INSTITUT FÜR INFORMATIK
der Ludwig-Maximilians-Universität München

Automatic Semantic Categorization of Image Annotations Generated by Games with a Purpose

Nicola Greth

Masterarbeit

Aufgabensteller	Prof. Dr. François Bry
Betreuer	Prof. Dr. François Bry, Martin Bogner
Abgabe am	30.04.2019

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst habe und keine anderen als die angegebenen Hilfsmittel verwendet habe.

München, den 30.04.2019 Nicola Greth

Abstract

ARTigo is a web platform which collects semantic annotations on digitized artwork using Games with a Purpose. The platform's goal is to provide a large database of tagged artwork, which can be retrieved by search engines for art historic purposes. Most of ARTigo's games follow the model of the ESP Game by Luis von Ahn. In this type of game two players are presented with the same image and try to enter tags the other player might also use to describe it. By this approach, only tags that are entered by both players independently are validated.

While the current version of ARTigo already contains a huge amount of tags there is only few knowledge about their relations or semantic groups. For this reason, the collected data is enhanced with each tag's cleaned version as well as its lexical and semantic categories. Therefore, a Semantic Categorization Pipeline is implemented in Python for the three languages German, English and French. An evaluation of samples of the resulting categories shows their correctness at around 70% for all three languages.

As the platform has been in operation since 2008 it has encountered some problems with the underlying technology and the performance of the servers. Therefore, during this thesis the three main games ARTigo Game, ARTigo Taboo and Combino are reimplemented with JavaScript technology and combined with the functionality of the Semantic Categorization Pipeline. New features, like the usage of cleaned tags, spell checking and tag corrections are presented.

Zusammenfassung

ARTigo ist eine Web-Plattform, die mit Hilfe von Games with a Purpose semantische Annotationen (Tags) von digitalisierten Kunstwerken sammelt. Das Ziel der Plattform ist es, eine große Datenbank annotierter Kunstwerke zur Verfügung zu stellen, welche von Suchmaschinen für kunsthistorische Zwecke abgefragt werden kann. Die meisten Spiele der ARTigo Plattform basieren auf dem ESP Spiel, welches von Luis von Ahn entwickelt wurde. Bei diesem Ansatz wird zwei Spielern jeweils dasselbe Kunstwerk präsentiert. Die Spieler versuchen dieses mit Tags zu versehen, welche der Gegenspieler ebenfalls eingegeben hat. Durch dieses Konzept werden nur Tags, die von beiden Spielern eingegeben wurden, validiert.

Obwohl die aktuelle Version von ARTigo bereits eine große Menge an Tags gesammelt hat, gibt es nicht viele Informationen über die Zusammenhänge der Tags untereinander oder semantische Gruppen, die diese bilden. Aus diesem Grund werden die gesammelten Daten um die bereinigte Version jedes Tags sowie seiner lexikalischen und semantischen Kategorien erweitert. Eine Pipeline zur semantischen Kategorisierung wird in Python für die Sprachen Deutsch, Englisch und Französisch implementiert. Die Evaluation einer Stichprobe der daraus resultierenden Kategorien zeigt deren Korrektheit bei circa 70% für alle drei Sprachen.

Da die Plattform seit 2008 in Betrieb ist, ist sie einigen Problemen mit der zugrundeliegenden Technologie und der Performanz der Server ausgesetzt. Aus diesem Grund sollen die drei Spiele ARTigo Game, ARTigo Taboo und Combino während dieser Arbeit mit modernster JavaScript Technologie neu implementiert und mit der Funktionalität der semantischen Kategorisierungs-Pipeline verknüpft werden. Zusätzliche Features, wie die Benutzung der bereinigten Tags, eine Rechtschreibprüfung sowie der Möglichkeit zur Korrektur von Tags werden vorgestellt.

Acknowledgments

I would first like to thank Prof. Dr. François Bry and Martin Bogner for their outstanding support of this thesis and the possibility to collaborate in the context of the ARTigo project. I highly appreciate that you led me in the direction of Semantic Categorization, which turned out to be a great topic for my master's thesis, and always had an open ear for my questions and concerns.

I also want to express my gratitude to all members of the teaching and research unit for Programming and Modelling Languages for all of their helpful advice, comments and ideas during my presentations that also contributed a great deal to the success of my thesis.

Finally, I would like to thank my friends and family, especially my brother Peter, who have supported me through the months of implementing and writing this thesis, brew a lot of Coffee for me and motivated me tremendously.

Contents

CHAPTER 1

Introduction

In the last decades, the power of computers has increased enormously. What previously lasted months for humans to calculate can now be done in seconds using computing power. The approximation of the mathematical constant pi, for example, has been of interest even before christ. Archimedes approximated three of its decimal places correctly in 250 before christ. In 480 after christ Tsu Ch'ung Chi, a chinese mathematician, was able to estimate seven correct decimal places. William Shanks calculated 707 decimal places in 1874, which took him fifteen years, however only 527 of them were later deemed correct. But with the uprise of computer technology in the 1950s thousands and later millions of decimal places could be approximated, for example by Jean Guilloud and Martin Bouyer, who were the first to find over a million of pi's decimal places [BBBP97]. In 1997, IBM's system Deep Blue won a chess game against the russian grandmaster Garry Kasparov, simply by following mathematical rules and calculating the best moves. Even in the field of Artificial Intelligence (AI), major accomplishments have been achieved. In 2011, IBM's computer system Watson beat the two best human players in the television quiz show Jeopardy, for which it had to understand spoken questions in natural language, associate and interpret information and understand irony, jokes or wordplay. Computers are outperforming humans in many situations like quick responses, multi-tasking or storing huge amounts of data [COC13, Mar11, Tho10]. However, there are still some matters, especially in the area of AI, that computers cannot yet accomplish on their own: The most important abilities for which humans stand out against computers are intuition, creativity, perception and a general understanding of the world. Where computers have to rely on computing every possible solution of a given task and ranking those by statistical confidence values, humans can simply use their basic understanding of concepts and cognitive capabilities to come up with the correct solution. Additionally, humans tend to notice extraordinary characteristics and scrutinize what they see, whereas computers are only focused on the task they were given [Sav12]. For example during the project Zooniverse that uses humans to identify objects in astronomical images, a new galaxy was found by a participant, who spotted and questioned a mysterious glow [COC13, Sav12, vA06].

1.1 Human Computation

Human Computation can be defined as "multi-agent information processing systems in which humans participate as computational elements" [Mic13b] or "systems that combine humans and computers to solve large-scale problems that neither can solve alone" [vA]. It therefore describes the research area that utilizes human intelligence in solving computational problems that computer algorithms are not yet able to solve independently [LvA11].

An adjacent domain to Human Computation is Crowdsourcing, which stands for "outsourcing to crowds". In Crowdsourcing, a given task is split into smaller pieces and is distributed to multiple people, who each work on their piece of the task. The idea of collaborating to achieve a result that would not have been attainable when working alone, has been widely used in other domains as well. [LP13, Mic13b] In many areas like biology or system theory it is known as Swarm Intelligence or Emergence and can be witnessed for example in ants identifying their best way to their food source by working together through pheromone communication [BDT99]. Famous crowdsourced projects are Wikipedia, Flickr or the recommendation system of Amazon. All of these examples for Crowdsourcing could also be examples for Human Computation, as many projects in Human Computation also require a large number of people, but there is a major difference between the two fields. Crowdsourcing does not focus on the computation, the type of task the group members handle is not the main element. But in Human Computation the task has to be computational [Mic13a]. A computational task arises from a scientific problem that is partitioned into elemental components and can be solved by following specific directions. Human Computation also requires a cooperation of human and computer systems, the assignment therefore has to include at least one human and one computer and can develop either in the direction of crowdsourcing including at least on computer or in the direction of Machine Learning with human control. A crowdsourcing project, where people work on a task that does not require any form of computer interaction or is not computational would not be considered Human Computation [COC13, LvA11, Mic13a].

There are different incentives for people to spend their free time on Human Computation. While Citizen Science is mostly achieved by altruism, interest in science or learning [RBG+13, JKS+16], Games with a Purpose motivate people with the opportunity to play a game, have fun and rise up in high scores. Other projects like reCAPTCHA give purpose to an already existing activity and platforms like Amazon Mechanical Turk simply pay its workers small amounts of money for their time [LA17]. Examples for the different incentives will be presented in detail in the following sections.

1.1.1 Unconscious contribution

One example for the supremacy of humans are CAPTCHAs. CAPTCHAs are widely used to prohibit bots from accessing web pages like registration forms or online polls. For example in November 1999, an online poll about the best graduate school in computer science was outwitted by students of the schools in question and became a battle between voting bots. To assure the legitimacy of online polls, it therefore has to be made certain that only humans are allowed to vote. CAPTCHA stands for "Completely Automated Public Turing Test to Tell Computers and Humans Apart" and serves as a challenge that humans can solve, while computers cannot. As the name states, CAPTCHAs are similar to the Turing Test, where a human judge has to distinguish a human and a computer by asking questions. The difference to the Turing Test is that the judge in the case of CAPTCHAs is the distinguishing method and it is not based on questions but on human perception. There are different types of CAPTCHAs. The most common CAPTCHAs are distorted images

of text that the user should recognize and enter. These forms of CAPTCHAs are based on the still existing challenges of Optical Character Recognition (OCR), which is the field of digitizing text from images, for example from scanned books. Other types of CAPTCHAs are based on the recognition of visual patterns, for example shapes or typing numbers from a distorted sound clip [vAB05, vABL04].
In 2008, Luis von Ahn estimated that every day around a hundred million CAPTCHAs were typed by people around the globe, which summed up to hundreds of thousands of wasted human hours per day. But since CAPTCHAs were effectively distinguishing between bots and humans it was unthinkable to refrain from using them, Luis von Ahn and his team came up with the concept of reCAPTCHA, to convert those wasted human hours to something useful. Since CAPTCHAs were solving challenges of OCR, they decided to use real scanned words that OCR could not recognize correctly. However, to maintain the original purpose of CAPTCHAs the typed word had to be verified. Therefore reCAPTCHA provides the user with two words, one unknown scanned word and a control word. If the control word is typed properly, the unknown word is assumed to be valid as well, since the user has no knowledge about which word is chosen as control word. When multiple users agree on the same word for an unrecognizable word, its transcription is saved [vAMM+08]. By using reCAPTCHA, as of October 2015, around 25 million books have been successfully scanned. In 2009 reCAPTCHA has been acquired by Google, which added other challenges as well. In 2012 Google used reCAPTCHA to transcribe house numbers or other street signs from images to enhance data in Google Maps [Hav15]. Additionally in 2014, Google started to use reCAPTCHA to train its algorithms in the field of Artificial Intelligence which makes the user select all similar images to a given object on a grid, for example with the instruction *Click on all images that show cats!* [O'M18]. **Figure 1.1** shows examples for an original reCAPTCHA task and a reCAPTCHA task including street signs.

Figure 1.1: reCAPTCHA [Hav15]

1.1.2 Monetary reward

In many scientific projects, experiments or evaluations are quite expensive and time intensive. Data has to be collected from a large set of participants to make sure that the results are statistically reliable and those participants and their experimenters have to be paid. In 2005 Amazon therefore started its Mechanical Turk market to use crowdsourcing for tasks, which were costly in terms of labor. Mechanical Turk is a service that allows researchers, called requesters, to get results quickly and inexpensive by getting their users, called workers, to complete micro-tasks which can be rewarded by a small monetary reward. A requester's task has to be split into micro-tasks, called Human Intelligence Tasks (HITs), that should be easy and completable in a few minutes. Because of the short duration of those tasks, their rewards are also as low as one cent up to one dollar per HIT. Crowdsourcing research on platforms like Amazon Mechanical Turk has a lot of advantages for scientists. Experiments can be performed across different cultures, countries and levels of population, but can also be restricted to a specific demographic group. Studies also show that with a good task design and pre-tests to identify suspect workers, the results on Ama-

zon Mechanical Turk are comparable to results achieved in real-life studies or experiments [KCS08, PCI10]. While Amazon Mechanical Turk has a lot of advantages for requesters, their workers have to face a lot of drawbacks. While many workers only participate in HITs because it is fun or a good pastime, for others Amazon Mechanical Turk actually is their primary source of income. A recent study shows that the median hourly wage of a worker is about as low as two dollars an hour which is far below the federal minimum wage in America, where most of the workers live [HAM+18, PCI10, Sem18].

1.1.3 Organizational and social incentives

With Amazon Mechanical Turk and reCAPTCHA, the tagged images were provided by the system. The social tagging system Flickr follows a different approach. With Flickr, users can upload their personal self-taken pictures and tag them for their own organizational or social incentives while enjoying the possibilities of a social network with their friends. Additionally, only the image owners are able to tag their images, which leads to a set of tags that, instead of the image's content, focuses more on the event, time, location or persons in context with the image, for example *wedding*, *2006*, *New York* or *Susanna*. Flickr uses the tags for their public image retrieval search as well as for navigational or organizational purposes [MNBD06, Bö08]. Flickr has grown to contain billions of photos and is an important source for knowledge retrieval [Fli]. For example by analyzing the geographical labels and tagging patterns of photos of Flickr users, events and their locations can be extracted. Additionally, Flickr provides a large dataset of tagged images that could be used as training data for supervised learning algorithms in the field of computer vision [KNA+07].

1.1.4 Interest in science

As Flickr has shown, there can be large differences in how people tag images depending on the context of the tagging task. For example in museums the artwork is described with tags by their curators, which in many cases deviates from the way a museum's visitor would depict an artwork. The perspective of curators refers to a greater extent to the meaning of the artwork, while the visitor's perspective would rather name the content it can visibly observe in the artwork. To improve the communication between the museum and its visitors, a collaboration between nine art museums in America, called Steve.museum, united to research and develop social tagging tools to reconnect the visitor and the museum's collections. The project followed a constructivist education theory approach, in which the visitor could construct its own meaning of the artwork and connect with it on a personal level. The visitor's tags would additionally add value for the museum by expanding their perspectives. Between 2006 and 2008, Steve.museum collected around 36.000 terms, of which approximately 86% were not previously present in the museum's documentation [CCH+06, HH10, TW06].

1.2 Games with a Purpose and ARTigo Project

There are multiple ways to motivate people to contribute to science. However, paying people even at very low rates to perform complex tasks would still be highly expensive. Moreover to trust in their sense of duty or joy to contribute to scientific accomplishments might also not suffice to motivate enough people to complete complex tasks in a limited amount of time. Because of this, Luis von Ahn came up with a different idea similar to his reCAPTCHA project - to have people make a contribution without them even knowing about it, simply by playing a game and therefore giving them an incentive to collaborate [Sav12, vA06, vAB05]. Games with a Purpose (GWAPs) combine entertainment with a

meaningful use of the human brainpower that goes beyond the capabilities of computers [vAB05, vAD08]. As stated by game designer Jane McGonigal on a TED Talk in 2010 three billion hours are being spent each week by people on the whole planet playing video games [McG11]. If even a small percentage of those hours could be invested into science, a huge amount of problems could be solved. Human-Computer-Interaction researchers have proven that having fun while using interfaces and integrating gamification elements is an important factor in designing applications besides the application's main functionality [Shn04, vAB05]. According to Malone [Mal81], major elements for enjoyable user interfaces are challenge, fantasy and curiosity. Challenge provides an uncertain outcome, but users might be able to vary their own degree of difficulty. Additionally there has to be feedback for the user to evaluate his process in achieving the goal. Fantasy offers the user to go on adventures, live out his emotions or personality traits or socialize with other people. Curiosity contributes to a level of complexity that is interesting and arousing but not too complex to comprehend [Mal81, Shn04]. In GWAPs, enjoyable elements are included by using features like scores, skill levels, highscores or timed response. Additionally, most of the games are played with a partner, who provides a feeling of connectedness and socializing. However, in the design of a GWAP it is not sufficient to simply add gamification elements to any scientific task. The task should be integrated directly into the game itself and should also be split into smaller fragments to limit the time commitment for the game and to run within the human attention span. The player should not be asked to perform the given task, the result should rather be a side effect of a fun game [vAD08].

One platform for Games with a Purpose is ARTigo. ARTigo has been developed by the research project play4science in 2008 with the purpose of collecting different types of tags for images of artwork and providing a search engine in the field of arts. In the last decades, the science of art like other sciences has gone through the process of digitization. As in literature, where the main focus is the digitization of books to an electronic readable form with the help of OCR methods, the goal in the science of art is to create digital representations of artwork. In digitized books it is possible to search for titles, authors, topics or even specific objects. However, a huge database of images of artwork might still include information about the artist, title or year of creation, but it would never be possible to search for images depicting a dog or images that contain the color red. To create a search engine in the area of arts it is therefore of utmost importance to obtain additional information on the image, called tags. Tags can be distinguished into surface tags, which describe what the player can see in the image, for example objects or deep semantic tags, which offer more abstract information like emotions the player connects with the image [BS16, Koh11, WBBL13].

The ARTigo platform contains more than 65,000 images of artwork from museums or collections, which were tagged with around 300,000 unique tags by July 2018. ARTigo offers different types of Games with a Purpose that work together and build the ARTigo Gaming Ecosystem. Description games build the basis of the ecosystem as they collect general tags for images without any restrictions. An example is ARTigo Game, where two players are presented with an image and have to enter tags the other player might also use to describe it. ARTigo Game is based on the idea of the ESP Game by Luis von Ahn, which will be presented in **chapter 2**. Building on the surface tags collected by description games, diversification games aim at gathering deep semantic tags. For example, ARTigo Taboo offers the same gameplay as ARTigo Game but prohibits the players from using surface tags by displaying tags, which the players are not allowed to use. Finally, integration games establish relations between tags from all types of games, for example with the game Combino. Combino provides the players with an image and all tags, that have previously been gathered. The players have to connect tags that the other player might have also combined, for example the tags *red* and *dress* if the image depicts a woman wearing a red dress [BS16].

The relations between ARTigo's game types are presented in **Figure 1.2**.

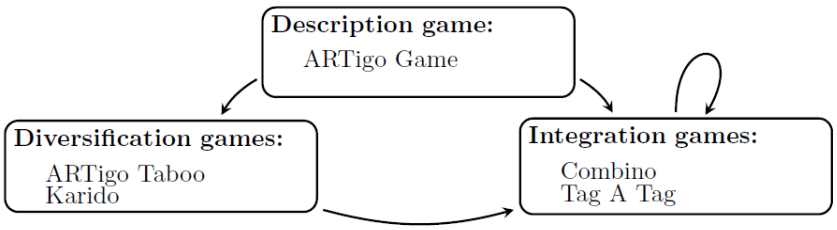

Figure 1.2: ARTigo Gaming Ecosystem [BS16]

1.3 Motivation and structure of thesis

While the current ARTigo version has already gathered a large amount of tags in its database, there is only little information about the links between the tags or their semantic classification. Therefore, the first goal of this thesis is to enhance the collected data by determining each tag's semantic category. For this purpose, a Semantic Categorization Pipeline is implemented in Python for the three languages German, English and French and the resulting categories are analyzed. This additional information could also be used to improve ARTigo's search function with semantic categories like *animals* or *shapes* and implement novel game ideas.

As ARTigo has been running since 2008 it has encountered some problems. As the current version of the ARTigo gaming platform is implemented with the Seam Framework[1], which has not been maintained since 2012, the platform either has to be upgraded to Java Server Faces 2 or reimplemented. Furthermore, the implementation of new games is cumbersome as knowledge about the Seam Framework decreases and its usage is unintuitive for most users. Additionally, the current ARTigo server faces stability and maintenance issues, which leads to recurring unavailability of the ARTigo platform. Finally, in the last decade new user devices, for example smartphones or tablets, have appeared. In 2017 around 63% of all visits in the United States originated from mobile devices whereas only 37% derived from desktop computers [Eng18]. Therefore, the second goal of this thesis will be to reimplement the three main games on the platform, ARTigo Game, ARTigo Taboo and Combino with state-of-the-art JavaScript software as basis for future game ideas. The new version should also allow easy and straightforward expandability to improve the process of adding future games to the platform.

In the current chapter, a general introduction to Human Computation and Games with a Purpose was given and the ARTigo project was introduced. Additionally motivation and scope of this thesis were defined. In **chapter 2** the related work on Games with a Purpose, Automatic Image Annotation and Semantic Categorization is outlined: After explaining the different types of Games with a Purpose, exemplary projects in the field of Image Tagging and other areas are examined closer. In contrast to gathering tags for images with a Human Computation approach, three projects on Automatic Image Annotation are layed out. To further analyze the collected tags, two concepts on the Semantic Categorization of tags and their challenges are presented. **Chapter 3** introduces the Semantic Categorization Pipeline and its resulting categories. Furthermore, the correctness of the categories as well

[1] http://seamframework.org/Seam3/Home.html

as sources of error for missing categorization results are examined. ARTigo's three main games ARTigo Game, ARTigo Taboo and Combino are presented in **chapter 4**. The concept of the reimplementation with JavaScript software, its software stack and additional features are explained in **chapter 5**. Finally, the thesis' conclusion is drawn and an outlook to future projects is given in **chapter 6**.

CHAPTER 2

Related Work

In this chapter, a closer look is paid at the types of Games with a Purpose and its most important projects. In the second part of this chapter, a brief overview of concepts of Automatic Image Annotation as an alternative to Human Computation is presented. Finally, different approaches to Semantic Categorization are explained.

2.1 Games with a Purpose

Games with a Purpose were first introduced by Luis von Ahn. He describes three types of two-player GWAPs, based on the method to verify the results: Output-Agreement Games, Inversion-Problem Games and Input-Agreement Games. In Output-Agreement Games the players are presented with the same input, for example an image. If both players independently come up with the same output, for example a describing tag, at some point in the game, the output is validated [vAD08]. **Figure 2.1** shows an example for Output-Agreement.

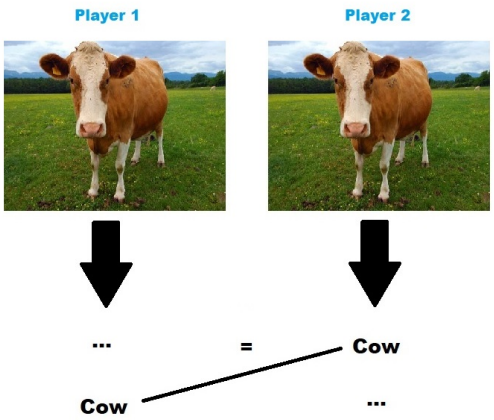

Figure 2.1: Output-Agreement (Own representation based on [vAD08], using [Bri])

While both players are given the same symmetric task in Output-Agreement Games, Inversion-Problem Games have at least two different asymmetric assignments. The first player is given an input, for example an object, for which he provides an output, for example a description of the object. If the second player can guess the original input based on the first player's output, the input is validated [vAD08]. **Figure 2.2** presents an example of an Inversion-Problem.

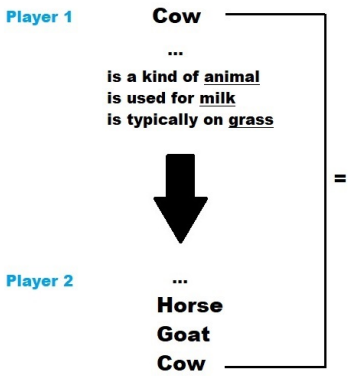

Figure 2.2: Inversion-Problem (Own representation based on [vAD08])

In Input-Agreement Games the players are presented with an input that might or might not be identical, for example an image. Both players have to find out if their input is the same by exchanging descriptions which is then used as output of the game if both players agree that their input is identical based on their descriptions [vAD08]. **Figure 2.3** shows an example for Input-Agreement.

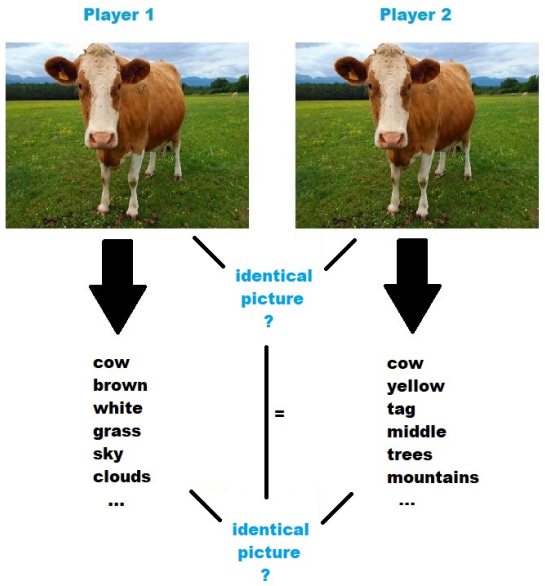

Figure 2.3: Input-Agreement (Own representation based on [vAD08], using [Bri])

In all of the above types of GWAPs, it is necessary to have two users playing together which ensures the validity of the results and the enjoyment of the game. However, it can not be guaranteed that there is an even number of players in the game at every time. Therefore, most of the GWAPs include pre-recorded game rounds that are replayed when a player cannot be matched. Since the actions of the recorded player are retaken including their exact times there is no way to distinguish the pre-recorded session from a real player. Additionally, there are also GWAPs that can be played by more than two players [vAB05].

Games with a Purpose do not only offer a fun incentive but also ensure the validity of the tags with the different game types. However, since all game types depend on the users, it is important to consider cheating attempts. In many GWAPs free communication between the two players is not allowed and words that are entered are restricted for letters, are spell-checked and looked up in a dictionary. Besides the pairing of two players is random and considers IP addresses, to ensure that players are not at the same place. Additionally results are only be persisted if they were achieved by a number of people that exceeds a given threshold [vAB05]. **Table 2.1** provides an overview of the GWAPs that will be examined further in the following subchapters.

Game	**Game Type**	**Output**
ESP Game	Output-Agreement	Tags of objects in images
Peekaboom	Inversion-Problem	Locations of objects in images
Verbosity	Inversion-Problem	Common-sense knowledge facts
TagATune	Input-Agreement	Tags of sound clips
Urbanopoly	Output-Agreement	Geographical data
Foldit	Other	Analysis and Reshaping of Proteins

Table 2.1: Games with a Purpose

2.1.1 ESP Game

Images are used by many applications on the web and almost every web page contains one or more images. However, statistics for example for Flickr show that not even half of its images are labeled properly [CZW13]. Labeled images are necessary for image search engines to find the correct pictures and filter out inappropriate content as well as for accessibility purposes like screen readers for blind or visually impaired persons. The ESP Game by Luis von Ahn aims to provide tags for most objects in an image. The goal for the players of the ESP game is to agree with their partner on words for each image by anticipating what their partner will type [vAB05].

Gameplay The ESP Game uses symmetric gameplay with Output-Agreement, where both players have the same task to accomplish. Both players have two and a half minutes to agree on as many images as possible. Each image is shown to both players, who can enter as many guesses for this image as possible. If both players have entered the same word they receive points and continue to the next image. Bonus points can be achieved with agreeing on fifteen images in a round. The gained points are displayed with a thermometer on each player's screen. To not only produce general terms but a variety of specific words and to increase challenge, taboo words are added to the images. Those are terms that the image has already been tagged with that the players are not allowed to guess anymore. When an image has received too many taboo words or has been passed on repeatedly by players it will be taken out of the game, as it has been labeled sufficiently [vAB05, vAD04]. **Figure 2.4** shows a player's screen during the ESP Game.

Purpose Labels are created when both players agree on a term for an image. Since the players are connected randomly and have no possibility of communicating outside of the game, they have to enter meaningful terms that describe the image in order to receive points. Additionally, a good label threshold X is used, to make sure that the labels are accurate for the image: to assign a label to an image, this label must have been agreed on by at least X player couples [vAB05].

Figure 2.4: The ESP Game [vAB05]

After two years, the ESP Game had collected over a million labels for around 300,000 images. For evaluation, a small sample of the tagged images was compared to manually tagged images and tagged labels of a probe of images were checked by humans. It was shown that the produced labels were in fact meaningful. Luis von Ahn anticipated that if the game was published on a popular gaming site and was played by five thousand people at each point of the day for six months all images on Google Images could be labeled with six words each [vAB05, vAD04].

2.1.2 Peekaboom

While the ESP Game provides information about which objects can be found in an image, Peekaboom aims to locate those objects in the image. Peekaboom uses tagged images from the ESP Game and further improves the collected data by adding location information [vAB05].

Gameplay Peekaboom is an Inversion-Problem Game, where the second player validates the first player's output by tracing back the first player's original input. The first player, called Boom, can see an image and a related word that is located somewhere in the image. The second player can only see a blank screen. Boom can reveal sections of the image to the second player, called Peek, allowing Peek to guess which word was shown to Boom. Boom also has the possibility to grade Peek's suggestions with *hot* or *cold* and ping a specific area of the revealed part of the image to distinguish this specific area from the rest of the object e.g. to point to the nose of a face instead of the whole face. Additionally,

Boom can also supply Peek with hint cards about the relation between the image and the wanted word, for example *Verb* or *Related Noun*. If Peek can name the object, both players score and switch their roles. The Usage of hint cards also adds points because hints help to assess how word and image relate to each other. Additionally, in bonus rounds that appear after successfully finishing four images in a normal game round, both players can see images and have to click on an object. The points the players receive for each image depend on how closely their clicks are located to each other, which requires the players to click in the objects center [vAB05, vALB06]. **Figure 2.5** shows the screens of both players for the word *cow*.

Purpose If Peek correctly guesses the word that Boom was supposed to reveal, it verifies the location of the object in the image. When overlaying revealed areas from multiple players for a specific object in an image, the object's boundaries can be found. The results also provide information about how the object relates to the image and how important the specific object is for this image. Furthermore, it can sort out bad image-word pairs and therefore increase the data's quality. The object's locations can then be used to fill databases to train computer algorithms or to improve image search results [vAB05, vALB06].

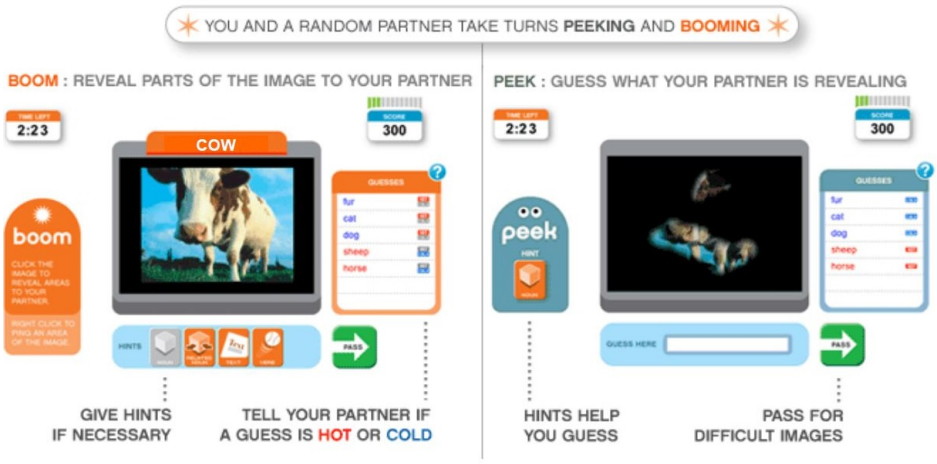

Figure 2.5: Peekaboom [vAB05]

2.1.3 Verbosity

Verbosity, another game by Luis von Ahn, does not aim at tagging images but at generating a database for common-sense facts. Facts like *grass is green* may be trivial for humans, but is necessary to determine for computers to improve in the field of AI. Verbosity is based on the party game Taboo that asks players to paraphrase terms without using taboo words [vAB05, vAKB06].

Gameplay Similar to Peekaboom, Verbosity also is an asymmetric Inversion-Problem Game. The first player, called Narrator, has to explain a given word to the second player, called Guesser. To explain the word, the Narrator has to fill sentence templates, for example *is a kind of* or *is used for*. If the Guesser concludes the correct word, both players get points. Similar to Peekaboom, the narrator can state if a guess is *hot* or *cold*. A game round of Verbosity takes six minutes and players switch roles after each word [vAB05]. **Figure 2.6** shows the narrator's screen for the word *eye*.

Purpose The facts that led the Guesser to the correct conclusion are stored in a knowledge database. By using sentence templates the type of information can be stored in categories like *spatial data* or *purpose* and natural language processing can be avoided [vAB05].

Figure 2.6: Verbosity [Wel08]

2.1.4 TagATune

TagATune is the first Input-Agreement Game and is not focused on images but on music. In 2009 Law and von Ahn [LvA09] attempted to use the ESP Game approach to tag sound clips, but soon realized that agreeing on tags for sound clips is far more difficult than for images and thus decreases the fun aspect of the game.

Gameplay TagATune's goal is for the players to decide whether the sound clips they were both listening to are identical or different. For this purpose, both players can hear a sound clip and tag it, for example with the genre of the music, the number of artists, their instruments or the provoked emotions. Both players can see the other player's tags. If the players agree correctly on whether or not they heard the same clip, they gain points. A major difference to the previous games is that the entered text is not restricted, since there is no other way to score than to describe the music, but postprocessing is applied to remove spelling errors or communication tags. Similar to Peekaboom there are bonus rounds in TagATune as well. In those rounds, the players observe three sound clips and have to agree which one differs most from the others.

Purpose Tags from TagATune are persisted whenever the players agree on identical or different clips. The evaluation showed that the most frequent tags use genre, the artist's instruments or descriptions of the music. Other frequent tags were of communicative function, like *same*, *diff* or negation tags like *no vocals* to state a difference when the partner was tagging *singing*. In the bonus rounds additional data on the similarity of music is gathered which could provide a lot of training data for music recommendation systems. Moreover, the data from bonus rounds can then be used to pair more similar sound clips in the regular rounds which could lead to more specific tags and a greater challenge.

Figure 2.7: TagATune [LvA09]

2.1.5 Urbanopoly

As tagging the world of art was for the longest time only done by curators, tagging or measuring the real world belonged to cartographers, geographers or city inspectors. The first approach to crowdsourced location tagging was done by the OpenStreetMap project in 2004 which made use of the emerging GPS technology and collected data about anything that can be depicted on a map worldwide. The goal of OpenStreetMap is to offer free access to geographic information for everyone by crowdsourcing the collection of the geo-data to volunteers with GPS devices [HW08, Deu]. To validate, correct and enhance the OpenStreetMap data, in 2012 an italian research group [CCC+12] created Urbanopoly, an app-based Game with a Purpose inspired by the board game Monopoly.

Gameplay Each player in Urbanopoly acts as a landlord and has the goal to buy venues. For this purpose the player has a starting budget and can receive a daily bonus as a reward for logging into the app regularly. Players can acquire venues if they are free, otherwise they have to hope for a good outcome in the wheel of fortune which could either be a monetary reward or a punishment, but also a chance to steal a venue from the current occupant. **Figure 2.8** shows multiple screens of the Urbanopoly app.

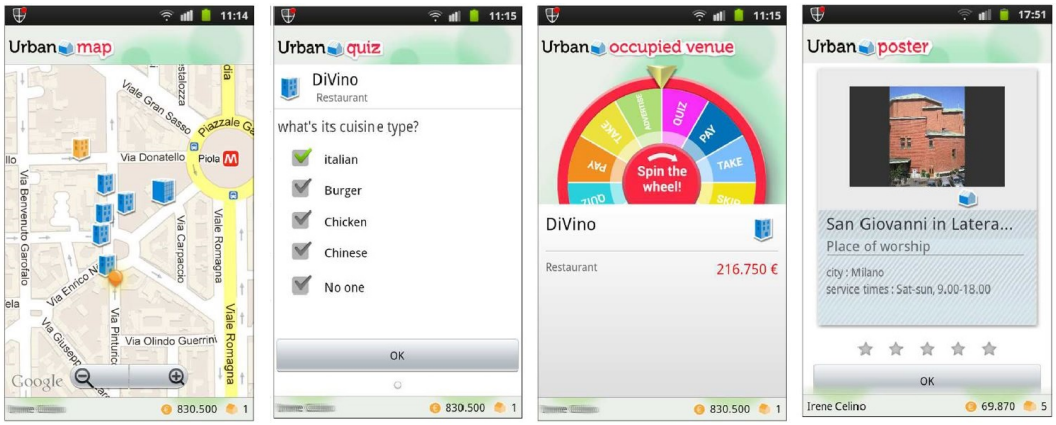

Figure 2.8: Urbanopoly [CCC+12]

Purpose The purpose of Urbanopoly is achieved through challenges on the venues, that the player would like to acquire. There are three types of challenges: Data Collection tasks, Data Validation tasks and Data Ranking tasks. For the collection of data, the player has to create an advertisement poster which will be rated by its current landlord and contains basic facts on the venue. Data validation can either be achieved by quizzes or ratings on the venue. Quizzes are also used for the ranking of data, the player could for example choose the best image of the building. Urbanopoly follows an Output-Agreement game type, where the gathered data is aggregated and outliers are removed.

2.1.6 Foldit

Foldit is an online game in the area of molecular biology, that was released in 2008. Players can rearrange a protein's structure with the goal to find their most compact form. Proteins are formed by chains of amino acids which then fold up into compressed forms to ensure its stability. The final form of a protein, for example as part of a disease, is very important for its examination and potentially finding a cure. While it is also a longsome option to calculate every possible solution with computing power, the human abilities of puzzle-solving and intuition are predetermined to achieve results in a shorter period of time [Sav12, Cur15, fol].

Gameplay Foldit presents the players with three-dimensional single proteins, as presented in **Figure 2.9**. When new players start the game, they have to practice on 32 tutorial proteins that show the most important concepts of protein folding, for example to clear clashes of acids, minimize the protein's energy or achieve the best distance between backbones to help non-chemists understand how to play the game. Points are distributed for each protein, the lower the energy the more points a player can receive. While Foldit has a competitive aspect by creating the protein with the lowest energy, it also supports collaborative work on proteins and allows communication between the players [Cur15, fol].

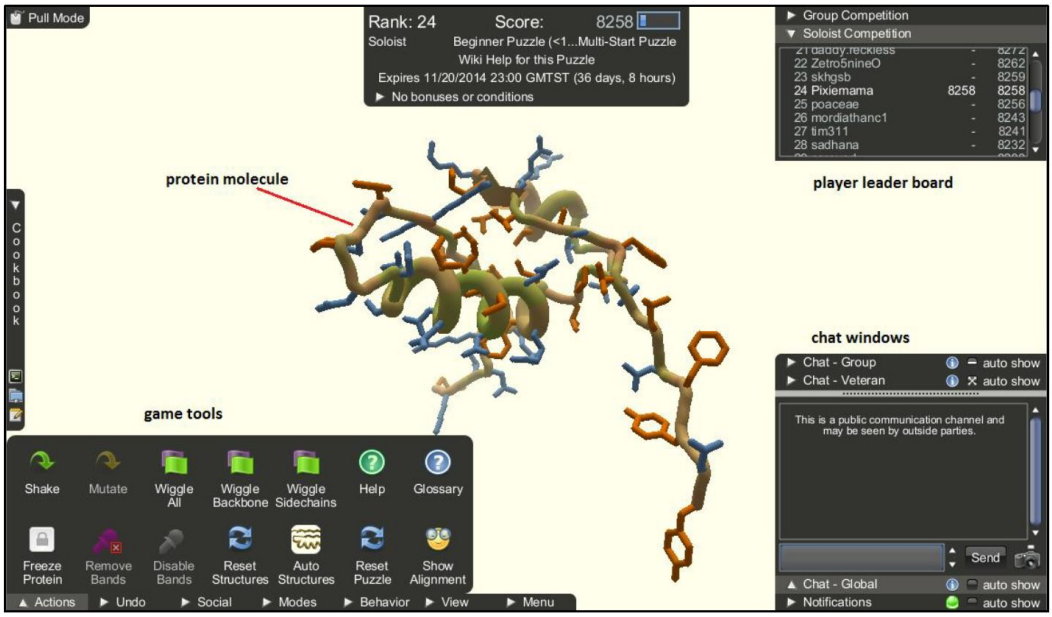

Figure 2.9: Foldit [Cur15]

Purpose Foldit solves two major challenges in the treatment of diseases. On the one hand it helps to analyze proteins and as a result find treatments for diseases. On the other hand it helps to create unknown proteins that can be used for cures as well. Some typical diseases in this field are the HIV virus that consists to a great part of proteins and reproduces through the creation of other proteins or cancer that can evolve because humans' own proteins are flawed [Cur15, fol].

2.2 Automatic Image Annotation

As shown in the previous section, there are many excellent approaches to tag images manually with methods of Human Computation. However, statistics of 2017 show that 90 millions of photos were uploaded each day to Instagram alone [Bag18] which again raises a demand for automatic tagging of images. Automatic Image Annotation can be classified into five general categories [CZF+18]. First, **generative model-based methods** aim at maximizing the generative likelihood of image features and label words from the training data. The training data can be based on tagged image data from Human Computation approaches. **Nearest neighbor model-based approaches** assume that similar images also share similar labels. Therefore, algorithms try to find the nearest neighbor images to the input image and then use their tags as label output for the input image. **Discriminative model-based concepts** trace image annotation back to a multi-label classification problem. To solve this problem multiple binary classifiers have to be trained for a set of training labels, which can then be applied to the input image. A novel approach in Machine Learning, the use of neural networks, which are based on the human brain, is employed in **deep learning-based methods**. Finally, **tag completion-based approaches** have a different focus. They assume that manually tagged training data is incomplete and concentrate on filling in missing tags and correcting the training data. In the following, the fundamentals of two exemplary state-of-the-art algorithms for Automatic Image Annotation will be introduced. TagProp is an example for generative model-based methods, whereas FastTag is based on a discriminative model-based concept. Additionally, a study by Saleh and Elgammal [SE16] which examines the best visual features and metrics especially for the prediction of an artwork's style, genre and artist will be presented.

2.2.1 TagProp

TagProp, which is short for Tag Propagation, uses a nearest neighbor approach. The model was created by Guillaumin et al in 2009 [GMVS09]. The classifier examines training data images similar to its given input image and maximizes the likelihood of the neighbor's image tags belonging to the input image as well. All neighbor images are weighted based on their distance to the input image. Additionally fifteen features like color histograms, which focuses on the distribution of colors or bag-of-visual-words, which tries to identify objects in the image, were applied. TagProp outperformed other approaches by far and reached precision values of 32% to 46% and recall values of 27% to 42% depending on the tested data set. However, its runtime of $O(n^2)$ makes it impractical for large data sets [CZW13, DVJ18].

2.2.2 FastTag

Another approach was presented in 2013 by Chen, Zheng and Weinberger [CZW13]. Their system FastTag yields comparable results to TagProp, but can be trained and applied in much less time. Additionally, it is also able to work with sparse training data, for example images with only a small number of tags. FastTag's approach is to connect two classifiers

with co-regularization. The first classifier predicts tags for the given image by analyzing its characteristics, while the second classifier tries to extend those results by finding additional tags that might be associated. All tags that can be employed by the first classifier result from the training tags that were available during the training phase with manually tagged image data. FastTag was evaluated using data from the ESP Game and two other data sets. It received very similar Precision, Recall and F1 values to TagProp, which surpasses other methods. However, TagProp's training time for sets with a higher number of images lies within hours, whereas FastTag's training finished within minutes. Nonetheless, results from TagProp and FastTag also show that at the current state of research in Automatic Image Annotation there is still a lot to achieve and Human Computation methods are the superior choice for now.

2.2.3 Classification of artwork

While TagProp and FastTag aim at annotating images by describing their content, other systems try to forecast an image's style, genre or artist. In 2016 Saleh and Elgammal [SE16] analyzed different visual features and metrics especially for artwork and used the best combinations to propose the system depicted in **Figure 2.10**.

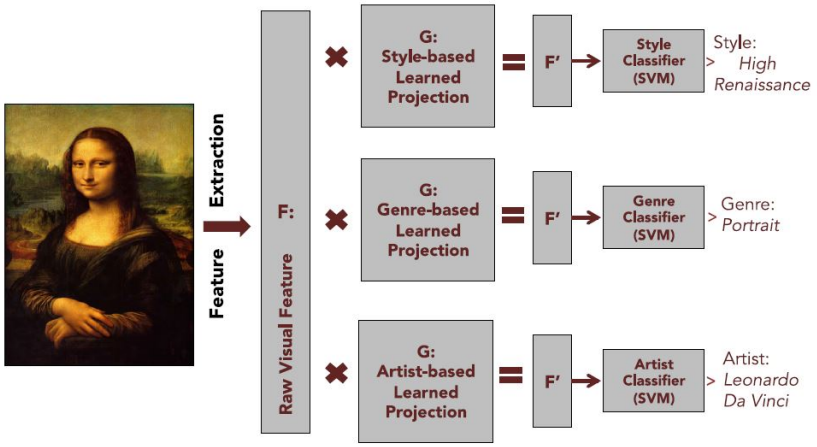

Figure 2.10: Classification of style, genre and artist of artwork [SE16]

Visual features can be divided into low-level and high-level features. Low-level features are general features that describe the gist, a viewer of the artwork recognizes, whereas high-level features refer for example to the objects in the image. Low-level gist features are based on perception and use dimensions like naturalness or roughness. The results showed that high-level features by the name of Classeme-features work best on all three classification categories. Saleh and Elgammal also tested different metrics to measure the similarity of paintings and concluded that a combination of different features together with using Large-Margin Nearest-Neighbor (LMNN) metric learning produced the best performance. LMNN tries to find the k same-class-nearest neighbors to the training image, while maximizing the distance to neighbors from a different class. Overall the results revealed that style classification is the most difficult task, because many of the style categories, for example *Minimalism* and *Color field paintings*, look very similar and are hard to distinguish. Additionally, high-level object recognition features reach its limits for more abstract styles like cubism, surrealism or expressionism.

2.2.4 Challenges in Automatic Image Annotation

However, the automatic annotation of images with computer vision algorithms is still an evolving field that does not yet reach the precision of Human Computation approaches. Most models use supervised Machine Learning algorithms, which rely on valid training data. However, the training data could have missing or irrelevant labels, it could only be labeled sparsely or could provide many common labels, which appear for multiple images. This imbalance of tags is a major problem for nearest neighbor algorithms, as labels which are used frequently have a higher probability to be chosen as output label than labels which are only used for a few images. Additionally, especially in the area of artwork, the differences in the styles of images are a challenge for Machine Learning algorithms. For example the tags *flower* and *red* were used in the ARTigo ecosystem to describe a picture of a cup with flowers, textitMaandkopje by an anonymous artist, the drawing of a flower, *Tulp Grand Roy de France* by artist Jan Vincentsz. van der Vinne and even a painting of a red vase with white flowers by Claude Monet, *Stilleben mit Chrysanthemen*. Those three very different images with the same tags are presented in **figure 2.11**. The other way around, similar images for example in the field of abstract art could lead to very different tags, according to the subjective perception of each individual observer [CZF+18].

Figure 2.11: Images tagged with *red* and *flower* [Images from ARTigo Game]

2.3 Semantic categorization

With the different approaches to image tagging huge amounts of tags have been and will be created. However, tags from different systems may describe different categories of information. While tags from the ESP Game focus more on the content that is displayed in the image, tags from Flickr aim at describing the event, time, location or type of camera that are connected to the image. Additionally, even when focusing on tags from the ESP Game there are multiple categories of tags that players employ, for example persons or objects displayed in the image, colors or painting techniques used to create the image, additional information on the creation of the image like its artist, date of origin or even emotions the players connect to the image. Therefore the Semantic Categorization of the player's tags will provide valuable information about the image and possible semantic groups and establish new potential for search engines. Furthermore, it is fundamentally important to help computers understand the meaning and content of documents and therefore automatically extract their semantic groups and categories [DEG+03, OSvZ09].

Currently, the main approach in Semantic Categorization relies on the use of knowledge bases, external databases with structured information about the input data. A major problem for any system in this field is the disambiguation of homonyms, for example the word *bank*, which could represent either a credit institute or a seating-accommodation. In the following subchapters two systems, SemTag and ClassTag, will be presented and their approach on disambiguation will be explained [OSvZ09, RAPM04].

2.3.1 SemTag

SemTag is a system in the field of unsupervised learning for automatic semantic tagging of large corpora, that follows a knowledge-based approach by including the TAP ontology. The TAP ontology offers information on objects from different categories like sports, music or authors including their taxonomy. With the help of the TAP ontology SemTag aims at equipping documents on the web with semantic annotations. Its algorithm works in three steps. First, the source documents are scanned for labels from the TAP ontology. All existing labels are persisted with their context words, which are the ten next words, that are written on each side of the label word. In the second step, a disambiguation algorithm called Taxonomy-Based Disambiguation (TBD) is applied to the whole document. For knowledge-based approaches there are two main sources of ambiguity: On the one hand, some labels might have different meanings in the ontology, like *Bush*, which could be linked to two presidents of the United States of America, a beer company or even a rock music band. On the other hand, some labels might exist in the ontology, however might not have been meant in this way. For example *Rihanna* could relate to the famous vocalist, but could also relate to any other female person by that name. To disambiguate both cases SemTag employs a similarity function including the label's context to calculate the similarity of each taxonomy node to the ambiguous label and accordingly the similarity of the taxonomy node to references outside the TAP ontology. Finally, the disambiguated labels are connected to their respective TAP object and persisted in the database including all associated metadata. As of 2003, SemTag had been used on millions of web pages and could provide a label bureau, which is an online server, that can be queried to request information about a specific label [DEG+03, UCI+06].

2.3.2 ClassTag

Another knowledge-based approach is employed by ClassTag, which uses structural patterns from the Wikipedia corpus and lexical data from WordNet to categorize Flickr tags [OSvZ09]. Wordnet is a lexical resource for the English language, which provides meaning and hypernyms of words, that can be used as category information. The ClassTag system consists of a classifier for Wikipedia articles, which is used by a pipeline to map Flickr tags to WordNet categories. An overview of the ClassTag system pipeline can be found in **Figure 2.12**.

To classify Wikipedia articles, ClassTag applies a binary Support Vector Machine (SVM) to each potential WordNet category. The features for the classifier are based on structural patterns of the Wikipedia articles. Each Wikipedia article is then classified by each SVM and receive the category with the highest confidence output value by the respective SVMs. In the second step, Flickr tags and Wikipedia articles are connected with each other. With the help of the classifier, the Wikipedia articles and therefore the Flickr tags can then be matched to the respective WordNet category. Since the classification outputs a confidence value for each category, ClassTag does not have to deal with a great amount of ambiguity. Additionally, outlier mappings were defined as less than 5% of mappings from an anchor

label to an article, which is removed before the classification takes place. A main advantage of ClassTag is its transferability to other languages as Wikipedia is present in many languages and translations with the help of Wikipedia's Interlanguage links could be applied [OSvZ09].

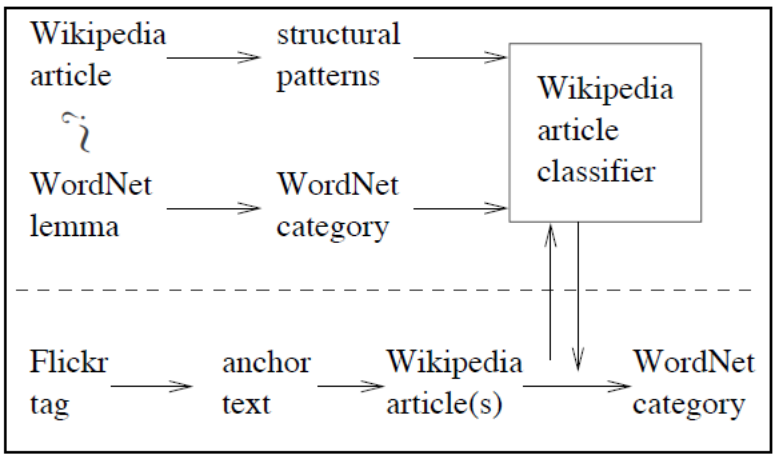

Figure 2.12: ClassTag [OSvZ09]

CHAPTER 3

Semantic categorization of tags

The previous version of ARTigo has already collected over 300,000 tags in the languages German, English and French for around 65,000 works of art as of July 2018 and has also created around 47,000 relations between tags with the game Combino. However, there is no information about the semantic classification of tags or which other tags might belong to the same category. The collection of this additional semantic information could also be used to find synonyms among the tags of each picture or to determine the correct meaning of homonyms. Furthermore, this additional semantic information offers a lot of input data for future game ideas, which will be discussed further in **chapter 6**. In this chapter, the design, implementation and an analysis of the results of the Semantic Categorization Pipeline are presented.

3.1 Semantic Categorization Pipeline

To determine the semantic categories of a tag it has to pass through a newly created Semantic Categorization Pipeline, which will be explained in the following subchapters. The pipeline is written in Python, which is predestined for the processing of natural languages, because of its popular and well-documented library Natural Language Toolkit (NLTK). This Semantic Categorization Pipeline is called every time a new tag is added to incorporate the categorization into future games or give players feedback on typing errors like *Did you mean ...?*. Additionally, to categorize and analyze all preexisting tags up to July 2018 a Onetime Load was created. The Onetime Load loops over all preexisting tags from table `tag`, receives each tag's categories from the Semantic Categorization Pipeline and stores semantic categories and their subcategories in their respective tables.

An overview of the proposed Semantic Categorization Pipeline is shown in **Figure 3.1**. At the beginning each tag is processed into a cleaned, singular and lemmatized form without punctuation characters. Additionally, multiple-word tags are split into a list of single words. Those steps are necessary to ensure that the subsequent tools are able to process the tag correctly. Second, the lexical category of the tag is retrieved. The subsequent steps of the pipeline are only executed if the tag is a noun, adjective, adverb or verb. Otherwise the Semantic Categorization Pipeline stops and returns an empty result list. Afterwards, named entities in the tag are determined by Named Entity Recognition algorithms and their pos-

sible categories *location*, *organisation* or *person* are retrieved. The main step of the Semantic Categorization Pipeline is the determination of the semantic relation. In this step the APIs of ConceptNet.io, Wikidata and WordNet are called to identify the tag's hypernyms via different relations and save them as categories. In the Postprocessing step invalid categories are rejected or shortened and duplicates are removed. This is necessary as the API output does not necessarily correspond to a valid category. Finally, if all of the previous steps did not result in categories, the tag might include a typing error. Therefore, autocorrection is used and the Semantic Categorization Pipeline is called once more with the corrected tag. **Table 3.1** gives a summary of all tools used in the Semantic Categorization Pipeline per pipeline step and language. All steps of the Semantic Categorization Pipeline and all employed tools will be presented in detail in the following subchapters.

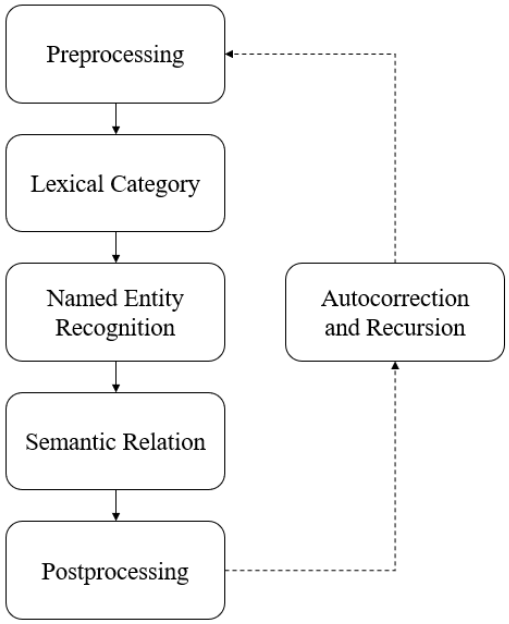

Figure 3.1: Semantic Categorization Pipeline

	German	**English**	**French**
Preprocessing Tokenization Lemmatization	 NLTK GermaLemma	 NLTK WordNetLemmatizer	 NLTK FrenchLefffLemmatizer
Lexical Category	Stanford PosTagger	NLTK PosTagger	Stanford PosTagger
Named Entity Recognition	Stanford NER	Stanford NER	Spacy NER
Semantic Relation	Wikidata ConceptNet.io	Wikidata ConceptNet.io WordNet	Wikidata ConceptNet.io
Postprocessing	Stanford PosTagger	NLTK PosTagger	Stanford PosTagger
AutoCorrection	PySpellChecker	PySpellChecker	PySpellChecker

Table 3.1: Usage of tools per step and language

3.1.1 Preprocessing

In the first layer of the Semantic Categorization Pipeline each tag is processed into a cleaned singular form of itself. This step is necessary since ARTigo games do not provide any form of input validation. However, some steps in the Semantic Categorization Pipeline require singular nouns and cannot process punctuation characters or multiple-word tags. Additionally, as the ARTigo games have a countdown the users are trying to type as fast as possible and are therefore submitting typing errors and additional characters, for example *woman#* as the number sign is close to the enter key. With the help of regular expressions, all punctuation characters are removed. Since numbers provide semantic information like the year of the creation of a work of art or even titles of artwork like *4 Schmetterlinge und 3 Raupen* (four butterflies and three caterpillars), a painting of Gabriel von Max which is displayed in **Figure 3.2**, they are not removed.

Figure 3.2: 4 Schmetterlinge und 3 Raupen [Image from ARTigo Game]

In the second step, the tag is tokenized with the help of Python's NLTK [BKL09]. Since there are no restrictions on the tag input for ARTigo players, it is also possible and reasonable to enter multiple-word tags, for example *Mona Lisa*. Nevertheless, steps like the identification of the lexical category or the Named Entity Recognition of a tag are only able to work with single word tags, whereas other steps in the Semantic Categorization Pipeline require the whole tag as a semantic unit. The tokenization of a tag results in a list of the single words the tag consists of, which takes into account language specifics and is more sophisticated than simply splitting the tag at each space character.

The last part of the preprocessing is the lemmatization of plural to singular for noun tags. This step requires the lexical category of a tag and is therefore occuring after the second layer of the Semantic Categorization Pipeline. However, thematically it belongs to the preprocessing step and will therefore be discussed in this subchapter. Lemmatization describes the conversion of a word into its basic form, which would also be stored in a dictionary. The lemmatization step is considered not necessary for verbs or adjectives, as those will mostly be entered in their basic form during the ARTigo games, whereas players often enter nouns in both singular and plural to gain more points. An example for the preprocessing would be the tag *gestures* which would be transformed into the cleaned tag *gesture*. Unfortunately, since the three languages German, English and French use very different forms of inflection there is no single tool that will lemmatize all three languages. Therefore, three separate tools are used to perform the lemmatization of a tag: WordNetLemmatizer (NLTK) [BKL09] for English tags, GermaLemma[1] for German tags and FrenchLefffLemmatizer[2] for French tags.

[1]https://github.com/WZBSocialScienceCenter/germalemma
[2]https://github.com/ClaudeCoulombe/FrenchLefffLemmatizer

3.1.2 Lexical category

The second layer of the Semantic Categorization Pipeline is the classification of lexical category or part of speech of each tag. The lexical category is required for the lemmatization of the tags and additionally allows the Semantic Categorization Pipeline to stop the evaluation if a tag is not a noun, verb, adjective or adverb and therefore does not possess a semantic category. The previous example tag *gestures* would be recognized as a *noun* and therefore allowed to be evaluated by the pipeline. For English words, the part-of-speech tagging (PosTagging) algorithm of NLTK [BKL09] and for German and French words the PosTaggers of Stanford University[3] are used to determine a tag's lexical category. As those tools only work with single words, the tokenized tag list is used to determine each single word's PosTag, for example present tense verb (VBP) for the English verb *discuss* or noun (NN) for the English noun *discussion*. The resulting PosTags is then converted into the five more general PosTags noun, verb, adjective, adverb or stopword. A stopword in this context is anything else than the other four categories and will not be evaluated in the following layers of the Semantic Categorization Pipeline.

For multiple-word tags an additional logic to determine the whole tag's lexical category was applied. If all single words of the tag have the same PosTag it is used as the whole tag's lexical category as well. If the tag includes at least one noun the whole tag's lexical category is noun as well, because the other words are only further describing the noun. For example the multiple-word tag *sleeping man* would consist of the PosTags verb and noun or the multiple-word tag *old man* would consist of the PosTags adjective and noun, however, the overall tag's lexical category would be noun in both cases. Any other combination of PosTags like for example verb and adjective or adjective and adverb do not make sense and should therefore not be analyzed further. For this reason any other combinations of PosTags are declared as stopwords.

3.1.3 Named Entity Recognition

APIs that produce semantic relations between tags are mostly based on resources like dictionaries or thesauri and therefore cannot handle proper names. However, since the name of the artist like *Leonardo da Vinci* or artwork like *Mona Lisa* or even recognizable places like *Italy* clearly belong to the semantic categories *name* and *place*, it is important to categorize named entities as well. For German and English tags the Named Entity Recognition (NER) tool of Stanford University[4] was used to produce the three categories *location/Ort*, *organization/Organisation* and *person/Person*. For French tags another NER tool called Spacy[5] provided the tags *lieu*, *organisation* and *personne*. Since NER algorithms require single words as well, the tokenized tag list was used to obtain the NER category of each single word in the tag. For multiple-word tags the NER category was only chosen if every word in the tag produced the same NER category. For example *New York* would be identified as a location as the NER algorithm would select *New* as the beginning of a location and *York* as the end of a location. However, the tag *New York skyline* would not be chosen as a location, as *skyline* does not refer to a named location.

[3] https://nlp.stanford.edu/software/tagger.shtml
[4] https://nlp.stanford.edu/software/CRF-NER.shtml
[5] https://spacy.io/usage/linguistic-features

3.1.4 Semantic relation

The main step of the Semantic Categorization Pipeline is the determination of each tag's semantic categories from semantic relations with the help of knowledge bases or graphs. The previous example tag *gestures* would in this case receive categories like *nonverbal communication* and *motion*. One knowledge graph that combines miscellaneous sources and offers its service in ten core languages including German, English and French is ConceptNet.io[6]. ConceptNet.io was considered the best alternative because it already combines its own data with it's competitors like WordNet, Wiktionary and DBPedia and offers many relations for a lot of languages. To determine the semantic category of a tag ConceptNet's API was called and the results of its Is-a-Relation were considered as categories. As ConceptNet uses many sources including free text input a filter on trusted sources like WordNet, Wiktionary, DBPedia and Open Mind Common Sense Initiative's Template 2 was applied. Unfortunately, the resulting categories are returned in lowercase, which provided some problems for the Postprocessing step, where among other things the PosTags of category words were used to determine the quality of a category. However since the PosTagging algorithms, especially for the German language, are to some extent based on case information all words in result categories from ConceptNet for the German language are capitalized, for example the result category *astronomisches objekt* (astronomical object) is capitalized to *Astronomisches Objekt*. The capitalization of all German words might lead to some mistakes in the PosTagging of adjectives, verbs or adverbs as well, but since there is no sensible way to determine the case without PosTagging and as it is more important to correctly identify existing nouns, those mistakes are condoned to escape from this vicious cycle. Furthermore, some categories for the English language were returned with and without a leading article like *a plant* and *plant*, which was cleaned up to one category without article.

The second knowledge base that was used in the Semantic Categorization Pipeline is Wikidata[7], which offers most of Wikimedia's data in a structured way. Results can be retrieved with SPARQL queries to Wikidata's webservice. The most relevant relations that connect entities in Wikidata are the Subclass-Of- and Instance-Of-Relations. The Subclass-Of-Relation describes a hierarchy of items and allows to find the superclass of a tag, which is equivalent to its category. For example, the Subclass-Of-Relation of *paint brush* would be *brush* or *writing implement*. However, there are many items of Wikidata that do not have a Subclass-Of-Relation, because they rather are entities of a specific category. For example, *Mona Lisa* or *New York* would be instances of the categories *human* or *city*, but are not a subclass of it. Therefore, since the Instance-Of-Relation still provides the category of those entities, it is also utilized. Unfortunately, Wikidata at times returns also some erroneous results like Q-Numbers, which are the identifiers of the items or Wikidata category pages, which are removed from the Semantic Categorization Pipeline results.

If neither ConceptNet nor Wikidata could provide any category results, there is another approach to determine the semantic category of a tag via its hypernyms. NLTK's tool WordNet [BKL09] provides the meaning of a tag, called synset and allows to retrieve a synset's hypernyms or superclasses. However, there are some limitations in the use of WordNet: On the one hand, WordNet is only available for English tags, which limits its usefulness for the Semantic Categorization Pipeline. On the other hand, it is very difficult to determine the correct categories from all of WordNet's data. For each tag there are multiple synsets and all of those synsets have a list of hypernyms, which have synsets and hypernyms themselves. For example, the tag *dog* owns among other things the synsets *dog* and *cad*. Synset *dog* connects, for example, to the hypernyms *canine* and *domestic animal*. In the hierarchy of

[6] http://conceptnet.io/

[7] https://www.wikidata.org/wiki/Wikidata:Main_Page

canine, the next hypernym is *carnivore*, then *placental* and after some intermediate steps the root hypernym *entity*. As most words somehow lead to the root hypernym *entity*, it is unfeasible to determine at which layer in the hierarchy the algorithm should stop. Depending on the accuracy of a tag it might make sense to look way up the hierarchy of hypernyms or to simply use the current layer of hypernyms. Because of those constraints, WordNet data is only queried if an English tag did not receive any category from Named Entity Recognition, ConceptNet or Wikidata and only the first synset and the first layer of hypernyms are considered as valid categories.

As ConceptNet.io, Wikidata and WordNet each have their disadvantages, combined they provide meaningful and valid category information. While ConceptNet.io and Wikidata offer their service for many languages, WordNet is limited to the English language. Indeed, even if ConceptNet.io offers its service for many languages the amount of results is still much higher for English than for German or French words. Additionally, Wikidata and WordNet's API are very stable, while ConceptNet.io indicated some stability issues and even a downtime for more than three days during the run of the Onetime load. A deeper analysis of the performance of the Semantic Categorization Pipeline also revealed that ConceptNet.io is taking up almost 60% of the overall time. Furthermore, ConceptNet.io's results in lowercase as well as the different output styles of their combined sources, for example the return categories *plant* and *a plant*, complicate the postprocessing. On the other hand, ConceptNet.io's API is very easy to query by simply adding the parameters to the request URL, while Wikidata uses SPARQL queries, which are not intuitive and presume a basic understanding of the SPARQL language. Although WordNet's usage is also very simple, its structure of synsets with a chain of hypernyms complicate the task to find the correct categories. Against this, ConceptNet.io and Wikidata's output in JSON format lightens the workload. The output quality of WordNet and ConceptNet.io - after filtering on trusted sources - is convincing. Wikidata, unfortunately, sometimes returns erroneous results like Q-Numbers or Wikidata category pages. Finally, a main advantage for ConceptNet against the other tools is the combination of many trusted sources like Wiktionary or DBPedia. An overview of all advantages and disadvantages is presented in **table 3.2**.

	ConceptNet.io	Wikidata	WordNet
Stability	-	+	+
Performance	-	+	+
Language offer	+	+	-
Language quality	-	+	+
Query simplicity	+	-	+
Output simplicity	+	+	-
Output quality	+	-	+
Combination of different sources	+	-	-

Table 3.2: Advantages and disadvantages of semantic relation tools

3.1.5 Postprocessing

All category results from Named Entity Recognition, ConceptNet and Wikidata or alternatively from WordNet are gathered in one result list, which has to pass through some Postprocessing steps before it is returned to the server and written into the database tables. The Postprocessing consists of two parts: First, invalid categories are rejected or shortened and second, duplicate categories from the shortening process or from different sources are removed. Since category data is retrieved from APIs with various sources, some result categories would still not be considered valid categories by a human, like *a relatively easy form*

of art or *usually*, which are two examples of results from the Is-a-Relation of the English tag *painting* by ConceptNet. Therefore, some simple rules were experimentally developed to filter for more relevant categories.

To select or shorten correct categories, PosTagging is applied to all words in the resulting category. If the category only consists of one word and this word is a noun then it is considered a correct category. However if the category consists of more than one word, it might still be a correct category. If the multiple-word category contains exactly one noun and no stopwords it might either be of the form adjective+noun like *nonverbal communication* or verb+noun like *painted image*. In this case the category is shortened and only the noun is saved as result category. For example the previous category, *nonverbal communication* which was found for the tag *gestures*, is shortened to the category *communication*. The original result category is however still persisted as subcategory of the shortened category for future usecases. Compound nouns like *oil painting* are also considered as valid nouns. For example, the category *antique oil painting* is processes as category *oil painting* with subcategory *antique oil painting*. While compound nouns in German and English are build by simply combining nouns, for French tags compound nouns might however include prepositions like *nom de famille* or *groupe de musique*. Therefore, another case was build that considered categories for French tags that include exactly two nouns and one stopword as correct as well. All other multiple-word categories are removed from the results list because they are mostly likely not category terms but rather phrases which are not considered valid.

3.1.6 Autocorrection and recursion

If no category was found in the Semantic Categorization Pipeline the tag might not have been a correct word. Therefore, in the case of an empty result list, autocorrection is applied and the most suitable correction is sent through the Semantic Categorization Pipeline recursively. After the Semantic Categorization Pipeline has been called recursively it will not autocorrect the already corrected tag again but return its results even if they are empty. For the autocorrection of all three languages the tool PySpellChecker[8] was used. A correction was calculcated with a Levenshtein distance of one, which means that exactly one letter in the word might be substituted, removed or added, for example the tag *trea* might be corrected to the word *tree*. If the autocorrection does not find a fitting word or the recursion does not produce a result list as well the Semantic Categorization Pipeline would return an empty list to the JavaScript Server and the results would not be persisted in the respective database tables.

3.1.7 Database model

All information about the semantic categorization are persisted in five new database tables. The original tag id from the existing table `tag` is connected to its cleaned tag id with the help of table `tag_cleaned_tag`. The cleaned tag consists of the cleaned name of the tag after the Preprocessing and its lexical category and language. It is stored in table `cleaned_tag`. Table `cleaned_tag_semantic_category` stores the connection between each cleaned tag and its respective semantic categories via their ids. The semantic categories itself can be found in table `semantic_category`, where semantic category ids are mapped to the name and language of each semantic category. To preserve the hierarchy between tags a third table `semantic_category_hierarchy` was added, which secures the relation between the ids of cleaned tag, semantic category and semantic subcategory. Those subcategories are not saved in `cleaned_tag_semantic_category`, as this table should only store the main

[8] https://github.com/barrust/pyspellchecker

categories of each tag. However, if the subcategories would be needed for future game ideas or evaluations, they could be easily joined to their respective tags. An example for a hierarchy between tags would be the tag *gestures*, which is categorized as *communication* with subcategory *nonverbal communication* or the tag *axe*, which is categorized as *tool* with the subcategories *heraldic tool* and *cutting tool*. An example for the table structure for the semantic categorization of tag *gestures* can be found in **Tables 3.3**, **3.4**, **3.5**, **3.6** and **3.7**.

id	tag_id	cleaned_tag_id
31419	2001511369	30642

Table 3.3: Content of table `tag_cleaned_tag` for tag *gestures*

id	cleaned_tag_name	lexical_category	language
30642	gesture	noun	en

Table 3.4: Content of table `cleaned_tag` for tag *gestures*

id	cleaned_tag_id	semantic_category_id
60834	30642	1886
60835	30642	16149
60836	30642	5121
60837	30642	3669
60838	30642	2011

Table 3.5: Content of table `cleaned_tag_semantic_category` for tag *gestures*

id	semantic_category_name	language
1886	communication	en
2011	motion	en
3669	indication	en
5121	action	en
9968	visual communiation	en
14708	human action	en
15291	nonverbal communication	en
16149	user interface	en

Table 3.6: Content of table `semantic_category` for tag *gestures*

id	cleaned_tag_id	semantic_category_id	semantic_sub_category_id
3846	30642	1886	15291
3847	30642	5121	14708
3848	30642	1886	9968

Table 3.7: Content of table `semantic_category_hierarchy` for tag *gestures*

3.1.8 Open questions and difficulties

The Semantic Categorization Pipeline provides fitting category results, however there are still some problems that could not be addressed in the scope of this thesis. While noun tags receive many result categories, there are significantly less results for tags of other parts of speech. Especially for German or French tags there are little to none results for adjectives or verbs from ConceptNet. Some results do not make any sense, for example the German adjective *klein* (little) has very strange results in the ConceptNet Is-a-Relation like many small letters like *a* or *b* or unreasonable categories like *käsepappel* (mallow) or *zittergras* (quaking-grass). Additionally, the amount and quality of results varies within the languages, which will also be further discussed in **chapter 3.2**. Since for parts of the Semantic Categorization Pipeline there are different tools for each language, the quality of the tool mainly influences the quality of the results. Furthermore, since ConceptNet and Wikidata are continuing to expand their knowledge, the Semantic Categorization Pipeline is highly dependend on their continued accuracy and the quality inspection of their source contributors. This is especially problematic for the identification of the PosTag since many subsequent steps are based on it. One main problem with the PosTagging is the lowercase output of ConceptNet for German tags, which was solved by capitalizing all results from ConceptNet for the German language. However, with this solution the PosTagger recognizes capitalized adjectives as nouns and considers German categories like *Ausdauernde Pflanze* (perennial plant) or *Gefährliche Pflanze* (dangerous plant) as compound nouns. In the Postprocessing steps those compound nouns are then considered as correct categories and are not shortened to *Pflanze* (plant) with its respective subcategories. For the Onetime load another problem occurs in the table `tag` itself. The first part of the tag table is persisted in upper case, which leads to a loss of information about the case and leads to inferior results of the PosTagging and therefore decreases the quality of the whole Semantic Categorization Pipeline.

Another main question is to decide which category is the best describing category of a tag. With the current version a tag can and should have many categories and the meaning of homonyms is not resolved. Filters were applied on the result categories in the Postprocessing step and a hierarchy of categories was implemented to select only valid categories. However this still results in a huge amount of semantic categories. A solution could be to implement Machine Learning algorithms and determine probabilistically how good each category is and how good it fits to its respective tag. Higher weights could be given to art specific category terms. However since most of the players are not art experts themselves, it might also make sense to keep collecting more general categories. Additionally, homonyms could be resolved by considering other tags that were inserted for the same work of art as context and disambiguating based on those tags.

The final difficulty is the performance of the Semantic Categorization Pipeline. As it calls APIs for ConceptNet and Wikidata and has to process a lot of data with Classifiers like for Named Entity Recognition or PosTagging, the Semantic Categorization Pipeline takes around thirty seconds for each tag that results in categories, which is too long for a real-time game. Tags which are stopwords only take ten seconds. A deeper analysis revealed that almost 60% of the time is consumed by the call to ConceptNet.io. In the current version, all newly added tags are categorized and persisted in the background and do not affect the performance of the game itself. However, the computation of the categories of over 300.000 preexisting tags took around three months on a single workstation. Additionally, for future game ideas that would use real-time category information, a duration of thirty seconds per tag would be too long. A solution to enhance performance would be to store the ConceptNet and Wikidata knowledge data locally to request its information without network overload.

3.2 Evaluation of the collected semantic categories

To evaluate the results of the Semantic Categorization Pipeline a Onetime Categorization Load was created which processed all existing tags in table `tag` over the course of around three months. As table `tag` includes all tags entered by any player at any time it is important to distinguish between all tags and validated tags. A tag is validated if it has been entered for the same resource by at least two players. At first, a general overview into the existing tag and gathered category data is given. Then, different samples of 50 tags per language are analyzed in detail. The first sample of 50 tags is used to evaluate the correctness of missing categorization and its sources of error for not-categorized tags. As many tags in the first sample are incorrect tags, the second sample is based only on validated tags. In a third sample a closer look is given to the correctness of the resulting categories. All tags and categories of the different samples are listed in the appendix.

3.2.1 Tag and category data

Up to July 2018, 317903 unique tags were collected in table `tag`. Out of these tags, 35% were collected before 2011 when the tags were still persisted in upper case while the remaining 65% were stored in the original case the player used. The language which is most often used is German, which covers 77% of tags, followed by English with 17% and French covering 6%. 28% of all tags are multiple-word tags. Out of all tags, 39136 (12%) are considered validated. Out of those validated tags, 71% were collected in upper case before 2011 and 29% in original case. The significant difference in the numbers of all tags compared to the number of validated collected tags arises from multiple factors. First, it is easier to find matching tags between two players when all tags are converted to upper case as this removes the factor case from the possible distinctions. Second, the best way to score points in ARTigo's games is to use surface tags, which describe objects visible in the artwork, because they do not require prior knowledge and show a high probability to be entered by both player and opponent. Therefore, it follows that the biggest share of validated tags is found in the first phase of the game as it gets harder to find new matching tags, which do not exist in the database yet. The distribution of tags into the different languages is similar to the distribution for all tags: 69% of all validated tags are German, 23% English and 8% French. 10% of all validated tags are multiple-word tags. This distribution of tags is shown in **Figure 3.5**.

16% of all tags and 44% of all validated tags were classified with semantic categories. 24204 semantic categories and 3486 category hierarchies were collected. The cleaned version of a tag was used to find categories - on average, one tag was assigned to one cleaned tag. However, it is also possible that more than one tag is assigned to a cleaned tag, for example for tags in singular and plural or tags with different case. A maximum of twelve tags was assigned to the cleaned tag *woman*. Those twelve tags varied in the use of grammatical number and punctuation like *women´*, *#woman* or *womens*. For validated tags, a maximum of three tags was assigned to the same French cleaned tag *musicien* (musician), which only varied in the use of grammatical number and grammatical case. Those respective tags are *MUSICIEN*, *MUSICIENS* and *MUSICIENNE*. Out of the collected semantic categories, two categories were assigned to one cleaned tag on average. The maximum number of categories per cleaned tag was found for the German cleaned tag *HUND* (dog) with a count of 48 categories. The ratio of collected categories per language showed that only 4% of all German tags received categories against 19% of all French and 18% of all English tags. Accordingly, for the 12616 categories collected for validated tags, the ratio of collected categories per language showed 25% of all validated German tags categorized as well as 50% of all English and 44% of all French tags. These results might suggest the categorization

working more comprehensively for English and French tags. Reasons for this behavior could be the high relevance of case for German tags which could lead to a misclassification especially for tags collected in upper case until 2011. However, the exact sources of error for samples for all three languages are evaluated in detail in the following subchapter. A closer look into the category hierarchies revealed maximum 50 hierarchies for one category. In this case, the French category *langues* (languages) was connected to different languages, for example *langues indo* (indo-european languages), *langues italo* (italo-romanic languages) or *langues brittoniques* (britonnic languages). On average, 2 hierarchies were stored per category and 7% of all categories were found in a hierarchy. 10% of all hierarchies were found for German tags, 62% for English and 28% for French tags. The coverage of categories is shown in **Figures 3.3** and **3.4**.

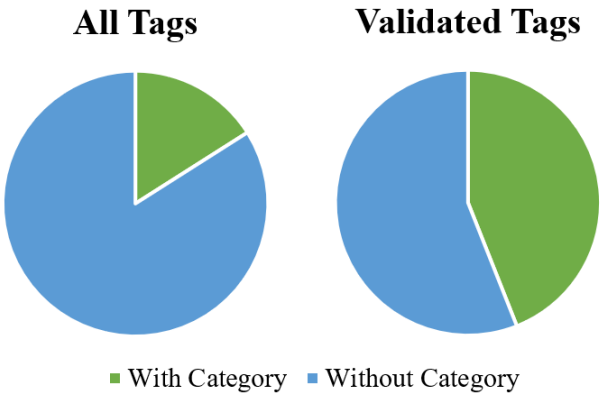

Figure 3.3: Coverage of categories for tags

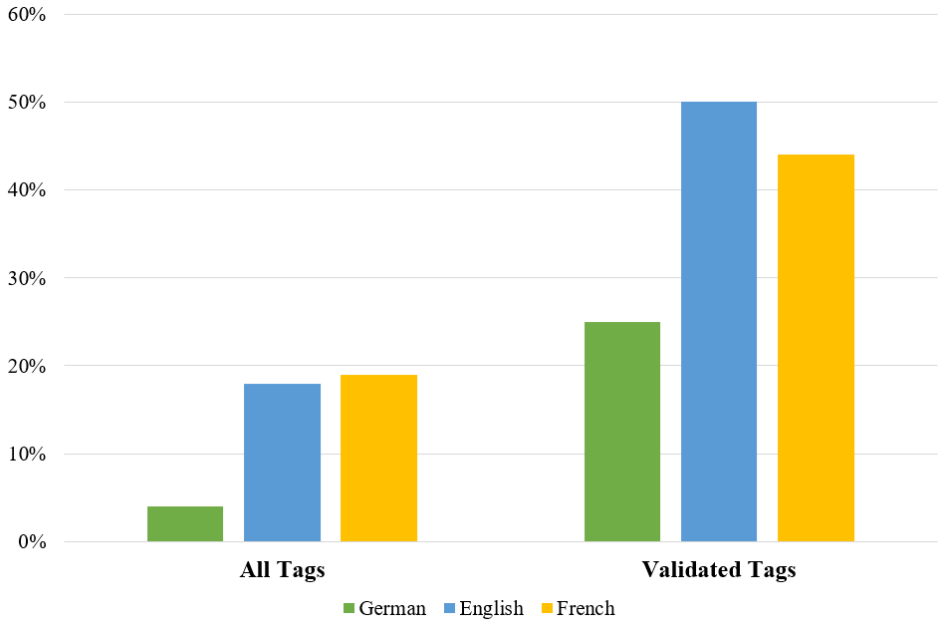

Figure 3.4: Percentage of tags with category per language

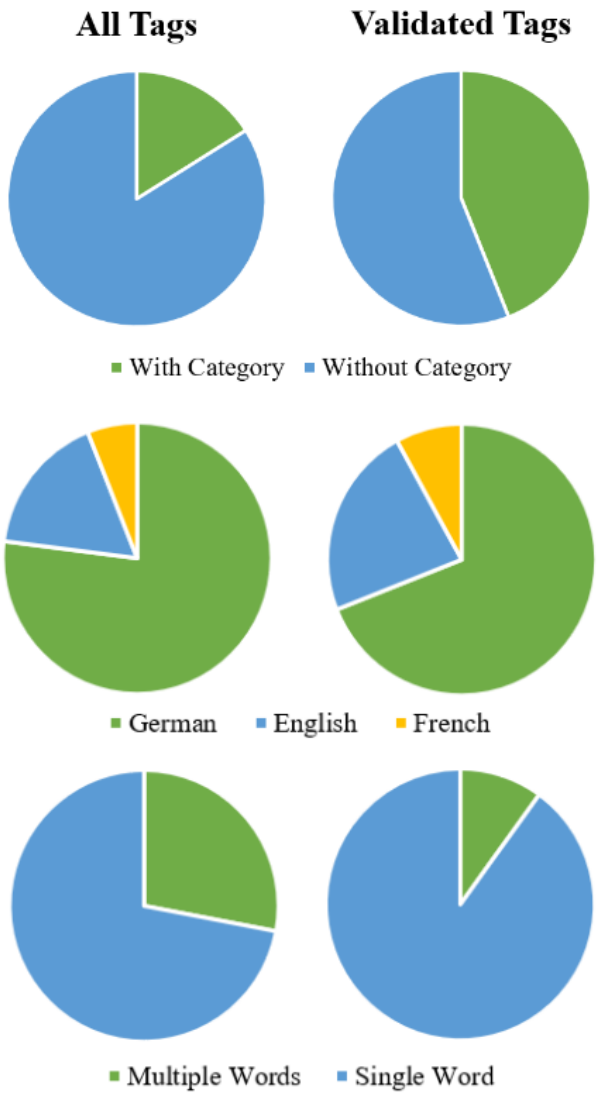

Figure 3.5: Distribution of tags

The collected data from the Semantic Categorization Pipeline also includes the lexical category of a cleaned tag. This lexical category can either be *noun*, *adjective*, *adverb* or *verb*. All other lexical categories are reassigned to the category *stopword* and excluded from the evaluation as they are most likely not a word which could be categorized. As expected, the most prominent lexical category out of all collected cleaned tags are nouns with 92% for German, 90% for English and 84% for French cleaned tags. While for German and French cleaned tags the second most used lexical category are adjectives with 6% and 9%, the second most used lexical category for English are verbs with 8%. The exact distribution of the lexical category per language is shown in **Figure 3.6**.

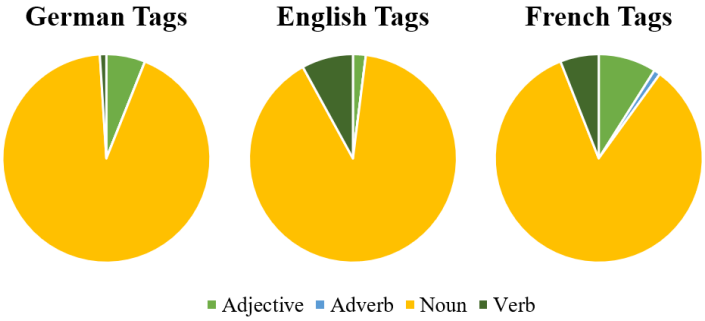

Figure 3.6: Distribution of lexical category per language

The top five collected categories regarding the frequency of their usage revealed the most-used types of words when tagging artwork in ARTigo's games. These categories are presented in **Table 3.8**. *Organization* is used as frequent category for all three languages, followed by *person*, *family name* and *location* which can be found for two out of the three languages. As *Organization*, *location* and *person* are categories awarded by Named Entity Recognition, they could be expected to be found in the top five as those three categories are predetermined while the API could return anything as category. For example for a fifteen-year old boy the APIs could return *young person*, *adolescent* or *teenager* while Named Entity Recognition has to use the category *person*. While the categories *person*, *family name* and *location* seem reasonable as players might describe the persons and locations they see in the artwork and perhaps know the name of the artist which is then categorized as *family name*. Furthermore, the English category *structure* and the French category *élément* provide more general universal categories for buildings or parts of buildings and specific elements the user could point out as archs, ornaments or patterns. Even the French categories *genre* and *profession* can be expected. The category *genre* includes tags for styles or tendencies which can be seen in the artwork like *art sacré* (sacred art), *féminin* (feminine) or *portrait* (portrait) and the category *profession* could be an generic term for what people in the artwork do. However, discovering the categories *organization* and *gene* in the top five categories seems odd. An examination of the tags which where assigned to the category *organization* reveals peculiar tags like *ADFJKL*, *STUBBS* or *Woman Red Pale*. In those sample tags the category *organization* was assigned by the Named Entity Recognition tools. A big part of the assigned tags were stored in upper case, which could be falsely recognized by the Named Entity Recognition tools as abbreviations for organizations. Future work could analyze furthermore if the Named Entity Recognition algorithm in case of the resulting tag *organization* should be substituted by alternative knowledge bases. Additionally, a closer look at the respective cleaned tag names of the English category *gene* shows many arbitrary combinations of letters, like *tu*, *ww* or *mn*, which were falsely identified as genes by Wikidata's API. The correctness of the categories' assignments is further evaluated with sample data in **Chapter 3.2.3**.

German	English	French
Person (person)	person	organisation (organization)
Organisation (organization)	organization	profession (occupation)
Familienname (family name)	gene	lieu (location)
Ort (location)	structure	élément (element)
Stadt (city)	family name	genre (genre)

Table 3.8: Top five categories per language

3.2.2 Sources of error for missing categorization results

As only 16% of all and 44% of all validated tags were classified with semantic categories, it is important to identify sources of error for the tags without categorization. To discover these sources of error, samples of 50 tags were analyzed manually per language for all tags and all validated tags. To eliminate bias, all relevant tag ids for the respective sample were identified and assigned with a continous row number. The row number has to be added as the ids of the selected tags are not continous and therefore a source range could not be retrieved. The range of continous row numbers was then transfered to a Random Number Generator[9] which selected 50 random numbers for each sample set. The corresponding ids of these numbers were then taken as sample for the evaluation. At first, the sample tags were examined for being correct or incorrect words. If a tag is not a correct word, like the misspelled tag *tre* instead of *tree*, it would be expected for the categorization not to deliver category results. Therefore, missing categorization for incorrect tags were considered appropriate. The remaining correct tags were further examined for their sources of error, if they were not assigned with a category, by processing them through the separate steps of the Semantic Categorization Pipeline and evaluating the results of each step.

For the first sample of 50 German tags, 36 were not considered correct because of the misuse of multiple words (20x) like *transparente bluse schwarz* (transparent black blouse), spelling errors (12x) like *HMMEL*, use of the wrong language (1x) like *GARDEN* or because they were not considered existing words at all (3x) like *ahin*. For the second sample of 50 **validated** German tags only 10 were not considered correct tags because of the incorrect use of multiple words (1x), not being an existing word at all (5x), spelling errors (1x) or use of the wrong language (3x).

Out of the remaining 14 German tags of the first sample which were considered correct, seven received categories. The remaining seven correct tags were not assigned with categories and were, therefore, analyzed further. Six out of seven tags did not use the correct case with five of them stored in complete upper case. In five out of those six tags with incorrect case, category results would have been returned if the correct case was provided to the Semantic Categorization Pipeline. Three of the seven correct tags were provided in plural, however none of these tags were converted to singular by the preprocessing module which could also be related to their incorrect case. For one of the three plural tags results would have been found if the tag would have been converted to singular. As this tag, *FILMSZENEN* (scenes in a movie), also belongs to the tags which would have provided results if the case was corrected, in this example two sources of error, plural and case, have been found. For the remaining two tags *grasüberwachsen* (overgrown with grass) and *FINGERZEIG* (cue) the APIs did not provide category results.

For the second sample of 50 **validated** German tags, 40 tags were considered correct tags out of which 20 received categories. Out of a total of 18 tags with incorrect case seven tags would have been assigned to categories if they were provided in the correct case. Three tags were stopped from evaluation because they were not recognized as nouns, adjectives, adverbs or verbs and therefore declared to be a stopword. Otherwise, they would have been categorized successfully. Two out of the three stopword tags were numbers, which showed that, unfortunately, numbers were excluded from the evaluation at all. In future work, numbers should also be allowed as evaluated lexical category and should not be rejected as stopwords. For the remaining ten tags the APIs did not provide categories. **Figure 3.7** shows the correctness of the missing categorization and the sources of error for German tags.

[9] https://www.random.org/integers/

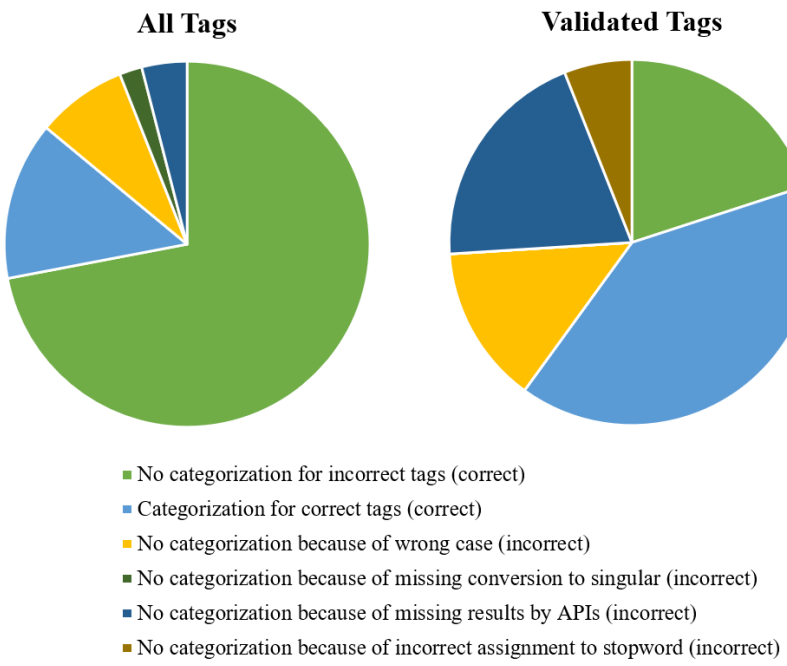

Figure 3.7: Correctness of missing categorization and sources of error for German tags

In the first sample of 50 tags selected of all English tags, 34 were not considered correct tags. As for German tags, the reasons are the misuse of multiple words (11x), spelling errors (16x), use of wrong language (3x) and not being an existing word (4x). The remaining 16 tags were considered correct and ten were assigned with categories. Four of the six uncategorized tags were stored in incorrect case. Out of those four, two tags would have been categorized if they were provided in correct case. Three of the remaining uncategorized tags, *FOGGY*, *augustanus* and *recumbent* did not receive results from the APIs. One tag, *mint*, should have been categorized, however, there might have been an error in the categorization at this point in time as no categories were stored for this tag.

For the second sample of 50 **validated** English tags, ten were not considered correct because of misuse of multiple words (4x), spelling errors (3x), use of wrong language (2x) and not being an existing word (1x). Out of the 40 correct tags 29 received category information while eleven were not categorized. Four out of six tags with incorrect case results would have been categorized if the correct case would have been provided. One tag was stored in plural and could not be converted. However, this tag would also not have been categorized if it would have been converted to singular. Three tags, all numbers, were stopped from evaluation because of their incorrect lexical category *stopword*. For the remaining four tags *BRUSHSTROKES*, *damaged*, *plateware* and *draconites* no categories were returned from the APIs. The correctness of the missing categorization and the sources of error for English tags are presented in **Figure 3.8**.

Finally, two samples were evaluated for French tags. In the sample of all tags only twelve were considered incorrect. The reasons for their incorrectness are again misuse of multiple words (3x), spelling errors (6x), use of wrong language (1x) and not being an existing word (2x). For the sample of **validated** French tags only four were not considered correct tags because of misuse of multiple words (1x) and spelling errors (3x).

For 38 correct tags out of all French tags, 14 received category information. The remaining 24 tags showed incorrect case only seven times. Only one of the tags with incorrect case would have returned categories if it was converted to its correct case. Four tags were provided in plural, only one of them was converted to singular by the pipeline. Two of the tags would have been categorized if the conversion to singular was successful. One number was stopped from evaluation because it was recognized as a stopword. 16 tags did not receive results from the APIs. Furthermore, in two cases the APIs did provide results, however those categories were rejected in the postprocessing as they were falsely not recognized as nouns. This logic is applied in the postprocessing to reject erroneous categories like verbs or adjectives as in general categories should be nouns. Two cases were not categorized because of a bug in the system. Those tags, *salle de conseil* (assembly hall) and *l'échelle de Jacob* (Jacob's ladder) included the preposition *de* which was falsely identified as language code for German as the Semantic Category Pipeline is called through a console script with the language code as one of the arguments and the tags as the remaining arguments. Therefore, those tags were also not categorized correctly. Out of the 46 correct **validated** tags 36 were categorized successfully. For the remaining ten tags five would have been categorized with correct case and one if it was provided in singular. A category for one tag was falsely removed in the postprocessing because of not being recognized as a noun. The remaining three tags did not receive category information from the APIs. **Figure 3.9** displays an overview of the correctness of missing categorization and the sources of error for French tags.

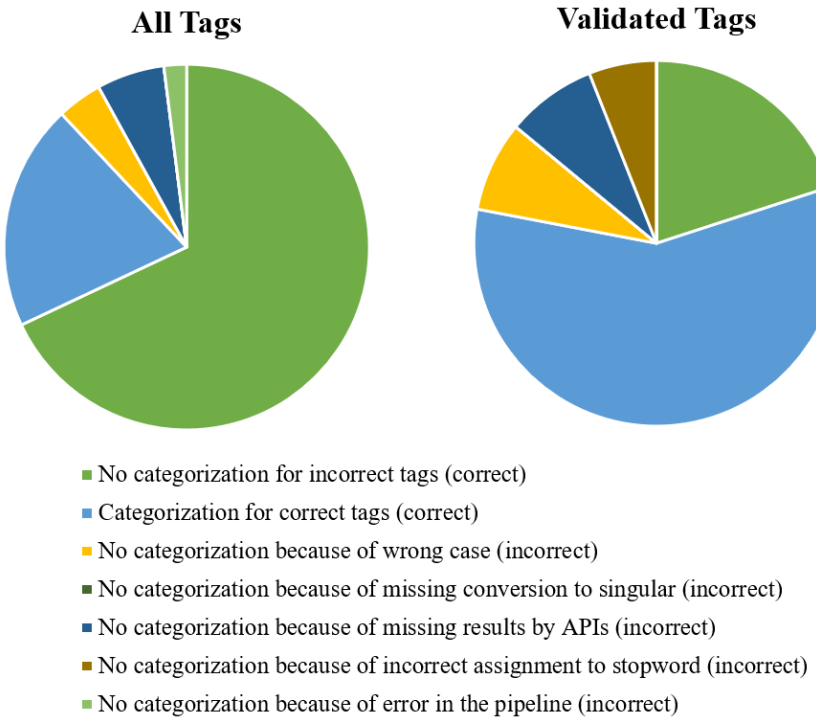

Figure 3.8: Correctness of missing categorization and sources of error for English tags

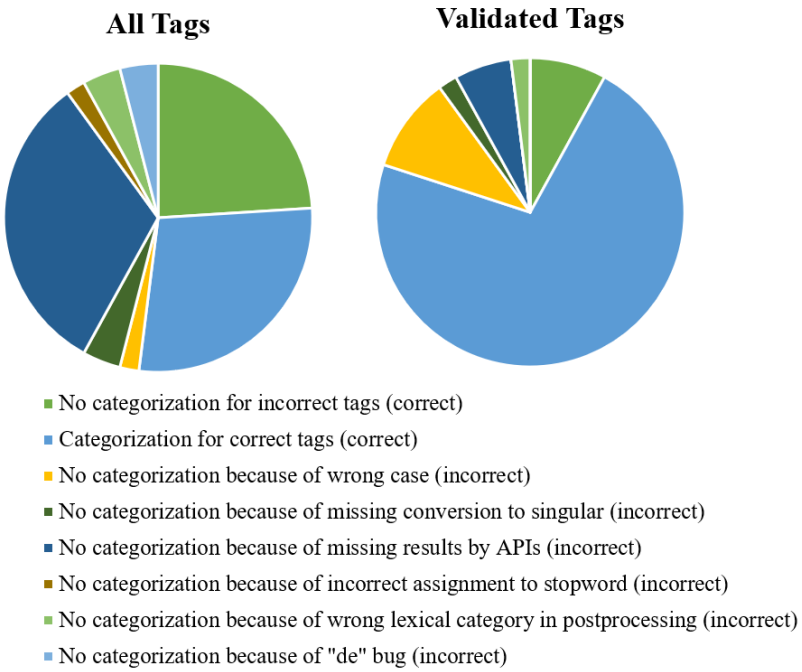

Figure 3.9: Correctness of missing categorization and sources of error for French tags

Overall, the evaluation of sources of error of the separate languages shows that a big percentage of tags does not represent valid words which could then be processed by the Semantic Categorization Pipeline. To provide tags with a higher quality a spell check algorithm with the possibility for the player to delete or correct a tag was added in the new implementation. With this option the amount of invalid words and spelling errors could be reduced. The high number of invalid multiple-word tags shows that the instructions of the game are not clear to all players. A solution for this problem could be to automatically split multiple-word tags if they are not recognized by the spell checking algorithm. Another possible solution could be to show an instruction screen with example tags before each new game session. This could additionally solve the issue of tags which are entered for the wrong language. The language of the instruction screen could get the player to notice that there is an option to switch the language.

The analysis of correct tags which were not categorized, however, reveals additional sources of error. Most frequently, the APIs did not return results for the given tag or the case of the tag was incorrect. The improvement to store the original case of the tag has already been implemented in 2011. Additionally, the spell checking algorithm should also improve the correctness of case for future tags. As the German language depends strongly on the correct use of case, this could also be an explanation of the categorization's weak performance for German tags. Two chances for improvement of the current Semantic Categorization Pipeline were identified by the evaluation. First, numbers should also be evaluated by the APIs and should not be rejected as stopwords. Currently, the preprocessing stops the evaluation for all tags which are not recognized as nouns, verbs, adjectives or adverbs. Numbers, which can also be recognized by the Part-of-Speech Taggers, could easily be added to this list. Second, the language codes should be transfered to the Semantic Categorization Pipeline at a fixed position in the list of arguments of the console call. With

this change, French tags including the prepositions *de* and *en* would not be misclassified as German or English tags. The results also show differences in the performance of the lemmatizers and Part-of-Speech Taggers between the languages. Especially the conversion to singular and the falsely rejected nouns in the postprocessing for French tags show the dependency of the results to external tools.

3.2.3 Evaluation of categorization data

To evaluate the correctness of the collected category data, a sample of 50 cleaned tags per language was examined. Similar to the previous samples, the respective ids were retrieved from the database, combined with row numbers and 50 cleaned tags were selected with the help of the Random Number Generator. The 50 cleaned tags resulted in 124 German, 121 English and 122 French categories.

Out of the 124 German categories, seven were assigned the category *Person* (person) for an incorrect multiple-word tag, for example *Frau Porträt* (woman portrait). While in some cases - as in the previous example - one of the words could be related to a person, for most of the incorrect tags the category did not make sense, for example for tag *GEDREHTE SÄULEN* (rotated columns). For eight cleaned tags, categories were assigned which related to the cleaned tag but could not be considered valid categories, for example the cleaned tag *Landung* (landing) which received the categories *Luftfahrt* (aviation) and *Militär* (military). A more appropriate category for a landing would be for example *maneuver in flight*, however, it is correct that landings take place in the field of aviation and military. Out of the remaining 109 categories, 85 were regarded appropriate, for example the categories *Kaltblüter* (cold-blooded animal), *Lebewesen* (creature), *Tier* (animal) and *Wirbeltier* (vertebrate) which were correctly assigned to the cleaned tag *AMPHIBIE* (amphibian). The remaining 24 categories were considered incorrect, for example the category *Esoterik* (esotericism) for the cleaned tag *Schwarz* (black) or the category *Gedicht* (poem) for the cleaned tag *Segler* (sailor). **Figure 3.10** displays an overview of the results of the German categorization.

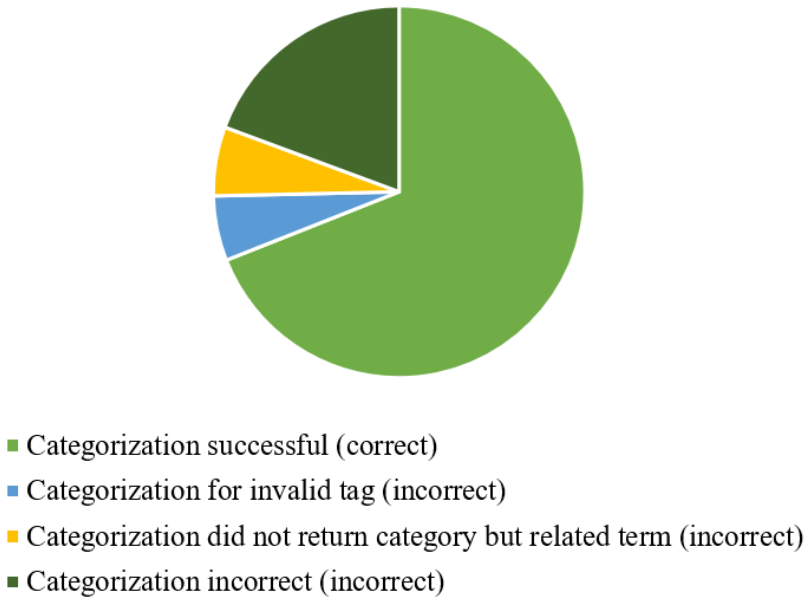

Figure 3.10: Results of semantic categorization for German tags

Out of the 50 cleaned tags 43 were assigned the correct lexical category. The remaining seven cleaned tags received the lexical category *noun* for verbs and adjectives, for example *Schwarz* (black) or *SCHÜTTEN* (pour), or the lexical category *adjective* for nouns with lowercased first letter like *klimazone* (climate zone).

The sample of 121 English categories provided three categories for tags which did not represent a valid word, for example the category *approach* which was rewarded to the tag *facees*. Eight categories were related to the original tag but could not be considered correct categories like category *hearing* for tag *ear*. In two cases the category was identical to the cleaned tag name which should be disallowed in future applications of the Semantic Categorization Pipeline as the category of a tag can never be the tag itself. For example the tag *rocket* should not receive the category *rocket* but rather *missile* or *spacecraft*. 86 categories were considered correct, for example the categories *organ*, *structure*, *sense organ* and *plant part* for the cleaned tag *ear*. The remaining 22 categories were considered incorrect, for example the categories *attention*, *fruit*, *gene* or *inflorescence* for the cleaned tag *ear*. These results are displayed in **Figure 3.11**.

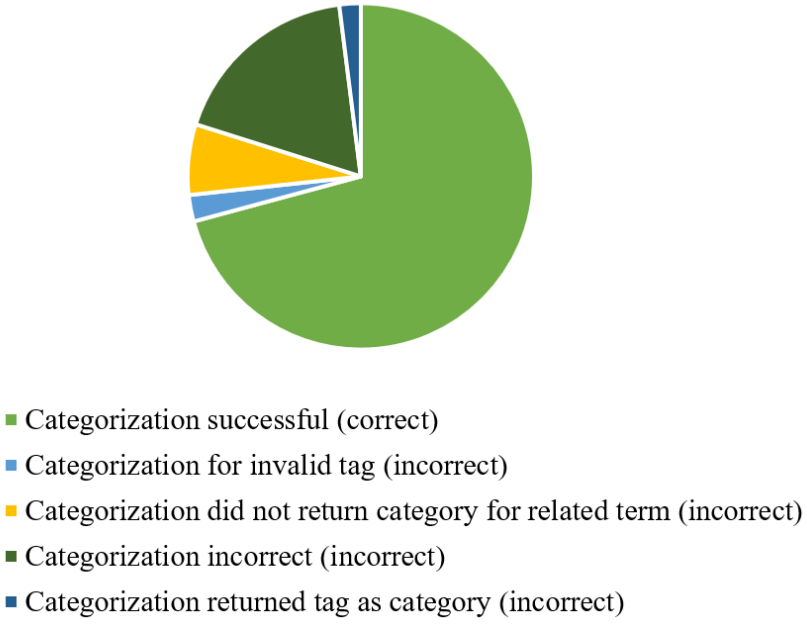

Figure 3.11: Results of semantic categorization for English tags

44 of the cleaned tags obtained the correct lexical category. The remaining six cleaned tags received the lexical category *noun* for verbs and adjectives like *engage* or the categories *verb* and *adverb* for nouns like *gondoliere*.

Finally, out of 122 French categories, four were not considered a valid word but received category *organisation* (organization), for example the cleaned tag *PROUDHON*. In three cases the category was identical to the cleaned tag name, for example for tag *symbole* (symbol). 86 categories were regarded suitable, for example the categories *discipline* (discipline) and *sciences* (science) for the cleaned tag *géographie* (geography). The remaining 29 categories were not considered appropriate, for example the category *grade universitaire* for the cleaned tag *géographie*. These results are displayed in **Figure 3.12**.

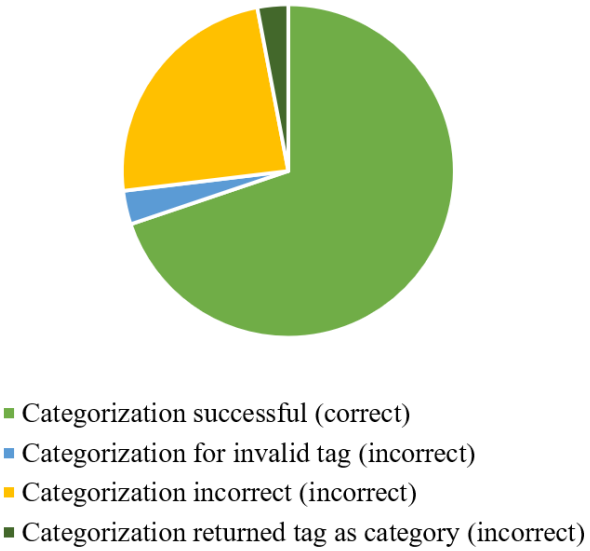

Figure 3.12: Results of semantic categorization for French tags

Out of the 50 cleaned tags 45 received the correct lexical category. The remaining five cleaned tags assigned lexical category *adjective* to numbers, for example *1663*, or categories *verb*, *adjective*, and *adverb* to nouns.

The evaluation has identified the most important sources of errors for the Semantic Categorization Pipeline and has analyzed if the resulting categories can be considered appropriate for the cleaned tags by examining random samples. The correctness of the categorization of 68% for German, 71% for English and 70% for French tags in this sample can be considered as a great start for the Semantic Categorization Pipeline. Additionally, future work could analyze further if the resulting categories are complete. For this evaluation a study could be designed which asks participants to categorized a sample of cleaned tags which were also categorized by the Semantic Categorization Pipeline. The collected study data could then be used to identify which categories were not provided by the pipeline and in which areas the pipeline could be enhanced with additional knowledge bases or computational linguistic approaches. One improvement for future applications should be to disallow the identical tag returned as category. A closer look into those categories revealed Wikidata's API as their source. While it seems strange that identical words are related to themselves via the Subclass-of- or Instance-of-relations those errors can easily be rejected.

CHAPTER 4

ARTigo ESP Games

The second goal of this thesis is the reimplementation of the three main games ARTigo Game, ARTigo Taboo and Combino, which all follow the model of the ESP Game. All three games are Output-Agreement Games, however, they differ in the collected types of tags. While ARTigo Game aims at producing more general surface tags, which can be used as input for the other two games, ARTigo Taboo gathers deep semantic tags. Finally, Combino creates relations between the tags accumulated by ARTigo Game and ARTigo Taboo. In this chapter, the game logic of the three games is presented in detail.

4.1 ARTigo Game

ARTigo Game, the first game of the ARTigo Gaming Ecosystem, presents two players with the same image of a randomly selected artwork. Both players enter tags they associate with the image. If both players enter the same tag in a game round they are rewarded with 25 points. If a player enters a tag that was previously persisted in the database in another game round he still receives five points. Tags that are neither entered by the opponent nor exist in the database do not give points to the players. A game round of ARTigo Game with the artwork "Gezicht op kasteel Purmerstein" by artist Abraham de (II) Haen is presented in **Figure 4.1**. There is a main difference to the ESP Game: In the ESP Game the players continue to the next image after their first matching tag and have two and a half minutes to describe as many images as possible. Players of ARTigo Game, however, have one minute per image in which their goal is to match as many tags as possible and each game session is finished after they have dealt with five images. As most ARTigo players have at least a basic interest in arts they enjoy to examine each artwork closely instead of switching to the next image after their first matching tag. Additionally, at the end of each game session the five images and their respective information are presented to the players, as displayed in **Figure 4.2**. Since the player's best strategy is to enter tags that their opponent might also enter ARTigo Game mostly provides surface tags, which describe what the player can see in the image, for example displayed objects or persons. [BS16, WBBL13].

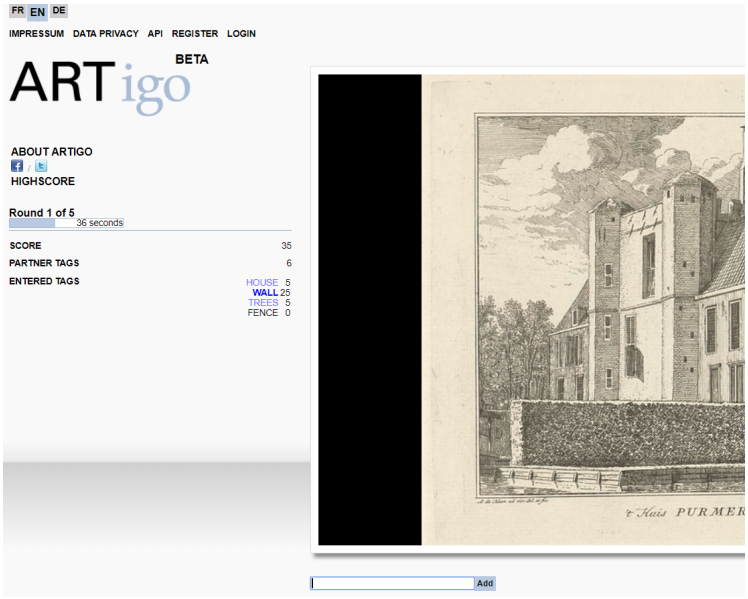

Figure 4.1: ARTigo Game (Screenshot from [Prob], significant parts only)

Figure 4.2: Information about images at the end of a game round (Screenshot from [Prob], significant parts only)

4.2 ARTigo Taboo

ARTigo Taboo has a similar game logic to ARTigo Game. As the goal of ARTigo Taboo is to collect deep semantic tags, taboo tags were added to oblige players to enter new terms. Taboo tags are the seven most frequently entered tags for each artwork and every language that have been stored at least two times. In each game round, the image and its taboo tags are displayed. A player can only enter tags that are not contained in the list of Taboo tags. Apart from the usage of Taboo tags, its gameplay follows ARTigo Game [BS16, WBBL13]. A game round of ARTigo Taboo presenting the artwork "Schottische Küstenlandschaft" by artist Hans Gude is displayed in **Figure 4.3**.

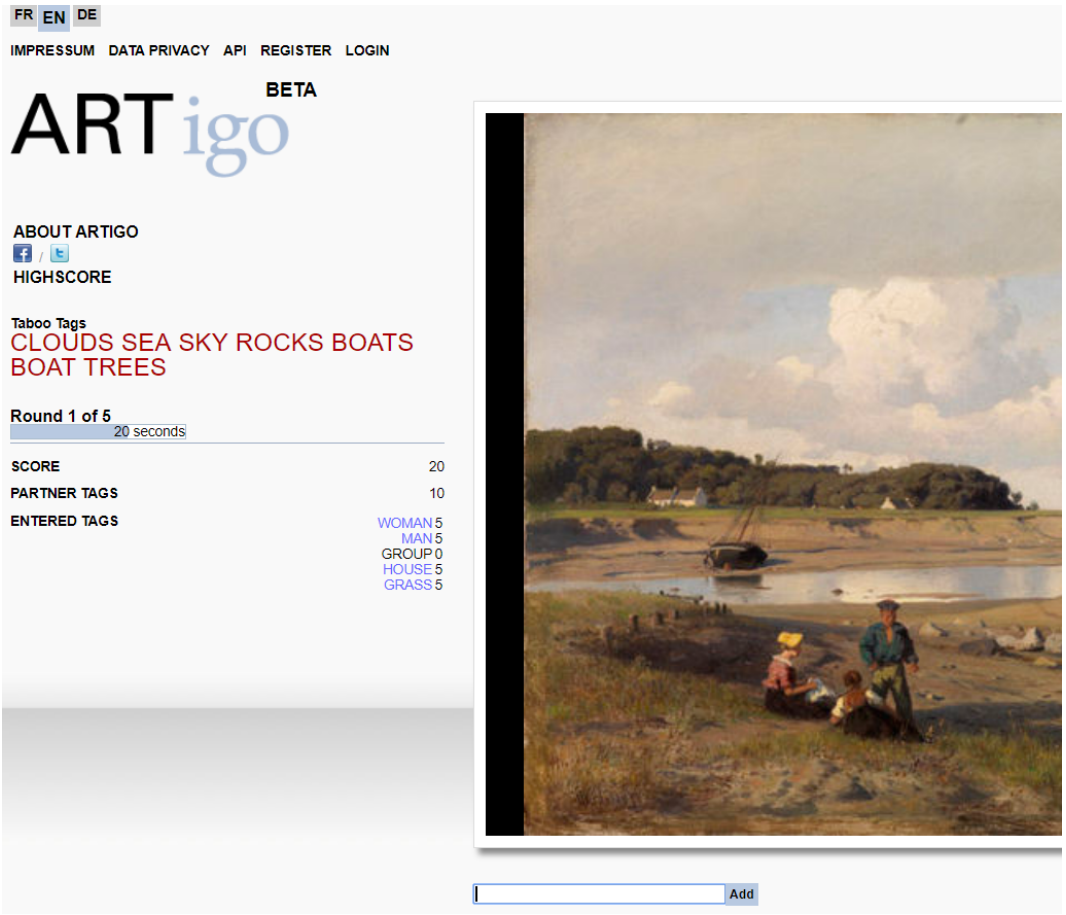

Figure 4.3: ARTigo Taboo (Screenshot from [Proa], significant parts only)

4.3 Combino

While ARTigo Game and ARTigo Taboo collect mostly single tags, like *pale*, *red*, *skin*, *vase* or *woman*, Combino provides information about how the tags relate to each other. In a game round of Combino the players are presented with an image of the artwork and fifteen tags it was previously tagged with. The players have to combine two tags by clicking on them. A possible combination for the above example could be *red+vase* or *pale+skin*. The combined tags are rewarded similar to ARTigo Game. However, as the task of combining tags is more complex than entering tags, a game round of Combino takes 90 seconds. A player is able to

request an additional set of fifteen tags as long as this amount of tags is available. As input tags for Combino can also be already combined tags, it is possible to connect more than two tags, for example *nude+woman+bathing* [BS16, WBBL13]. A player's screen of a game round of Combino with the painting "The Tub" by artist Edgar Degas shown in **Figure 4.4**.

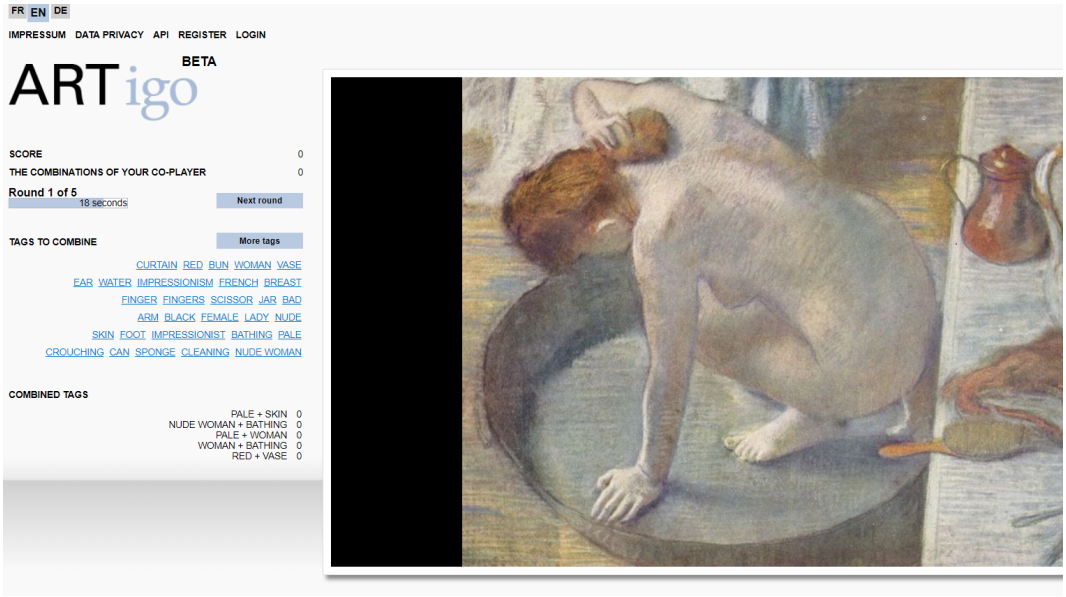

Figure 4.4: Combino (Screenshot from [Proc], significant parts only)

4.4 Bot opponent

The three above games can only be played with exactly two players. However, between 2008 and 2014 around 150 persons have played ARTigo each day [BSS15] which leads to an average of six players per hour over all of ARTigo's games. Therefore, as a new player might have to wait a long time to be matched with an opponent with the same language and game type ARTigo uses a bot opponent. In the beginning of each game round the bot opponent fetches fifty randomly chosen tags for the given resource and language from the database for ARTigo Game and ARTigo Taboo. For Combino the bot fetches only fifteen tags as it takes a longer time to combine tags than to enter them and therefore a smaller amount of tags is sufficient. At first, an initial delay is given to simulate the bot opponent examining the image or reading taboo or to be combined tags. In the following, all tags are played after some delay until the game round is over. All delays can vary in a given margin to disguise a bot playing as opponent. **Table 4.1** shows the amount of tags and the margin of delay for the respective game types.

Game Type	Amount of Tags	Initial delay (in seconds)	Tag delay (in seconds)
ARTigo Game	50	2 - 4	1,5 - 4,5
ARTigo Taboo	50	4 - 7	1,5 - 4,5
Combino	15	1,5 - 4,5	2,5 - 6,5

Table 4.1: Bot settings per game type

4.5 Shortcomings of the current implementation

As stated in the previous section ARTigo's games are played on average by around six persons per hour. Therefore, in the current version of ARTigo it is not possible to be matched against a real opponent. Additionally, players might notice the regularity of the bot opponent's moves inspite of the delay margin. The current bot logic could be improved by replaying game rounds by real players - as implemented by the ESP Game - once the system has collected sufficient game round data. Additionally, if by chance two players connect at the same time, they should have the chance to play against each other and not be matched to their bot opponents.

Another improvement could be to allow subsequent scoring for registered users. In the current version players can only score points for tags which are already present in the database and can therefore be used by the bot opponent. However, when future players enter tags the previous player has entered as well they score 25 points while the previous player, which has entered the tag first, does not receive subsequent points. As the score data is consulted for the monthly highscore it makes a difference for some registered users and could even be an incentive to create an account. Therefore, a new feature could be implemented to update the scores of the user which has entered the matched tag in the previous game round.

Furthermore, there are some issues with the storage of the tag input in the database. In the beginning of ARTigo all tags were persisted in the database in upper case. As this lead to a loss of information, for example to distinguish between the german words for the body part arm (*Arm*) and poor (*arm*), since 2011 the tags are persisted in the originally entered case. Additionally, in the current version the tag input is not checked or validated. As the players have to input the tags under time pressure there are many tags with typing errors or extra characters, for example *smakl* or *small#* instead of *small*. While the players can enter the tag in the corrected version additionally as well, it would be great if the system could automatically spell check and correct the entered word. Alternatively, the system could allow users to correct tags after the input to remove their spelling errors. Finally, the system also offers the possibility to enter multiple-word tags. This feature is important for composed words like *Mona Lisa* or *New York*, however, some players use this to enter many non-related tags at once, for which they will not receive any points. Therefore, an improvement could be to notify the user if he has entered a multiple word tag and give him the opportunity to split those words into single tags.

CHAPTER 5

Implementation

The second goal of this thesis is the reimplementation of ARTigo's three main ESP games with state-of-the-art JavaScript software to provide a new stable, generic and expandable gaming platform. Additionally, as concluded in the previous chapter, there are some shortcomings to the current version of ARTigo's games, which should be improved. Therefore, at first, the additional features which have been added to the new implementation are presented. Afterwards, its software stack is introduced and the existing database model is outlined. A closer look is given to the design of server and client.

5.1 New features

The newly created Semantic Categorization Pipeline provides the potential to enhance the previous version of ARTigo with linguistic features. Parts of the pipeline, the preprocessing and autocorrection modules, have been extracted and used to enhance the precision of the gathered tag data. Additionally, an improved version of ARTigo's bot opponent has been created. Those new features are presented in the following subchapters.

5.1.1 Bot types

In the previous version of ARTigo it was not possible to play against a real opponent. While it makes sense to provide a bot opponent if only one player is waiting to start a game, it would be a pity if two players could be matched to each other at some time but the system would not allow it. Therefore, in the new implementation the matching algorithm waits for six seconds for another player to join the game. If there is no opponent, the player will be matched to a bot.

Additionally, the existing bot opponent randomly sends some tags which were previously entered for the given artwork and language by other players. The new implementation allows the player to choose between two types of bots. The bot which will be chosen by default is called `RandomTagsBot` and provides the same logic as the existing bot. The second type of bot is called `ReplayRoundsBot` which provides a more realistic game experience by reenacting previously recorded game rounds for the given artwork and language by real players as employed in the ESP Game by Luis von Ahn [vAB05].

5.1.2 Cleaned tag

Furthermore, the tag input was not checked or validated in the previous version of ARTigo apart from removing leading or trailing spaces and disallowing empty words. However, as the game design compels the player to enter tags under time pressure, typing errors can be expected. For example, as presented in **Figure 5.1**, the cleaned tag for the German adjective *schwarz* (black) can be mapped to fifteen different spellings in table `tag` which differ mainly by the use of punctuation characters. Nonetheless, as punctuation characters and typing errors are not removed, an entered tag with different punctuation characters could not be mapped to an existing tag and would therefore not receive any points. Hence, the new implementation should first transform each entered tag into its cleaned version and use this cleaned tag to compare with the database's or the opponent's tags. Additionally, by using the preprocessing module of the Semantic Categorization Pipeline to transform the tag into its cleaned version it should now not be possible anymore to enter the same words in singular and plural. Previously, if a player would describe an artwork displaying a forest he could use the tags *tree* and *trees* and possibly even receive points for both words. In this example, however, there is a semantic difference between an artwork showing a single tree and an artwork displaying many trees. Therefore, this distinction between singular and plural could invite the player to think about which number of the word is more accurate to use and therefore enhance the accuracy of the tag input. Moreover, when playing ARTigo Taboo the player was able to trick the game by entering singular taboo tags in plural or vice versa and receive points.

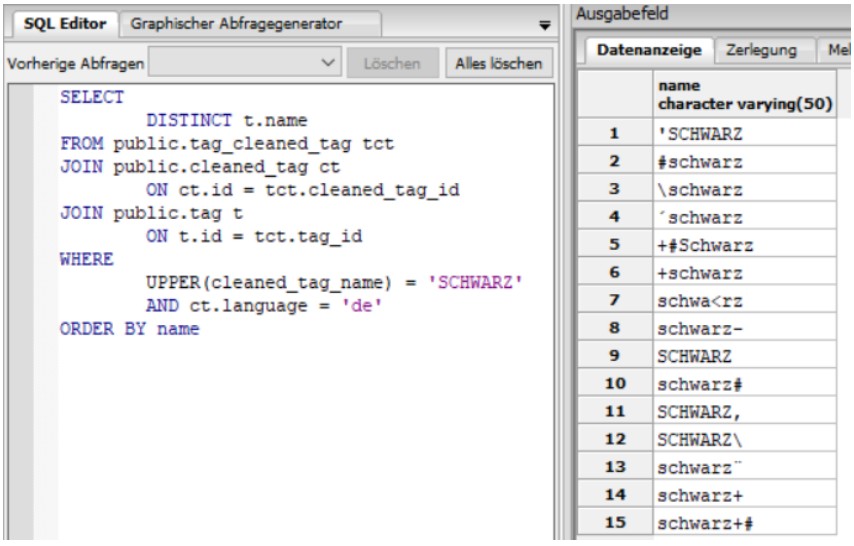

Figure 5.1: Spellings of *schwarz* (black) in the database

Unfortunately, tests on the new feature showed that the call to the preprocessing module of the Semantic Categorization Pipeline takes three to four seconds each time a player enters a new tag and therefore the player has to wait too long for the server's response containing the updated tag data to allow for a great gaming experience. Especially when playing ARTigo Taboo the conversion of all taboo tags to their cleaned version at the beginning of each game round leads to a visible delay in starting the game. An analysis showed that the conversion of the tag into a cleaned tag itself does not take more than one second, however, each time the preprocessing module is called by the server for each tag the lemmatizers and part-of-speech taggers have to be initialized, which takes two to three

seconds. Therefore, entered tags are currently transformed into two cleaned versions. The cleaned tag resulting from the preprocessing module is fetched in the background allowing the game to proceed. A second cleaned tag is used to compare the player's tag against the database's and the opponent's tags during the game round. To ensure an enjoyable gaming experience only punctuation characters as well as leading and trailing spaces are removed directly in JavaScript. When a game session is finished, the cleaned tag from the preprocessing module is used to subsequently remove invalid tags from the tag list and recalculate the player's score to show the updated tags without their duplicates in the overview of played resources. If the player has entered a tag in singular and plural only the first version of this tag is chosen while the other one is discarded. A resources overview with removed tags is shown in **Figure 5.2** with the respective artwork "Toernooi op de Grote Markt te Brussel" by an unknown artist.

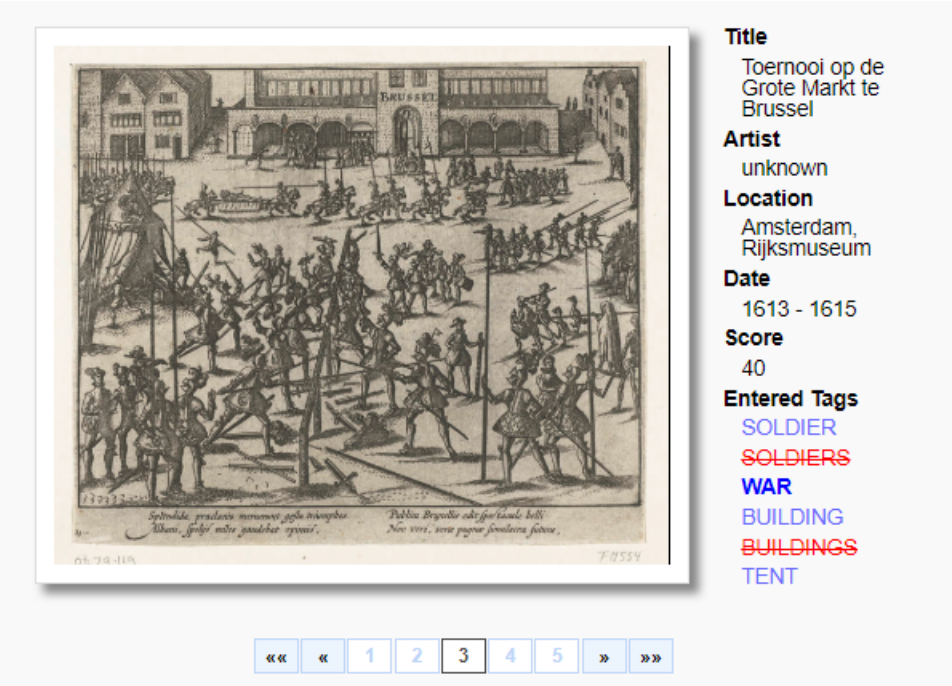

Figure 5.2: Removal of invalid tags in the resources overview (Screenshot from [Prob], significant parts only)

5.1.3 Spell check and tag corrections

In addition to the usage of cleaned tags to validate the tag input a spell checking algorithm is added to the new implementation. When the player enters a tag, the algorithm checks asynchronously if this tag is known. If the tag is unknown it will be underlined in orange. If the algorithm finds corrections for the misspelled word the tag will be colored completely in orange and suggested tag corrections will be provided to the player if he moves the mouse over the respective word. The player, additionally, has the possibility to remove entered tags which did not receive points yet for example if he identifies a spelling error. He can then choose to correct the tag into one of the suggested words, delete it or ignore the warning. This new feature is presented in **Figure 5.3**.

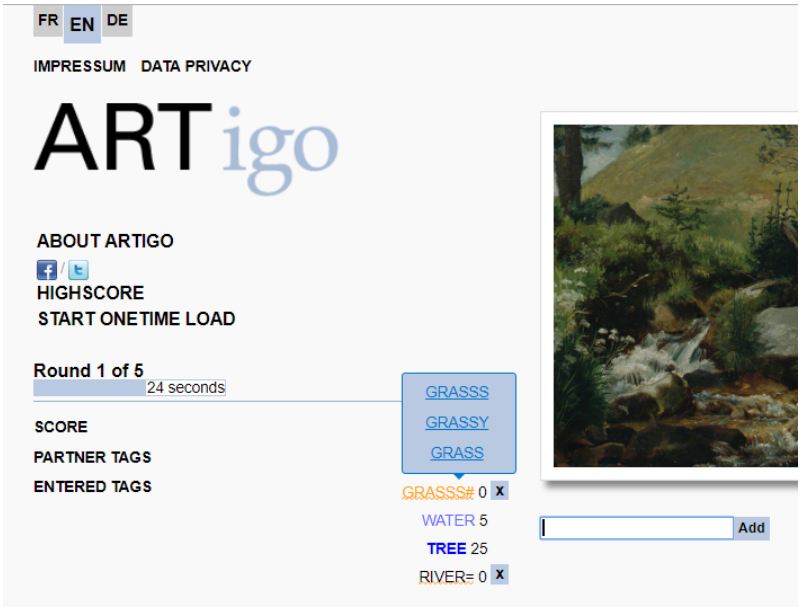

Figure 5.3: Spell check feature (Screenshot from [Prob], significant parts only)

To find suggestions for incorrect tags, the autocorrection module from the Semantic Categorization Pipeline is used. If a tag is "unknown" the autocorrection module suggests a list of candidates with a Levenshtein distance of at most two which corresponds to one or two letters in the word being substituted, added or removed. Additionally, the first suggestion provided to the player is the cleaned version of the entered tag as this cleaned tag will most likely provide the best correction if it is not identical to the original tag and is also "known". The resulting list from the autocorrection module is then processed in the order the candidates were returned by python until three "known" tag corrections are found. Therefore, the most promising candidates which offer a smaller Levenshtein distance are evaluated first. If a candidate is a "known" word it will be used as a suggestion. If up to three tag corrections are found, the tag will be displayed in orange and the user can choose a correction. If no tag corrections are found, the tag will remain underlined in orange.

To check if a new tag or a tag correction is allowed four possibilities to define "known" were evaluated. The first option considers a tag "known" if the autocorrection algorithm recognizes it as a valid word. However, as the autocorrection algorithm does recognize many words as valid which do not exist in the respective languages this option was discarded. The second possibility considers a tag as "known" if it is persisted as cleaned tag by the Semantic Categorization Pipeline in the `cleaned_tag` table. Still, with this option some valid words would be discarded as they do not exist as a cleaned tag for this language. Third, a tag could be classified as "known" if it was persisted in the `tag` table for the current language and was used by at least two unique players. As the `tag` table is large enough after ten years of gathering data this option does only correct words if they are wrong or infrequent and therefore do not exist in the `tag` table already. However, this option also allows all words as tag corrections which are in the `tag` table including many misspelled words. One example for a misspelled word acting as a tag correction is shown in **Figure 5.3**. While the valid correction *grass* for tag *grasss#* is also provided, the first suggestion is *grasss* as it has a smaller Levenshtein distance to the original misspelled tag and is found in the `tag` table. However, as only corrections provided by the autocorrection algorithm will be checked against the `tag` table and as it seems more important to only correct words

which are actually wrong than to provide correct tag suggestions this option is currently in use. Additionally, by increasing the number of unique players which should have entered the tag in the table for the tag to be suggested as a correction, the wrong tag suggestions could be removed as well. A fourth possibility would be to further restrict the previous option to consider only tags which were entered for this artwork as valid. However, as this option would recognize correct words as wrong because they have not been used for this resource before this option was also discarded. Future research on this topic could also examine the possibility of combining multiple definitions of "known". One combination could be to use the existing definition of "known" only for entered tags and to evaluate tag corrections based on their existence in the cleaned tag table.

5.2 Software stack

The main goal of the reimplementation is to provide a solution which is stable, generic and allows the extension of the gaming platform with additional games in future work. Therefore, the reimplementation of the games uses JavaScript as it is the most popular and extensively used programming language. Node and Express were chosen to provide scalability, performance and cross-platform support. Server and client were implemented as separate projects to allow them to run independently on different hosts while communicating with each other through sockets. Reliable real-time message exchange between client and server is provided by Socket.io.

To simplify the debugging process, monitor the execution of server and client and identify potential problems, Winston was added as logger supported by logger middleware Morgan. Consistent code quality is ensured by the use of linter eslint. Additionally, unit tests were implemented with the help of Mocha, Chai and Sinon. Mocha is a testing framework for Node which offers the possibilitiy to write tests, which can be run after each code change. To specify the expected output of the tests in a readable way, an assertion library is provided by Chai. As the test subject might include calls to the database which should be avoided or require a specific setup, Sinon supplies simulations to imitate their behavior, called stubs. The stubbed function acts as a replacement for the original function, which will not be called. Additionally, the number of calls, their parameters or any return value of the function can be analyzed or specified.

To persist collected data, ARTigo uses the preexisting PostgreSQL database `gwap` of the previous ARTigo platform. Details on this database will be presented in **Chapter 5.3**. Access from the server to the database is provided by Objection and Knex. Objection is an object-relational mapping tool which allows to access the database directly from Node without writing SQL statements. Furthermore, it secures the database against attacks like SQL Injection when using parameter bindings. In addition to Objection, its underlying query builder Knex allows to specify more complex SQL queries if needed.

The client is build with JavaScript library React. React employs basic reusable components which are then composed to build a complex front-end web application. Additionally, React offers create-react-app, a tool which allows to start developing a cross-platform, cross-browser and cross-device React App without taking care of configuration or dependencies of other tools including Webpack and Babel. To allow the translation of texts in the three ARTigo games, internationalization library react-intl was added. With this library, translations between German, English and French are taken from json files.

An overview of the implemented software stack is given in **Table 5.1**.

	Software	**Usage**
Platform	Node[1]	Both
Framework	Express[2]	Both
Real-time communication	Socket.io[3]	Both
Database	PostgreSQL[4]	Server
Query Builder	Knex[5]	Server
Object-relational mapping	Objection[6]	Server
Logger	Winston[7]	Server
Logger Middleware	Morgan[8]	Server
Linter	eslint[9]	Server
Unit Testing Framework	Mocha[10]	Server
Unit Testing Assertion Library	Chai[11]	Server
Unit Testing Stubs	Sinon[12]	Server
JavaScript Framework	React[13]	Client
React Tool	create-react-app[14]	Client
Internationalization	react-intl[15]	Client

[1] https://nodejs.org
[2] https://expressjs.com
[3] https://socket.io
[4] https://www.postgresql.org
[5] https://knexjs.org/
[6] https://vincit.github.io/objection.js
[7] https://www.npmjs.com/package/winston
[8] https://www.npmjs.com/package/morgan
[9] https://eslint.org
[10] https://mochajs.org
[11] https://www.chaijs.com
[12] https://sinonjs.org
[13] https://reactjs.org
[14] https://github.com/facebook/create-react-app
[15] https://github.com/yahoo/react-intl

Table 5.1: Relevant tables in gwap database

5.3 Database model

As only three games of the ARTigo gaming ecosystem are reimplemented in the scope of this thesis while the other existing games should still work with the previous implementation, the database model has to be kept intact to allow for parallel execution of both versions. An overview of all relevant tables for the reimplementation in the gwap database including the new tables is given in **Table 5.2**.

Table Name	Type of game	Content
gamesession	All	All game sessions
gametype	All	Available game types
gameround	All	All game rounds
person	All	Player and artist information
tagging	ARTigo Game ARTigo Taboo	Mapping of tag to artwork
tag	ARTigo Game ARTigo Taboo	All tags
combination	Combino	Mapping of combined tag to artwork
combinedtag	Combino	All combined tags
artresource	All	General information about artwork
artresourcecache	All	Temporary storage for upcoming artwork
artresourcetitle	All	Titles of artwork per language
source	All	Source of artwork
cleaned_tag	ARTigo Game ARTigo Taboo	Cleaned versions of all tags
tag_cleaned_tag	ARTigo Game ARTigo Taboo	Mapping of tags to cleaned tags
semantic_category	ARTigo Game	All semantic categories
cleaned_tag_semantic_category	ARTigo Game ARTigo Taboo	Mapping of cleaned tag to semantic category
semantic_category_hierarchy	ARTigo Game ARTigo Taboo	Hierarchies for semantic categories

Table 5.2: Relevant tables in gwap database

At first, a new game session is created in table gamesession. It includes a reference to the game type of the session, for example ARTigo Game. Table gametype defines the different types of games with their number of players, number of rounds and round duration. Each player in the game session is assigned to separate game rounds in table gameround with a reference to the game session. Therefore, for a complete game session between two players there will be ten entries in table gameround - five rounds for each of the two players. The players themselves are stored in table person which also includes information on the artwork's creators.

The three reimplemented games ARTigo Game, ARTigo Taboo and Combino offer two different types of collected data. In ARTigo Game and ARTigo Taboo, the player enters words, which will be persisted as tags in table tag. Table tagging stores the mapping between artwork, tag, game round and person with the tagging's timestamp and score while table tag stores only the unique tags and their language. For Combino, the players do not enter new words but combine entries from table tag with each other to create combined tags. In this setting, table combination accordingly stores the mapping between artwork,

combined tag, game round and person together with the combination's timestamp and score. Differing to the setting of tagging and tag, `combination` also includes the combined tag's language. Table `combinedtag` offers information on the combined tag's value, which is a concatenation of both tags' names, and references to the tags which where combined.

Information on the artwork which are presented in each game round are stored in tables `artresource`, `artresourcetitle`, `artresourcecache` and `source`. Table `artresource` maps the artwork to its source and artist and offers additional information on the artwork including the name of its respective image on the server. The titles of the artwork in the different languages can be found in table `artresourcetitle` with a reference to their respective art resource. An image's path on the server can be found in table `source`. To retrieve a specific art resource for a game round promptly without the player noticing a delay, table `artresourcecache` provides a temporary storage for the next art resources to be played which is refreshed once a certain minimum threshold of art resources for the current game type is reached.

Additionally, tables `cleaned_tag`, `tag_cleaned_tag`, `cleaned_tag_semantic_category`, `semantic_category` and `semantic_category_hierarchy` have been created to persist the information gathered by the additional features and the Semantic Categorization Pipeline. Detailed information on these tables is given in **Chapter 3.1.7**.

In the previous implementation, only one sequence is used to increment the key ids for all tables. Therefore, the tables would not have continous ids themselves but one continuous id over all tables. For a new game session id **1** could be given to the game session, ids **2** and **3** to both players, ids **4** and **5** to the first game rounds for both players et cetera. As this behavior is rather unusual in the design of a database in the previous version several sequences were added in the reimplementation to allow each table to increment its own id.

5.4 Implementation of server and client

In the new implementation server and client were created in separate projects to allow them to be executed independently on different hosts. Additionally, the implementation follows the Model-View-Controller (MVC) paradigm. With this paradigm a strict separation is ensured between the underlying data model, the view a player can see and interact with in the browser and the controllers which manage the game logic. This separation, additionally, ensures easy compatibility with multiple views. The access to the database is handled by model classes representing the different database tables which are controlled by the object-relational mapping tool Objection. All other classes of the server constitute the controllers, which implement the game logic and communicate with model and view. The view is implemented in the client project and presents all data received from the server via socket communication. Details on the design of server and client will be explained in the following subchapters.

5.4.1 Server

The entry point for all communication between server and client is class `CommunicationHandler`. `CommunicationHandler` listens for events from the clients and transfers them to the respective classes. Messages are sent for example to notify the server about new players, to start or end a game round with the correct game round data or to handle entered tags. An overview of all communication between client and server is given in **Table 5.3**.

Message	Sender	Recipient	Reaction of recipient
Request random resource	Client	Server	Retrieve and send random resource
New player	Client	Server	Find game session for player
New tag	Client	Server	Process new tag
Correction	Client	Server	Correct tag
Delete tag	Client	Server	Remove tag
Request more tags	Client	Server	Send additional tags for Combino
Round finished	Client	Server	Start subsequent game round
Disconnect	Client	Server	Inform opponent about end of game
Quit game	Client	Server	Inform opponent about end of game
Onetime	Client	Server	Start onetime categorization load
Highscore	Client	Server	Send highscore data
Random resource	Server	Client	Display random resource
Connected	Server	Client	Display connecting screen
Start gameround	Server	Client	Start game round
Round data	Server	Client	Update game round
Update tags	Server	Client	Update tag list
Error message	Server	Client	Display error message
More tags	Server	Client	Display additional tags for Combino
Resources overview	Server	Client	Display resources overview
User disconnected	Server	Client	Display resources overview
Highscore data	Server	Client	Display highscore

Table 5.3: Communication between client and server

If `CommunicationHandler` receives a message from the client, he coordinates his response with `GamesManager`. For example, for message "new player" `CommunicationHandler` informs `GamesManager` about the new player and waits for `GamesManager` to tell him about an existing opponent. If there is an opponent, `CommunicationHandler` informs the client to start the game round. If there is no opponent, `CommunicationHandler` waits for six seconds to allow `GamesManager` to find a real opponent. If there is still no opponent after six seconds have passed, `CommunicationHandler` asks `GamesManager` to create a bot according to the bot type the client chose and informs the client about the start of his game round.

Class `GamesManager` manages all game sessions currently in play in the ARTigo gaming ecosystem. If `CommunicationHandler` informs `GamesManager` about a new player, `GamesManager` tries to find a game slot for the player in a game session with the same game type and language. If `GamesManager` can find a game slot, he adds the player to this game session. Alternatively, he creates a new game session for the player. Additionally, `GamesManager` transfers all messages from `CommunicationHandler` to the correct game session once the game has started and retrieves data for the highscore from the database.

Game sessions created by `GamesManager` are managed by class `GameSession`. Once a game session is created and persisted in the database, it is connected to its game type retrieved by class `GameType`. `GameSession` retrieves artwork from `ArtResourceManager` and creates new game rounds for this artwork if the game session consists of enough players. If a game round is finished, `GameSession` saves the player's score and tags with the displayed artwork to send them as resources overview at the end of the game session.

Each game round of a game session is controlled by class `GameRound`. The most important task of `GameRound` is the processing of newly entered tags which `CommunicationHandler` receives from the client and transfers to the correct game round with the help of `GamesManager` and `GameSession`. Tags can only be added if they have been received in time before the game round is finished and if they are not empty. If a new tag passes this first test it is converted to its first cleaned version by replacing all punctuation characters. Then, `GameRound` checks if the respective player is allowed to enter the cleaned tag by verifying that the player has not entered the tag previously with the same or different case. If the player is allowed to enter the tag, it is saved and its score is calculated. Therefore, `GameRound` checks if the opponent has entered the cleaned tag before with the same or a different case or if the tag already exists in the database for this artwork. Additionally, the second cleaned tag, as calculated by python, is retrieved in the background to allow for the elimination of tags which were entered in singular and plural at a later time in the resources overview. If the player's opponent has also entered the tag, both players' scores are updated with twenty-five points. If the opponent did not enter the tag before but it is already present in the database for this artwork, the player still receives five points. If the tag was rejected for any of the above reasons the respective error message is distributed to the client via `CommunicationHandler`. Other functionality class `GameRound` offers, is for example to send game round information including the duration, number of rounds, current round and artwork at the beginning of each round and to handle the deletion or correction of tags.

To ensure straightforward expandability of the implementation for new games and avoid duplication of code, inheritance is used. Therefore, additional games can extend or overwrite the functionality of `GameRound` to their needs. To add ARTigo Taboo to the new implementation, it suffices to create a new class `TabooGameRound`, which expands the existing methods of class `GameRound`. To augment the round data which is sent to the client at the beginning of each game round, `TabooGameRound` adds a list of taboo tags which the player is not allowed to enter during the game round. Additionally, to validate the player's tag input or possible tag corrections, `TabooGameRound` checks if a word is contained in the list of taboo tags before calling the respective methods of the super class `GameRound`. Likewise, `CombinoGameRound` adds a list of displayed tags to the round data which the client can choose tags from to combine into a new tag. If the client sends a new tag combination, `CombinoGameRound` makes sure both tags are not identical and, therefore, allowed to be combined before calling the respective methods of `GameRound`.

The players in the game session can either be real humans playing the game in realtime or one human player and one bot. Class `Player` represents a human player and stores the list of tags a player has entered in a game round with methods to update scores and modify tags. If there is no opponent, a bot is created to allow the human player to start the game. The player can choose between two bot types, `RandomTagsBot` and `ReplayRoundsBot`. Both bots inherit functionality from class `Player` and additionally use methods of class `CommunicationHandler` to send tags and start or finish game rounds like human players. `RandomTagsBot` represents a bot that randomly selects tags which have been used for the given artwork previously. To simulate a real opponent delays have been added at the beginning of the game round and between each tag. Detailed information on this bot logic, which has also been used in the previous implementation, has been given in **Chapter 4.4**. A different logic is offered by class `ReplayRoundsBot` which extends `RandomTagsBot`. This bot type retrieves previously played game rounds for the given artwork with more than five taggings which have not been performed by the player himself from the database and reenacts those rounds' moves. As the timestamp of each tagging is stored, it is possible to play each move in the exact second it has been played in the original game round. With this behavior, the bot acts more human and might not be identified as a bot player as easily.

The displayed artwork is provided to GameSession by ArtResourceManager. This class offers two ways to retrieve the subsequent artwork. If the game session includes only human players or one RandomTagsBot, the next artwork is chosen from the prepared cache of art resources for the given type of game. The chosen art resource is then removed from the cache. Once the count of art resources for the current game type in the cache falls at the threshold of ten images, the cache is refreshed in the background. However, if the game session includes a ReplayRoundsBot, the chosen artwork has to be tagged sufficiently as ReplayRoundsBot can only reenact game rounds with at least five taggings to ensure some action happening in the game round and, therefore, entertainment for player. Therefore, in game sessions with a ReplayRoundsBot, ArtResourceManager retrieves the artwork directly from the database. As this delays the start of a game round, all artwork of the whole game session are retrieved at the beginning of the first game round. Therefore, the delay is only visible at the beginning of the first game round, but the following game rounds can be started immediately. Another cache might be implemented for game rounds with ReplayRoundsBot to increase performance in future work.

The entered tags are represented by class Tag, which stores a tag's language, score, possible tag corrections and the two versions of its cleaned name. If a tag is new, it is persisted in the database in table tag and tagging. If the tag existed in the database previously only its tagging is stored and connected to the existing tag. When a player chooses to correct a tag the tag itself is only updated in the database if it was created by the current player in the current game round. If the corrected tag does already exist in the database, the tagging is updated to use the id of the corrected tag and the previously entered tag is blacklisted. By blacklisting a tag it will not be considered in any other query and therefore acts as if it was deleted. If the tag has not been created by the current player, only the tagging is updated to the corrected tag. If the corrected tag is a new tag itself, it will also be persisted in table tag.

For Combino, a tag does not represent an entered input value but a combination of two previously existing tags. This combination is represented by class CombinedTag, which extends class Tag but almost only varies in the use of the different database tables combinedtag and combination instead of tag and tagging. A third representation of a tag, TagWithCategorization was created to connect the tag with its categorization. This version if currently used as the default tag version. In this class, a tag is analyzed by the Semantic Categorization Pipeline if it is a new tag and therefore will be persisted in the database. If categories are available for a tag, this information is persisted in the semantic categorization tables as described in **Chapter 3**. Additionally, the Onetime Categorization Load is offered by class TagWithCategorization which is used to categorize all existing tags in table tag which had been added to the database before the new categorization feature was added. The Onetime Categorization Load had to be run before the feature was added as it will start at the last categorized id in table tag_cleaned_tag in order to make the Load interruptable. In batches of a hundred tags it then retrieves categories from python and stores the results in the semantic database tables.

Figure 5.4 provides an overview of all controller classes used to build the server.

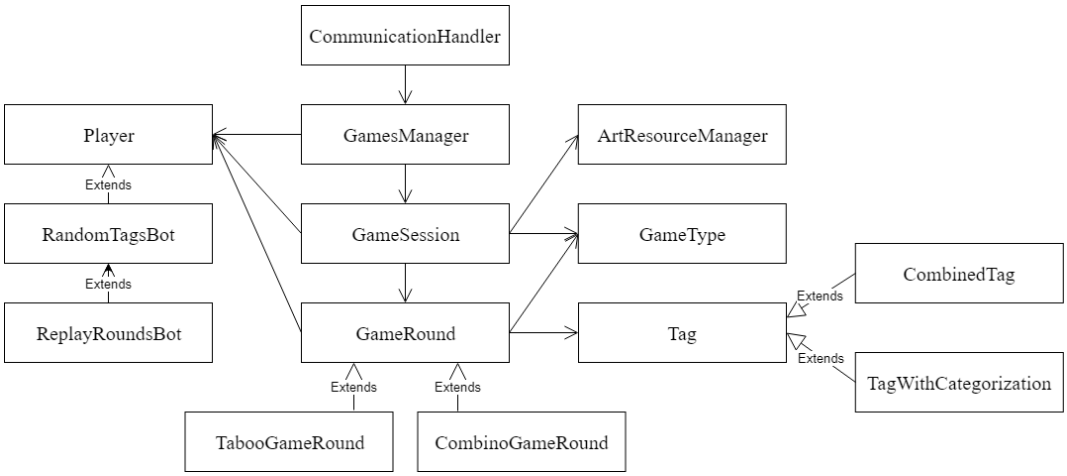

Figure 5.4: Overview of server's controller classes

5.4.2 Client

The React client consists of many small components as React favors the use of composition over inheritance. The new ARTigo App is wrapped in an `AppWrapper` class which allows to switch between the languages German, English and French. The class `App` itself is the root of a tree of components with branches into the navigation panels as well as the content of the page. Therefore, `App` offers methods to transfer information between those branches.

The Home screen of the App is presented in **Figure 5.5** displaying the image "Staande jonge man" by artist Jacob de Gheyn with added colored boxes marking the different components. This screen also appears if the user clicks on the ARTigo logo. At the top of the screen two navigation panels are located, `LanguageHeader` (blue) at the left and `BotHeader` (green) at the right, which both allow for the user to switch between the different languages and bots. Below `LanguageHeader`, `NavigationHeader` (yellow) provides links to the `Impressum` and `DataPrivacy` components. Component `SidePanel` (grey) displays the ARTigo logo, the `NavigationPanel` (brown) and the `GamesNavigationPanel` (pink) in this view. `Navigation-Panel` offers links to the components `About` with information about the ARTigo Project and its games, `Highscore` displaying the highscore for all games for the periods current day, current month and previous month as well as `OneTime` which starts the Onetime Categorization Load. Additionally, it displays icons to redirect the user to ARTigo's Facebook and Twitter pages. The main part of the screen is directed by component `Content` (red) which in this case displays a random resource with the help of component `Resource` (orange) to welcome the user to ARTigo.

Once the user has started a game some components are replaced as shown in **Figure 5.6** displaying a round of ARTigo Taboo with the artwork "El Escorial" by artist Juan B. de Toledo. In the `SidePanel` component `GamesNavigationPanel` has been replaced by `RoundDataPanel` (purple) displaying all information of the current game round. At the top of `RoundDataPanel`, component `TabooTags` displays the list of taboo tags for this game round. Below, the `Timer` component shows a progress bar with the remaining seconds. At the bottom of `RoundDataPanel`, the component `TagList` presents all tags the user entered with their score. The main part of the screen is again wrapped in component `Content` which includes the component `Connection`. This component creates the socket connection to the server and shows a connecting screen while the server finds a fitting opponent or bot. Once

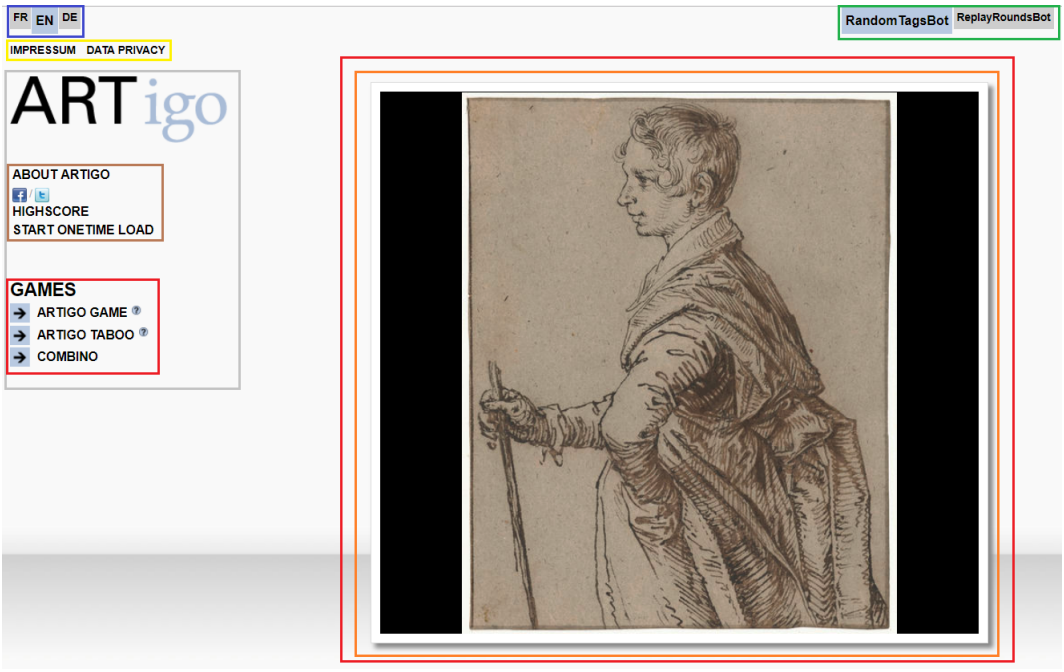

Figure 5.5: Home Screen

the game is ready to start component `Connection` displays component `Game` which then resumes the communication with the server. Wrapped in component `Game` are the displayed `Resource` as well as the `TagForm` (light green) which allows the user to enter tags. For Combino, the SidePanel displays the component `CombinoRoundPanel` which also shows the tags to be combined and a button to load more tags.

When the game is finished, an overview of all resources in the game session is presented as displayed in **Figure 5.7**. Wrapped in `Content`, the component `ResourcesOverview` (turquoise) displays the `Resource`, a button group to switch between the different resources as well as its `ResourceData` (light blue). In the SidePanel to the left, the `ResourcesOverviewPanel` (light purple) allows the user to start a new session of the game he just finished.

To ensure the view's design to be identical to the previous version of ARTigo, similar stylesheets were used. The switch between the languages is dependent on tool `react-intl`, which allows the use of variables in the different components which will then be substituted by the text in the current language controlled by `AppWrapper`. All translations are stored in JSON files which allows for straightforward translations to additional languages in future work.

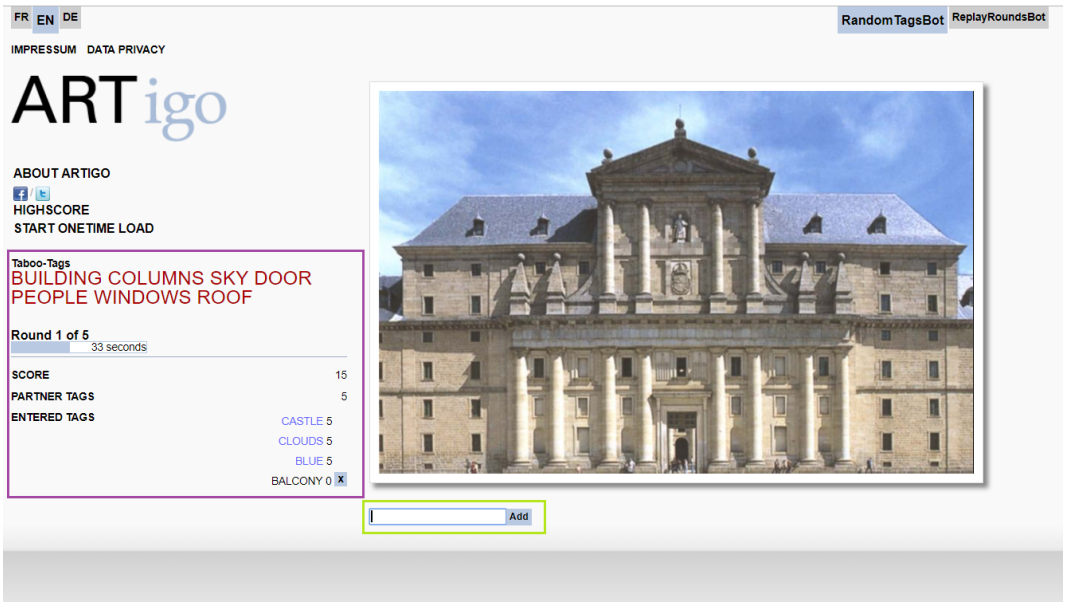

Figure 5.6: Game Screen

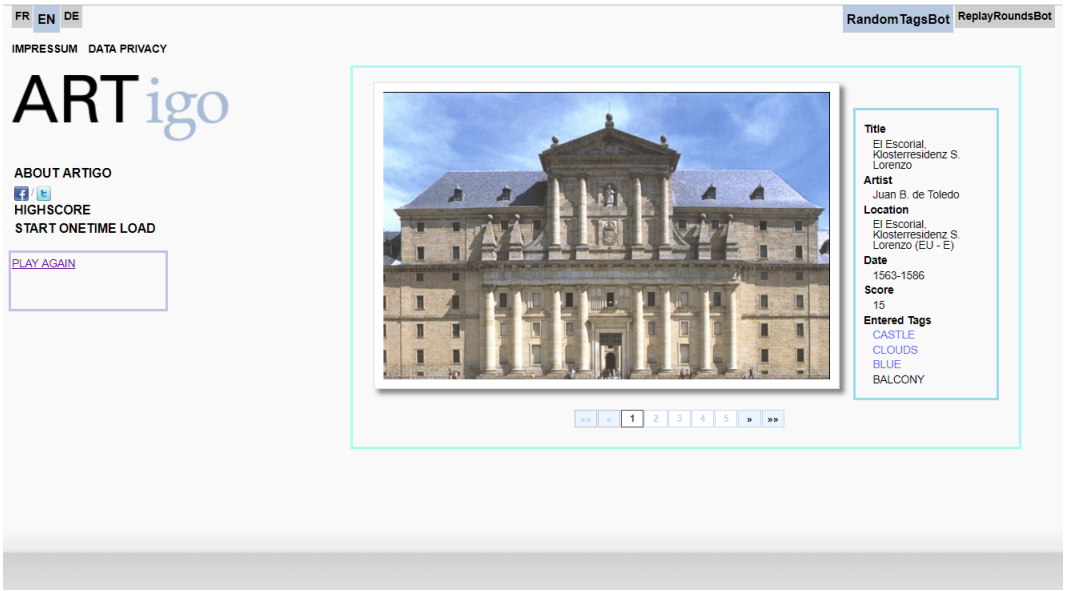

Figure 5.7: Resources Overview Screen

CHAPTER 6

Conclusion and Outlook

The ARTigo Gaming Ecosystem is a Human Computation system. Human intelligence is utilized to solve computational problems that exceed the capabilities of computer algorithms. There are different incentives for humans to contribute their time to scientific purposes, for example getting paid, altruism, interest in science or learning. Additional incentive is provided by Games with a Purpose, an idea by Luis von Ahn to motivate people to contribute to science in a playful manner. The ESP Game by Luis von Ahn serves as a model for the three main games ARTigo Game, ARTigo Taboo and Combino in the ARTigo Gaming Ecosystem. In the ESP Game, two players are presented with the same image and, in order to receive points, have to enter tags that their opponent might also use.

The first goal of this thesis was to implement a Semantic Categorization Pipeline to gather information about each tag's categories. The Semantic Categorization Pipeline consists of six layers which each tag has to pass through. At first, a tag is preprocessed into a normalized form. In the second layer, the lexical category of each tag is conducted and tags are only further processed if they are noun, verb, adjective or adverb. Next, Named Entity Recognition algorithms are used to find locations, persons or organizations in the tag. Semantic Relations are used in the fourth layer to determine the hypernyms of tags. There are three APIs, Wikidata, ConceptNet.io and WordNet, which define several relations between items of their knowledge bases used to determine a tag's category. Eventually, all categories have to be postprocessed to ensure the uniqueness and validity of the returned categories. If no category was found, there might have been a typing error in the tag. In this case autocorrection is used with a distance of one and the Semantic Categorization Pipeline is recursively started with the corrected tag exactly once.

Some limitations to the pipeline were presented that could not be addressed in the scope of this thesis, for example differences in the quality of the results depending on the part-of-speech and language of each tag. A main obstacle was to determine the best-describing category out of the many associated categories of a tag. This problem was solved by allowing multiple categories for each tag and persisting original categories as subcategories before shortening. Finally, the moderate performance of the Semantic Categorization Pipeline was discussed, which could potentially be enhanced by storing API data locally.

The Semantic Categorization Pipeline successfully categorized 16% of all tags and 44% of all validated tags. As a high percentage of tags did not receive categories in the process, the sources of error for missing classification were examined with sample data for each language. The analysis revealed incorrect grammatical case and missing results from the APIs as the most important sources of error. Additionally, many tags were not considered correct because they used invalid multiple-word tags, contained spelling errors, did not provide the correct language or were not even considered a real word. Furthermore, especially for French tags the postprocessing and conversion to singular led to a few missing classifications and demonstrated the dependency of the pipeline to the external tools. Some sources of error could be avoided, for example the rejection of numbers as stopwords or the misclassification of French tags including the language codes "de" or "en" as German or English tags. In a second evaluation, the correctness of the collected category data was examined. Correctness values of around 70% were determined for all three languages. In some cases invalid tags were categorized or categories were returned which were related to the cleaned tag but could, however, not be considered valid categories. In some cases the cleaned tag name itself was returned as category by Wikidata's API. Future evaluation could detain a study to identify the completeness of the resulting categories.

As the Semantic Categorization Pipeline produced a lot of resulting categories, this data could be put to use not only in the ARTigo search engine but also in future applications. A new game could be created similar to ARTigo Game, which could offer bonus points if the player uses words out of one randomly displayed category. The displayed categories could either be retrieved from categories that were assigned to this image by the Semantic Categorization Pipeline previously or some general categories which fit for almost all images could be manually defined. With this game, more specific tags could be gathered and the player could be guided to some main categories, for example *painting techniques* or *emotions*. There could also be bonus rounds which could be employed to disambiguate a tag's meaning and to find the most fitting category for an artwork's tags. In those bonus rounds, ten tags could be displayed individually for one image with their categories. The players would have to choose the most fitting category for each tag in a short time and would receive bonus points if their chosen category fits their opponent's.

Furthermore, the Semantic Categorization Pipeline could be enhanced. On the one hand, categories could be further specified by collecting additional subcategories. It could, for example, be interesting if a tag categorized as a person represents an artist and if this artist is a painter or a sculptor and so forth. For this case there are several additional APIs, for example the Union List of Artist Names (ULAN) which could be used to identify tags which represent artists, the Thesaurus of Geographic Names which could be used to specify tags which connote locations and more generally Wikipedia which could be queried to further specify the category of almost any tag. On the other hand, the Semantic Categorization Pipeline has provided thousands of categories and it might therefore make sense to reduce this amount to a smaller set of verified categories. For this purpose, some main categories, for example object, person, location or emotion, could be defined manually and tags could be assigned with probabilities to those main categories with the help of Machine Learning techniques.

As the ARTigo platform was facing some difficulties with their underlying technology and the performance of their servers, the second goal of this thesis was to reimplement the three main games with JavaScript software. Additionally, new features were added to utilize the power of the Semantic Categorization Pipeline and further improve the ARTigo gaming ecosystem. As the categorization could benefit a great deal if the input was validated, for example by checking for typing errors, two versions of cleaned tags as well

as a feature to identify, correct and remove misspelled words were added. Furthermore, another type of bot was implemented which reenacts previously recorded game rounds to simulate a more realistic opponent.

However, the implementation of the cleaned tag feature still shows room for improvement as the current implementation only discards duplicate tags in both singular and plural when the game session is already over as the prolonged period of time it takes to retrieve the cleaned tag during the game prevents a great gaming experience. To accelerate the retrieval of the cleaned tag from the preprocessing module it might be an option to keep the python scripts running in the background to remove the delay which seems to be caused by the loading of the background data at the start of the python module.

Another feature for input validation could be to further evaluate tags consisting of multiple words. For example, the tag *Mona Lisa* consists of two words and makes sense as a single tag as it describes one entity. However, tags like *säule vor romanischem bogen* (column in front of romanesque arch), *bildnis eines knaben* (portrait of a boy) or *der gleiche typ wie gerade* (the same guy as before) should not be allowed as valid tags. This kind of input validation could be implemented with lookups in lexical resources and Named Entity Recognition algorithms, as implemented in the Semantic Categorization Pipeline. Additionally, it should be ensured that synonyms of already entered tags are not accepted with the use of lexical resources.

The new feature and the additional enhancements discussed in this chapter show only the beginning of the extension of ARTigo with linguistic features. By combining ARTigo's games with the Semantic Categorization Pipeline and additional features like spell checking, tag corrections and the usage of a cleaned tag this thesis has provided a solid foundation and start for future applications and studies.

APPENDIX A

Evaluation

A.1 Sources of error (samples 1 & 2)

A.1.1 German tags (all)

HMMEL	sambol
GARDEN	drei durchgänge
FINGERZEIG	gerhard eduard
NO. 112	adürer
SCHNEEDECKE	minevra
PHOOGRAPHIE	altar figuren gold
JUD	Amphitehater
SPANISCHER MALER	landschaft berge wasser
TRÄUMENDE	fassade rundbogen
MANSCHATTEN	schwarz vasen
SGRAFFITTO	ahin
ORCHIDEEN	Personen farbe
BILLETT	van Beusel
LICCHT	blau lasziv
FILMSZENEN	Verkündigung des Herrn
PRPHET	seite aus zeitung
VORSRPUNG	Rohrschach fleck
ZOTTE	
WAFFE KLEIDUNG KOPFBEDECKUNG	
langestrecktes gebäude	
6 personen	
transparente bluse schwarz	
decke rokoko ornament	
interieur tisch menschen	
theaterloge zeichnung	
täfelschen	
skizze landschaft kohle	
mann hund vogel	
grasüberwachsen	
laube turm häuser	
grün grau blau	
ütabstrakt	
tromp l´euil	

A.1.2 German tags (validated)

FEDERN	FABEL	FELDHASE	CISNERIUS
BILD	HAUSARBEIT	POLIEREN	ERSTKOMMUNION
SEITLICH	MOULIN ROUGE	MURILLO	1476
BERGHANG	AKREUZ	GASSI GEHEN	schwarze flügel
BAUERNHOF	NETT	BUCHHANDLUNG	henry ii
PICKNICK	DACHGAUBE	BISCHOFSMÜTZE	ecclesiam
KLINGER	REKONSTRUKTION	MUSCHELSUCHER	bolognius
SURREAL	GEBORGEN	ANGELEINT	1916
EISEN	EINSCHIFFIG	ZAHNLOS	mosaici
POLIZIST	DELAUNAY	HERMANN	s10
ZIRKUS	SÄUGEN	BOURGEOIS	theeves
KINDERSARG	VERLETZTER	U.S.MAIL	
SCHIEFER	KUTSCH	LEONHARD FUCHS	

A.1.3 English tags (all)

MAIN BUILDING	caos	Cloven feet	mortuus est
FOGGY	witness	walkwaty	sailship
EYELLOW	Illegal	Infanterie	deggeurotype
SORROW	pink blouse	shireld	dath
eye brows	comunism	naked crowd	albania
lissan	horrifying	of coast	woman with braids
esppelie	shore line	augustanus	aposte
windom	mockery	vegetable atrichoke	recumbent
portrait girl color	open field	gehangen	Old face diagram
sandbag	marcus	blosom	no image shown
mint	stands out	vw	bierd
hprses	plunderen	mortus	
buiodojg	soestdyk	holds rod	

A.1.4 English tags (validated)

HAND	BALLA	brown eyes	1909
LANE	APPLES	damaged	patriotism
NARROW	DAYLIGHT	architrave	peg-leg
FOAM	MODES	red boots	1300
MEADOW	CAPS	green trees	draconites
BLUSH	lucifer	tails	1627
NICHES	naked woman	plateware	saintonge
MONK	nimbus	boeuf	dostoevsky
BRUSHSTROKES	cobweb	cheers	hvlst
BLAUER REITER	urbanus	red grapes	consume
STEAL	balck	debating	eaten
MARIA	houres	art works	
ROTUNDA	wellen	binding	

A.1.5 French tags (all)

FLEURS	MOUCHONS	hollandaise	l'échelle de Jacob
ANTIQUITÉ	PIRATE	petites	porter les marchandises
ROUGE*	salle de conseil	disciple	monuement
HORACE VERNET	caravagesque	requête	Kopfbedeckung
BRODE	caravelles	pointue	bousculade
INACHEVÉ	mousses	contact	crucifier
TISCH	pastem	chevauché	homes
TABOU	âgé	pont fermé	peninsule
TENTURES	aphrodite	pages d'un livre	platre#
REGULARITÉ	âtre	comfortable	ruineux
POINTS	ea	document*	épolette
AUGE	roicher	481	
MONT	canope	ffeu*	

A.1.6 French tags (validated)

ESCALIER	FONTAINE	ARGENT	antoine
EDGAR POE	DIGUE	TRITON	boiseries
CENTAURE	CALLIGRAPHIE	BOITE	oreilles
FILLE	CANONS	SILENCE	douve
CHAT	CHRÉTIEN	PAUVRES	dorures
MONET	LANTERNE	VIE	caleche
MATIN	COLIMASSON	LEÇON	oculus
BERGER	BONNET	BRUXELLES	sièges
LIGNES	FAMILLE	HOMME DE DOS	article
COMMEMORATION	FANTOMES	PÂLE	abricot
MUNCHEN	OPÉRA	AUTEUR	lac gelé
CRISTAL	GAUGUIN	TROIS QUARTS	
PYRAMIDE	MARIE ANTOINETTE	PLAISIR	

A.2 Correctness of categories (sample 3)

A.2.1 German cleaned tags

AMPHIBIE	MARTIN LUTER
ANTON VON WERNER	MEDICI WAPPEN
ARMBRUCH	MENSCHENHANDEL
BEHINDERT	MONT SAINT MICHEL
BEKRÄNZEN	Obia
Crustacea	OBRIST
DEHNUNG	R WASSERSPEIER KÄLTE MELANCHOL
DIEGO RIVERA	RAUPUTZ
EDUARD MUNCH	ROTE FAHNEN
ELENDSVIERTEL	SAKO
FÄCHER FOTO JAPAN ROSEN FRAU	SATELLIT
FLIC	SCHMELZE
FRAU PORTRÄT	SCHÜTTEN
FÜNFZIGER	Schwarz
GEDREHTE SÄULEN	SCHWIMMBAD
GEORGE	Segler
HAIN	SIAMESISCHER ZWILLING
HANG	STANDER
HOLZBRÜCKE	TAGELÖHNER
JEAN DRC	TAGESZEITUNG
klimazone	TORF
kreuzworträtsel	TROPFEN
KRÜCKSTOCK	UMGEKNICKTE BÄUME
LANDUNG	weihnachtsabend
LEUCHTREKLAME	ZEITMESSUNG

A.2.2 English cleaned tags

abstracted	doctored	Louis Carint	rocket
Aphricana	doped	Marshland	SAIL
apsis	DRESSER	MAT	scotland
archipelago	ear	mel	shied
Bianco	engage	mutton	shore
bite	EXPRESSIONNISM	Needlwork	sleepyhead
BOUQUET	facees	new moon	slum
Burgh	gondoliere	PAWS	STONES
cantering	guest	Photogragh	symposium
Carave	insane	PLANS	Trafficking
croquet	KOLLER	privilege	varnish
derelict	lawman	ref	
digging	likeness	Regni	

A.2.3 French cleaned tags

1663	dais	Love	rouille
appartement	dance	Malbork	ruche
Apple	enrobé	MARINS	signature
architecte	géant	mât	symbole
BERGERE	géographie	mer	Thomas Becket
bienheureux	glauque	NOIR	Tuer
CADET	harem	nuage	typologie
calèche	HIROSHIGE	ouvrage	veine
cathédrale	INÉTRIEU	palmette	voyeurisme
cauchemar	ingrédient	présentation	XIII
cible	introduction	PROUDHON	XVIe siècle
colonel	jambon	RAPE	
colonisateur	lapidation	RENOIR	

Bibliography

[Bö08] Börkur Sigurbjörnsson and Roelof van Zwol. Flickr Tag Recommendation based on Collective Knowledge. *WWW 2008 / Refereed Track: Rich Media*, April 2008.

[Bag18] Jimit Bagadiya. 171 Amazing Social Media Statistics You Should Know in 2018. `https://www.socialpilot.co/blog/social-media-statistics`, January 2018. Retrieved 2018-09-01.

[BBBP97] David H. Bailey, Jonathan M. Borwein, Peter B. Borwein, and Simon Plouffe. A Quest for Pi. *Mathematical Intelligencer*, 19(1):50–57, 1 1997.

[BDT99] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artifical Systems*. Oxford University Press, New York, 1999.

[BKL09] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media Inc., June 2009.

[Bri] Britannica. Cow. `https://cdn.britannica.com/55/174255-004-9A4971E9.jpg`. Retrieved 2018-08-30.

[BS16] François Bry and Clemens Schefels. An Analysis of the ARigo Gaming Ecosystem With a Purpose. 2016.

[BSS15] François Bry, Corina Schemainda, and Clemens Schefels. A Gaming Ecosystem Crowdsourcing Deep Semantic Annotations. *Citizen Science Games*, January 2015.

[CCC+12] Irene Celino, Dario Cerizza, Simone Contessa, Marta Corubolo, Daniele Dell'Aglio, Emanuele Della Valle, and Stefano Fumeo. Urbanopoly - a Social and Location-based Game with a Purpose to Crowdsource your Urban Data. *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 910–913, September 2012.

[CCH+06] Susan Chun, Rich Cherry, Doug Hiwiller, Jennifer Trant, and Bruce Wyman. Steve.museum: An Ongoing Experiment in Social Tagging, Folksonomy, and Museums. *Museums and the Web 2006: Proceedings*, March 2006.

[COC13] R. Jordan Crouser, Alvitta Ottley, and Remco Chang. Balancing Human and Machine Contributions in Human Computation Systems. In Pietro Michelucci, editor, *Handbook of Human Computation*, chapter 5, pages 615–623. Springer Science & Business Media, 1 edition, December 2013.

[Cur15] Vickie Curtis. Motivation to Participate in an Online Citizen Science Game: A Study of Foldit. *Science Communication*, October 2015.

[CZF+18] Qimin Cheng, Qian Zhang, Peng Fu, Conghuan Tu, and Sen Li. A survey and analysis on automatic image annotation. *Pattern Recognition*, 2 2018.

[CZW13] Minmin Chen, Alice Zheng, and Kilian Q. Weinberger. Fast Image Tagging. *Proceedings of the 30th International Conference on Machine Learning*, January 2013.

[DEG+03] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. *WWW '03 Proceedings of the 12th international conference on World Wide Web*, pages 178–186, 5 2003.

[Deu] OpenStreetMap Deutschland. FAQs: Fragen und Antworten. `https://www.openstreetmap.de/faq.html`. Retrieved 2018-08-31.

[DVJ18] Ayushi Dutta, Yashaswi Verma, and C. V. Jawahar. Automatic image annotation: the quirks and what works. *Multimedia Tools and Applications*, pages 1–21, June 2018.

[Eng18] Eric Enge. Mobile vs Desktop Usage in 2018: Mobile takes the lead. `https://www.stonetemple.com/mobile-vs-desktop-usage-study/`, April 2018. Retrieved 2019-01-31.

[Fli] Flickr. Lass dich inspirieren. `https://www.flickr.com/`. Retrieved 2018-08-29.

[fol] foldit. The Science Behind Foldit. `https://fold.it/portal/info/about`. Retrieved 2018-08-31.

[GMVS09] Matthieu Guillaumin, Thomas Mensink, Jakob Verbeek, and Cordelia Schmid. TagProp: Discriminative Metric Learning in Nearest Neighbor Models for Image Auto-Annotation. *2009 IEEE 12th International Conference on Computer Vision*, pages 309–316, September 2009.

[HAM+18] Kotaro Hara, Abi Adams, Kristy Milland, Saiph Savage, Chris Callison-Burch, and Jeffrey P. Bigham. A Data-Driven Analysis of Workers' Earnings on Amazon Mechanical Turk. *CHI 2018*, April 2018.

[Hav15] John Havel. reCAPTCHA: The Genius Who's Tricking the World Into Doing His Work. `https://thehustle.co/the-genius-whos-tricking-the-world-into-doing-his-work-recaptcha`, December 2015. Retrieved 2018-08-24.

[HH10] Yvonne Hellin-Hobbs. The constructivist museum and the web. *EVA'10 Proceedings of the 2010 international conference on Electronic Visualisation and the Arts*, pages 72–78, July 2010.

[HW08] Mordechai (Muki) Haklay and Patrick Weber. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing*, 7(4):12–18, October 2008.

[JKS+16] Charlene Jennett, Laure Kloetzer, Daniel Schneider, Ioanna Iacovides, Anna L. Cox, Margaret Gold, Brian Fuchs, Alexandra Eveleigh, Kathleen Mathieu, Zoya Ajani, and Yasmin Talsi. Motivations, learning and creativity in online citizen science. *Journal of Science Communication*, 15(3), 2016.

[KCS08] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing User Studies With Mechanical Turk. *CHI 2008*, pages 453–456, April 2008.

[KNA+07] Lyndon Kennedy, Mor Naaman, Shane Ahern, Rahul Nair, and Tye Rattenbury. How Flickr Helps us Make Sense of the World: Context and Content in Community-Contributed Media Collections. *MM '07 Proceedings of the 15th ACM international conference on Multimedia*, pages 631–640, September 2007.

[Koh11] Hubertus Kohle. Kunstgeschichtes goes Social Media. *Aviso: Zeitschrift für Wissenschaft und Kunst in Bayern*, (3):38–43, 2011.

[LA17] Matthew Lease and Omar Alonso. Crowdsourcing and Human Computation, Introduction. *Encyclopedia of Social Network Analysis and Mining*, pages 1–12, 2017.

[LP13] Aidan Lyon and Eric Pacuit. The Wisdom of Crowds: Methods of Human Judgement Aggregation. In Pietro Michelucci, editor, *Handbook of Human Computation*, chapter 5, pages 599–614. Springer Science & Business Media, 1 edition, December 2013.

[LvA09] Edith Law and Luis von Ahn. Input-Agreement: A New Mechanism for Collecting Data Using Human Computation Games. *CHI '09 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1197–1206, April 2009.

[LvA11] Edith Law and Luis von Ahn. *Human Computation*. Morgan & Claypool Publishers, 1 edition, July 2011.

[Mal81] Thomas W. Malone. Heuristics for Designing Enjoyable User Interfaces: Lessons from Computer Games. *CHI '82 Proceedings of the 1982 Conference on Human Factors in Computing Systems*, pages 63–68, 1981.

[Mar11] John Markoff. Computer Wins on 'Jeopardy!': Trivial, It's Not. *New York Times*, February 2011.

[McG11] Jane McGonigal. Reality Is Broken, 2011.

[Mic13a] Pietro Michelucci. Introduction. In Pietro Michelucci, editor, *Handbook of Human Computation*, pages xxxvii–xxxix. Springer Science & Business Media, 1 edition, December 2013.

[Mic13b] Pietro Michelucci. Synthesis and Taxonomy of Human Computation. In Pietro Michelucci, editor, *Handbook of Human Computation*, chapter 1, pages 83–86. Springer Science & Business Media, 1 edition, December 2013.

[MNBD06] Cameron Marlow, Mor Naaman, Danah Boyd, and Marc Davis. HT06, Tagging Paper, Taxonomy, Flickr, Academic Article, ToRead. *HYPERTEXT '06 Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 31–40, August 2006.

[O'M18] James O'Malley. Captcha if you can: how you've been training AI for years without realising it. *World of tech*, January 2018.

[OSvZ09] Simon Overell, Börkur Sigurbjörnsson, and Roelof van Zwol. Classifying Tags Using Open Content Resources. *WSDM '09 Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 64–73, 2009.

[PCI10] Gabriele Paolacci, Jesse Chandler, and Panagiotis G. Ipeirotis. Running experiments on Amazon Mechanical Turk. *Judgment and Decision Making*, 5(5):411–419, August 2010.

[Proa] ARTigo Project. ARTigo Taboo Tagging Game. `https://www.artigo.org/tabooTaggingGame.html`. Retrieved 2019-01-10.

[Prob] ARTigo Project. ARTigo Tagging Game. `https://www.artigo.org/taggingGame.html`. Retrieved 2019-01-10.

[Proc] ARTigo Project. Combino. `https://www.artigo.org/combino.html`. Retrieved 2019-01-10.

[RAPM04] Paul Rayson, Dawn Archer, Scott Piao, and Tony McEnery. The UCREL Semantic Analysis System. *Beyond Named Entity Recognition Semantic Labeling for NLP Tasks in LREC'04*, 2004.

[RBG+13] Jordan Raddick, Georgia Bracey, Pamela L. Gay, Chris J. Lintott, Carolin N. Cardamone, Phil Murray, Kevin Schawinski, Alexander S. Szalay, and Jan Vandenberg. Galaxy Zoo: Motivations of Citizen Scientists. *Astronomy Education Review*, 12(1), December 2013.

[Sav12] Neil Savage. Gaining Wisdom from Crowds. *Communications of the ACM*, 55(3), March 2012.

[SE16] Babak Saleh and Ahmed Elgammal. Large-scale Classification of Fine-Art Paintings: Learning The Right Metric on The Right Feature. *International Journal for Digital Art History*, (2), October 2016.

[Sem18] Alana Semuels. The Internet Is Enabling a New Kind of Poorly Paid Hell. `https://www.theatlantic.com/business/archive/2018/01/amazon-mechanical-turk/551192/`, January 2018. Retrieved 2018-08-29.

[Shn04] Ben Shneiderman. Designing for Fun: How Can We Design User Interfaces to Be More Fun? *Interactions - Funology*, 11(5):48–50, September + October 2004.

[Tho10] Clive Thompson. What Is I.B.M.'s Watson? *New York Times*, June 2010.

[TW06] Jennifer Trant and Bruce Wyman. Investigating social tagging and folksonomy in art museums with steve.museum. January 2006.

[UCI+06] Victoria Uren, Philipp Cimiano, José Iria, Siegfried Handschuh, Maria Vargas-Vera, Enrico Motta, and Fabio Ciravegna. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, pages 14–28, 2006.

[vA] Luis von Ahn. Luis von Ahn. `https://www.cs.cmu.edu/~biglou/`. Retrieved 2018-08-23.

[vA06] Luis von Ahn. Games with a Purpose. *Computer*, 39(6):92–94, June 2006.

[vAB05] Luis von Ahn and Manuel Blum. *Human Computation*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburg, PA 15213, December 2005.

[vABL04] Luis von Ahn, Manuel Blum, and John Langford. Telling Humans and Computers apart automatically. *Communications of the ACM*, 47(2):57–60, 2004.

[vAD04] Luis von Ahn and Laura Dabbish. Labeling Images with a Computer Game. *CHI '04 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 319–326, April 2004.

[vAD08] Luis von Ahn and Laura Dabbish. Designing Games with a Purpose. *Communications of the ACM*, 51(8):58–67, August 2008.

[vAKB06] Luis von Ahn, Mihir Kedia, and Manuel Blum. Verbosity: A Game for Collecting Common-Sense Facts. *CHI '06 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1:75–78, April 2006.

[vALB06] Luis von Ahn, Ruoran Liu, and Manuel Blum. Peekaboom: A Game for Locating Objects in Images. *CHI '06 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 55–64, April 2006.

[vAMM+08] Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *SCIENCE*, 321:1465–1468, September 2008.

[WBBL13] Christoph Wieser, François Bry, Alexandre Bérard, and Richard Lagrange. ARTigo: Building an Artwork Search Engine With Games and Higher-Order Latent Semantic Analysis. *Association for the Advancement of Artificial Intelligence*, November 2013.

[Wel08] Terri Wells. Harnessing Video Game Power for Good. `https://images.devshed.com/dh/stories/Harnessing_Video_Game_Power_for_Good/VG4G05.jpg`, June 2008. Retrieved 2018-08-30.