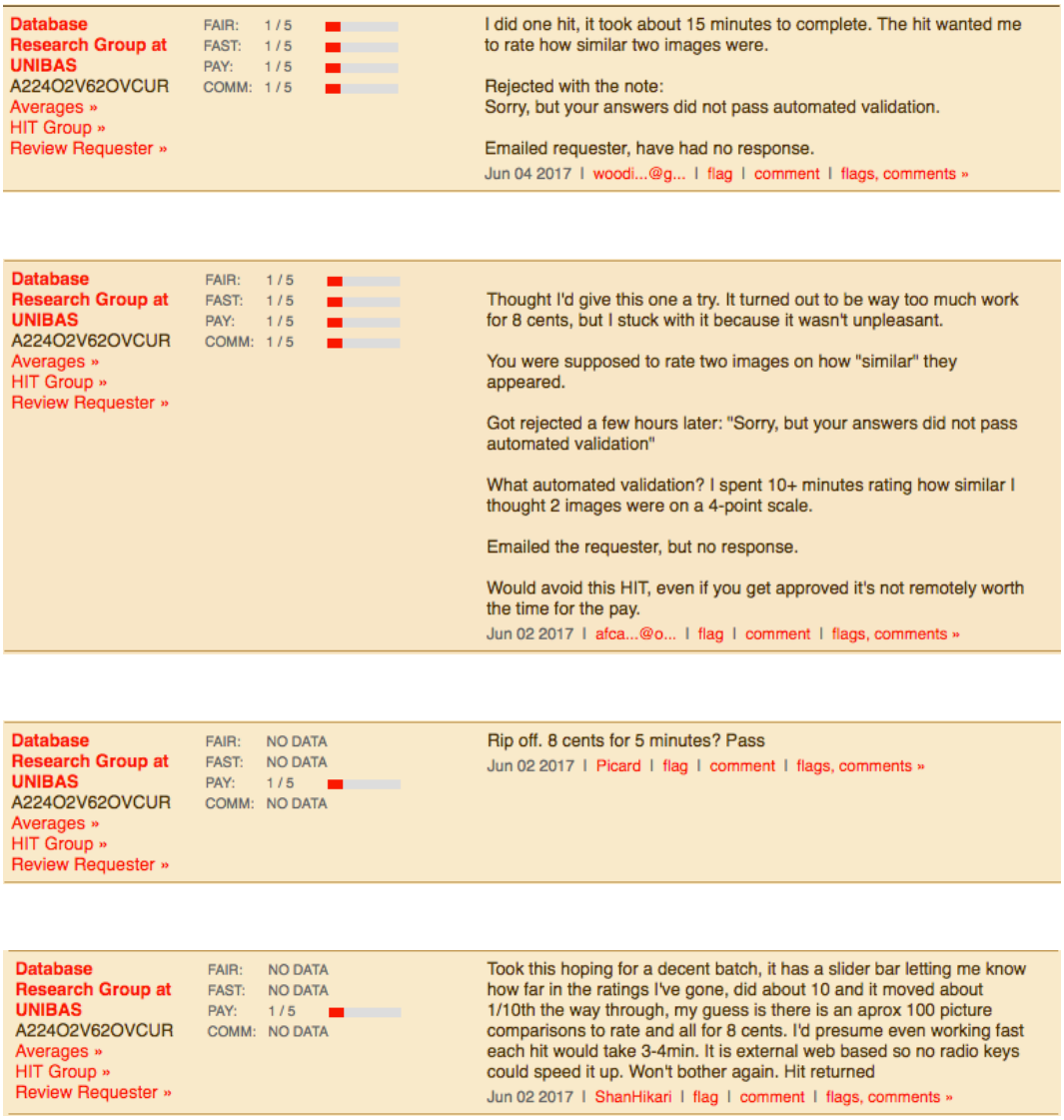



Figure A.1: Reviews on Turkopticon (https://turkopticon.ucsd.edu)

# B
# Appendix

## B.1 Database Model

Figure B.1 shows the database model of ArtSimily.

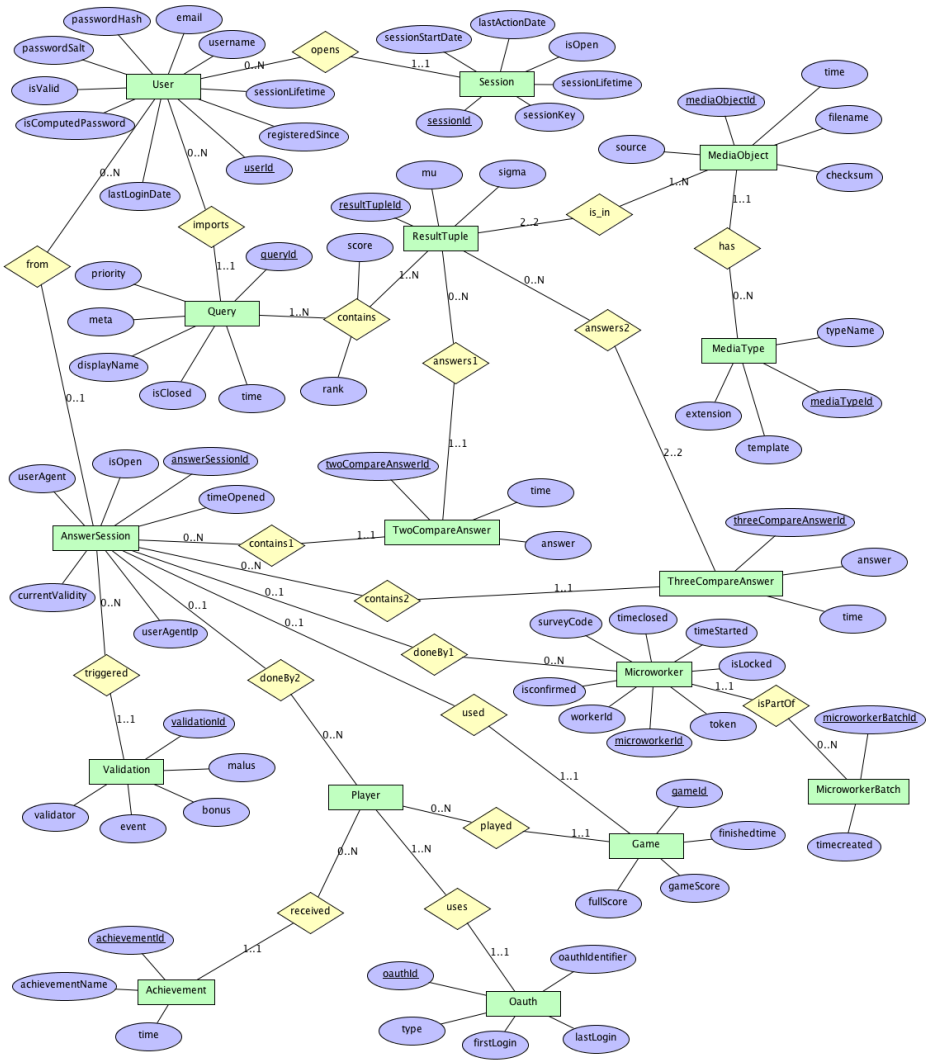



Figure B.1: ER diagram