Building a Semantic Search Engine with Games and Crowdsourcing

Dissertation

zur Erlangung des akademischen Grades des
Doktors der Naturwissenschaften
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

von
Christoph Wieser

6. März 2014

Erstgutachter:	Prof. Dr. François Bry
	(Ludwig-Maximilians-Universtät München)
Zweitgutachter:	Prof. Dr. Klaus U. Schulz
	(Ludwig-Maximilians-Universtät München)
Drittgutachter:	Dr. Georg Groh
	(Technische Universtät München)
Tag der mündlichen Prüfung:	8. Mai 2014

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Wieser, Christoph

Name, Vorname

München, 6. März 2014

Ort, Datum

Unterschrift Doktorand/in

Formular 3.2

Abstract

Semantic search engines aim at improving conventional search with semantic information, or meta-data, on the data searched for and/or on the searchers. So far, approaches to semantic search exploit characteristics of the searchers like age, education, or spoken language for selecting and/or ranking search results. Such data allow to build up a semantic search engine as an extension of a conventional search engine. The crawlers of well established search engines like Google, Yahoo! or Bing can index documents but, so far, their capabilities to recognize the intentions of searchers are still rather limited. Indeed, taking into account characteristics of the searchers considerably extend both, the quantity of data to analyse and the dimensionality of the search problem. Well established search engines therefore still focus on general search, that is, "search for all", not on specialized search, that is, "search for a few".

This thesis reports on techniques that have been adapted or conceived, deployed, and tested for building a semantic search engine for the very specific context of artworks [BKW11]. In contrast to, for example, the interpretation of X-ray images, the interpretation of artworks is far from being fully automatable. Therefore artwork interpretation has been based on Human Computation, that is, a software-based gathering of contributions by many humans.

The approach reported about in this thesis first relies on so called Games With A Purpose, or GWAPs, for this gathering: Casual games provide an incentive for a potentially unlimited community of humans to contribute with their appreciations of artworks. Designing convenient incentives is less trivial than it might seem at first. An ecosystem of games is needed so as to collect the meta-data on artworks intended for. One game generates the data that can serve as input of another game. This results in semantically rich meta-data that can be used for building up a successful semantic search engine. Thus, a first part of this thesis reports on a "game ecosystem" specifically designed from one known game and including several novel games belonging to the following game classes: (1) Description Games for collecting obvious and trivial meta-data, basically the well-known ESP (for extra-sensorial perception) game of Luis von Ahn, (2) the Dissemination Game Eligo generating translations, (3) the Diversification Game Karido [SWKB11] aiming at sharpening differences between the objects, that is, the artworks, interpreted and (3) the Integration Games Combino, Sentiment and TagATag that generate structured meta-data [BW12a, BW12b].

Secondly, the approach to building a semantic search engine reported about in this thesis relies on Higher-Order Singular Value Decomposition (SVD). More precisely, the data and meta-data on artworks gathered with the afore mentioned GWAPs are collected in a tensor, that is a mathematical structure generalising matrices to more than only two dimensions, columns and rows. The dimensions considered are the artwork descriptions, the players, and the artwork themselves. A Higher-Order SVD of this tensor is first used for noise reduction in manner of a so called Latent Semantic Analysis (LSA).

This thesis reports also on deploying a Higher-Order LSA. The parallel Higher-Order SVD algorithm applied for the Higher-Order LSA and its implementation has been validated on an application related to, but independent from, the semantic search engine for artworks striven for: image compression. This thesis reports on the surprisingly good image compression which can be achieved with Higher-Order SVD. While compression methods based on matrix SVD for each color, the approach reported about in this thesis relies on one single (higher-order) SVD of the whole tensor. This results in both, better quality of the compressed image and in a significant reduction of the memory space needed.

Higher-Order SVD is extremely time-consuming what calls for parallel computation. Thus, a step towards automatizing the construction of a semantic search engine for artworks was parallelizing the higher-order SVD method used and running the resulting parallel algorithm on a super-computer. This thesis reports on using Hestenes' method and R-SVD for parallelising [SWB12] the higher-order SVD. This method is an unconventional choice which is explained and motivated. As of the super-computer needed, this thesis reports on turning the web browsers of the players or searchers into a distributed parallel computer. This is done by a novel specific system and a novel implementation of the MapReduce data framework to data parallelism [LWB13]. Har-

nessing the web browsers of the players or searchers saves computational power on the server-side. It also scales extremely well with the number of players or searchers because both, playing with and searching for artworks, require human reflection and therefore results in idle local processors that can be brought together into a distributed super-computer.

Zusammenfassung

Semantische Suchmaschinen dienen der Verbesserung konventioneller Suche mit semantischen Informationen, oder Metadaten, zu Daten, nach denen gesucht wird, oder zu den Suchenden. Bisher nutzt Semantische Suche Charakteristika von Suchenden wie Alter, Bildung oder gesprochene Sprache für die Auswahl und/oder das Ranking von Suchergebnissen. Solche Daten erlauben den Aufbau einer Semantischen Suchmaschine als Erweiterung einer konventionellen Suchmaschine. Die Crawler der fest etablierten Suchmaschinen wie Google, Yahoo! oder Bing können Dokumente indizieren, bisher sind die Fähigkeiten eher beschränkt, die Absichten von Suchenden zu erkennen. Tatsächlich erweitert die Berücksichtigung von Charakteristika von Suchenden beträchtlich beides, die Menge an zu analysierenden Daten und die Dimensionalität des Such-Problems. Fest etablierte Suchmaschinen fokussieren deswegen stark auf allgemeine Suche, also „Suche für alle“, nicht auf spezialisierte Suche, also „Suche für wenige“.

Diese Arbeit berichtet von Techniken, die adaptiert oder konzipiert, eingesetzt und getestet wurden, um eine semantische Suchmaschine für den sehr speziellen Kontext von Kunstwerken aufzubauen [BKW11]. Im Gegensatz beispielsweise zur Interpretation von Röntgenbildern ist die Interpretation von Kunstwerken weit weg davon gänzlich automatisiert werden zu können. Deswegen basiert die Interpretation von Kunstwerken auf menschlichen Berechnungen, also Software-basiertes Sammeln von menschlichen Beiträgen.

Der Ansatz, über den in dieser Arbeit berichtet wird, beruht auf sogenannten „Games With a Purpose“ oder GWAPs die folgendes sammeln: Zwanglose Spiele bieten einen Anreiz für eine potenziell unbeschränkte Gemeinde von Menschen, mit Ihrer Wertschätzung von Kunstwerken beizutragen. Geeignete Anreize zu entwerfen in weniger trivial als es zuerst scheinen mag. Ein Ökosystem von Spielen wird benötigt, um Metadaten gedacht für Kunstwerke zu sammeln. Ein Spiel erzeugt Daten, die als Eingabe für ein anderes Spiel dienen können. Dies resultiert in semantisch reichhaltigen Metadaten, die verwendet werden können, um eine erfolgreiche Semantische Suchmaschine aufzubauen. Deswegen berichtet der erste Teil dieser Arbeit von einem „Spiel-Ökosystem“, entwickelt auf Basis eines bekannten Spiels und verschiedenen neuartigen Spielen, die zu verschiedenen Spiel-Klassen gehören. (1) Beschreibungs-Spiele zum Sammeln offensichtlicher und trivialer Metadaten, vor allem dem gut bekannten ESP-Spiel (Extra Sensorische Wahrnehmung) von Luis von Ahn, (2) dem Verbreitungs-Spiel Eligo zur Erzeugung von Übersetzungen, (3) dem Diversifikations-Spiel Karido [SWKB11], das Unterschiede zwischen Objekten, also interpretierten Kunstwerken, schärft und (3) Integrations-Spiele Combino, Sentiment und Tag A Tag, die strukturierte Metadaten erzeugen [BW12a, BW12b].

Zweitens beruht der Ansatz zum Aufbau einer semantischen Suchmaschine, wie in dieser Arbeit berichtet, auf Singulärwertzerlegung (SVD) höherer Ordnung. Präziser werden die Daten und Metadaten über Kunstwerk gesammelt mit den vorher genannten GWAPs in einem Tensor gesammelt, einer mathematischen Struktur zur Generalisierung von Matrizen zu mehr als zwei Dimensionen, Spalten und Zeilen. Die betrachteten Dimensionen sind die Beschreibungen der Kunstwerke, die Spieler, und die Kunstwerke selbst. Eine Singulärwertzerlegung höherer Ordnung dieses Tensors wird zuerst zur Rauschreduktion verwendet nach der Methode der sogenannten Latenten Semantischen Analyse (LSA).

Diese Arbeit berichtet auch über die Anwendung einer LSA höherer Ordnung. Der parallele Algorithmus für Singulärwertzerlegungen höherer Ordnung, der für LSA höherer Ordnung verwendet wird, und seine Implementierung wurden validiert an einer verwandten aber von der semantischen Suche unabhängig angestrebten Anwendung: Bildkompression. Diese Arbeit berichtet von überraschend guter Kompression, die mit Singulärwertzerlegung höherer Ordnung erzielt werden kann. Neben Matrix-SVD-basierten Kompressionsverfahren für jede Farbe, beruht der Ansatz wie in dieser Arbeit berichtet auf einer einzigen SVD (höherer Ordnung) auf dem gesamten Tensor. Dies resultiert in beidem, besserer Qualität von komprimierten Bildern und einer signifikant geringeren des benötigten Speicherplatzes.

Singulärwertzerlegung höherer Ordnung ist extrem zeitaufwändig, was parallele Berechnung verlangt. Deswegen war ein Schritt in Richtung Aufbau einer semantischen Suchmaschine für Kunstwerke eine Parallelisierung der verwendeten SVD höherer Ordnung auf einem Super-Com-

puter. Diese Arbeit berichtet vom Einsatz der Hestenes'-Methode und R-SVD zur Parallelisierung [SWB12] der SVD höherer Ordnung. Diese Methode ist eine unkonventionell Wahl, die erklärt und motiviert wird.

Ab nun wird ein Super-Computer benötigt. Diese Arbeit berichtet über die Wandlung der Webbrowser von Spielern oder Suchenden in einen verteilten Super-Computer. Dies leistet ein neuartiges spezielles System und eine neuartige Implementierung des MapReduce Daten-Frameworks für Datenparallelismus [LWB13]. Das Einspannen der Webbrowser von Spielern und Suchenden spart server-seitige Berechnungskraft. Ebenso skaliert die Berechnungskraft so extrem gut mit der Spieleranzahl oder Suchenden, denn beides, Spiel mit oder Suche nach Kunstwerken, benötigt menschliche Reflektion, was deswegen zu ungenutzten lokalen Prozessoren führt, die zu einem verteilten Super-Computer zusammengeschlossen werden können.

Acknowledgments

Foremost, I want to thank my advisor Prof. Dr. *François Bry*. He gave me enormous freedom and inspired me with confidence. During the development of this thesis he exactly knew, which kind of motivation was needed in which stage. His skills and keen wit were the motor of many results.

Dr. *Norbert Eisinger* put heart and soul into university education. The refined and comprehensive knowledge of a vast number of students traces back to him. Dr. *Norbert Eisinger* took time for fruitful and thorough discussions even on very busy days. Giving me advice and hints in his modest manner was priceless.

The unofficial German word "Doktorvater" stands for doctoral advisor. Directly translated it means doctoral father. In this vein, *Fabian Kneißl* is my doctoral brother. I gained many insights into software development from him during the joined work on the ARTigo gaming platform used in both of our theses. More than once, he selflessly helped out in misery.

I want to express my gratitude to the academic and technical staff of the Institute for Informatics for giving precious advice and for offering a family-style atmosphere. Special thanks go to Ingeborg von Troschke, Elke Kroiß, and Martin Josko, who taught me so many technical skills.

The work on this thesis was enriched by the following students while I supervised their theses: *Alexandre Bérard*, *Diego Havenstein*, *Richard Lagrange*, *Philipp Langhans*, *Sebastian Mader*, *Philipp Schah*, *Oliver Schnuck*, *Bartholomäus Steinmayr*, and *Sebastian Straub*. I gratefully look back to working with them in the context of my thesis.

I am obliged to a group of researchers belonging to the project "Play4Science" funded by the Deutsche Forschungsgemeinschaft (DFG). *Dr. Gerhard Schön* developed the first version of the ARTigo platform. Further on, many aspects of the ARTigo platform were discussed with *Prof. Dr. Hubertus Kohle*, *Prof. Dr. Thomas Krefeld*, *Dr. Stephan Lücke*, *Dr. Christian Riepl*, and *Prof. Dr. Klaus Schulz*.

Finally, I thank my family for their unconditional love and continuous support. My parents *Ingrid* and *Raimund* provided me a worry less life and aroused my interest in science. My wife *Anne Katrin* strenuously encouraged me and covered my back whenever needed during the development of this thesis and took care of our beloved children *Konrad*, *Eva*, and *Lucia*.

Contents

Part I

Introduction

"Those who seek shall find" is a famous quotation [Sop91] from the tragedy *Oedipus Rex* by Sophocles. Search appears to have been quite a long-standing concern of mankind. The quotation refers to the search for the murderer of Oedipus' father, King Laios. This search is a *semantic* search because it presupposes an interpretation of the term "murderer". Indeed, humans find semantic search quite natural and are rather capable of this kind of search whereas they tend to be less inclined and less apt to do *syntactic* search like finding a name in an unordered list. This is in stark contrast to computers, which excel in syntactic search, thus facilitating human work tremendously. It is therefore not surprising that humans, realizing the ease with which computers perform syntactic search, strive for similarly quick and reliable computer support for semantic search as well. Unfortunately, semantic search has not exactly been a success of computers so far. For example, recognizing a murderer or even just a human face in an image, although an easy task for humans, is still rather difficult for computers.

This thesis contributes to improved computer support for semantic search, that is "to improve search accuracy by understanding searcher intent and the contextual meaning of terms" [Wik13h]. Note that "understanding", "meaning", and "intent" refer to a mapping of human semantic search to syntactic calculation models for computers. This mapping has been stressed in research on machine learning and consists of two components to simulate human interpretation abilities: *data acquisition* and *data analysis*. The use of these two components in machine learning is rather conventional, whereas their characteristics in this thesis are rather unconventional.

In this thesis, data acquisition (addressed in Part II) is done via games with a purpose (GWAPs) [VA06] and data analysis (addressed in Part III) takes into account *individual* human data acquired by these games. Analysing individual human data aims for better accuracy compared to analysing aggregated data as used in machine learning for saving computational efforts. The semantic search engine ARTigo (www.artigo.org) described in Part II of this thesis retrieves (images of) artworks. Nevertheless, the proposed techniques are also applicable to retrieving other items than images. First experiments point to the usefulness of the approach for semantic search. An elaborated statistically significant validation was beyond the scope of this thesis, but obviously ought to be done as part of future work. Part IV describes further perspectives for analysing the ARTigo data.

Contemporary Web search engines like Google Search, Yahoo!, or Bing basically rely on a kind of syntactic search (beside additional undisclosed techniques). Thereby, answering a query means listing all text documents of a collection that contain certain words, for example, "murderer of Laios". Such lists of results can become really long depending on the size of the queried collection. For example, on the Web Google Search retrieved about 5.1 million results containing "murderer of Laios" (in October 2013). This huge amount of results demands an elaborate ranking to present documents of highest relevance first. Ranking, a core technique of which is PageRank [PBMW99], is a key competence of search engines and belongs to the well-kept company secrets.

In practice humans interpret highly ranked results in terms of their relevance. It is this interpretation *a posteriori* by humans that adds all the semantics to syntactically retrieved results. Or rather, *almost* all the semantics: certain restricted forms of semantic search are supported by techniques already being used by contemporary search engines. For example, Google Search retrieved about 81 thousand results when searching for "killer of Laios" (in October 2013) and the top results did not contain the word "killer" but the word "murderer". This kind of behaviour can be achieved with syntactic means such as synonym lists [BB99, Joh12] as well as more sophisticated techniques, but obviously there are limits to such syntactic means.

The approach proposed in this thesis differs from the approaches of contemporary Web search engines in two major aspects.

First, search in this thesis relies on the analysis of items by humans and not on an automatic analysis by computers as performed by Web crawlers and indexers for Web search engines. While text items such as Web pages can be analysed relatively easily by computers, the analysis of images still remains a challenge for computers. For example, suitable metadata (data about items) for Figure 1 are "man" and "sphinx", which is quite easy to recognize for humans but not for computers. Image analysis is still subject to active research in pattern recognition. In this respect the proposed approach is well suited for items that can be analysed by humans and not or not so easily by computers.

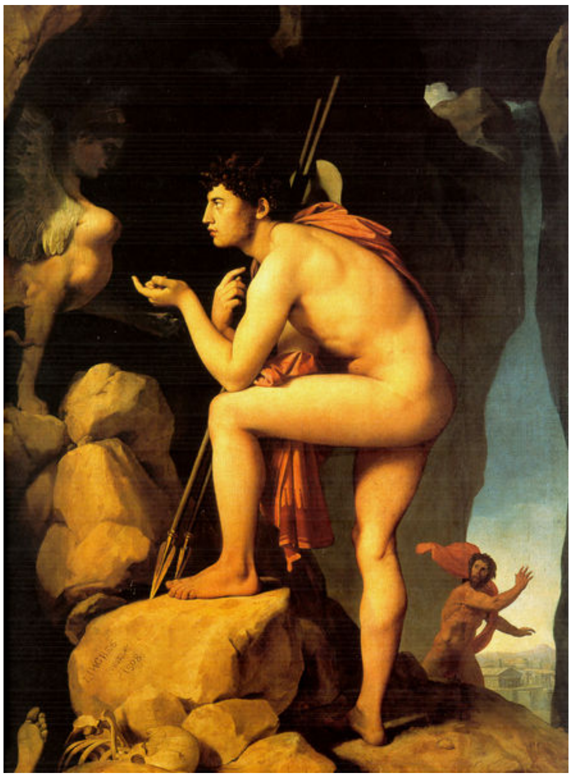


Figure 1: J. Ingres, Oedipus and the Sphinx, 1808, Paris, Musée du Louvre.

Second, the approach differs from Web search engines because metadata for one item usually originates from different sources. This is natural for games with a purpose where different humans play games leaving footprints that provide data for the items seen during the game. However, it is unnatural for Web search engines collecting data from a single source, the item itself.

Search based on various human impressions acquired with games, as opposed to an objective analysis of items by computers, can be a boon and a bane.

On the one hand humans can make use of a lot of information that computers cannot currently access. For example, humans can make implicit content of an item explicit. In Figure 1 another suitable metadata is “conversation”, which goes way beyond pattern recognition. Furthermore, humans are capable of adding supplemental information like “Oedipus” being the depicted “man” who is the “murderer of Laios” in the tragedy, which would definitely be hard to derive for computers.

On the other hand humans may submit wrong information, consciously or not, or simply use different terminology. For example, Figure 1 can prompt wrong information like “Egypt” because of the depicted “sphinx”. In experiments by Cleverdon, as reported in [DVGV12], the overlap of the index terms was only 60% for two groups of people that constructed thesauri in a particular subject area. Hence among the challenges for this thesis is to propose games that acquire *accurate* data (cf. Part II) and to *filter out* inaccurate data (cf. Part III).

This introductory part of the thesis is arranged as follows. Chapter 1 addresses human computation as broader context of games with a purpose. Subsequently Chapter 2 refers to approaches to semantic search engines conceived so far. Both chapters prepare the contributions of this thesis in Parts II and III.

Chapter 1

Human Computation (HC)

"Computer" in its original meaning denoted a human solving arithmetic problems [Gri05, LA11]. Before automated computers were invented, problems like the precise trajectory of Halley's Comet or the ballistic trajectory of a projectile had to be calculated by hand. Later on, humans specified problems and let automated computers calculate solutions [Tur50].

Nowadays, *human computation* is an approach involving humans in (algorithmic) problem solving from the computer's perspective. In other words, problem solving is getting (partially) outsourced from computers back to humans. At a first glance, outsourcing computational problems to humans might seem paradox because computers outperform humans in many respects ranging from solving arithmetics to memorising huge amounts of data. However, humans gain advantage over computers, e.g., in image recognition and interpretation. A natural step is combining both, the strengths of computers and the strengths of humans to achieve higher quality results while saving time.

Human computation is related to the more general term *crowdsourcing* [How06, QB11], denoting harnessing a large anonymous group of people for problem solving. Notably, crowdsourcing does not necessarily rely on computers and can proceed completely offline. The important word in the explanation of crowdsourcing is "large" focusing on available manpower. One of the earliest examples of crowdsourcing is a public call to contribute to an English dictionary, nowadays known as the "Oxford English Dictionary" [Lan11]. Contributors sent definitions of English words and after 70 years the dictionary could be completed.

In many applications, like the ARTigo platform described in this thesis, the combination of automated computation, human computation, and crowdsourcing is needed to solve a problem like building a semantic search engine (cf. JSMapReduce in Section 10.4).

Major challenges of this thesis for data acquisition (cf. Part II) as well as for data analysis (cf. Part III) arise from human computation, which is briefly introduced in this chapter. A subtopic of human computation is the field of games with a purpose. Law and von Ahn contributed to an initial declaration of human computation as a research field [LA11]. This introductory chapter is based on many of their findings as well as their overview of human computation (HC) [LA11].

1.1 Applications of HC

Human computation is a rather young field of research getting apparent by the public and not only by researchers and techies. According to [LA11], various applications of human computation exist today: games with a purpose, crowdsourcing marketplaces, and identity verification.

Games With a Purpose

A Game With A Purpose, as first defined by Luis von Ahn [VA06], is a human-based computation technique in which a computational process performs its function thanks to the intervention of humans in an amusing way that is in a game. The first GWAP (game with a purpose) proposed by Luis von Ahn was the ESP Game [VAD04], whose purpose was tagging images, sometimes denoted as image labeling game. ESP stands for extrasensory perception. Beyond the ESP game, a wide range of GWAPs was published such as FoldIt[1] for predicting protein structures. A typology of human computation games by Pe-Than et al. [PTGL13] reflects the current state of the art of GWAPs. Crucial for the success of a GWAP is the motivation of the players.

Crowdsourcing Marketplaces

On the Web, platforms like the Amazon Mechanical Turk[2] offer a marketplace for crowdsourcing tasks. *Requesters* can submit so called human intelligence tasks (HITs) that are offered by the Mechanical Turk platform to *workers*. Attending to typically simple HITs is usually rewarded with a small amount of money like some cents. An impressive (but unpaid) crowdsourcing application was the project "Help find Jim Gray". In January 2007, the famous computer scientist Jim Gray and his boat were reported missing during a sailing trip. Satellite images of the suspect area were ported on Amazon Mechanical Turk and suspicious objects looking like a sailing boat could be reported by workers. About 12.000 users participated and scanned thousands of images. Apparently, none of the images showed Jim Gray's boat.

Identity Verification

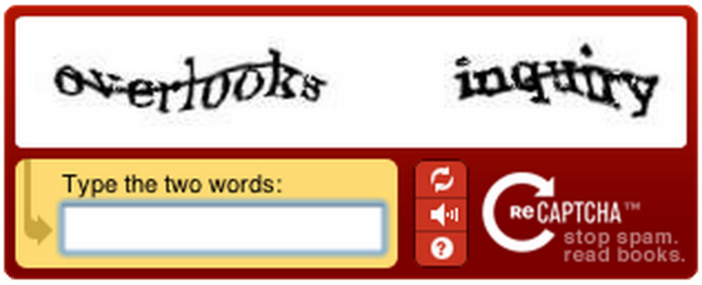


Figure 1.1: CAPTCHA: Telling Humans and Computers Apart Automatically.

Checking whether a client is human or not is important for many Web sites offering means for interaction. Such Web sites are often referred to as belonging to the Web 2.0 [O'R05]. Posting comments is a typical example for such Web sites that can be used by humans and abused by automated spamming software, which posts advertisements rather than constructive comments. Filtering out automated postings belongs to the class of identity verification problems [LA11]. Maybe the first identity verification software is CAPTCHA [VABHL03] telling human and computer clients apart. CAPTCHA exploits the advantage of humans over computers to recognize obfuscated letters building words as depicted in Figure 1.1[3]. For computers this is an optical character recognition (OCR) problem. While humans can relatively easily recognize obfuscated letters and enter them in cleartext, OCR software designed for recognizing normal, non-obfuscated characters tends to struggle here. The subsequent version of CAPTCHA called reCAPTCHA [VAMM$^+$08] asks to enter two words instead of one (cf. Figure 1.1). While one word is known to reCAPTCHA and can be used for validating the input of a client, the other one might be unknown. Unknown words may be scans of books. CAPTCHA as well as reCAPTCHA can be used to digitize scans while the identity of a client – human or not – is determined. For instance, 129 year's issues of the New York Times were digitized within two years [Sca08].

[1]http://www.fold.it/
[2]https://www.mturk.com/
[3]http://www.captcha.net/

1.2 Challenges of HC

The applications mentioned in the previous section indicate that harnessing humans for solving problems has certain advantages. However, two major challenges arise when working with humans. First, humans need an *incentive* to participate in human computation. Second, when participating the *accuracy* of the commitment needs to be ensured. Both, providing an incentive and ensuring accuracy are the conditions of successful human computation.

Incentives for Humans to Participate in HC

Quinn and Bederson [QB11] discovered several kinds of motivations for humans to participate in human computation. All of them are relevant for the ARTigo platform.

- *Getting paid* is a very simple motivation. Basically, the working world aside from volunteering is run by financial rewards. The above mentioned crowdsourcing platform Amazon Mechanical Turk uses this motivation. Financial rewards were tested without success on the ARTigo platform because only few players competed for the money.
- *Altruism* is another motivation that can be found to participate in human computation, especially in scientific projects like ARTigo. Involving people in science is a promising approach called citizen science [Han10]. Apart from applications of human computation in science, the search for Jim Gray mentioned above is another example for altruism.
- *Enjoyment* is most likely the highest motivation to play games with a purpose as provided by the ARTigo platform because it is a strong incentive for humans. However, turning computational tasks into games can be a difficult task [VAD08].
- *Reputation* as motivation is probably a subordinate motivation on the ARTigo platform. The highscore list, which presents the best players, may reflect reputation to some extent. Reputation is not explicitly exploited as incentive in the ARTigo platform apart from that.
- Doing *implicit work* is the standard case for all games of the ARTigo platform. All of those games acquire data for the semantic search implicitly. However, doing implicit work is most likely no motivation for the players of the ARTigo platform because search functionality is less used compared to the games.

Further motivations being relevant for the ARTigo platform are listed in the Wikipedia article "Human-based computation" [Wik13d]:

- Players *participate in the result* of their efforts because the tags are used as basis for image search. This is the basic motivation not for players but for running the ARTigo platform.
- In the context of academic research, dealing with images for getting acquainted with art and art history or in other words *education* is an incentive for students to play the games of the ARTigo platform.

Accuracy and Efficiency

In human computation, accuracy [LA11] is mainly influenced by subjective decisions, mistakes, and abuse by humans, e.g., when labeling images. A reason for subjective decisions while tagging an image may be, e.g., the educational background. A typical mistake is misspelling a tag. A reason for abuse may be advertising.

In contrast, accuracy in computing typically refers to coping with irrational numbers, which can only be approximately represented by computers. Such approximations can lead to error-prone or inaccurate results.

In this thesis ensuring accuracy and efficiency is addressed by the game design as a first step (cf. Part II) and data analysis as a second step (cf. Part III).

1.3 Designing HC Systems

Law and von Ahn [LA11] identify three aspects of human computation systems that need to be taken into account:"who", "what", and "how" as means to apply explicit control instead of natural dynamics like coordination or competition. Figure 1.2 visualizes these aspects. In the following, these three aspects are briefly explained. Law and Ahn state research questions (see highlighted colored boxes) referring to these aspects. As the ARTigo platform can be seen as a human computation system, the research questions are answered according to the implementation of the ARTigo platform as well as according to approaches proposed in this thesis. It is not claimed that the answers are general-purpose answers, solving all problems in the research field of human computation. Chapter 2 of [LA11] strives for a first characterization of human computation algorithms, which is based on the definition of algorithms by Knuth [Knu73]. For the sake of simplicity, the term "operation" is informally used to denote computation in absence of a formal definition of human computation algorithms.

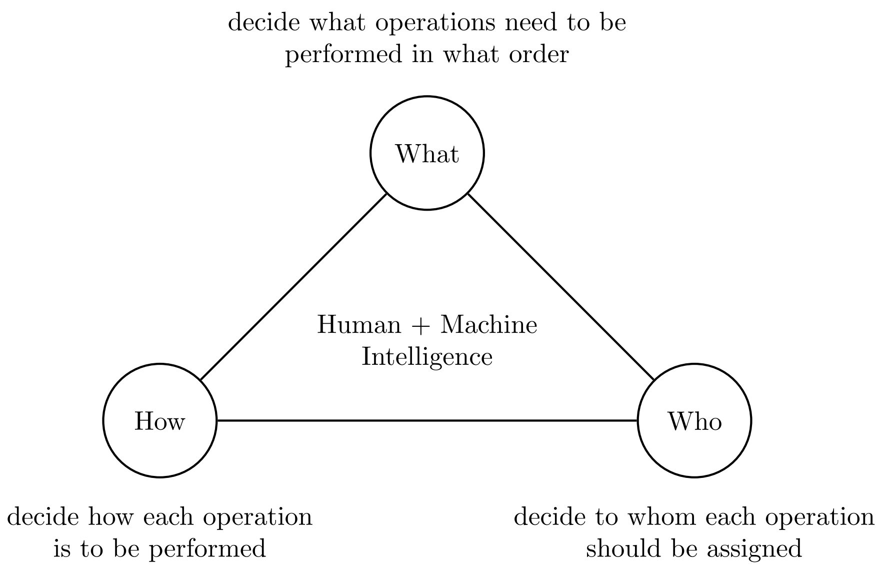

decide what operations need to be performed in what order
What
Human + Machine Intelligence
How
Who
decide how each operation is to be performed
decide to whom each operation should be assigned

Figure 1.2: Three central aspects of human computation systems [LA11].

"What" Aspect

The "What" aspect addresses declaratively what operations need to be executed in a human computation system. This includes also the flow of operations. This aspect is independent from the concrete implementation and whether operations are executed by humans or computers.

> "What tasks can be performed adequately by machines, therefore eliminating the need for human involvement?" [LA11]

The ARTigo platform cedes all operations except human image perception to the computer. Basically, the lacking image perception capabilities of computers [vA13] are the only reason for the development of the ARTigo platform.

> "Can we leverage the complementary abilities of both humans and machines to make computation more accurate and efficient?" [LA11]

This is a goal of the ARTigo platform. Humans bring in image perception capabilities, while computers take over the organization of data acquisition (cf. Part II). The games of the ARTigo platform strive for a high accuracy based on human computation, while the computer subtracts out inaccurate data (cf. Part III).

Both, humans and computers complement each other in the gaming ecosystem (cf. Chapter 5): The computer chooses images for games. For instance, description games (cf. Section 5.1) as entrance of the gaming ecosystem get least tagged images, whereas diversification games (cf. Section 5.2) get similarly tagged images assigned.

A novel technique of leveraging the abilities of computers is using the computers of *website users* (for instance playing online games) as grid. A grid [FK03] is a "collection of computer resources from multiple locations to reach a common goal" [Wik13b]. The common goal in this thesis is a parallel computation of the data analysis proposed in Part III. The term grid computing was created a decade ago and distributed computation projects such as SETI@home[4] started shortly thereafter. The novelty is that the Web browser is used for distributed computation (cf. Section 10.4). Used in crowdsourcing platforms, human and automated computation form a computational unit that is obviously capable of both solving human computation problems and automated computation. Notably, the approach of building a semantic search engine with games and crowdsourcing proposed in this thesis scales because the computational power for data analysis purposes increases with the amount of data submitted by the users.

"How do we decompose complex tasks into manageable units of computation and order them in such a way to handle the idiosyncrasy of human workers?" [LA11]

The complex task concerning human computation in this thesis is acquiring comprehensive descriptions of images consisting of various kinds of tags (cf. Chapter 4). Acquiring such tags is decomposed in different kinds of games like description games and diversification games among others in the Gaming Ecosystem (cf. Chapter 5).

"How do we aggregate noisy and complex outputs from multiple human computers in the absence of ground truth?" [LA11]

Aggregating noisy output is addressed in Part III by an approach for noise reduction via Higher-Order Singular Decomposition (HO-SVD). HO-SVD is a generalisation of matrix Singular Value Decomposition (SVD) accepting tensors (multidimensional matrices) as input. Hence, in addition to tags (terms) and documents, users can be taken into account. This noise reduction technique is the core of Latent Semantic Analysis (LSA) [DDF$^+$90] aiming at statistically determining ground truth. The novel Higher-Order Latent Semantic Analysis proposed in this thesis is on one hand meant to determine observed ground truth with higher accuracy than to standard LSA where the output of all users is aggregated (summarized) to a matrix. On the other hand Higher-Order Latent Semantic Analysis is meant to calculate personalized results for Semantic Search. As already mentioned, the validation of this approach is not part of this thesis.

"Who" Aspect

Operations defined in the "What" aspect need to be assigned to humans. In traditional distributed computing, parameters like network and CPU speed can be taken into account for decisions on distributing tasks, parameters that are irrelevant for human computation. Factors in human computation like expertise to ensure accurate output are hard to grasp. The ARTigo project does not implement a strategy for the selection of players. Based on the data collected so far with the ARTigo platform, a classification of players looks promising to choose effective selection of game partners.

[4] http://setiathome.ssl.berkeley.edu/

"What are some effective algorithms and interfaces (e.g., search or visualization) for routing tasks?" [LA11]

Routing is done implicitly in the Gaming Ecosystem (cf. Chapter 5). Each game type selects images or artworks according to a certain strategy. The game Eligo (cf. Section 5.4), e.g., selects images having tags assigned, that are not translated to another language so far. Explicit flow control in terms of controlling the sequence of games ought to be done as part of future work.

"How do we model the expertise of workers, which may be changing over time?" [LA11]

ARTigo draws on observed ground truth as mentioned above. Over the time an observation or opinion of a human can change. For example, a student of Art History will improve his knowledge about arts resulting in another tagging behaviour over the years. The taggings might reflect the increasing knowledge of the student. This can be modelled using an additional mode[5] for the tensor data structure. Instead of a 3rd-order tensor modelling (tag, image, user) triples, a 4th-order modelling also the time can be used: (tag, image, user, time). This integration is compatible with the data analysis approach (cf. Part III) but of course the complexity of a data analysis will increase.

"What are some optimal strategies for allocating tasks to workers, if their availability, expertise, interests, competence and intents are known versus unknown?" [LA11]

In future, the selection of two players for a game by forming similar pairs (for instance pairs of experts) might improve the efficiency of games such as input agreement games. A selection of players might be done on basis of the data analysis approach reported about in Part III via a similarity measure. This technique might allow to cluster players of the same kind. This might be beneficial for input agreement games because players of the same cluster might input tags of the same kind. For instance, two art historians might contribute tags reflecting their knowledge.

"How" Aspect

After "What" and "Who", the "How" aspect is devoted to the interface for humans that are expected to solve problems. This can be a jazzy game as well as a mundane working environment. Obviously, the design of the interface affects the usability of a human computation system and therefore contributes to "truthful, accurate, and efficient" [LA11] results.

"How do we motivate people to have a long-term interaction with the system, by creating an environment that meets their particular needs (e.g., to be entertained, to have a sense of accomplishment or to belong to a community)?" [LA11]

The main motivation of the ARTigo platform is enjoyment, which is exploited in the games of the platform (cf. Chapter 5). Further motivations are mentioned above in Section 1.2.

The community aspect is not focus of this thesis, but obviously ought to be addressed as part of future work. Modules preparing the implementation of community software were conceived such as means for creating own exhibitions or collections of images that can be shared and discussed with other players.

[5] A 3rd-order tensor has 3 modes and is a "cubic matrix". The term dimensions is avoided because it collides with denoting the number of matrix columns as dimensions in information retrieval.

"How do we design game mechanisms that incentivize workers to tell the truth, i.e., generate accurate outputs?" [LA11]

The games of the ARTigo platform draw on input agreements as in the ESP game [VA06]. A modified input agreement strategy is used by the Karido game in Section 5.2 where two players slip into complementary roles (describer and guesser). Another game using complementary roles ("positive" and "negative") is Polarity [LSS+11] described in Chapter 3.

"What are some new markets, organizational structures or interaction models for defining how workers relate to each other (as opposed to working completely independently)?" [LA11]

This question is beyond the scope of this thesis, but obviously ought to be investigated as part of future work.

Chapter 2

Semantic Search Engine

A semantic search engine is a software system that is conceived to answer semantic queries. However, specifying such semantic queries, e.g., using Semantic Web technology like SPARQL (cf. Section 2.1) is challenging even for experts. A complementary approach is "to improve search accuracy by understanding the searcher's intent and the contextual meaning of terms" [Wik13h]. The challenge for this complementary approach is to make "understanding", "meaning", and "intent" accessible to a software system.

Two (sometimes intertwined) approaches have been conceived so far. One approach is knowledge representation combined with reasoning and the other one is machine learning. Knowledge representation requires an explicit formalization of knowledge and expertise on its syntax, whereas machine learning is based on observations called training data that can be offered without expertise, e.g., by entering simple tags without any restrictions.

A major (and still active) area of research in the last decade was the Semantic Web [BLHL+01] which is based on knowledge representation research. So when reading "semantic search" a first association might be that the term refers to Semantic Web powered search (addressed in Section 2.1). Indeed Semantic Web search engines have evolved so far and even games with a purpose for the Semantic Web [SH08] have been proposed. However the focus of this thesis is semantic search based on machine learning.

Full-text Search on Metadata and its Limits

A syntactic (not semantic) approach for retrieving items is full-text search. Items are textual documents and can be retrieved based on words contained in the documents. For instance a query for "bow" will retrieve those documents of a collection that contain the string "bow". Full-text search is a complex field of research offering a lot of sophisticated methods, e.g., for ranking results or for matching word forms like plural or genitive.

Typically, full-text search is not applied on metadata such as tags but on data such as textual documents. Since images as items for retrieval do (usually) not contain text, metadata are a means for applying textual search on image data.

Using full-text search indirectly on metadata and not directly on data causes differences that a searcher should be aware of.

First, the completeness of metadata decides on the quality of search. Obviously, if appropriate metadata is missing an item cannot be found. This contrasts with retrieving textual documents directly. If a word is not contained in a document, the searcher does not expect that the document is retrieved.

Second, metadata is an interpretation of data. This indirection has advantages and disadvantages. On one hand an interpretation of data might offer a much more comprehensive access to metadata. For instance the name of an item's author might be part of the metadata but not mentioned in a document. On the other hand, interpretations may be subjective and error-prone.

Third, from the perspective of a searcher, retrieved items are supposed to fulfill his own interpretations. This is a semantic search engine as striven for in this thesis.

Basic challenges for semantic search on metadata as well as for search on data are, e.g., synonymy and homonymy.

Synonyms [Wik13l] denote words with similar meaning such as "broad" and "wide". A typical means o overcome the search problem posed by synonyms is using so called synonym lists for resolving synonyms.

Homonymy [Wik13c] is an even harder problem for retrieval because homonyms are syntactically identical but have different meanings. For example, the word "bow" can mean the front of a ship, a weapon, or a gesture of bending forward in respect. A common approach to differentiate homonyms is using the context such as other tags, for example "ship" as context for "bow".

Full-text search based on metadata is a first approach that can be applied for building a (syntactic) search engine using the tags acquired by the games described in Part II. Approaches for semantic search are addressed in the following sections.

2.1 The Semantic Web: a Description Logics Approach

The goal of the Semantic Web [BLHL$^+$01] is providing the World Wide Web (WWW) with "semantics". Semantic applications like semantic search as addressed in this thesis or an automated coordination of appointments are expected to become reality. Semantics in this context stands for a mapping of semantics (in the sense of human understanding) to a *predefined* syntax and rules for computing. In the following, a brief and in no way exhaustive introduction to the Semantic Web is given to make this section selfcontained.

Brief Introduction Into the Semantic Web

When explaining the Semantic Web bottom up, its foundation is metadata. Metadata connects the Semantic Web with the WWW. The formalism for expressing metadata is the Resource Description Framework (RDF) [KCM04], that uses a set of triples in *(subject,predicate,object)* style. For example, a triple for an artwork showing Oedipus and the Sphinx (as in Figure 1) could be `(figure1,depicts,sphinx)`. `figure1`, `depicts`, and `sphinx` are unique (abbreviated) identifiers that can refer to the WWW as well as to the Semantic Web, as statements about metadata itself. A set of RDF triples can be seen as serialization of a RDF graph, where *subjects* and *objects* are vertices and *predicates* are edges.

The mapping based on such, lets say, "RDF three word phrases" is pretty natural for humans. However, crucial for the Semantic Web is that users agree on using the predefined syntax in the same way. Syntax refers to the *structure* as well as to its instances like `sphinx`. For example, an item must be identified with (or mapped to) `sphinx` by all users. Note that an arbitrary identifier can be used such as a number instead of mnemonic identifiers like `sphinx`. Numbers however are much harder to grasp for humans and would hamper a mapping of syntax and (human) semantic.

The agreement of the commonly used syntax is formalized in ontologies written in RDF. For instance, the Web Ontology Language (OWL) [MVH$^+$04] among others allows to represent the facts that users agree on. This includes relations between items like `(sphinx,isA,demon)`. Note that both, `sphinx` and `demon` belonging to the Semantic Web and not representing Web pages. Such a triple could be part of an ontology about Greek mythology providing concepts additional to `sphinx` and `demon`. A survey of current approaches on information integration based on ontologies and their expressivity is given by Wache et al. [WVV$^+$01].

The Semantic Web based on RDF and OWL allows to infer additional RDF triples that can be mapped back to (human) semantics. Given the triples `(figure1,depicts,sphinx)` and `(sphinx,isA,demon)`, an algorithmic approach, called deductive reasoning, can infer the so far implicit fact `(Figure1,depicts,demon)`, that Figure 1 depicts a demon. This very simple example must not hide, that deductive reasoning is a powerful means to support human work well. On top of such simple examples, customized rules make deductive reasoning very flexible.

The SPARQL Protocol and RDF Query Language [PS+08] is a query language for RDF triples. With an SQL-like syntax graph patterns can be expressed. The evaluation of SPARQL queries yields a set of triples. The higher the reasoning capabilities of a SPARQL query engine, the more triples can be inferred. SPARQL can be seen as interface to Semantic Web-based search, but this field is larger as reported in the following section.

Semantic Web-based Search

Mäkelä [Mäk05] reports on the current state of the art in Semantic Web-based Search. He identifies five clusters of research directions. Most of the nine types of semantic search identified by Grimes [Gri10] can be assigned to these clusters. The remaining types belong to machine learning techniques (cf. Section 2.2) and not to Semantic Web Research. The following paragraphs summarize these five clusters identified by Mäkelä [Mäk05].

Augmenting traditional keyword search with semantic techniques follows a basic scheme: "first, the keywords are located in the ontology, then various other concepts are located through graph traversal, after which the terms related to those concepts are utilized to either broaden or constrain the search" [Mäk05]. An example for an ontology that is frequently used is WordNet[1] supplying applications with relations like synonyms or generalizations.

Basic concept location is a sub-problem of Augmenting Traditional Keyword Search with Semantic Techniques and addresses how instances like `Figure1` can be identified with concepts in ontologies like `sphinx` efficiently.

Complex constraint queries refers to expressing and evaluating queries on RDF graph data. Using, e.g., SPARQL as an interface for querying needs training in both, the query language and the underlying ontology. While evaluating queries can build on a long research tradition in database systems and logics, the expression of queries for unexperienced users is challenging.

Problem solving has been mentioned above as an example of inferring that Figure 1 depicts a demon, what was not explicitly stated before. According to Mäkelä [Mäk05] only basic usage of inference techniques for Semantic Web-based Search is deployed.

Connecting path discovery addresses finding connections between concept nodes in an RDF graph.

Status Quo of Semantic Web-based Search

Productive Search Engines for the Semantic Web are often based on full-text search on ontologies such as the Semantic Web Search Engine (SWSW)[2] or Watson[3]. Some Semantic Web Search Engines like for instance Sindice[4] allow SPARQL queries.

Shadbolt, Berners-Lee, and Hall observed in 2006 that "this [Semantic Web] simple idea, however, remains largely unrealized." [SHBL06]. Potential reasons for the challenges to realize a Semantic Web are given in [Wik13i]: In contrast to Search on the WWW, *Vastness* is a huge problem and in particular dealing with duplicated RDF statements. *Vagueness* refers to imprecise concepts like "young" or "tall" but even if concepts are precise their instances suffer from *uncertainty.* Statements in the Semantic Web can be contradictory leading to *Inconsistency* such that inference fails. Finally, *deceit* is a problem that is one reason for wrong data.

2.2 Machine Learning-based Semantic Search

In the previous section about the Semantic Web, sophisticated methods are introduced based on *explicit* specifications of the mapping between syntax and semantics. It turns out that humans and in particular a larger group of humans struggle with specifying and using the same mapping

[1]http://wordnetweb.princeton.edu/
[2]http://www.swse.org/
[3]http://watson.kmi.open.ac.uk/WatsonWUI/
[4]http://www.sindice.com/

of syntax to semantics. Challenges like vagueness, uncertainty, and inconsistency need to be approached for a successful Semantic Web as already mentioned.

Machine learning however is a "field of study that gives computers the ability to learn without being explicitly programmed" as stated by Samuel [Sim13]. Hence, dealing with *vagueness*, *uncertainty*, and *inconsistency* is usual in this field. Semantic search as proposed in this thesis is based on machine learning to approach these challenges. The remaining challenges of the Semantic Web reported in [Wik13i] are addressed using further approaches beside machine learning: Parallelization (cf. Chapter 10) is used to address *vastness* and games with a purpose are used to address *deceit* (cf. Part II).

Machine learning offers various approaches for searching, which is a branch in the large field of machine learning [Wik13g]. The type of input data restricts the selection of suitable methods in that field. In this thesis, the data collected by games with a purpose are so called "labelled examples". An item (or image) is labelled by an image labeling game (cf. Chapter 3). In machine learning such data are analysed with supervised learning [MRT12] techniques attempting "to generalise a function or mapping from inputs to outputs which can then be used to speculatively generate an output for previously unseen inputs" [Wik13k]. This is a difference to the field of Data Mining which focuses on "discovery of (previously) unknown properties on the data" [Wik13g].

The approach proposed in this thesis is based on a supervised learning technique called Latent Semantic Analysis (LSA) [DDF$^+$90] which is described in detail in Chapter 7 based on latent variable models. LSA is also referred to as Latent Semantic Indexing (LSI). According to Thomas Hofmann[5] such models "are quite useful in modeling and discovering hidden structure that often leads to 'semantic' data representations". In other words LSA calculates a mapping between syntax and semantics based on training data. In contrast to Semantic Web approaches, such training data may by vague, uncertain, and inconsistent. By means of LSA, the Latent Semantics as observed by the training data is computed. As consequence, no explicit mapping of syntax and semantic needs to be conceived by humans.

Latent Semantic Analysis as proposed by Deerwester et al. [DDF$^+$90] is based on matrix data. Rows of such a matrix model labels and columns model documents. A matrix entry determines how often a label was assigned to a document. This is a suitable data model for collections of documents. Games with a purpose however offer the source of assignments as well, the player who entered a label (or tag). Obviously, a matrix is not sufficient for representing individual users in addition to documents and labels, but only for the sum of assignments.

In the mathematical structure series of scalar, vector, and matrix, a tensor is the next step consisting of kind of matrices with more "dimensions" than just rows ans columns. The "dimensions" are called modes. With a third mode in addition two rows and columns, single users can be modelled in a sort of cuboid matrix, where document-term matrices –one for each user– are stacked. This integrated data structure requires a generalization of LSA which is the proposed approach called Higher-Order LSA for realizing Semantic Search based on training data acquired by Game With a Purpose in this thesis. The approach is developed in Part III.

Integrating LSA to Higher-Order LSA is not claimed to be the only approach to realize semantic search based on term–document–user training data as provided by games with a purpose.

Probabilistic LSA (pLSA) by Hofmann [Hof99] – being already successfully tested on automatic image labeling [MGP04] – has advantages over LSA in (1) model interpretation because pLSA models *conditional* probabilities for aspects and (2) no parameter guessing for the level of reduction like in LSA (cf. Chapter 7) is needed. pLSA is based on a latent variable model[6], called aspect model, for co-occurrence data. The model is fitted by an iterative expectation maximization (EM) algorithm (unlike LSA) starting with a random distribution. Thus, the result of the EM algorithm is a local maximum of the maximum likelihood function. Blei et al. [BNJ03] report about overfitting problems of the aspect model in pLSA.

[5]http://research.google.com/pubs/author1113.html
[6]This is not a matrix like in LSA.

Latent Dirichlet Allocation (LDA) by Blei et al. [BNJ03] is an approach similar to pLSA. In contrast to pLSA[7], LDA offers a probabilistic model at the level of documents and not on the level of words. LDA is claimed to yield "more reasonable mixtures of topics" [Wik13e] in practice.

pLSA and LDA cannot be integrated in the same way as LSA in this thesis because they are not based on matrices. It would be interesting to investigate how the improvements which pLSA and LDA achieve over LSA can be incorporated into the Higher-Order LSA approach proposed in this thesis.

[7]pLSA is denoted as pLSI in [BNJ03]

Part II

Game-based Collection of Semantically Rich Image Tags

Chapter 3

Image Labeling With Games

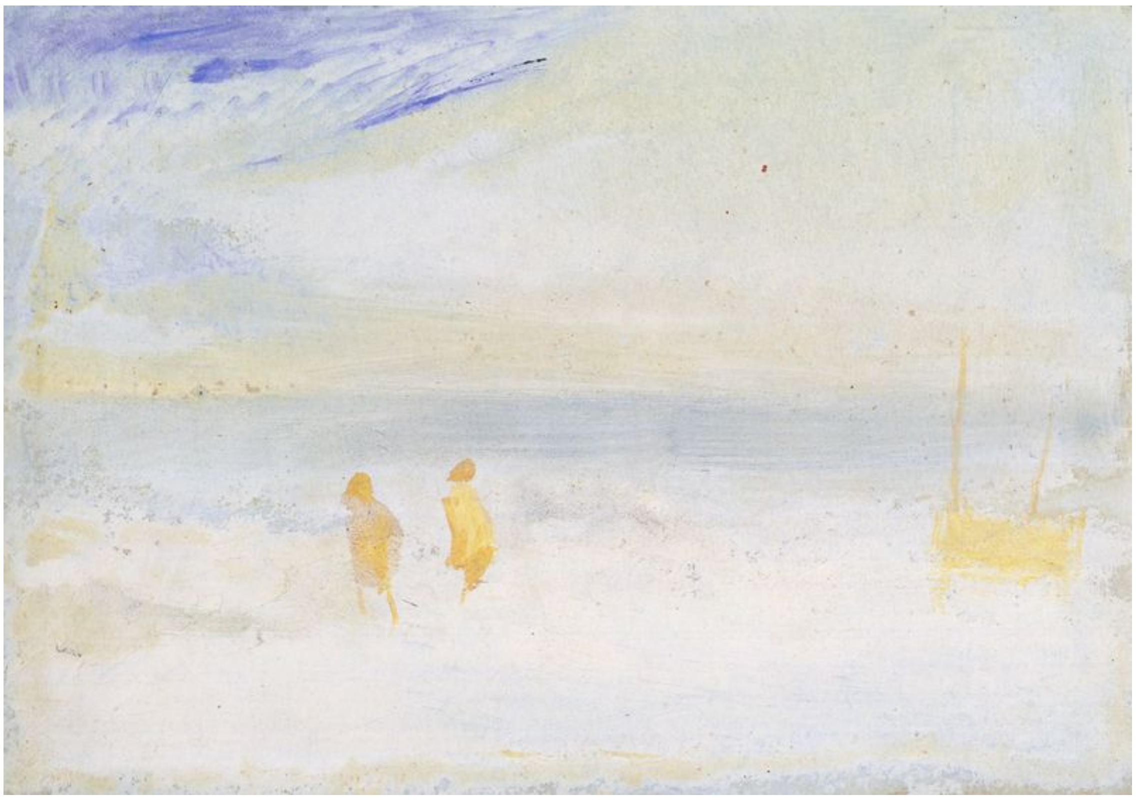


Figure 3.1: W. Turner, Two Figures on a Beach with a Boat, 1840/1845, London/British Museum.

Image Labeling denotes assigning (finite) character strings called tags to images. For example, an image in a private photo album can be assigned tags like “beach”, “boat”, or “sun”. However tags are not constrained to common words as defined, e.g., by a thesaurus or an ontology. Any tag can be assigned to an image including random character strings. The freedom to choose arbitrary tags makes it easy for humans to assign tags because no prior knowledge is needed.

In the context of a search engine, tags make resources like images available for text search. In contrast to text documents, images or music recordings do not contain machine-recognizable words that can be exploited for retrieval. Thus textual meta-data like tags are needed to bridge between text search and non-textual resources.

The basis for retrieving images by tags is the hypothesis that humans use the same tags for labeling as for retrieving images. Hence descriptive tags (seen from the user's perspective) are highly welcome in contrast to random character strings. This part of the thesis reports about acquiring tags by games. How tags can be exploited for image retrieval is treated in Part III.

Image Labeling is a task that can easily be done by humans. Tagging holiday photographs according to their contents like "beach", "boat", or "sun" is fairly easy for humans, whereas algorithmic image recognition by computers is still hard to solve. Great progress have been achieved in computer vision like face recognition [LJ11]. Another promising is the field of Content-based Image Retrieval (CBIR), which is "still a widely unresolved problem" [Deb04]. Nevertheless, humans outperform current algorithmic approaches significantly [vA13]. This superiority becomes more explicit, if artworks as in Figure 3.1 need tags that are well suited for image retrieval. While identifying colors might be easy for computers, recognizing the two figures on the beach and the boat on the right side might be reasonable for humans but hard for computers. This example illustrates the main reason for the need of human image labeling.

As mentioned in Chapter 1, humans need an incentive to execute tasks. In this case the main incentive is playing a game. Additional incentives are curiosity and the wish of getting in touch with art. The following section reports about the current state of the art in image labeling with games and is based on a publication [SWKB11] of which the author of this thesis is coauthor.

Related Work

Images Labeling in this thesis is built on previous contributions reported about in this section. The ARTigo Game and ARTigo Taboo described in Chapter 5 are close adaptions of existing games [VAD04]. Lessons learned from these games are discussed in the following to motivate the novel games proposed in Chapter 5. Further images labeling games in this section are described for the sake of a complete overview of related work. However the goal of these games is not to acquire Deep Semantics Tags as wanted for building a Semantic Search Engine.

Probably the most important and successful of all image labeling games is the ESP Game[1], created by von Ahn and Dabbish [VAD04]. The ESP Game has been adapted by Google to improve the results of their image search engine. The game has been in productive use from 2006 to 2011[2] and has been very successful in terms of numbers of collected tags and players. The basic design of the ESP Game tends to collect too generic tags (cf. Surface Semantics Tags in Section 4.1) until now. To solve this issue, von Ahn and Dabbish propose the use of so called *taboo words*. The game keeps track of the number of times a given tag has been assigned to an image. Once this number exceeds a given threshold, the tag is added to the list of taboo words for the given image. This list is shown to both players and all tags on it can no longer be used. Taboo words are meant to force players to use different and eventually more specific terms.

Game-theoretical analysis of ESP Game

Jain and Parkes present a game-theoretic model of the ESP Game [JP08]. The goal of this model is to formalize player strategies in the ESP Game and prove which ones are most successful. The authors analyze the most basic version of the ESP Game in which the scores players receive for a match are independent of the matched word. Therefore, the goal of the players is to complete as many rounds as possible within the given time limit. Jain and Parkes call this *match-early preferences*. They furthermore assume no taboo words are used in the game.

The model designed by Jain and Parkes assumes that a describing set of words exists for each image (which the provider of the game is trying to learn). The words in this set possess and are ordered by a *frequency*, which defines the likelihood of a word being assigned to the given image. Jain and Parkes then proceed to prove that playing the describing terms in order

[1]The name ESP Game stems from *Extra-Sensory Perception*, a concept that describes the communication of information between humans without relying on senses, but solely on the mind.

[2]http://images.google.com/imagelabeler/help.html

of descending frequency leads to a Bayesian-Nash equilibrium for the ESP Game. In such an equilibrium, no player can gain an advantage by changing their strategy. The authors conclude that the basic version of the ESP Game tends to lead to common tags. The analysis of taboo words and incentive structures which would lead to more uncommon tags is left as future work.

Analysis of Taboo Words

Weber et al. performed an extensive evaluation [WRV08] of the results of a version of the ESP Game. In the data they extracted, the authors found a number of redundant and generic labels. Furthermore, many tags where highly correlated. Weber et al. therefore argue that this kind of data does not necessarily need to be created by human players. To prove their point, they implemented a software that successfully plays the ESP Game. This is somewhat paradoxical, since the main objective of the game (i.e., labeling images) cannot yet be achieved reliably by computers.

The software created by Weber et al. disregards the visual content of the images and predicts likely tags by analyzing the taboo words. The software played over 400 games and achieved a match about 80% of the time. This means that the tags entered by human players are highly predictable given the taboo words. Thus, human players add little information to the existing tags even in presence of taboo words.

Further Image Labeling Games

Aside from the works of von Ahn et al., a number of alternative image labeling games with very different approaches have been proposed. For example, *PhotoSlap* by Ho et al. [HCH07] translates an existing board-game into a GWAP and allows four players to cooperate. *Picture This*, by Bennett et al. [BCM09] is designed not to label images directly, but instead improve query results using existing tags.

KissKissBan

To solve the issue of tag diversity in the ESP Game, Ho et al. created KissKissBan [HCL+09]. The fundamental difference between the two games is the introduction of a third player and a competitive element in KissKissBan. The first two players are called *Couple* and try to achieve the same goal as in the ESP Game. The third player in KissKissBan is called the *Blocker* and is competing with the Couple players. Before each round, the Blocker has seven seconds to enter as many words as possible which the Couple players are not allowed to use. In contrast to the taboo words in the ESP Game, the Couple players cannot see this list of words. If a player enters a blocked word, five seconds are subtracted from their remaining time. If the timer runs out before the Couple players achieve a match, their scores are decreased and the Blocker's score is increased. If the Couple players succeed, the opposite is the case.

Phetch

Von Ahn et al. argue that the labels created by the ESP Game are very well suited for image retrieval, but do not allow humans to form an accurate mental model of the described image. To solve this problem, von Ahn et al. propose Phetch, an image labeling game targeted at collecting entire sentences describing an image [VAGK+06]. Phetch is designed to be played by three to five players. One of the players is randomly selected as the *Describer*, whereas the other players form the group of *Seekers*. A picture is shown to the Describer, who can enter arbitrary sentences to describe the image to the Seekers. The Seekers must then use a special search engine to locate the described image. If a Seeker selects the correct image from her list of results, she and the Describer are awarded a score bonus. To discourage random guessing, points are deducted whenever a player ventures a wrong guess. After the correct image has been found, the Seeker who found it becomes the Describer in the next round.

Peekaboom

Peekaboom was created by von Ahn et al. to collect tags for images, along with the regions of the respective objects depicted in an image [VALB06]. Like in most other Games With A Purpose, Peekaboom matches two random players for each game session. An image is shown to the first player, along with a description of an object in the image. These descriptions have been generated using the ESP Game. The first player can successively reveal parts of the image to their partner. The second player can only see the areas of the image that were revealed to her. Her goal is to guess the term that was shown to the first player. If this goal is achieved, both players are awarded a score bonus and a new round is initiated.

Polarity

Polarity [LSS+11] by Law et al. tries to overcome the shortcomings of the ESP Game as mentioned above using a new gaming mechanism called complementary-agreement. A player has to select all of 9 displayed images that match a certain term. This player is called the "positive" one. The counterpart is the "negative player" who has to select all of the same 9 images that do not match the same term. Both player get rewarded, if one player selected an image that was not selected by the other one.

Polarity looks similar to a game of the ARTigo platform called Karido (cf. 5.2) because of the arrangement of images as 9 by 9 matrix. Notably the order of images in Polarity is the same for both players. In Karido the same order of images would offer an opportunity to cheat because Karido is based on communication between players via tags. However, the game mechanism is completely different as described in Section 5.2 but the main goal to overcome the shortcomings of the ESP Game unifies both games. Indeed the complementary-agreement gaming mechanism could be a promising extension to the game Eligo presented in Section 5.4.

Chapter 4

Collecting Semantically Rich Tags

The main purpose of image labeling games in this thesis is collecting tags that can be used as basis for image retrieval. Existing image labeling games as described in the previous Chapter 3 already fulfill this task to some degree. However shortcomings of these approaches have been identified. The main criticism is that for game tactical reasons players do not provide all tags they know and that they would use for image retrieval. Hence, a challenge in this thesis targeted in this chapter is to collect diverse tags that are as comprehensive as possible to provide a wide tag base for image retrieval. This is mainly done by proposing new games on one hand and by interconnecting existing and new games on the other hand. The interconnected games form a so called gaming ecosystem described in Section 4.2. Tags are the input for games yielding new tags. The output of one game is the input of other games. This principle results in a gaming ecosystem, a network of games. Furthermore, the gaming ecosystem profits from techniques called squaring and scripting, which are described in Sections 4.3 and 4.4.

4.1 Surface vs. Deep Semantics Tags

This section classifies tags mainly collected by existing image labeling games (cf. Chapter 3) as Surface Semantics Tags. Surface Semantics Tags contrast with Deep Semantics Tags, which new games and game design techniques focus on. This section is based on two publications [BW12b, BW12a] coauthored by the author of this thesis.

Consider the artwork displayed by Figure 4.1. The most frequent annotations proposed so far for it are as follows:[1]

> *black, bow tie, brow, button, dark, dress coat, eye, green, hand, leaned, look, man, moustache, portrait, shadow, shirt, suit, white.*

They are all perfectly suitable and, in their set, more informative than it might at first seem. But even though the noticeable look of the man is mentioned among the tags frequently given for this portrait, this look has so far not been characterised as, say, melancholic. For artwork search, the melancholy conveyed by this portrait is more than worth noticing. It is of primary importance.

There are no reasons to assume that the ESP Game players have little interest in mentioning, or are not sufficiently capable of recognising, the melancholy of that man's look. Exchanges with some players indicate that they are interested in artworks and often do recognise complex feelings conveyed by artworks.

Most likely, an ESP Game player confronted with a picture such as that given of Figure 4.1 will enter "easy tags", that is, tags that his counterparty player is likely to enter as well. Since "suit" and "look" are more likely than "melancholy" or "melancholic look" because they are both

[1] The tags were collected on http://www.artigo.org/ and translated from German, listed in alphabetic order and disregarding their relative frequencies.

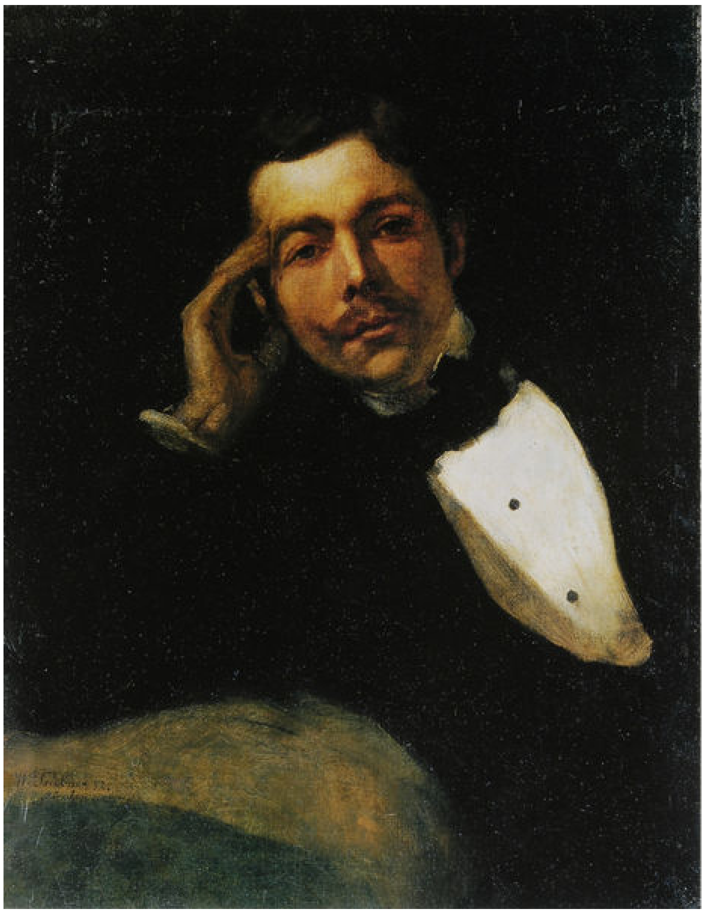


Figure 4.1: H. Wilhelm Trübner, Self-Portrait, 1882.

simpler and more concrete concepts, the former tags are very often entered while the latter are not or very rarely entered.

Let us call "Surface Semantics Tags" tags expressing what an art work or any image –art or not– directly conveys, among other, what it explicitly represents –in the case considered, a man, a bow tie, etc.–, what it is –in our case, a portrait–, or what it consists of –in our case the colours black, green, and white. Surface Semantics Tags are very likely non-controversial. Let us call "deep semantics tag" any tag expressing what an artwork, or more generally an image, might convey as well but does not fall under the notion of "Surface Semantics Tag". Admittedly, there is no clear cut between surface and Deep Semantics Tags. Such a clear cut is not needed, though. Since labelling images with tags using a GWAP is a crowd-sourcing activity, frequencies will at last reveal what a community considers to be Surface and Deep Semantics. In other words, we intend to exploit the strength of the ESP Game at collecting tags of the one kind and to devising additional games, concretely variations on the ESP Game, well capable of collecting tags of the other kind.

We found that the ESP Game in its standard realisations collects mostly Surface Semantics Tags and performs particularly poorly at collecting Deep Semantics Tags. This finding is consistent with former observations [WRV08, LSS$^+$11, SWKB11, JP08]. Our hypothesis is that this lies at the very nature of the ESP game: With this game, success is better achieved by focusing at the most likely, therefore at the simplest and most concrete. This hypothesis is sustained by the following observation: ESP game players tend to tag abstract art works –such as that of Figure 4.2– first and mostly with geometric shapes and colours. The composition of Figure 4.2 has for example

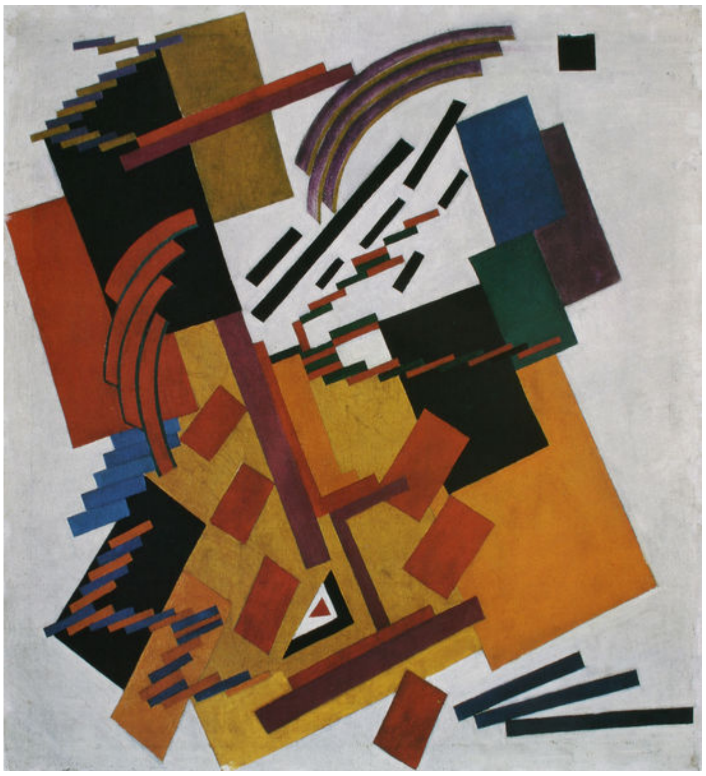


Figure 4.2: O. Rosanowa, Non-objective Composition, 1916, Swerdlowsk/State Artmuseum

been mostly tagged with colours and geometric shapes:[2]

> *angular, arcs, black, blue, brown, geometry, green, multicolour, orange, rectangles, round, shapes, squares, white, and yellow.*

It could also have been tagged with *movement* and *order*, though.

The great ability of the ESP Game to collect simple, concrete and rather immediate descriptions is not to be wrongly interpreted as a weakness of that game. The simplest and most concrete tags are undoubtedly needed for most applications, among other for the artwork search engine proposed in this thesis.

4.2 Game Types Building a Gaming Ecosystem

Many image labeling games have been conceived as reported in Chapter 3. The major outcome of these games are Surface Semantics Tags and not deep semantic tags (cf. Section 4.1). From the perspective of image retrieval, all kinds of tags and in particular Deep Semantics Tags are wanted to improve search results. The approach reported in this section called gaming ecosystem is meant to acquire Deep Semantics Tags. Experiments reported about in Chapter 5 show that the gaming ecosystem indeed collects Deep Semantics Tags. An approach for acquiring Deep Semantics Tags with standalone image labeling games is proposed in Section 4.4 later in this chapter. This section is based on a publication [WBBL13] coauthored by the author of this thesis.

The gaming ecosystem approach differs from standalone image labeling games mentioned in Chapter 3. Tags collected by such games are used as input for other games yielding additional

[2]The tags were collected on http://www.artigo.org and translated from German, in alphabetical order and disregarding the relative frequencies.

Surface Semantics Tags as well as Deep Semantics Tags. The output of games in turn can be used as input for other games. This is the basic principle of the gaming ecosystem.

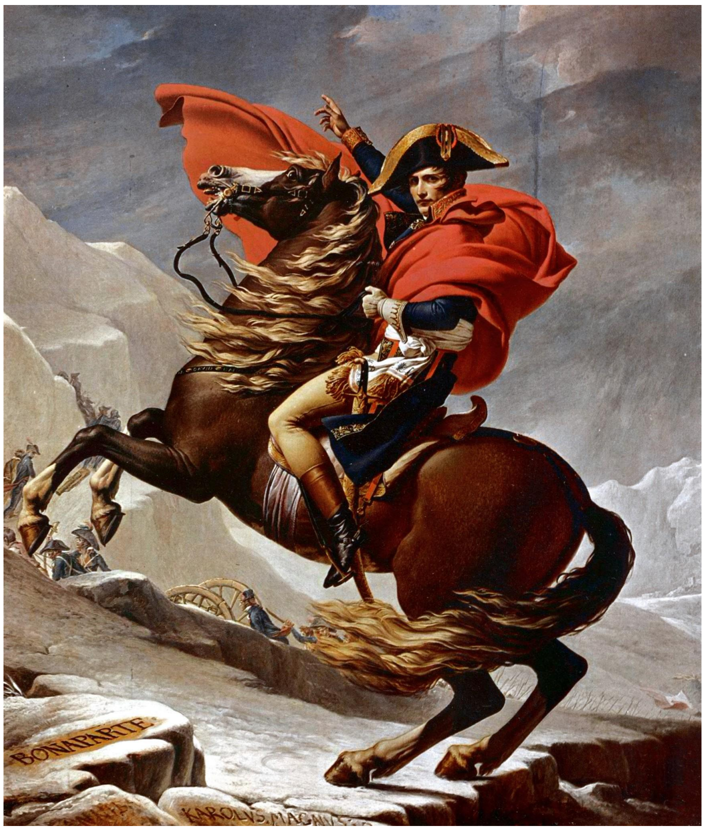


Figure 4.3: J.-L. David, Napoleon crossing the Alps, 1800, Malmaison/Musée National.

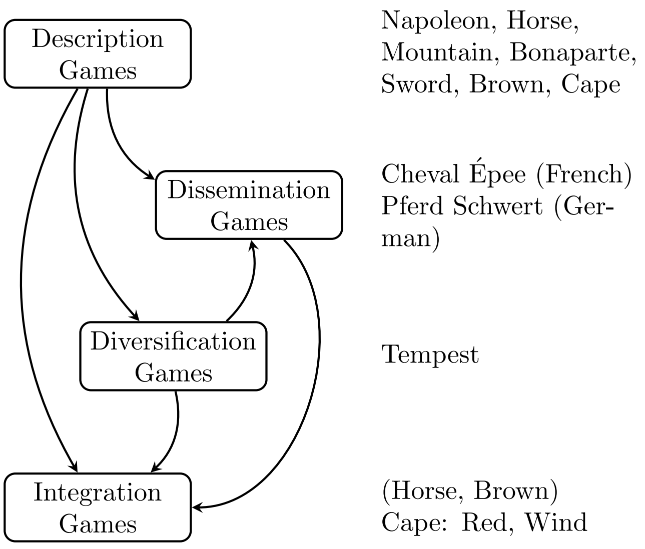

Description Games
Dissemination Games
Diversification Games
Integration Games
Napoleon, Horse, Mountain, Bonaparte, Sword, Brown, Cape
Cheval Épee (French) Pferd Schwert (German)
Tempest
(Horse, Brown) Cape: Red, Wind

Figure 4.4: Tag flow in the ARTigo Gaming Ecosystem using the example of Figure 4.3.

The graph on the left side of Figure 4.4 illustrates the conceived gaming ecosystem. Boxes stand for classes of games. The classic ESP Game, e.g., belongs to the class of description games.

Arrows visualize the flow of tags. For example, tags acquired by the classic ESP game are used as input for diversification games like Karido (cf. Section 5.2).

The classes of games can be seen as a kind of programming framework acquiring diverse kinds of tags for a resource. Unlike in programming frameworks, the flow of tags is not a classical data stream in the sense that one game sends tags to the other. Indeed, games can access the database of acquired tags and make use of them. Hence an arrow indicates, that tags collected by one game are used by another game.

The center of Figure 4.4 lists tags acquired by Napoleon crossing the Alps shown on the right side of Figure 4.4 is tagged. Description Games collect shallow surface tags like "Horse" that can be translated to "Cheval" or "Pferd" via Dissemination Games. Translations are strongly dependent on the context. Therefore automatic translations based on dictionaries are often not sufficient. Diversification Games collect tags for similarly tagged images, which again can be input for Dissemination Games. Obviously, the resulting tags can be disseminated. Finally all collected tags can be combined to integrated tags for deep semantic descriptions of images. Dissemination of integrated tags is not conceived in this thesis and subject to future work.

The ARTigo platform (cf. Chapter 5) offers several games classified in four different categories: description, dissemination, diversification and integration games, each category collecting different types of artwork descriptions. Each category of tags is important for the artwork search engine to provide answers as exhaustive and as precise as possible. The games and their specificities and why they collect data of the aforementioned categories is discussed below.

Description Games

Description Games are simple games whose players have to describe an artwork by proposing tags. The tags can be related to anything referring to the artwork like objects or characters it depicts, its colours, the materials it is made of.

A tag is validated if one or more other players enter it as well. This validation method is necessary to collect correct data. Without validation, the gaming platform could be misused. Experiences with full-text search based on validated taggings of the ARTigo platform indicate that even onetime validation yield good results.

Even though they mostly are rather immediate descriptions, or surface Semantics Tags, the tags collected by description games can be used for several purposes, among other to "feed" games with data to make these other games playable. Thus, surface tags generated by description games such as the ESP Game are necessary not only as artwork descriptions but also for collecting Deep Semantics Tags with other kinds of games like Karido (cf. Section 5.2).

Dissemination Games: Spreading Information

Dissemination Games propagate description tags to other artworks or to other languages, that is for translation purposes. Usually a tag has many valid translations suggested by a dictionary. Filtering out wrong translations depends strongly on the context like an artwork. This is a standard task for Human Computation and hard to solve for computers [vA13]. A dissemination game can be used to discriminate several artworks that are similarly tagged by adding the so far missing tags to an artwork's description.

Dissemination games can also be useful to generate tag translations. For instance, a game called Eligo [WBBL13] (cf. Section 5.4) which so far is not offered on ARTigo but still undergoes tests, uses German tags assigned to artworks and translates them into another language (English or French in the currently running tests). Eligo players then check these translations by selecting the images that correspond to automatically translated tags.

The tags generated with dissemination games are validated like those generated with description games. If, for example, at least two (or more) Eligo players accept a translation, then it is deemed correct. Experiences suggest that this validation is acceptable in practice.

Diversification Games: Telling Artworks Apart

Diversification Games produce more precise tags and/or tags of a deeper semantics. In order to produce these tags, diversification games use description tags, which had already been collected. An example of a diversification game is Karido (cf. Section 5.2): the algorithm chooses artworks that have been similarly described so far and ask players to discriminate between them with additional tags. Once again, tags generated with diversification games can be validated like those produced with description or dissemination games.

Integration Games

Integration games cluster tags yielding more precise descriptions than unstructured sets of tags. Tag clusters are often more difficult for a player to suggest since they require a deeper analysis of an artwork, in some cases even specific knowledge. Integration games are therefore often more challenging than games of other kinds what, in turn, contributes to the attractiveness of the gaming platform.

The game Tag A Tag [BW12a, BW12b] makes players propose combinations of tags. The game Sentiment [BW12a, BW12b] requires for players to reflect and report on the feelings an artwork may convey. Both games are described in Section 5.3 in detail.

Related work for the integration of tags is Cascade [CLE$^+$13] by Chilton et. al. Cascade is a crowdsourcing approach to generate taxonomies consisting of structured tags but it is not a game. Workers perform three tasks to construct a taxonomy. The first task called Generate is to suggest suitable categories for items like images. The second task is called SelectBest, where a worker selects matching categories for one item. The third step is called Categorize, which lets a worker check if certain categories fit with an item. The steps can be performed by different workers independently and the workers do not need to be trained to use the taxonomy generated so far. In some sense Cascade is a tagging ecosystem as well taking the three steps as classes that enrich the tags collected so far. Finally, the resulting taxonomy is be calculated by a computer.

Cold Start

Cold Start is a problem that the ARTigo's game face like most other GWAPs. None of its games are playable if too few artworks are sufficiently described by tags.

This problem is solved through a preliminary, and sufficient, tagging of a sufficiently enlarged collection of artworks mostly by volunteer and/or payed students. To this aim, the ARTigo platform has an interface to its artwork database called ARTigo Seed. This interface is, of course, not accessible to players since the tags entered using it are considered validated.

4.3 Squaring the ESP Game

Integration games, one of the classes in the gaming ecosystem (cf. Section 4.2), profit from a novel game design technique called Squaring. Squaring is a bootstrapping technique making integration games out of description games. This approach was first reported about in [BW12a, BW12b] where the author of this thesis is coauthor. The following section is based on these publications.

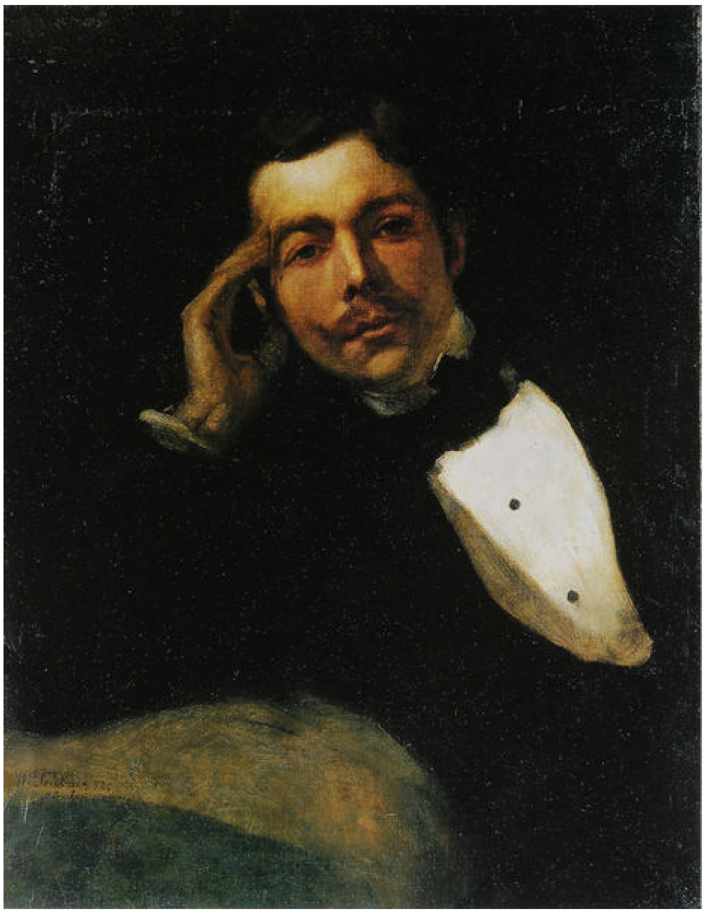


Figure 4.5: H. Wilhelm Trübner, Self-Portrait, 1882.

Consider the portrait of Figure 4.5 which has been often tagged with “look”, the precise nature of the man’s look, however, so far has not been characterised in any manner by players. A variation of the ESP Game we call Tag A Tag asks players to tag the pair consisting of the artwork of Figure 4.5 and of the tag “look”. More generally, Tag A Tag is an ESP Game that asks players to tag a pair consisting of an image and a tag that has been formerly collected for this image, for example with the standard ESP Game or with any other games with a purpose. We call this variation of the ESP Game a “squared ESP Game”, or (ESP Game)^2, because the ESP Game is applied to data collected with the ESP game itself.

Squaring the ESP Game is possible without any selection of the tags selected for the pairs (image, tag) to be presented to players. Squaring the ESP Games does not require for the gaming platform operators to develop a new game and for players to learn how to perform well at a new game. Some pairs (image, tag) turn out to be tricky to play well with. This, however, as indicated by first evaluations, contributes to the attractiveness of the gaming platform. Furthermore, pairs (image, tag) difficult to tag turn out to collect only a few tags but highly interesting ones. The pair consisting of the drawing of Figure 4.6 associated with the tag “black”, in German “schwarz”, can well be tagged with “everything” or, referring to the family name of the portrayed man. Both

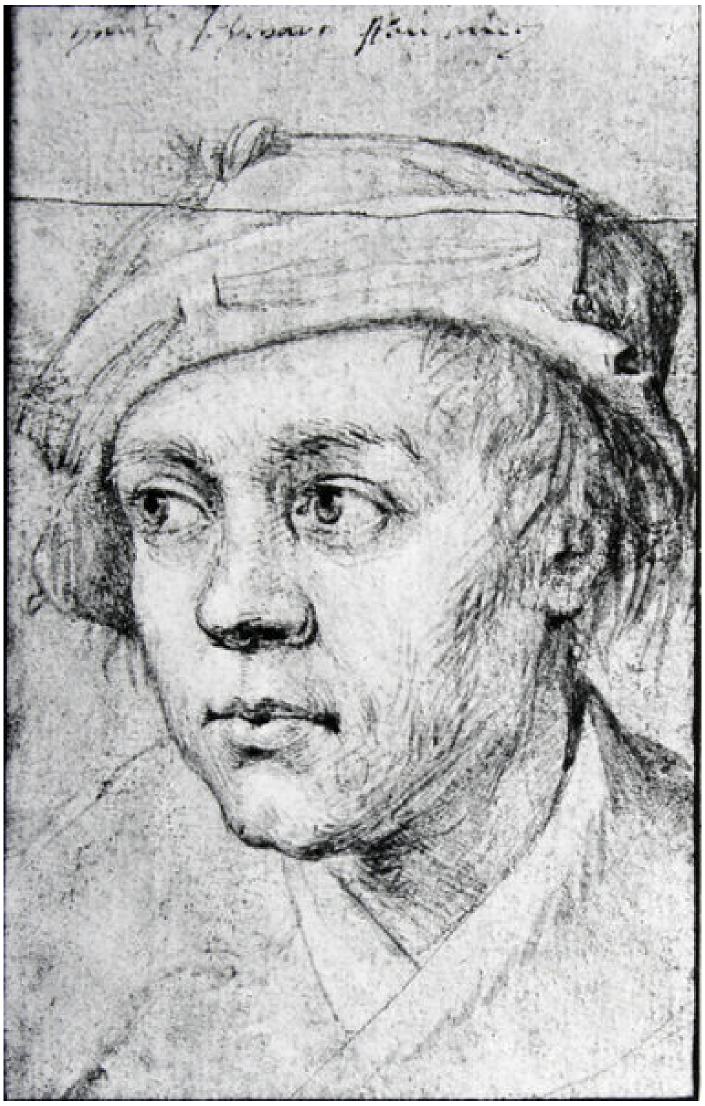


Figure 4.6: Holbein, Portrait of Hans Schwarz, 1494/1522, Berlin/State Museum.

tags are highly valuable for the purpose of developing an artwork search engine. Arguably, squared ESP Games are likely to be very useful for other applications as well. First experiments point to the pertinence of squaring the ESP Game –it does generate Deep Semantics Tags– and to its attractiveness for players –players do not leave or avoid the game.

The ESP Game can be squared in more sophisticated manners than the one mentioned above. A few are discussed in the following.

First, automatically generated questions can be posed. A pair consisting of a tag W and an image can be presented with one of the following questions, depending whether W is a noun, an adjective or a verb:[3] *How is W in this image? What is W in this image? What is W-ing in this image?*

Second, squaring can be realised using a set of tags instead of a single tag. The approach is especially promising if the set of tags is selected –of course automatically– for semantic reasons. Tags that are semantically related can for example be selected. A semantic selection of tag sets

[3]Tags that are neither nouns, nor adjectives, nor verbs can be omitted in generating questions. More sophisticated forms of automated question generation capable of coping with more sophisticated tags can be considered, too.

for squaring is likely to be especially effective, that is, to lead to useful tags, if the set is built up considering the objectives of the platform.

Third, squaring can be realised with one tag and several images. Consider the images of Figures 4.7 and 4.8 that, understandably, have both been tagged with "bridge". They could be presented with one of the following two questions: *What have the bridges in common in these images? What distinguishes the bridges in these images?* Answers to these questions are not as difficult as it might first seem. For example, both bridges are old and located in England, and in Europe, the one is well visible, small and a photograph, the other is hardly recognisable, large and a painting. Tags entered as answers to the aforementioned two questions express semantic relationships –in case of the images of Figures 4.7 and 4.8 on their "bridge-ness". The second question is especially interesting because, like the games Polarity [LSS$^+$11] and Karido [SWKB11] it yields tags differentiating the images presented. A sophisticated semantic selection of a few images is likely to both generate semantically rich tags and make the game particularly engaging.

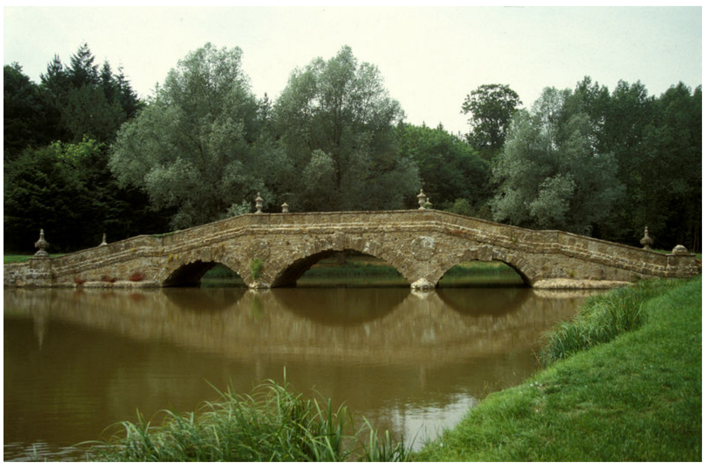


Figure 4.7: Anonymous, Oxford Bridge in Stowe Landscape Gardens, 1761, Buckinghamshire/Stowe Landscape Gardens

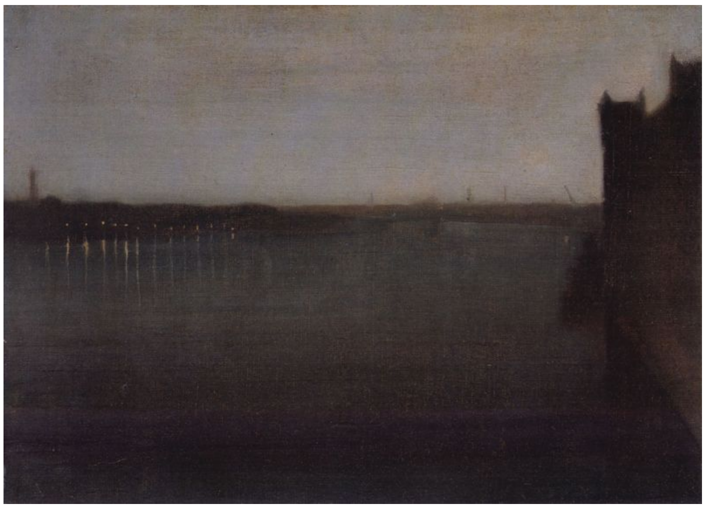


Figure 4.8: J. Whistler, Nocturne in Gray and Gold, 1871/1874, Westminster Bridge, Glasgow/Burrell Collection

4.4 Scripting the ESP Game

The ESP Game as conceived by Luis von Ahn [VAD04] does not restrict players to certain tags, which leads to Surface Semantics Tags. In this section scripting is proposed as technique to let the ESP Game acquire Deep Semantics Tags. This section is based on a publication [BW12a, BW12b] coauthored by the author of this thesis.

If one wants players of an ESP Game to propose tags that express, say, the sentiments an artwork conveys, then the simplest approach is to explicitly ask them to do so! This approach turns out to be very effective. On a beta version of an extension of the ARTigo platform, a an ESP Game called Sentiment is proposed which does that. The tags collected so far by Sentiment are very promising.

Playing Sentiment is, admittedly, at first not always simple. Consider for example Figure 4.9. It is not at first obvious what sentiment such a plan might convey. A second thought, however, is that this plan clearly conveys order, which, most interestingly, is a characteristic of 17th century palace gardens in Europe. Another feeling this plan conveys is that of depth which, once again, is a very appropriate description of what the architects of baroque theatres, as opposed to theatres of former ages, have intended to convey.

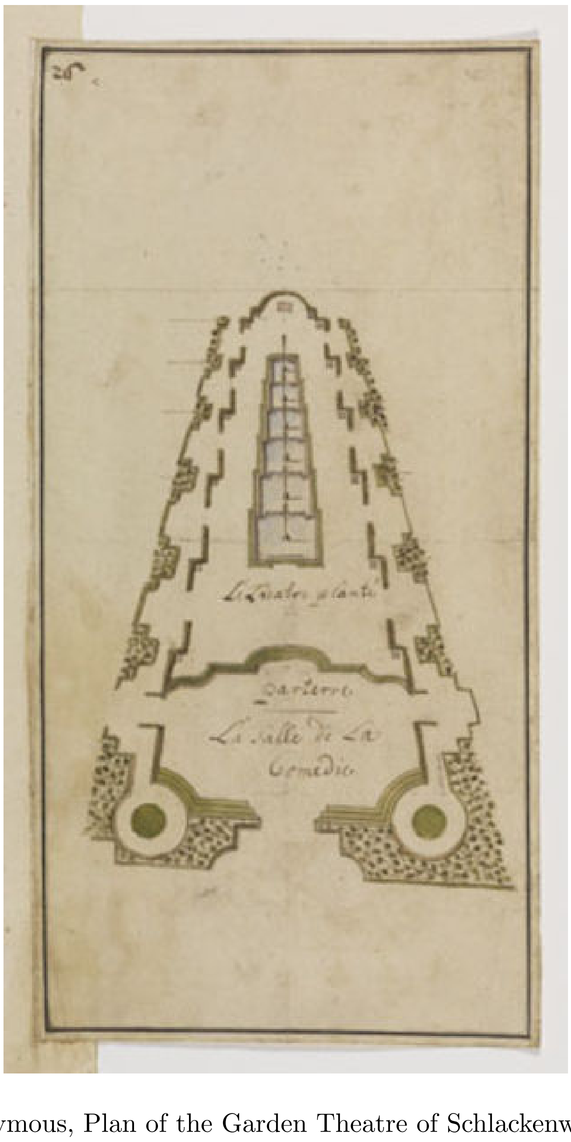


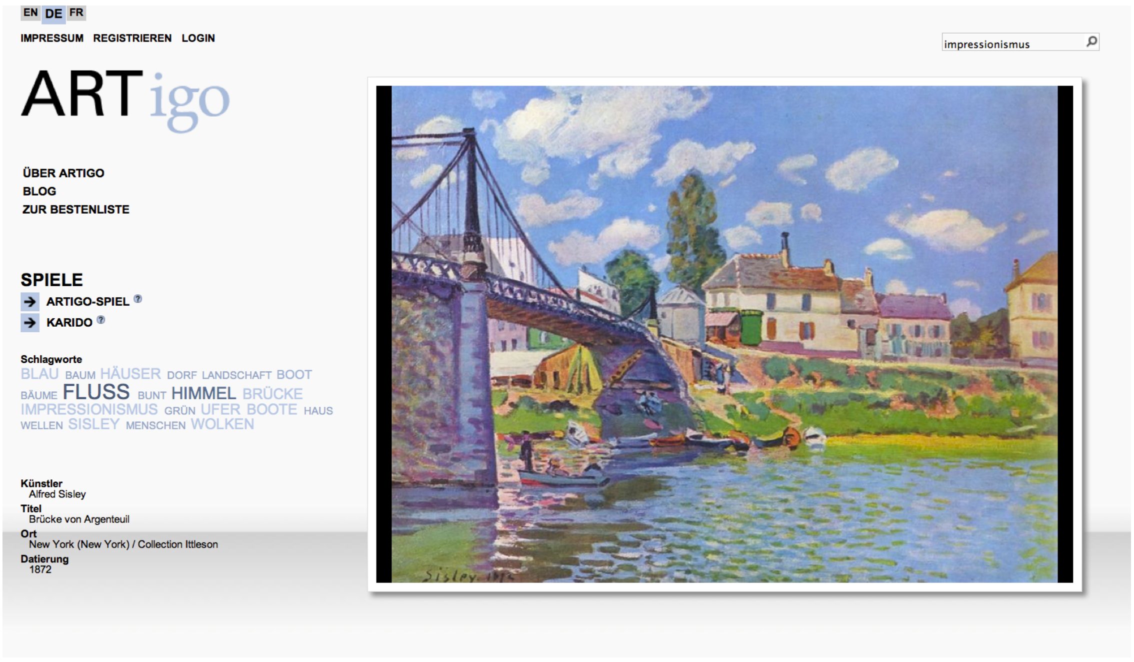
Figure 4.9: Anonymous, Plan of the Garden Theatre of Schlackenwerth Palace, 1690

The approach outlined above, or more precisely a generalisation of it discussed in the remaining of this section, is called “scripting the ESP Game”. In pedagogical psychology, scripting denotes techniques to bring learners to perform as desired, for example to tackle the resolution of linear equations in one of the manners that easily and surely lead to success [OD92, RWC+08, Kin07].

Explicitly telling people what to do is a bit crude. It is often not so well received. Setting a context that leads them to do what is desired, that is providing guidance, is more promising an approach. In several fields, among other Human Machine Interaction, researchers have investigated how such contexts can be set for users.

A common observation is that appropriate contexts are more effective at making users act as expected than explicit instructions. A conjecture is that this applies to players, too. The remaining of this section is devoted to reflexions on how to “script” the ESP Game so as to lead its players to propose Deep Semantics Tags.

First and foremost, the scoring scheme is a form of scripting. A scoring scheme is a good scripting, if it is well understood by the players.


EN DE FR
IMPRESSUM REGISTRIEREN LOGIN
impressionismus
ARTigo
ÜBER ARTIGO
BLOG
ZUR BESTENLISTE
SPIELE
ARTIGO-SPIEL
KARIDO
Schlagworte
BLAU BAUM HÄUSER DORF LANDSCHAFT BOOT BÄUME FLUSS BUNT HIMMEL BRÜCKE IMPRESSIONISMUS GRÜN UFER BOOTE HAUS WELLEN SISLEY MENSCHEN WOLKEN
Sisley 1872

Künstler
Alfred Sisley
Titel
Brücke von Argenteuil
Ort
New York (New York) / Collection Ittleson
Datierung
1872

Figure 4.10: ARTigo Starting Page – Alfred Sisley, Bridge of Argenteuil, 1872

Second, Deep Semantics Tags can be displayed on the starting page of a game. This is done on ARTigo. Every day a different artwork is presented with its tag cloud. Figure 4.10 shows how ARTigo’s starting page might look like. The tag cloud displayed with Sisley’s painting of the bridge of Argenteuil reads:[4]

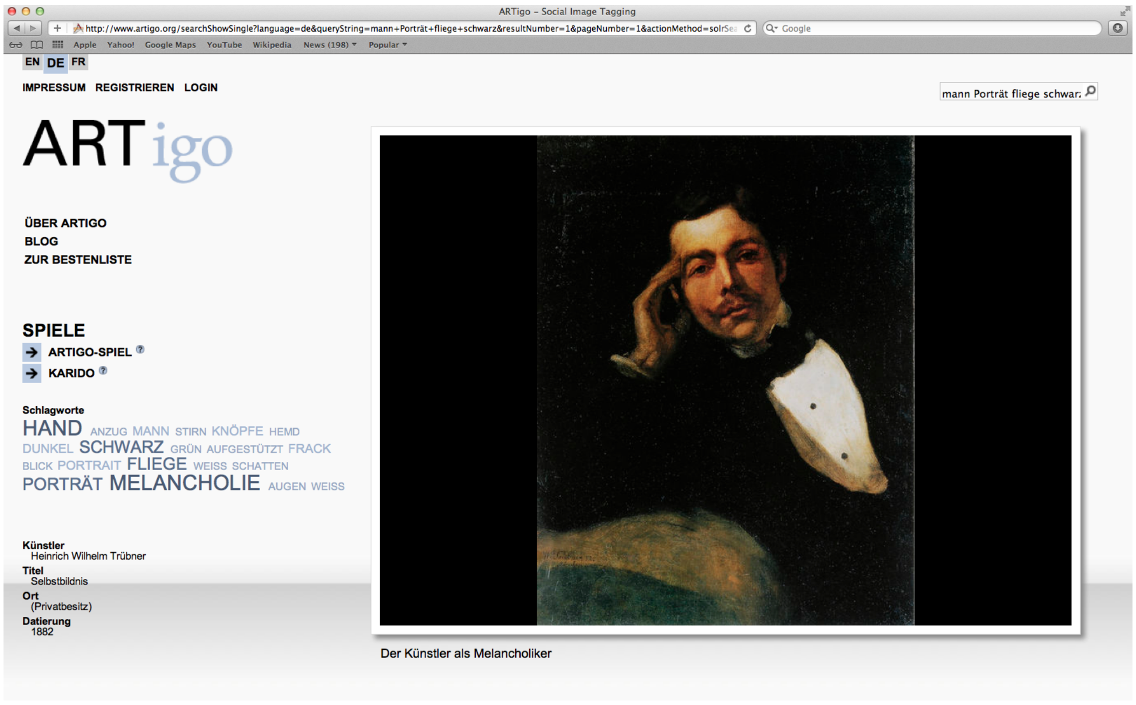
> *sky, trees, Sisley, landscape, bank, waves, impressionism, bridge, blue, village, boat, boats, tree, multicolour, river, clouds, houses, people, green, house.*

We script to deep semantics by extending this tag cloud with:

> *peace, harmony, repose.*

Third, captions are displayed with the artwork on the starting page of ARTigo so as to suggest a critical, or deep semantics, look, at art works. Figures 4.11 and 4.12 give examples of such captions.

[4]In German: Himmel, Bäume, Sisley, Landschaft, Ufer, Wellen, Impressionismus, Brücke, blau, Dorf, Boot, Boote, Baum, bunt, Fluss, Wolken, Häuser, Menschen, grün, Haus.

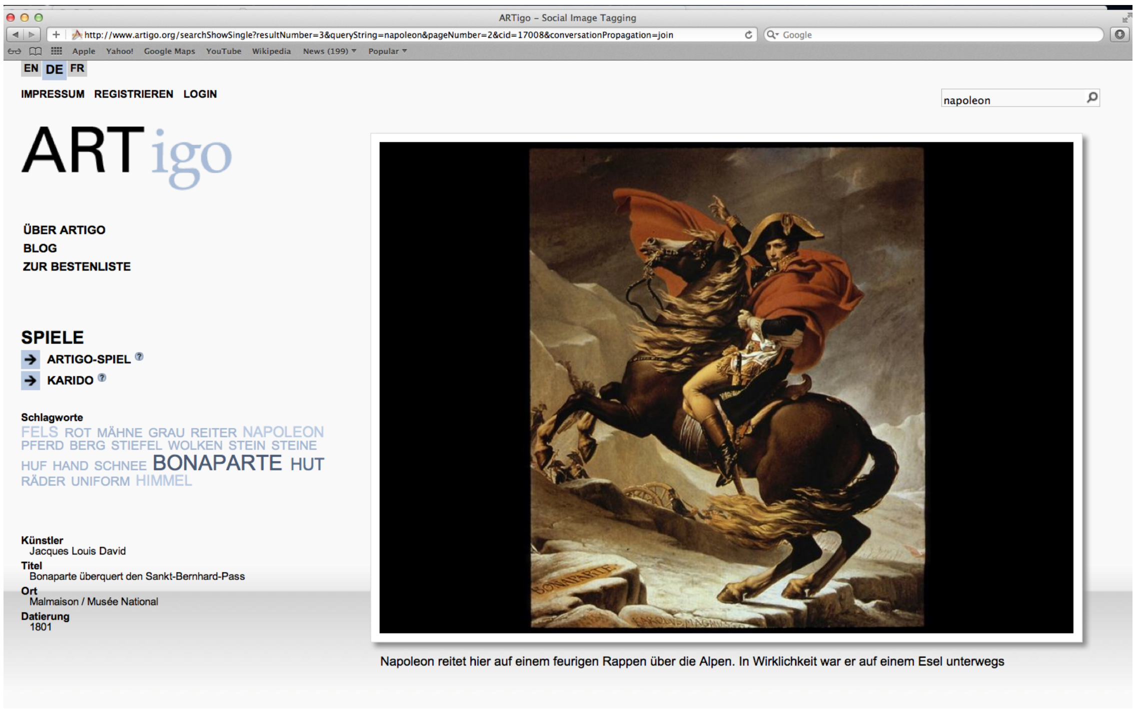

ARTigo – Social Image Tagging
http://www.artigo.org/searchShowSingle?language=de&queryString=mann+Porträt+fliege+schwarz&resultNumber=1&pageNumber=1&actionMethod=solrSe
EN DE FR
IMPRESSUM REGISTRIEREN LOGIN
mann Porträt fliege schwarz
ARTigo
ÜBER ARTIGO
BLOG
ZUR BESTENLISTE
SPIELE
ARTIGO-SPIEL
KARIDO

Schlagworte
HAND ANZUG MANN STIRN KNÖPFE HEMD DUNKEL SCHWARZ GRÜN AUFGESTÜTZT FRACK BLICK PORTRAIT FLIEGE WEISS SCHATTEN PORTRÄT MELANCHOLIE AUGEN WEISS

Künstler
Heinrich Wilhelm Trübner
Titel
Selbstbildnis
Ort
(Privatbesitz)
Datierung
1882

Der Künstler als Melancholiker

Figure 4.11: ARTigo Starting Page – Heinrich Wilhelm Trübner, Self-Portrait, 1882. “The artist as a melancholiac”


ARTigo – Social Image Tagging
http://www.artigo.org/searchShowSingle?resultNumber=3&queryString=napoleon&pageNumber=2&cid=17008&conversationPropagation=join
EN DE FR
IMPRESSUM REGISTRIEREN LOGIN
napoleon
ARTigo
ÜBER ARTIGO
BLOG
ZUR BESTENLISTE
SPIELE
ARTIGO-SPIEL
KARIDO

Schlagworte
FELS ROT MÄHNE GRAU REITER NAPOLEON PFERD BERG STIEFEL WOLKEN STEIN STEINE HUF HAND SCHNEE BONAPARTE HUT RÄDER UNIFORM HIMMEL

Künstler
Jacques Louis David
Titel
Bonaparte überquert den Sankt-Bernhard-Pass
Ort
Malmaison / Musée National
Datierung
1801

Napoleon reitet hier auf einem feurigen Rappen über die Alpen. In Wirklichkeit war er auf einem Esel unterwegs

Figure 4.12: ARTigo Starting Page – Jacques Louis David, Napoléon crossing the St. Bernard Pass, 1800, “Napoleon rides a fiery black horse over the Alps. In reality, he rode a donkey.”

ARTigo displays every day a different artwork with a caption. The captions are informative, sometimes surprising, in some cases even provocative. They always trigger reflexions as in the case displayed by Figures 4.11 and 4.12 that respectively point to the mood of the portrayed person, the propagandising of the painting and the social criticism expressed by a rather strange street view. These captions greatly contribute both to the attractiveness of the gaming platform, to its focus on art history, and its collecting Deep Semantics Tag. Some players have reported looking almost every day at the platform starting page for discovering the artwork and caption of the day. We conclude that these captions are successful a means for scripting.

ARTigo's captions come, however, at a price: They cannot be automatically generated and require human work. The very quality of the captions that make them good a scripting means also make them more a craft than an –at least partly– automatable task. This being said, captions are only needed for setting a context, and only a few captions suffice. A few hundred captions suffice for the scripting thought for: After a couple of months, a same caption is displayed once again.

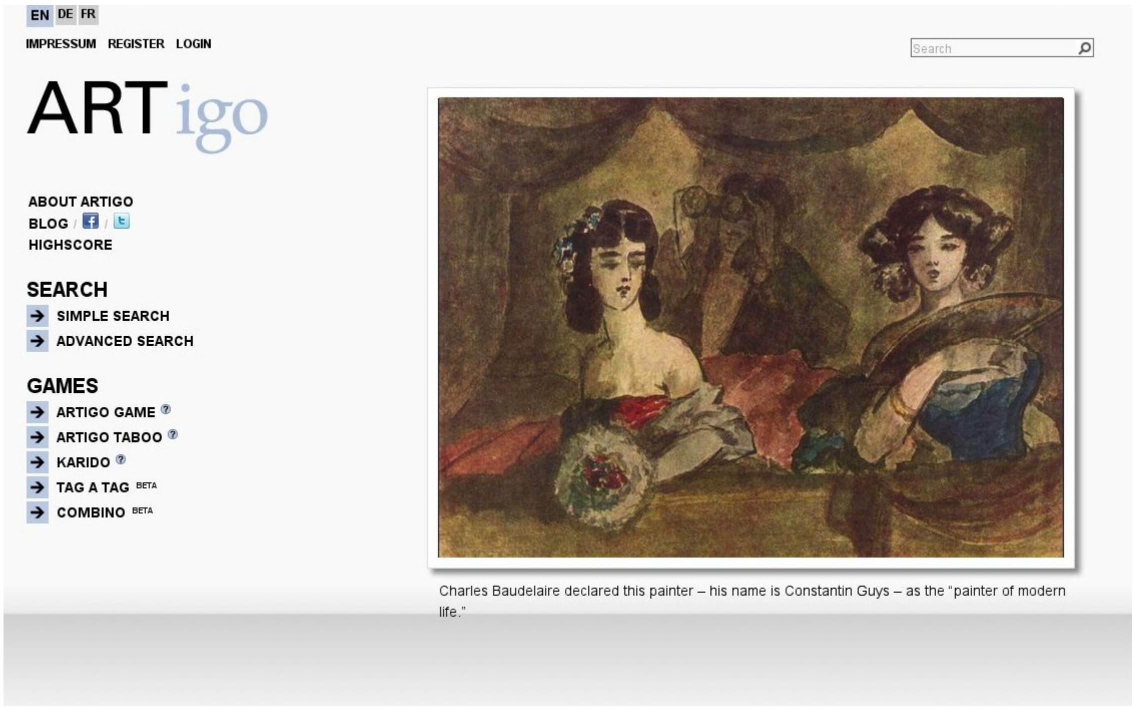
Fourth, the images selected for a round of the ESP Game can be selected in such a way to induce an atmosphere, for example by selecting only abstract or Impressionist or Gothic artworks. Such a notion of "thematic round" can be conveyed to the users, for example by providing the rounds with titles. An essential requirement is that such title do not need being worked out by human. Indeed, this would soon, especially if the game becomes even more popular, require more human work than might be provided with. Finally, combinations of the aforementioned forms of scripting can be though of.

Like squaring the ESP Game, scripting it is appealing because it does not require for the gaming platform operators to develop new games and for players to learn how to perform well at new games.

Chapter 5

The ARTigo Gaming Ecosystem

ARTigo (http://www.artigo.org/) is a web platform that provides a large database of (images of) artworks of different kinds and several GWAPs referring to these artworks. The data collected thanks to these GWAPs is used to build a semantic search engine (cf. Part III) for the (images of the) artworks of the database. Figure 5.1 displays the front page of ARTigo leading to its main features search and games.



Charles Baudelaire declared this painter – his name is Constantin Guys – as the "painter of modern life."

Figure 5.1: C. Guys, The Loge of the Opera, 1821/1892, Wien/Albertina

ARTigo started in November 2007 with an adaption of the ESP Game (cf. Chapter 3) implemented by Dr. Gerhard Schön using PHP[1] and MySQL[2]. The current version of ARTigo is a reimplementation of the former version based on Java Enterprise Edition[3] and PostgreSQL[4] enriched with novel games described in this chapter. The data collected by the former version

[1]http://www.php.net/
[2]http://www.mysql.com/
[3]http://www.oracle.com/technetwork/java/javaee/overview/index.html
[4]http://www.postgresql.org/

were transferred into a new database schema and are of major importance as data basis for the gaming ecosystem described in this chapter.

Since 2007 ARTigo has been growing in terms of the community, the number of images, and the number of tags. Until 6. März 2014, 6.722.045 tags have been submitted by 18.607 registered players[5] for 46.176 images. Every day new users continue the work of indexing ARTigo images. Since the ARTigo platform was taken over by the author of this thesis in 2009, the gaming ecosystem with the ARTigo Game as basis has been stepwise extended.

So far, ARTigo offers five GWAPs (cf. left-sided menu in Figure 5.1). ARTigo's GWAPs are complementary: They have been designed for collecting semantically rich tags (cf. Chapter 4). ARTigo's image labeler, called ARTigo Game on the platform, is effective at collecting tags describing artworks but much less at collecting tags discriminating between similar artworks with similar content. ARTigo Taboo is a version of the ARTigo game constraining the input of tags. Karido excels at generating artwork discriminating tags. While the image labeler and Karido are good at generating surface tags, Tag A Tag is very successful at generating deep semantic tags. Combino in turn generates semantically richer tags by combining surface or deep semantic tags. Further games like Eligo are currently go through tests before reaching productive status.

This chapter is based on the following publications [WBBL13, BW12a, BW12b, BKW11, SWKB11] coauthored by the author of this thesis.

5.1 Description Games: Variants of the Image Labeler

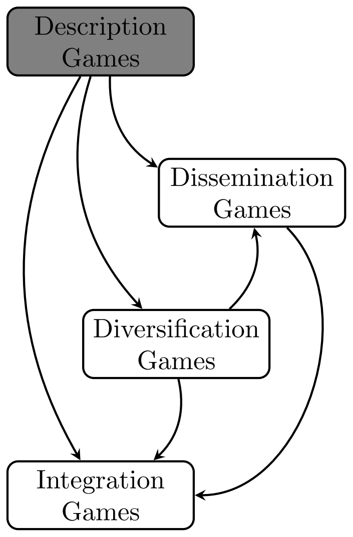


Figure 5.2: Description Games in the visualization of the Gaming Ecosystem in Figure 4.4.

The ARTigo Game

The ARTigo Game is a variation of the ESP game from Luis von Ahn [VAD04] and the prime game on the ARTigo platform. The letters "ESP" stand for ExtraSensory Perception, a term used by psychologists working in the field of parapsychology [Rhi66]. ESP refers to Extra-Sensorial Perception, as if playing the ESP Game would require such powers.

The ARTigo Game is aimed to essentially collect surface tags. As already mentioned (cf. Section 4.2), surface Semantics Tags are the basis of the gaming ecosystem because they are the main input of other games to acquire, e.g., Deep Semantics Tags. Quite rightly, the ARTigo Game was not only the first game of the ARTigo platform but also the principal game.

[5]In total 164.745 players contributed *anonymously* and, hence, may be counted several times.

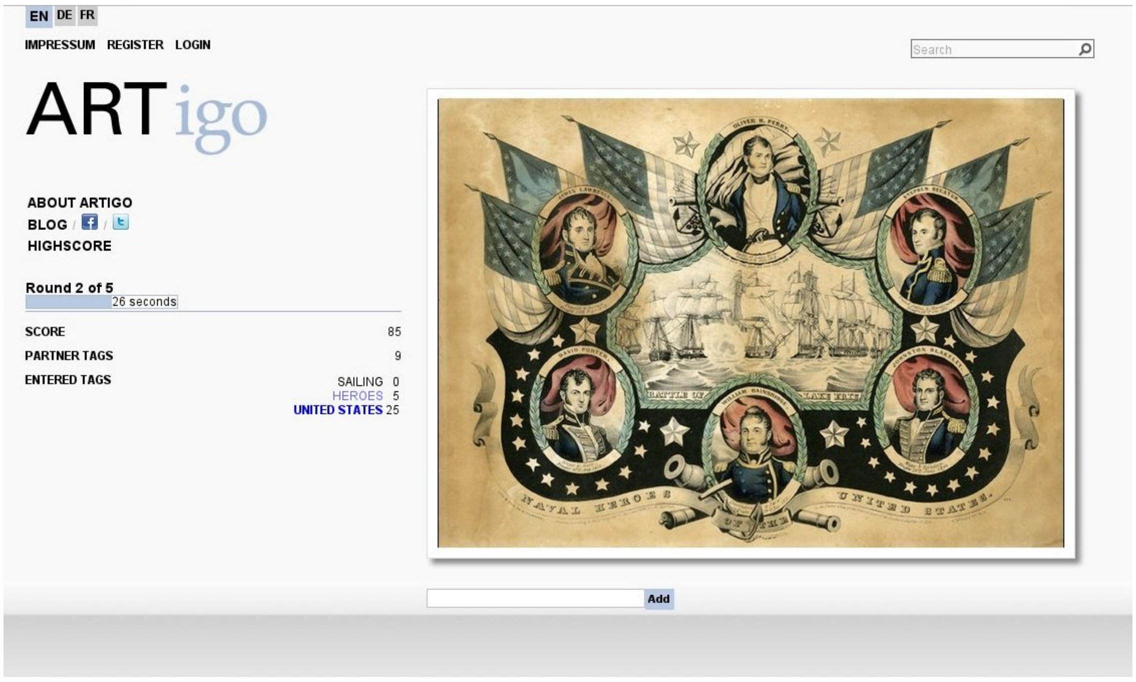


Figure 5.3: Game round of the ARTigo Game, N. Currier, Naval Heroes of the United States, 1846, Massachusetts, USA/Mead Art Museum.

As described in the original paper about the ESP Game [VAD04], the ESP Game is an output agreement game, where both players see the same image, and then have to agree on descriptive tags. Figure 5.3 shows such a game round from the perspective of one of the two players. The player is supposed to enter tags for the image displayed. In Figure 5.3 the player entered the tags "Sailing", "Heroes", and "United States". Each of these exemplary tags is differently rewarded with points according to the level of agreement. "Sailing" has never been entered for this image before, which yields 0 points. "Heroes" is rewarded with 5 points because the tag was entered before, not in this game session but in a previous one. Finally, "United States" was entered by the other player in this game session and is rewarded 25 points, which is the highest amount of points. Rewarding players with points is part of the Scripting Strategy described in Section 4.4.

The need for agreements on entered tags has various reasons. First, undesired tags like inappropriate descriptions or even politically incorrect tags should be filtered out. The filtering strategy is based on probability, or more rather improbability. It is very unlikely that two mutually unknown players without the chance to communicate enter unwanted tags. This filtering strategy is not perfect but was found reliable in practice. Second, input agreement is a means to motivate players to enter tags because it is thrilling to empathize with the other player.

The levels of agreement rewarded with diverse amounts of points reflect the value of an agreement. Obviously, different tags are on the lowest level and yield no point. The next level –not rewarded in ARTigo– are tags entered for the first time for an image. Such tags are often Deep Semantics Tags suffering from the low probability of being agreed on in the ARTigo Game. However, tags on this level are wanted input for complementary games of the gaming ecosystem. The third level of agreement regards tags that were entered in previous game rounds and that were entered in the current game round like "Heroes" in the example above. Agreed tags on this level are weakly reliable. Experiences have shown that the weak reliability can be sufficient for a search engine. A strategy to raise the level of reliability of tags at this level is taking the number of agreements in previous round into account. The highest level of agreement is an agreement in the same game round because the agreement is on one hand temporally coupled. On the other hand the agreement is independent of a strategic selection of game partners by the gaming platform.

The best strategy is to enter tags your partner is most likely to propose, too [BW12a]. For example, one of the two players might know the name of the author of the artwork of Figure 5.3, Currier, or the artwork title, “Naval Heroes of the United States”. However, it is not that likely that a randomly chosen partner knows this. As a consequence, more obvious tags like “United States” are likely to be more successful, in particular because it is written on the right-sided banner of the Figure. The experience shows that most players choose such a strategy.

The tags entered by the other player are not displayed explicitly. The main reason is that players could simply repeat the tags just entered by the other player and earn points easily. This would reduce the ARTigo Game to absurdity. The idea of showing the current number of “Partner Tags” (cf. Figure 5.3 is motivation. Experiments with the ARTigo Game show that a permanently increasing number of tags encourages a player to keep up with entering own tags.

The status bar showing the time left in the current round pursues a similar target. The player should be thrilled to enter as many tags as possible. Such a game design is reasonable for acquiring surface Semantics Tags. Deep Semantics Tags tend to consume more time for considerations about the image. Hence time pressure might have counter-productive effects on acquiring Deep Semantics Tags.

Each game round is concluded by a summary of the game session. Beside tags entered by oneself and the corresponding points, basic meta-data of each image like title, artist, date of creation and current location are shown. This is a kind of feedback for art afficionados and has educational character. The summary could reveal the tags entered by the other player. Since politically incorrect tags could be displayed, only the own tags are displayed in the summary of a game round.

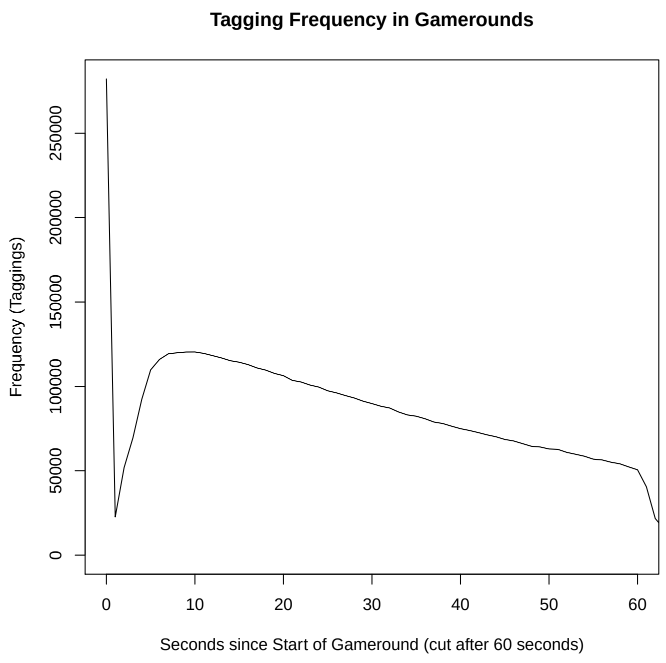


Figure 5.4: Tagging Frequency per Gameround.

An ARTigo Game Session consists of five game rounds. Each game round lasts 60 seconds. Thus, a game session takes five minutes. The number of game rounds is chosen so as to provide a reasonable playing time. The decision for a game round to last 60 seconds is derived from the players’ average behaviour as shown in Figure 5.4. The Figure shows how many tags were submitted during all game rounds played so far grouped by seconds. The peak at the beginning of the plot is caused by direct URL calls by crawlers of web search engines. This is possible because anonymous submission of tags via HTTP/GET calls is allowed. However, such tags can be identified easily to prevent a biased search engine on basis of the tags. The following curve shape comes up with expectations. It takes some time until a player has analysed the image. That is why the submission rate is increasing until a maximum after about 10 seconds. Afterwards to rate of submissions is decreasing almost linearly. After 60 seconds a game round ends because the submission rate is so low, that players' motivation could be negatively influenced. The decay of the submission rate after 60 seconds has technical reasons. A game session ends after 60 seconds at the browser but tags may be persisted later, e.g., due to low transfer speed. The plot in Figure 5.4 is also presented in Chapter 11 in the context of the characteristics of data collected with the ARTigo

Yet, the range of tags collected by the ARTigo Game is probably larger than that produced by the original ESP Game. ARTigo players are art aficionados who appreciate having a close look at artworks and, as a consequence, in general take the time necessary to describe it properly.

For this reason, the ARTigo Game departs from the ESP Game. The ESP Game displays the next image at the first word that matches. The objective is to match on as many images as possible. With the ARTigo Game, the players have to find, and match on, as many tags as they can on each artwork during five rounds of one minute each. They have enough time to look at, and describe, the artworks they are shown.

ARTigo Taboo

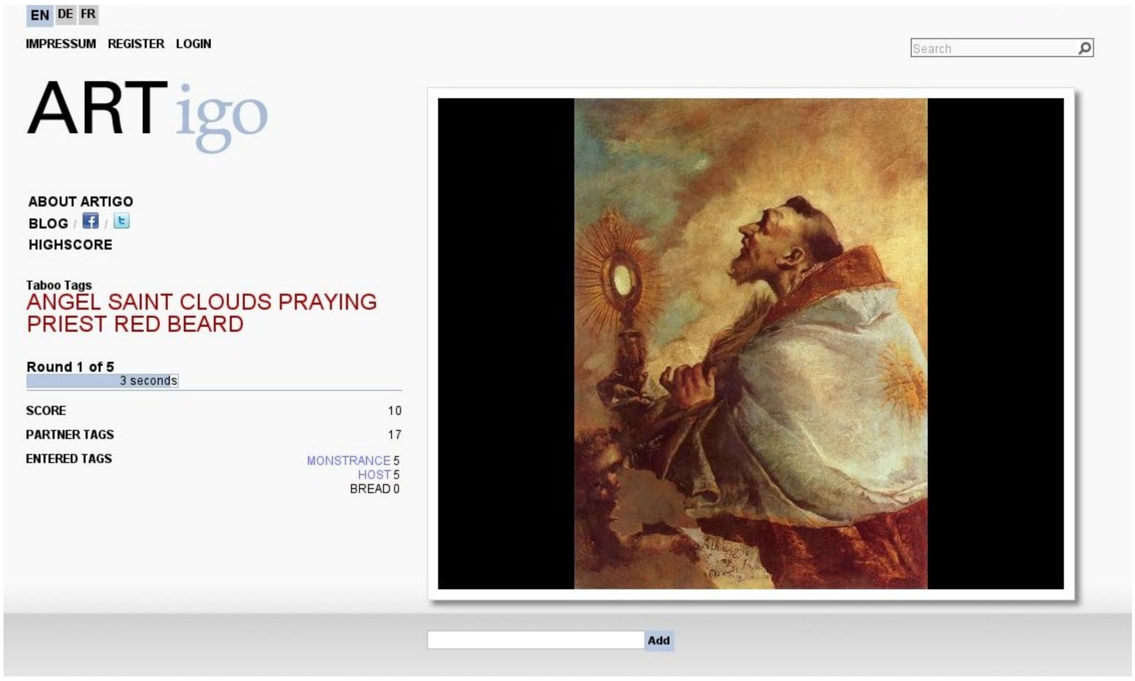


Figure 5.5: F. Guardi, Saint in Ecstasy, 1736/1740, Trento.

ARTigo Taboo is a version of the ARTigo Game offered by the ARTigo platform. The game play of ARTigo Taboo is similar to that of the ARTigo Game except that it prohibits suggesting certain tags formerly entered by other players. As a consequence, ARTigo Taboo forces its players to suggest less common tags than those so far proposed for an artwork. This technique is an instance of Scripting as described in Section 4.4, a way of instructing a player.

Figure 5.5 shows a typical screen during a game round of ARTigo Taboo. In addition to a game round in the ARTigo Game, Taboo Tags are listed in red letters on the left side of the screen. If one the tags “angel”, “saint”, “clouds”, “praying”, “priest”, “red”, or “beard” is entered, it will be denied. In contrast, the tags “monstrance”, “host”, and “bread” were accepted.

It is worth stressing an important complementarity of the data generated by the ARTigo Game on the one hand, and by ARTigo Taboo on the other. While ARTigo Taboo forces to generate less immediate tags, or tags of a deeper semantics, than the ARTigo Game, it does not recognize those tags players frequently associated with an artwork. The ARTigo Game generates such frequencies that are of considerable interest for art historians.

5.2 Diversification Games: Telling Artworks Apart

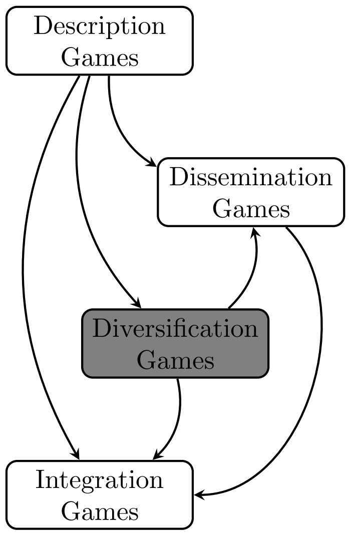


Figure 5.6: Diversification Games in the visualization of the Gaming Ecosystem in Figure 4.4.

Karido

Karido [SWKB11, Ste11] offers a completely different game play than the ARTigo Game. This game play leads the players to enter more specific tags, so-called Deep Semantics Tags. This section is based on [SWKB11] coauthored by the author of this thesis.

Karido is a diversification game since its purpose is to refine the tags collected so far. Played with artworks that have only few tags, Karido produces Surface Semantics Tags like the ARTigo Game but at a lower speed than the ARTigo Game. In order to effectively perform its task, Karido needs a certain amount of surface tags, so that it is run with similar images.

Karido is designed to inherently ensure both tag validity and tag diversity. Karido is a co-operative game in which two players strive to gain as many points as possible by describing the content of images. The two players alternatively assume the roles of Guesser and Describer.

Before each round of Karido, nine similar images are randomly selected from a given database. The selection of these images is crucial to the goal of increasing tag diversity and is discussed in more detail in [Ste11]. The selected images are displayed as regular grids to both players (see Figure 5.7). In the view of the Describer, a single image is highlighted (this is called the *goal*). The Describer’s task is to explain the goal image to the Guesser. To achieve this, the Describer can send short textual messages to the Guesser.

The Guesser’s view shows the same images as the Describer’s, but in randomized order[6] and without a highlighted image. The Guesser’s task is to deduce the current goal image from the given

[6]This randomization ensures that players cannot describe images by their position in the grid. If both players shared the same view, the content of the images would not have to be described.

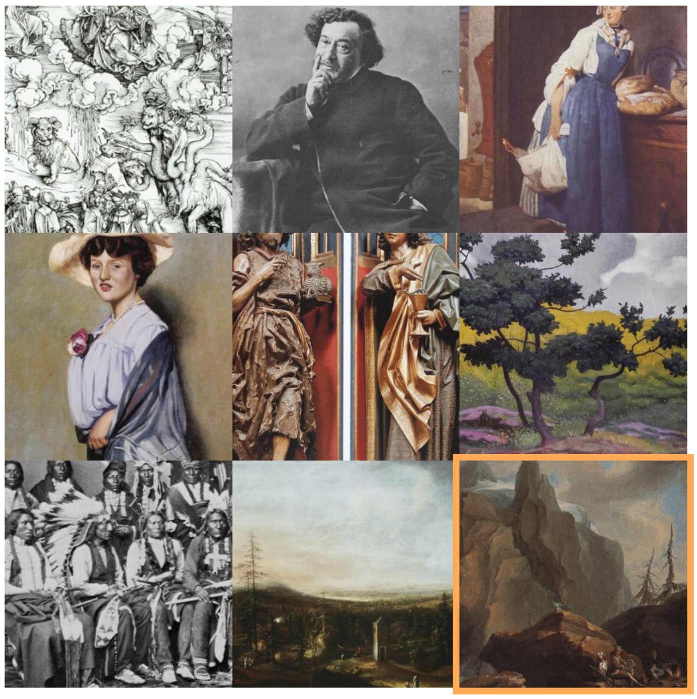


Figure 5.7: Image grid for Describer (Hovering over a picture enlarges the whole image).

description. At any time during the game, the Guesser can select an image by doubly clicking it. If this guess is correct, the image is removed from the grid of both players and both receive a score bonus. Furthermore, the text recently entered by the Describer is considered a valid description of the image. The Describer must then select another image, which becomes the next goal. The game proceeds in this fashion until only one image remains. As selecting this image would be trivial, the current game round is ended at this point, the players switch roles and a new round is initiated.

Karido falls into the class of inversion-problem games, as defined by Luis von Ahn [VAD08]. In these games, one player (the Describer) transforms a given input (the selected goal image) into an intermediary output (the textual description). The second player (the Guesser), tries to transform the intermediary output back into the original input (by selecting the correct image). Success of the players implies that the intermediary output is a representation of the original input.

Design of Karido

The primary goal of Karido is to collect more diverse tags than previous image labeling games, which –as discussed above– struggle to create comprehensive tag sets (cf. Chapter 3). The key element for achieving this goal is what we call "input-similarity". If the images displayed are similar as detected by overlapping taggings, the players of Karido are forced to find differentiating properties of the images. Therefore, players must contribute new information. For example, consider a grid in which each image contains only a unique single color. In this case, the Describer can use the color to clearly describe the goal. In contrast, given a grid in which all images contain a red car, the Describer can no longer use "red" or "car" and thus has to find characteristic traits of the goal image that differentiate this image from all others.

Karido relies on player-created tags as a measure of image similarity. After selecting a random base image, the game calculates the number of tags shared with this base image for every other image in the database. This selection method implies that for a new set of images without any tags, all images are considered equally similar. In this case, the images in the grid are selected randomly and can be expected to be relatively diverse. As a result, players can use general tags to distinguish the images. As a result of these tags, the images now possess different similarity ratings. For example, all photographies in a collection could be tagged “photo”, whereas paintings could be tagged “painting”. Thus, the players will be given grids which contain either only photographies or only paintings. Therefore, the attributes “photo” and “painting” can no longer be used to distinguish the images and the Describers are forced to find new properties of the images.

If any two images have no distinguishing tags, they will necessarily both be selected for a game round at some point. One of the images will become the goal image, while the second image is still in the grid. Thus, the players either have to come up with a label that distinguishes these two images or use random guessing. As the scoring of the game is designed to make random guessing a poor strategy (see below), a player who is able to find a distinguishing feature is likely to use this trait to describe the image. Thus, the process of refinement of the labels continues until all images possess a set of tags that sets them apart from all other images in the game.

Player Communication

Most GWAPs rely on player agreement to verify that the entered data is correct. It is therefore essential that players do not have the possibility of artificially creating an agreement. The most common way of creating such an agreement is by using a communication channel outside of the game. However, Karido does not produce matches on entered tags, but requires selection of an image. Therefore, to circumvent the verification method, players would need to be able to communicate the content of the goal image directly (for example, by using a screen-sharing software). This in turn implies that arbitrary textual communication inside the game could be allowed. However, to ensure that proper keywords for the images are collected (as opposed to free-form texts), descriptions in Karido are restricted to a maximum of three words and all punctuation is removed.

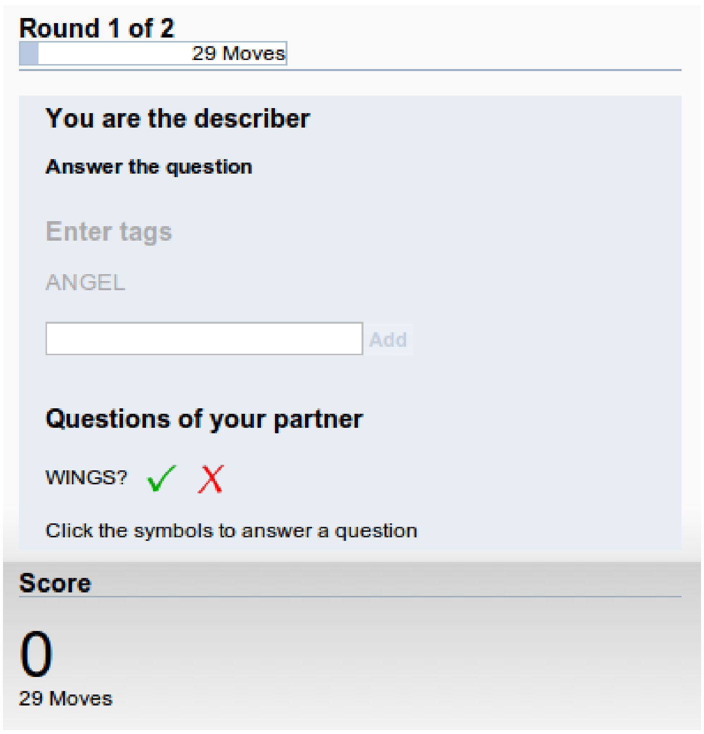


Figure 5.8: User interface of Karido for the Describer.

To allow the Guesser to help the Describer, Karido allows posing Yes/No-questions, which the Describer can answer by clicking on an icon instead of typing (see Figure 5.8). In addition to aiding the Describer, these Yes/No-questions enable the Guesser to actively take part in the labeling process: Whenever a question is answered with “Yes”, the question is added as a label to the description of the current goal image. If a question is answered with “No”, a negative label could be attached to the current image. However, because of the questionable informative value of negative labels (the list of things *not* depicted in any given image is nearly endless), no tags are added for questions answered with “No”.

Game Modes

To make the scores given to players comparable and to provide a goal for the players to achieve, the duration of a game session must be limited. There are several possibilities for limiting the duration of game sessions and game rounds. Limiting the duration of an entire game session is the most commonly used approach for GWAPs. It is also possible to use the inverse approach of limiting the number and duration of the rounds in a game session. This approach is used in Karido. Each game session consists of two rounds. While this seems very little in comparison to the ESP Game, each round in Karido contains nine images and generally lasts longer than a round in other games.

The most straight-forward metric for “duration” is time. Therefore, the first game mode of Karido limits each round to 90 seconds. A second game mode relies on the number of turns performed by the players to limit the duration of game rounds. Sending tags and questions and performing guesses are the atomic actions of the game. By limiting the number of allowed actions, the players can take as much time as they want for considering, typing and submitting their labels. This means that success in the game only depends on the quality of the labels, making the number of actions a reasonable alternative to the metric of time. However, whereas time is always shared equally between the players of a game, more actions are “consumed” by players who take less time. Thus, a quick player can acquire a disproportionate share of the available actions. This introduces a potential competition between the players, which goes against the cooperative nature of Karido.

To enforce an equal distribution of the actions between the players in the second mode of Karido, players are forced to take turns to describe images, send and answer questions and venture guesses. However, there are scenarios in which a rigid succession of player turns can be a problem. Occasionally, only two images are potential candidates for the Guesser. Thus, if the Guesser takes her turn and guesses wrong, the correct solution is immediately obvious. Using a strict turn-based approach, the players would have to waste another action (the Describer’s turn, which has become superfluous) and the Guesser would have to wait her turn until she could perform the now trivial action of selecting the right image. To avoid such situations, the Guesser in Karido can take a guess at any point in the game. While this approach allows the guesser to take a disproportionate share of the available actions, we consider it an acceptable trade-off for an improved game flow.

Scoring

In other GWAPs such as the ESP Game, both players enter their answers in form of textual labels. Therefore, the set of possible answers contains all character sequences and thus millions of items. Thus, the probability of a player being successful by chance or guessing is very low. In contrast, the Guesser in Karido can only select from a given set of images. At most, this set comprises nine images (at the beginning of a game round). It would therefore be easy for a player to try all possible answers. This would eliminate the verification of the entered tags, as a player relying solely on guessing can ignore the description given by her partner. It is therefore necessary to take measures that discourage random guessing.

In Karido, the score of both players is reduced as a penalty for selecting a wrong image. This penalty exceeds the bonus for selecting the correct image. Therefore, the expected value of the score for a player relying on guessing is always negative. However, malicious users could still cooperate to introduce wrong data into the system by entering wrong descriptions and using random guessing to verify the entered data.

Data Verification in Karido

To ensure that the collected descriptions are valid, Karido assigns a real-valued relevance score to each pair of an image and a tag. When a tag is first applied to an image, this score is initialized with zero. Once a player correctly guesses a goal image, the score of all tags in the description leading to the correct guess is increased. To avoid gathering wrong data, the amount by which the scores are increased depends on the number of wrong guesses performed before the selection of the goal. If the Guesser tried more than 30% of the images before selecting the right one, all tags are considered irrelevant. For example, at the beginning of a game (with nine images remaining), the Guesser must be right on the third try or the labels will be discarded. The full score increase is only assigned to a tag if the Guesser is right on the first try. Otherwise, the increase is interpolated linearly between 1 and 0 depending on the percentage of wrongly selected images (from 0% to 30%).

In contrast to the ESP Game, Karido can collect several labels for an image in one round of the game. The Describer usually enters several tags before the Guesser correctly selects an image. Thus, the labels entered first were not sufficiently precise to enable the Guesser to select the correct image. It is therefore reasonable to assume that the labels assigned later are more relevant to the image. Karido uses a weighting scheme to take this increasing relevance into account. In addition to a base weight of 0.5, an additional weight of 0.5 is distributed over all tags in a linearly increasing fashion. The result is multiplied with the factor calculated above and added to the existing verification score.

Simulated Player in Karido

Although most GWAPs are advertised as multi-player games, many implementations [VAD04, VAGK$^+$06, VAKB06, VALB06] include simulated players (or *bots*), which enable single (human) players to participate in the game. There are two reasons for supporting single-player operation. Firstly, the number of players in the game can be uneven, leading to players who cannot be paired with a partner. Additionally, a single person may be the only player of the game at any given time and thus would have to wait for a second player to join the game.

The bot of Karido must fulfill two different roles in the game, Describer and Guesser.

The Describer's task is to explain the current goal image and answer the Guesser's questions. The first task can be achieved by replaying previous rounds. The tags of the goal image are sent using the same delays and in the same order as in the recorded round. Once the right image has been selected, the simulated Describer chooses a new image. To ensure that all tags are replayed in the proper context, the simulated Describer selects the same sequence of goal images that was played in the previous round. If a human Guesser takes longer than the Guesser in the original round to select the correct image, there are no further tags that can be replayed. In this case, the simulated Describer selects random tags and sends them with randomized delays. In addition to sending descriptions, a simulated Describer must be able to answer questions posed by the human Guesser. To achieve this, the bot relies on the labels already assigned to the current goal image. If the content of a question has ever been used to describe an image, the question is answered positively. If no label with the content of the question exists, the question is answered negatively.

Like the Describer bot, the Guesser bot in Karido must fulfill two tasks: Firstly, it must interpret the descriptions given by the Describer and select an image once it is reasonably certain that it is the goal image. Secondly, if several potential images remain, it should ask questions to reduce the number of candidate images. For both tasks, the bot relies on the tags previously assigned to the images. As described in Section 5.2, the behaviour of the simulated Guesser does not directly influence the quality of the data collected in the game, as all tags are validated by an independent player. However, if players discover that they can score points as Describers without entering valid descriptions, no more valid information will be entered into the system and the verified information will stagnate. It is therefore necessary to simulate a Guesser that only selects the goal image if it has been described accurately.

As a first step, the percentage of entered tags assigned to the images in the grid is calculated. This value will from now on be called the *match percentage* of an image. For example, if the tags “car” and “dog” have been entered, an image tagged “car” has a match of 50%, an image tagged “dog”, “house”, “car” has a match of 100%. All images with a match percentage larger than zero (and which have not yet been wrongly guessed) are sorted by decreasing match percentage. The resulting list of images is used to decide which image to select. The first image from the list is selected if the list contains only one image, contains only one image with a match of 100% or if its match percentage is at least 1.2 times larger than the one of the next image. Additionally, the first image is selected if a sufficient number of descriptions have already been sent. This ensures that the Guesser bot starts to guess randomly if the Describer fails to refine the description.

If none of these criteria are applicable, the simulated Guesser prepares to ask a question. All images which have at least half the match percentage of the first image are used to ask a question. The simulated Guesser selects the first image from this list of candidates. From all tags assigned to this image, those which have been applied to any of the remaining candidate images are discarded. The remaining tags are unique for the selected image. One tag is randomly selected from the three most common tags of this list and sent to the Describer as a question. This approach yields questions which are well suited to discriminate the candidate images. At the same time, the selected questions tend to be obscure.

Lastly, answered questions must be used to reduce the number of candidate images. All questions which were answered positively are treated like tags created by the Describer. In contrast, questions which were answered negatively are added to a list of blocked tags. For each candidate image, the number of blocked tags assigned to this image is calculated. This value is subtracted from the number of matching tags used for calculating the match percentage of the images.

A problem arises if the current goal image has not yet been tagged a reasonable number of times. In this case, whether a tag entered by a human player is valid cannot be determined and therefore the Guesser bot must make a trade-off. Either unknown tags are accepted with a high probability, at the risk of inciting players to enter wrong tags, or unknown tags are rejected with a high probability, effectively punishing players for entering new tags. As any data collected by the game is independently verified, the first alternative in Karido is chosen.

5.2.1 Case Study

To assess whether Karido fulfills its goal of collecting more diverse tag sets, an empirical evaluation was performed. On 13th April 2011, the ARTigo platform replaced the previous version of the ARTigo game. Karido was released alongside the new version of the ARTigo game. For the first three weeks after the release, no public announcement of the update was made. Therefore, the users of the game primarily consisted of regular players of the old ARTigo game, returning to the Web site. On 3rd May 2011, announcements of the newly released games were made on several channels (among others, the Web site of the University of Munich). Another three weeks later, a snapshot of the database of the games was taken on 24th May 2011. This snapshot has been used to gather the results described below.

During the evaluation period, 664 players completed at least one round of Karido. The participants played a total of 1939 game sessions, consisting of 3542 rounds. In the same timespan, 3766 sessions of ARTigo were played. The higher popularity is likely caused by the large number of regular players of ARTigo participating in the evaluation.

Results

An analysis of the number of played rounds for individual players reveals that the majority of players completed only two rounds. This number seems to indicate that players do not enjoy playing the game. However, this contradicts the high satisfaction scores of the game (see below). An alternative explanation can be found in the way players are identified by the game. Players may choose to play the game without registering. In this case, only the number of played rounds during one visit to the Web site can be counted. This number must necessarily be less than or equal to the total number of played rounds of a player. The number of registered players is too small to draw reliable conclusions, however, the distribution of played rounds of registered players appears to be more even than the distribution of played rounds of all players.

To measure the subjective enjoyment players gain from Karido, a rating mechanism was added to the user interface. After each completed game session, players are invited to "rate this game session". To submit their rating, players can select from a scale of five stars. This scheme is commonly used to express ratings from dislike (one star) to approval (five stars). A total of 1051 ratings were created. As described above, 1939 game sessions were played by a human and a simulated player, causing at most one rating per game session. Thus, a rating was submitted for 54% of all sessions.

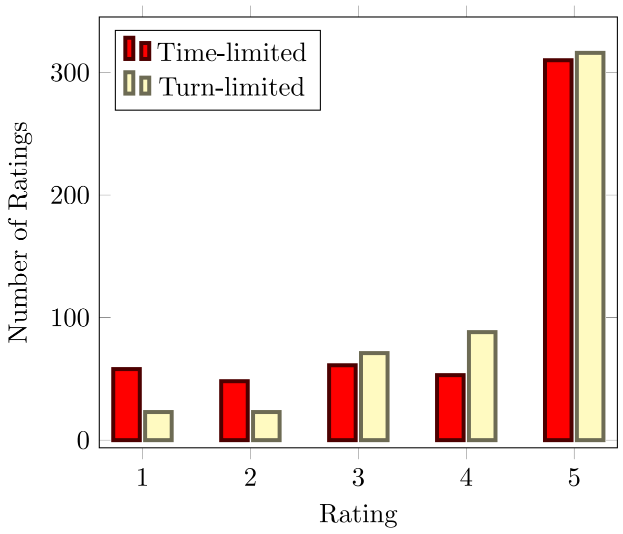


Figure 5.9: Histogram of the game session ratings submitted during the evaluation.

Figure 5.9 contains a histogram of the submitted ratings for both modes of Karido. In contrast to the low average number of played rounds (possibly indicating low acceptance of the game), the submitted ratings show a strong peak for the maximum rating of 5. The turn-limited mode is slightly more popular, with an average rating of 4.2 (as opposed to 4.0 for the time-limited mode). It should be noted that the design of the rating interface might introduce a bias: As described before, the rating interface is displayed after each completed game session. Therefore, players who do not complete a session do not get a chance to vote. Additionally, players who dislike the game and stop playing only get to vote once, whereas players who like the game and play many sessions can vote more often. To eliminate the influence of multiple votes, the collected ratings are averaged in a grouped fashion. First, the average rating for each player is calculated. Then, the average of these ratings is calculated, attributing an equal weight to the votes of each player. This results in an average score of 4.0 for the time-limited mode and 4.4 for the turn-limited mode. As returning players cannot always be identified correctly, a bias might remain in the grouped data. A comparison between Karido and ARTigo is not possible, because no game session ratings were collected for the latter.

One problem that became apparent during the evaluation is the player matching mechanism. If a human player has not been matched with another partner after ten seconds, she is automatically matched with a simulated player. Even with a relatively large number of concurrent players, the probability of two players being matched in this ten second window is quite low. As a result, all rounds in the evaluation were played by a human and a simulated player. However, the high satisfaction ratings of the game indicate that the simulated player –while certainly no replacement for an actual human– is adequate at the very least.

The primary goal of Karido is to collect more diverse sets of tags. To evaluate whether this goal was reached, we consider the average number of unique tags created per game round. We define a tag to be unique if it has not been previously assigned to any image. During the six week evaluation period the players created 6933 unique tags in Karido and 16635 unique tags in ARTigo. This corresponds to 2.0 (Karido) and 1.2 unique tags per round respectively. Thus, Karido arguably leads to a higher rate of unique tags per round. However, this rate can be expected to decrease when more tags are submitted, thus favoring Karido (because of its lower number of tags). To compensate for this, we have also calculated the average number of unique tags during the first three weeks of the trial. It amounts to 2.1 (an increase of 5%) for Karido and 1.6 (an increase of 33%) for ARTigo. This indicates that Karido indeed collects more unique tags and thus more diverse tag sets.

The evaluation has shown an issue concerning the verification of entered tags. Of 11372 tags (unique per image), only 2364 posses a verification score larger than 1.0. This behavior is to be expected: Usually, each player alternates between creating tags and verifying tags. Thus, as long as the creation rate of unique tags remains high, these tags are –on average– only verified once. See the next section for our proposed solution.

5.3 Integration Games: Gaining Deep Semantics

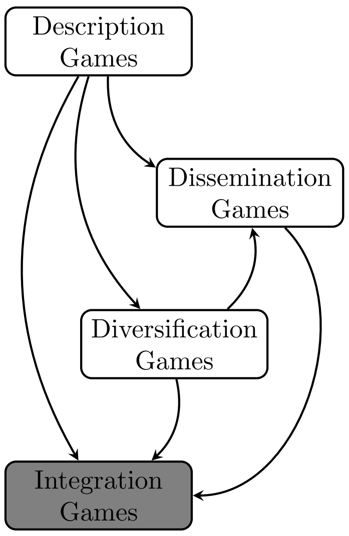


Figure 5.10: Integration Games in the visualization of the Gaming Ecosystem in Figure 4.4.

Combino

Combino sets tags into relation. While playing games like the ARTigo Game users sometimes have phrases (a group of words) in mind as a possible description, that are likely to split into single words and to suggest each of these single words independently of the others. Indeed, doing this increases a player's probability to get a match.

Given Figure 5.11, which shows a book page, a player of the ARTigo Game, of Karido or of ARTigo Taboo will rather enter the tags "Latin" and "text" instead of the tag "Latin text" to increase the probability of a match with input of the other player. Combino is designed to combine tags to phrases as acquired by description games.

Combino's input consists of an image and a sequence of tags that belong to this image. The user has to build pairs of tags that are semantically linked. The output of Combino consists of

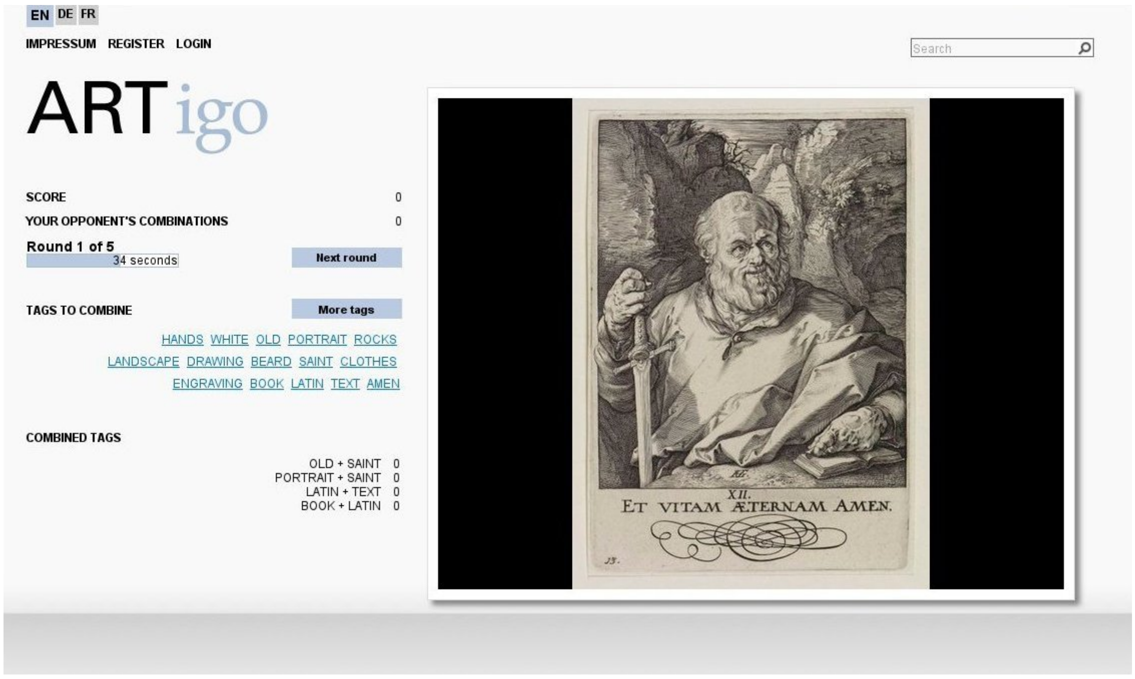


Figure 5.11: H. Goltzius, Christus, the 12 Apostles with Paulus, 1589, Karlsruhe/State Art Gallery Karlsruhe.

tuples of the form (tag, tag) for an artwork. Using tuples preserves the order of tags from the input sequence.

Like the ARTigo Game, Combino is an input agreement game. Two players need to agree on relations between provided tags. This game design is supposed to ensure an output of semantically related tags, only. Note, that the actual relation such as "refinement" for ("latin", "text") or "location" for ("head", "center") in Figure 5.11 is not recorded. Information of this kind is acquired by other games of the ARTigo platform like Tag A Tag.

It is remarkable that Combino can be played by clicking or tapping on the screen, only. This makes Combino easier to use with mobile devices like smartphones. Other games like the ARTigo game rely on quick typing, which is difficult with the small keyboards of smartphones.

Game design relying on clicking or tapping as only input for an input agreement game like Combino must ensure that a maximum number of possible combinations can be selected. Otherwise, if all combinations could be selected by both players, the game would become useless. Combino offers 15 tags that can be combined. Hence $\binom{15}{2} = 105$ combinations could be selected. If a game round lasts 60 seconds as in Combino, about 2 combinations would be necessary per second to generate all combinations, which makes it unlikely that the game becomes a useless exhaustive search.

Combino is a good candidate for the squaring technique proposed in Section 4.3. A squared Combino or Combino^2 would allow tags like ((old, man) sitting). Further squaring in this manner would allow easily to construct sentences or even stories told by an artwork based on surface or Deep Semantics Tags.

Tag A Tag

Tag A Tag is the result of squaring the ESP Game (cf. Section 4.3) and scripting the ESP Game (cf. Section 4.4). As the name Tag A Tag implies, tags are tagged, or more precisely tuples consisting of a tag and an artwork are tagged. This is the squaring aspect of the game.

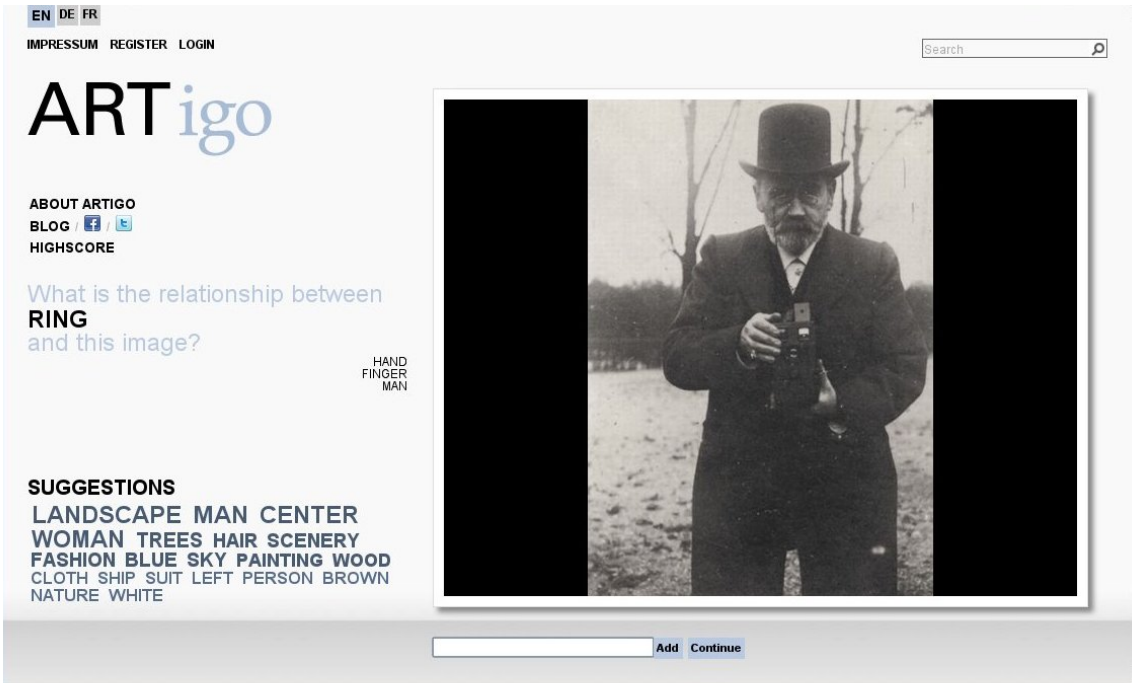


Figure 5.12: E. Zola, Zola and his “Box”, location and data unknown. Zola is wearing a ring on a finger on his right hand.

Unlike Combino, Tag A Tag acquires semantic relations between tags. This is achieved by scripting, which determines the semantic relation. In Figure 5.12 the player gets asked for the “relationship” of the tag “ring” and the displayed image. A player in Tag A Tag is presented an artwork and a single tag (like “ring”) formerly assigned to this artwork. Based on this setting, the player is instructed to enter tags related to both, the artwork and the tag displayed (cf. Figure 5.12) such as the tag “finger”.

Tag A Tag can also be seen as a diversification game. It confines the description to one specific entity to be described and thereby sharpens the description. For example, given the image of Figure 5.12 and the tag “ring”, the player has to describe the ring instead of the whole image. He could then provide more specific words about the ring’s appearance, such as “hand”, “finger”, and “man”.

Sentiment

Sentiment is a version of Tag A Tag that focuses explicitly on certain Deep Semantics Tags by asking directly, which feelings are expressed in an image. The only difference to Tag A Tag is the question asked. This is rather trivial but in the context of ARTigo dealing with artworks asking purposeful questions is an important means to enrich the descriptions of artworks in sparsely supported fields like feelings.

Despite the simplicity of Sentiment, it is one of the most difficult games to play because empathy is one of the hardest problems in Human Computation and needs a lot of experience.

5.4 Dissemination Games: Spreading Information

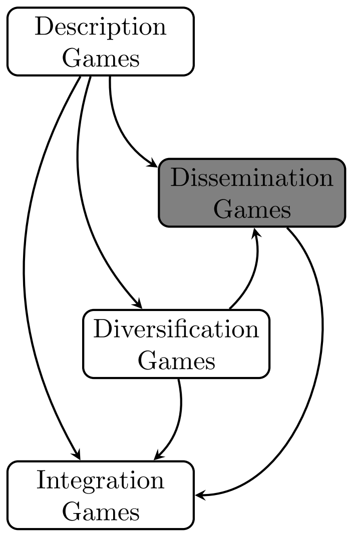


Figure 5.13: Dissemination Games in the visualization of the Gaming Ecosystem in Figure 4.4.

Eligo

Eligo is a game that is not yet available on the platform because it is still undergoing a test phase. Eligo's purpose is to translate already collected tags in one language into other languages.

Translating words from one language to another is far from being easy. For example, the German word "Himmel" can mean, heaven or sky depending on the context. Translating tags automatically with a good accuracy would probably require as much effort as trying to analyze the image in the first place, what would suffice to fully describe it. Instead, the ARTigo gaming ecosystem relies on Human Computation with Eligo.

Eligo players are presented with several artworks as shown in Figure 5.14. Some of these artworks have both a common tag like "Himmel" in the input language and other tags that are not shared by all images. The common tag is translated to the output language using a dictionary like http://www.dict.cc/. Eligo players have to select the artworks that correspond to the translated tag.

The choices of Eligo players are validated as usual by agreements and, in case of validation are rewarded scores. A negative scoring, in case of disagreement between the players, ensures that players neither select random images.

Validating a tag proposed by a game is much easier, and therefore quicker, than suggesting a new tag. As a consequence, playing requires less human work to produce an equal amount of tags and the Eligo game play is characterized by a particularly high speed, thus contributing to the diversity of the entertainment the ARTigo gaming platform offers.

It is worth stressing that Eligo can be used to produce seed tags in a new language, that is, the initial collection of tags necessary for the games to be playable. Thus, Eligo can be used as a seed game for porting ARTigo to further languages. (Such portings, for example to the Arabic language, are currently considered.)

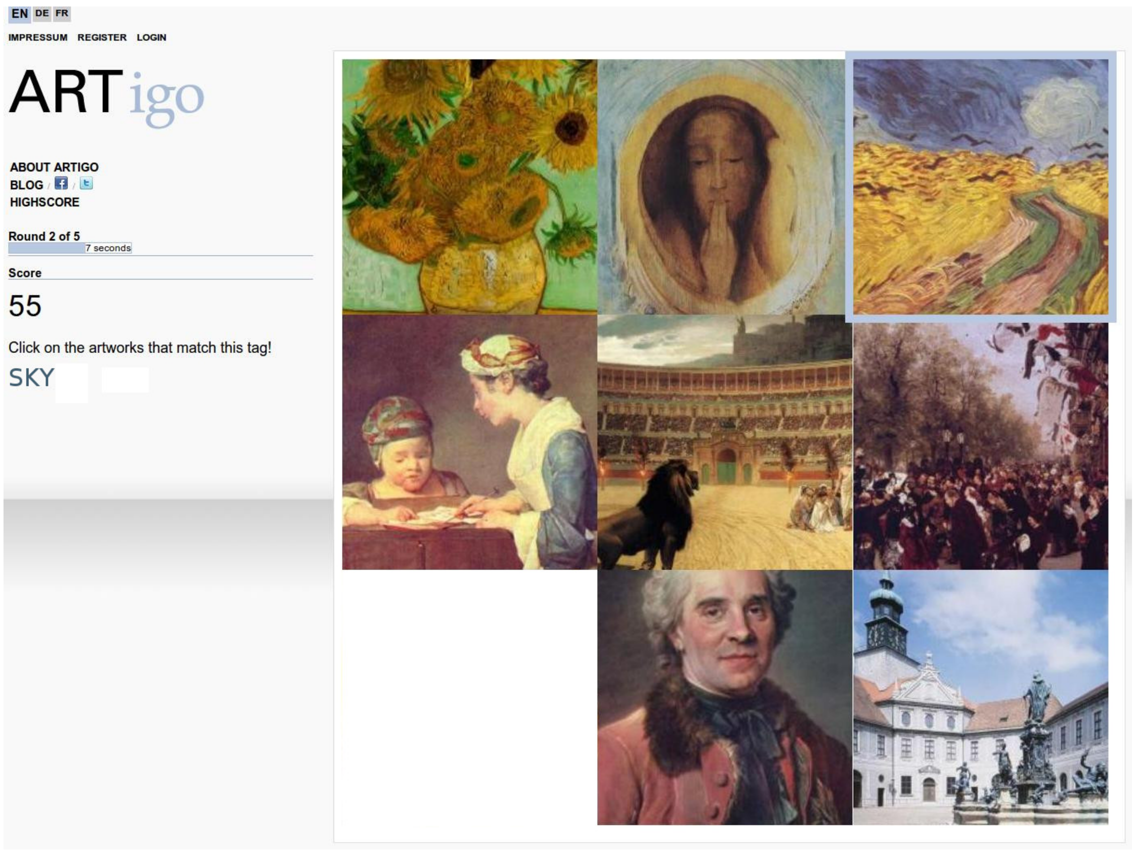


Figure 5.14: Screenshot from a game round in Eligo.

Chapter 6

Perspectives for Future Work

This part of the thesis proposed game design techniques, which were applied in several games in the ARTigo platform for collecting semantically rich tags. The tags aim at building a foundation for a semantic search engine. However both, the game techniques and the games proposed in this thesis make no claims of being complete. It is highly probable that new games and gaming concepts can be envisioned in addition to the games offered on the ARTigo platform and to related games conceived so far. This chapter first points to opportunities to extend and enrich the gaming ecosystem described above. That is an obvious perspective for future work. This chapter further suggests another way to acquire semantically rich tags looking beyond games to other sources of information like comments on artworks.

6.1 Improvements of the Gaming Ecosystem

The gaming ecosystem by now proposes classes of games interconnected by a tag flow. A look at the gaming ecosystem on this level of abstraction reveals opportunities for future work concerning guidance for gaining a balanced artworks index consisting of the different kinds of tags.

Game and Tag Selection based on Tag Evaluation

Image retrieval lives off diverse tags acquired for artworks on the ARTigo platform. When starting such a gaming platform, Surface Semantics Tags are the first ones to be acquired. This is easy to understand because most games acquiring Deep Semantics Tags rely on the input of Surface Semantics Tags. On the longer term a strategy is welcome to keep the balance between the different kinds of tags. The games perceived in the ARTigo platform after the initial ESP Game focused mainly on acquiring all but Surface Semantics Tags that have been collected until then.

Choosing a sophisticated sequence of games for balancing the collection of tags is bound to preconditions. First, the current shares of tag types in the ARTigo platform must be determined. This is not an easy task to perform because the distinction, e.g., between Surface Semantics Tags and Deep Semantics Tags is blurred. Second, a verification of the games' output is needed, which is expected to confirm the experiments run so far. On basis of an exhaustive evaluation of the data collected so far, not only sequences of games can be arranged but also the input for those games can be controlled according to the current needs.

Educated Selection of Players for Game Sessions

Many image labeling games rely on the cooperation of players like in input agreement games as the ARTigo Game. An assumption is that the success of such games depends on the compatibility of players. For example, an Art Historian would use another vocabulary to label images compared to a philistine. As consequence, the rate of matching tags would be rather low. A perspective for future work could be an educated selection of players for game sessions. This could be rather simply done based on information offered by players in their profiles on the ARTigo platform.

A more sophisticated way would be a selection based on the players' behaviour. Similar players could belong to same clusters that could be used as pool for selections. Many techniques for clustering users come into question. Arguably, a good starting point for clustering could be exploiting higher-order latent semantics as analysed via Higher-Order LSA (cf. Part III). As a vector space technique, this method allows to measure the similarity of users (beside the similarity of tags and the similarity of documents/artworks). Having a recipe for similarity, established clustering algorithms can be applied for finding promising game partners. For the success of some games, even the selection of as different as possible players might be the best tactic. This can be easily done with a clustering of users by selecting players from different clusters.

ARTigo selects players for game sessions randomly. If only one player is available, simulated sessions are offered either replaying old game sessions or using simulated players. Obviously, replaying old game sessions is only possible for games where players act independently from each other. Naturally, the selection of replayed sessions can be

6.2 Extension of the Gaming Ecosystem

Garbage Collection Games

Collecting tags with gaming ecosystem is based on continuously adding tags. According to the law of large numbers [Wik13f] one could expect, that with an increasing number of tags per image a good description could be achieved. The experiences with the ARTigo platform and especially with its search engine support this hypothesis. Nevertheless, another class of games in the gaming ecosystem could be "Garbage Collection Games" for identifying undesired tags. Instances of this game class would most likely not be standalone games but rely on tags acquired by preceding games. Currently garbage collection is done only by statistical noise reduction as described in Part III. Games supporting garbage collection could help to make the search engine retrieve results of higher quality.

Highly Integrated Tag Structures

Another promising field for extensions of the ecosystem is the class of integration games (cf. Section 5.3). So far, the first steps towards structured tags have been made. Combino couples two tags, Tag A Tag and Sentiment tag the relation between a tag and an image. Further steps could be higher levels of integrations like tagging tagged relations. For example, it could be expressed, which part of an apple is harmless, a level of expressivity that might have helped Snow White. Extending the gaming ecosystem by integrating data or tag structures should not be an end in itself. Instead, the benefit, e.g., for semantic search should be in the foreground. Currently, the ARTigo platform does not exploit integrated tags as acquired by Karido or Tag A Tag. However, this fact belongs to perspectives of future work in data analysis.

Absence of Information is Information

Deep Semantics Tags as well as Surface Semantics Tags –integrated or not– have in common that they express what can be sensed by a player while looking at an artwork. What is not expressed by the tags acquired so far is what is not sensed. In other words, the tags of the ARTigo platform are positive. Indeed it is very unlikely that two players playing the ARTigo Game would accidentally find the same tag in a game round if it is not expressed by an artwork. This might seem trivial at a first glance, but information on what is not expressed by an artwork is highly welcome for search engines. In many cases and particularly if searching for artworks catchy queries consisting of positive tags can be hard to ask.

For example the query "+war -dead" yields satirical illustrations about war. The plus sign in front of a query term ensures that only images are retrieved having the query term assigned.

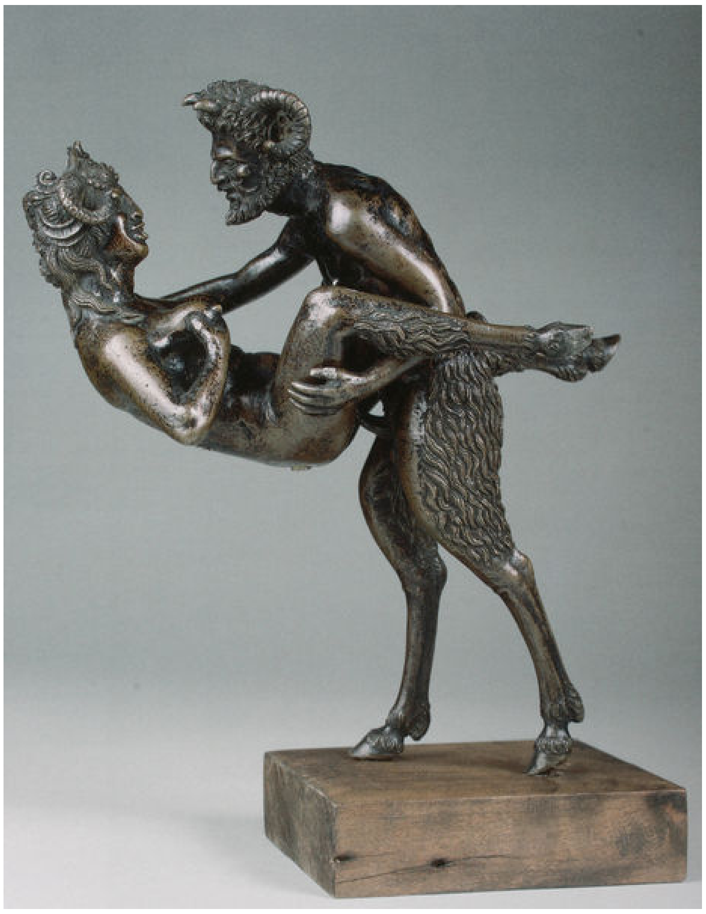


Figure 6.1: D. da Firenze, Male and Female Satyr, 1525/1535, Ecouen/National Museum of Renaissance

Vice versa, the minus sign means absence of a tag. Another query asking "+sex -love" yields pornographic artworks such as in Figure 6.1 although the term "pornography" was not used as tag for the image. Queries like this or further queries like "+sea -blue" are most helpful tools in image retrieval. Making relevant negated information explicit could be a challenging class for proposing new games called negation games.

Turning negated information into positive tags could be realized using data analysis combined with a scripted input agreement game. The task of the players is to select (not to enter) tags that are not expressed by an image. The tricky part of such a game is finding negative tags that are not trivial. A good candidate for finding such tags is association rule mining [AS+94]. Association rules express co-occurrences of terms. This is used, e.g., by web shops for offering further products: If eggs, lard, sugar, and milk are already part of your shopping basket, the vendor could suggest to buy also flour and saffron for baking a cake. Such rules can be derived from the corpus of the 6.722.045 tags collected so far via the ARTigo platform.

Saturation Games

Association rule mining [AS+94] is not only a candidate for making negative tags explicit. Association rules can also help to saturate descriptions of images that are still missing, maybe because an image was just added to the ARTigo platform a short time ago. Suggesting tags has substantial advantages over the classical ARTigo Game (but in contrast relies on previously collected tags). First, only tags that are not associated with an image can be proposed. Second, tags can be proposed according to their estimated relevance for an image as determined via data analysis. This relevance should be decreased, if a tag was refused by players. And third, selecting tags instead of typing makes games better suited for mobile devices.

6.3 Not Everything is a Game: Non-Game Data Sources

Aside from gaming sources for acquiring tags, other sources seem to be suitable for extending the ARTigo platform. Such non-game data sources focus on communication between the users of ARTigo, while most games of the platform try hard to avoid any communication between players. A welcome aspect of communicative features in ARTigo is offering the users incentives (1) to stay as long as possible on the platform which is known as prolonging screen time and (2) to raise the rate of recurrence on the ARTigo platform.

Exploiting Comments on Artworks

So far posting comments on images is not supported on the ARTigo platform. The main reason is that –unlike in an input agreement game– comments would need to be released manually to avoid legal issues. Comments have interesting characteristics that could be exploited for labeling images.

Comments are usually sentences. The field of Natural Language Processing offers several techniques starting from stop lists and ending with complex statistical approaches to analyse sentences. These techniques could be applied to distill tags from comments.

An interesting aspect of comments is that the incentive comes for free. Posting comments is not competition as in games but communication. The design of games requires detailed consideration and esprit to find a reasonable incentive. Harnessing the natural need for communication seems to be promising as easy way to motivate.

Exploiting User Generated Collections

Another extension of the ARTigo platform could be user generated collections of images complying with personal interests of users like architecture or post-impressionism. Obviously, such collections would be perfect for posting comments as proposed above. However, from the perspective of data analysis collections can be seen as clusters. In other words, user generated collections are an approach for a clustering based on human computation. Clustering is usually a computationally intensive problem. Based on human computation the main remaining questions would be how to distribute the work to a community and how to get the desired clusters or collections.

Part III

Data Analysis for Semantically Rich Annotations

On the way to building a Semantic Search Engine, the metadata acquired as described in the previous Part II about items/images provide the basis, or metadata on images, for answering queries. This part addresses how these metadata can be used for a semantic search or more precisely, how a mapping of human semantics to syntax can be computed automatically (cf. Part I).

Relying on tags produced by humans as metadata for search brings some problems:

1. Trivially, only those items can be retrieved that have been tagged by humans before.
2. Tagging a large collection of items is only feasible for a large group of people. Such a group can divide the workload to accomplish a reasonable coverage with metadata in a reasonable time. The ARTigo platform, e.g., hosts 46.176 images. Tagging this amount of images would not be reasonable for one person or a few persons.
3. The choice of tags for an image depends on the person tagging. This results, e.g., in synonymous taggings for one item.
4. Tagging is subjective making mistakes and fraud possible.

The first problem mentioned above has been addressed in Part II by means of human computation. Automatic approaches for image tagging belong to the research field of Content-based Image Retrieval (CBIR), which is "still a widely unresolved problem" [Deb04]. The remaining two problems are addressed in this part of the thesis.

This part is structured as follows. First, LSA as method for realizing semantic search is described both, in its original version by Deerwester et al. [DDF$^+$90] and in the new integrated version called Higher-Order Latent Semantic Analysis (HO-LSA). Second, Singular Value Decomposition (SVD) [Ste93] is developed, on which LSA is mainly based, followed by Higher-Order Singular Value Decomposition (HO-SVD), on which HO-LSA is mainly based. Remarkably HO-SVD builds on (matrix) SVD. Third, HO-SVD is evaluated using image compression. Forth, HO-SVD is parallelised and distributed saving calculation time. Note that realizing HO-SVD is the main challenge on the way to HO-LSA.

Some passages of this part are based on publications coauthored by the author of this thesis. Section 7.3 and Chapter 8 are partly published in [SWB12]. Section 10.3 reports on Diego Havenstein's contribution to parallelising HO-SVD resulting from his bachelor thesis [Hav12] co-supervised by the author of this thesis. Finally Section 10.4 is based on [LWB13].

Chapter 7

Latent Semantic Analysis (LSA)

Indexing by Latent Semantic Analysis (LSA[1]) by Deerwester et al. [DDF+90] is an approach that analyses the "contextual-usage meaning of words" [LFL98], which can be seen as mapping of human semantic and syntax as requested in the introductory Part I for building a semantic search engine. Related work concerning LSA such as pLSA and LDA is discussed in Section 2.2 in the context of machine learning approaches for semantic search engines.

This chapter reports on the standard approach of LSA in the first section. The second section proposes Higher-Order LSA (HO-LSA), which is based on standard LSA.

7.1 Brief Introduction Into Standard LSA

Standard LSA is supposed to answer queries such as "human computer interaction" by returning semantically related documents of a repository. For instance, such a document repository contains technical memos as in Figure 7.1.

memo id	memo content
c_1:	**human** machine **interface** for Lab ABC **computer** applications
c_2:	A **survey** of **user** opinion of **computer system response time**
c_3:	The **EPS user interface** management **system**
c_4:	**system** and **human system** engineering testing of **EPS**
c_5:	Relation of **user**-perceived **response time** to error measurement
m_1:	The generation of random, binary, unordered **trees**
m_2:	The intersection **graph** of paths in **trees**
m_3:	**graph minors** IV: Widths of **trees** and well-quasi-ordering
m_4:	**graph minors**: A **survey**

Figure 7.1: Technical memo example consisting of a repository with nine memos [DDF+90]. The bold highlighted terms are taken into account by LSA.

A full-text search for "human computer interaction" would retrieve the documents c_1, c_2, and c_4, because these documents contain at least one term of the query. The documents c_3 and c_5 would be missing in the results of a full-text search because none of the query terms is contained in those documents. However, it is easy to grasp for humans that the documents c_3 and c_5 would be valid results as well – not on the syntactic but on the semantic level. The reason is that "user interface" and "user response time" is semantically closely related to "human computer interaction" in contrast to , e.g., "graph" (cf. documents m_3 and m_4) and "survey" (cf. document m_4). Remarkably, LSA retrieves the semantically related documents c_3 and c_5 of the example as well.

[1]LSI for Latent Semantic Indexing and LSA are equivalently used as abbreviations in literature.

Term-Document Matrix

The first step of LSA is generating a term-document matrix from a repository for data analysis. The term-document matrix in Figure 7.2 represents the repository of technical memos in Figure 7.1. An entry of the matrix indicates how often a term (identified by the entry's column) occurs in a document (identified by the entry's row). For example, the term "system" occurs twice in document c_4. Some of the words in the repository like "for" and "of" are so called stop words and are not considered in the term-document matrix. For keeping the example simple, other meaningful words like "machine" and "management" are also treated as stop words. The selection of stop words is application-dependent.

$$\begin{array}{l} & c_1 & c_2 & c_3 & c_4 & c_5 & m_1 & m_2 & m_3 & m_4 \\ human & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ interface & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ computer & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ user & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ system & 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ response & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ time & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ EPS & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ survey & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ trees & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ graph & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ minors & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}$$

Figure 7.2: Term-document matrix of Figure 7.1.

The term-document matrix is used to analyse the semantics of *terms* (rows) in documents (columns) based on row similarity via SVD (cf. Section 8.1). An intuitive illustration of row similarity is given in Section 9.5 comparing two (pixel) rows of an image as application for SVD instead of LSA.

Note that the transposed term-document matrix is the document-term matrix (cf. vector space model [SWY75]) as used, e.g., in PageRank [PBMW99].

Principal Components

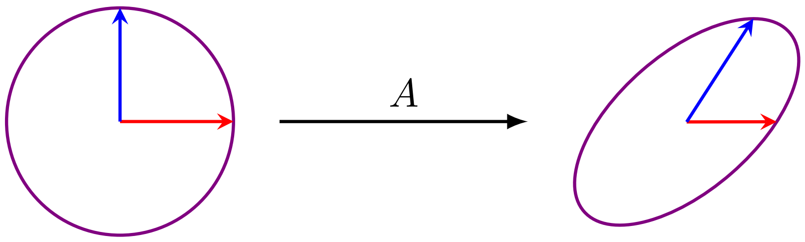


Figure 7.3: Two-dimensional, real shearing by matrix A.

In the second step of LSA, the principal components of the term-document matrix are computed, which can be determined via SVD, a factorization of a matrix (cf. Section 8.1). The first principal component represents the largest variance of column/document-vectors, the second principal component represents the second largest orthogonal variance and so on.

The term-document matrix in Figure 7.2 is well suited for demonstrating LSA. However the matrix is too complex for a visualization of principal components. A smaller, alternative example for the visualization of principal components is the following (arbitrary) matrix $A = \frac{1}{9}\begin{pmatrix} 6 & 8 \\ 9 & 0 \end{pmatrix}$, which represents a two-dimensional, real shearing[2].

In Figure 7.3, the (position) vectors $\vec{v_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\vec{v_2} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ are sheared via $A \cdot \vec{v_1}$ and $A \cdot \vec{v_2}$. The purple ellipse illustrates the shearing of the unity disc.

Note that the result of the shearing on the right side of Figure 7.3 can be seen as visualisation of the column vectors of matrix A, $\frac{1}{9}\begin{pmatrix} 6 \\ 9 \end{pmatrix}$ and $\frac{1}{9}\begin{pmatrix} 8 \\ 0 \end{pmatrix}$.

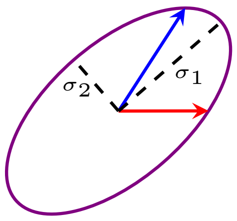


Figure 7.4: Black coloured principal components of matrix A.

The visualisation of matrix A in Figure 7.4 illustrates the principal components as black, dashed lines. The larger of the two principal components has length σ_1, the smaller one has the length σ_2. Intuitively, the dashed lines characterize the main extent of both, the red and the blue vector.

Note that the term-document matrix of Figure 7.2 has nine principal components, one for each document, which is hard to illustrate on a two-dimensional sheet of paper.

Reduction

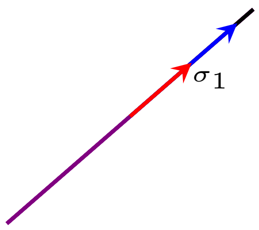


Figure 7.5: Visualization of the matrix $\widehat{A}$ resulting from a reduction of the matrix A from two to one dimension keeping the larger principal component.

In a third step of LSA, the dimensions of the term-document matrix are reduced to a certain extent ranging from 0 to the number of dimensions of the input matrix. A reasonable extent of reduction can be determined experimentally. This is a point of criticism against LSA because an automatic approach for determining a reasonable amount of reduction is unknown (cf. Section 2.2).

Reduction of dimensions in LSA via SVD means that all vectors are “aligned” to the *smaller* principal components. Loosely speaking, the arrowheads are projected to the larger principal components, which looks like squeezing the figure. The smaller principal components represent dimensions with statistical noise. Hence a reduction of these dimensions means reduction of statistical noise.

[2]Matrix A is adapted from a visualization in [Wik13j].

Reducing the principal component annotated with σ_2 in Figure 7.5 yields the matrix $\widehat{A} = \begin{pmatrix} \sim 7.95 & \sim 4.69 \\ \sim 6.67 & \sim 3.94 \end{pmatrix}$, which is visualized using column vectors in Figure 7.5.

Figure 7.6: Figure 7.5 rotated to main axes, which is only the x-axis in this example.

For technical reasons (cf. Section 8.1), the main components are arranged along the main axes (x- and y-axis) via a plane rotation. The result of such a plane rotation of Figure 7.5 is visualized in Figure 7.6.

Semantic Space

As a fourth step in LSA, each document d_i of the repository represented as column vector of the corresponding term-document matrix (cf. Figure 7.2) is transformed via

$$\widehat{d_i} = \Sigma_k^{-1} U_k^{\mathsf{T}} d_i$$

The matrices Σ and U are factors of the singular value decomposition of the term-document matrix (cf. Section 8.1). The scalar k indicates the number of dimensions for the image of the projection. In terms of LSA, the documents are projected into the "semantic space".

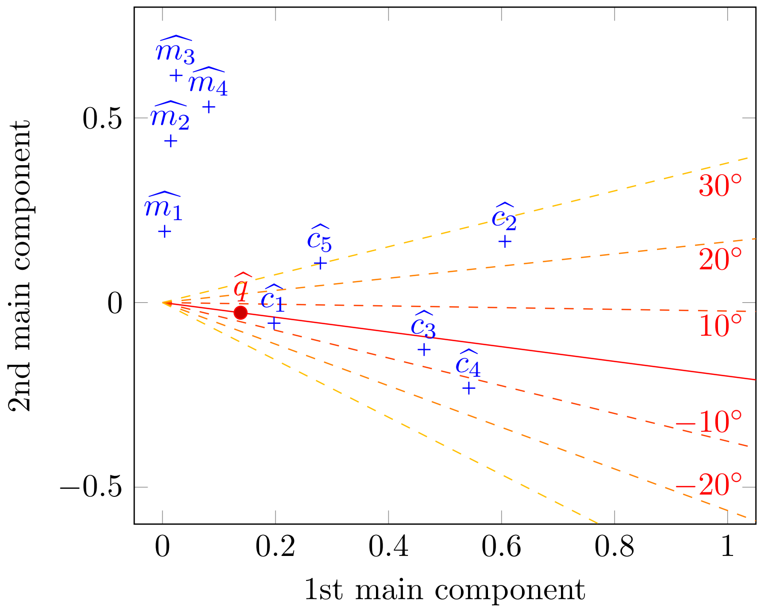


Figure 7.7: Visualization of the documents of Figure 7.2 transformed into the two-dimensional semantic space, adapted from [DDF+90].

Figure 7.7 illustrates the 12-dimensional documents $\widehat{c_1}, \ldots, \widehat{c_5}$ and $\widehat{m_1} \ldots, \widehat{c_4}$ of the technical memo example (cf. Figure 7.1) transformed into the 2-dimensional semantic space[3]. The red point $\widehat{q}$ in the plot visualizes the query "human computer interaction" in the semantic space. A query q is represented like a document as vector in the corresponding term-document matrix: $\vec{q} = (1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)^{\mathsf{T}}$. The first entry 1 stands for "human", the third entry 1 stands for

[3]Note that for verifying the plot in Figure 7.7, the vectors need to be reflected on the y-axis.

computer. The other entries of vector $\vec{q}$ are 0 because the corresponding terms are not contained in the query "human computer interaction". The term "interaction" cannot be represented because the term is not represented in the corresponding term-document matrix (cf. Figure 7.2). This vector $\vec{q}$ is transformed like a document to the semantic space via $\widehat{q} = \Sigma_k^{-1} U_k^\top q_i$.

Ranking of Results Based on Document Similarity

Retrieving documents that are semantically related to the query is based on the similarity of each document vector $\vec{d_i}$ and the query vector $\vec{q}$ transformed into the semantic space, which can be read off the angle θ between the vectors: the narrower the angle, the higher the similarity. This can be calculated via cosine similarity [Wik13a], which ranges from -1 (opposed) over 0 (orthogonal) to 1 (parallel). Hence 1 indicates closely related documents:

$$\cos(\theta) = \frac{\widehat{q} \cdot \widehat{d_i}}{\|\widehat{q}\| \cdot \|\widehat{d_i}\|}$$

The answer to a query q is the list of analyzed documents ordered ascending by the cosine similarity between each document $\widehat{d_i}$ and the query $\widehat{q}$ in the semantic space. The red line in Figure 7.7 illustrates the base line for very similar documents. Sectors highlighted with dashed lines help to classify the documents. The best hits are the documents c_1, c_3, and c_4. The position of document $\widehat{c_5}$ is close to query $\widehat{q}$ in the semantic space but the angle between the vectors $\vec{\widehat{q}}$ and, e.g., $\vec{\widehat{c_2}}$ is smaller than the angle between the vectors $\vec{\widehat{q}}$ and, e.g., $\vec{\widehat{c_5}}$. Hence, document c_2 is a better hit compared to c_5. Different distances between documents in the semantic space and the origin arise from varying term frequencies in the documents.

LSA in a Nutshell

To sum it up, LSA calculates a transformation that allows to compare documents of a repository in a so called semantic space. Input for calculating the transformation is the document repository in the form of a term-document matrix. Comparing two documents means to map each document represented as vector by the transformation into the semantic space, where the angle between the transformed vectors indicates the similarity.

Queries are represented by vectors in the same way as documents. In contrast to documents, queries do not influence the calculation of the transformation to the semantic space. Comparing a query vector with all documents of the repository in the semantic space yields a ranking of the best hits for that query.

Shortcomings of LSA

LSA is well suited for semantic search as already mentioned in Section 2.2. The brief introduction into LSA in this chapter allows to grasp some point of criticism beside overfitting (influenced by parameter k) and optimization of the extent of reduction. Choosing an appropriate parameter k needs some experience and cannot be done automatically.

LSA allows semantic search via a latent model, but the meaning of the main components stays hidden. Deerwester et al. [DDF+90] project the terms of a document repository represented as vectors into the semantic space, which gives some intuition. Terms similar to a main component can be seen as explanations. However an interpretation of a main component based on a linear combination of terms is not easy, in particular in high-dimensional semantic spaces.

Another shortcoming is that the order of terms in a query cannot be represented in a query vector $\vec{q}$. However, the order of terms in queries can influence the semantics of a query. Contemporary Web search engines can cope with varying orders of terms as can be easily verified.

The shortcoming of LSA approached in this thesis is related to the metadata collected by the games of the gaming ecosystem (cf. Chapter 5). In contrast to document repositories targeted by LSA, the creator of the collected metadata in the gaming ecosystem can be distinguished, which could be used for more detailed analyses. Using LSA, however, the term-document matrix has not enough expressivity to represent several creators for one matrix entry, which is a scalar. However, LSA can cope with these metadata, if the creator is not taken into account. This means summing all occurrences of tags for one item yielding one matrix entry.

7.2 Tensors: a Generalization of Matrices

Latent Semantic Indexing [DDF+90] is designed for semantic search based on document collections. However, semantic search based on the metadata of the ARTigo platform poses a new challenge: From the perspective of the data model, the expressivity of a term-document matrix is not enough to model also which *user* assigned a term to a document. For modelling users in addition to terms and documents a new mathematical structure is needed, which is addressed in this section. The consequences for semantic search are discussed in Section 7.3.

In the mathematical structure series of scalar, vector, and matrix, a tensor is the next step. As a matrix can be seen as a rectangular array of numbers, a tensor is a multidimensional array of numbers, in other words a hyper-rectangle containing numbers. Analogously to matrices, elements of a tensor can be addressed using subscripts, e.g., $a_{2,3,5}$ denotes the element in the 2nd row, the 3rd column at depth 5 of Tensor $\mathcal{A}$.

The term *dimensions* of a tensor, like 3 dimensions for a cubic tensor, would clash with the term dimensions in a vector space [SWY75]. In terms of a vector space, the number of dimensions is the number of possible features for an object. Represented as matrix, rows correspond with objects and columns with features or dimensions of a vector space. Recall, that the term-document matrix of Figure 7.2 has nine dimensions (columns). Therefore the term *order* of a tensor is used for declaring its extent, for example, a cubic tensor is a $3rd$-order tensor. Its dimensions are called *modes*.

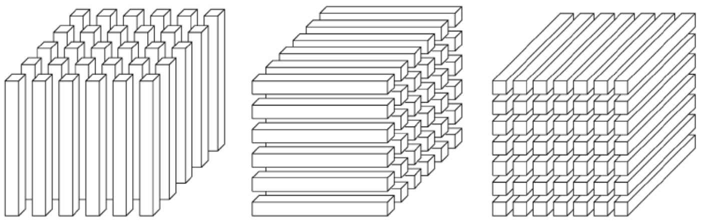


Figure 7.8: nth mode vectors for $n = 1, \ldots, n = 3$ [Kol06].

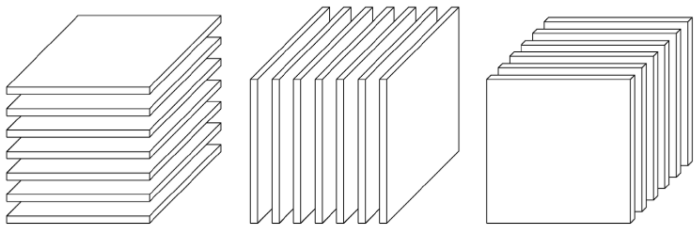


Figure 7.9: nth mode slices for $n = 1, \ldots, n = 3$ [Kol06].

Just like a matrix can be seen as a set of vectors, a tensor can be seen as a set of matrices or vectors. Figure 7.8 visualizes the orientations of matrices and vectors for a 3rd-order tensor. Obviously for a matrix, a 1st-mode vector is a column and a 2nd-mode vector is a row.

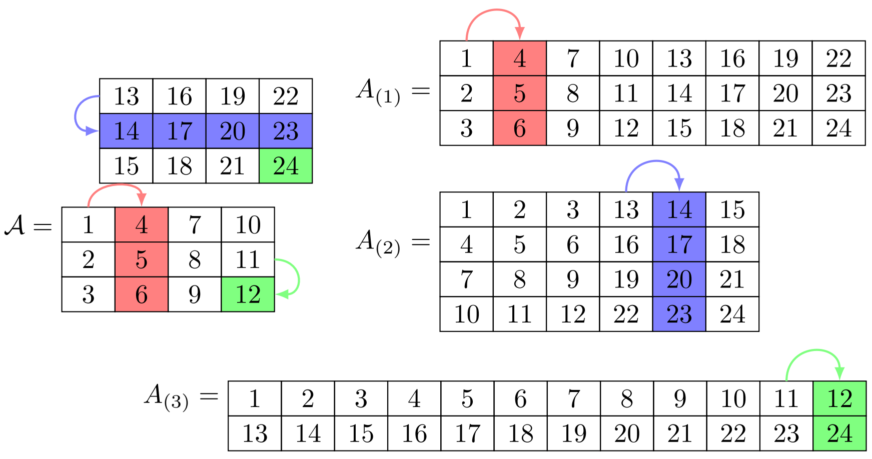
Recall that tensors generalise matrices to more than two modes: A matrix is a 2nd-order tensor. A dth-order tensor $\mathcal{A}$ is an element of $\mathbb{R}^{I_1 \times \ldots \times I_d}$ with tensor elements $a_{i_1 \ldots i_d}$ where the I_i are positive integers. Thus, a dth-order tensor is a multi-dimensional array with d dimensions.

Collecting triples (user, tag, image) yields a 3rd-order tensor. Collecting 4-tuples (user, tag, image, year) yields a 4th-order tensor. The tensors considered here are not the tensors, for example stress tensors, of physics and engineering (called in mathematics “tensor fields”).

Mode-n vectors (or fibers) of a dth-order tensor are Higher-Order analogues of matrix rows and columns. They result from fixing every index of the tensor but one: A matrix column is a mode-1 vector, a matrix row is a mode-2 vector. 3rd-order tensors have column, or mode-1, row or mode-2 and tube or mode-3 vectors. A mode-1 vector of a tensor is a column vector within this tensor, a mode-2 vector a row vector within the tensor, etc. The vectors of a tensor, whatever their modes, are represented as column vectors.

Quite obviously, tensors draw on their own operations like, e.g., the n-mode product, which is built on the matrix product. The following reports on tensor operations that are needed for Higher-Order SVD. Further tensor operations are defined in [Kol06].

Tensor Unfolding



$$\mathcal{A} = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \quad \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}$$

$$A_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix}$$

$$A_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix}$$

$$A_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \end{bmatrix}$$

Figure 7.10: Unfoldings of a sample 3rd-order tensor $\mathcal{A}$

A *tensor unfolding* (or flattening or matricisation) $A_{(n)}$ projects a tensor to a (single) matrix according to parameter n specifying the mode (or orientation) of an unfolding. A cubic tensor, e.g., has three unfoldings. Figure 7.10 lists all three unfoldings $A_{(1)}$, $A_{(2)}$, and $A_{(3)}$ of tensor $\mathcal{A}$. While one mode is fixed, the vectors forming the resulting matrix are collected in ascending order of the number of the mode. Looking at the unfoldings in Figure 7.10, all elements of the tensor are represented in the matrix, the tensor structure, however, is lost. Note, that for matrices $A_{(1)}$ yields the identity, and $A_{(2)}$ is the transpose.

The mode-n unfolding of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots \times I_d}$ is denoted by

$$A_{(n)} \in \mathbb{R}^{I_n \times (I_{n+1} \ldots I_d I_1 \ldots I_{n-1})}$$

Formally, the tensor element $a_{i_1 \ldots i_d}$ yields the matrix element

$$m_{i_n j}, \text{ where } j = 1 + \sum_{k=1; k \neq n}^{d} (i_k - 1) \cdot J_k \text{ with } J_k = \prod_{h=1; h \neq n}^{k-1} I_h$$

.

For example for a 3rd-order tensor $\mathcal{A} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$ the three unfoldings are

- $A_{(1)} \in \mathbb{R}^{J_1 \times J_2 J_3}$,
- $A_{(2)} \in \mathbb{R}^{J_2 \times J_1 J_3}$, and
- $A_{(3)} \in \mathbb{R}^{J_3 \times J_1 J_2}$.

n-mode Product

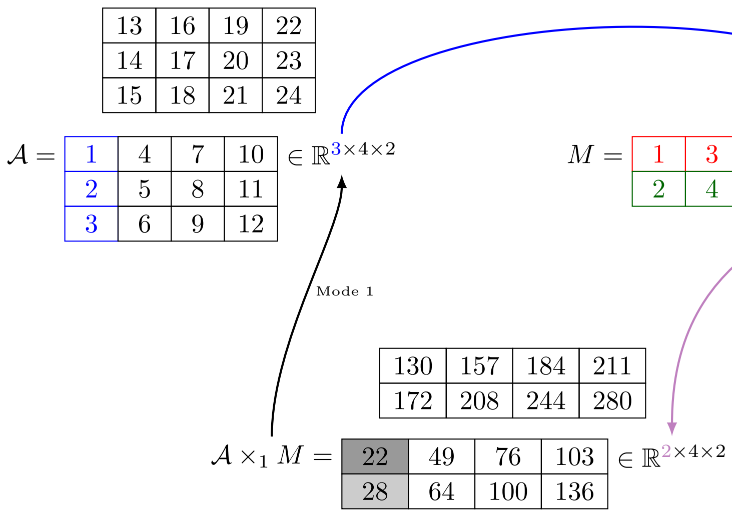


Figure 7.11: n-mode product.

The n-mode product is defined for a tensor as first factor and a matrix as second factor. In contrast to unfoldings, the result is a tensor. Figure 7.11 demonstrates how tensor $\mathcal{A} \in \mathbb{R}^{3\times4\times2}$ is multiplied by matrix $M \in \mathbb{R}^{2\times3}$ in mode 1 (cf. $\times_1$) yielding a product $\in \mathcal{A} \in \mathbb{R}^{2\times4\times2}$. Hence, the multiplication by matrix M projected the number of elements of the 1st-mode vectors from 3 to 2 as indicated by the arrows in Figure 7.11. Note that $\mathcal{A} \times_2 M$ and $\mathcal{A} \times_3 M$ cannot be multiplied because matrix M has 3 columns and fits in with the same extent of 3 in the corresponding mode n of the n-mode product. However, the extent of $\mathcal{A}$ is 4 in mode 2 and 2 in mode 3. Only the extent of $\mathcal{A}$ in mode 1 is 3 and fits in with matrix M.

The highlighted 1st-mode vector $\begin{pmatrix}22\\28\end{pmatrix}$ in Figure 7.11 is the result of the matrix product $M \cdot \begin{pmatrix}1\\2\\3\end{pmatrix}$, which indicates that the n-mode product is based on matrix products. The remaining 1st-mode vectors of $\mathcal{A} \times_1 M$ are products of matrix M and the corresponding 1st-mode vectors in tensor $\mathcal{A}$.

Note that the order of factors is varying. In the n-mode product, matrix M is the second factor. If $\mathcal{A}$ is a 2nd-order tensor, denoted as matrix A, the 1-mode product $\mathcal{A} \times_1 M$ is equivalent to the matrix product $M \cdot A$, and the the 2-mode product $\mathcal{A} \times_2 M$ is equivalent to the matrix product $M \cdot A^\mathsf{T}$.

Formally, the *n-mode product* of a tensor $\mathcal{A} \in \mathbb{R}^{I_1\times\ldots\times I_d}$ with a matrix $M \in \mathbb{R}^{J\times I_n}$ is denoted by

$$\mathcal{A} \times_n M \in \mathbb{R}^{I_1\times\ldots\times I_{n-1}\times J\times I_{n+1}\times\ldots\times I_d}$$

and is defined by

$$(\mathcal{A} \times_n M)_{i_1\ldots i_{n-1} j i_{n+1}\ldots i_d} = \sum_{i_n=1}^{I_n} a_{i_1\ldots i_d} \cdot m_{j i_n}$$

Sub-tensor

The *sub-tensors* of a tensor $\mathcal{A}$ are denoted by $\mathcal{A}_{i_n=a}$ for $n = 1, \ldots, d$ where the nth index of $\mathcal{A}$ is fixed to a.

7.3 Higher-Order Latent Semantic Analysis (HO-LSA)

Obviously, standard LSA (cf. Section 7.1) and tensors (cf. Section 7.2) are not compatible (except for 2nd-order tensors, which are matrices) because standard LSA draws on a term-document as input and not on tensors of arbitrary order. Realizing semantic search based on the metadata collected with the ARTigo platform in Part II requires changes either of the input of standard LSA or of LSA itself. More concretely, either the input tensor must be transformed to a suitable input *matrix* for allowing standard LSA, or standard LSA needs to be extended so as to accept *tensors* as input. This section pursues the latter approach and directs stepwise from standard LSA to Higher-Order LSA.

	Aggregation (Section 7.3.1)	Noise-reduced Aggregation (Section 7.3.2)	HO-LSA (Section 7.3.2)
Input	tensor	tensor	tensor
Output	matrix	general matrix	personal matrices
Approximation	terms	terms, documents, users	terms, documents, users
Approximation Parameters	k	m_1, m_2, m_3	m_1, m_2, m_3
Reduction	SVD	HO-SVD	HO-SVD
Ranking	cosine-similarity	cosine-similarity	cosine-similarity
Personalized Search			✓

Figure 7.12: Approaches for handling tensors as input for LSA.

This section proposes three approaches (cf. Figure 7.12) realizing semantic search based on tensor data as input. The main goal of these approaches is calculating a matrix that aims at transforming documents/items and queries (represented as vectors) into a semantic space (characterized by a matrix). Hence, all approaches allow document/item retrieval based on cosine similarity as in LSA (cf. Section 7.1). Retrieval in a multi-linear (tensor) semantic space was not in the scope of this thesis but obviously ought to be done as part of future work. Cosine similarity might be a candidate for integration by analogy of SVD and HO-SVD.

Note that the approach proposed in [SNPM06] combines matrix SVDs and is not based on Higher-Order LSA.

7.3.1 Tensor Aggregation for Standard LSA

A rather simple approach to cope with tensors in semantic search via standard LSA is aggregating the users held in a term-document-user tensor as input into a general term-document matrix. Hence, an entry of such a matrix indicates how often a document (column) was tagged with a certain term (row). This term-document matrix can be used for standard LSA as reported about in Section 7.1. Naturally, queries need to be represented as vectors.

Note that this approach differs from standard LSA where an entry in the term-document matrix represents the number of occurrences of a term in a document.

Aggregation is computationally cheap, which is an advantage of this approach but – quite obviously – information gets lost that might be useful for calculating a more precise transformation into a semantic space for semantic search. In particular, the entry of a term-document matrix is anonymous and does not reveal its sources.

7.3.2 Noise-reduced Tensor Aggregation for Standard LSA

A shortcoming of the tensor aggregation approach (cf. Section 7.3.1) is –as already mentioned– the loss of (potentially) useful information by aggregating the tensor to a matrix. This loss of information can be decreased using Higher-Order SVD for noise reduction.

Recall that noise reduction via SVD is the basic principle of LSA (cf. Section 7.1). In the visualization of technical memos (documents) in the semantic space (cf. Figure 7.7) the formerly twelve terms in the term-document matrix are reduced to two main components in the semantic space. SVD applied on a document-term matrix can be used to reduce the nose of the documents instead. Reducing both, terms and documents cannot be achieved with matrix SVD.

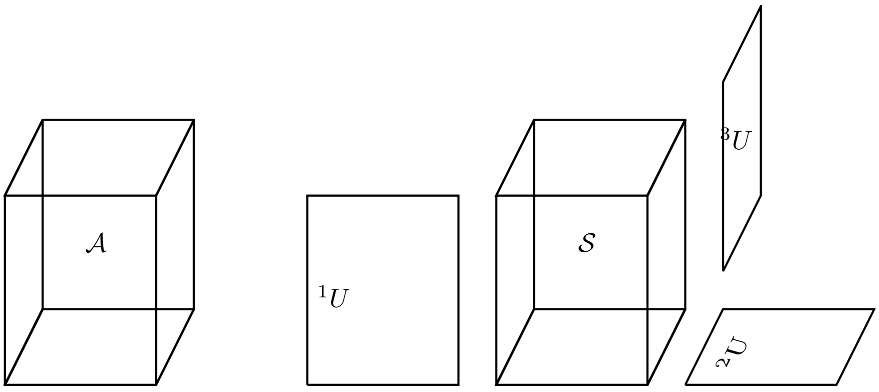


Figure 7.13: Sketch of the Higher-Order SVD of tensor $\mathcal{A}$.

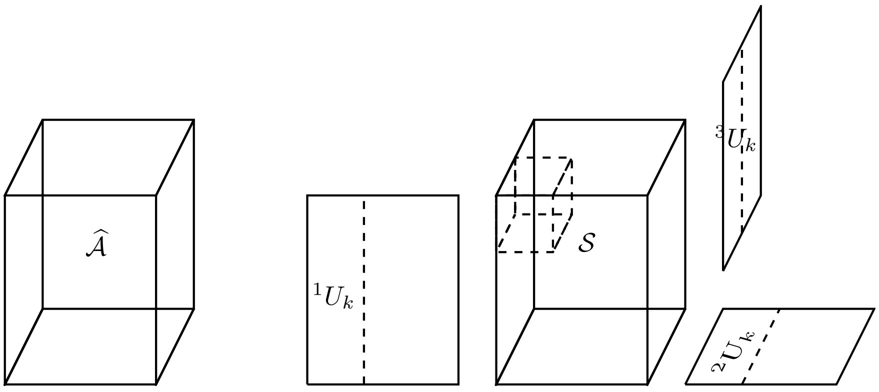


Figure 7.14: Sketch of a reduction resulting in an approximated tensor $\widehat{\mathcal{A}}$.

Higher-Order SVD (cf. Chapter 8) allows noise-reduction in all modes of a tensor. Figure 7.13 sketches a decomposition of a 3rd-order tensor $\mathcal{A}$ into its factors tensor $\mathcal{S}$ and matrices ^1U, ^2U, and ^3U. Figure 7.14 sketches a noise-reduced approximation of tensor $\mathcal{A}$ denoted $\widehat{\mathcal{A}}$. This approximation results from a product of pruned factors on the right-hand side of Figure 7.14. The remaining factors are tensor $\mathcal{S}$ and matrices ^1U_k, ^2U_k, and ^3U_k.

The dashed lines illustrate the crop mark. The position of the crop mark or in other words the extent of reduction can be controlled with parameters m_i, where i indicates the mode. For a 3rd-order term-document-user tensor, the parameters m_1 (terms), m_2 (documents), and m_3 (users) determine the extent of reduction in the range of 0 to the number of entries in that mode. For instance, if such a tensor represents 12 terms, m_1 ranges from 0 to 12, where 12 means no reduction. The parameter name m_i is most likely inspired by the "mode". The letter k is the analogous parameter of matrix SVD.

Note that the tensors $\mathcal{A}$ and $\widehat{\mathcal{A}}$ have the *same shape* although tensor $\widehat{\mathcal{A}}$ is the product of pruned factors, which obviously have different shapes. This is a common misunderstanding. Chapter 9 uses image compression as application of Higher-Order SVD, which is supposed to provide an intuitive understanding of tensor reduction.

Reducing the noise of tensor $\mathcal{A}$ via Higher-Order SVD is supposed to be a promising preprocessing step for the tensor aggregation approach of Section 7.3.1. A direct validation of this approach of noise-reduced tensor aggregation is not in the scope of this thesis. Indirectly Chapter 9 can be seen as validation of the approach, because it is shown that Higher-Order SVD is a valid integration of SVD using image compression as application.

based on Higher-Order SVD indicates that the approach is promising, because the main principle of standard LSA is noise-reduction for gaining a mapping of syntax to semantic.

7.3.3 Personalized Higher-Order LSA

In contrast to noise-reduced tensor aggregation (cf. Section 7.3.2), Higher-Order SVD can be used for *personalized* semantic search. A precondition of personalized search is specifying the context of the searcher. In the ARTigo platform this context is implicitly specified by the contribution of a registered user. Logged in into the ARTigo platform, the context of this user can be assigned easily.

As in the approaches before, the goal of Higher-Order LSA is to provide a transformation characterized by a matrix that allows to map documents and queries into a semantic space. In the case of Higher-Order LSA, each user has an own transformation matrix according to the corresponding context. Reasons for differences of matrices may be various tagging behaviour or different languages for tagging.

Recall that documents and queries need to be transformed into the semantic space for answering queries. This transformation is expressed by the following formula (cf. Section 7.1).

$$\widehat{d_i} = \Sigma_k^{-1} \ \cdot \ U_k^{\top} \ \cdot \ d_i$$

This transformation can be extended for a personalized semantic search using a personalized matrix Σ, denoted as $\acute{\Sigma}$. The calligraphic notation indicates that matrix $\acute{\Sigma}$ is a personalized Σ originated from a tensor. Furthermore, matrix U_k can be identified with matrix ${}^1U_k$ of the Higher-Order SVD of the input tensor $\mathcal{A}$ [SWB12], yielding:

$$\widehat{d_i} = \acute{\Sigma}^{-1} \ \cdot \ {}^1U_k^{\top} \ \cdot \ d_i$$

Matrix $\acute{\Sigma}$ is the core tensor $\mathcal{S}$ transformed to a 2nd-order tensor, which can be interpreted as matrix. The following equation yields matrix $\acute{\Sigma}$ for 3rd-order tensors (modelling, e.g., users, tags, and resources). Index l determines the reduction in terms of user data. Vector u selects the data of a certain user identified by the corresponding slice of input tensor $\mathcal{A}$. The entry of vector u with the same index as the user's slice in the input tensor $\mathcal{A}$ is 1, the other entries are zero.

$$\acute{\Sigma} = \mathcal{S} \ \times_3 \ ({}^3U_l^{\top} \cdot \underbrace{u}_{\begin{pmatrix}0\\ \vdots\\ 0\\ 1\\ 0\\ \vdots\\ 0\end{pmatrix}})$$

This approach can be generalized for tensors of higher order than 3, e.g., modelling changes over time in the same manner. Scalar t in this example stands for time and scalar m for its extent of reduction.

$$\acute{\Sigma} = \mathcal{S} \times_3 ({}^3U_l^{\mathsf{T}} \cdot \underbrace{u}_{\begin{pmatrix}0\\ \vdots \\ 0\\ 1\\ 0\\ \vdots \\ 0\end{pmatrix}}) \times_4 ({}^4U_m^{\mathsf{T}} \cdot \underbrace{t}_{\begin{pmatrix}0\\ \vdots \\ 0\\ 1\\ 0\\ \vdots \\ 0\end{pmatrix}})$$

Note that this approach has not been validated yet. A precursor of a validation based on ARTigo data was conducted by Schnuck [Sch10] for the sake of tag recommendation reproducing the results of Symeonides et al. [SNM08].

Chapter 8

Higher-Order SVD (HO-SVD)

Latent Semantic Analysis in its standard or higher-order form is based on Singular Value Decomposition (SVD). Higher-Order LSA draws on an integrated SVD, called Higher-Order SVD (HO-SVD) that is itself based on SVD. This chapter reports on SVD and its integration into HO-SVD. While the factors resulting from decompositions of matrices by SVD and decompositions of tensors by HO-SVD are taken as granted in the previous Chapter 7, this chapter focuses on the mathematical foundations of SVD and HO-SVD as well as on algorithms for calculating the needed factors of SVD or HO-SVD. The chapter is organized as follows. Section 8.1 is an informal intuitive introduction to SVD. Section 8.2 encapsulates concisely the needed mathematical definitions regarding SVD and Higher-Order SVD. Finally, Section 8.3 reports on related work on SVD as needed for a later parallelization of Higher-Order SVD (cf. Chapter 10).

8.1 Singular Value Decomposition (SVD) and Reduction

A singular value decomposition (SVD) is a factorization of a matrix $A \in \mathbb{R}^{m \times n}$ into the matrices U, Σ, and V, where $A = U\Sigma V^{\mathsf{T}}$. Every real matrix has at least one singular value decomposition. Figure 8.1 shows an introductory example of such a factorization.

For the first time, the term "valeurs singulières" was used around the year 1910 by Emile Picard, a French mathematician [Ste93]. Interestingly enough, "valeurs *singulières*" denote "*curious* values". This initial naming did not hint at a connection to singular matrices. At that time, the computation of an SVD was quite difficult without the support of computers. Even with computers, an efficient algorithm to compute an SVD was still missing. In 1965, the first efficient SVD algorithm was published by Golub [GK65].

$$\overset{A}{\begin{pmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \end{pmatrix}} \overset{}{=} \overset{U}{\begin{pmatrix} \mathbf{0} & 0 & 1 & 0 \\ \mathbf{0} & 1 & 0 & 0 \\ \mathbf{0} & 0 & 0 & -1 \\ \mathbf{1} & 0 & 0 & 0 \end{pmatrix}} \overset{\Sigma}{\begin{pmatrix} \mathbf{4} & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}} \overset{V^{\mathsf{T}}}{\begin{pmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 1 & 0 & 0 \\ \sqrt{0.2} & 0 & 0 & 0 & \sqrt{0.8} \\ 0 & 0 & 0 & 1 & 0 \\ -\sqrt{0.8} & 0 & 0 & 0 & \sqrt{0.2} \end{pmatrix}}$$

Figure 8.1: Example Singular Value Decomposition.

In an SVD of a matrix A, matrix $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix. The main diagonal holds the singular values σ_i (4, 3, $\sqrt{5}$, and 0 in Figure 8.1 or refer to Figure 7.4 for a geometric interpretation in an ellipse). The entries of the main diagonal are sorted in descending order. $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are square matrices. Both, matrix U and matrix V are unitary, hence, the columns of each matrix are orthonormal and $U \cdot U^{\mathsf{T}} = V \cdot V^{\mathsf{T}} = I$. The column vectors of U (V) are called left (right) singular vectors of matrix A. Left and right singular vectors correspond to singular values as highlighted in Figure 8.1: The first left singular vector in matrix U corresponds with the first singular value in Σ and the first (transposed) right singular vector in matrix V.

Instead of computing a full singular value decomposition of matrix A as in Figure 8.1, only k singular values with corresponding singular vectors can be computed as shown in Figure 8.2. This decomposition is called truncated SVD.

$$\begin{array}{ccccc} A' & = & U' & \Sigma' & V'^{\mathsf{T}} \\ \begin{pmatrix} \mathbf{0} & 0 & 0 & 0 & \mathbf{0} \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \end{pmatrix} & = & \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} & \begin{pmatrix} \mathbf{4} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \\ A'' & = & U'' & \Sigma'' & V''^{\mathsf{T}} \\ \begin{pmatrix} \mathbf{1} & 0 & 0 & 0 & \mathbf{0} \\ 0 & 0 & \mathbf{0} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \end{pmatrix} & = & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} & \begin{pmatrix} \mathbf{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 8.2: 25% – 75% of the singular values are set to 0.

The product of a truncated SVD is denoted as reduction of the originally decomposed matrix A as illustrated in Figure 8.2. The resulting matrix A' results from keeping $k = 2$ singular values in matrix Σ and setting the remaining ones to 0. Matrix A' differs only in two values compared to the original matrix A (cf. highlighted entries on the upper left side of Figure 8.2). Keeping only $k = 1$ singular values (cf. bottom of Figure 8.2) causes three differences comparing matrix A and matrix A''. This behaviour can be interpreted as approximation of matrix A, denoted as $\widehat{A}$. The more and more singular values are set to 0, the approximation of matrix A is getting worse.

A common misunderstanding is, that a reduction based on SVD of (an exemplary) matrix A would lower the number of columns of that matrix. In fact, the number of columns of matrices U, Σ, and V is "reduced". And indeed, the dimensionality of the semantic space (cf. Chapter 7) is reduced. However, the product of the reduced factors has the same number of columns and rows as the original matrix A (cf. Figure 8.2).

8.2 SVD and Higher-Order SVD

This section focuses on the mathematical foundation of SVD and Higher-Order SVD.

Singular Value Decomposition (SVD)

A matrix $A \in \mathbb{R}^{I_1 \times I_2}$ can be decomposed as $A = U \Sigma V^{\mathsf{T}}$ where:

- Σ is a (pseudo) diagonal matrix. In a pseudo diagonal matrix, nonzero elements can only occur in the diagonal of a (left) upper sub-matrix.
  $diag(\sigma_1, \ldots, \sigma_z) \in \mathbb{R}^{I_1 \times I_2}$, with $z = min\{I_1, I_2\}$ and ordered $(\sigma_1 \geq \ldots \geq \sigma_z >= 0)$q
- $U \in \mathbb{R}^{I_1 \times I_1}$ and $V \in \mathbb{R}^{I_2 \times I_2}$ are orthogonal matrices.
- The σ_i are the *singular values* of A, the columns u_i of $U = [u_1, \ldots, u_{I_1}]$ (v_j of $V = [v_1, \ldots, v_{I_2}]$, resp.) are the *left* (*right*, resp.) *singular vectors* of A.

Higher-Order SVD (HO-SVD)

A tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots \times I_d}$ can be decomposed [DLDMV00] as $\mathcal{A} = \mathcal{S} \times_1 U_1 \times_2 U_2 \ldots \times_d U_d$ where:

- The *core tensor* $\mathcal{S} \in \mathbb{R}^{I_1 \times \ldots \times I_d}$ is ordered and all-orthogonal.
- Every matrix $U_n = [u_{n,1}, \ldots, u_{n,I_n}] \in \mathbb{R}^{I_n \times I_n}$ (with $u_{n,i} \in \mathbb{R}^{I_n \times 1}$, $1 \leq n \leq d$) is orthogonal.
- $\sigma_{n,i} = \|\mathcal{S}_{i_n=i}\|$ are the *n-mode singular values* (cf. sketch in Figure 7.14); the vectors $u_{n,i} \in \mathbb{R}^{I_n \times 1}$ are *n-mode singular vectors* of $\mathcal{A}$.

The *n-mode singular vectors* of a tensor generalise the left and right singular vectors of a matrix, the value of the Frobenius-norm $\|.\|_F = \sqrt{\sum_{i,j} |a_{ij}|^2}$ of the $(d-1)$ th-order sub-tensors of the core tensor $\mathcal{S}$ correspond to the singular values of a matrix.

The *n-rank* of $\mathcal{A}$, $\text{rank}_n(\mathcal{A})$, is the dimension of the vector space spanned by the n-mode vectors.

Every tensor has a Higher-Order Singular Value Decomposition and its n-mode singular values are uniquely defined [DLDMV00].

The following theorem [DLDMV00] relates (matrix) SVD and (tensor) Higher-Order SVD: If $\mathcal{A} = \mathcal{S} \times_1 U_1 \ldots \times_d U_d$ is a Higher-Order SVD, then the SVD of the matrix unfolding $A_{(n)}$ is $U_n \Sigma_n V_n^{\mathsf{T}}$ where $\Sigma_n = diag(\sigma_{n1}, \ldots, \sigma_{nI_n}) \in \mathbb{R}^{I_n \times I_n}$.

It follows from this theorem that a Higher-Order SVD of a dth-order tensor can be computed from the SVD of its mode-n unfoldings $A_{(n)} = U_n \Sigma_n V_n^{\mathsf{T}} = \Sigma_n \times_1 U_n \times_2 V_n$ for $n = 1, \ldots, d$ as follows: $\mathcal{A} = \mathcal{S} \times_1 U_1 \times_2 \ldots \times_d U_d$ where U_n contains the nth left singular vectors of unfolding $A_{(n)}$. These equations result in the following algorithm that generalises the algorithm given in [SZL+05] for 3rd-order tensors to tensors of any order:

```
Input           dth-order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots \times I_d}$,
                pruning tuple $(m_1, \ldots, m_d) \in [1, I_1] \times \ldots \times [1, I_d]$
Output          dth-order tensor $\widehat{\mathcal{A}} \in \mathbb{R}^{I_1 \times \ldots \times I_d}$
                as the best rank-$(m_1, ..., m_d)$ approximation to $\mathcal{A}$

Unfolding       for $i = 1, \ldots, d$
                  Compute the unfolding $A_{(i)}$ of $\mathcal{A}$
                end
Matrix SVD      for $i = 1, \ldots, d$
                  Compute the SVD $A_{(i)} = U_i \Sigma_i V_i^{\mathsf{T}}$
                end
Pruning         for $i = 1, \ldots, d$
                  $W_i := [u_{i,1}, \ldots, u_{i,m_i}]$ where $u_{i,j}$ column vectors of $U_i$
                end
Core Tensor     Compute $\mathcal{S} := \mathcal{A} \times_1 W_1^{\mathsf{T}} \times_2 W_2^{\mathsf{T}} \ldots \times_d W_d^{\mathsf{T}}$
Approximation   Compute $\widehat{\mathcal{A}} := \mathcal{S} \times_1 W_1 \times_2 W_2 \ldots \times_d W_d$
```

Algorithm 8.1: Higher-Order SVD

Note that, in principle, any SVD algorithm can be used at line 5. The parameters m_i aim at a reduction of the dimensionality of U_i $(i = 1, \ldots, d)$ as follows: At line 8 the length of the column vectors of u_i is pruned according to m_i, that is, keeping the first $m_i \leq I_i$ rows of u_i for W_i, only. W_i (instead of full U_i) is used for the computation of $\widehat{\mathcal{A}}$ in lines 10 and 11.

8.3 Related Work on SVD

Higher-order SVD has been considered in few publications. Lathauwer et al. [DLDMV00] has been one of the first systematic studies of Higher-Order SVD called there multilinear SVD. The article investigates what properties (such as uniqueness and first-order perturbation effects) and how symmetries in a tensor can be exploited for simplifying its SVD. Like all other work on Higher-Order SVD, this thesis builds upon the observation by Lathauwer et al. that SVD can be generalised to tensors. This present article extends [DLDMV00] with a tractable parallelisation of Higher-Order SVD.

Symeonidis et al. [SNM08] present an algorithm for the SVD of 3rd-order tensors, called (HO)SVD, based on the *CubeSVD* algorithm of Sun et al. [SZL$^+$05] (correcting a little mistake in that article). Symeonidis et al. [SNM08] also explain how LSA [DDF$^+$90] can be extended to 3rd-order tensors and, interestingly, for tag recommendation. The paper reports on significantly better tag recommendation with Higher-Order SVD than with standard SVD. In contrast to the results of this thesis, Symeonidis et al. [SNM08] is limited to 3rd-order tensors and does not address parallelising their 3rd-order tensor SVD. Vasilescu et al. [VT02] explain how a Higher-Order SVD, called there N-mode SVD, can be used for the analysis of facial images with multifactor structure such as scene geometry, view point and illumination. Parallelisation is not addressed and the method's complexity is not discussed. Drineas et al. [DDH01, DM05] introduce randomised algorithms for SVD and Higher-Order SVD with aiming at a better runtime than non-randomised methods. Some time complexity bounds given in this paper seem to be too optimistic.

Brent et al. [BL85] state and explain a parallel algorithm for the SVD of matrices based on *Hestenes' Method* [Hes58] making use of a variable number of processors. Chuang [CC89] extends Brent's approach to a cube connected SIMD for a fixed number of processors (that can be lower than the number of columns). The runtime analysis of [BL85] seems to be too optimistic. Instead, the analysis of [CC89] is used in this thesis. Berry et al. [BMPS06] report on the state of the art of parallel (matrix) SVD computation (relying on [BL85, CC89]). In particular the algorithm for parallel basis orthogonalisation ROCVEC using the algorithm ROCDEC corresponds to ORTH (algorithm 10.3) using Parallel Matrix SVD (algorithm 10.2) in this thesis. The Hestenes' Method is described as an "efficient way to compute the decomposition".

None of the above mentioned articles report on parallelising Higher-Order SVD and achieving an efficiency comparable to that reported about in Chapter 10. With the exception of Sun et al. [SZL$^+$05] and Symeonidis et al. [SNM08] only considering 3rd-order tensors, there are no former reports on experimental evaluations pointing to the superiority of Higher-Order LSA for social tagging-based similarity search.

Recently, RESCAL[1] was published reaching extraordinary time-complexity for calculating Higher-Order SVD. RESCAL is restricted to third-order tensors and reduction of two modes.

[1]https://github.com/nzhiltsov/Ext-RESCAL

Chapter 9

An Experimental Validation of Higher-Order SVD Using Image Compression

This chapter is devoted to validate, and to make better understandable, Higher-Order SVD using an practical application. The experimental validation given in this chapter is intended to support the rather dry material of Chapter 8 by some human intuition.

Image compression has been chosen as an application for an experimental evaluation for the following reasons. First, the representation of images as 3rd-order tensors after the RGB (Red Green Blue) model is closely related to the perception of images by human eyes. Second, the human brain interprets such tensor-like representations of images thus making it possible to evaluate at a glance the quality of an image compression by Higher-Order SVD. Image compression is in particular well-suited for validation of SVD algorithms because, based on computation that is hard-wired in the visual centres of the human brain, faults result in messy or unnatural pixels that can be easily observed having a look at the whole image.

The objective of this chapter is not improving image compression even though the results presented in this chapter might result in such a contribution. A comparison of image compression based on both, matrix SVD and the popular JPEG[1] approach is given by Arnold [Arn00] using three matrices, one for each color channel red, green, and blue (cf. Figure 9.7). Liu et al. [LZH09] evaluate Higher-Order SVD yielding better performance results compared to JPEG.

In the following a first section addresses human image perception. Second, the principle of human image perception is directly transferred to a memory model for representing images. Third, the memory model is interpreted as a tensor that can be decomposed by Higher-Order SVD. Fourth, an image is represented as tensor and decomposed to factors. Fifth, the factors are used to approximate the image while losing quality but saving memory. Sixth, the relevance of such an image compression for Higher-Order SVD is discussed.

9.1 Human Image Perception

Human image perception begins at the human eye. Waves of light pass the pupil and reach the (hemispherical) retina like in a pinhole camera. Figure 9.1 [KFN95] shows a sketch of an eye-ball and an enlarged part of the retina on the right-hand side. All waves of light of various wave lengths reaching the retina define the field of view and form an image that is perceived by two classes of photoreceptors called cones and rods. Humans are using trichromatic color vision[2] based on red, green, and blue color (RGB) [Row02]. Waves of light are perceived by the cone photoreceptors,

[1] `http://www.jpeg.org/`

[2] Interestingly enough, only primates (including humans) are using trichromatic color vision.

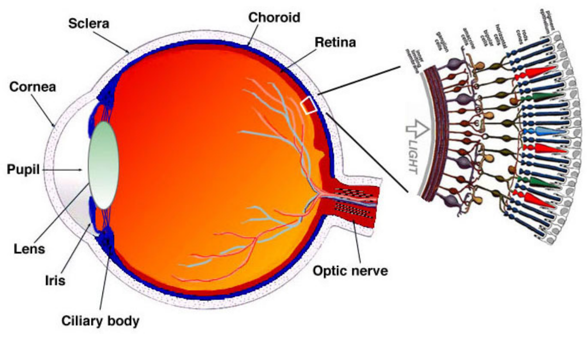


Figure 9.1: Human eye-ball with a schematic enlargement of the retina [KFN95].

that are subdivided into three distinct types, each type being capable of percepting the red, green, or blue light waves. The other class of photoreceptors called rod cells is specialized on the vision of less intense light, e.g., monochrome night vision.

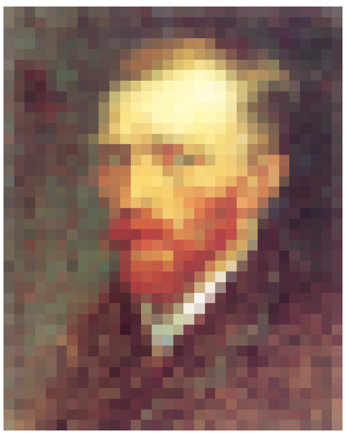


Figure 9.2: Van Gogh, self portrait, 1887, The Art Institute of Chicago.

Figure 9.2 visualizes the structure of a colored image (a celebrated self-portrait van Gogh painted in 1887 –now at Art Institute of Chicago– in low resolution. The low resolution reveals small (square) unicoloured parts of the image called pixels, which would be considerably smaller for images of a higher resolution. Usually the size of pixels ranges below the human perception threshold. For the perception of a single pixel all three types of rod cells are involved yielding red, green, and blue brightness information. A red, green, or blue color component of a pixel is denoted as subpixel.

The brightness values of these subpixels are perceived by about 6 million rod cells per eye. Figure 9.3 visualizes the color channels of the self portrait (Figure 9.2) containing the subpixels of all types. Finally the human brain uses the red, green, and blue brightness values to generate the colored perception of an image.

9.2 Tensor as Memory Model for Images

For representing images a data structure is needed. Thinking of the principles of human image perception as briefly recalled above, representing pixels and their location is sufficient. The number of different brightness values that can be distinguished by the human eye seems to be sufficiently close to the number 256 to make $2^8 = 256$ brightness values an appropriate postulate for binary data representation.

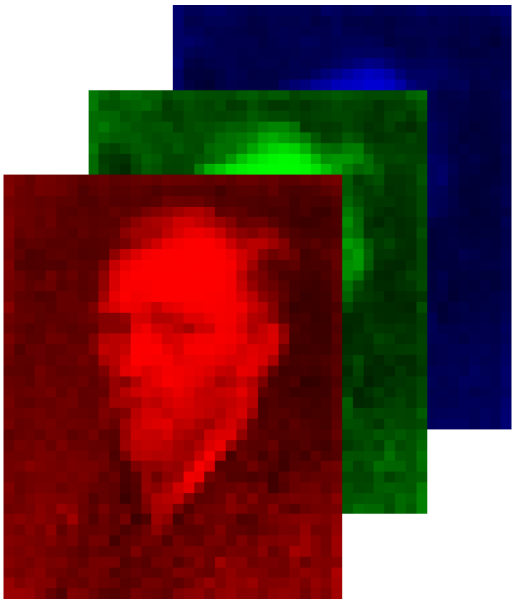


Figure 9.3: Tensor representation of Figure 9.2.

This range fits well into the needs of binary data representation as used by memory chips. A natural way to represent the location of pixels in an image is to use two-dimensional arrays, where the column (rows) index represents the width (height) axis of an image, and array cells hold brightness values. One pixel contains three brightness values, hence, a reasonable approach is to use three arrays, one for each color channel as Figure 9.3 illustrates. Mathematically a two-dimensional array of brightness values is a matrix. An image thus corresponds to three matrices, each matrix representing one colour channel.

$$\begin{pmatrix} \mathbf{124} & 105 & 98 & 123 & \cdots \\ 120 & 100 & 85 & 100 & \cdots \\ 112 & 115 & 109 & 96 & \cdots \\ 113 & 118 & 117 & 102 & \cdots \\ 125 & 110 & 106 & 107 & \cdots \\ 127 & 100 & 91 & 100 & \cdots \\ 122 & 89 & 74 & 82 & \cdots \\ 116 & 92 & 78 & 72 & \cdots \\ 114 & 86 & 66 & 78 & \cdots \\ 118 & 99 & 83 & 83 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad \begin{pmatrix} \mathbf{126} & 104 & 89 & 112 & \cdots \\ 119 & 97 & 76 & 86 & \cdots \\ 107 & 108 & 96 & 83 & \cdots \\ 104 & 106 & 102 & 87 & \cdots \\ 109 & 93 & 87 & 88 & \cdots \\ 109 & 80 & 70 & 79 & \cdots \\ 102 & 69 & 50 & 58 & \cdots \\ 95 & 71 & 52 & 46 & \cdots \\ 95 & 65 & 40 & 49 & \cdots \\ 100 & 79 & 57 & 54 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad \begin{pmatrix} \mathbf{104} & 86 & 82 & 110 & \cdots \\ 98 & 80 & 69 & 83 & \cdots \\ 88 & 92 & 88 & 77 & \cdots \\ 87 & 92 & 95 & 80 & \cdots \\ 96 & 83 & 80 & 82 & \cdots \\ 99 & 71 & 65 & 74 & \cdots \\ 95 & 62 & 48 & 56 & \cdots \\ 90 & 66 & 53 & 47 & \cdots \\ 91 & 62 & 43 & 53 & \cdots \\ 96 & 78 & 60 & 58 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Figure 9.4: Tensor visualization of Figure 9.3 using matrices for the color channels.

Stacking the matrices of the 3 color channels one above the other (see Figure 9.4) ties the corresponding subpixels together. This is a multidimensional matrix or, in other words, a tensor. The entries **124** (red), **126** (green), and **104** (blue) in the upper left of the matrices that are highlighted in Figure 9.4 correspond to the pixel in the upper left corner of the self portrait of Figure 9.2 and stand for a greenish shade of color.

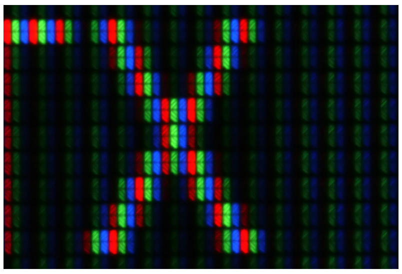


Figure 9.5: Pixels on a screen consisting of red, green, and blue sub-pixels.

TV or computer displays can make use of subpixels directly instead of displaying the true color of each pixel. Figure 9.5 shows a close-up view onto such a display. Obviously, the subpixels become visible and form the letter “X” in red, green, and blue color. However, narrowing the own eyes to slits or looking at the letter from a distance makes the subpixels fuse to pixels and the letter “X” appears in white color.

9.3 Decomposition of an Image Tensor

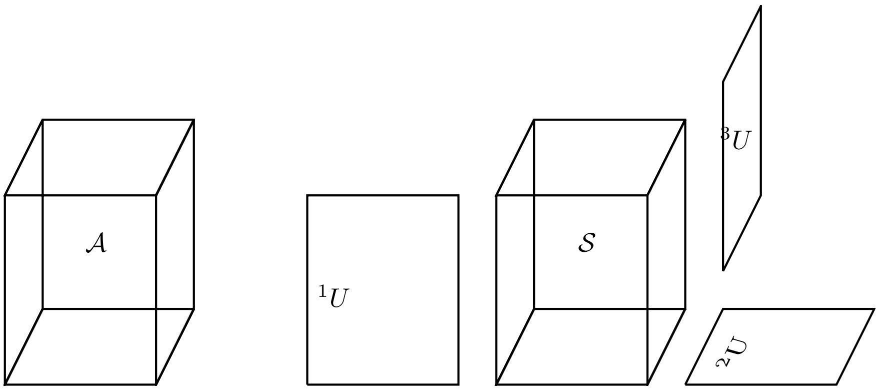


Figure 9.6: Visualization of Higher-Order SVD.

Tensors can be decomposed into factors using Higher-Order SVD (cf. Chapter 8). This section illustrates the decomposition of a tensor $\mathcal{A}$ that represents an image using the RGB color model as described above. Tensor $\mathcal{A}$ on the left side of Figure 9.6 sketches the tensor representation of the self portrait as shown in Figure 9.4. The right side of Figure 9.6 sketches the factors of the Higher-Order SVD of tensor $\mathcal{A}$. The product $\mathcal{S} \times_1 {}^1U \times_2 {}^2U \times_3 {}^3U$ yields Tensor $\mathcal{A}$ or in other words the image. The matrices ${}^1U$, ${}^2U$, and ${}^3U$ are the left singular vectors of the unfoldings of tensor $\mathcal{A}$.

The core tensor $\mathcal{S}$ corresponds to the matrix Σ in the case of matrix SVD. The matrices U and V correspond to the first and second unfolding of a 2nd-order tensor $\mathcal{A}$, which in fact is a matrix. Generally spoken, the core tensor of a Higher-Order SVD is of the same order as the decomposed tensor and one unfolding is needed for each mode.

9.3.1 Calculation of Left Singular Vectors ${}^1U$, ${}^2U$, and ${}^3U$

The following illustrates how matrices ${}^1U$, ${}^2U$, and ${}^3U$ can be calculated. First, the unfoldings of tensor $\mathcal{A}$ are visualized using subpixels. These unfoldings are the input for calculating the matrices ${}^1U$, ${}^2U$, and ${}^3U$ by matrix SVDs.

The unfoldings of tensor $\mathcal{A}$ are visualized in Figures 9.7, 9.8, and 9.9. One should keep in mind, that the red, green, and blue subpixels in fact represent matrix entries. An unfolding can be seen as matrix representation of a tensor regarding one mode. The unfolding in mode 3 has three rows, one for each color channel. Starting a new (indented) line after the width of the original image (see Figure 9.10) yields the visualization of the unfolding (cf. Figure 9.9), which gives a better perception revealing the image again.

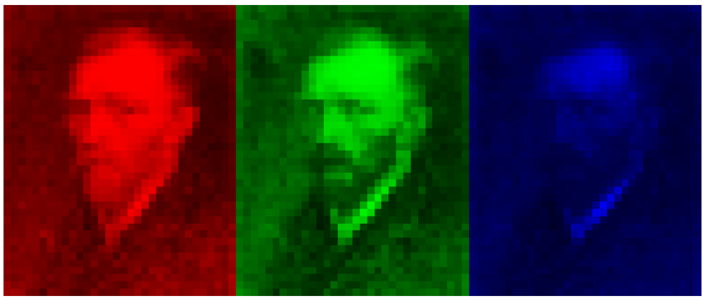


Figure 9.7: Visualization of unfolding $A_{(1)}$

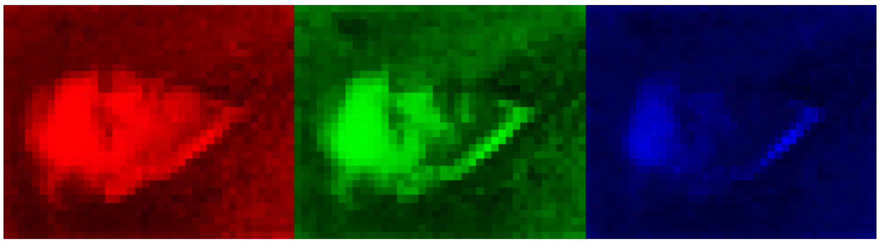


Figure 9.8: Visualization of unfolding $A_{(2)}$

Figure 9.9: Visualization of the first (about 10%) rows of unfolding $A_{(3)}$

The three unfoldings visualized in Figures 9.7, 9.8, and 9.9 are decomposed using matrix SVD as follows.

$$\begin{aligned} A_{(1)} &= {}^1U \cdot {}^1\Sigma \cdot {}^1V^\mathsf{T} \\ A_{(2)} &= {}^2U \cdot {}^2\Sigma \cdot {}^2V^\mathsf{T} \\ A_{(3)} &= {}^3U \cdot {}^3\Sigma \cdot {}^3V^\mathsf{T} \end{aligned}$$

The left singular vectors held in matrices ${}^1U$, ${}^2U$, and ${}^3U$ of the unfoldings $A_{(1)}$, $A_{(2)}$, and $A_{(3)}$ correspond to the left singular values being factors in the Higher-Order SVD as sketched in Figure 9.6. These matrices allow the calculation of the yet missing core tensor $\mathcal{S}$.

9.3.2 Calculation of Core Tensor $\mathcal{S}$

The core tensor $\mathcal{S}$ can be calculated on basis of the left singular vectors ${}^1U$, ${}^2U$, and ${}^3U$ by a rather simple transformation of the Higher-Order SVD equation [SZL+05]:

$$\begin{aligned} \mathcal{A} &= \mathcal{S} \times_1 {}^1U \times_2 {}^2U \times_3 {}^3U \\ \mathcal{S} &= \mathcal{A} \times_1 {}^1U^\mathsf{T} \times_2 {}^2U^\mathsf{T} \times_3 {}^3U^\mathsf{T} \end{aligned}$$

This approach yields all factors that represent a Higher-Order SVD as visualized at Figure 9.6 in the beginning of this section.

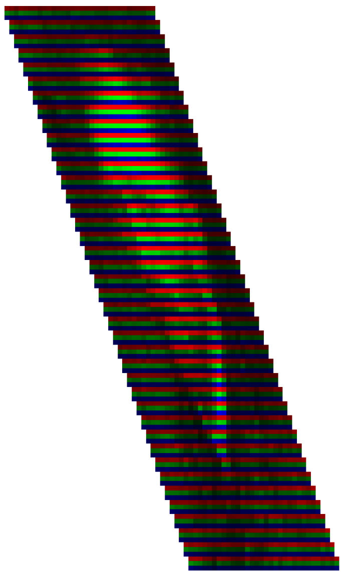


Figure 9.10: Wrapped visualization of unfolding $A_{(3)}$. The indentation marks the line break.

9.4 Geometric Interpretation of SVD Factors

The previous sections explain the motivation for representing an image as tensor and how to calculate the Higher-Order SVD of the tensor. This section is devoted to the anatomy of the factors of an SVD and their meaning using the example of a tensor representing an image.

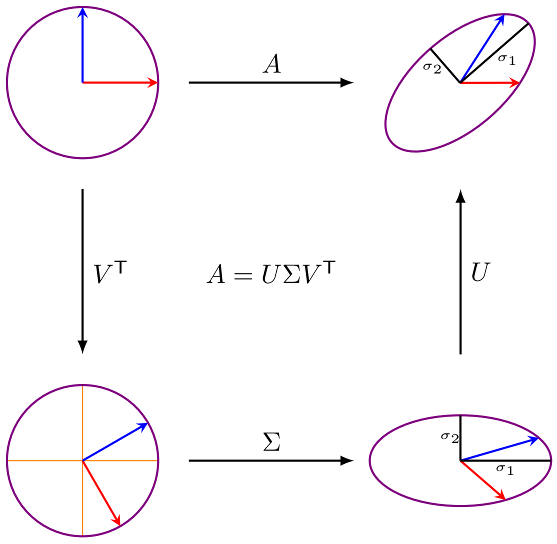


Figure 9.11: SVD as affine transformations.

The factors of a matrix SVD can be seen as linear transformations. In Figure 9.11, the vectors on the upper left side having the length 1 are transformed using matrix A. The vectors can be represented as column vectors of the identity matrix I. Note that the visualization of SVD given in Figure 9.11 exploits that, $A \cdot I = A$. Hence, matrix A can be interpreted as transformation (arrow on the upper side) and as data (sheared vectors on the upper right side).

This transformation can be decomposed using matrix SVD resulting in matrices U, Σ, and V. Instead of using matrix A directly for transforming the unit circle, matrices U, Σ, and V can be applied succesively for getting the sheared unit circle by a devious route. Obviously, the matrices U and V express rotations of the input as can be recognized from the red and blue arrows. The transformation caused by Σ scales the input. In this case, the singular value σ_2 squeezes the circle a bit along the y-axis.

Matrix U defines an orthogonal change of basis according to the principal components of the column vectors of matrix A. This can be observed on the lower right side of Figure 9.11 ($U^\mathsf{T} \cdot A$). The new first dimension represents the largest variance (see singular value σ_1) followed, by the second largest variance (see singular value σ_2). The number of singular values is equal to the rank of the decomposed matrix. The singular values are arranged in descending order.

9.5 Correlation of Pixel Rows and Reduction

This section illustrating the basis for reduction and compression can be seen as heart of this chapter. Reduction is the basis for Latent Semantic Analysis [DDF+90]. and can also be used for image compression. Looking at the results of continuously higher image compressions below gives an intuitive understanding of reduction and also of Latent Semantic Analysis.

Assume that the entries of matrix A represent color values and that they are integers. The color values can be used to visualize the correlation of two pixel rows of an image as in Figure 9.12. One point in the plot on the right-hand side encodes a pair of pixels. One of the pixels comes from row 22 and the other one from row 23, both having the same column number. For the sake of simplicity, only three pixel pairs are shown. However, since row 22 and row 23 are located next to each other, the pixels of the pixel pairs have similar colors. Thus, all pixel pairs are located next to the red dashed line. In other words, the correlation of both rows is fairly high. Comparing non-contiguous rows would yield a lower correlation in general.

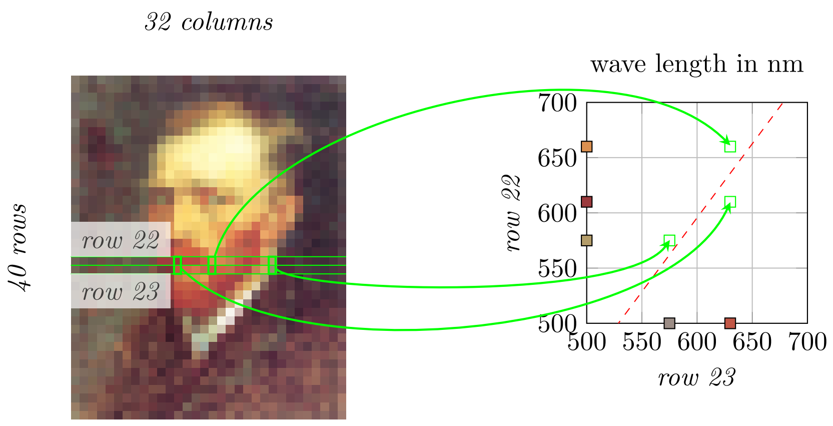


Figure 9.12: Correlation of rows. (Van Gogh, self portrait, 1887, The Art Institute of Chicago).

The red dashed line illustrates the first principal component of matrix A (cf. singular value σ_1) that can be determined by an SVD. The plot corresponds to the visualization of matrix A in the upper middle of Figure 9.13. (The Figure is an extension of Figure 9.11 above.)

Reduction can be seen as setting singular values to 0. In Figure 9.13 the smaller one of the two singular values σ_2 is set to 0 as implied by the arrow annotated with $\sigma_2 \to 0$ in the lower right. The result of the transformation is, that the red and blue arrows are projected to the remaining first principal component. Matrix U rotates the data back to the original base resulting in an approximation of matrix A: the ellipse is squeezed along its shorter semi-axis.

In terms of pixel data as shown in Figure 9.12, the green points representing pixel pairs are projected onto the red dashed line. This projection causes slight changes of the color at both of the affected lines. This makes it harder for humans to recognize such changes. In practice, pixel pairs ranging at the edges of their domain (very light or very dark brightness values) could be projected

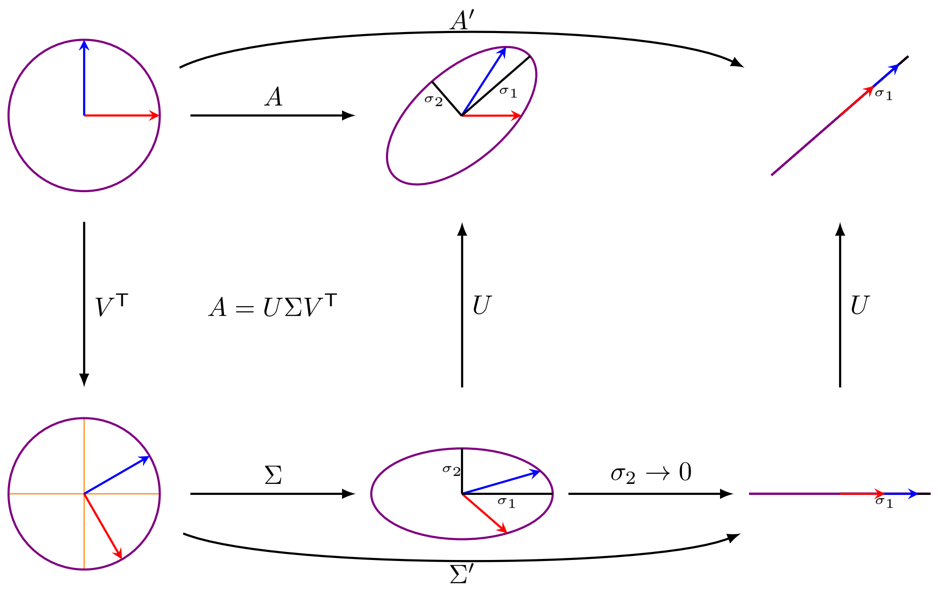


Figure 9.13: SVD and reduction as affine transformation.

outside of their domain. However, in such cases the maximum (or the minimum, respectively) of the brightness range is used to encode affected pixels.

In the reduction above, the smallest singular value was set to 0, only. Instead, the highest singular value could have been set to 0, which would have been a reduction, as well. However, since the higher singular values represent lower correlated data, changes in the original matrix A would have been more conspicuous.

This section explained reduction by means of the correlation of pixel rows of color values. Beside pixel rows (1st mode), however, also column rows (2nd mode) can be reduced. This can be done by matrix SVD for a single mode or by higher-order SVD taking all modes (rows and columns) into account at the same time. This is the main advantage of higher-order SVD over matrix SVD. (Keep in mind, that higher-order SVD requires matrix SVD on unfoldings.)

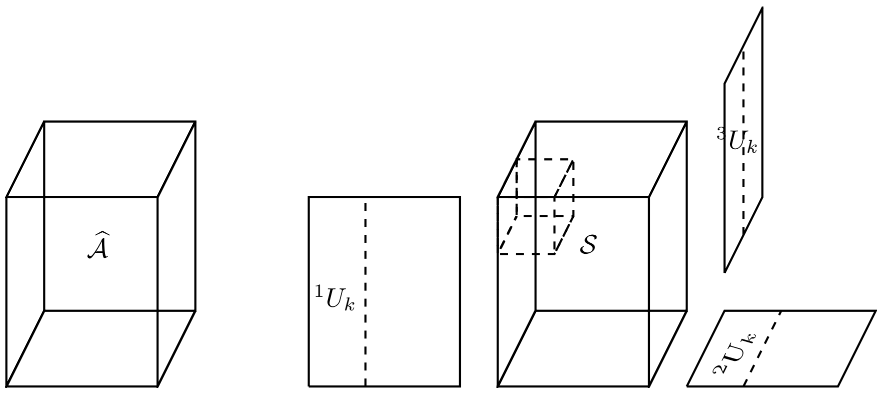


Figure 9.14: Visualization of Higher-Order SVD with reduction (dashed).

At the beginning of this chapter, images are represented using tensors with brightness values as entries. This opens rows of subpixels as third option for reduction. Now, three parameters are needed to specify the amount of reduction, one for each mode. Such a higher-order reduction is sketched in Figure 9.14 by pruning some of the complete factors using dashed lines (cf. the “left” parts of ${}^1U$, ${}^2U$, and ${}^3U$, denoted as ${}^1U'$, ${}^2U'$, and ${}^3U'$ and tensor $\mathcal{S}'$ as part of tensor $\mathcal{S}$). The product of the factors of the right side is an approximation $\widehat{\mathcal{A}}$ of the original tensor $\mathcal{A}$ preserving the same amount of rows, columns, and subpixels.

9.6 Compression by Reducing less Significant Dimensions

The goal of image compression is to save memory. A simple way to store images is using a bitmap storing each single brightness value of an image tensor $\mathcal{A}$. An alternative is storing the factors of the SVD or Higher-Order SVD. Without reduction, the factors consume significantly more memory compared to storing the original image tensor (see Figure 9.19 later in this chapter). However, using reduction makes many columns of the factors be discarded or vanish. Discarded cells represent the value 0 implicitly and do not need to be stored explicitly. At a certain reduction rate, saving the factors is getting cheaper compared to storing all brightness values.

The price for saving memory is loosing precision of the approximated image because singular vectors with higher singular values are affected. This can be recognized by humans if a certain level is exceeded. Therefore, this kind of compression is called lossy compression: The original image cannot be retrieved from the reduced factors but only a similar one.

For a visualization of an image the product of the (reduced) factors must be calculated, which obviously consumes the original amount of memory for storing the image tensor as bitmap.

9.7 Visual Evaluation of Image Compression through Higher-Order SVD

The main goal of using higher-order image compression in this thesis is evaluating the proposed algorithm (cf. Chapter 8, Algorithm 8.1) beside making Higher-Order SVD better understandable. This is done by a visual evaluation as shown in this chapter. In the following a portrait of Napoleon Bonaparte Crossing the Alps (created in 1801 by Jacques Louis David, Château de Malmaison) is compressed reducing each mode.

The grids of images in figures 9.15, 9.16, and 9.17 demonstrate reduction in steps of 10% for grid rows and columns, and in steps of $\frac{1}{3}$ of singular values for the color combinations of pixels. The grid columns represent increasing reduction of image rows, the grid columns represent the reduction of image columns, where Figure 9.15 represents $\frac{3}{3}$ of mode 3, Figure 9.16 $\frac{2}{3}$ of mode 3, and Figure 9.17 $\frac{1}{3}$ of mode 3. Hence, the best approximation of the image is shown in the (first) Figure 9.15 in the upper left, and the most inexact approximation is shown in the (last) Figure 9.17 in the lower right part. Moving right and/or down, results are a more compressed but a less precise image.

Each individual image in Figures 9.15, 9.16, and 9.17 was obtained by applying the algorithm from Chapter 8 to the 3rd-order tensor representing the original image and re-visualising the result. If a human scans a sequence of individual images in the direction of a row, a column, or figure-to-figure, each individual image looks to the human eye very plausibly as it should according to its position in the sequence. There are no unexpected blotches, odd-looking colours, abnormal brightness values or similar disturbances in the images.

This fact can be considered evidence for the validity of the algorithm and its implementation, and this evidence was the primary purpose of this experiment.

However, although image compression is not the focus of this thesis, the rest of this section discusses some aspects of compression based on Higher-Order SVD in more detail.

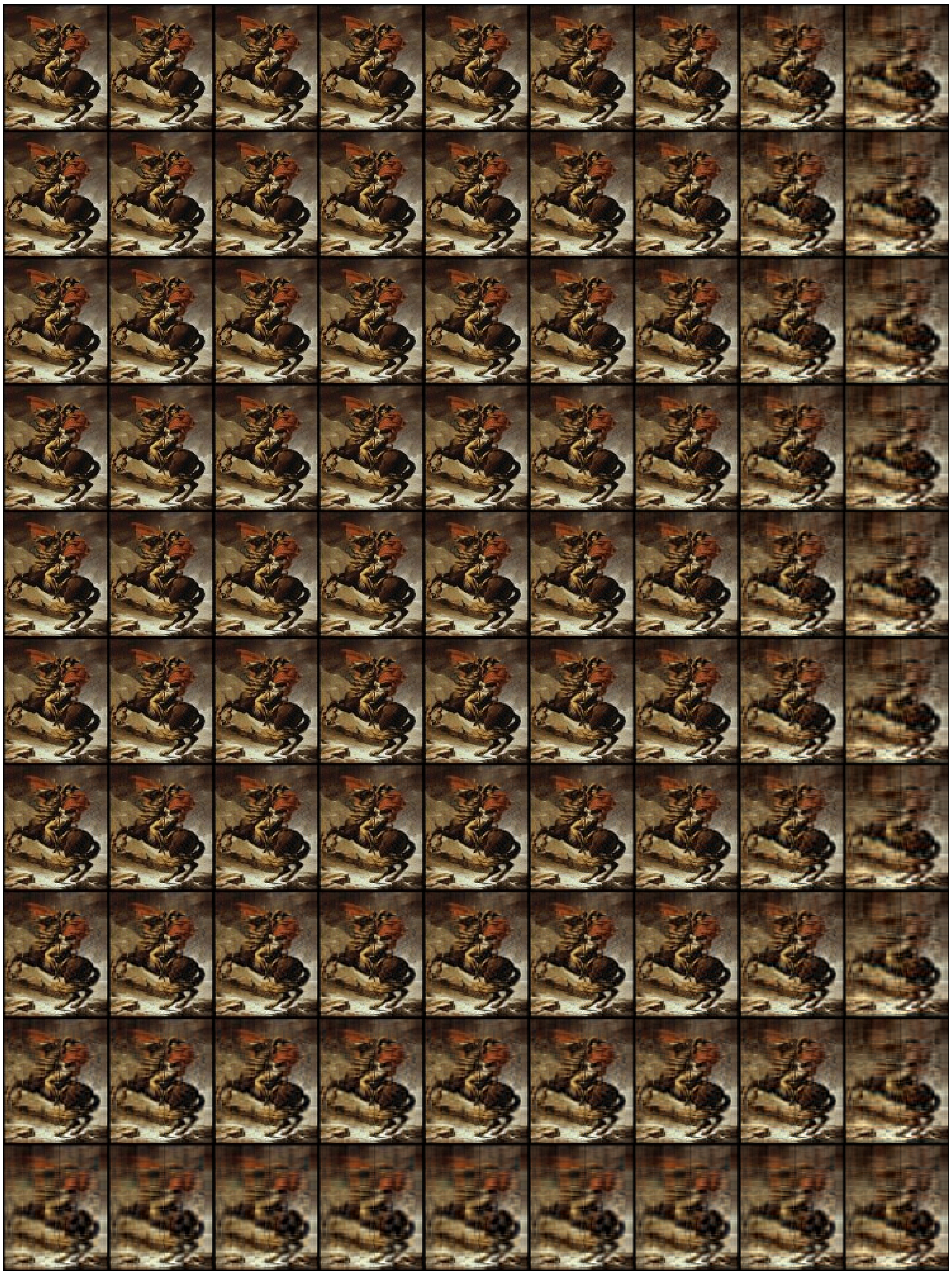


Figure 9.15: Reduction of mode 1 column-wise and of mode 2 row-wise keeping $\frac{3}{3}$ of mode 3. Jacques Louis David, Napoleon Bonaparte Crossing the Alps, 1801, Malmaison

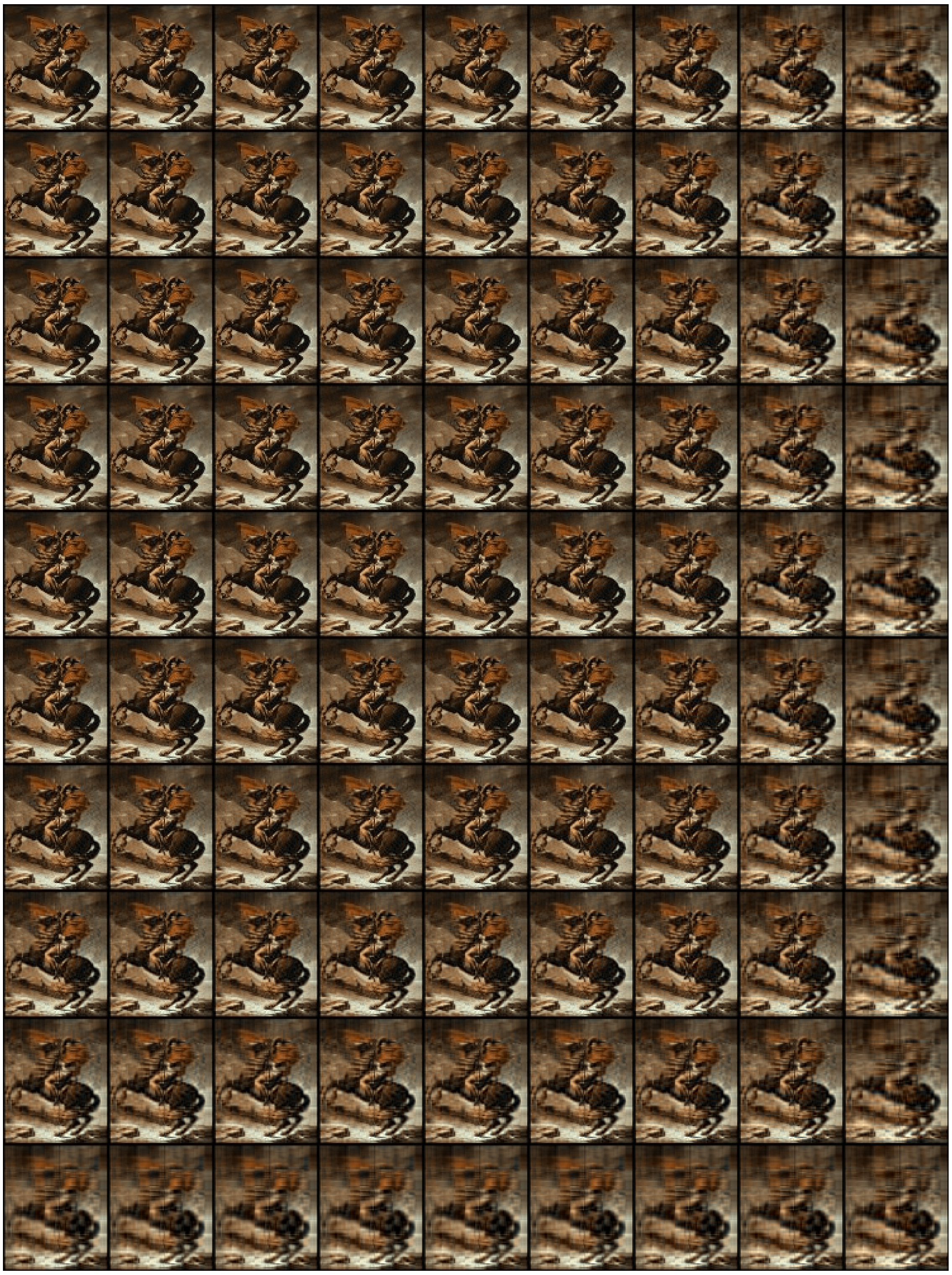


Figure 9.16: Reduction of mode 1 column-wise and of mode 2 row-wise keeping $\frac{2}{3}$ of mode 3. Jacques Louis David, Napoleon Bonaparte Crossing the Alps, 1801, Malmaison

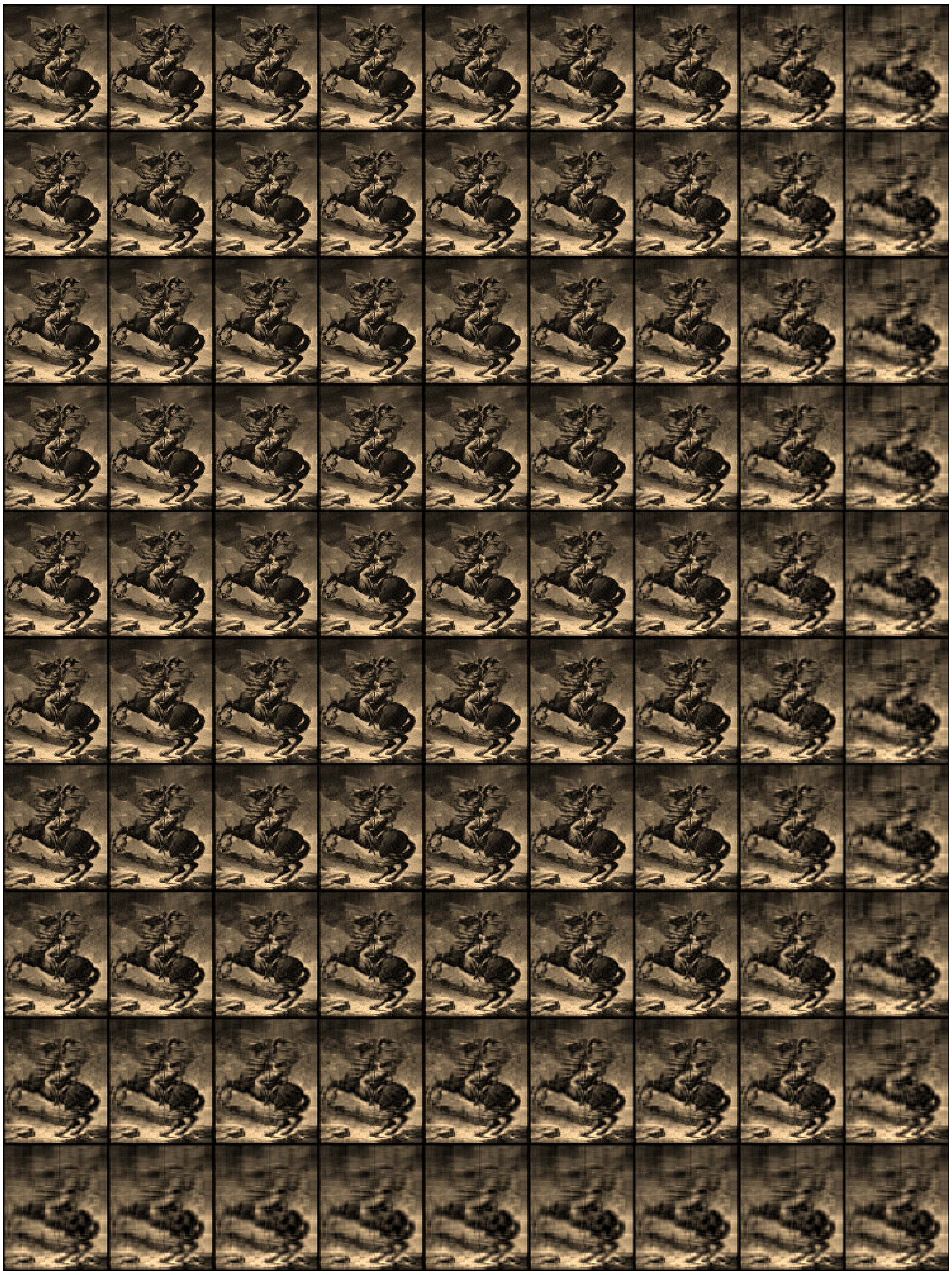


Figure 9.17: Reduction of mode 1 column-wise and of mode 2 row-wise keeping $\frac{1}{3}$ of mode 3. Jacques Louis David, Napoleon Bonaparte Crossing the Alps, 1801, Malmaison

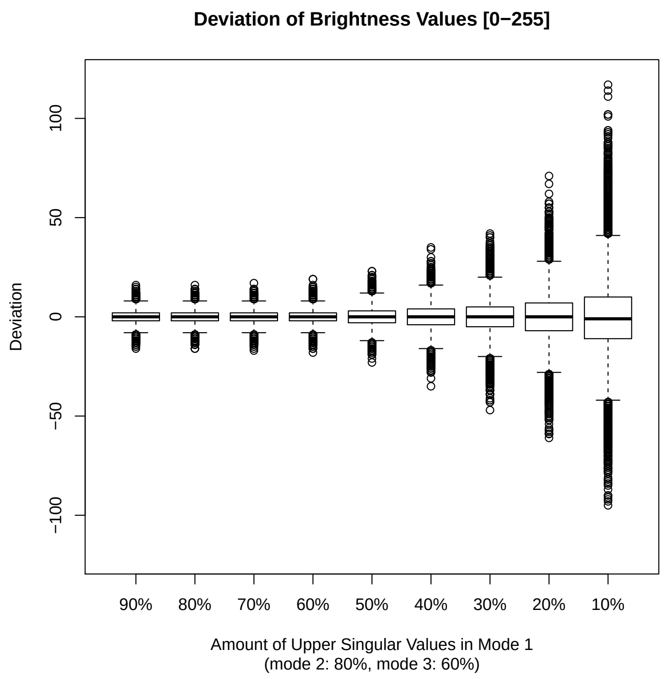


Figure 9.18: Distribution of the deviation of brightness values (cf. row 2 in Figure 9.16).

The deviation of brightness values is normally distributed (cf. Figure 9.18). The plot visualizes the deviations of the 2nd row of Figure 9.16. The distributions are represented by box plots. In a box plot the box contains half of all values, the line inside the box is the median separating the halves of values from each other. The length of the dotted lines is 1.5 times the length of the box. Circles denote outliers. The variance increases significantly below about 30% of reduction.

The compression rate for the compressions shown above can be determined as follows. First, the reference value for the rate is the memory consumption for the image stored as bitmap representing each pixel as brightness values in red, green, and blue color. In total length · widthdepth values of a certain range (e.g. 0–255) must be stored. For the image of Napoleon being reduced in Figures 9.15, 9.16, and 9.17 $83 \cdot 100 \cdot 3 = 24,900$ values are needed (ignoring image dimensions and other meta data). Second, the variable part of the compression rate is the memory needed for storing the factors of the higher-order SVD. Without any reduction, this is $83^2 + 100^2 + 3^3$ for the singular vectors of all modes, and $83 \cdot 100 \cdot 3$ values for the core tensor, which is in total $41,816$ values. Hence, the formula for reduced images is:

$$\lceil m_1 \cdot \text{height}^2 \rceil + \lceil m_2 \cdot \text{width}^2 \rceil + \lceil m_3 \cdot \text{depth}^3 \rceil + \lceil m_1 \cdot \text{height} \rceil \cdot \lceil m_2 \cdot \text{width} \rceil \cdot \lceil m_3 \cdot \text{depth} \rceil,$$

where the $m_i \in [0,1], i \in 1,2,3$ represent the amount of remaining singular values and corresponding singular vectors. Figure 9.19 lists the compression rates in proportion to the memory needed for storing the image as bitmap. Therefore, compression rates over 100% are possible, which, obviously, is a bad choice.

Apart from the fairly bad compression rates in the upper left of the table, significantly good compression rates are possible as shown in the lower right of the table in Figure 9.19. Unfortunately, the compression is lossy. Figure 9.20 gives a summary of expected deviance of brightness values. The higher the expected deviance, the worse is the quality of the approximation.

First experiments indicate that a reduction of about $\frac{2}{3}$ of rows and columns, and $\frac{1}{3}$ in depth are the borderline for approximations to be recognized by humans or not. This level of reduction means a memory reduction of about $\frac{3}{4}$ compared to storing an image as bitmap.

Mode 3	Mode 2	Mode 1 (row)								
(depth)	(column)	90 %	80 %	70 %	60 %	50 %	40 %	30 %	20 %	10 %
$\frac{3}{3}$	100 %	154	140	126	112	98	84	70	56	42
	90 %	142	129	116	103	90	77	64	51	38
	80 %	131	119	107	95	83	71	59	46	34
	70 %	120	108	97	86	75	64	53	42	31
	60 %	107	97	87	77	67	57	47	37	27
	50 %	96	87	77	68	59	50	41	32	23
	40 %	84	76	68	60	52	44	36	27	19
	30 %	72	65	58	51	44	37	29	22	15
	20 %	60	54	48	42	36	30	24	18	12
	10 %	49	44	39	33	28	23	18	13	8
$\frac{2}{3}$	100 %	124	113	103	92	81	70	60	49	38
	90 %	115	105	95	85	75	65	55	45	35
	80 %	107	97	88	79	69	60	50	41	32
	70 %	98	89	81	72	63	54	46	37	28
	60 %	89	81	73	65	57	49	41	33	25
	50 %	80	73	66	58	51	43	36	29	21
	40 %	72	65	58	52	45	38	31	25	18
	30 %	63	57	51	45	38	32	26	20	14
	20 %	54	49	43	38	33	27	22	16	11
	10 %	45	41	36	31	27	22	17	12	8
$\frac{1}{3}$	100 %	94	86	79	72	64	57	50	42	35
	90 %	88	81	74	67	60	53	46	39	32
	80 %	83	76	69	62	56	49	42	36	29
	70 %	77	70	64	58	51	45	39	32	26
	60 %	71	65	59	53	47	41	35	29	23
	50 %	65	59	54	48	42	37	31	25	20
	40 %	60	54	49	43	38	33	27	22	16
	30 %	54	48	43	38	33	28	23	18	13
	20 %	48	43	38	34	29	24	20	15	10
	10 %	42	38	33	29	25	20	16	12	7

Figure 9.19: Rates of memory consumption in percent regarding Figures 9.15, 9.16, and 9.17.

9.8 Relevance of this evaluation for the Semantic Analysis of ARTigo data

This chapter used the perception of compressed images for validating matrix SVDs and Higher-Order SVDs by humans on one hand. On the other hand, the effect of reducing data was visualized for a better understanding of Latent Semantic Indexing used for the semantic analysis of ARTigo data. While the focus of compression is saving memory, the focus of latent semantic indexing is eliminating statistical noise learning latent variables only. For latent semantic indexing a matrix entry stands for how often an object was tagged, or in case of Higher-Order Latent Semantic Indexing, a tensor entry encodes whether a user tagged an object.

A major difference between a tensor representing image data and ARTigo data is that the tensor representing ARTigo data is very sparse because players tag only a minor parts of the image collection with a part of the whole bandwidth of tags. However, this is supposed to affect memory consumption only, and does not affect the approach. Note that reducing a sparse tensor results in a dense tensor as approximation.

As a side note, compression based on Higher-Order SVD can easily be applied to other applications like videos. Videos can be seen as images in a row requiring one more tensor mode: a 4th-order tensor. Even audio data can be compressed using the SVD approach [BSWD03]. Con-

Mode 3	Mode 2	Mode 1 (row)								
(depth)	(column)	90 %	80 %	70 %	60 %	50 %	40 %	30 %	20 %	10 %
$\frac{3}{3}$	100 %	0	0	0	1	1	2	2	4	6
	90 %	0	0	0	1	1	2	2	4	6
	80 %	0	0	0	1	1	2	2	4	6
	70 %	1	1	1	1	1	2	2	4	6
	60 %	1	1	1	1	1	2	2	4	6
	50 %	1	1	1	1	2	2	2	4	6
	40 %	2	2	2	2	2	2	2	4	6
	30 %	3	3	3	3	3	3	3	4	6
	20 %	4	4	4	4	4	4	4	4	6
	10 %	6	6	6	6	6	6	6	6	6
$\frac{2}{3}$	100 %	1	1	1	1	1	2	3	4	6
	90 %	1	1	1	1	1	2	3	4	6
	80 %	1	1	1	1	1	2	3	4	6
	70 %	1	1	1	1	1	2	3	4	6
	60 %	1	1	1	1	1	2	3	4	6
	50 %	2	2	2	2	2	2	3	4	6
	40 %	2	2	2	2	2	2	3	4	6
	30 %	3	3	3	3	3	3	3	4	6
	20 %	4	4	4	4	4	4	4	4	6
	10 %	6	6	6	6	6	6	6	6	6
$\frac{1}{3}$	100 %	3	3	3	3	3	3	4	5	6
	90 %	3	3	3	3	3	3	4	5	6
	80 %	3	3	3	3	3	3	4	5	6
	70 %	3	3	3	3	3	3	4	5	6
	60 %	3	3	3	3	3	3	4	5	6
	50 %	3	3	3	3	3	3	4	5	6
	40 %	4	4	4	4	4	4	4	5	6
	30 %	4	4	4	4	4	4	4	5	6
	20 %	5	5	5	5	5	5	5	5	6
	10 %	7	7	7	7	7	7	7	7	7

Figure 9.20: Deviation of brightness values in percent regarding Figures 9.15, 9.16, and 9.17

ceivably, video and audio compression should be done separately for efficiency reasons and because the correlation between audio and video signals might be low.

Chapter 10

Parallel Higher-Order SVD

The factorization of a tensor via Higher-Order SVD is expensive from a computational point of view. Essentially, one matrix SVD needs to be calculated for each mode of a tensor. Hence, a 3rd order tensor requires the factorization of 3 matrices. The remaining operations of a Higher-Order SVD are computationally cheaper. It is therefore useful to try gaining efficiency by parallel approaches, which focus in particular on parallelising the necessary matrix SVDs. The following describes the backbone of how Higher-Order SVD is parallelised in this chapter.

```
Input          dth-order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots \times I_d}$,
               pruning tuple $(m_1, \ldots, m_d) \in [1, I_1] \times \ldots \times [1, I_d]$
Output         dth-order tensor $\widehat{\mathcal{A}} \in \mathbb{R}^{I_1 \times \ldots \times I_d}$
               as the best rank-$(m_1, ..., m_d)$ approximation to $\mathcal{A}$

Unfolding      parallelFor $i = 1, \ldots, d$
                 Compute the unfolding $A_{(i)}$ of $\mathcal{A}$
               end
Matrix SVD     parallelFor $i = 1, \ldots, d$
                 Compute the SVD $A_{(i)} = U_i \Sigma_i V_i^\mathsf{T}$
               end
Pruning        parallelFor $i = 1, \ldots, d$
                 $W_i := [u_{i,1}, \ldots, u_{i,m_i}]$ where $u_{i,j}$ column vectors of $U_i$
               end
Core Tensor    Compute $\mathcal{S} := \mathcal{A} \times_1 W_1^\mathsf{T} \times_2 W_2^\mathsf{T} \ldots \times_d W_d^\mathsf{T}$
Approximation  Compute $\widehat{\mathcal{A}} := \mathcal{S} \times_1 W_1 \times_2 W_2 \ldots \times_d W_d$
```

Algorithm 10.1: Parallel Higher-Order SVD uses parallel for-loops and parallel SVD (Algorithm 10.2)

The parallelisation of the Higher-Order SVD algorithm of section 8.2 consists of both, a straightforward parallelisation of independent computations and on a parallelisation of matrix SVD. The following steps for computing Higher-Order SVD (algorithm 10.1) are the subject of the parallelisation intent:

- *Unfolding:* Computing the unfoldings $A_{(i)}$ is independent from each other. They can be performed in parallel.
- *Matrix SVD:* The SVDs of the unfoldings $A_{(i)}$ are independent from each other. They can be performed in parallel. The result of this step is necessary for the computation of the following steps. Below in this chapter the computation of an SVD itself will be subject to the parallelisation intent.
- *Pruning:* The prunig of the (independent) column vectors of U_i can be performed in parallel.

- *Core Tensor:* The result of an n-mode product is a tensor. An entry of the resulting tensor does not depend on the other entries. Thus this step can be performed in parallel.
- *Approximation:* As the approximation consists of n-mode products. it can be performed in parallel.

Obviously, the mentioned steps describe a straightforward parallelisation of the Higher-Order SVD algorithm. However, a parallel computation of matrix SVD itself is needed for achieving the time complexity established in section 10.2.

This thesis reports about two approaches of parallelising matrix SVD Section 10.1 proposes a parallelisation of SVD based on Hestenes' method [BL85, CC89]. This approach is suited for multi-processor architectures (cf. Figure 10.1). Section 10.3 proposes a distributed parallel computation of matrix SVD based on R-SVD. This method is suitable for a calculation based on MapReduce addressed in Section 10.4.

10.1 Parallel Matrix SVD based on Hestenes' Method

This section is based on a publication [SWB12] of which the author of this thesis is coauthor.

Input	matrix $A \in \mathbb{R}^{m \times n}$, precision $\varepsilon \geq 0$, constant s [DLDMV00]
Output	matrix $W \in \mathbb{R}^{m \times n}$, matrix $V \in \mathbb{R}^{n \times n}$
Assume	$n/2$ processors are working in parallel

```
$A_1 := A$ , $V_1 := I$ , $z := 0$
for $n = 1, \ldots, s$
  $L_k := 2k - 1$ with $k \leq n/2$
  $R_k := 2k$     with $k \leq n/2$
  for $i = 1, \ldots, n-1$
    $z := z + 1$ ;
    if $L_k < R_k$
      then ORTH($A_z, V_z, \varepsilon, L_k, R_k$)
      else ORTH($A_z, V_z, \varepsilon, R_k, L_k$)
    if $k = 1$ then $outR_k := R_k$
      else if $k < n/2$
        then $outR_k := L_k$
    if $k > 1$
      then $outL_k := R_k$
    {wait for outputs to propagate to
     inputs of adjacent processors}
    if $k < n/2$
      then $R_k := inR_k$
      else $R_k := L_k$
    if $k > 1$
      then $L_k := inL_k$
  end
end
$W := A_{z+1}$ , $V := V_{z+1}$
```

Algorithm 10.2: Parallel Matrix SVD based on Hestenes' method and the scheme in Figure 10.1.

Parallel Matrix SVD (algorithm 10.2) computes the SVD of a matrix $A = U\Sigma V^{\mathsf{T}}$ using the *Hestenes' Method* [BL85, Hes58] as follows: An orthogonal matrix V is constructed such that the matrix $W = AV$ has orthogonal columns in several orthogonalisation steps. One step orthogonalises two columns with plane rotations (ORTH algorithm 10.3). These steps are performed in parallel (algorithm 10.2). Finally for the SVD, the matrix U is obtained by normalising the length of each nonzero column of the matrices $W = [w_1, \ldots, w_n]$ and with $W = U\Sigma$ yields $\Sigma = diag(\|w_1\|, \ldots, \|w_n\|)$. Thus $W = AV$ implies the SVD of $A = U\Sigma V^{\mathsf{T}}$.

Parallel Matrix SVD (algorithm 10.2) uses the following communication scheme between the processors $P_1, \ldots, P_p$, where $n/2 = p$ is the total number of processors p in this scheme: The processors are arranged in logical groups g_i with $i = 1, \ldots, d$. Two adjacent processors P_k and P_{k+1} can communicate. Each processor P_k has registers R_k and L_k with output and input lines $outR_k$, $outL_k$ (and inR_k, inL_k, resp.). The output line $outR_k$ is connected to the input line inL_{k+1}. The scheme for 3 processors is described in figure 10.1.

ORTH (algorithm 10.3) is used in Parallel Matrix SVD (algorithm 10.2) aiming at approximating the orthogonalisation of two columns i and j of a matrix A_k up to a precision ε. ORTH yields the matrix A_{k+1} operating on a single processor. ORTH (algorithm 10.3) is the ORTH algorithm of Chuang [CC89] with a slight extension: The matrix V_k (of the SVD $A_k = U_k \Sigma_k V_k^{\mathsf{T}}$) is additionally computed. Computing the matrix V in ORTH is cheaper in terms of time complexity than reconstructing V out of the matrices W and A with $W = VA$.

Input	matrix $A_k \in \mathbb{R}^{m\times n}$, matrix $V_k \in \mathbb{R}^{n\times n}$, precision $\varepsilon \geq 0$, column indices $i, j \in \mathbb{N}$, $i, j \leq n$
Output	matrix $A_{k+1} \in \mathbb{R}^{m\times n}$, matrix $V_{k+1} \in \mathbb{R}^{n\times n}$

```
$A_{k+1} := A_k$
$V_{k+1} := V_k$
$\gamma := a_{k,i}^{\mathsf{T}} a_{k,j}$
if $|\gamma| > \epsilon$ then
    $\alpha := \|a_{k,i}\|^2$
    $\beta := \|a_{k,j}\|^2$
    $\xi := \frac{\beta-\alpha}{2\gamma}$
    $t := \frac{sign(\xi)}{|\xi|+\sqrt{1+\xi^2}}$
    $cos(\theta) := \frac{1}{\sqrt{1+t^2}}$
    $sin(\theta) := t \cdot cos(\theta)$
    $[a_{k+1,i}, a_{k+1,j}] := [a_{k,i}, a_{k,j}] \cdot \begin{pmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{pmatrix}$
    $[v_{k+1,i}, v_{k+1,j}] := [v_{k,i}, v_{k,j}] \cdot \begin{pmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{pmatrix}$
end
```

Algorithm 10.3: ORTH

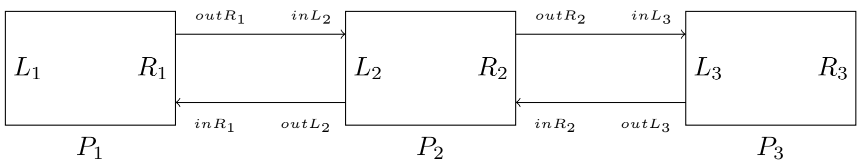


Figure 10.1: Communication scheme for 3 processors.

10.2 Time Complexity of Parallel Higher-Order SVD based on Hestenes' Method

As already mentioned, Higher-Order SVD is expensive from a computational point of view. Without parallelisation, the serial Higher-Order SVD algorithm 8.1 has the time complexity $O(I \cdot \prod_{i=1}^{d} I_i)$, $I := \max\{I_1, \ldots, I_d\}$. In the following, a proof is given for the time complexity of a parallel version of Higher-Order SVD based on Hestenes' method (cf. algorithm 10.1), which is $O(n^{d+1} \cdot log(n)/p)$ or even $O(n^{d+1}/p)$ where n denotes data per d dimensions, and p the number of processors. The proof is needed to show that a *parallelisation* of Higher-Order SVD can scale down the time complexity in proportion to the *number of processors* used.

Note that the following Section 10.3 realizes Higher-Order SVD based on R-SVD instead of Hestenes' method to compute matrix SVDs. This has technical reasons concerning the execution environment of MapReduce (cf. Section 10.4). A proof of the time complexity of Higher-Order SVD based on R-SVD is not in the scope of this thesis but obviously ought to be done as part of future work.

The proof is arranged as follows. Section 10.2.1 proves the time complexity of the serial Higher-Order SVD algorithm in preparation of the parallel version. Section 10.2.2 proves the time complexity of parallel matrix SVD and, finally, Section 10.2.3 incorporates the time complexity of parallel matrix SVD into the time complexity of parallel Higher-Order SVD.

10.2.1 Time Complexity of Serial Higher-Order SVD

```
Input           dth-order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_d}$,
                pruning tuple $(m_1, \dots, m_d) \in [1, I_1] \times \dots \times [1, I_d]$
Output          dth-order tensor $\widehat{\mathcal{A}} \in \mathbb{R}^{I_1 \times \dots \times I_d}$
                as the best rank-$(m_1, ..., m_d)$ approximation to $\mathcal{A}$

 1 Unfolding      for $i = 1, \dots, d$
 2                  Compute the unfolding $A_{(i)}$ of $\mathcal{A}$
 3                end
 4 Matrix SVD     for $i = 1, \dots, d$
 5                  Compute the SVD $A_{(i)} = U_i \Sigma_i V_i^\mathsf{T}$
 6                end
 7 Pruning        for $i = 1, \dots, d$
 8                  $W_i := [u_{i,1}, \dots, u_{i,m_i}]$where $u_{i,j}$column vectors of $U_i$
 9                end
10 Core Tensor    Compute $\mathcal{S} := \mathcal{A} \times_1 W_1^\mathsf{T} \times_2 W_2^\mathsf{T} \dots \times_d W_d^\mathsf{T}$
11 Approximation  Compute $\widehat{\mathcal{A}} := \mathcal{S} \times_1 W_1 \times_2 W_2 \dots \times_d W_d$
```

Algorithm 10.4: Serial Higher-Order SVD (algorithm 8.1 repeated).

lines 1–3	d times
line 2	$O(I_1 \cdot \ldots \cdot I_d)$
lines 4–6	$O(\sum_{i=1}^{d} (I_i \cdot \prod_{j=1}^{d} I_j)) = O(I \cdot \prod_{j=1}^{d} I_j)$
lines 7–9	$o(d \cdot \prod_{j=1}^{d} I_j) = o(\prod_{j=1}^{d} I_j)$
lines 10/11	$O(I \cdot \prod_{j=1}^{d} I_j)$

Figure 10.2: Time complexity of serial Higher-Order SVD (cf. algorithm 10.4)

The costs of the unfoldings $A_{(1)}, \dots, A_{(d)}$ are linear in the number of entries in the tensor.

Note: $A = U\Sigma V^\mathsf{T} \iff A^\mathsf{T} = V\Sigma^\mathsf{T} U^\mathsf{T}$ and the time complexity of computing the SVD of a matrix $A \in \mathbb{R}^{m \times n}$ is $O(mn^2)$, i.e. the time complexity of the SVD of the transposed matrix A^T is $O(m^2 n)$ and transposing costs at most $O(mn)$.

The computational costs of the SVD of $A_{(i)} \in \mathbb{R}^{I_i \times (I_{i+1} \cdot \ldots \cdot I_d I_1 \cdot \ldots \cdot I_{i-1})}$ can be minimised by computing the SVD of $A_{(i)}^\mathsf{T} \in \mathbb{R}^{(I_{i+1} \cdot \ldots \cdot I_d I_1 \cdot \ldots \cdot I_{i-1}) \times I_i}$, which is $O(I_i \cdot I_1 \cdot \ldots \cdot I_d) = O(I_i \cdot \prod_{j=1}^{d} I_j)$, thus line 4–6 costs $\sum_{i=1}^{d} O(I_i \cdot \prod_{j=1}^{d} I_j) = O(\sum_{i=1}^{d} (I_i \cdot \prod_{j=1}^{d} I_j))$.

If $I := \max\{I_1, \dots, I_d\}$, then $O(\sum_{i=1}^{d} (I_i \cdot \prod_{j=1}^{d} I_j)) = O(d \cdot I \cdot \prod_{j=1}^{d} I_j) = O(I \cdot \prod_{j=1}^{d} I_j)$, as d is fixed.

Also line 7–9 can be implemented efficiently because pruning the last $I_i - m_i$ columns of U_i can be done in constant time. The slowest method would be to read every single entry and to store only the needed elements into a new matrix. This would cost $O(\prod_{j=1}^{d} I_j)$ for each matrix, thus $O(\prod_{j=1}^{d} I_j)$ is an upper bound. However, it would also be possible just to ignore the last columns of U_i which results in $O(1)$; but as $1 \ll I_d \cdot \prod_{j=1}^{d} I_j$ and $\prod_{j=1}^{d} I_j \ll I_d \cdot \prod_{j=1}^{d} I_j$ this won't change the total time complexity.

The n-mode product ($1 \leq n \leq d$) of a tensor $\mathcal{F} \in \mathbb{R}^{I_1 \times \dots \times I_d}$ and a matrix $M \in \mathbb{R}^{J_n \times I_n}$ has a time complexity of: $O(\prod_{j=1}^{d} I_j \cdot J_n)$, e.g. for n=2 the product has $\prod_{i=1, i \neq 2}^{d} I_i \cdot J_2 = I_1 \cdot J_2 \cdot I_3 \cdot \ldots \cdot I_d$ entries and to compute one entry I_2 numbers need to be summarized. Remember: $(\mathcal{F} \times_n M)_{i_1 \dots i_{n-1} j_n i_{n+1} \dots i_d} = \sum_{i_n \leq I_n} f_{i_1 \dots i_d} m_{j_n i_n}$

Thus $\mathcal{A}\times_1 W_1$ needs $O(\prod_{i=1}^d I_i\cdot m_1) = O(\prod_{i=1}^d I_i\cdot I_1)$, $(\mathcal{A}\times_1 W_1^\mathsf{T})\times_2 W_2$ needs $O(\prod_{i=1}^d I_i\cdot m_1 + \prod_{i=2}^d I_i\cdot m_1\cdot m_2)$ and finally $(\ldots(\mathcal{A}\times_1 W_1^\mathsf{T})\times_2\ldots)\times_d W_d^\mathsf{T})$ needs $O(\prod_{i=1}^d I_i\cdot m_1+\ldots+I_d\cdot\prod_{i=1}^d m_i) = O(\prod_{i=1}^d I_i\cdot I)$ with $I := \max\{I_1,\ldots,I_d\}$ $(m_i\leq I_i)$. As $m_1,\ldots,m_d$ have switched roles with $I_1,\ldots,I_d$ in line 11, the time complexity will be the same. (line 10: $\mathcal{A}\in\mathbb{R}^{I_1\times\ldots\times I_d}, W_1^\mathsf{T}\in\mathbb{R}^{m_1\times I_1},\ldots,W_d^\mathsf{T}\in\mathbb{R}^{m_d\times I_d}$, line 11: $\mathcal{S}\in\mathbb{R}^{m_1\times\ldots\times m_d}, W_1\in\mathbb{R}^{I_1\times m_1},\ldots,W_d\in\mathbb{R}^{I_d\times m_d}$)

The total time complexity of algorithm 8.1 in its serial version is $O(d\cdot\prod_{i=1}^d I_i + I\cdot\prod_{i=1}^d I_i + \prod_{i=1}^d I_i + 2I\cdot\prod_{i=1}^d I_i) = O((d+3I+1)\cdot\prod_{i=1}^d I_i) = O(I\cdot\prod_{i=1}^d I_i)$. This result is published in [SWB12], which this section is based on.

10.2.2 Time Complexity of Parallel Matrix SVD

With p processors and n columns, the *parallel SVD* of an $m\times n$ matrix A (cf. algorithm 10.2) has a time complexity of $O(mn^2\cdot s/p)$ [CC89], if s is the number of needed sweeps (according to [BL85, CC89] s could be $O(\log(n))$). However, as mentioned in [BL85] s can be chosen as a constant between 6 and 10 for practical purposes.

10.2.3 Time Complexity of Parallel Higher-Order SVD

lines 1–3	$d\cdot O(1\cdot\frac{I_1\cdot\ldots\cdot I_d}{p_d}) = O(\frac{\prod_{i=1}^d I_i}{p})$
lines 4–6	$d\cdot O(\frac{I\cdot\prod_{i=1}^d I_i}{p}s)$
lines 7–9	$o(\frac{d}{p}\cdot\prod_{j=1}^d I_j) = o(\frac{\prod_{j=1}^d I_j}{p})$
lines 10–11	$2\cdot O(\frac{I\cdot\prod_{j=1}^d I_j}{p})$

Figure 10.3: Time Complexity of Parallel higher-order SVD (Algorithm 10.4, parallel version)

The calculation of the d unfoldings $A_{(i)}$ could be implemented efficiently, like in the normal higher-order SVD algorithms. But this time p processors are available for the computation, which could work independently.

The computational costs of the SVD of $A_{(i)}\in\mathbb{R}^{I_i\times(I_{i+1}\cdot\ldots\cdot I_d I_1\cdot\ldots\cdot I_{i-1})}$ using the *parallel SVD* is $O(\frac{s_i}{p_i}\cdot I_i\cdot(\prod_{j=1;j\neq i}^d I_j)^2)$. (Remember that the time complexity of the parallel SVD using Hestenes' method is $O(s\frac{mn^2}{p})$ if p processors are available for decomposing a $m\times n$ matrix.)

This time could be *optimised* by computing the SVD of the transposed ($A=U\Sigma V^\mathsf{T}$ iff $A^\mathsf{T} = V\Sigma^\mathsf{T}U^\mathsf{T}$) unfolding $A_{(i)}^\mathsf{T}\in\mathbb{R}^{(I_{i+1}\cdot\ldots\cdot I_d I_1\cdot\ldots\cdot I_{i-1})\times I_i}$ (if $I_{i+1}\cdot\ldots\cdot I_d I_1\cdot\ldots\cdot I_{i-1} > I_i$), which has a time of $O(\frac{s_i}{p_i}\cdot I_i\cdot(\prod_{j=1}^d I_j))$. If $s := \max\{s_i : 1\leq i\leq d\}$, then the maximal time complexity of all parallel SVDs together will be $d\cdot O(s\cdot\frac{I\cdot\prod_{i=1}^d I_i}{p})$. Again, s is a constant or logarithmically depends on products of the variables of $I_1,\ldots,I_d$.

Line 7–9 could be implemented efficiently, as well. As only the last $m_1,\ldots,m_d$ columns of U_i are pruned.

One processor needs $O(I\cdot\prod_{i=1}^d I_i)$ for line 10 and 11 as described in the paragraph about the serial higher-order SVD algorithm (Algorithm 8.1). And because p processors are available and each processor could compute one entry of a x-order tensor product, the complexity is $O(I\cdot\frac{1}{p}\cdot\prod_{j=1}^d I_j)$.

To sum it up: the parallel Higher-Order SVD algorithm has a time complexity of

$$O(s\cdot I\cdot\frac{1}{p}\cdot\prod_{i=1}^d I_i)$$

In other words, using the parallel Higher-Order SVD algorithm is p-times faster than a single-threaded Higher-Order SVD algorithm, if p is the number of processors on the same system.

Hence, a parallel Higher-Order SVD of a dth-order tensor $\mathcal{A} \in \mathcal{R}^{n\times\cdots\times n}$ has a time complexity of $O(n^{d+1} \cdot s/p)$. As mentioned above s, is either a constant or depends logarithmically on n.

10.3 Parallel Higher-Order SVD based on R-SVD

This section reports on Diego Havenstein's contribution resulting from his bachelor thesis [Hav12] supervised by François Bry and the author of this thesis. The contribution allows a distributed computation of Higher-Order SVD using MapReduce [DG08]. Essentially, Havenstein uses R-SVD [GL96] instead of Hestenes' method (cf. Section 10.1) for calculating the matrix SVDs of unfoldings in the Higher-Order SVD algorithm (cf. Section 8.2 and below). The reason for using R-SVD is that major part of R-SVD (in contrast to Hestenes' method) can be reasonably parallelised based on the MapReduce approach like, e.g., QR decomposition.

Recall that the Higher-Order SVD of a tensor $\mathcal{A}$ can be computed by

1. getting the **unfoldings** $A_{(i)}$ of tensor $\mathcal{A}$,
2. computing the **matrix SVDs**[1] of the unfoldings $A_{(i)}$.
3. **pruning** the matrix SVDs (determining the degree of approximation of tensor $\mathcal{A}$),
4. computing the **core tensor** based on tensor $\mathcal{A}$ and the matrix SVDs (cf. step 2), and
5. computing the **approximation** tensor $\widehat{\mathcal{A}}$ of tensor $\mathcal{A}$.

While the general Higher-Order SVD algorithm can be parallelised straightforwardly (cf. Section 10.1), parallelising the computation of matrix SVDs in step 2 is the main challenge for parallel Higher-Order SVD based on MapReduce.

The following steps exemplarily implement step 2 of the Higher-Order SVD above using R-SVD for the decomposition of a 3rd-oder tensor. The generalized approach combining the Higher-Order SVD algorithm and R-SVD is formally given in algorithm 10.5 below.

1. For each tensor unfolding $A_{(i)}$, compute its (full) QR decomposition

   $$A_{(i)} = Q_i R_i$$

   using `HouseholderMR` and `GivensForRows` [Hav12].

2. Compute bidiagonal matrices B_i starting from R_i,
   using Householder reflections (R-Bidiagonalisation), yielding

   $$R_i = U_{2,i} B_i V_{2,i}^H$$

3. Starting with matrix B_i, compute diagonal matrices Σ_i using Givens rotations (Golub-Kahan SVD step), yielding the factorisation

   $$B_i = U_{3,i} \Sigma_i V_{3,i}^H$$

4. Compute the SVD of matrix $A_{(i)}$, setting

   $$A_{(i)} = Q_i U_{2,i} U_{3,i} \Sigma_i (V_{3,i} V_{2,i})^H = U_i \Sigma_i V_i^T$$

Each of these steps can be parallelised using the MapReduce approach allowing a reasonable (not a trivial) distributed computation of Higher-Order SVD. Algorithm 10.5 is a formal overview of Havenstein's approach to compute Higher-Order SVD based on MapReduce. Details on the implementation of ROC in line 8, R-Bidiagonalization in lines 10 and 14 as well as the Golub-Kahan step in lines 11 and 15 are given in [Hav12].

[1]with Hestenes' method, R-SVD, or any other (parallel) matrix SVD algorithm

```
Input          dth-order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots \times I_d}$,
               pruning tuple $(m_1, \ldots, m_d) \in [1, I_1] \times \ldots \times [1, I_d]$
Output         dth-order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots \times I_d}$
               as the best rank-$(m_1, ..., m_d)$ approximation to $\mathcal{A}$

Unfolding      for $i = 1, \ldots, d$
                 Compute the unfolding $A_{(i)}$ of $\mathcal{A}$
                 if $I_i < \prod_{k=1,k\neq i}^{N} I_k$
                   $A_{(i)} \leftarrow A_{(i)}^T$
                 end
               end
Matrix SVD     for $i = 1, \ldots, d$
                 $[U_{i,1}, R_i] \leftarrow A_{(i)}$                 ROC
                 if $I_i < \prod_{k=1,k\neq i}^{d} I_k$
                   $[U_{i,2}, B_i, V_{i,2}] \leftarrow R_i$          R-Bidiagonalization, V explicit
                   $[U_{i,3}, \Sigma_i, V_{i,3}] \leftarrow B_i$     Golub-Kahan step, V explicit
                   $U_i \leftarrow (V_{i,3} V_{i,2})^T$
                 else
                   $[U_{i,2}, B_i, V_{i,2}] \leftarrow R_i$          R-Bidiagonalization, U explicit
                   $[U_{i,3}, \Sigma_i, V_{i,3}] \leftarrow B_i$     Golub-Kahan step, U explicit
                   $U_i \leftarrow U_{i,1} U_{i,2} U_{i,3}$
                 end
               end
Pruning        for $i = 1, \ldots, d$
                 $W_i := [u_{i,1}, \ldots, u_{i,m_i}]$ where $u_{i,j}$ are column vectors of $U_i$
               end
Core Tensor    Compute $\mathcal{S} := \mathcal{A} \times_1 W_1^{\mathsf{T}} \times_2 W_2^{\mathsf{T}} \ldots \times_d W_d^{\mathsf{T}}$
Approximation  Compute $\widehat{\mathcal{A}} := \mathcal{S} \times_1 W_1 \times_2 W_2 \ldots \times_d W_d$
```

Algorithm 10.5: Higher-Order SVD based on R-SVD adapted from [Hav12].

10.4 Crowdsourcing MapReduce: JSMapReduce

JSMapReduce is an implementation of MapReduce [DG08], which exploits the computing power available in the computers of the users of a Web platform such as the ARTigo platform by giving tasks to the JavaScript engines of the users' Web browsers. JSMapReduce in combination with Higher-Order SVD based on QR decomposition (cf. Section 10.3) allows a distributed computation of Higher-Order LSA (cf. Chapter 8). Notably, JSMapReduce allows to combine automated computation and crowdsourcing as promised in Chapter 1. Playing games on the ARTigo platform (with a Web browser) adds human computation on top.

This section describes the implementation of JSMapReduce exploiting HTML 5 features, the heuristics it uses for distributing tasks to workers, and reports on an experimental evaluation of JSMapReduce. This section is based on a publication [LWB13] of which the author of this thesis is coauthor.

10.4.1 A Brief Introduction into MapReduce

MapReduce [DG08] has evolved from a proprietary Google programming model for data parallel computation of PageRank [PBMW99] to a popular approach for solving data parallelisable problems with a cluster or grid of computers. In the meantime, the applications of MapReduce range from machine learning [CKL$^+$07] over evaluating queries to databases like CouchDB [ALS10]. Implementations of MapReduce have reached productive state, among others Apache Hadoop [Whi09] or Skynet [JIQ$^+$11].

Many Web platforms are intended for an interactive use: They deliver data or services upon request of users browsing through texts such as catalogs, documentation, or data summaries. Such a function leaves much computing power at the user's side unused because of the latency inherent to human reactions. JSMapReduce aims at tapping in this computing power by shifting load to the idle processors of website visitors. In the following, potential application areas for JSMapReduce are looked at in more detail.

E-Learning

Most E-Learning platforms would be ideal candidates for deploying JSMapReduce because learning requires time inducing high latencies to user reactions. Furthermore learning analytics, a key feature of today's E-Learning platforms, are mostly well amenable to data parallelism. Learning analytics can be used among others to adapt teaching material to learners' profiles. This can be done, e.g., by using a clustering algorithm like K-Means [M$^+$67]. [RVS11] describes a MapReduce implementation of K-Means.

Social Network

Social network analysis heavily relies on linear algebraic computation like eigenvector computations that are perfect candidates for data parallel computation using MapReduce [2]. [TSWY09] for example describes an application of MapReduce to determine social influences in a social network.

E-Commerce

E-Commerce often exploits the "long tail" of a market, which means, benefits from goods sold in small quantities which in turn requires to adapt the offers to the clients. Recommender systems used for this purpose by E-Commerce platforms are often based on algorithms amenable to data parallelism and MapReduce (see for example [LHX10]) and therefore to JSMapReduce.

GWAP platforms

Games with a Purpose [VAGK$^+$06, VAD08] are the initial motivation for JSMapReduce. Several GWAPs are used for field research collecting data about artworks with ARTigo[3] and about the usage of the Italian language with Metropolitalia[4]. Data collected with GWAPs is processed with MapReduce for building a semantic search index.

Common Characteristics of the Use Cases

Human activity in the foreground and machine computing in the background are the key characteristics of the afore mentioned use cases. More generally, these characteristics are met on websites with high *screen time* and *low user interaction.* Many such websites rely on MapReduce. Most interestingly, in using JSMapReduce the computational power provided for running MapReduce scales with the number of users.

10.4.2 Client-side Technical Requirements

The barriers to participate in calculating JSMapReduce jobs are low. Basic requirements are an Internet connection and a Web browser implementing the *Web Worker* [Hic] API of HTML 5 [BLN$^+$12] enabling multi-threaded JavaScript. The Web Worker API allows processing MapReduce jobs in the background without affecting the functionality of a website in the foreground.

[2]In fact, eigenvector computations were the very reason why Google developed MapReduce.
[3]`http://www.artigo.org`
[4]`http://www.metropolitalia.org`

For ensuring high quality of service, users should have long screen times, that is, staying on a website as long as possible. Additional software installations or network configurations (e.g., firewall settings) are not necessary.

10.4.3 Related Work

MapReduce [DG08] is a programming model for the concurrent processing of large data sets on clusters or grids. MapReduce has been implemented in several languages. The most famous implementations are the internal implementation of Google and its open source adaption Hadoop [Whi09]. MapReduce applications are working on the basis of key/value pairs. The user of MapReduce provides a Map and a Reduce function. During a first step, the application of the Map function on provided data yields intermediate data being input to the Reduce step. The Reduce step calculates the final result. The Runtime System of MapReduce is responsible for parallelizing, error handling, scheduling, and load balancing. One of the strengths of MapReduce is hiding the complexity of the execution from the user.

BOINC and similar systems also orchestrate PC grids and in particular allow solving MapReduce jobs [CSD11]. However, these systems are not Web browser-based.

To the best of the authors' knowledge, there has been no approach using a grid of Web browsers for MapReduce, yet.

A different system also called JSMapreduce[5] aims at testing MapReduce in one single Web browser. Productive calculations are not performed on the Web browser.

10.4.4 JSMapReduce

JSMapReduce adapts MapReduce to the Web context. The *User* submits MapReduce jobs via a Web browser to the *Runtime System*, which manages the processing on many *Workers*. Each worker is a Web browser with a JavaScript engine, that is for example displaying a website of the mentioned application areas in Section 10.4.1.

Challenges in the Web Context

In contrast to the controlled cluster or grid environments for traditional MapReduce calculations, the Web as one factor and human beings as the other factor cause challenges for processing MapReduce jobs. In the Web context HTTP 1.1 is the only widespread choice for client server communication for HTML and JavaScript usage. The first challenge is, that only clients can establish connections to the server for sending and receiving messages. As a consequence, the Runtime System of JSMapReduce can act as a managing instance, only if a client asks for new instructions. Therefore, decisions on distributing MapReduce tasks (i.e. a sub-problem of a MapReduce job) to Web browsers are based on uncertainty. Another challenge concerning HTTP is that the protocol is stateless. Hence, each request of a worker is treated independently and requires extra management effort.

Uncertainty is also an issue concerning the perspectives of an efficient and exhaustive processing of MapReduce jobs. First of all the diversity of workers regarding computational power and the quality of the JavaScript engine affect the efficiency. Secondly the quality of the transmission rates between workers and the runtime environment for exchanging jobs and results has a high impact on the performance of a worker. In fact both, the speed of calculations as well as the quality of the data transfer, are crucial for completing MapReduce jobs successfully. Finally, human beings are a source of uncertainty. As mentioned above, JSMapReduce runs successfully if deployed in an environment where the screen time is long enough to complete the job because leaving a website with JSMapReduce support makes the run time engine wait for results it will never receive. Running a grid of Web browsers with JSMapReduce means that one has a dynamic number of workers that are available to perform tasks, while the whole MapReduce job may have already started. This means that one has to handle new workers as well as vanishing workers.

[5]`http://www.jsmapreduce.com`

Fraud (intended or not) is an additional challenge in the JSMapReduce approach. Since the clients' environment is not under control (like in a traditional cluster or grid environments), calculated results need to be verified.

These above-mentioned kinds of uncertainty require several strategies described below. We identified profiles of workers that can be detected in an early state of processing. These profiles allow a significant optimization during the execution of MapReduce calculations.

Architecture of JSMapReduce

Starting a JSMapReduce job means submitting input data and functions to the *JSMapReduce Client*, that interacts with the *Runtime System* being responsible for coordinating the *JSMapReduce Workers.* Solutions of Map or Reduce jobs calculated by the *JSMapReduce Workers* are aggregated by the *Runtime System* and presented to the user via the *JSMapReduce Client.*

JSMapReduce Client The JSMapReduce client is the interface for users who want to process MapReduce jobs. Obviously, the client allows submitting data as well as the Map and the Reduce function. The client manages the communication with the Runtime System via a simple protocol. To compensate the stateless HTTP requests, the client can be set to states like listening.

Runtime System The Runtime System, consisting of the three components *Bridge*, *Master* and *Scheduler*, is responsible for creating, distributing, monitoring and bundling tasks.

The **Bridge** is a standalone socket-server receiving commands and MapReduce jobs from users. The bridge reports on the status of processing, initiates the distribution of data and actuates preprocessing steps in the Master. The **Master** is the core component of the JSMapReduce framework and communicates with the remaining components to manage the processing of a MapReduce job. The Master receives input data from the bridge and preprocesses them. Based on the preprocessing steps, the master creates tasks (sub-problems) managed in a database. A task can be processed immediately or delayed. The master is also responsible for the registration of new workers. To prevent management overhead and abuse, only workers fulfilling minimal requirements are accepted Additionally, the master manages intermediate results and aggregates partial results. The **Scheduler** supervises each single worker by detecting timeouts (e.g., if a user left a website) and delivering tasks available in the database. The scheduler can assign tasks redundantly to ensure the quality of service.

JSMapReduce Worker If a Web browser is online on a website with JSMapReduce support, it is called a JSMapReduce worker. Visiting a JSMapReduce website executes JavaScript code in the background via the HTML 5 Web Worker API and connects to the Runtime System. Due to the limited stateless HTTP protocol the server-push technique (COMET [Rus06]) is used to keep the connection to the Runtime System alive for further communication. As soon as tasks are available, the worker receives them. This approach grants enough freedom to the runtime engine to control the amount of processing on the worker without busy waiting. Before being sent to the workers, original input data is partitioned into *chunks.* The size of a chunk varies according to the identified profile of the worker. For example, reliable workers with good Internet connections and fast processors will get the largest chunks. Chunks are compressed for the transfer.

The strategy for scheduling workers is hybrid. First choice are workers with high quality rating, afterward quantity replaces quality, and finally the Runtime System itself is the last standing worker to process the data chunks until new workers register, when visiting a JSMapReduce supported website.

Evaluation

For the sake of evaluating JSMapReduce, the classic video game Snake runs in the foreground. The implementation is based on the new HTML 5 canvas element. When playing Snake, users collect items by controlling a snake with the keyboard. In the background JSMapReduce tasks are processed without influencing the highly interactive game.

The 240 website visitors during the evaluation processed more than 70.000 tasks with a failure rate of 0.25 %. Each player processed 449 MB (uncompressed) on average, playing 2:11 minutes. The data chunks were compressed to a tenth of their original sizes for transfer. The average transfer rate was 1 GB in 113 seconds. The average processing time of the chunks was 26 seconds/1 GB, while the users stayed for 131 seconds on the website on average. Hence, more than 1 GB of (uncompressed) data was processed during one game session.

As a positive result of a subsequent online-survey, no user reported effects of the massive calculations in the background on the game application in the foreground. 21 % complained about occasional glitches, the others felt confident. Concerning allowing additional calculations in the background only 15 % of the users raised doubts but none rejected the idea.

In the evaluation the server was a bottleneck and caused overheads of 98 seconds per GB. Optimizing the server is subject to further research on JSMapReduce. The tests showed that the best performance could be obtained with 20 concurrent workers.

A bottleneck that cannot be influenced by software optimization of JSMapReduce (in contrast to controlled grid environments) is the transmission rate between the Runtime System and workers. This fact makes JSMapReduce suitable for MapReduce jobs that draw on exploiting the CPU rather than relying on massive data exchange.

The evaluation proved that the quality of the workers in terms of Internet connection and processor was of significant importance in comparison to the quantity of workers. Strategies, based on measurements of response time and time requirements (see Figures 10.4 and 10.5), to preselect workers of high quality are discussed in [Lan12].

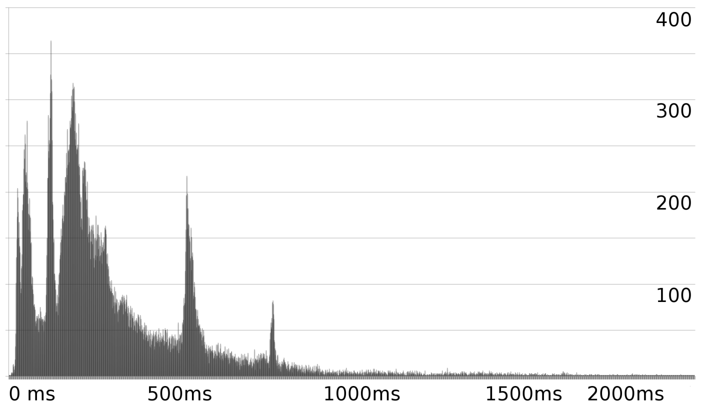


Figure 10.4: The histogram shows that the workers' response time was on average less than half a second.

A demonstration of JSMapReduce based on the classic example "log file analysis" is available as a screencast on `http://www.pms.ifi.lmu.de/publikationen/#PMS-FB-2013-1`. Sample log files are generated that should be transformed to monthly statistics about website visitors. A basic log file contains 1 million entries occupying 150 MB space. The data is provided via a database and the Map and the Reduce functions for this demonstration case are also provided.

The demonstration case starts with declaring in the JSMapReduce Client how to portion the 1 million entries into smaller data chunks for the workers. Data chunks of 2000 lines each are chosen.

The progress of the whole job is visualized by dots arranged in a box. One dot stands for the status of one data chunk. White means *untreated*, blue means *in progress*, green means *completed*, and red means *error*.

The number of concurrent workers in progress can be determined by the number of blue dots. At first we demonstrate a JSMapReduce job executed with only one worker rendering a session of

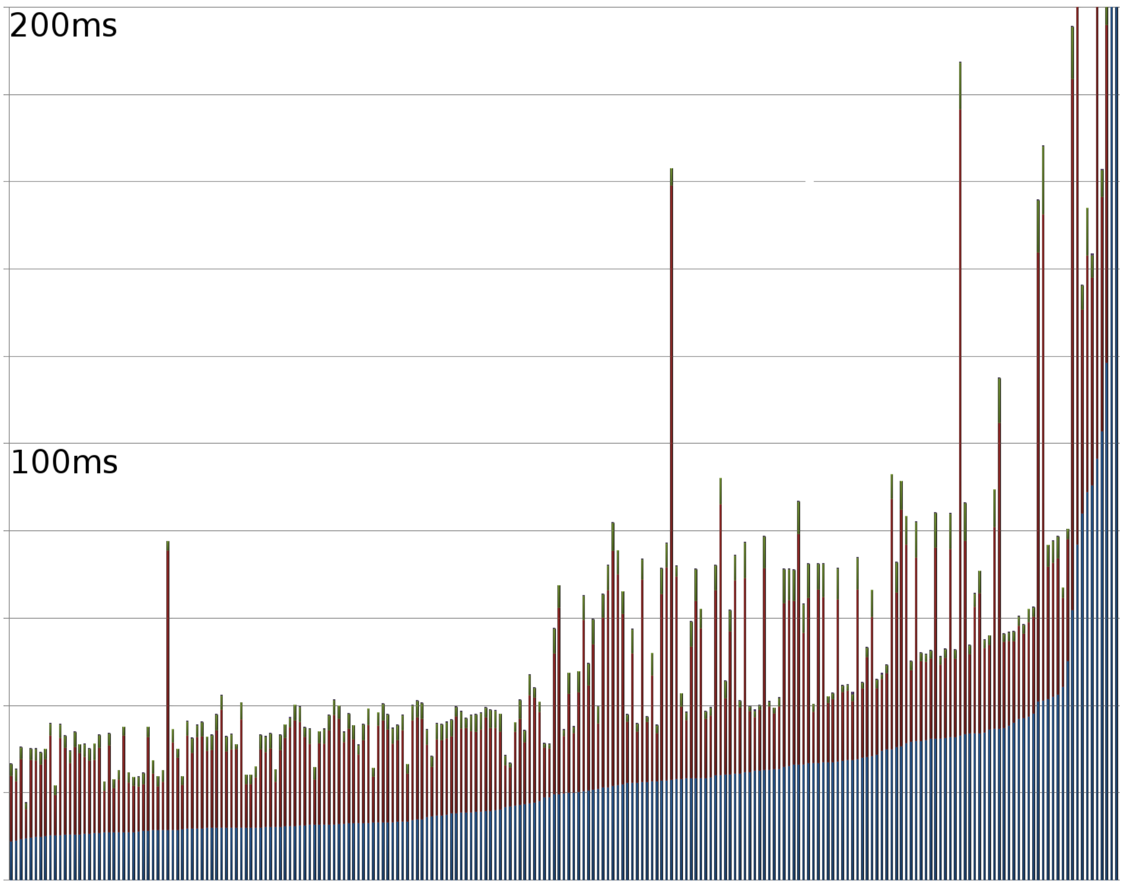


Figure 10.5: Time requirements of the steps Init (blue), Map (red), Group (green), and Reduce (purple).

the video game Snake in the foreground. Obviously this highly interactive game runs smoothly, while a data chunk is processed in the background at the same time, although multiple snake updates need to be rendered every second.

As a second step processing data chunks with multiple workers is demonstrated as expected in MapReduce frameworks. Therefore several browser games are started at the same time registering themselves as workers on the Runtime System. When running the MapReduce job, several blue dots are displayed at the same time, corresponding to the active workers. As expected all Snake instances run smoothly.

10.4.5 Conclusion and Future Work

This section reports on an implementation of MapReduce in JavaScript using the Web browsers of common website visitors as workers. JSMapReduce was originally conceived for a distributed calculation of an index for the semantic search engine of the GWAP platform ARTigo.

The implementation benefits from the Web Worker API introduced in HTML 5 allowing multi-threaded processing of JavaScript programs. We proved that intense processing and transmission of data in the background has no perceptible effect on the Web browser interactions in the foreground. This makes JSMapReduce suitable for application areas needing cheap and scalable computational power with a growing number of users on their website. The longer a user stays on the website, the more MapReduce problems can be calculated. Despite having evaluated the approach with a prototypical implementation, the results are promising in comparison to established

MapReduce implementations such as Hadoop. Subtracting the overhead of the Web transmissions, the performance is similar to the Hadoop implementation of MapReduce as described in [Lan12].

Further work on implementations of the JSMapReduce approach will benefit from the given heuristics to ensure quality of service on the one hand, mainly by choosing reliable workers in an early stage of processing, and from coping with a dynamic number of workers on the other hand.

The JSMapReduce approach allows bundling human thinking and computational power, therefore upgrading the crowdsourcing approach to a higher stage.

The first JSMapReduce prototype has potential for many optimizations, amongst others analyzing OS performance, error rates, load balancing, intra-worker performance, performance variations, client-side storage, and overhead reduction to mention some of them.

Part IV

Perspectives: Further Analysing the ARTigo Data

The previous parts of this thesis focus on *Building a Semantic Search Engine with Games and Crowdsourcing*. Part I is an introduction to human computation and search engines. Part II aims at acquiring meta-data being leveraged in Part III for semantic search.

The millions of data records as collected by the gaming ecosystem of the ARTigo platform are something special. Mainly for two reasons these data make analysis beyond image retrieval by a semantic search engine attractive. First, a lot of distinguishable individuals contributed to building the collection of data. Second, the data domain is restricted to historic art images. This combination presents the prospect of unconventional art history insights apart from the academic tradition of humanities. Undiscovered patterns or relationships between images originating from different parts of the earth could be unveiled. The perception of art images could be studied looking at the chronology of entered tags. Since historic art images may reflect the life style of former times, the ARTigo meta-data could be used to answer questions like: when was a characteristic fashion or mood depicted for the first time and how did it spread globally? Many more of such research questions are meant to be triggered in this part of the thesis. Results based on an analysis of the ARTigo data starting from (but not restricted to) a linguistic perspective are published in the doctoral thesis of Elena Levushkina [Lev14].

Chapter 11 of this part reports on basic characteristics of the ARTigo data as starting point for further investigations. Chapter 12 describes perspectives for further work on basis of the ARTigo data.

Chapter 11

Determining Characteristics of the ARTigo Dataset

This chapter introduces the data as collected by the ARTigo platform from the start of the project in December 2007 until July 2013. Since then, 164.745 users have contributed to the dataset, and 18.607 of those have registered to the platform. In the meanwhile, the dataset covers 6.722.045 taggings. Most of the taggings 6.066.007 originate from the basic ARTigo game. Some of the taggings are of higher value: 1.220.973 taggings were validated on the same image, that is, two users entered the same tag (eventually in the same game) for one image. The data give some insights into development and life of a social media platform such as user behaviour or the influence of advertising.

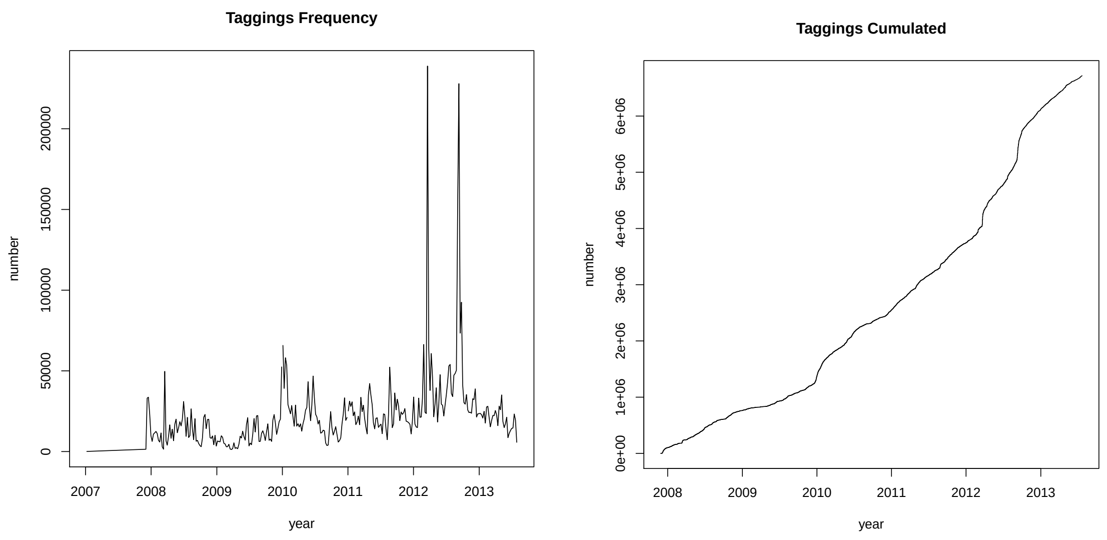


Figure 11.1: Taggings per week.

Figure 11.1 describes the submissions of tags per week. The graph on the left side shows the frequency, accomplished by the cumulated frequencies of entered tags on the right side. Both plots represent the same information. Looking at the frequencies, the average level of submissions grows slightly until the end of 2012. In 2013 the submissions are shrinking. Most notably, several peaks (periods with a high tag submission rate) occur in the frequency plot. The first higher one dates back to spring 2012 when a famous German online magazine Spiegel Online[1] published an

[1]`http://www.spiegel.de/`

article about ARTigo with a link to the platform. The second peak is the result of a kind of viral marketing via StumbleUpon[2]. StumbleUpon recommends websites based on experiences of its clients. Lower peaks are often caused by competitions organized by the project. The response to articles in the local newspaper could not be recognized by a significantly higher tagging rate. Even a presentation on TV (3sat nano) had no significant influence to the tagging rate.

The slope of the graph on the right side of Figure 11.1 demonstrates the growth rate of taggings. Obviously the highest slopes correspond to the striking peaks in the left graph. Although the growth is varying, it has a linear tendency.

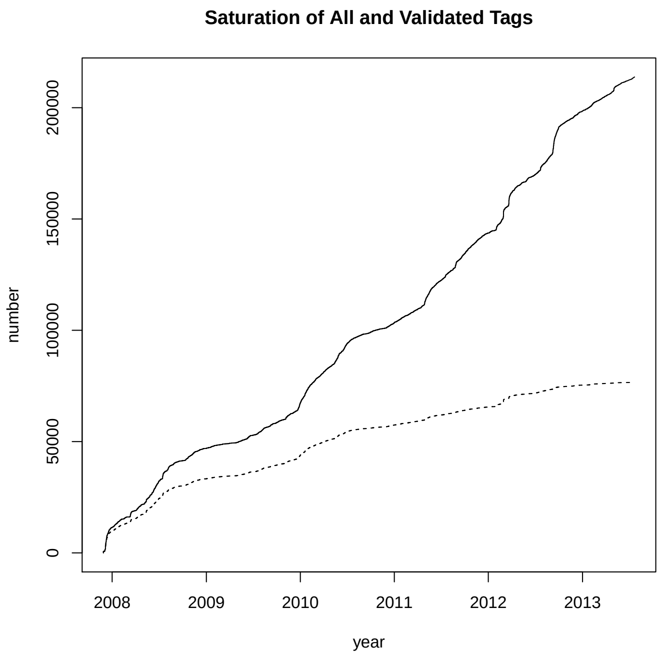


Figure 11.2: Tags (solid line) and validated tags (dashed line).

Whenever a user enters a tag on the platform, references to the tag, the user, and the tagged object are memorized. This triple is called tagging. Whenever a tag was entered for the first time on the ARTigo platform, the tag is added to the database and referenced afterwards by the tagging triple. In other words, a tag is stored only once in the database. Hence, a saturation of tags (not of taggings) can be expected due to the limitation of words/tags in a spoken language.

Surprisingly, Figure 11.2 shows a rather linear growth of tags (solid line). A reason for the slow saturation may be misspellings of tags. The dashed line shows the growth of validated tags, i.e. tags being entered at least twice for an image. In ARTigo such tags are used for searching to ensure the quality and to filter spam tags. In contrast to the unvalidated tags, a saturation effect can be observed because the graph is flattening out over the years. Misspellings are only represented by the dashed line if entered repeatedly.

[2]`http://www.stumbleupon.com/`

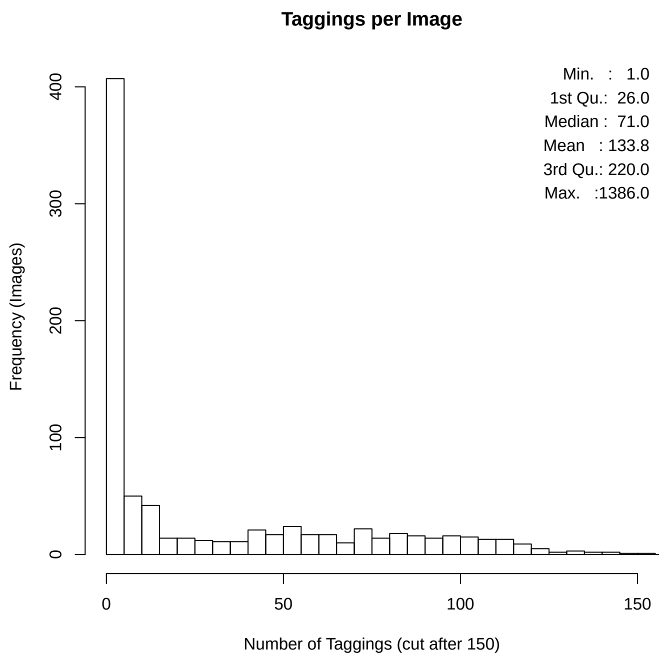


Figure 11.3: Taggings per Image.

Figure 11.3 reports on the frequency distribution of taggings per image. An image can have assigned taggings in a range from 1 to 1.386. (Note, that the histogram is limited to a maximum of 150 on the x-axis for a better visualization of the left part with weaker tagged images.) Only images without easement of the owner have no tagging at all. About 400 images out of 46.176 (less than 1%) have less than 10 taggings. In average an image has 133.8 taggings. One of the basic image selection strategies for games of the platform is gaining tags for rarely tagged images. The goal of this strategy is reaching a uniform frequency distribution. Obviously, adding new images to the platform means increasing the left-most (largest) group in the histogram.

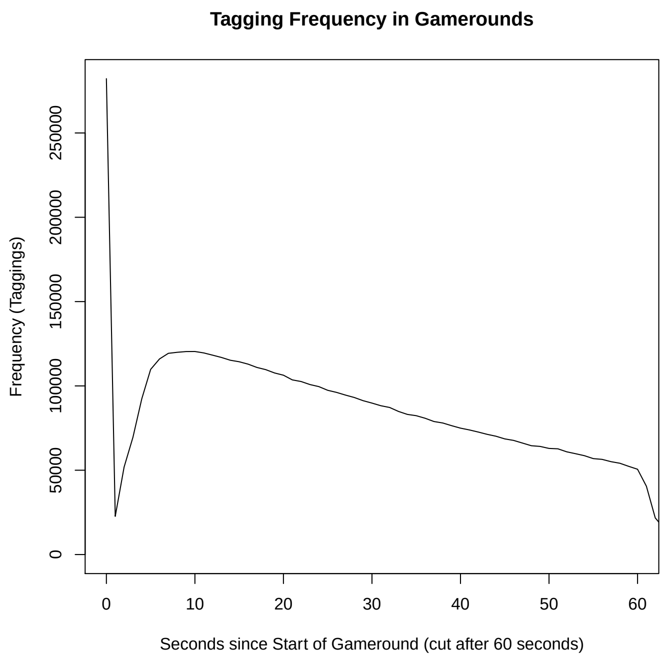


Figure 11.4: Tagging Frequency per Gameround.

The ARTigo-Game asks players to enter tags within a period of 60 seconds in a game round. Figure 11.4 shows the frequency distribution of taggings as entered by users during the whole lifetime of the platform. Generally, the number of taggings is linearly increasing until a maximum after 10 seconds. Afterwards the number of taggings is slightly decreasing.

There are two anomalies at the beginning and at the end of a game round. First, there is a peak in the beginning of the plot. This peak originates from crawlers of search engines calling URLs that submit tags directly via CGI parameters while opening the game round. Hence, these actions are cumulated at the beginning and result in a peak. Second, there is a decrease of tag submissions after 60 seconds. Since a game round lasts 60 seconds a cut would match the expectations. However, the phenomenon originates from network latencies between tag submission at the user's browser and the database commit.

Chapter 12

Perspectives for Future Work

This chapter addresses perspectives for future work concerning data analysis. Please refer to Chapter 12 for perspectives of future work on gaming aspects, which is anticipated for ease of readability.

12.1 Historic Trend Analysis

For historians art is an important portal to the past. Artworks complement written sources such as literature by very explicit insights into history in the absence of photography. However an automated analysis of artworks is significantly more complex compared to the automated analysis of texts as already mentioned in Part I. The ARTigo data bridge the gap between graphic and text, serving as textual interface to the art perspective for automatic analyses.

The following Figures 12.1 and 12.2 are adopted from Clemens Schefels (Institute for Informatics, LMU, Munich).

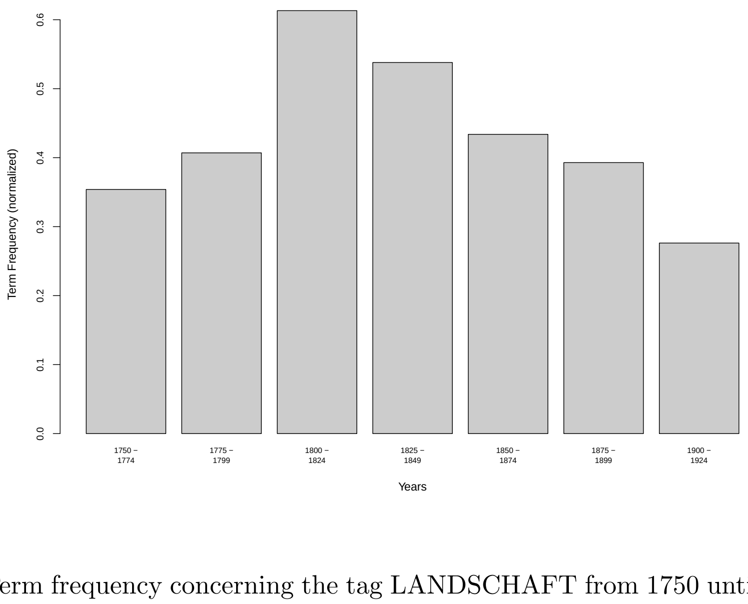


Figure 12.1: Term frequency concerning the tag LANDSCHAFT from 1750 until 1924 by Schefels.

As the date of creation is known for most art works of the ARTigo data, historic trends can be easily detected by exploring tagging frequencies over time. For example, some tags may arise after a certain date for the first time reflecting historic circumstances. Figure 12.1 visualizes the normalized[1] term frequency of the tag LANDSCHAFT (German for LANDSCAPE). Apparently, landscape art became significantly popular at the beginning of the 19th century. Afterwards the interest was declining.

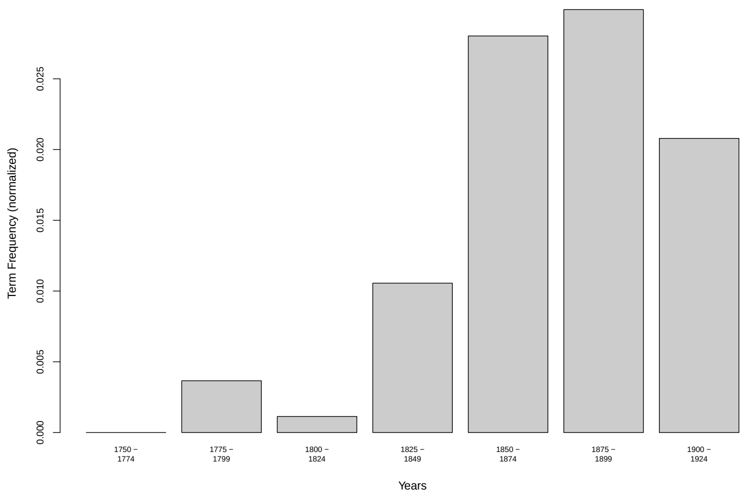


Figure 12.2: Term frequency concerning the tag SCHNURRBART 1750–1924 by Schefels.

Figure 12.2 illustrates the advent of growing a MOUSTACHE identified by the German tag SCHNURRBART. According to the ARTigo data this fashion was most popular from 1850 until 1900. Assuming that the ARTigo artworks are statistically representative and historic painters depicted the reality, the moustache trend can by detected rather easily by the ARTigo tags.

Tag-based trend analyses allow an perspective on art over centuries. This is rather untypical for Art Historians because specialists of a certain era tend to outweigh generalists. Tags may fill the gap between specialists and may lead to joined research results.

12.2 Detecting Art Movements

Art movements like Renaissance or Impressionism are abstract concepts to group artworks with similar features together. Assigning an artwork to an art movement can be tricky because suspicious tags may be absent or the classification may rely on a special combination of tags as collected, e.g., by the game Combino (cf. Section 5.3). In addition recurring art fashions can make differentiations difficult. Therefore, an automatic and reliable detection of art movements for artworks is desirable.

A classifier for the artworks on the ARTigo platform can be trained based on the corresponding taggings. Unlike the date of creation for artworks, art movements are not part of the given metadata in ARTigo. Therefore the taggings of the ARTigo players are the basis for training a classifier. Since players have heterogeneous backgrounds, artworks can unintentionally be tagged with wrong art movements. This phenomenon occurs especially at the transition from one art movement to another, if an artwork cannot be clearly assigned to one of two art movements. A simple approach[2] for training a classifier based on ARTigo data is choosing the art movement that was most often assigned by the ARTigo players out of a list of *known* art movements. A classifier trained on art movements could be used for selecting images for ARTigo games. For example, the diversification of Baroque paintings could be the goal of playing Karido (cf. Section 5.2).

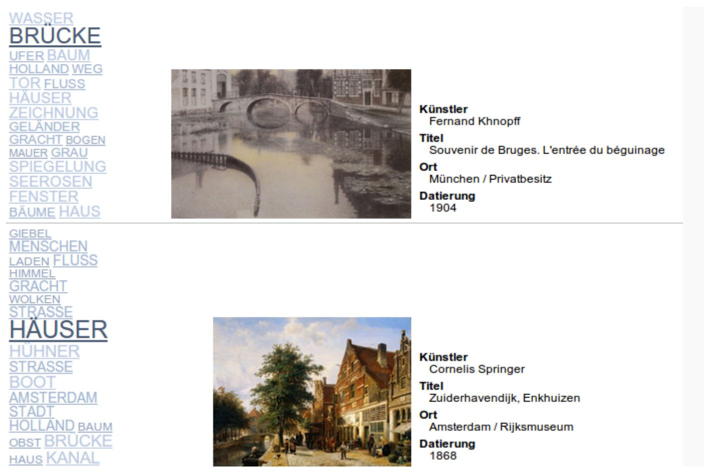
[1]Normalization via term frequency-inverse document frequency (tf-idf)

[2]A hierarchical approach is proposed in Section 12.3.

Cluster analysis is a technique to group objects with similar features together. In contrast to classification, the groups are apriori *unknown*. For example a cluster analysis on the ARTigo data may result in clusters that match with art movements. Clusters that do not match with art movements may be of even higher interest because the corresponding artworks could belong to so far unknown art movements. Furthermore, clusters may be hierarchically arranged such that art movements could be unexpectedly linked together. Many of such clusters will be outliers caused by misleading taggings by players. Detecting the remaining clusters may be an interesting exercise for Art Historians approaching Art History from a new perspective.

12.3 Decision Tree for Browsing

The ARTigo platform offers means for retrieving artworks: Validated taggings are the source for a full-text search on the ARTigo artworks. This feature is useful for concrete queries. However for discovering artworks and getting a feel of the collection of artworks, means for browsing with proposed links are welcome. Browsing allows to explore the collection without explicit knowledge about the content.



Künstler
Fernand Khnopff
Titel
Souvenir de Bruges. L'entrée du béguinage
Ort
München / Privatbesitz
Datierung
1904

Künstler
Cornelis Springer
Titel
Zuiderhavendijk, Enkhuizen
Ort
Amsterdam / Rijksmuseum
Datierung
1868

Figure 12.3: Tag clouds for art works on the ARTigo platform.

The ARTigo platform offers such links arranged in tag clouds as shown in Figure 12.3. Clicking on one of the links triggers a query for the corresponding tag. These tag clouds could be integrated into a hierarchy.

A classification hierarchy could enrich the ARTigo platform with educational games, which teach how to recognize the art movement of an artwork. For example, the earlier a player detects the right art movement of an artwork out of a list of candidates, the better the player could be rewarded. Entering tags describing the artwork could stepwise remove wrong art movement candidates from the list. As consequence, a smaller amount of point would be the prize for the correct answer.

12.4 Context-based Translations of Tags

The games on the ARTigo platform are offered in several languages. Most of the taggings were collected in Germany and, therefore, most of the tags are in German. Obviously, a widespread tagging in all offered languages is desired to support search in as many languages as possible. One approach based on human computation is described in Section 5.4 with the game Eligo. Another approach for future work is exploiting the co-occurrence of tags for one artwork in different languages. As already mentioned in the context of Eligo, translations are strongly dependent on the context. Therefore a simple mapping is not sufficient for extracting ready-to-use translation rules.

In the context of this thesis, Higher-Order LSA (cf. Section 7.3) is an obvious candidate for translations of tags because it can be expected that the semantics are the same for tags describing an image in different languages. The design for such an analysis would base on a 3rd-order tensor modelling tags, documents, and users with a slight modification. The users are aggregated according to their spoken language[3]. Hence, the number of 3rd-mode slices of the tensor corresponds with the number of languages and not with the number of users.

Noh et al. [NPY+09] exploit tagging co-occurrences on images for language-independent image retrieval. The approach analyses similarity of sub-networks. An analysis of a folksonomy based on Higher-Order LSA could provide means for context-based translation of tags in addition to language-independent image retrieval.

[3]The ARTigo allows to modify the language for each user and takes the language of the Web-browser as default.

Bibliography

[ALS10] J Chris Anderson, Jan Lehnardt, and Noah Slater. *CouchDB: the Definitive Guide.* O'Reilly, 2010.

[Arn00] Ben Arnold. An Investigation Into Using Singular Value Decomposition as a Method of Image Compression. *University of Canterbury Department of Mathematics and Statistics*, 9 2000.

[AS$^+$94] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast Algorithms for Mining Association Rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, volume 1215, pages 487–499, 1994.

[BB99] Michael W Berry and Murray Browne. *Understanding Seach Engines: Mathematical Modelling and Text Retrieval*, volume 8. Siam, 1999.

[BCM09] Paul N Bennett, David Maxwell Chickering, and Anton Mityagin. Picture This: Preferences for Image Search. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 25–26. ACM, 2009.

[BKW11] François Bry, Fabian Kneißl, and Christoph Wieser. Field Research for Humanities with Social Media: Crowdsourcing and Algorithmic Data Analysis. In *Proc. 4th Workshop Digitale Soziale Netze*, volume 41, 2011.

[BL85] R. Brent and F. Luk. The Solution of Singular-Value and Symmetric Eigenvalue Problems on Multiprocessor Arrays. *SIAM Journal on Scientific and Statistical Computing*, 6:69–84, 1985.

[BLHL$^+$01] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The Semantic Web. *Scientific american*, 284(5):28–37, 2001.

[BLN$^+$12] Robin Berjon, Travis Leithead, Erika Doyle Navara, Edward O'Connor, and Silvia Pfeiffer. HTML5. *World Wide Web Consortium (W3C)*, 2012.

[BMPS06] Michael W Berry, Dani Mezher, Bernard Philippe, and Ahmed Sameh. Parallel Algorithms for the Singular Value Decomposition. *STATISTICS TEXTBOOKS AND MONOGRAPHS*, 184:117, 2006.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[BSWD03] Adiel Ben-Shalom, Michael Werman, and Shlomo Dubnov. Improved Low Bit-Rate Audio Compression Using Reduced Rank ICA Instead of Psychoacoustic Modeling. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings of ICASSP'03*, volume 5, page 461. IEEE, 2003.

[BW12a] François Bry and Christoph Wieser. Squaring and Scripting the ESP Game. In *Proceedings of the 4th Human Computation Workshop (HC)*, 2012.

[BW12b] François Bry and Christoph Wieser. Squaring and Scripting the ESP Game: Trimming a GWAP to Deep Semantics. In *Serious Games Development and Applications*, pages 183–192. Springer, 2012.

[CC89] H. Y. J. Chuang and L. Chen. Efficient Computation of the Singular Value Decomposition on Cube Connected SIMD Machine. In *Supercomputing '89*, pages 276–282, New York, NY, USA, 1989. ACM.

[CKL$^+$07] Cheng Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Y Ng, and Kunle Olukotun. Map-Reduce for Machine Learning on Multicore. *Advances in neural information processing systems*, 19:281, 2007.

[CLE$^+$13] Lydia B Chilton, Greg Little, Darren Edge, Daniel S Weld, and James A Landay. Cascade: Crowdsourcing Taxonomy Creation. In *Proceedings of the 2013 ACM annual conference on Human factors in computing systems*, pages 1999–2008. ACM, 2013.

[CSD11] Fernando Costa, Luis Silva, and Michael Dahlin. Volunteer Cloud Computing: Mapreduce Over the Internet. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 1855–1862. IEEE, 2011.

[DDF$^+$90] Scott Deerwester, Susan T. Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by Latent Semantic Analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[DDH01] Petros Drineas, Eleni Drinea, and Patrick S. Huggins. An Experimental Evaluation of a Monte-Carlo Algorithm for Singular Value Decomposition. In *Panhellenic Conference on Informatics*, pages 279–296, 2001.

[Deb04] Sagarmay Deb. *Multimedia Systems and Content-based Image Retrieval*. Idea Group Pub., 2004.

[DG08] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[DLDMV00] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A Multilinear Singular Value Decomposition. *SIAM J. Matrix Anal. Appl*, 21:1253–1278, 2000.

[DM05] Petros Drineas and Michael W. Mahoney. A Randomized Algorithm for a Tensor-Based Generalization of the Singular Value Decomposition. Technical report, Yale University, June 2005.

[DVGV12] Roberto De Virgilio, Francesco Guerra, and Yannis Velegrakis. *Semantic Search Over the Web*. Springer, 2012.

[FK03] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a new computing infrastructure*. Access Online via Elsevier, 2003.

[GK65] Gene Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial & Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965.

[GL96] G. Golub and C. Van Loan. *Matrix Computations*. John Hopkins Press, 3rd edition, 1996.

[Gri05] David Alan Grier. *When Computers Were Human*. Princeton University Press, Princeton, NJ, USA, 2005.

[Gri10] Seth Grimes. Breakthrough analysis: Two + nine types of semantic search, 2010. [Online; accessed 11-October-2013].

[Han10] Erice Hand. Citizen science: People power. *Nature*, 466:685–687, 2010.

[Hav12] Diego Havenstein. Towards MapReduce Algorithms for the Higher Order-Singular Value Decomposition. bachelor thesis, Institute of Computer Science, University of Munich, 2012.

[HCH07] Chien-Ju Ho, Tsung-Hsiang Chang, and Jane Yung-jen Hsu. Photoslap: A Multi-Player Online Game for Semantic Annotation. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 22, page 1359. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[HCL$^+$09] Chien-Ju Ho, Tao-Hsuan Chang, Jong-Chuan Lee, Jane Yung-jen Hsu, and Kuan-Ta Chen. KissKissBan: a Competitive Human Computation Game for Image Annotation. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 11–14. ACM, 2009.

[Hes58] Magnus R. Hestenes. Inversion of Matrices by Biorthogonalization and Related Results. *Journal of the Society for Industrial and Applied Mathematics*, 6(1), March 1958.

[Hic] Ian Hickson. Worker. http://www.w3.org/TR/workers/.

[Hof99] Thomas Hofmann. Probabilistic Latent Semantic Analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.

[How06] Jeff Howe. The Rise of Crowdsourcing. *Wired Magazine*, 14(6), 06 2006.

[JIQ$^+$11] Hai Jin, Shadi Ibrahim, Li Qi, Haijun Cao, Song Wu, and Xuanhua Shi. The Mapreduce Programming Model and Implementations. *Cloud computing: principles and paradigms. Wiley, Hoboken*, pages 373–390, 2011.

[Joh12] Tony John. What is Semantic Search and how it works with Google search, 2012. [Online; accessed 21-October-2013].

[JP08] Shaili Jain and David C Parkes. A Game-theoretic Analysis of Games with a Purpose. In *Internet and Network Economics*, pages 342–350. Springer, 2008.

[KCM04] Graham Klyne, Jeremy J Carroll, and Brian McBride. Resource Description Framework (RDF): Concepts and Abstract Syntax. *W3C recommendation*, 10, 2004.

[KFN95] Helga Kolb, Eduardo Fernandez, and Ralph Nelson. Simple Anatomy of the Retina–Webvision: The Organization of the Retina and Visual System. Technical report, University of Utah Health Sciences Center, Utah, 1995.

[Kin07] Alison King. Scripting Collaborative Learning Processes: A Cognitive Perspective. In *Scripting computer-supported collaborative learning*, pages 13–37. Springer, 2007.

[Knu73] Donald Ervin Knuth. *The Art of Computer Programming.* Addison-Wesley, 1973.

[Kol06] Tamara Gibson Kolda. *Multilinear Operators for Higher-Order Decompositions.* United States. Department of Energy, 2006.

[LA11] Edith Law and Luis von Ahn. Human Computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(3):1–121, 2011.

[Lan11] Nate Lanxon. How the Oxford English Dictionary started out like Wikipedia. http://www.wired.co.uk/news/archive/2011-01/13/the-oxford-english-wiktionary, Jan 2011.

[Lan12] Philipp Langhans. JSMapReduce - Distributed Computing with Web Clients and LAMP. bachelor thesis, Institute of Computer Science, University of Munich, 2012.

[Lev14] Elena Levushkina. *Computerlinguistische Methoden in Community-basierten Anwendungen*. PhD thesis, Center for Information and Language Processing, University of Munich (LMU), 2014.

[LFL98] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[LHX10] Huizhi Liang, Jim Hogan, and Yue Xu. Parallel User Profiling Based on Folksonomy for Large Scaled Recommender Systems : an Implimentation of Cascading MapReduce. In *10th Industrial Conference on Data Mining (ICDM 2010)*, Berlin, December 2010. IEEE.

[LJ11] Stan Z Li and Anil K Jain. *Handbook of Face Recognition*. Springer, 2011.

[LSS$^+$11] Edith Law, Burr Settles, Aaron Snook, Harshit Surana, Luis von Ahn, and Tom Mitchell. Human Computation for Attribute and Attribute Value Acquisition. In *Proceedings of the First Workshop on Fine-Grained Visual Categorization (FGVC)*, 2011.

[LWB13] Philipp Langhans, Christoph Wieser, and François Bry. Crowdsourcing MapReduce: JSMapReduce. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 253–256. International World Wide Web Conferences Steering Committee, 2013.

[LZH09] Chang Liu, Jiliu Zhou, and Kun He. Image Compression Based on Truncated HOSVD. In *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*, pages 1–4. IEEE, 2009.

[M$^+$67] James MacQueen et al. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, pages 281–297. California, USA, 1967.

[Mäk05] Eetu Mäkelä. Survey of Semantic Search Research. In *Proceedings of the Seminar on Knowledge Management on the Semantic Web*, 2005.

[MGP04] Florent Monay and Daniel Gatica-Perez. pLSA-based Image Auto-Annotation: Constraining the Latent Space. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 348–351. ACM, 2004.

[MRT12] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[MVH$^+$04] Deborah L McGuinness, Frank Van Harmelen, et al. OWL Web Ontology Language Overview. *W3C recommendation*, 10(2004-03):10, 2004.

[NPY$^+$09] Tae-Gil Noh, Seong-Bae Park, Hee-Geun Yoon, Sang-Jo Lee, and Se-Young Park. An Automatic Translation of Tags for Multimedia Contents Using Folksonomy Networks. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 492–499. ACM, 2009.

[OD92] ANGELA M O'Donnell and DONALD F Dansereau. Scripted Cooperation in Student Dyads: A Method for Analyzing and Enhancing Academic Learning and Performance. *Interaction in cooperative groups: The theoretical anatomy of group learning*, pages 120–141, 1992. http://goo.gl/YY1bX2.

[O'R05] Tim O'Reilly. What is Web 2.0, 2005.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web, 1999.

[PS$^+$08] Eric Prud'Hommeaux, Andy Seaborne, et al. SPARQL Query Language for RDF. *W3C recommendation*, 15, 2008.

[PTGL13] Ei Pa Pa Pe-Than, Dion Hoe-Lian Goh, and Chei Sian Lee. A Typology of Human Computation Games: An analysis and a review of current games. *Behaviour & Information Technology*, page 1–48, 2013.

[QB11] Alexander J Quinn and Benjamin B Bederson. Human Computation: a Survey and Taxonomy of a Growing Field. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1403–1412. ACM, 2011.

[Rhi66] Joseph Banks Rhine. *Extra-sensory Perception After Sixty Years.* Branden Books, 1966.

[Row02] Michael H Rowe. Trichromatic color vision in primates. *Physiology*, 17(3):93–98, 2002.

[Rus06] Alex Russell. Comet: Low latency data for browsers, 2006.

[RVS11] R Raju, V Vijayalakshmi, and RT Showmya. E-Learning Using Mapreduce. *International Journal on Computer Science and Engineering*, 3(4):1584–1590, 2011.

[RWC$^+$08] Carolyn Rosé, Yi-Chia Wang, Yue Cui, Jaime Arguello, Karsten Stegmann, Armin Weinberger, and Frank Fischer. Computer-Supported Collaborative Learning in Higher Education: Scripts for Argumentative Knowledge Construction in Distributed Groups. *International journal of computer-supported collaborative learning*, 3(3):237–271, 2008.

[Sca08] Jessie Scanlon. Luis von Ahn: The Pioneer of "Human Computation". *Business Week*, 3, November 2008.

[Sch10] Oliver Schnuck. Tag Analysis with Higher-Order SVD. Bachelorarbeit/bachelor thesis, Institute of Computer Science, LMU, Munich, 2010.

[SH08] Katharina Siorpaes and Martin Hepp. Games with a Purpose for the Semantic Web. *Intelligent Systems, IEEE*, 23(3):50–60, 2008.

[SHBL06] Nigel Shadbolt, Wendy Hall, and Tim Berners-Lee. The Semantic Web Revisited. *Intelligent Systems, IEEE*, 21(3):96–101, 2006.

[Sim13] Phil Simon. *Too Big to Ignore: The Business Case for Big Data.* Wiley. com, 2013.

[SNM08] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. Tag Recommendations Based on Tensor Dimensionality Reduction. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 43–50. ACM, 2008.

[SNPM06] Panagiotis Symeonidis, Alexandros Nanopoulos, Apostolos Papadopoulos, and Yannis Manolopoulos. Scalable Collaborative Filtering Based on Latent Semantic Indexing. In *Proc. 21st Assoc. for Advancement of Artificial Intelligence (AAAI) Workshop Intelligent Techniques for Web Personalization (ITWP'06)*, pages 1–9, 2006.

[Sop91] Sophocles. *Oedipus Rex.* Dover Pubn Inc, 1991.

[Ste93] Gilbert W Stewart. On the early history of the singular value decomposition. *SIAM review*, 35(4):551–566, 1993.

[Ste11] Bartholomäus Steinmayr. Designing Image Labeling Games For More Informative Tags. Diplomarbeit/diploma thesis, Institute of Computer Science, LMU, Munich, 2011.

[SWB12] Philipp Shah, Christoph Wieser, and François Bry. Parallel Higher-Order SVD for Tag-Recommendations. In *Proceedings of the IADIS International Conference WWW/Internet 2012, Madrid, Spain (18th–21st October 2012)*, 2012.

[SWKB11] Bartholomäus Steinmayr, Christoph Wieser, Fabian Kneißl, and François Bry. Karido: A GWAP for Telling Artworks Apart. In *Computer Games (CGAMES), 2011 16th International Conference on*, pages 193–200. IEEE, 2011.

[SWY75] Gerard Salton, Anita Wong, and Chung-Shu Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[SZL+05] Jian-Tao Sun, Hua-Jun Zeng, Huan Liu, Yuchang Lu, and Zheng Chen. CubeSVD: a novel approach to personalized Web search. In *WWW '05*, pages 382–390, New York, NY, USA, 2005. ACM Press.

[TSWY09] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social Influence Analysis in Large-Scale Networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 807–816. ACM, 2009.

[Tur50] Alan M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[VA06] Luis Von Ahn. Games with a Purpose. *Computer*, 39(6):92–94, 2006.

[vA13] Luis von Ahn. Augmented intelligence: the Web and human. *Philosophical Transactions of The Royal Society*, 371(1987), Feb 2013.

[VABHL03] Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. CAPTCHA: Using hard AI problems for security. In *Advances in Cryptology—EUROCRYPT 2003*, pages 294–311. Springer, 2003.

[VAD04] Luis Von Ahn and Laura Dabbish. Labeling Images with a Computer Game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM, 2004.

[VAD08] Luis Von Ahn and Laura Dabbish. Designing Games with a Purpose. *Communications of the ACM*, 51(8):58–67, 2008.

[VAGK+06] Luis Von Ahn, Shiry Ginosar, Mihir Kedia, Ruoran Liu, and Manuel Blum. Improving Accessibility of the Web with a Computer Game. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 79–82. ACM, 2006.

[VAKB06] Luis Von Ahn, Mihir Kedia, and Manuel Blum. Verbosity: a Game for Collecting Common-Sense Facts. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 75–78. ACM, 2006.

[VALB06] Luis Von Ahn, Ruoran Liu, and Manuel Blum. Peekaboom: a Game for Locating Objects in Images. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 55–64. ACM, 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(5895):1465–1468, 2008.

[VT02] M. Alex O. Vasilescu and Demetri Terzopoulos. Multilinear Image Analysis for Facial Recognition. In *ICPR (2)*, pages 511–514, 2002.

[WBBL13] Christoph Wieser, François Bry, Alexandre Bérard, and Richard Lagrange. ARTigo: Building an Artwork Search EngineWith Games and Higher-Order Latent Semantic Analysis. In *Proceedings of 1st Workshop on Human Computation and Machine Learning in Games at HComp 2013, Palm Springs, CA, USA (9th November 2013)*, 2013.

[Whi09] T. White. *Hadoop: Definitive Guide*. O'Reilly, 2009.

[Wik13a] Wikipedia. Cosine similarity, 2013. [Online; accessed 22-November-2013].

[Wik13b] Wikipedia. Grid computing, 2013. [Online; accessed 31-October-2013].

[Wik13c] Wikipedia. Homonym, 2013. [Online; accessed 19-November-2013].

[Wik13d] Wikipedia. Human-based computation, 2013. [Online; accessed 25-October-2013].

[Wik13e] Wikipedia. Latent dirichlet allocation, 2013. [Online; accessed 11-November-2013].

[Wik13f] Wikipedia. Law of large numbers, 2013. [Online; accessed 09-October-2013].

[Wik13g] Wikipedia. Machine learning, 2013. [Online; accessed 25-October-2013].

[Wik13h] Wikipedia. Semantic search, 2013. [Online; accessed 11-October-2013].

[Wik13i] Wikipedia. Semantic web, 2013. [Online; accessed 24-October-2013].

[Wik13j] Wikipedia. Singular value decomposition, 2013. [Online; accessed 21-November-2013].

[Wik13k] Wikipedia. Supervised learning, 2013. [Online; accessed 25-October-2013].

[Wik13l] Wikipedia. Synonym, 2013. [Online; accessed 19-November-2013].

[WRV08] Ingmar Weber, Stephen Robertson, and Milan Vojnovic. Rethinking the ESP game. In *Proc. of 27th intl. conf. on Human factors in Computing Systems, ser. CHI*, volume 9, pages 3937–3942, 2008.

[WVV$^+$01] Holger Wache, Thomas Voegele, Ubbo Visser, Heiner Stuckenschmidt, Gerhard Schuster, Holger Neumann, and Sebastian Hübner. Ontology-based Integration of Information–a Survey of Existing Approaches. In *IJCAI-01 workshop: ontologies and information sharing*, volume 2001, pages 108–117. Citeseer, 2001.