Serious Games for large-scale Argumentation Mining

Master-Thesis von Raffael Hannemann
30th March 2015

TECHNISCHE
UNIVERSITÄT
DARMSTADT

UBIQUITOUS
KNOWLEDGE
PROCESSING

Serious Games for large-scale Argumentation Mining

vorgelegte Master-Thesis von Raffael Hannemann

Supervisor: Prof. Dr. Iryna Gurevych
Coordinator: Dr. Ivan Habernal, Christian Stab

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 25. April 2015

(Raffael Hannemann)

Zusammenfassung

Den eigenen Standpunkt kommunizieren oder seinen Gesprächspartner mit einer klaren, durchdachten Argumentation überzeugen zu können, ist heutzutage eine oft geforderte Fähigkeit. Auch in den täglichen Entscheidungsprozessen kommt das Gegenüberstellen von Pro- und Kontra-Argumenten zum Einsatz. Der Theorie des Argumentierens lässt sich deshalb ein hoher Stellenwert zuschreiben.

Aus der noch jungen Domäne der Computer-gestützten Analyse und des Schreibens argumentativer Texte ergibt sich der Bedarf nach strukturierten, annotierten Korpora an Argumenten. Denn erst mithilfe solcher Datensätze werden neue Ansätze und Anwendungen im Bereich des Machine Learning, der Künstlichen Intelligenz und des Data Mining möglich. Oft stellen die Charakteristika des jeweils anzureichernden Korpus sehr spezielle Anforderungen, sodass in der Regel auf den Korpus individuell zugeschnittene Lösungen benötigt werden. Da oft das manuelle Erstellen annotierter Korpora sehr zeit- und arbeitsaufwendig ist, sind Ideen zu alternativen Methoden wünschenswert.

Diese Arbeit stellt einen neuartigen Ansatz an der Schnittstelle von Argumentation Mining und Serious Games vor, dessen Ziel es ist, einen möglichst großen Korpus qualitativ hochwertiger, strukturierter Argumente anzureichern. Dazu wird ein spielbasiertes Konzept mit Elementen des Crowd-Sourcings erprobt. Das entwickelte Spiel ist eine Anwendung für den Browser und Smartphones, über welche die Benutzer motiviert werden, zu bestimmten aktuellen Themengebieten nach und nach Argumente zu verfassen und zu analysieren. Das Spielkonzept fördert die aktive Teilnahme am Prozess, und versucht, durch gegenseitiges, im Spielkonzept verankertes Validieren die Qualität des Korpus zu gewährleisten. Im Sinne der didaktischen Spielkonzepte vermittelt das Spiel darüber hinaus grundlegende Konzepte einer guten Argumentation, was zusätzlichen Anreiz bietet, am Spielgeschehen teilzunehmen.

Abschließend wird zum Prüfen von Teilaspekten des entwickelten Konzepts eine Benutzerstudie durchgeführt und deren Ergebnisse evaluiert.

Abstract

Communicating a point of view and convincing a conversation partner with a clear, sophisticated line of argument is a generally demanded skill nowadays. Also, balancing pros and cons plays an important role in the daily decision making process. Thus, generally a high importance can be attributed to the theory of argumentation.

There is a need for structured, annotated corpora emerging from the still young domain of computer based analysis of argumentative texts and the computer assisted writing. New approaches in the domain of Machine Learning, Artificial Intelligences and Data Mining demand such types of data sets. The characteristics of the required corpus data may demand very distinct requirements, which often implies individually tailored solutions for the use cases. However, since the manual creation of annotated corpora consumes a lot of time and labour resources, the research for novel, alternative approaches is highly appreciated.

This thesis proposes a novel approach at the interface of Argumentation Mining and Serious Games to collect a large corpus of structured, high quality arguments by leveraging a game based approach with crowd-sourcing elements. The developed game is an application for browsers and smartphones which motivates the users to both generate and analyze arguments for current hot topics. The proposed game mechanics cultivate the active participation in the creation process, and furthermore have integrated a mutual validation of the generated data by the users themselves to ensure the quality of the resulting corpus. In the sense of didactic games, the proposed game furthermore imparts the foundations of argumentation theory, which provides additional incentive to participate.

Finally, to examine particular aspects of the developed concept, a user study has been conducted and its results have been evaluated.

Contents

1 List of Abbreviations

API Application Programming Interface

BaaS Back-end as a Service

CLI Command Line Interface

CSS Cascading Stylesheets

DOM Document Object Model

ESP Extra Sensory Perception

GWAP Game with a Purpose

HCI Human Computer Interaction

HIT Human Intelligence Task

HTML Hypertext Markup Language

HTTPS Hypertest Transfer Protocol Secure

IAA Inter-Annotator Agreement

Ionic Ionic Framework

IT Information Technology

JSON JavaScript Object Notation

MVC Model View Controller

NLP Natural Language Processing

REST Representational State Transfer

RST Rhetorical Structure Theory

SDK Software Development Kit

SQL Structured Query Language

SSL Secure Sockets Layer

SVM Support Vector Machine

TF-IDF Term Frequency Inverse Document Frequency

URL Unique Resource Link

WWW World Wide Web

XML Extensible Markup Language

2 Introduction

2.1 Motivation

The art of thoughtful reasoning requires capabilities in multiple intellectual subjects, such as rhetoric, linguistics, logic, philosophy, gesticulation and mimic art. People may argue in order to convince conversation partners, to state a viewpoint or to oppose one, to defend their actions, or to reason a hypothesis. Thus, arguing is crucial in professions like economics, law, politics and research, but also is an essential part of everyday's natural human communication.

Treating arguing as an art is not modern, but reaches back to ages of Aristoteles in the fourth century BC. The broad, active research however, the so called Argumentation Theory, has started in the 20th century, when various terminologies and formal models have arisen.

With the extensive introduction of computational power in science, a lot of previously complex and expensive problems suddenly became manageable. Research spawned the field of Natural Language Processing (NLP), which, at the interface of Computer Science and Linguistics, endeavors to facilitate the human-computer interaction via natural informal language, and therefore helps, as Weiser [1] describes, in the Human Computer Interaction (HCI) to provide more natural, invisible and ubiquitous interfaces. Newly created products, that make use of Natural Language Processing technology, currently often are perceived as downright magical, and artificially intelligent. With the vast amount of information publicly available in the World Wide Web (WWW), users are confronted with an information overload while searching for specific data. Search engines already help reducing the signal-noise ratio, and NLP will help to understand and structure the data further as a mean against the plethora of information.

Another kind of possible applications of NLP are sentiment analysis tools, which try to estimate the author's temper within a text excerpt. With the rise of the Web 2.0 and the availability of masses of user-generated contents on Social Media platforms, large pools of textual data became accessible, and thus, usable for analysis on a large scale. Sentiment analysis technology can be used to globally supervise public discussions automatically, e.g., in particular product reviews on blogs and e-commerce platforms, which embodies an enormous value for corporate interests.

However, what probably is even more desirable, is to not only mine sentiments, but to actually understand the cause and reasonings for a reviewer's attitude. Hence, applications, which help analyzing argumentative texts, are the consequent step further. Such kind of applications, that recognize individual argument components, and even their stances, and analyze discourse structures, are called *Argumentation Mining* tools. Next to the mentioned product review analysis, they find a use in several other usage scenarios, such as assistive writing, automated text summarization, deducing logical reasoning chains, or even to find logical contradictions in texts.

To avoid intractable parsing approaches, which usually make use of rule or pattern matching, and which are known to be error-prone and often incapable to scale with the requirements, the core of such applications frequently rather is based on super- or semi-supervised Machine Learning. This technique is conceptually designed to artificially learn from existing, known data, called *training data* or *gold data*, to derive a model, that captures the knowledge of the algorithm over time. The model then can be applied on new, previously unseen data to predict the class of the data as the output of the algorithm.

One aspect that mainly influences the performance of prediction algorithms is a good selection of the so called *features* of a Machine Learning system, which is also known as *Feature Engineering*. The features

are the distinguishing marks that characterize the input, which the learning algorithms involve to learn a model. In addition to that, the quality and the size of the training data are critical for good prediction results. Both aspects justify the yet to be improved prediction accuracies of the underlying Machine Learning components, that affect the outcome quality of the previously mentioned Argumentation Tools.

The highly inter-disciplinary, and yet young research on Argumentation Theory and Argumentation Mining generated a lot of different concepts and interpretations of how to model argument data formally, specifically suited for the particular usage scenarios. This spread led to the situation that current existing corpora that are used to train Argumentation Mining tools, are rare, relatively small, and often tailored to particular domains, such as medical research or law. Thus, they are not universally applicable for the training of bigger, domain independent systems. Being equipped with a set of high-quality labeled training data however would substantially further advance the research on Argumentation Mining.

One way to obtain an annotated corpus that meets the requirements for a researcher's project, is to employ experts who annotate raw text manually according to pre-defined guidelines. This procedure raises various new problems, such as unstable Inter-Annotator Agreement (IAA) measures that emerge from different personal opinions, which is particularly the case for annotations in the domain of Argumentation Mining and argumentative structures. Yet even more important are the significant costs of resources, that is, the cost for highly skilled labor, for annotating data manually.

2.2 Contribution

The research question of this thesis investigates whether it is possible to tackle the lack of labeled training data by a novel, crowd-sourcing based approach integrated into a Serious Game. More precisely, the thesis elaborates if on one hand, a game concept can motivate users to compose up-to-date arguments for various topic domains as a side effect of playing the game. And on the other hand, it examines, if the user base of the game as a crowd can be incentivized to reliably assess and validate the generated data and correct wrong annotations of the arguments dynamically.

Therefore, this thesis proposes a promising novel concept for a game, which aims to motivate users of the internet to actively participate in collaboratively generating and validating a database of structured arguments, that is, a corpus of arguments. The game will be released both as a smartphone and a desktop browser application, so that it will be accessible to the users in many different situations throughout the day. Additionally, the game will be designed to teach the foundations of good arguments, and thus, will offer educational benefits. Hence, the concept can be considered as a so called Serious Game. In addition to that, since the finished product will serve a certain purpose, which is creating reliably annotated data, the concept can be classified as a Game with a Purpose (GWAP).

The proposed game consists of multiple different game rounds, in which the users fulfill a certain task whose outcome contributes to the size and reliability of the resulting corpus. The corpus at the same time serves as a source for the contents of the game. Thus, since the majority of the contents of the proposed game are user-generated, the quality of this data created by the users is treated as initially unknown. Over time, by being used by the different game rounds, the user-generated data will be assessed and validated by the crowd of players, so that the quality can be determined and wrongly defined attributes of the data will be corrected. One of the key questions is, if this initial lack of knowledge about the data quality could negatively impact the fun of the users. The statistical evaluation, that has been conducted via a user study, reveals that the proposed game design is capable of handling this critical factor smoothly.

Thus, the claim is, that the presented game based approach to crowd-source the efforts of creating and annotating argumentative data can be used to tackle the lack of training data for Argumentation Mining approaches. The game does have the potential to generate a corpus of arguments over time

that is expected to become much larger than any existing comparable corpus. And furthermore, as an obvious, yet noteworthy side effect, the approach is considered to be tremendously more cost efficient than letting experts manually create and annotate data.

2.3 Risks and Challenges

The stated research question and the game pose several interesting challenges, but also fundamental risks. The success of the proposed game directly depends on the engagement of an active user base. If no users can be attracted from the very beginning on, the proposed solution not only can not meet the expectation to crowd-source the creation and assessment of data, but also is hard to evaluate scientifically.

As in any game, the application that is to be developed must generally be carefully designed in terms of the mechanics and the balancing of the gameplay, its visual appeal, providing incentives and a long time motivation to keep the users playing the game.

Initially, a well curated, editorially composed set of bootstrapping data is required, used by the game to provide data that the users can interact with. This will be time and labor intense work, but will be necessary to eventually ignite the collaborative gaming process.

Next, the mutual validation of the generated data amongst the users must be robust and work reliably to produce the desired outcome. The users also not only need to be willing to perform the cognitively demanding task by playing the game, it also must be ensured that the users perform the right task, which produces the actually desired effect.

2.4 Thesis Organization

Chapter 3 Related Work presents the current state of the research in Argumentation Theory, Argumentation Mining, Serious Games and crowd-sourcing approaches for annotating corpora. In this section, the research gap will be depicted in detail and the value of the contribution of this thesis will be further explained.

Chapter 4 Game Design describes the game based approach examined to tackle the research question. The chapter provides a detailed overview of the game design and the integrated loop of creating and analyzing data.

Chapter 5 Implementation is about the software design of the developed game, separated into its main components, the client and the server side. Furthermore, the mechanics of the mutual validation are explained.

Chapter 6 Evaluation describes the approach taken to evaluate the developed product with regards to the initial research question. In particular, an online user study has been conducted for the evaluation, which is described in detail. Furthermore, the conclusion based on the user study results is made.

Chapter 7 Conclusion provides a final conclusion for the researched topics of this thesis. The chapter also contains recommendations for future work.

3 Related Work

This chapter provides an overview of the existing research related to the two main fields of this thesis, Argumentation Mining and Serious Games. These two fields will be examined in detail: With regards to Argumentation Mining, first Argumentation Theory, Argumentation Models, Argumentation Mining itself, and Argument Corpora will be presented. The domain of Serious Games is subdivided into Games with a Purpose, Human Computation, and Serious Games to generate Data.

Not only shall this chapter present the theoretical background of these areas and illustrate some potential applications, but it is also meant to illustrate the research gap this thesis attempts to fill. Tangent existing solutions are briefly explained and the reasoning that backs up the decision to develop a novel approach is clarified. Furthermore, this chapter also precisely delimits the scope of this thesis.

3.1 Argumentation Theory

Many explanations exist that attempt to capture what argumentation is specifically about. Van Eemeren et al. [2] describe argumentation as "*a verbal activity, which is normally conducted in an ordinary language (such as English). A speaker or writer, engaged in argumentation, uses certain words and sentences to state, question, or deny something, to respond to statements, questions or denials, and so on*" [2, p. 2]. This definition contains many important aspects what the concept of argumentation accounts for. First of all, it notes, that argumentation is something a human can perform to express herself with a certain goal in mind. The actor's output may be either oral or in text-form, but in natural language either way, and it is made out of thoughtfully ordered sub-components.

Furthermore, "*in an attempt to justify a standpoint, the constellation of propositions consists of one or more pro-arguments ('reasons for'); in an attempt to refute a standpoint, it consists of one or more contra-arguments ('reasons against'). [...] Argumentation is aimed at increasing (or decreasing) the acceptability of a controversial standpoint for the listener or reader*" [2, p. 4]. This describes the existence of certain roles of the sub-components within an argument structure and it mentions the act of purposefully making use of these roles to influence an audience. These complex aspects separate the concept of an argument from a simple explanation, which is "*a statement or account that makes something clear*"[1], or "*a reason or justification given for an action or belief*"[2].

3.1.1 Aristotle's Rhetoric

When giving an overview of the research on Argumentation Theory, the Greek philosopher Aristotle and his work need to be mentioned, which describe the art of persuasion. Aristotle [3] subdivides the technique of persuasion, that is, the act of convincing an audience, into the three elements *Ethos*, the credibility and character of the speaker, *Pathos*, the emotion of the receiver or receivers the speaker has to awake, and *Logos*, the usage of reasoning. Aristotle emphasizes the latter aspect to be the most important one in persuasion.

[1] cf. entry for *explanation* in Oxford Dictionary British English
[2] cf. entry for *explanation* in Oxford Dictionary British English

Generally, Argumentation Theory is a fairly ancient, inter-disciplinary field covering aspects of philosophy, psychology, linguistic, and others. Abstracting the informal aspects of argumentation and focusing on creating precise models is required to provide a solid foundation for the computational examination of arguments. In the following, the main concepts of Argumentation Theory from an computational perspective are introduced.

3.1.2 Toulmin Model

The British philosopher Stephen Toulmin [4] demanded a more sophisticated distinction of the argument components than into either premise or conclusion. Therefore, he describes "*The Layout of Arguments*", which relies on six different types of components: First of all, the *claim* is the statement which is being argued, "*whose merits we are seeking to establish*" [4, p. 90]. Next, the *data* are "*the facts we appeal to as a foundation for the claim*" [4, p. 90]. Thus, the data is what is being used, to prove an argument. In other literature, the data is sometimes described as *ground* or *evidence*.

However, Toulmin explains that "*[our] task is no longer to strengthen the ground on which our argument is constructed, but is rather to show that, taking these data as a starting point, the step to the original claim or conclusion is an appropriate and legitimate one*" [4, p. 91]. Therefore, he defines the so called *warrants*, which are the "*hypothetical statements, which can act as bridges*" between the claim and the data (cf. [4, p. 91]). As an example, the following warrant illustrates its purpose as a bridge from data D to claim C.

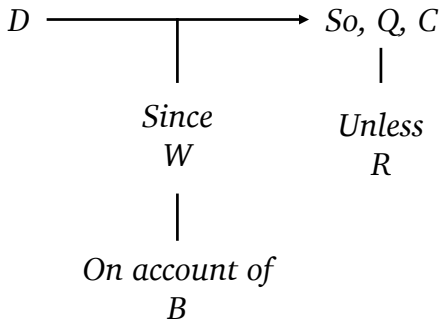
$$\text{If D, then C.} \tag{3.1}$$

Since sometimes "*the degree of force which our data confer on our claim in virtue of our warrant*" [4, p. 93], a further qualification of the data may be necessary, which Toulmin calls *qualifiers* and *rebuttals*. Qualifiers can be seen as additional elements that cut back the power of warrants for the claim to be true. Rebuttals are components, that introduce conditions, or exceptions, under which a warrant can not be used to bridge from the data to the claim. Here, for example, counter-examples may be used.

Backings, the sixth of Toulmin's component types, are described as the components used to back up, that is, to support, a warrant, "*without which the warrants themselves would posses neither authority nor currency*" [4, p. 96]. Toulmin provides an example, that covers the component types in one argument, which is shown in figure 3.1 (cf. [4, p. 97]).

This example not only illustrates the individual components, but also demonstrates the scope of the Toulmin Model: Arguments are isolated units with no direct relation to previous or following arguments within a text. Therefore, important aspects of the rhetoric are not considered in this model. This does not necessarily introduce a real disadvantage concerning the usage of the model in real-life applications: Argumentation structures can be examined on a *macro-argumentation* or on a *micro-argumentation* level. The two perspectives differentiate between either considering argumentation as an act of one or multiple parties. In macro-argumentation, multiple arguments are provided to form text as a whole, while on the other hand, the micro-argumentation level considers the individual sub-components, and their functions and roles, of one specific argument. Since this thesis aims to generate data mainly to enable further argument *component* analysis, it is focused on argumentation on a micro-level.

Due to a broad adoption of Toulmin's model, a lot of further suggestions have emerged with regards to the inter-argument connections, as Peldszus and Stede [5] point out. For example, both Öhlschläger [6] and Kienpointner [7] propose, that if an argument contains a backing for a warrant component, this should be interpreted as a new, connected argument instance, in which the warrant acts as claim component, and the backing as data. Thereof, inter-argument dependencies can emerge. Similarly, Klein [8] suggests a recursive tree structure argumentation scheme, consisting of a *main claim* as the



Harry was born in Bermuda. *(Data)*

Since a man born in Bermuda will generally be a British subject. *(Warrant)*

On account of the following statues and other legal provisions: [...] *(Backing)*

So, presumably *(Qualifier)*, Harry is a British subject. *(Claim)*

Unless both his parents were aliens/he has become a naturalized American/... *(Rebuttal)*

Figure 3.1: Toulmin's model of argument components, with D as the data, W as the warrant, B as the backing, Q as the qualifiers, C as the claim, and R as the rebuttal.

root and supporting arguments underneath. This has been extended by Wunderlich [9], who not only integrates supportive, but also attacking arguments into the tree, which allows to further model the arguments on a more fine-grained level.

Although alternative approaches and additional techniques on how to model arguments and argumentation discourse structures exist, in particular the work of Toulmin influenced the research profoundly, which is why its details have been presented here in detail. The Toulmin Model of argumentation serves as a great tool to analyze arguments, not least because of the clear distinction of argumentative text passages into sub-components, that have particular roles and relationships amongst each other. Therefore, Toulmin's model and his research often serve as a foundation, especially in NLP, where extracting structured data out of unstructured, natural language information is one of the main efforts.

3.1.3 Walton's Argumentation Schemes

One of the alternative approaches to Toulmin's model on a micro-level has been proposed by Reed and Walton [10]. Instead of providing a generic model of components with certain roles, they provide a scheme-set of prototypical patterns, with which arguments can be detected. Taking up again concepts of Perelman and Olbrechts-Tyteca [11] and Hasting [12], they outlined an extensive list of the so called *argumentation schemes*, that allow the evaluation of the type of an argument. Walton et al. [10] originally describe argumentation schemes as "*forms of argument (structures of inference) that represent structures of common types of arguments used in everyday discourse*" [10, p. 1].

In their terminology, the typical components of an argument are the *major premise*, the *minor premise* and the *conclusion*. Their work not only helps identifying arguments, but also identifying and avoiding *fallacies*, that is, wrong reasoning.

3.1.4 Other Models

To provide an in-depth overview of the contemporary research of Argumentation Models, Bentahar et al. [13] summarized the most popular models. According to them, due to the variety of fields of application, such as knowledge representation, explanation, proof elaboration and others, a lot of argumentation models with fine-granulated differences have emerged, tailored to specific needs. The authors remark the lack of a global view of existing argumentation models and hence, they propose a framework, which summarizes the characteristics, the advantages and disadvantages of the various existing approaches, motivated by the opportunity to allow researchers to choose the appropriate model for their use case more easily.

3.2 Argumentation Mining

Out of the preparatory work on Argumentation Theory and the progress of computational analysis of natural language, the field of *Argumentation Mining* emerged. Generally in Computer Science, *Data Mining*, as Chakrabarti et al. [14] describe, refers to "*the science of extracting useful knowledge from [...] huge data repositories*". That is, Data Mining can be seen as the act of automatically gathering data and inferring new data. It is "*an interdisciplinary field at the intersection of artificial intelligence, machine learning, statistics, and database systems*" [14], which "*combines techniques from different areas in computer science and statistics*" [14].

Argumentation Mining is a fairly new sub-field in NLP and Information Retrieval. As Mochales-Palau and Moens [15] explain, it aims to detect argumentative discourse in textual natural language data. Furthermore, Argumentation Mining also attempts to detect the local structure of arguments to classify text components and assign functional roles to them, such as *claim* or *premise*, and their connections, such as *support* or *attack*.

Peldszus and Stede [5] define Argumentation Mining as "*the task of building a formal representation for an argumentative piece of text*", and the "*automatic discovery of an argumentative text portion, and the identification of the relevant components of the argument presented here*". They name legal debates, product review opinion reasoning analyzing, and political public opinion analyzing as the motivation for their survey.

The applications of Argumentation Mining are highly diverse. The dominating motivation for web search is decision making and information seeking, according to Rose and Levinson [16]. Argumentation Mining techniques can help finding relevant information in the information overload users are confronted with on the web. Additionally, Argumentation Mining techniques can be used in automated text summarization [17], computer assistive writing [18], and assessment [19], and enhanced sentiment analysis [20].

Teufel [21] proposes *Argumentative Zoning*, an Argumentation Mining approach, that allows to analyze rhetorical structures in scientific articles. Generally, the approach aims to push forward the generation of document surrogates with regards to document retrieval, and thus, to improve document search. More precisely, the approach also attempts to derive a scientist's stance towards existing research. The zoning classifies each sentence of an article into one of seven rhetoric roles, that capture the rhetorical meaning of the sentence with respect to the document in total. The first three, *Background*, *Other* and

Own, indicate author ownership attribution, whereas the remaining four, *Aim*, *Textual*, *Contrast* and *Basis*, signify information about the author's stance. While classifying, the argumentative zones within the document emerge. Then, the articles can be summarized tailored to a specific interest, such as the stance of the author towards existing research. Teufel deployed both a *Naïve Bayes* classifier and a rule-based classifier, with features such as *Term Frequency Inverse Document Frequency (TF-IDF)*, sentence-to-headline relations, statistical features, such as positioning and lengths, and semantic features, such as verb tenses, modality, and others.

Motivated by the need for improved summarization and question-answering techniques, Merity et al. [22] present an advancement of Argumentative Zoning in scientific literature, which improves the F-score, a measure for the accuracy of classification algorithms, by 23% to nearly 96.9%. They apply a *Maximum Entropy* classifier, using fairly simple, primarily lexical features and sentence positioning features.

Schneider et al. [23] present a tool that supports the identification of arguments in product reviews. The tool aims to help both producers to analyze, and their customers to better understand product reviews. Their approach identifies arguments semi-automatically based on rules using a set of argumentation schemes specifically tailored to reviews for camera products. They justify their approach with the fact that automated systems to reliably extract the desired information were still too hard to accomplish with regards to the expected accuracies.

Stab and Gurevych [24] emphasize the importance of identifying argumentative relations and the argumentation structure for argumentation mining. They model argumentative relations, such as *support* or *attack*, as directed relations between two argument components, that represent the argumentation structure. These structures are between either non-adjacent or adjacent sentences or clauses, and may be signaled by indicators, such as discourse markers. The emerging argumentative discourse can exhibit reasoning chains across multiple arguments.

3.2.1 Separating Argumentative and Non-argumentative Texts

Being able to separate argumentative from non-argumentative text passages is an important capability in Argumentation Mining, for example, as pre-processing steps for more fine-grained approaches in the sub-sequential step. However, these approaches do not examine the exact argumentative roles of the argument components.

Motivated by the chance to improve the visualization of argumentative texts and the improved indexing for information retrieval, Moens et al. [25] propose an approach to detect arguments in mainly legal texts and newspapers as a classification problem. Their classifier takes into account lexical, syntactic and semantic features, and furthermore also discourse properties.

3.2.2 Classifying Argument Components

Mochales-Palau and Moens [15] present an approach that aims to first detect the argumentative propositions within a text using a Maximum Entropy classifier, and then to determine their argumentative function, premise or conclusion, using a Support Vector Machine (SVM). They have applied their approach on legal texts. Their evaluation reveals a *F1* measure of 68.1% for the detection of premises and 74.1% for the detection of conclusion propositions.

Stab and Gurevych [26] note that most Argumentation Mining approaches mostly focus on the identification of argument components, whereas the discourse structure is important for the overall argument

quality as well. Therefore, they make use of a Machine Learning based multi-class classifier to identify argumentative discourse structures in persuasive essays, including the *support* or *attack* relation. Their Machine Learning features include structural features, such as statistics and token positions, lexical features, such as word pairs, and also syntactic and contextual features. In the evaluation they have reached macro *F1* measures of 72.6% for identifying argument components and 72.2% for identifying argumentative relations.

3.2.3 Identifying Argumentation Structures

Next to Argumentation Mining approaches, that examine arguments on a *micro-level*, analyzing and determining argumentative structures on a *macro-level* is also a challenging, yet interesting task. These approaches treat the set of arguments of a text as a whole and identify the inter-argument relations, in order to extract a tree of subordinate arguments related to a main argument. Although this thesis is focused on the micro-level, a few concepts are introduced here.

According to Peldszus and Stede [5], the argumentation structure is related to Rhetorical Structure Theory (RST): They mention the *coherence relation*, which describes that spans of text which are near to each other, have a semantic relatedness. According to them, to judge a relation between two text components, one needs to re-construct the writer's intention.

Based on *Textual Entailment*, a relation between two passages of text, which is given if the meaning of one of the two can be derived from the user, Cabrio and Villata [27] propose an approach to extract arguments and determine the accepted positions from online platforms, that allow debating. Their goal is to support users, who would like to enter a debate without having to read all existing arguments.

3.3 Current Corpora for Argumentation Mining

Argumentation Mining based – in particular the Machine Learning based – approaches, require large corpora that can be used for training the models. Often, the current research is based on domain dependent and fairly small corpora, which can be shown by some of the examples below.

Reed and Rowe [28] propose *AML*, a Extensible Markup Language (XML) based mark-up language to describe argument structures. Their software, called *Araucaria*, provides an interface for analyzing and diagramming structures in argumentative texts, and is therefore valuable both for pedagogical purposes and also argumentation theory oriented research activities. Based on the tool, the so called *AraucariaDB* database of arguments has emerged, which by the end of 2004 contained 80,000 words, forming 1,500 premises (cf. [29]). However, due to the different, international sources of the corpus, the quality of the contained arguments can not be precisely judged.

Stab and Gurevych [30] emphasize the lack of corpora with argument annotations. Motivated by the demand for such corpora, that include annotations for argument components and the relations, they propose a novel annotation scheme, which aims and manages to improve the IAA, as shown in an annotation study. The scheme takes into account arguments, their components, that is, claims and premises, and their relations, support and attack. They have applied the scheme on a set of 90 persuasive essays and provide the corpus publicly, which includes 429 claims and 1,033 premises.

In a study on the structure of argumentation in case law, Mochales-Palau and Moens [31] investigate the discoursive and argumentative characteristics specifically of documents of the European Court of Human Rights. Their scheme annotates claims and both supporting and refuting premises, however it

is not described in detail though. It has been applied on a set of 30 documents by three independent expert annotators, from which a corpus emerged, however no further details could be found.

Other approaches are specifically tailored to certain domains, such as product reviews (cf. Villalba and Saint-Dizier [32]). Yet another problem is the lack of studies (cf. Cabrio and Villata [27]), with regards to the IAA among the annotators, so that the reliability of the corpora can not be judged. Clearly, there is a need for domain independent, up-to-date, reliable corpora of arguments, that include argument component annotations. For the approach presented in this thesis, Serious Games play an important role, which are further discussed in the following section.

3.4 Serious Games for Generating Data

Zyda [33] describes a Serious Game as a "*mental contest, played with a computer in accordance with specific rules, that uses entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives*" [33, p. 26]. This is a fairly abstract and generic description. One could also describe Serious Games as computer games with a main goal other than pure entertainment. Michael and Chen [34] define a Serious Game as "*a game in which education (in its various forms) is the primary goal, rather than entertainment*" [34, p. 17], hence, they focus on the educational aspect.

Alvarez et al. [35] differentiate between five major categories of Serious Games: The first is the class of *Edutainment games*, that is, games that transfer knowledge or train in a playful manner. Next, *Advergaming games* are a mean to advertise a product or a brand. *Edumarket games* generally try to sensitize the player for a critical topic or message, such as the need for humanitarian aid. *Political games*, as the name suggests, attempt to bring political topics closer to the player. The fifth class of Serious Games are the *Training and Simulation games*, which simulate real-life activities or situations more realistically. The latter one is probably the most prominent category, with popular representatives, such as *Sim City* (Maxis) or *Flight Simulator* (Microsoft).

In a survey conducted by Michael and Chen [34], 63 game developers indicated that students of any level are the top target group with nearly 53% of their Serious Games projects. The general public is the target of 47% of the games, and corporate management and/or executives are targeted by nearly 27% (cf. [34, p. 26]). It is noteworthy that many of the survey results also list education and government personnel, healthcare professionals, military personnel and even activists as target groups.

3.4.1 Games with a Purpose (GWAP)

The so called *Human Computation* is a new field that, as Von Ahn [36] explains, studies the effects of combining human and computer based computation to solve problems, which would be otherwise not solvable. Similarly to the crowd-sourcing approach, Von Ahn equates human brains with processors in large-scale distributed systems. In the age of the Web 2.0, the internet can be exploited to enable a massive collaboration of hundreds of millions of users, according to Von Ahn.

Games with a Purpose, also known as Human-computation games, are computer games that exploit humans to solve problems (cf. Von Ahn [37]). One of the first truly successful GWAPs is the Extra Sensory Perception (ESP) game. Here, Von Ahn [38], as one of the pioneers of GWAPs, explains a gameplay in which two independent, randomly chosen players need to propose a description for a random image. The game mechanics make sure that playing the game is enjoyable and motivates to keep playing. Besides, the game is designed to prevent cheating and vandalism by various methods. Hence, as a side effect of this gameplay, the users generate verified labels for images, which can be exploited to improve image recognition algorithms.

Multiple derivates of the ESP game are part of Artigo [39], a set of labeling games specifically tailored to artwork. Between 2008 and 2013 more than seven million tags have been collected by nearly 180 thousand players (cf. Kohle et al. [40]). The resulting data are being used to allow visitors to search a database of artworks.

Lots of other GWAPs have been developed that attempt to tackle research motivated problems in various domains. *Peekaboom* [41] lets players annotate specific objects in images to improve computer vision algorithms. *EteRNA* [42] is a game where puzzles need to be solved. The results are utilized in *RNA* molecule research. Players of *Disguise* [43] evaluate color blending algorithms in the field of data visualization.

Von Ahn [44] backs up the success of Games with a Purpose by three aspects: First of all, the fraction of the world population which has access to the internet is increasing. Next, there are tasks which are impossible to solve for computers, but easy for humans. And third, people invest a lot of their time playing digital games. The remarkable feature of a GWAP is that as a side effect of playing the game, a valuable, desired computation is being performed. This computation is usually hard for computers, but easy for humans to solve. However people still do not play the game because of their interest in solving this task, but to be entertained. Furthermore, a GWAP can be assumed as successful, if enough human-hours are invested by the players solving the computational task.

Generally, Von Ahn models a game as a tuple of the goal the player or players have to achieve, and the rules that define the possible actions that can be performed in the current game state (cf. [44]). With regard to a GWAP, he derives that the rules of such a game should encourage the players to fulfill the computational task correctly.

3.4.2 Mechanical Turk

Mechanical Turk (Amazon Corp.), is a commercial crowd-sourcing platform on which so called *Workers* can accomplish *Human Intelligence Tasks (HITs)* that are provided by *Requesters*. In return they receive a financial compensation. Due to the substantial costs of annotating or generating gold data for NLP by experts, the platform quickly gained popularity among researchers. As the platform provider, Amazon applies multiple strategies to improve the reliability of the Workers' output: Insufficient or low-quality Results can be rejected and individual Workers can be blocked. Next, HITs can be set up to require completion by multiple, different Workers to achieve redundancy. Additionally, Requesters can specify a minimum qualification for Workers required to participate in a HIT. But still, blindly relying on the credibility of the Workers carries a risk. Several studies have assessed the platform as a tool for NLP related problems.

Callison-Burch et al. [45] consider various issues concerning the use of Mechanical Turk and micro-tasks involving human intelligence as a cost-effective source for NLP gold data. As in general crowd-sourcing based approaches, the resulting data may be noisy and of poor quality, due to the fact that the participants are mainly non-experts, due to the diverse demographics, and due to the existence of spammers, which misuse the platform for their own economic purpose. Next, the definition of a HIT can be difficult for the Requester to compose, and on the other hand hard for the Workers to understand, resulting in an increased amount of bad samples.

Akkaya et al. [46] collect non-expert annotations for *Subjectivity Word Sense Disambiguations* with the goal to improve Sentiment Analysis. They reveal the down-sides of using non-experts, for example, the presence of spammers who do not follow the guidelines of the researchers, and the challenging noisiness of the resulting data due to the lack of experience of the workers. They also monitored the over-time

performance of the workers to see if there is a learnability in the absence of any other rewarding or feedback, which however they could not observe.

Fort et al. [47] raise awareness for further questions one should consider when Mechanical Turk is an alternative, which include the debatable ethics of exploiting human work for wages below $2 per hour, and the legal issues, emerging from a de facto employment activity, that need to be considered.

3.5 Chapter Summary

This chapter provided an in-depth overview of the current state of the research in the various domains this thesis covers topics of, such as Argumentation Theory, Argumentation Mining and Serious Games. As shown, there is a demand for large, domain independent corpora of arguments, which can be used for Argumentation Mining.

Furthermore, the chapter clarified the research question of this thesis, which is, whether a game based approach can be used to tackle this lack, and also illustrated the value of the contribution, since, as shown, annotated data are hard to obtain and to create. The overview showed that the approach to use a GWAP to harvest a corpus of annotated arguments based on crowd-sourcing is novel, yet highly promising.

The next chapter will introduce the design of the game in detail, that aims at providing a fun way to learn the basics of argumentation, but also at generating a large of corpus of validated arguments as a side-effect.

4 Game Design

This chapter describes the design of the developed game in detail. In particular, this chapter describes the individual sub-goals which are achieved by means of the game, the individual elements of it, and respectively which alternatives have been considered.

4.1 Game Design as an Art

It is a challenging task to conceive a game design around a generally rather abstract and pale topic, which is creating and analyzing textual arguments. Consequentially, the individual pieces that form such a game have to be carefully considered, and additional motivating elements, which not necessarily contribute to our final goal, but augment the joy of playing it, deliberately need to be integrated.

Casual Gaming

The art of game design is a whole research domain of its own. With the rise of mobile devices, equipped with a persistent internet connection and powerful computing units, games have found their ways into the lives of many people. More importantly, the so called casual games recently gained a lot of traction, and therefore, a nascent market emerged, covering interests of many different kind of stakeholders. Casual games aim to offer amusement while waiting, traveling or to relax without the burdens of deep and grueling storytelling, and immersive gameplay of big blockbuster game titles that are developed for game consoles at home. As Juul describes, the "*casual revolution [...] is a breakthrough moment in the history of video games. This is the moment in which the simplicity of early video games is being rediscovered, while new flexible designs are letting video games fit into the lives of players. Video games are being reinvented, and so is our image of those who play the games. This is the moment when we realize that everybody can be a video game player*" [48, p. 2].

Debating as a Game

Debating is a challenging cognitive activity to train many different aspects of one's own soft skills: eloquence, rhetorics, repercussiveness, analytical thinking, and gaining political common knowledge, just to mention a few of them. Backed by many supporters, lots of debating clubs have been founded in schools, universities or privately[1]. There are even international debating contests periodically organized[2], which seems like a strong indicator for the popularity of debating as a fun mental exercise.

Out of this, a potential to exploit debating as a phenomenon emerges, that is considered to be useful as part of a gameplay, which contributes to the goals of the thesis. Hence, the proposed gameplay attempts to capture the strong commitment people are having while debating, by offering multiple differing mini-games that are inspired by the concept of debating.

With the decision to craft a game around it, the developed game stands out from other existing approaches, that usually try to replicate the concept of debating in an online platform. Some of the existing platforms[3] make use of similar elements as the proposed game, such as high-scores and gamification, but

[1] See Verband der Debattierclubs an Hochschulen, `http://www.vdch.de/`
[2] See World Schools Debating Championships, `http://schoolsdebate.com/`
[3] See for example Create Debate, `http://www.createdebate.com`

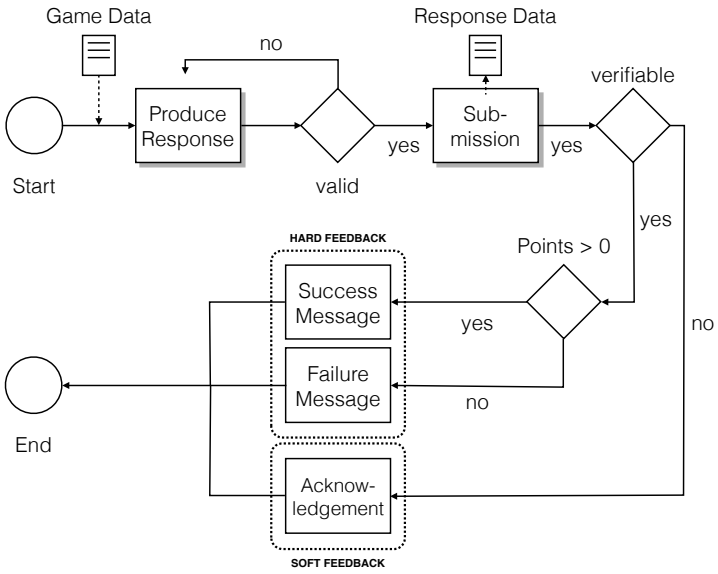


Figure 4.1: The procedure through one game round from start to end. Game and user actions are shown as boxes, whereas the latter ones have a shadow. Conditions are shown as diamonds. The *valid* condition checks if the response produced by the user is valid in general. The *verifiable* condition checks if the game can verify the response.

generally they are designed as online discussion boards at their core. Furthermore, the proposed game is not only meant to be accessible, but also usable from all major digital platforms, and therefore, users will be able to participate in all kinds of different daily situations.

Next, the proposed sophisticated gameplay ensures a mutual validation amongst the users, and thereby, not only improves the game contents the users interact with, but also improves the quality of the resulting corpus. The proposed system therefore is considered to be self-adjusting and does require only a minimum of editorial intervention from outside.

4.2 Main Game Components

A game can be described as a system consisting of different types of interacting entities, that have certain attributes (cf. Salen at al. [49, p. 50]). Broadly speaking, in the case of the proposed game, this system can be depicted as a hierarchy of worlds, levels, and game rounds, and thus, it follows a commonly known structure. Deliberately, the game has not been designed by choosing a known game genre, and then trying to integrate tasks, that help achieving the thesis goal. Instead the individual possible tasks the users would have to fulfill to reach the thesis goal have been identified first, and then have been composed into a completed gameplay, by adding necessary elements, such as levels, points, multiplayer, and more.

Game Rounds

The game components are explained from the bottom up, starting at the most detailed level: The game presents atomic mini-games to the users, the so called *game rounds*. In the game rounds they can take action and therefore will be rewarded with points. Conceptually, every game round follows the same procedure, which is illustrated in figure 4.1: Initially, the users are faced with data, which they need to interact with, or alternatively, some sort of input form is provided that allows to actually compose data. This initial data, the users need to process, is called *game data*. The users then handle and answer to

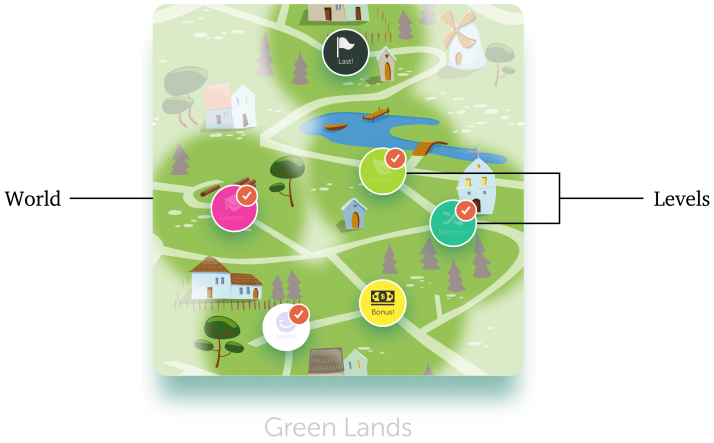


Figure 4.2: One of the worlds of the final game, consisting of multiple levels.

the game data by producing a response, that is, they make a choice or compose an argument. Once they are ready and would like to submit their answer, the data will be validated by the game, which means checking that the response makes sense according to the current game round, similar to form validation on web pages.

If the validation succeeds, the answer submission will occur, and thereby the users produce what is called *response data* in order to finish the round. Based on whether the game can verify the response data, that is, determine the correctness of the response data, the game will reward the user with a certain amount of points: For wrong response data, no points are given, for correct response data, the full amount as specified by the game round, will be given. In the case that the response data can not be verified, a reduced amount of points, compared to the amount for a correct answer, will be given. The game then will eventually either present *hard feedback* for verifiable response data, or *soft feedback* otherwise, which both will be explained in detail later. The types of game rounds which are available will be detailed later.

A sequence of multiple game rounds form a *level*. To complete a level, each of the game rounds must be finished, independently of whether the user successfully fulfilled the respective task or not. Since the user's game progress is not being stored on a game round granularity, but on a level granularity, all contained game rounds are meant to be played in a sequence. Once completed, the users cannot return to the previous game round, but have to move on to the next one. In contrast to game rounds, levels exist only once within the game. Game rounds on the other hand are repeatable units which the user is confronted with again and again. Additionally, levels can be individually designed, for example, to be dedicated to a certain topic domain, or a particular game round type.

Finally, all levels reside in a so called *world*. A world is a visual wrapper for all included levels, visually resembling a treasure map, see figure 4.2. The look of worlds can be freely customized, which allows to capture a certain atmosphere, or even a particular theme, which may indicate the game contents of the contained levels. There can be multiple worlds within the game next to each other, and all worlds together shape the game as a set of hierarchical entities.

Users in the game are represented as small circular comic faces, for example, in the ranking lists. The game allows the users to choose one of multiple different faces, called *avatars*, from a list. As Madigan [50] points out, avatars are a powerful game element. Customizing avatars is "*[...] often about building a better version of ourselves*" [50]. Although the ways to adapt the avatar in this game are very limited, it is expected that their customization can indeed bind the users to the game psychologically.

4.3 User Goals

It is important to keep as many users actively playing the game as possible in order to ensure a growing corpus. Therefore, the gameplay has to offer valid goals, which provide enough incentive in a whole to come back to the game every once in a while. Multiple goals were designed, which are described hereafter.

Completing Levels and Worlds

It is one of the user's goals to finish all levels of all worlds within the game. Initially, the game worlds are covered by a fog, which can be cleared by the user by completing levels. Each completed level will be visually marked, but can be played again, if the user would like to, and if the level configuration generally permits it. Any level can be set up to require the completion of one or multiple other levels within the same world. This can be used to integrate unlocking mechanisms, so that, for example, there is a final level within a world which will be unlocked to play only once all other levels have been completed. This provides a further incentive for the users to keep on playing. This stepwise completion of levels can also be justified by the *Ovsiankina effect*, a psychological phenomenon that describes a need to resume an unfinished task, if it previously has not been finished yet (cf. [51]).

Once all levels within a world have been completed, the world itself is being marked as complete. This is only a visual indicator to reward the users for their efforts. In the case that each world of the game has been finished, a message will appear which acknowledges the user's success.

Improving Global Ranking

Ranking is the second important game goal. The game allows to complete the individual components stepwise, but for the sake of growing the corpus, an *infinite* gameplay must be part of the game, so that the users can keep on playing after having completed the levels. Therefore, repeating levels allow the users to increase their amount of collected points, and hence, improve their global ranking in the comparison with other players. Consequently, since a user's progress is publicly visible within the game, this also endangers the users to lose a good position, and to even fall off into the last places of the global ranking. The game therefore provides a global high-score list which every user can access to view their specific ranking. This allows social comparison, with the best and the worst players, but also with your acquittances and friends.

4.4 Handling User-Generated Game Contents

So far, the game has been described with regards to its elements, the goals for the users and the motivation to keep playing it. However, the game serves a scientific purpose, which is to harvest a large corpus of arguments over time. What delimitates the game from traditional games, is the fact that the game content, that is, the data source for the game rounds, is generated by the users themselves. The contents at the same time represent the corpus that is being collected. So from the perspective of the users, their generated data ensures a steadily up-to-date set of game data, whereas from the scientific perspective, the enlarging of the corpus is secured.

Before describing how the users actually create and analyze data in different types of game rounds, and as a result, provide game contents for the game, the mutual validation system is explained in the following paragraphs.

Mutual Validation

Since the game contents are user-generated, the quality of them is initially unknown. This involves several risks for the gameplay: Although the generated data is annotated by their author, there is no evidence for the correctness of these annotations. For example, a freshly composed argument may be set up to be a *Pro* argument by the author, but actually the opposite is true. Other players, which are faced with the falsely annotated argument may be confused or even frustrated, when their response action is incorrectly judged by the game to be wrong. Furthermore, malicious users may intentionally inject offensive content or content of low quality. As a result, the correctness of the data must be questioned. For example, if the users need to guess the stance of an argument or the type of an argument component within the game, the system is faced with the following issues:

- Is the argument indeed a *Pro* argument?
- Is this component really the *claim* of the argument?

To stem against these risks, a mutual validation system is integrated, which exploits the wisdom of the crowd to assess and also correct the annotations of the authors. The system is based on majority voting and validates if specific data is what it is described to be. This concept merges the ideas of manual expert annotation and crowd-sourcing and thereby reduces the weaknesses of both approaches: Expert annotation is highly labor intensive and furthermore a taxing task. On the other hand, the players of the game are far from being annotation experts, but the masses of participants embody the potential to eventually reveal the correct labeling.

Based on the current state of the majority voting, the individual data can be separated into either *verifiable* or *non-verifiable*. To be verifiable, the users as a crowd must have taken a clear stance upon the correct value of the data. More precisely, this means, that the majority voting for the data has clearly indicated one specific outcome.

Hard and Soft Feedback

Next, if the data used by a game round is verifiable, the game can provide *hard feedback* on the user's response data, that is, it can clearly judge if the users's response is correct or not. Otherwise, only a *soft feedback* will be returned by the game. In the latter case, the user will receive an acknowledgement by the game, and a reduced amount of points. Note that in these situations, users will consequently receive points for their response, even though the response might have been incorrect, as the mutual validation may have revealed at some later time. However, the users can not exploit this weakness, since the game does not reveal if the current game data is verifiable or not, up to the point when the feedback will eventually be provided.

Collecting Votes

The voting is no separate aspect of the game, but instead an integral part of it: Each response produced in a game round, that *assesses* data, will be taken into account as one vote during the calculations for future game rounds. The details of the mutual validation system and the model behind it are described in chapter 5. With the voting system, ideally, a loop of data creation and validation is put into operation.

Reporting

Next to the mutual validation, all data can additionally be reported by the users while playing the game. Therefore, whenever they are faced with user-generated data in a game round, the game provides a function to report the entire state of the current game round. The function intentionally is described

fairly broadly to the users, so that they can make use of it whenever they assume the data to feature any type of anomaly, or if the data seems to be spam. All reported data is stored and intended to soon be viewed and processed by editorial human assistants of the game.

4.5 Type of Game Rounds

In the following section, the different types of game rounds are described. Game rounds are the atomic units that act as the building blocks for levels. In analogy to the overall goals to let users both *compose* and *analyze* arguments, there are two main classes of game rounds, which are characterized by the output they produce by the users' actions: Game rounds in which arguments are composed and game rounds in which arguments are analyzed by the users. Furthermore, to allow descriptive and introductory static contents placed in between two game rounds, there is an additional third type of game rounds.

Commonalities

In the game rounds, whenever the game presents an argument or demands to create one, relevant context information is also being displayed. A shown argument always refers to a certain *debating topic*, such as "*Should cellphones be banned from class rooms?*". The debating topic aims to motivate the players to compose or analyze an argument carefully. Additionally, each debating topic can optionally reference a web article which the debating topic is based on, or which backs up the debating question. The game displays only the headline and the publisher's name of the article, and allows the users to visit it on the original web article site to study the details if they would like to. Furthermore, the domain that the debating topic belongs to, for example, *Conspiracy* or *Economics*, will be displayed.

4.5.1 Game Rounds to Compose Arguments

The first class of game rounds contain those, which produce corpus data instances as their result, which is the case for the round in which new arguments are composed by the users.

In the *Compose an Argument* game round, the users create a new Pro or Contra argument, motivated by a specific debating topic that is shown. The topic either is specifically pre-defined, or randomly chosen. In addition to the topic, the topic domain, and optionally a referenced article, is shown as previously explained. The topic however should provide sufficient incentive, and should be stated clearly enough to take action. It is important to mention that the set of available topics within the game must be carefully editorially composed, so that they do not introduce any ambiguity and it is at all times very clear what a Pro or Contra stance would precisely mean with regards to the topic.

Instead of offering a free text input mask, the users compose a new argument by typing the distinct argument components individually into a template like form, whereas each component deserves one input field, as shown in figure 4.3. The form then allows reordering the components and adding additional ones at the end. This approach is based on findings in [52] and [53], which describe the advantages of a prompt and template based interrogation in educational contexts.

Due to the simplified argument model used in this thesis, the form requires to specify exactly one claim, and one or more premises. At the current stage, these are the only component types that are present in the game, since they represent the most important ones in Argumentation Mining. Future iterations may cover additional types, such as *backing* components. Within the game user interface, the premises are called *reasons* instead, which has the same meaning, but sounds more familiar and less

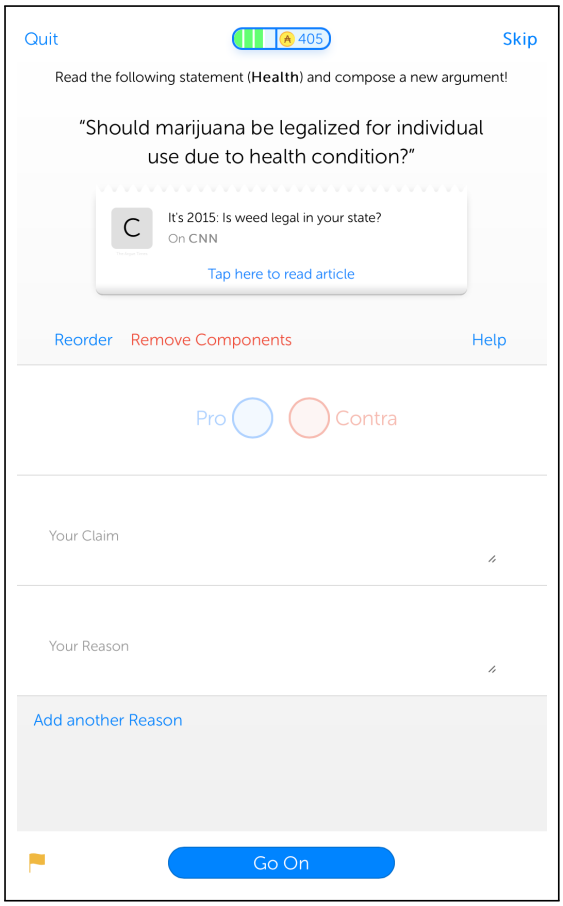


Figure 4.3: The Compose an Argument game round. The users need to specify if their new argument is either a Pro or a Contra argument. Next, they need to type a claim and one or more premises. The form also allows to reorder the components.

technical to the users. This semi-structured template based form is required to ensure that the argument is annotated right from the beginning at least in terms of the ranges of its components.

Surely this constrained kind of input probably will not match the users' expectations. However it can be justified by the fact that thinking about which of the user's thoughts belong into which type of argument component contributes to a clearer understanding of what the components semantically mean, and for the sake of the argument quality, why a distinction actually is required.

Apart from that, the constrains may carry the risk, that the quality, at the very least, the genuineness of the argument could suffer. The template may lead to an unnatural argument consisting of a chain of sentences sticked together, instead of a well written unit with a natural flow. In future implementations therefore an alternative template-less game round could be added, in which the users can freely compose an argument on their own, and, in another game round type, let the other users determine the argument component spans.

Contribution

By creating an argument, the authoring user specifies several of its attributes right from the beginning, such as the mentioned components and their types, the stance and the topic, which the argument relates to. So next to the actual argument, the author also provides additional meta-data for it, which are called *annotations*. These however can be false, either due to nescience, to laziness or to willful malignity. Whatever might be the reason, it is important to recognize that it constitutes the main cause for the necessity of mutual validation amongst the users. The engagement of other users is needed, who analyze and assess the author's annotations and as a consequence correct them, or sort out completely invaluable instances.

Validation

Before the users submit the new argument, the input will be validated rudimentarily by applying simple metrics: One of the two possible stances must be selected. Next, each of the provided components must have a minimum length of characters. Furthermore, to avoid having multiple sentences in one component, the game will ask to split the sentences into multiple components, in the case that a full stop, a question or exclamation mark is being detected. And finally, the constrained input form provides a basic validation from the very beginning which ensures that there is one claim and at least one premise contained.

Solution

In contrast to other game rounds, the Compose an Argument game round does not reveal any solution, since it is an opened-ended task, which can not be done right or wrong, as long as it passed the validation.

Rewarding

Because of the amount of time and thinking the users have to invest in the completion of the Compose an Argument game round, and because of the value they contribute to harvesting arguments, it is rewarded by a fairly large amount of points. In contrast to other game rounds, the generated response data by the users can not be verified. Unless the argument components consists of completely nonsense text or spam, an argument – at least from the perspective of this thesis – can not be true or false. Each correctly annotated argument, either by the author, or by the mutual validation loop, is a valuable instance for the growth of the corpus. Even fallacies can be of worth, since their detection and recognition is also being studied in current research.

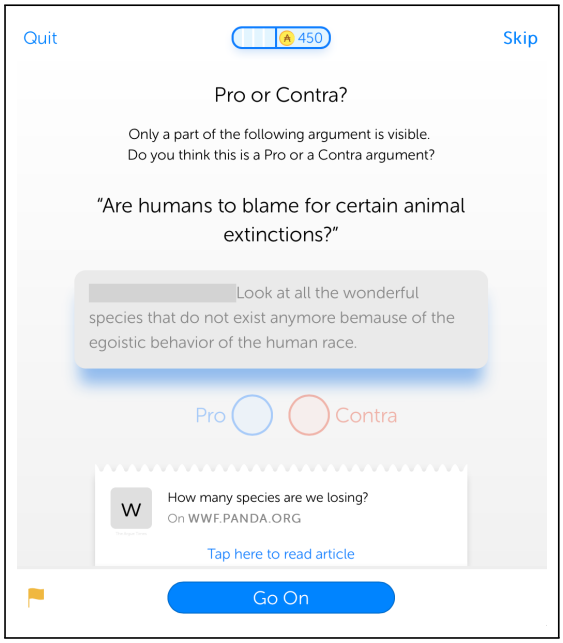


Figure 4.4: The Stance Recognition game round. Only one of the argument components is visible. Based on that, the users need to guess if the argument is a Pro or a Contra argument.

4.5.2 Game Rounds to Analyze Arguments

The game rounds that belong to the second class are characterized by the users' analyzing output they produce. These game rounds do not contribute to the size of the corpus, but only to the validity of it. Technically, the gameplay allows any arbitrary attribute of the arguments to be assessed by the users. At the current stage however, only the stance and the component types are being considered.

Stance Recognition

The first game round whose response data embodies an analysis of existing data, is the *Stance Recognition*. In this game round the users have to guess the stance of the shown argument. Accordingly, the game data consists of an argument, the corresponding topic, the domain, and optionally an article. To increase the challenging aspect, only one of the components of the arguments is visible, while the rest remains hidden. Otherwise this game round could be quickly perceived as too easy, and therefore, boring.

Contribution

The contribution of this type of game round is the validation of the stance of a given argument. Although the author needs to specify it upon composing, it still may be invalid. Therefore, there is by all means a justification for this game round. Beyond that, due to its low challenge, this game round additionally can act as a further reward, and therefore, a motivation, because the users over time are expected to associate their success in this game round with positive feelings.

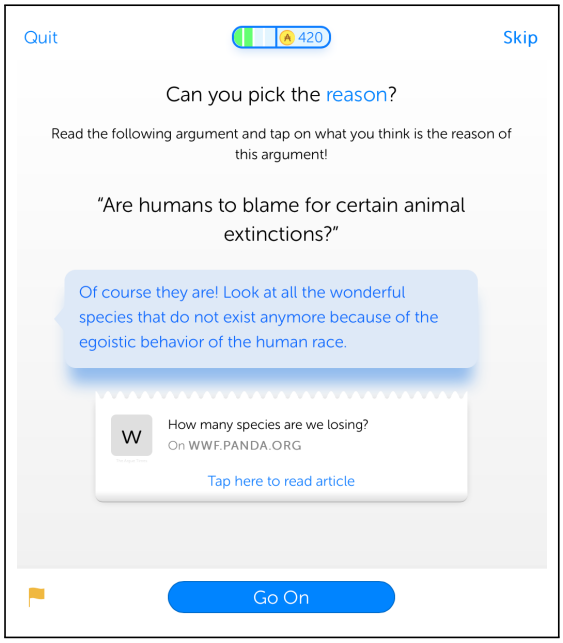


Figure 4.5: The Pick the Component game round. The users need to select all components that match the specified component type.

Validation

To provide their input, the users can either choose between pro or contra. The only required validation is to check if the user has chosen any of the two options at all.

Solution

The game reveals the solution simply by finishing with a message, which signifies either success or failure, after the user's response data submission.

Rewarding

The Stance Recognition game round certainly is the easiest of all rounds, and therefore, the rewarding is adequately small. This game round is considered to be a good opportunity to allow the users to prove their diligence by repeatedly playing the round in order to increase their amount of collected points, and therefore, improve their ranking.

Pick the Component

The *Pick the Component* game round is a more challenging game round to analyze arguments. Here, similarly to the Stance Recognition game round, an argument is presented to the users, including all relevant context information as previously mentioned. In this game round, each of the components is visible, though. The users' task is to select all of the individual components, which they think belong to a certain pre-defined component type, that is, either claim or premise. All remaining components should stay untouched. Note that again on the front-end, the premises are called reasons.

Contribution

By playing this game round, the users will submit many individual opinions about each of the argument component types. These opinions will later be used in the attempts to determine the true component types.

Validation

In this case, validation is trivial: The users' response data is valid, if at least one component has been selected.

Solution

In order to educate the user, after having submitted the response data, the game will present the correct selection, by highlighting the expected selection in green, and all of the user's wrongly chosen components in red color. This allows to the users to rethink their choice in the case that they were wrong.

Rewarding

Selecting the correct components is not a primitive task, since possible ambiguities or unclearness may imply difficulties in the classification task.

In the case that the claim needs to be highlighted by the users, only one component needs to be picked, which probably implies a lower error rate, than picking all, but only, premises. For example, the users' choice is wrong even if they have highlighted three of the four premise components of an argument. Accordingly, due to the increased cognitive overhead, the reward for this game round is moderately high, but still not as high as for the Compose an Argument game round.

4.5.3 Other Types of Game Rounds

In some situations while playing the prototypes, it has been noticed that game rounds alone are not enough to explain the gameplay to the users, and as promised, the fundamental aspects of good argumentation. Initially each of the game rounds were designed so that they can be viewed and played as isolated units and that they always contained every explanation necessary to fully understand what to do, but also what the learning aspects of this respective game round is. For example, the Pick the Components game round would explain which different types of components an argument usually consists of.

During the iterations of the game design it quickly has been noticed though, that on one hand the game rounds visually do not offer sufficient space to host both the explanations and the game round itself at the same time in one view. Instead, both of these two substantial parts should be placed in a dedicated frame. The user should be allowed to thoroughly read the educational aspects without being disturbed by the challenging, and eye-catching elements of the current game round, such as, e.g., a countdown timer. These static educational contents should not be related to any specifics of the game round gameplay at all, but be separated from them. On the other hand, the users will learn how the game rounds work very quickly, hence, presenting repeating explanations would be tedious and negatively influence the fun.

Therefore, a third type of game rounds dedicated to displaying static contents has been added, which can be integrated in between the normal game rounds to introduce the users to what they are about to learn. While testing, it became obvious that this clearly was the right choice to have a slow, painless introduction to the game at the beginning, and a more challenging speed in later levels.

4.6 Further Motivating Gameplay Elements

The different game rounds are challenging on their own, but it has been noticed that additional gameplay elements can be integrated, which provide further incentives to feel challenged and to be motivated to keep playing.

Countdown

A countdown is a simple game element to further put the users under pressure, while being cognitively urged by the current task. The countdown has been integrated as a simple visual element as part of the game rounds, that is, a visual box that shrinks until the countdown has timed out. However, the countdown has no impact on the rewarding or the success of a game round, and their only purpose is to provide a sportive element.

Future implementations could further reward the users in the case that they manage to solve the task within the predefined time interval. Currently, the two game rounds of the analyzing class both make use of the countdown element, with custom defined durations each. The Compose an Argument game round obviously should not urge the player, since the quality of the resulting argument could suffer.

Sound Effects

In some cases, and if the platform the game is running on technically allows it, the visual feedback by the system for the users' actions is additionally backed by short sound effects, which aim to embody the feedback acoustically. For example, the feedback, that the user's response has been correct, is accentuated by a positive triad sound. Analogously, an incorrect answer is accompanied by a darker, more negative sound. These effects, in particular the positive ones, further reward the users for their actions by calling up positive emotions.

Success Notifications

Improving the high-score ranking is a positive moment while playing the game and is therefore notified to the users: Whenever they have just improved their ranking by gaining points, a shiny overlay appears, that celebrates the success. In this way, users are more likely to visit the high-score every once in a while to compare their ranking with others.

4.7 User Retention Methods

The game cognitively demands a lot from the players. In the Argumentation Theory oriented scope of the game setting, establishing good retention rates for the game, that is, to keep the users coming back to the game on a regularly basis, is certainly more challenging, than in usual casual games which solely

focus on amusement. Therefore, all sorts of annoyances had to be reduced to a minimum, which could easily be identified while playing the first few prototypes. The following enhancements were made to minimize frustration and annoyances.

Skipping Game Rounds

Unless the superordinate level is configured differently, each contained game round can be skipped by the users. Some game rounds, such as the Composing an Argument game round, can be perceived as exhaustive, or the users may be in a situation where typing is cumbersome. Therefore, skipping game rounds is a reasonable solution to discourage the users from actually leaving the game immediately. However each Skip action is figured with a certain amount of costs: The user need to pay with their collected coins for performing the Skip, and if they can not afford it, skipping is not allowed. The costs are individually defined by each game round, and are usually estimated at 60% of the amount of coins the users would earn by successfully completing the game round.

Quitting a Level

The users can decide to quit the current level to return back to the respective world. Quitting a level will retain all of the collected points within this level so far and is not bound to any costs. Similarly to skipping game rounds, each level however can be configured to disallow quitting a level. Users might want to quit and restart the current level to receive differently chosen random game data, or to have a look again at a certain game round where they missed to succeed.

Registration-less Gameplay

Registration can easily form an obstacle for many visitors of the game. Therefore, it has been designed to allow playing the game for a while without having to register before. The game asks the visitor to register for an account during their first visit, but also lets them skip the registration easily. However, to motivate the users to eventually register after the first progress they made in the game, a popover notifies them that their progress might get lost if they choose to not register now, including all other advantages of registering.

Registration is important for several reasons: Users feel more responsible for what they do if all of their actions are bound to their virtual identity, and thus, the quality of their generated data is expected to be better than what visitors, that is, non-registered users, produce. Next, it is only possible to a certain extent to recognize a previous visitor again in a second gameplay session, and fairly error prone. Their game progress will be lost and therefore, their reward and rankings. Visitors, that provide many valuable data to our corpus over time, should be motivated by good high-score rankings, and in possible future iterations, by good reputation scores, to stay.

4.8 Chapter Summary

In this chapter, the design of the game has been described in detail. First of all, the art of game design has been highlighted and casual games have been explained. Next, the individual sub-components of the game, the motivating game elements and the methods to reduce frustration due to the particular nature of the game have been described.

The game consists of game rounds, which are short tasks for the users encapsulated in a game setting, and in which as a result of playing them, the users either compose new data or analyze existing data. Composing data enhances the size of the corpus of arguments, whereas analyzing data improves the quality of the corpus by correcting wrongly annotated data attributes based on a mutual validation mechanism.

Multiple game rounds are chained to form a game level. A game world is a set of game levels, which need to be completed, before the game world will be marked as done. The two main goals for the users are to complete all game worlds, but also to improve their ranking in the global high-score list. This list is based on the points which the users collect over time by completing game rounds. The following chapter will explain the design of the game as a software in detail.

5 Implementation

This chapter provides an in-depth explanation for the implementation of the developed game, which, viewed as a software, can be separated into several sub-projects. Since the game will run on different devices, and the data, that is generated while playing the game, must be centrally stored, a conventional client-server oriented architecture seemed like a reasonable choice. Therefore, the sub-projects consist of the *client application*, which is the actual software that the users will be in immediate contact with, the *server application*, also called *back-end*, that feeds the client with game data and manages users, assets, or tracking information, and an administrative *front-end*, that allows to create and manage the game data and all user-generated contents.

5.1 The Client

Clearly, the client software of the game is the most important part of the software as a whole. The client embodies the interface between the users and the software, and provides what makes it a fun game. The goal to deliver an entertaining game software, which shall be accessible from all major platforms, poses a lot of challenging requirements to the design and its implementation. Some of the main software requirements, next to the previously described game design, are the following ones.

Cross-Platform Compatibility

Users must be able to play the game on all of the modern, popular main consumer internet platforms, except gaming consoles and televisions, which include the desktop computer, the tablet and the smartphone. This requirement is important to ensure a maximum potential reach of the application, to allow any internet user to play the game at any time. Despite the different screen sizes of the devices, the game must run properly and be accessible on each of the different configurations.

User Management

Users must be able to create an account for themselves, where user related information, such as the game progress and preferences, are stored. Every created data entity must be linked to the account of the creating user.

Modularity

The game must be built to be easily extendable in future implementations. In particular, creating new types of game rounds or levels and configuring the game contents must be feasible without knowing the implementation of the application logic in detail.

User Tracking

The activities of the anonymized users must be traceable. These activities are recorded as simple atomic events such as *Opened Application*, *Successful Authentication* or *Skip Game Round*. Tracking this information is important to reveal bottlenecks in the game flow, that lead to, for example, declined

retention rates or participation. The results can be used in future to improve the game, and in particular, the game design.

5.1.1 Technology Decisions

For economic reasons, the client side has been developed using the *HTML5* technology stack, which consists of the three languages Hypertext Markup Language (HTML), a XML based format, *JavaScript*, a scripting language, that runs in the browser, and Cascading Stylesheets (CSS), a language to create styling rules, that define the look of elements within a web page. HTML5 is supported by all modern browsers and provides all required capabilities for rich web applications, such as games.

Compared to natively compiled applications built in for example *C* or *C++*, HTML5 as a platform for software has some major drawbacks: Since JavaScript code is being interpreted whenever the browser downloads the corresponding web page, HTML5 based applications currently do not reach the performance of native applications, which in situations of extensive screen animations or data loading operations influences the user experience. Next, HTML5 based applications that run in the browser usually only have very limited access to low-level system operations, such as file or hardware access, and are constrained by fairly strict security measures.

On the other hand what makes HTML5 a great choice for applications such as the developed game in this thesis, is first of all the spread of the platform it runs on, which is the browser. Every modern computer and every handheld device today comes with a sufficiently modern browser pre-installed, and therefore, can run HTML5 based applications without any further software installation. This is a huge advantage in times of platform fragmentation (cf. [54]), and operation system wars. Next, the development costs can be drastically reduced not alone because only one implementation that covers all device families is needed, but also because of the easy-to-learn and very high-level stack of languages in the HTML5 development environment. So for the use case of this project, HTML5 obviously meets all of the requirements for rapidly building a well-running game, that is accessible on all important platforms.

Ionic Framework

As previously examined, the combination of languages bound to HTML5 generally provide everything required to build rich-featured applications that run in the browser. With the rise of HTML5 as a platform, several frameworks have been developed, that attempt to accelerate the development and reduce the costs by providing commonly used components out of the box. For the developed game, it has been decided to choose the *Ionic Framework (Ionic) (MIT License, maintained by Drifty Co.)*[1], which is an open-source framework for HTML5 applications. Ionic comes with both visual building blocks and application logic fragments, but also provides a Command Line Interface (CLI) that acts as a handy tool during project instantiation, development and deployment.

Ionic is built on top of *Cordova (Apache License 2.0, Apache Foundation)*[2] and *AngularJS (MIT License, maintained by Google, Inc.)*[3], which are both established open-source projects for HTML5 based application development, backed by a vivid community. AngularJS supplies a Model View Controller (MVC) architecture and adds lots of helpful extensions to HTML and JavaScript, including new Document Ob-

[1] http://ionicframework.com/
[2] http://cordova.apache.org/
[3] https://angularjs.org/

ject Model (DOM) node specifications and JavaScript helper functions. Cordova on the other hand is a bundle of tools and Application Programming Interfaces (APIs) , that attempt to reduce the drawback of web applications in terms of their limited access to system functionalities. Therefore, Cordova provides a lightweight native wrapper application for each of the supported platforms, including *iOS (Apple, Inc.)*, *Android (Google, Inc.)*, *Windows Phone (Microsoft, Inc.)*, general desktop browsers, and more. The wrapper application basically acts as a browser for the HTML document of the application, which provides access to system functions via JavaScript based APIs, such as camera or file access. Thereby, the developer is enabled to build one single web application, that makes use of a shared API on all supported platforms.

5.1.2 Client Architecture

The Ionic Framework, and AngularJS in particular, provides a reasonable way to structure the application into exchangeable modules. The application logic is separated into many controllers, that handle specific user input and game output, and communicate with the back-end. In particular, back-end data exchange is encapsulated in so called *Stores*, for example, *TopicStore* or *ArgumentStore*, which allow fetching data from the back-end, but also creating, publishing and linking data, whereas creating means saving new entity instances, publishing means unlocking the content for everyone, and linking means associating two or more entities.

One particularly important aspect has been the separation of *application logic* from *game content*. The client must act as a platform for different types of game rounds for future extensions. While the current implementation provides three types of game rounds, future versions might add additional ones, via which different types of data could be collected. The controlling unit of each game round instance and each level or world instance is a dedicated controller, and each of them may require specific resources, such as image assets. These game content related files reside in a dedicated directory, which is separated from the rest of the application logic file hierarchy. Figure 5.1 illustrates the separation of all types of files belonging to the client implementation into application logic and game content.

Next to the explicit separation of files, there is a JavaScript Object Notation (JSON) based remote game configuration file, that defines which world, levels and game rounds are presented within the game. Upon application start, this configuration is always freshly being downloaded from the back-end. For situations with a loss of connectivity, a local game configuration, that is bundled with the game content assets, acts as a fallback. The remote game configuration can be used to control which contents are currently being played by the user base, and therefore, acts as a tool to control the generation and assessment of data. Within the file, specific domains, topics and arguments can be specified. Altogether, the game configuration file and the separated game content file system will easily allow to extend the game in future.

5.1.3 Deployment

To deploy and distribute the application, the underlying Cordova stack lightened the workload significantly. During deployment, Cordova is bundling all required resources, minifying the JavaScript code to optimize the loading performance of the application, and thereby, also obfuscating the code, which would otherwise be easily accessible and prone to potential malicious attacks. The app bundle then will be uploaded to an ordinary HTTP server with no further technical requirements.

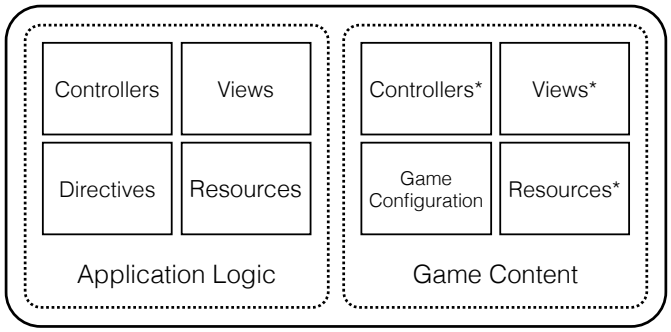


Figure 5.1: The separation of the client implementation into application logic and game content. Sets marked with * are specifically related to game worlds, levels or game rounds and are meant to be easily exchangeable while treating the application logic untouched.

5.2 The Server

The server-side implementation of the game, which is also called *back-end*, is a software that steadily runs on a publicly accessible computer. This software acts as a central endpoint for the client software and provides data to, and consumes data coming from the users. Because of its role, there are certain particularly important requirements, which the back-end must meet, which are explained next.

Public API

The back-end must permanently provide an API which allows accessing the data and storing data from clients applications.

Security

The back-end must have restricted access to ensure that only authenticated and privileged users can access or save data.

5.2.1 Technology Decisions

Due to the constrained time limits of the thesis, developing a fully fledged back-end software from scratch would have been no option, as it would not contribute to the research goal of the thesis. There are many existing frameworks which partly offer what is required for this project, but even bundling existing frameworks and taking care of a proper integration of these would imply too much overhead. Recently, a trend to offering a so called *Back-end as a Service (BaaS)* has emerged from the rapidly growing mobile application market, so that the developers are enabled to solely focus on the client side, and skip the back-end implementation.

As an alternative to commercial services, one open-source project called *BaasBox (Apache License Version 2.0, maintained by BaasBox S.R.L)*[4] has recently been founded. While still in a pre-release version, this bundle is offering everything required for this project. So for the back-end, the open-source software BaasBox has been chosen. The Java based software aims to facilitate the development of (mobile) applications by providing commonly demanded functionalities out of the box, such as user, friends and asset management, a database and a *Representational State Transfer (REST)* based API. Next to the server application, which can be deployed on Unix and Windows systems by starting a simple Java process, BaasBox also provides client Software Development Kits (SDKs) for iOS, Android and JavaScript. Furthermore, the server API can be extended by custom plug-ins, and therefore, BaasBox covers all of the requirements of the game software.

There is an administrative interface for the BaasBox instance, called the *Console*, which is accessible via any browser. The Console allows to configure the BaasBox, manage the users, assets, plug-ins and the database. Furthermore, the Console provides functionalities to create and restore from back-ups, which is an important feature due to value of the corpus that will be stored in the database.

5.2.2 Server Architecture

Similarly to Structured Query Language (SQL) based databases, the embedded *OrientDB* database instance, that is part of BaasBox, can store entities, that is, abstract model descriptions, that are used by the application to represent state. In BaasBox, these entities are called *Documents*, and each Document belongs to one *Collection*, which is similar to the *Rows* and *Tables* based approach in SQL. Documents are stored as JSON documents, and therefore, can hold boolean, string, and numeric data. For the project, the Collections listed in Appendix B are used.

The database furthermore allows to link entities. A link between two entities is modeled as a simple document, that stores a *link label*, an *in* and an *out* object reference, whereas the *out* object is linked to the *in* object. There are only a few types of links currently being used in the database, which are listed in Appendix C.

To let the clients store data, such as newly composed arguments, several BaasBox *plug-ins* have been developed, that store new entities in the server database, make them publicly accessible and create the links between the database entities, for example, between an argument and the topic it is referencing. Performing these tasks on the server side also improves security against malicious attacks.

5.2.3 Deployment

To deploy the server as a software, a computer with a permanent connection to the internet has been used, that has a Java runtime environment available. The computer is publicly addressable, and thus, reachable by the clients. To secure the communication between server and clients, a Secure Sockets Layer (SSL) certificate is used, so that Hypertest Transfer Protocol Secure (HTTPS) based data exchange is possible.

[4] `http://www.baasbox.com/`

5.3 Voting System

As explained in chapter 4, a voting system is an integral part of the game, in order put a loop of data creation and validation into operation. Generally, there are types of game rounds, in which users compose data, whereas in other game rounds, they assess such data. The voting system is fairly simple, yet powerful enough to improve the quality of the corpus.

Whenever an assessing game round has finished, such as Pick the Component, one single *vote* will be submitted from the client to the voting system, that resides on the server. This vote relates to one specific *attribute* of an *entity* and contains the *value* for this attribute which the *voter*, that is the user, that submitted the vote, assumes the attribute should have.

When a newly composed entity, for example an argument, has been submitted, the author automatically also submits one vote for each of the assessable attributes of the entity, according to what input the author provided. So in case of an argument, the software also makes sure to store a vote for the stance of the argument, and a vote of the type of each of the argument components.

All entities in the database are stored as JSON documents, which are tree data structures. The state of an entity, that is, its properties and values, are stored in the leaves of the tree structure, and every leaf can be reached by a unique key-path beginning at the root of the JSON entity down to the leaf. As an example, the following pseudo-code retrieves by a key-path the *type* attribute of the first premise of an argument consisting of one claim and two subsequent premises:

```
keyPath  = "components.1.type";
argument = {
  "stance": "pro",
  "components": [
    {"type":"claim",    "body":"Cellphones should be banned from class rooms."},
    {"type":"premise",  "body":"I've seen it in schools: the pupils are [...]"},
    {"type":"premise",  "body":"The only one who should be able to use a [...]"},

  ]
};
valueForKeyPath(argument, keyPath); // 'premise'
```

Each vote contains the two data properties *keyPath* and *value*. The *keyPath* is the key path of the attribute, that has been assessed by this vote, beginning at the root of the object whose attribute has been assessed. Consequently, the *value* is the value which the user thinks the referenced attribute should contain.

So if a user in a Stance Recognition game round for example is being asked whether a shown argument is a *Pro* or a *Contra* argument, and *Pro* has been chosen, a vote modeled by the following JSON object will be submitted:

```
{ "keyPath": "stance", "value": "pro" }
```

Similarly, when a user is being asked to pick the premises of a shown argument, for each of the picked components (in this case, the components array entries with indexes 1 and 2), the following votes will be submitted:

```
{ "keyPath": "components.1.type", "value": "premise" },
{ "keyPath": "components.2.type", "value": "premise" }
```

Voting Distributions

A *voting distribution* is the statistical evaluation of the set of all votes regarding one specific entity attribute. Thus, a voting distribution represents the current outcome of the crowd-sourced voting for the respective attribute. For each the of different voted values contained in the votes for a specific object attribute, one corresponding key is inserted into the resulting voting distribution. The value for the key is the relative amount of the votes for this value.

As an example, based on the following set of five votes for the *Stance* of an argument, the two keys *Pro* and *Contra* will be inserted into the voting distribution:

```
pro, pro, contra, pro, contra
```

Since three of the five votes vote for a *Pro* value, the voting distribution will be the following:

```
{
  "pro": 0.6,
  "contra": 0.4
}
```

Independent Confidence Value

Two additional values are part of every voting distribution: The *independent confidence* α and the *confidence* value α'. Both values indicate how informative and reliable the distribution is, and therefore, how it should be handled by the game mechanics. There, the confidence value is being examined to decide if game data is verifiable or not, and accordingly, if a user response can be judged precisely, or if only a *soft feedback* can be provided instead.

The independent confidence is an indicator for how obvious and clear the outcome of the voting currently is, and thus, is a function of all votes for the respective key-path. A value of 1 means that the voting altogether is very clear and unambiguous, that is, the crowd has fully decided for one, and only one, of the options. A value of 0 on the other hand means that the crowd has not determined a preferred value at all.

Let V be the set of all values within a voting distribution for an attribute of a specific entity. As illustrated in the following formula, the independent confidence is calculated by subtracting from the maximum value the average of all remaining values of V, that are part of a voting distribution. This means that the independent confidence equals the difference between the biggest value and the average of all other values:

$$\text{Independent Confidence } \alpha = \begin{cases} \max(V) - \frac{\sum V - \max(V)}{|V|-1}, & \text{if } |V| > 1 \\ 1, & \text{otherwise} \end{cases} \tag{5.1}$$

In the voting distribution example from above with the two values 0.6 and 0.4, the independent confidence α would be equal to $0.6 - 0.4 = 0.2$. By comparing this value to a minimum threshold, for example 0.6, this value would indicate that the crowd has not agreed yet on the *Stance* attribute of the argument. In contrast, the following distribution would have an independent confidence value of $0.95 - 0.05 = 0.9$:

```
{
  "pro": 0.95,
  "contra": 0.05
}
```

Compared to inter-rater reliability degrees, such as *Fleiss' Kappa* [55], which also take into account the potential agreement between the raters by chance, the independent confidence is fairly simply calculated. However, it is still assumed to be robust enough for this use case. Future iterations may require further advancements of the model, or use Fleiss' Kappa instead, in the case that observations reveal, that its value is more informative.

Confidence Value

However, in the case of a lack of votes, the independent confidence value is not truly expressive. If for example the stance of an argument has only been assessed by one single user, the value will be 1, but the expressiveness of the voting in this state is by far not given. One single voter surely can not represent a reliable source for verification. To tackle this, the second of the additional statistical values, the *confidence*, is used.

This value is generally based on the value of the independent confidence, but in addition to it takes into account the number of votes so far. Therefore, a minimum number of required votes is specified as a constant k, that acts as a threshold up to which the voting distribution is assumed to be still unreliable. In the case that the number of votes exceeds the constant, the confidence value equals the independent confidence. Otherwise, the value equals the independent confidence, multiplied by a decreasing factor called λ:

$$\lambda = \frac{1}{1 + \max(0, k - |V|)} \tag{5.2}$$

$$\text{Confidence } \alpha' = \alpha \cdot \lambda \tag{5.3}$$

For example, if the minimum number of required votes k is set to 10, and only one vote has been generated so far, the independent confidence is 1.0, λ is 0.1, and the resulting confidence is $1.0 \cdot 0.1 = 0.1$. This adjusted value is far more reliable than the original independent confidence and therefore is being used instead within the verification of the user input in the game.

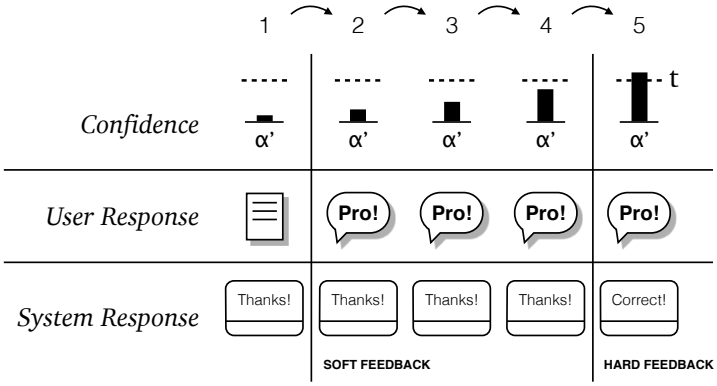


Figure 5.2: The simplified illustration of the development of the confidence level over time, in this case with regards to the stance attribute of an argument. In game round 1, an argument has been created and a first vote is added. In game rounds 2 to 4 additional votes for the entity to be a Pro argument are submitted, but the users receives soft feedback for their choice due to the lack of confidence yet. In game round 5, the confidence then exceeds the threshold due to the set of previous votes and accordingly, the user receives a hard feedback.

5.4 Providing Game Contents

The calculation of the voting distributions are performed server-side and the results are injected into the data that is sent to the clients. As long as the client requests an arbitrary argument, which it can display within a game round, at the current stage, a random argument entity is fetched from the database, which acts as the baseline implementation. Future iterations may choose specific entities based on the state of their voting distribution.

Within the game, the confidence level will be taken into account whenever the users provides a response in an assessing game round, that is, Pick the Component or Stance Recognition. A constant threshold t is defined in the game application logic, which the confidence value of a voting distribution will be compared with. A value of 0.65 for t has empirically been determined to be reasonable.

If the confidence exceeds the threshold, the maximum value contained in the voting distribution will be determined. This value also has to exceed another threshold, t', which is set to 0.75. In the case that both thresholds are exceeded, the the game will present hard feedback to the user, that is, it will confidently judge the user's response and provide a point reward accordingly. If the confidence value however is lower, only a soft feedback will be shown to the user. Figure 5.2 illustrates the development of the confidence level for the stance of a newly created argument and the adapting feedback type over time, in a simplified way.

For specific entities in the database, it may be known that their attributes are all correct. For the initial phase of the game, the game requires some bootstrapping data which can be presented as game data to the first users in order to ignite the gameplay and the mutual validation. This editorially injected data should not require any crowd-sourced validation, and therefore, an *isEditorial* flag for database entities can be set.

5.5 Administrative Front-End

All of the data that are used by the game, including arguments, topics and domains, can be managed by an administrative front-end, which is built on top of the same technology stack, as the client itself. The front-end allows to create new or modify existing data, mark data as editorial and hide specific data. Furthermore, the spam reports can be viewed, that have been generated by the users while playing the game.

Only the front-end is meant to be used to access the game data, since accessing the BaasBox Console directly to access or modify data, for example, individual arguments, bears the risk to damage the links between entities. Next to the administrative functions, the front-end also provides corpus exporting capabilities. Doing so will export all of the arguments, fully loaded including their referenced domain, topic and article, in JSON format.

5.6 Chapter Summary

In this chapter, details concerning the technical implementation of the developed game have been presented. The game as a software product has been separated into a client and a server-side implementation, which both have been described in terms of their requirements and the technology that has been used to build them. Furthermore, the voting system has been explained from the development perspective.

The voting system is the core of the mutual validation, that is part of the game design. Whenever the users provide their response for a game round in which data is assessed, they are submitting a vote to the voting system as a hidden side-effect. These votes embody the users' assumptions about the correct value of an attribute of an argument, such as for example the stance attribute, or the type of an argument component. All votes for a particular attribute of an argument will be considered and summarized by a voting distribution, which is injected into the data, whenever the argument is served to the clients. This distribution not only contains the different assumed values and their respective voting proportions, but also a confidence level, that indicates how obvious the current voting is, based on a statistical model.

The following chapter describes the evaluation of the developed game in detail, which is based on a user study and statistical tests performed on the resulting data.

6 Evaluation

This chapter describes the evaluation of the developed game in detail. Therefore, a justification for a statistical hypothesis test, details of the preparation and the conduction, and the final results are provided. Furthermore, additionally collected event data are analyzed. A conclusion summarizes the respective results.

Generally, traditional techniques to evaluate the usability of a software include standardized methods such as expert evaluation, field observation, interviews, and cognitive walkthroughs. Due to the particular characteristics of game software, these approaches however can only limitedly be applied on the developed game, and carry the risk that, when applied, their results may be misleading. To illustrate the mismatching characteristics of games and traditional desktop software, Korhonen et al. [56] explain that software should allow the users to work highly efficiently, while games on the other hand should provide the right exact amount of challenge to be fun.

The focus of the evaluation in this thesis is to evaluate the software with regards to the initial research question, which is whether a game can be used to crowd-source the creation and validation of an argument corpus.

What separates the developed game from the usual quiz games, which the game can roughly be compared to from certain perspectives, is the fact, that the system can not at all times provide precise feedback on whether the user's response is correct or not. In situations where the analyzed attribute of the presented game contents, which are generated by users, has not yet attained a sufficient amount of votes, the game can only present a *soft feedback*, which acknowledges the user's input by a reduced amount of gained points and a message of appreciation, but the game does not actually judge it.

Therefore, the crux of the matter is to investigate if a lack of *hard feedback* has a significant impact on the user's fun playing the game, and moreover, on the motivation to keep playing the game, which is crucial for the long-range objective to harvest a large corpus of arguments. This motivation also depends on the users' trust in the rating system of the game. In the following, all these aspects will be summarized by the *entertaining value* of the game.

6.1 User Study

The expectation is that there is no significant difference between the two groups of users, who either receive always hard feedback, and those who sometimes only receive soft feedback. To investigate this expectation, a user study has therefore been conducted.

For the participating users, the study consists of two parts: At first, they will play a pre-configured, dedicated variant of the game, and afterwards will have to answer a standardized questionnaire. The questionnaire aims to produce two statistically easily comparable, independent samples of the population and will be conducted online. To prove the initial expectation based on the interrogation data, multiple statistical hypothesis tests have been performed.

6.2 Statistical Hypothesis Testing

For the described evaluation, according to the usual methods to run a statistical hypothesis test, the hypothesis, the setting, the participant recruitment, the execution and the final evaluation will be described hereafter. Since two groups, that is, two samples of the populations, have been compared, a *between group study design* is applied. Next, since both groups are independent, because they did not interfere with each other and the members of the two groups have been assigned randomly without taking into account previous assignments, the conducted test can be categorized as a test of *independent samples* (cf. Heiman [57, p. 265]).

To capture the perceived entertaining value of the game, an interrogation model consisting of five statements has been provided in the questionnaire, which had to be judged by the participants and which are shown below. The outcome for each of the five statements has independently been evaluated statistically. Additional statements were part of the full questionnaire for further, non-statistical evaluation. The complete questionnaire is listed in Appendix D.

1. The game was fun in general.
2. I can imagine to play the game again once it is finished.
3. I can imagine to frequently play the game once it is finished.
4. I temporarily considered leaving the game.
5. I trust the rating system of the game.

The first four statements serve as a direct indicator for the entertaining value of the game for the interrogated user. The fifth statement affects the aspect of the points the users receive, and the feedback of the system to their responses, which also, but indirectly, influence the entertaining value.

6.2.1 Hypothesis

For the judgements of each of the statements, the *Null hypothesis* H_0 of the test is formulated as follows: There is no significant difference observable between two populations A, and B, where users of A always receive hard feedback, and users of B receive soft feedback in half of the cases. Mathematically the hypothesis is expressed by the validity of the equivalence of the two means μ_A and μ_B of the groups A and B:

$$H_0 : \mu_A = \mu_B$$

Consequently, the alternative hypothesis H_a is formulated as:

$$H_a : \mu_A \neq \mu_B$$

Before the evaluation to determine if the hypothesis H_0 should be either retained or rejected, the significance level α is set to .05, which means that a 5% chance is accepted to make a Type 1 error, that is, to reject H_0 even though it is true.

6.2.2 Study Design

Game Configuration

The participants were told right inside the game, before starting the first game round, that they are faced with a constrained game, that does not reflect the final game, for example, in terms of available levels, game goals, topic diversity and more.

This dedicated study game configuration consisted of one playable world, with only one level included, that contained five Stance Recognition game rounds, followed by one Compose an Argument game round and another set of five Stance Recognition game rounds. The game data of the game rounds have all been pre-defined and selected to be in English. Furthermore, all random aspects of the game rounds have been replaced by statically pre-defined selections.

This means that every participant has always been faced with exactly the same set of challenges, which ensures full commensurability among the resulting data samples. However, to avoid learning effects based on the sequence of the game rounds, which could negatively influence the validity of the study, the ordering of the contained Stance Recognition game rounds amongst themselves has been randomized per game visit.

User Group A

Unnoticeably for them, there were two groups of participants, A and B, to which every study invitation visitor has been randomly assigned upon accessing the online invitation for the first time. Depending on their assigned user group, the users have been faced with one of two variants A and B of the study game configuration. Both variants were exactly equal, but only differed in terms of the feedback the system provided: In game configuration A, the users always received hard feedback for all of the contained game rounds.

User Group B

Users of group B were faced with game configuration B, in which for a fixed, pre-defined set of game rounds, which in total accumulated to half of the Stance Recognition game rounds, the feedback was *soft* instead of *hard*, that is, the users did not receive a clear statement on whether their input was correct or not, and accordingly gained less points for correct answers, or more points for wrong answers, compared to users of group A. This is illustrated in figure 6.1.

Hence, the independent variable, that has differed between the two study groups, is the feedback of the system. So from outside, both user groups were faced with the same game, in terms of game data.

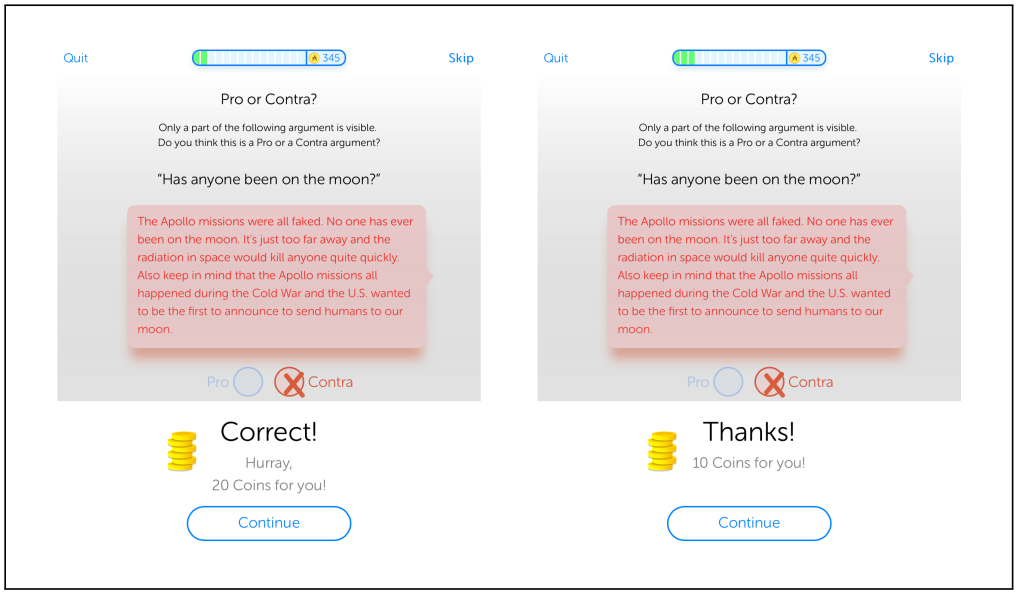


Figure 6.1: The same game round from the perspective of users of group A on the left, and users of group B on the right. Here, compared to users of group A, users of group B only receive a soft feedback and a reduced amount of points for the correct answer.

Questionnaire

Only after having played through the entire level defined by the study game configuration, the participants were asked from within the game to answer a questionnaire. For the interrogation, the web service *SurveyMonkey*[1] has been used. A questionnaire has been designed that asks a set of questions both directly related, but also unrelated to the hypothesis, to collect further data, which will be evaluated later. The complete flow beginning at the Unique Resource Link (URL) sent out to all potential participants, up to the final questionnaire is depicted in figure 6.2

All participants were faced with the same, standardized questionnaire. It contained statements the participants were asked to take up a stance to. Although such a standardized questionnaire involves the danger to not allow individual, personal input, and the given response options may potentially influence the interviewee, the benefit of being able to collect masses of interrogation data and all analyze them statistically at once, outweighed these drawbacks. Due to the estimated diversity of the nationalities of the participants, the questionnaire was presented both in German and English at the same time in order to not annoy users of different mother tongues and risking to lose them.

As illustrated in figure 6.3, which shows a screenshot of the questionnaire as seen by the participants, to express the agreement towards the statements, a *Likert scale* [58] has been provided, ranging from the values *1* to *5*, that is, from *I strongly agree* and *I agree* up to *I strongly disagree*. This non-continuous, but discrete Likert scale is an ideal interrogation method for this study: The uneven range of values is large enough to allow a thoughtful adjusted response, that can easily be re-interpreted. Furthermore, there is a value in the middle of all, that allows taking up a neutral stance. And from the perspective of analyzing the data afterwards, the Likert scale produces interval-scaled data, which is required for the applied statistical test techniques.

Additionally to the five values of the Likert scale, each statement could be answered with *Not specified*. This ensured, that people did not misuse the middle value in cases of uncertainty with regards to under-

[1] `http://www.surveymonkey.com/`

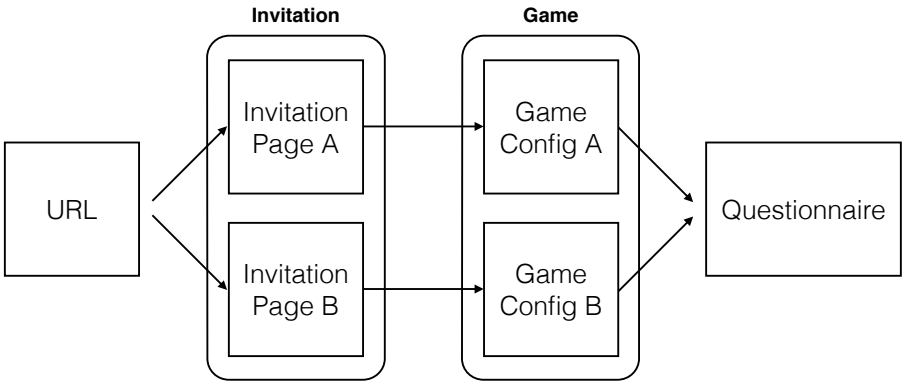


Figure 6.2: The flow from participant recruitment via the URL up to the final questionnaire from left to right. A participant is invited and randomly assigned to one of the two groups A or B. Accordingly, the participants will play game configuration A or B. Afterwards, all participants are redirected to the questionnaire.

standing the question, or annoyance by the question. Such responses are not taken into account in the evaluation of the data.

Recruitment and Execution

In order to have a simple mean to invite users to partake in the conducted study, a bi-lingual web page has been set up, containing an short introduction in German and English. Next to the introduction, this page randomly displayed one of two credential pairs, either *user31* or *user68*, including a simple password for each of the accounts, which the participants were asked to use to log in on the game web page. Based on the account they use, they were assigned either to group A or B.

To make sure that invitation visitors were always be assigned the same credential pair whenever they accessed the invitation page again, the random selection has been stored in a browser cookie, which has been read in subsequent web page reloads. The numeric suffix of the two available username attempts to obfuscate the assignment to one of the two groups of users.

Multiple channels have been exploited to recruit users for the study. Acquittances and research colleagues have been asked via an email containing a link to the invitation page to participate. The link also has been spread on Social Media channels, such as *Facebook* and *Twitter*. Furthermore, a list of locally close-by debating clubs and similar institutions have been invited by email to join the user study.

The invitations have been sent out for a period of 14 days. People could in total participate over a period of 19 days, which eventually led to the amount of 412 invitation page visits. No further actions were necessary until the end of the user study. In total, 37 filled questionnaires have been submitted, whereof 20 belong to group A users. The complete response data for the statements of the questionnaire are listed in Appendix E.

Argotario User Study

To finish the second and last step of the study, continue with the questions below!

Please answer the following questions unbiasedly and consider each of the provided options carefully.
/ Für die folgende Befragung würden wir Sie bitten, stets sämtliche Antwortmöglichkeiten abzuwägen und unbefangen zu antworten.

1. On a scale from 1 to 5, how would you rate the following questions?
Auf einer Skala von 1 bis 5, wie würden Sie die folgenden Fragen beantworten?

	I strongly agree *Stimme voll zu*	I agree *Stimme zu*	Neither agree nor disagree *Teils teils*	I disagree *Stimme nicht zu*	I strongly disagree *Stimme gar nicht zu*	not specified *keine Angabe*
The game was fun in general. *Das Spiel hat mir insgesamt Spaß bereitet.*	○	○	○	○	○	○
I can imagine to play the game again once it is finished. *Ich kann mir vorstellen, das Spiel im fertigen Zustand erneut zu spielen.*	○	○	○	○	○	○
I can imagine to frequently play the game once it is finished. *Ich kann mir vorstellen, das Spiel im fertigen Zustand regelmäßig zu spielen.*	○	○	○	○	○	○
I trust the rating system of the game. *Ich habe Vertrauen in das Bewertungssystem des Spiels.*	○	○	○	○	○	○
I temporarily considered leaving the game. *Ich hatte zwischenzeitlich überlegt, das Spiel vorzeitig zu verlassen.*	○	○	○	○	○	○
I consider myself to be an experienced user of online offerings. *Ich halte mich für einen erfahrenen Nutzer von Online-Angeboten.*	○	○	○	○	○	○
I could imagine to play this game on a smartphone. *Ich könnte mir vorstellen, das Spiel auf einem Smartphone zu spielen.*	○	○	○	○	○	○
In the finished game I would like to be able to compare myself with acquaintances and friends. *Ich würde mich gerne im fertigen Spiel mit Bekannten und Freunden vergleichen können.*	○	○	○	○	○	○

Figure 6.3: An excerpt of the questionnaire. Only some of the shown statements are directly related to the user study. The remaining statements have been part of the questionnaire for further interrogation.

6.2.3 Results

The amount of questionnaire response data has been considered as sufficient to run statistical tests in order to compare the two groups for each of the mentioned statements. The tests should detect if there is a real difference between the entertaining value of the two groups, or if deviations are caused by chance.

Per-Protocol Analysis

Despite the fact that slightly more users in group B actually started the game (43), compared to group A (40), there have been fewer questionnaire samples generated by group B (17), than by group A (20), which could also be regarded as a possible indicator for a reduced entertaining value of users in group B. Nonetheless, the analysis of the collected questionnaire data has been performed as a *per-protocol analysis*, that is, the following statistical evaluation is based on the questionnaire responses only, and is unaware of the users who have aborted their participation before submitting a filled questionnaire.

Generally, a per-protocol analysis bears the risk, as described by Montori and Guyatt [59], to lose valuable interrogation data of users, who were annoyed by the impact of their respective group setting. There are two reasons why it has still been decided to run the analysis as per-protocol: First of all, users who have aborted the study before submitting a filled questionnaire could not be reached by email or any other mean for further interviewing, and so out of necessity they could not be taken into account.

Furthermore, it has been observed by analyzing the amount of quit events per group, that a large amount of users have quit the game – and therefore the participation – within the first three Stance Recognition game rounds and the Compose an Argument game round. On one hand, the lack of feedback within the first three game rounds is assumed to be not influential enough to make the users quit. On the other hand, the Compose an Argument game round is completely independent of the changed variable of the study. Accordingly, it is assumed that the reason for the higher amount of quit events of users of group B is either independent of the changed variable, or caused by chance.

t Test

Generally, to determine whether the mean values of two populations differ significantly, the *independent t test* is a commonly used technique. More precisely, this test allows to determine if the difference of means of two independent groups or populations is in fact a real difference, or if the difference is only caused by chance, for example, due to measuring errors.

However, the t test poses multiple requirements to the data it is performed on. At least one of these requirements is not met for most of the data resulting from the questionnaires: The investigated feature of the population must be normally distributed (cf. Rasch et al. [60]). For each of the statement response data however, the two samples are in fact not normally distributed, which has been determined with a Shapiro-Wilk Normality test [61]. The results are listed in Appendix F. As an example, figure 6.4 depicts the two samples for the statement *The game was fun in general*, which visually signifies that the data is not normally distributed. Although there are opinions about the still given robustness of the t test, if the normality of the data is violated (cf. Rasch et al. [60]), the evaluation has been performed using a different statistical method, of which the data requirements are met.

Figure 6.4: Responses concerning *The game was fun in general.* 20 responses in group A, 16 in group B.

Mann-Whitney-U Test

Therefore, the *Mann-Whitney-U test* [62], has been chosen, which is a parametric-free statistical test with, from the perspective of this evaluation, the same goals as the t test. The clear benefit of this test, is that it does not require the samples to be normally distributed, but only to be ordinal scaled, and still manages to reliably detect significant differences between two populations. Since Likert scale constrained responses naturally are ordinal scaled, the requirement is met.

For each of the five statements contained in the questionnaire, a Mann-Whitney-U test has been performed. The results are listed in table 6.1. No significant difference has been revealed by all of the performed tests, and therefore, the hypothesis H_0 must be retained for each of the statements.

Metric	**Statement 1**	**2**	**3**	**4**	**5**
n_A	20	20	19	19	19
n_B	16	15	16	15	16
p	0.33	0.73	0.55	0.52	0.72
Significant	No	No	No	No	No

Table 6.1: Results of the Mann-Whitney-U test for the statements 1 to 5. n is the number of responses per statement per group. U* is the respective Critical Value. For two samples to be statistically significantly different, p must be smaller than the significance level $\alpha = .05$.

6.2.4 Conclusion

Based on the evaluation, it has been shown that no significant difference could be attested between the two groups A and B in terms of the entertaining value of the game. Therefore, the initial Null hypothesis H_0 can be accepted: The lack of *hard feedback* has no significant impact on the user's fun

playing the game, and moreover, on the motivation to keep playing the game, at least according to the set of statements, that have been designed.

This outcome in fact is considered to be more important than the actual result of the statement assessment by the users: Based on this evidence, the fundamental question if the way the user-generated content – with all its risks and quality aspects – is treated, harms the gameplay, can be assumed to be resolved. And based on that, the actual game design, including all of its entertaining aspects, can be refined in future iterations.

However it is important to note, that the overall outcome of the statement assessment by the participants has been fairly positive.

6.3 Additional Questionnaire based Interrogation

The questionnaire provided the chance to seek more feedback that goes beyond the evaluation of the user study, but still embodies valuable input. The questionnaire contained demographic questions and allowed the participants to provide freely written feedback. Furthermore, additional statements were asked to be judged:

1. I could imagine to play this game on a smartphone.
2. In the finished game I would like to be able to compare myself with acquaintances and friends.
3. If you could change one thing in the game, what would it be, and why?

Evaluation

The responses to these questions and statements have not been evaluated with regards to differences between the two groups. The results for the first two statements are depicted in figures 6.5 and 6.6. Although the noteworthy negative responses should not be neglected, in general the response with regards to these two statements is fairly positive, which confirms many of the decisions taken to build a cross-platform application with social multiplayer game elements, such as high-score lists.

The question *If you could change one thing in the game, what would it be, and why?* allowed the participants to enter any free text as feedback. Some of the unaltered, more interesting responses are:

- I think that one of the questions was wrong. IS NUCLEAR POWER A REASONABLE ENERGY SOURCE? The answer talked about how expensive and difficult it is - so I think it is a Contra argument, however that was marked as a wrong answer...
- the "build-your-own-argument" part was crap MUCH more features/variation needed the game has in fact nothing in common with debating, more a test for fast reading (and a bit decision) skills
- The game seems rather to be built for educational purposes, than for leisure time, where you only want to have fun. By all means, at school the game could make sense. [Translated from German]

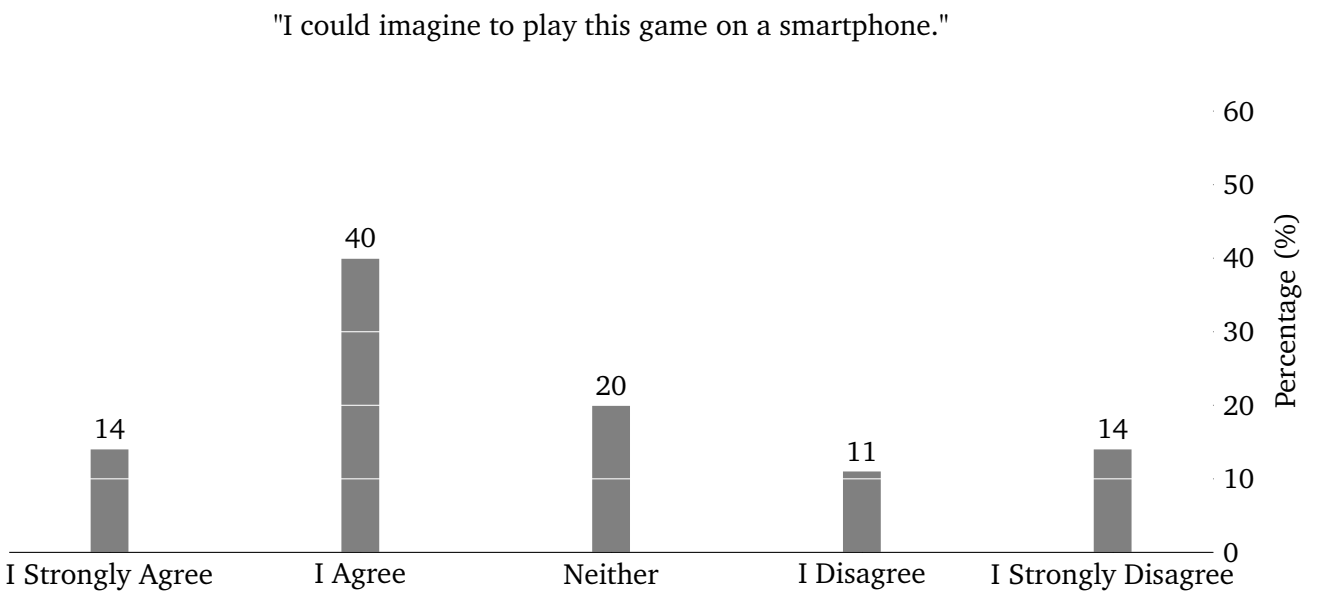


Figure 6.5: Responses concerning *I could imagine to play this game on a smartphone*. 35 responses in total.

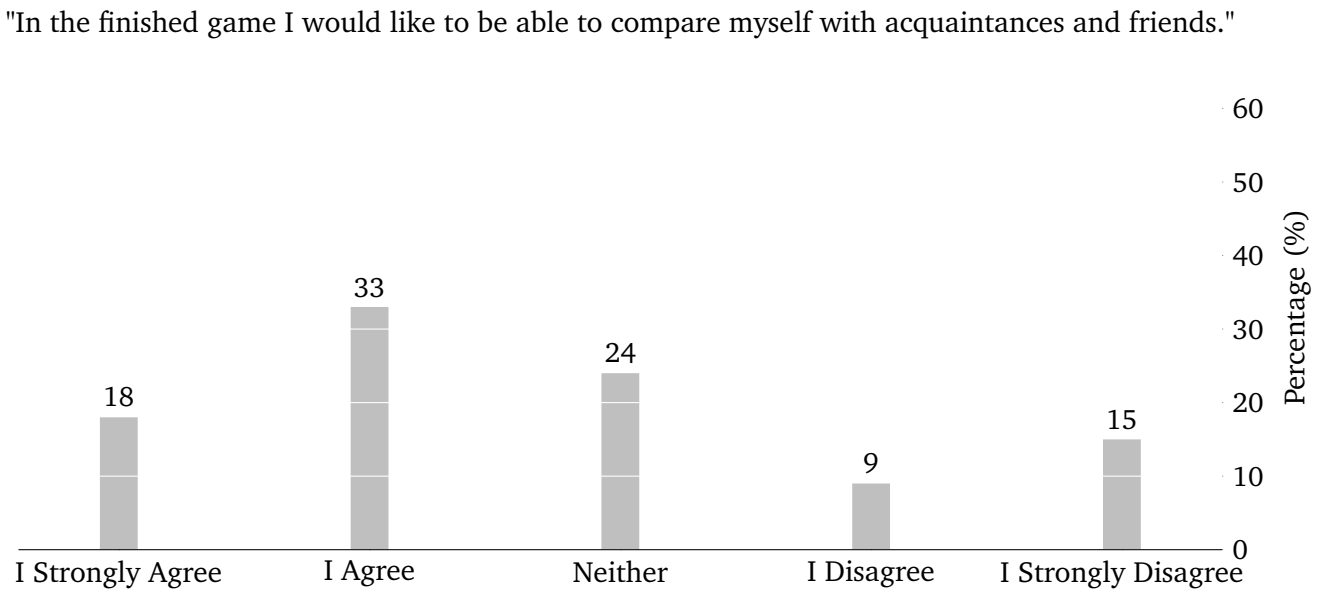


Figure 6.6: Responses concerning *In the finished game I would like to be able to compare myself with acquaintances and friends*. 33 responses in total.

6.4 Additional Findings

The demographic questions that were included in the questionnaire provided information about the participants. First of all, the questionnaire asked which age interval the participant belonged to. The results are shown in table 6.2.

Age interval	37 responses
Under 18	0.0%
18 – 20	5.4%
21 – 29	62.2%
30 – 39	27.0%
40 – 49	2.7%
50 – 59	0.0%
Over 60	2.7%

Table 6.2: Distribution of the age of the participants

Next, the educational level has been asked, as shown in table 6.3, sorted by the level. Due to the social connections the recruitment mostly was based on, the educational level is fairly high.

Educational level	36 responses
Doctoral Studies or PhD	25.0%
Bachelor Degree	22.2%
Master Degree	19.4%
Advanced Technical Certificate or High school graduation	19.4%
Dimploma	13.9%

Table 6.3: Results for the question *What is the highest educational level you have achieved?*

To get an overview of the Information Technology (IT) expertise of the participants, it has been asked how the participants assess their expertise with computers and information technology, which is summarized in table 6.4. The expertise is rather high, which again can probably be explained by the distribution during the recruitment.

IT Experiences	35 responses
Very poor	0.0%
Poor	0.0%
Fair	11.4%
Good	11.4%
Very Good	77.1%
Not specified	0.0%

Table 6.4: Results for the question *How would you rate your experiences with computers / IT?*

Furthermore, throughout the execution of the user study, a lot of event data has been generated by the participants as a side effect of visiting the user study invitation page and playing the game. These

generated events also provide meaningful insights in how the users have behaved before and during the game.

The complete set of all events tracked by the application is listed in Appendix H. Every event has at least a creation timestamp, a corresponding username, and a name. To track a sequence of related events, a random number is uniquely chosen when the user opened the app, and injected into each created event afterwards. Additionally, an event can hold arbitrary payload data.

It is important to note that the event tracking is not fully reliable. Due to code errors, certain events might have not been stored in the back-end, or events may be reported multiple times. Besides, a few anomalies in the event flow have been found in the event data, such as situations, where a subsequent game round has been started more times than a previous one. It is assumed that certain user behavior, sudden network dropouts and code errors have caused these contradictorily data. However their impact has judged to be fairly low, since they did not significantly alter the observed trends and conclusions.

Conversion Rate Funnel

The whole user study posed many different potential quitting points for the users, where they could stop their participation. Approximately 420 page visits have been registered on the user study invitation web page. The game then has been visited by nearly 90 users, which constitutes a loss of 78.6% on the invitation page. Figure 6.7 shows a conversion rate funnel, that depicts at which stage of the user study the users have quit or left the game before answering the final questionnaire. The data is based on the number of *OpenedRound* events tracked by the system.

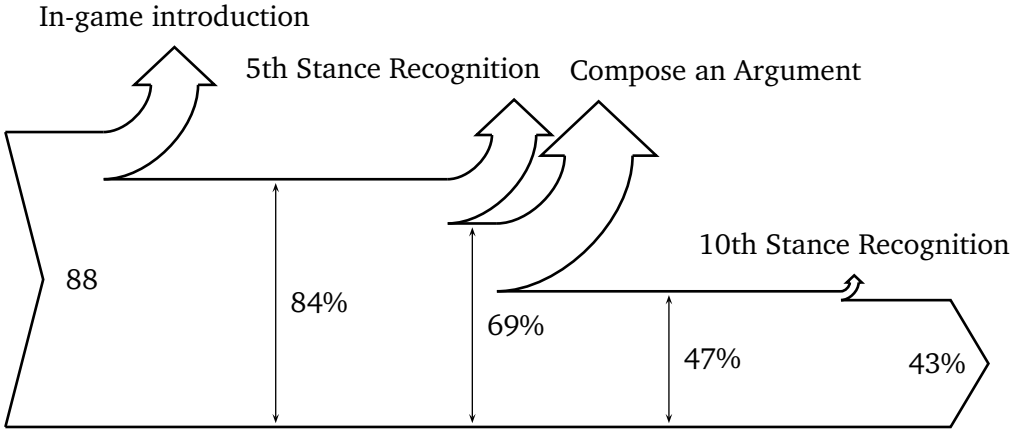


Figure 6.7: Conversion rates during the user study. 88 participants started the game. 16% of the users left the game in the In-game introduction. 15% in the 5th Stance Recognition game round. 23% in the Compose an Argument game round. Last but not least, 4% left the game before being redirected to the questionnaire. Percentages are rounded.

The initial loss of 16% during the first game round, which has been the introduction to the game itself, is not being judged further, since the reason can be a simple lack of interest on the user side. However the loss of users during the Compose an Argument game round is tremendous, but surely not surprising, due to the complexity of the task, and the contrast in terms of the entertainment compared to the preceeding Stance Recognition game rounds. Interestingly, there has not been a significant loss during the first set of Stance Recognition game rounds, but only at the end of the fifth one. A sequence of five identical game rounds obviously is hardly accepted by the users.

Evaluation of Composed Arguments

One of the game rounds the participants were faced with, has been the Compose an Argument game round, which produces a new argument instance in the system if the game round is successfully completed. The resulting arguments have been assessed from several perspectives. All of the generated arguments are attached in Appendix G.

First of all, the number of composed arguments per group have been determined, which is summarized in table 6.5. Study participants may have skipped the Compose an Argument game round or left before submitting an argument.

Group	Number of arguments
Group A	17
Group B	15
Total	32

Table 6.5: Number of composed arguments by study participants

Furthermore, it has been analyzed how many components have been created per argument. By default, the form the users are faced with provides space for one claim and one premise. Additional fields for premises could be added by the user. Fortunately, although the majority of the composed arguments contains the default number of components, two, there have also been created arguments with more than just two components. Table 6.6 lists how many arguments have been created with the number of components in the range of two to four.

Number of arguments	Number of components
28	2
3	3
1	4

Table 6.6: Distribution of the number of components in arguments

An additional option for the users to design the composed arguments is to arbitrarily rearrange the components. Therefore, the composed arguments have been analyzed in terms of the type of their first component. All of the submitted arguments have a claim at first, followed by one or more premises, which is the default configuration. This is not the expected outcome, and leads to conclusion, that the possibility to reorder components both should be better hinted but also encouraged to be used.

One of the drawbacks of user-generated data is the risk of poor quality. The number of arguments, which are considered to be of poor quality has been determined by manually scanning all composed arguments. An argument has been classified as of poor quality (spam), if the arguments is not semantically readable, that is, does not bear any meaning. The only found argument of such criteria is the following, comma separated by its components:

```
a a, b b b, bbb b, ba
```

At least this kind of content can surely be caught on the client side by implementing more sophisticated validation mechanism in the Compose an Argument game round, which checks for the length of the provided components.

To further assess the quality of the composed arguments, the number of typing errors have been counted by making use of the typing error detection of a word processor application. Only seven arguments contained such typing errors, excluding arguments considered as of poor quality.

Finally, by manually assessing the generated arguments, some further quality deficits have been revealed: The punctuation is either wrong in several cases, or completely missing. Automatic hints to add correct punctuation could be added in the game, but are quite challenging to implement, since not every components necessarily needs to end with a full stop. Next, some of the claims have been of poor quality, for example, they simply consisted of *Yes* or *No*. To conclude, nevertheless, the overall quality of the arguments is noteworthily good, and despite the one excluded argument, the users really seem to have thoughtfully composed the arguments.

Skipping Rate

Users were allowed to skip game rounds in the game variant of the user study for two reasons. First of all, skipping game rounds will be part of the final game and correspond to the normal game behavior. And furthermore, the possibility to skip game rounds should keep the users from leaving the game due to frustration. Their questionnaire response data is as valuable of those who completed all game rounds, so losing them would not have been desirable.

Game round	Number of visits	Number of skips	Percental
Static Content: Introduction	83	7	8.43%
1st Stance Recognition	72	4	5.56%
2nd Stance Recognition	66	2	3.03%
3rd Stance Recognition	74	5	6.76%
4th Stance Recognition	69	5	7.25%
5th Stance Recognition	74	4	5.41%
StaticContent: How To	61	1	1.64%
Compose an Argument	61	18	29.51%
6th Stance Recognition	39	3	7.69%
7th Stance Recognition	40	1	2.50%
8th Stance Recognition	41	2	4.88%
9th Stance Recognition	39	1	2.56%
10th Stance Recognition	41	4	9.76%
Static Content: Finish	38	1	2.63%

Table 6.7: Number of Skip events per game round both in absolute and relative numbers, the latter with regards to the number of the visits of the respective game round. Quit events are not taken into account here.

Table 6.7 lists the number of Skip events per game round, sorted by the depth of game level progress. Similarly to the observation in section 6.4 focused on the Quit events, the Compose an Argument game round caused most of the skipping actions of the participants, which can again be explained by multiple reasons. First of all, the input form could have confused or overwhelmed the users, which eventually may have led to the abortion of the participation. Next, the form may have posed a task, that has been too exhausting or costly in the participant's situation. Last but not least, technical misbehavior of the complex input forms may have frustrated the users furthermore.

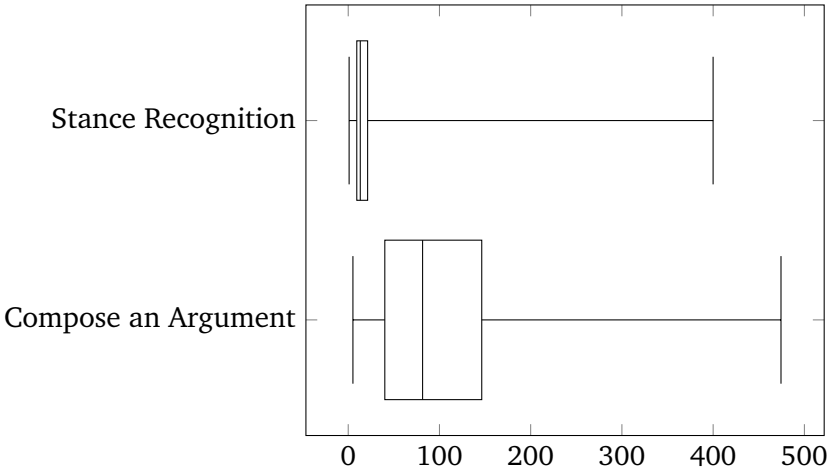


Figure 6.8: Distributions of the durations of the two interactive game rounds in seconds. Based on 607 samples for the Stance Recognition, and 69 samples for the Compose an Argument game round.

Round Duration

The game records the duration of each completed game round and sends a corresponding event to the back-end. The box plot in figure 6.8 illustrates the distribution of measured durations for the two types of game rounds that have been played as part of the user study in seconds. While a longer duration for the Compose an Argument game round is intuitively easily conceivable, the maximum duration of nearly 400s, that is, more than six minutes for the Stance Recognition game rounds raises questions. However, this possibly can be explained by the fact that users might have taken their time to study the articles, which the debating topics have linked to.

Device Information

Based on the accesses of the study invitation web page, it has been possible to gain further attributes of the study participants, such as the operating systems they have used, which is a valuable indicator for the usage scenarios of the application. The game itself does not track any device specific information, so the data illustrated in figure 6.9 is based only on the access data of the invitation web page. Thus, the actual game related data may be different.

It is important to note, that the incentives to visit the invitation page, and therefore subsequently, visiting the game, are completely different from normal conditions, where the users would most probably visit the game directly on their mobile device, which should be the main distribution channel for the application.

Clearly, due to the spread of different operating systems, the decision to develop the game as a cross platform application built using HTML5 technologies has been a good choice.

6.5 Chapter Summary

In this chapter it has been shown why evaluating the developed game was reasonable, and that a game, in particular in this case, poses special requirements towards the evaluation. The expectation has been

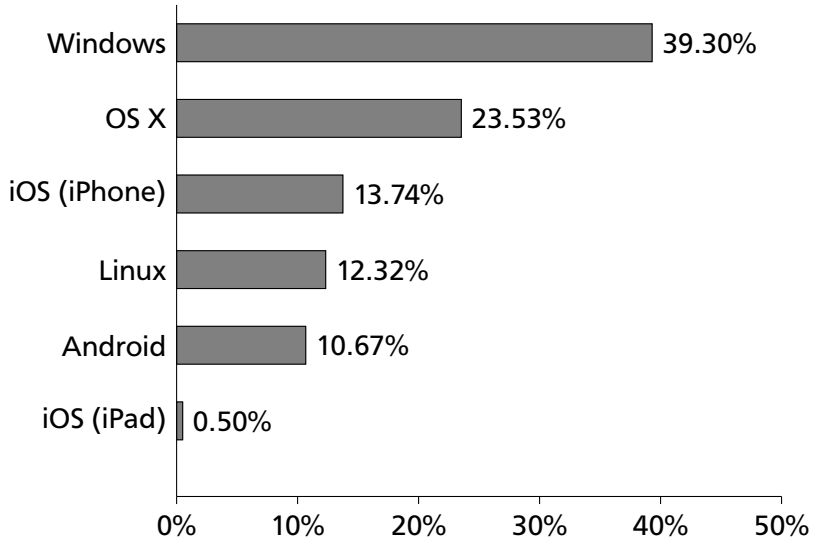


Figure 6.9: Distributions of the operating systems used to visit the user study invitation web page.

explained, that the lack of hard feedback does not harm the entertaining value of the game, in situations where arguments are used as game data, whose attributes have not been validated by the crowd yet.

Therefore, a user study has been conducted, consisting of a simplified game configuration, and a following questionnaire. All study participants have randomly been assigned to one of two groups, whereas members of the first group have received hard feedback only, and members of the second group have received soft feedback in half of the cases instead. The results have been evaluated with Mann-Whitney-U tests, which revealed that no significant differences between the two samples can be detected. From this, it has been concluded, that the proposed game design handles the lack of knowledge about the quality of the user-generated data smoothly, and therefore, can be used to harvest a corpus of annotated arguments.

Furthermore, additionally collected data has been illustrated and analyzed, such as the skipping rates per game round and the quality of the arguments, that have been created during the user study. The next chapter provides a summary of this thesis, including recommendations for future iterations.

7 Conclusion

This chapter provides a summarizing conclusion for this thesis with regards to the initial research question and the resulting outcome. Additionally, thoughts on potential future work are described.

7.1 Summary

In this thesis, the research question has been elaborated, whether a Serious Games based approach can be applied to tackle the lack of large, up-to-date, domain independent argumentation corpora, which are needed for micro-level Argumentation Mining.

Therefore, the current state of the related research in the areas of Argumentation Theory, Argumentation Mining, Argument Corpora, and Serious Games, with regards to the generation of data, has been described.

Next, the design of the game has been introduced, that is aimed at serving two aspects at once, providing a fun way to learn the basics of argumentation to the users, and also letting the users as a crowd generate and assess annotated arguments. Therefore, the game consists of multiple types of mini-games, with by being played, either create or assess argumentation data.

Founded on the responses generated by the users, the game has an integrated mechanism for user based mutual validation in order to ensure that wrongly annotated arguments are corrected over time. Afterwards, the game has been explained from the software implementation perspective. Therefore, the individual sub-components of the software are described with regards to their requirements, the chosen technologies and some selected software design decisions.

One of the key challenges of the game design has been how it should handle the lack of knowledge about the quality of the user-generated contents. In order to prove that the designed gameplay successfully absorbs this deficit, an online user study with 37 participants has been conducted. The participants have been divided into two samples and a statistical test has been performed, which revealed that no significant difference between the means of the two groups could be detected. The evaluation of further collected interrogation data emphasizes that several taken decisions could be proven to be correct.

7.2 Future Work

The developed game can be extended in several ways from many different interesting perspectives.

7.2.1 Extending the Game

Although many different ideas came up during the design of the gameplay, with the constrained time frame in mind the resulting game has been reduced to a meaningful minimum set of game elements. First of all, additional game round types could be developed, that generate further valuable data, such

as freely composed arguments, which have not been constrained by any template form. Consequently, a new game round in which the users detect the spans of the freely defined components is imaginable.

Furthermore, the automated recognition of fallacies is a challenging research question in Argumentation Mining, and therefore, a corresponding new game round type would make sense, not least because this task is fairly challenging for humans as well.

In between the argumentation theory related game rounds, some more purely entertaining mini-games could be inserted to offer some variety and to relax the otherwise sometimes monotonous game modes.

Furthermore, the game should make use of other psychological effects, such as the so called *Frog pond effect* [63]. This phenomenon describes that the people's behavior is more intensively influenced if they compare their performance with a small group of people, instead of with a larger group. Therefore, time, locality or topic domain focused high-score lists should be integrated into the game system, next to the global one, that pose different difficulties to the challenged users.

7.2.2 Quality of the Generated Data

One of the aspects that have been considered with regards to the quality of the generated corpus, is that users should have a reputation property, which is automatically determined by the system, and accordingly, their vote should be weighted during assessing data. This would allow a more fine-grained voting system, and potentially, would reduce the number of required votes to make game data truly verifiable.

More obvious improvements include automatically sorting out arguments, which repeatedly have been reported by the users, or which could not be reliably verified by the crowd despite a large participation.

Additionally, both loading the game configuration, as well as the selection of random game data could be more advanced by intentionally selecting game data, which need only few more user votes in order to be finally verifiable, or which need expert assessment only by players with high reputation.

Next, the voting system provides multiple aspects, which could be further examined after the game has been played by a stable user base for longer period of time. For example, different statistical models for the independent confidence value could be considered, such as, Fleiss' Kappa [55]. Which model fits better, could be investigated by performing a heuristic evaluation that compares the voting system outcome based on two different models for a fixed set of argument after a certain time interval.

Last but not least, the user study revealed some flaws, which should be tackled in future improvements. For example, due to its importance, the Compose an Argument game round could be improved in several ways, such as the reordering, but also the actual composing.

7.2.3 Improving the Distribution

With regards to the distribution of the game as an application, which is crucial for the growth of the corpus, all existing platforms should be served, which can be easily achieved due to the usage of cross-platform technology. Since the popular distribution platforms, such as the *Apple App Store* or *Google Play Store*, are known to be crowded, additional public relation efforts may be required.

8 Acknowledgments

I would like to thank Prof. Dr. Iryna Gurevych for offering me this challenging and interesting thesis, and for her support throughout the project within the Ubiquitous Knowledge Processing group.

I also would like to express my warm thanks to my coordinators Dr. Ivan Habernal and Christian Stab for helping me accomplish this thesis. Their guidance and assistance, the vivid discussions and their creativity helped me concretizing my research question, elaborating on it, and coming to a meaningful result.

Many thanks to the representatives of the Technical University of Darmstadt.

List of Figures

List of Tables

Bibliography

[1] Mark Weiser. Some Computer Science Issues in Ubiquitous Computing. *Commun. ACM*, 36(7):75–84, July 1993.

[2] Frans H. van Eemeren, Rob Grootendorst, Ralph H. Johnson, Christian Plantin, and Charles A. Willard. *Fundamentals of Argumentation Theory: A Handbook of Historical Backgrounds and Contemporary Developments*. Taylor & Francis, 2013.

[3] Aristotle and George A. Kennedy. *On Rhetoric: A Theory of Civic Discourse*. Oxford University Press, 1991.

[4] Stephen E. Toulmin. *The Uses of Argument, Updated Edition*. Cambridge University Press, 2003.

[5] Andreas Peldszus and Manfred Stede. From Argument Diagrams to Argumentation Mining in Texts: a survey. *International Journal of Cognitive Informatics and Natural Intelligence*, 7:1–31, 2013.

[6] Günther Öhlschläger. *Linguistische Überlegungen zu einer Theorie der Argumentation*. Linguistische Arbeiten. Walter de Gruyter, 1979.

[7] Manfred Kienpointner. *Argumentationsanalyse*. Innsbrucker Beiträge zur Kulturwissenschaft: Sonderheft. Verlag des Instituts für Sprachwissenschaft der Universität Innsbruck, 1983.

[8] Wolfgang Klein. Argumentation und Argument. *Zeitschrift für Literaturwissenschaft und Linguistik*, 1980:9–57, 1980.

[9] Dieter Wunderlich. Pro und Kontra. *Zeitschrift für Literaturwissenschaft und Linguistik*, (38/39):109f, 1980.

[10] Doug N. Walton and Chris A. Reed. Argumentation Schemes and Defeasible Inferences. *Workshop on computational models of natural argument, 15th european conference on artificial intelligence*, 2002.

[11] Chaïm Perelman and Lucie Olbrechts-Tyteca. *The new rhetoric: a treatise on argumentation*. Notre Dame Press, University of Notre Dame, London, 1969.

[12] Arthur C. Hastings. *A Reformulation of the Modes of Reasoning in Argumentation*. PhD thesis, Evanston, Illinois, 1963.

[13] Jamal Bentahar, Bernard Moulin, and Micheline Bélanger. A taxonomy of argumentation models used for knowledge representation. *Artificial Intelligence Review*, 33(3):211–259, January 2010.

[14] Soumen Chakrabarti, Martin Ester, Usama Fayyad, Johannes Gehrke, Jiawei Han, Shinichi Morishita, Gregory Piatetsky-Shapiro, and Wei Wang. Data mining curriculum: a proposal, Version 1.0 (2006), 2006.

[15] Raquel Mochales-Palau and Marie-Francine Moens. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the Twelfth International Conference on*

Artificial Intelligence and Law (ICAIL 2009), Twelfth international conference on artificial intelligence and law (ICAIL 2009), 8-12 June 2009, pages 98–109, 2009.

[16] Daniel E. Rose and Danny Levinson. Understanding user goals in web search. In *Proceedings of the 13th international conference on World Wide Web*, pages 13–19, New York, New York, USA, 2004. ACM Press.

[17] Claire Grover, Ben Hachey, and Chris Korycinski. Summarising Legal Texts: Sentential Tense and Argumentative Roles. In *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop - Volume 5*, HLT-NAACL-DUC '03, pages 33–40, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[18] Jodi Schneider. An Informatics Perspective on Argumentation Mining. *Proceedings of Frontiers and Connections between Argumentation Theory and Natural Language Processing*, 2014.

[19] Jodi Schneider. Automated argumentation mining to the rescue? Envisioning argumentation and decision-making support for debates in open online collaboration communities. In *Proceedings of the First Workshop on Argumentation Mining*, pages 59–63. Association for Computational Linguistics, 2014.

[20] Henning Wachsmuth, Martin Trenkmann, Benno Stein, and Gregor Engels. Modeling Review Argumentation for Robust Sentiment Analysis. In *Proceedings of the 25th International Conference on Computational Linguistics COLING*, 2014.

[21] Simone Teufel. *Argumentative Zoning: Information Extraction from Scientific Text*. Doctoral dissertation, 1999.

[22] Stephen Merity, Tara Murphy, and James Curran. Accurate Argumentative Zoning with Maximum Entropy models. In *Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries (NLPIR4DL)*, number August, pages 19–26, 2009.

[23] Jodi Schneider and Adam Wyner. Identifying Consumers' Arguments in Text. In *Proceedings of the Workshop on Semantic Web and Information Extraction (SWAIE 2012), Galway, Ireland, October 9, 2012*, pages 31–42, 2012.

[24] Christian Stab, Christian Kirschner, Judith Eckle-Kohler, and Iryna Gurevych. Argumentation Mining in Persuasive Essays and Scientific Articles from the Discourse Structure Perspective. In *Proceedings of the Workshop on Frontiers and Connections between Argumentation Theory and Natural Language Processing*, pages 40–49. CEUR-WS, 2014.

[25] Marie-Francine Moens, Erik Boiy, Raquel Mochales-Palau, and Chris Reed. Automatic detection of arguments in legal texts. *Proceedings of the 11th international conference on Artificial intelligence and law - ICAIL '07*, page 225, 2007.

[26] Christian Stab and Iryna Gurevych. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Qatar, 2014.

[27] Elena Cabrio and Serena Villata. Combining Textual Entailment and Argumentation Theory for Supporting Online Debates Interactions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 208–212, 2012.

[28] Chris Reed and Glenn Rowe. Araucaria: Software for puzzles in argument diagramming and XML. *Department of Applied Computing, University of Dundee Technical Report*, 2001.

[29] Chris Reed and Raquel Mochales-Palau. Language resources for studying argument. In *Proceedings of the 6th conference on language resources and evaluation*, pages 91–100, 2008.

[30] Christian Stab and Iryna Gurevych. Annotating Argument Components and Relations in Persuasive Essays. In Junichi Tsujii Hajic and Jan, editors, *Proceedings of the the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 1501–1510, Dublin, Ireland, 2014. Dublin City University and Association for Computational Linguistics.

[31] Raquel Mochales-Palau and Marie Francine Moens. Study on the structure of argumentation in case law. In *Frontiers in Artificial Intelligence and Applications*, 2008.

[32] Maria Paz Garcia Villalba and Patrick Saint-Dizier. Some facets of argument mining for opinion analysis. In *Proceedings of COMMA 2012*, volume 245, pages 23–34, 2012.

[33] Michael Zyda. From visual simulation to virtual reality to games. *Computer*, 38(9):25–32, 2005.

[34] David Michael and Sande Chen. *Serious Games: Games that Educate, Train and Inform*. Thomson Course Technology, 2 edition, 2005.

[35] Julian Alvarez, Olivier Rampnoux, Jean-Pierre Jessel, and Guilles Methel. Serious Game: Just a question of posture? *Artificial & Ambient Inteligence, AISB*, 2007.

[36] Luis von Ahn. Human computation. In *Proceedings of the 46th Annual Design Automation Conference on ZZZ - DAC '09*, page 418, New York, USA, 2009. ACM Press.

[37] Luis von Ahn. GWAP: 2008-2011. Available at http://www.gwap.com/2011/08/gwap-2008-2011.html. Online; Visited on 2015-02-23.

[38] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, volume 6, pages 319–326, New York, New York, USA, 2004. ACM.

[39] Francois 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*, 2011.

[40] Hubertus Kohle, François Bry, Thomas Krefeld, Christian Riepl, and Klaus Schulz. Abschlussbericht zum Projekt Entwicklung sozialer Web-Plattformen zur Datengewinnung in den Geisteswissenschaften (KO 1091/4-2). (July 2011), 2013.

[41] 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*, volume pages, pages 55–64. ACM, 2006.

[42] Unknown. EteRNA - Played by Humans. Scored by Nature. Available at http://eternagame.org/web/about/. Online; Visited on 2014-01-18.

[43] Nafees Ahmed. Disguise: A Game that Evaluates Visualization Algorithms. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, pages 269–272. ACM, 2014.

[44] Luis von Ahn and Laura Dabbish. Designing games with a purpose. In *Communications of the ACM*, volume 51, pages 58–67, August 2008.

[45] Chris Callison-Burch and Mark Dredze. Creating Speech and Language Data With Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 1–12. Association for Computational Linguistics, 2010.

[46] Rada Mihalcea, Alexander Conrad, Janyce Wiebe, and Wiebe Akkaye. Amazon Mechanical Turk for Subjectivity Word Sense Disambiguation. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 195–203. Association for Computational Linguistics, 2010.

[47] Karën Fort, Gilles Adda, and Bretonnel Cohen. Amazon Mechanical Turk: Gold Mine or Coal Mine? *Computational Linguistics*, 2011.

[48] Jesper Juul. *What is Casual?* MIT Press, 2010.

[49] Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. MIT Press, 2004.

[50] Jamie Madigan. The Psychology of Video Game Avatars. Available at http://www.psychologyofgames.com/2013/11/the-psychology-of-video-game-avatars/. Online; Visited on 2014-12-29.

[51] Kurt Lewin. Untersuchungen zur Handlungs-und Affektpsychologie. *Psychologische Forschung*, 11:302–389, 1928.

[52] Kyoo-Lak Cho and David H. Jonassen. The effects of argumentation scaffolds on argumentation and problem solving. *Educational Technology Research and Development*, 50(3):5–22, 2002.

[53] Dong Dong Li and Cher Ping Lim. Scaffolding online historical inquiry tasks: A case study of two secondary school classrooms. *Computers and Education*, 50(4):1394–1410, 2008.

[54] Michael Degusta. Android Orphans: Visualizing a Sad History of Support. Available at http://theunderstatement.com/post/11982112928/android-orphans-visualizing-a-sad-history-of. Online; Visited on 2015-03-04.

[55] Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[56] Hannu Korhonen and Elina Koivisto. Playability heuristics for mobile games. In *Proceedings of the 8th conference on Humancomputer interaction with mobile devices and services MobileHCI 06*, page 9, 2006.

[57] Gary W. Heiman. *Basic Statistics for the Behavioral Sciences*. Cengage Learning, seventh ed edition, 2013.

[58] Rensis Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 22 140:55, 1932.

[59] Victor M. Montori and Gordon H. Guyatt. Intention-to-treat principle. *Canadian Medical Association Journal*, 165(10):1339–1341, 2001.

[60] Björn Rasch, Malte Friese, Wilhelm J. Hofmann, and Ewald Naumann. Der t-Test, Quantitative Methoden 1. In *Einführung in die Statistik für Psychologen und Sozialwissenschaftler*, chapter 3, pages 59–60. 2010.

[61] Samuel S. Shapiro and Martin Wilk. *An Analysis of Variance Test for Normality (complete Samples)*, volume 52 of *Biometrika*. Rutgers, The State University, 1965.

[62] Henry Mann and Donald Whitney. On a test of wether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18:50–60, 1947.

[63] Mark D. Alicke, Ethan Zell, and Dorian L. Bloom. Mere categorization and the frog-pond effect. *Psychological science: a journal of the American Psychological Society / APS*, 21:174–177, 2010.

A Programming Manual

The following paragraphs provide an overview of the different technology stacks used for the game, and are meant to be read by developers.

A.0.4 BaasBox in a Nutshell

BaasBox (Apache License) is an open-source Back-end as a service application written in Java and built on top of the Play! Framework. The application aims to facilitate the development of (mobile) applications by providing commonly demanded functionalities out of the box, such as user, friends and asset management, a database, Push Notifications and a REST API. Next to the server application, which runs on Unix and Windows systems by starting a simple Java process, BaasBox also provides client SDKs for iOS, Android and JavaScript.

There is an administrative interface, called Console, for the BaasBox instance, which is accessible via any browser on the port and the path specified by the BaasBox configuration file. By default, this is *http://localhost:9000/*. The Console allows you to configure the BaasBox, and write plug-ins and to manage the database.

Instead of an MySQL based database as it is often the case, BaasBox internally makes use of a so called OrientDB database instance. This database provides several benefits compared to traditional SQL based relational databases, such as immediate JSON support. Database entities are linked via so called Links, which must be created manually similarly to how entities themselves are created. Please note that the OrientDB instance is internal and can not be accessed directly. Broadly speaking, the BaasBox must be handled as a black box with only few configuration interfaces. At the current stage it is also still a little bit unclear how BaasBox performs in heavy traffic situations with regards to the reachability and scalability.

The database can be backed up, and restored, via the Console. The database will then be bundled into one ZIP archive, that can be downloaded directly from within the Console, or my copying the file from the db subfolder on the server.

Generally, BaasBox allows you to create, read, update and delete both users and so called documents, which are stored in collections. For comparison, collections correspond to MySQL tables, whereas documents correspond to rows within the tables.

Accessing all kinds of database entities is regulated by an Access Control management system. This allows specifying rights to create, update, and delete entities, and to prevent malicious attacks. Similarly, all users can store data in their profile, which is visible either only for themselves, for registered users, or for all users.

Furthermore, each API call must contain the BaasBox App Code, which is a short string defined by yourself. If the API call requires authentication, the call must also include an access token, which you can get by logging in with user credentials.

The client SDKs are fairly simple to use, and facilitate storing and modifying entities in the back-end tremendously. Internally, the client SDK methods are transformed into REST API calls. Due to the client-server transmission delays, all client SDK methods are designed to work asynchronously. The results of the calls are either available via the promise or the callback design pattern.

BaasBox allows to extend its API via plug-ins. In BaasBox version 0.9.4 of, each plug-in is written in plain JavaScript, and is represented by one plug-in file, which can be created and modified only via the Console. Unfortunately, the files are stored in the in the database, and thus, there is no way to integrate them into version control systems, or to modify them with external IDEs. On the other hand, the plug-ins are backed up automatically with every normal back-up of the database.

Each plug-in can expose multiple new REST end-points, which allows to move application logic from the client to the server side. This can be reasonable for several reasons, such as enhanced security, speed, and access to restricted data. Within the plug-ins, you can make use of two existing JavaScript frameworks: Box and underscore. Box is a server-side SDK to access the BaasBox functionalities, similarly to the client SDKs. Underscore is a popular JavaScript based framework, that facilitates developing JavaScript applications in general. Plug-ins also allow you to share commonly used code. Therefore, one plug-in can act as a library for all other plug-ins, which is really helpful.

In order to test plug-ins, one should consider making use of a REST client to send requests towards the newly exposed APIs defined by the plug-ins. It is then important to include proper authentication information in the HTTP requests, such as *X-BAASBOX-APPCODE* and *X-BB-SESSION*.

During this thesis, the BaasBox has been in a beta stage (version 0.9.0 up to version 0.9.4), but generally has been perceived as fairly stable without any crash so far. It is important to note however, that APIs may change, certain functionalities may be reduced or new capabilities are added over time. However the benefits of using BaasBox overtopped all other considered alternatives, so that we decided to hazard the consequences to choose BaasBox.

Unless stated differently by the release notes of upcoming versions, the recommendation is to use version 0.9.4 on the production server, since this game has been developed and tested with this version.

A.0.5 AngularJS in a Nutshell

The AngularJS framework, developed by Google (MIT license), is a framework to develop applications for the browser, using plain JavaScript and HTML. At a first glance, while looking at some examples, a developer usually developing native applications might initially feel lost, but AngularJS can feel quite comfortable after the first few hurdles.

Basics

Just like most of all the web application frameworks out there, AngularJS claims to be a Model-View-Controller oriented architecture. Views are implemented by individual HTML files, or are embedded within <script> tags in the index.html file. Controllers, either separated into distinct files, or embedded into the main HTML file, are JavaScript functions that contain sub-functions to implement the application logic. Controllers also define and contain the models that are used by the views to display data.

Broadly speaking, AngularJS is about about scopes, directives, bindings, and dependencies.

Scopes are simple JavaScript objects, that contain both data and methods, which are available in the views to render data or provide interactivity. There is one global rootScope in the application, and multi-leveled sub-scopes, which inherit everything from their parent scope. Every controller you define creates a new scope, which is solely managed by the controller instance. Within the controller code, you add data to the scope, and add functions, that you can make use of in the view, that is controlled by the controller. If a view, controlled by a controller A, contains a sub-view, which is controlled by a controller B, B inherits every data and method from controller A automatically. B can also overwrite data and methods, if required. This way you can simulate the classical inheritance to a certain extent, which you might be used to from native coding.

Next, AngularJS' Directives are newly introduced HTML attributes and nodes, which AngularJS provides at runtime. With every directive you create in your code, you can create one new HTML node or node attribute. Directives allow you to further modulate your code by, e.g., wrapping often used HTML structures into one single HTML node (e.g., an HTML tag to display a full argument), or by adding additional capabilities to existing HTML nodes via new HTML attributes (e.g., a fallback image URL for the *<img>* tag). How these additions work, is defined by your directive's code.

AngularJS provides sophisticated two-way bindings throughout your application. If you set u p your view to display the content of a certain variable within a element, and you change that variable in the controller over time, the span will automatically be updated, without having to take care of re-rendering your view. On the other hand, if you have an input element in your HTML view, which is bound to a variable of your model defined in your controller, your controller will see these changes, the user puts into the input field, without further action immediately. This saves vast amounts of boilerplate code, and lets you focus on your actual application logic.

Last but not least, Angular is built on top of a dependency management architecture. Since everything you code in AngularJS belongs to a certain module, you can create dependencies among your modules (and of course 3rd party modules). Controllers, in which you would like to make use of certain services, must specify these dependencies in their function argument list. This is a little bit weird at the beginning, since AngularJS recognizes the names of the controller function's input variables, and maps them into the respective modules, but you get used to it.

It is also important to mention, that, next to the general application architecture foundation, AngularJS also provides a lot of utility functionalities. There are a lot of commonly used functions included, such as *angular.forEach()*, or *angular.element()*, which allow you to use AngularJS only, instead of relying on jQuery or underscore.

Speaking of jQuery, which every JavaScript developer probably once got in contact with, it is important to mention that mixing both frameworks in your web app code is not recommended. AngularJS follows some completely different approaches, which could break at runtime, should you make use of DOM manipulation with jQuery. The recommendation would be to not make use of jQuery at all, and to use AngularJS where possible. If AngularJS does not provide certain functionalities, try finding 3rd party modules in the vivid AngularJS community.

A.0.6 IonicFramework in a Nutshell

IonicFramework (MIT license) is a powerful set of tools that accelerate developing mobile applications using HTML5 tremendously. Built on top of AngularJS and Cordova, Ionic makes use of established open-source projects and adds several assets and tools to develop cross-platform applications very quickly.

The framework comes with several CSS classes that let you easily create user interfaces in HTML, which mimic Android and iOS interfaces. Next, Ionic contains JavaScript components built on top of AngularJS, which provide interactive user interface elements, such as scroll views, toggles, list elements, and more. Again, these components try to replicate the native look and feel of Android and iOS, choosing the right appearance at runtime based on which platform the web app is running.

Furthermore, Ionic comes with a Command Line Interface (CLI), that provides multiple very helpful tools to improve the developer's comfort. At the very beginning, the CLI lets you create a basic folder structure for your app, based on one of three general app templates (e.g., tabbed interface). Next, the CLI lets you run your app project on a localhost HTTP server, for which you can also configure proxies to 3rd party web services, which can be really helpful during development.

In addition to it, the CLI also is a wrapper for the Cordova CLI, and thus, lets you create the specific platform projects (iOS, Android, and more) with a one line command. What is great about Cordova, is that it also lets you define the browser as a platform, and so you can develop one single application, that later can be wrapped by an iOS wrapper, an Android wrapper, and a browser wrapper. You usually do not have to touch the different platform projects (e.g., Xcode project for the iOS app), since the entire application logic is handled by your JavaScript code, but you only have to build and distribute them.

Next, the underlying Cordova architecture lets you define so called hooks, which you can make use of during deploying your code. For example, one hook for the "afterprepare" phase can minify and uglify your entire Ionic project, so that your code is obfuscated and harder to reverse-engineer, once the app is deployed on a web server.

Ionic is backed by a vivid community, where developers can receive helped quite quickly.

A.0.7 Developing new Game Rounds

Game rounds are the atomic mini-games that are chained together to form one level within the overall game. The application has been developed in a module based architecture, so that altering the game logic, and more specifically, the game rounds is easy. Adding new kinds of game rounds means adding a new controller and then altering your game configuration to make use of the new round within the game. In the following section, a new ChooseTheArgumentQuality round will be developed.

Adding New Round Controller

In */www/src/App/GameControllers folder*, there is a file called *NewRoundTemplate.js*, which contains a skeleton for the new controller. Copy and paste its contents into a new *ChooseTheArgumentQualityController.js* file in the same directory. Rename the controller to *ChooseTheArgumentQualityController* at the head of the file.

Next, have a look at the top of the *RoundController.js* file, where you will find all the methods you can overwrite in the ChooseTheArgumentQualityController to customize the behavior of this round. You specify all kinds of round specific data by implementing the respective methods. For example to specify the amount of coins the user should receive for the input, by returning the desired integer in the *pointsForVerifiableInput()* method. Similarly, you define the countdown duration by implementing *countdownDuration()*.

The methods you implement also acts as hooks for certain events. The users will eventually click the Go On button to submit their choice or their data. This click will not be handled by you, but by the parent controllers, which will then call specific methods in your controller implementation. For example, the users' input must be validated first: Did they choose anything at all? If not, return false. The complete lifecycle of events, from the perspective of the parent RoundController, which controls your custom round controller, is listed below:

Notify the round that it will start ('roundWillStart()').

Request the message to be shown if the input is correct ('correctInputMessage()').

Request the message to be shown if the input is incorrect ('incorrectInputMessage()').

Request the message to be shown if the input is not verifiable ('acceptedInputMessage()').

Notify the custom round controller that the round did start ('roundDidStart()').

The user will not start playing the game round and eventually click the Go On button.

Check if the user took less time to provide an answer than defined to be reasonable by the custom round controller ('minimumAssumedPlayingTime()').

If so, 'downrateInput' in the scope will be set to true. If not, 'downrateInout' is false.

Check if the custom round continues immediately without further input validation or coin based gratification ('continuesImmediately()').

If so, the round will finished immediately and stop. Otherwise go on.

Check if the input is valid ('inputIsValid()'), that is, if the user made any valid choice at all. This choice can be the wrong choice, and the input still be valid.

If not, request the text that will be shown to indicate to the user that the input is invalid and stop ('invalidInputMessage()'). Otherwise, go on.

Let the the custom round controller reveal the solution ('revealSolution()').

Let the custom round controller submit any valuable user-generated data ('submitData()').

Let the custom round controller decide whether the users' input is verifiable, that is, if it can be determined if the input is correct or incorrect ('inputCorrectnessVerifiable()').

If the input can be verified, request from the custom round controller the amount of coins for the verified input ('pointsForVerifiableInput()'). This is usually the maximum amount of coins for this round, or 0 if the input is incorrect (wrong choice). If not, request from the custom round controller the amount of coins for a non-verified input ('pointsForNonVerifiableInput()'). This is usually a value between 0 and the maximum amount of coins, to reward the users for their input, which at the current stage cannot be verified.

Now wait for the specified duration in milliseconds ('delayBeforeGoingOn()'). This is used to let the users view the revealed solution for a short period of time.

(If the input could be verified, based on the amount of coins the users get, either the success or the failure modal will be shown. Otherwise, the general Round Done modal will be shown.)

After clicking the continue button within the modal, the modal will be hidden. Notify the custom round controller that the round will end ('roundWillEnd()'').

Request the output of the custom round controller ('output()'). This can be null. It will be stored in the 'previousOutput' key of the scope so that it will be available for the following round(s).

Notify the custom round controller, that the round did end ('roundDidEnd()').

A.0.8 Altering the Voting System Parameters

If the parameters, or the algorithm itself, must be altered to improve the voting outcomes, then there are multiple modifications required, both on the client side, and within the back-end.

The Voting System Explained

In general, users can both generate data, but also assess the data of others. The latter is called crowd-sourced voting, and technically it works as follows.

Whenever a user assesses any kind of attribute of any object, e.g., the stance of an argument, a new voting is being submitted to the back-end. The voting contains the three keys id, keyPath and value. The id is the record identity of the object whose attributes has been assessed by the user. The keyPath is the key path of the attribute, that has been assessed by this voting, beginning at the assessed object at the root. The value is the value which the user thinks the referenced attribute should contain.

So if a user for example is being asked whether a shown argument is a Pro or a Contra argument, and Pro has been chosen, a voting will be submitted with the following JSON content:

```
{
    id: ''
    keyPath: 'stance',
    value: 'pro'
}
```

Similarly, when a user is being asked to pick the premise(s) within an argument, for each (!) of the picked components (in this case, the components array entries with indexes 1 and 2, the following voting will be submitted:

```
{
    id: '',
    keyPath: 'components.1.type',
    value: 'premise'
},
{
```

```
    id: '',
    keyPath: 'components.2.type',
    value: 'premise'
}
```

The generated votes will be taken into account whenever an object is returned by the back-end via any of the plug-ins. For each of the attributes of the object which votes exist for, a voting distribution object is injected next to it as a sibling with the key for the form 'voted[CamelCasedAttributeName]', prefixed with a " symbol to indicate that it's a virtual key (to make clear it doesn't belong to the database object data). A voting distribution represents the current outcome of the crowd-sourced voting for the respective attribute, and furthermore contains data about the assumed confidence in the voting distribution:

```
{
    [value]: [percentual value],
    [],
    #confidence: [percentual value],
    #independentConfidence: [percentual value]
}
```

For each each of the values the users have submitted for the respective key path, a key with the name of the value exist. The key references a value between 0 and 1, that represents the statistical occurrence of this value among all of the votes for this keypath. Currently, this value is simply calculated by the proportion v of this value of all values (see thesis for details).

So if there is one 'Pro' voting and nine 'Contra' voting, the distribution would look as follows:

```
{
    Pro: 0.1,
    Contra: 0.9
}
```

Voting Distribution Confidence

Two additional values are part of every voting distribution. The independentConfidence is an indicator for how obvious and clear the outcome of the voting currently is, and thus is a function of all votes for the key path. A value of 1 means that the voting altogether is very clear and unambiguous, that is, the crowd has fully decided for one and only one of the options. A value of 0 on the other hand means that the crowd has not determined a preferred value at all.

Currently, the *independentConfidence* value is calculated by subtracting from the maximum value v the average of all remaining values, as illustrated by the following formula, where V is the set of all v values in this distribution (see thesis for formula).

For voting distributions with two possible values (e.g., the stance voting), this will always be the difference between the proportional occurrences of the voted values.

While playing the game, the *independentConfidence* value now can be used to check if a user's choice can be verified, that is, checked whether the user's game input is correct or not. If the crowd has not

produced a distinct, clear voting on, e.g., the argument's stance, the currently player user's input may not be verifiable yet. Verifying input should be possible though, if the independentConfidence exceeds a certain threshold, e.g., 0.65.

However the independentConfidence might not be as reliable as required, since it does not take into account the number of votes so far. If there is only one voting for the stance of an argument. The independentConfidence will automatically be 1. One voting however surely can not represent a reliable source for verification. Thus, another key is part of every voting distribution, which is the *confidence*.

The value of the confidence is based on the value of the independentConfidence, but in addition to it takes into account the number of votes so far. Therefore, a minimum number of required votes is specified, that acts as a threshold up to which the voting distribution is assumed to be unreliable with regards to the participation. If more votes have been generated, the *confidence* value equals the *independentConfidence*. Otherwise, the *confidence* value equals the *independentConfidence* value multiplied by a factor df:

```
df = 1/(1+ max(0, #requiredNumVotings - #numVotings) )
```

For example, if 10 is specified as the minimum number of required votes, and only one voting has been generated so far, the *independentConfidence* is 1.0, whereas the *confidence* is

```
#confidence = #independentConfidence *df = 1.0 *0.1 = 0.1
```

This value is far more reliable than the original *independentConfidence* and should is being used instead within the verification of user input in the game.

Example

As an example, an argument returned by the back-end may have a structure similar to the one that follows:

```
{
    id: '',
    components: [{
        body: '',
        type: 'claim',
        #votedType: {
            claim: 0.75,
            premise: 0.25,
            #confidence: 0.66,
            #independentConfidence: 0.75
        }
    },{
        body: '',
        type: 'premise',
        #votedType: {
            claim: 0.1,
```

```
            premise: 0.9,
            #confidence: 0.85,
            #independentConfidence: 0.9
        }
    }],
    stance: 'pro',
    #votedStance: {
        pro: 0.8,
        contra: 0.2,
        #confidence: 0.6,
        #independentConfidence: 0.8
    }
}
```

Modifying the Voting Algorithm

The outcome of the voting system can be influenced both by choosing different input parameters (thresholds), and by completely rewriting the voting distribution calculation. It is important to note however, that the actual structure of a voting distribution must remain unchanged, unless you alter the game mechanics code passages that make use of the voting distributions returned by the server. That means that every voting distribution should have the *confidence* key, the *independentConfidence* key, and one key for each of the different voted values, accordingly to the semantics described above.

While the parameters for the current voting distribution calculation algorithm reside in both the client and the server side, the actual passage that generates the voting distribution is part of the *arg.votings* plug-in. So to modify the algorithm itself, only the *arg.votings* plug-in must be altered.

The minimum number of required votes, that acts as a threshold up to which the voting distribution is assumed to be unreliable with regards to the participation, is specified in the back-end, since the voting distribution calculation is done server-side. Therefore, a variable called *kConstNumberOfVotingsForReliability* is defined at the head of the *arg.votings* plug-in.

In addition to that, the respective threshold up to which a voting distribution is being ignored due to a small confidence value, is stored on the client side, that is, in the game. A variable called kConstMinimumConfidenceThreshold is defined in *ArgLib.js*.

If the confidence of a voting distribution exceeds the minimum confidence level, then the highest of all voting values should be chosen, but only, if it exceeds the kConstMinimumMajorityThreshold, which is also defined in the *ArgLib.js* file.

A.0.9 Administrating Game Contents

All of the non-static contents that the users of the game can be faced with, are stored in the back-end, which is why with the back-end you have full control about what data the game makes use of. Since the back-end Console does not allow to conveniently access and modify the data, such as domains, topics, arguments and articles, a dedicated tool called ArgueAdmin has been developed.

ArgueAdmin is a web application, that not only connects to the same back-end, but also shares code libraries with the game. Although ArgueAdmin is built on top of the same technology stack as the game itself, and thus, is usable on small-screen mobile end devices, it is recommended to use a desktop browser, due to the visual space and the improved input via keyboard in desktop environments.

The interface of ArgueAdmin is mainly designed to let you create and modify the set of available arguments per topic per domain. After creation, every entity is automatically available by default in the game, e.g., via the randomArgument API endpoint.

Defining the Worlds, Rounds and Levels

Which worlds, levels and rounds are shown in the game is specified by the *gameConfiguration.json*, which is a JSON file stored in the back-end and downloaded and parsed during every app start. If the file can not be loaded for some reason, the *localGameConfiguration.json* file will be tried to be loaded as a fallback, which is bundled with the app locally.

The Game Configuration File Structure

The JSON based game configuration file is a data tree, with a 'worlds' key at the root of the object. Broadly speaking, the key references an array of world description. Within each world description, an array of levels is defined. Each level description then contains a set of rounds. All of the data defined in the game configuration is directly available in the respective controllers (WorldController, LevelController, RoundController, and custom round controllers) of the game. However, a certain structure is pre-defined and required to separate mandatory controlling data from custom data, that act as input for the custom controllers.

To go on, a specification of the individual description objects is listed below, with the respective keys, that will be taken into account during parsing the configuration file.

World Description

title The Title of the world, which will be shown below the world graphic.

id A unique id to identify the world within the configuration.

background An image resource filename, including the extension, which will be used to visualize the world. The file will be loaded from the /GameContents/Worlds/[id]/ folder.

enabled A boolean flag if the world is currently playable, or if it is grayed out for the user.

levels An array of level descriptions for all the levels, that are part of this world.

Level Description

title The title of the level, which will be shown on the map. This should be a very short one-word title, such as, e.g., 'Education'.

id A unique id to identify the level within the configuration.

icon This defines the icon that will be used to represent the level on the world map. The key should reference a string that will be used a the CSS class for the icon element. This should ideally be an ionicon selector, such as 'ion-happy'. See the ionicons documentation for more details.

iconColor The CSS class that will be used to paint the icon of the level bubble in the world map. This should ideally be an Ion color class, such as 'positive', 'royal', or 'energized'. See the IonicFramework documentation for more details.

backgroundColor The CSS color string that will be used to paint the background of the level bubble in the world map. This should ideally be a HEX color, such as 'fff000' or any CSS color name, such as 'white'.

isBonusLevel A boolean flag that defines if this level should be presented as a bonus level. The user will be notified after finishing the first bonus level, that this level can be repeatedly played to earn more coins quickly.

coordinates A coordinates object with the key x and y, which reference a value between 0 and 1 to geographically pin the level bubble on the world map. A definition like "x":0,"y":0 pins the level in the upper left corner of the world, whereas a level with "x":1,"y":1 is pinned in the bottom right corner.

customView (optional) A name of a custom level view, which will be used instead of the StandardLevel view. This allows to design a completely customized level in terms of its look and its behavior. This is currently being used by the first and the last level of the GreenWorld world where skipping rounds is not allowed. Note that within the specified view you can specify a custom level controller, in which you can overwrite methods of the LevelController instance to modify its behavior. Currently the following customView options are available: - StandardLevel (Standard behavior with skippable rounds) - NonskippableLevel (Standard behavior, except disallowed round skipping)

requiresCompletion (optional) An array of integers, that reference other levels by their array index within the world's levels array that must be completed by the user, before this level is unlocked to be played. Otherwise the user will only see a grayed out level bubble.

parameters (optional) An optional data object, which should be used to specify custom level data, to customize the behavior of the level controller instance.

rounds An array of round descriptions. Currently, the rounds will be played back accordingly to the order of the array.

Round Description

id A unique id to identify the round within the configuration.

parameters (optional) A set of custom data, which the custom round controller should make use of. Via the parameters the rounds can be set up to, e.g., show random arguments only coming from the 'Education' domain.

view The view that should be used for this round. Note that for rounds, the view key is not optional. The view also defines which custom round controller will be used, by appending 'Controller' as a suffix to the view name. Currently the following views are available:

- StanceRecognition The user has to choose either Pro or Contra for a given argument. By default, a random argument will be fetched from the back-end. A specific argument referenced by its its can be forced to be shown in this round by specifying the 'specificArgumentId' parameter. A specific domain which the random arguments are fetched from can be specified by the 'domains' parameter.

- PickTheComponent The user has to select all of the components demanded by their type. By default, a random argument will be fetched from the back-end. A specific argument referenced by its its can be forced to be shown in this round by specifying the 'specificArgumentId' parameter. One or multiple specific domain(s) which the random arguments are fetched from can be specified by the 'domains' parameter, which takes a comma-separated list of domain titles in any language.

- ComposeArgument The user has to compose an argument for a given topic.

- StaticContent A static content file will be presented and no coin based reward will be used. Use the HTMLFile parameter to define the name of the HTML file that shall be shown. The HTML file must reside in the GameContent/StaticContent folder. The '.html' suffix is not required. Additionally or alternatively, the HTMLContent parameter can be used to specify raw HTML content which will be shown as the content of the round.

Modifying the Game Configuration

Technically, the configuration stored in the back-end is a simple BaasBox Asset, and thus, a public document accessible by everyone. To modify the set of worlds, levels and rounds shown in the game, the configuration file must be modified accordingly. Since the BaasBox Console currently does not provide a way to modify an existing Asset, the old file must be deleted, and a new file must be uploaded again. However it is important to note, that you must make sure that the structure of the configuration file matches the one expected by the game. In the case that you decide to download the current *gameConfiguration.json* Asset to modify it, and re-upload it, the structure will be damaged, since BaasBox currently wraps a downloaded Asset by an outer object. So make sure to remove the outer structure to have the game configuration at the root of the JSON tree:

```
{"result":"ok","data":{"worlds":[
  {"title":"Green Lands",
  "id":"GreenWorld",
  "background":"map1.png",
  "enabled":true,
  "levels":[{"id":"FirstLevel",[]
```

Which would result in:

```
{"worlds":[
  {"title":"Green Lands",
  "id":"GreenWorld",
  "background":
  "map1.png",
  "enabled":true,
  "levels":[{"id":"FirstLevel",[]
```

Notice how the 'worlds' key is at the root of the object, which is what the game expects during parsing the configuration.

Modifying Database Records Directly

If the situation requires to modify any database records, but the respective functionality is not available via the ArgueAdmin tool, it may be necessary to modify the contents within the BaasBox Console directly. Therefore, you need to log into the Console with credentials which administrative roles are assigned to. Next, switch to the Collections or Documents section on the left side and modify the data as desired.

Be sure to not unintentionally delete data, which is still being used by the current game configuration. Otherwise unexpected behavior might occur.

It is also important to mention, that altering the data directly may break the links between multiple database entities, such as topic to article relations, or votes with key paths to specific sub attributes of the objects.

A.0.10 Administrating User Generated Data

The ArgueAdmin tool allows to both hide and delete data generated by the user. This however requires administrative rights for the user account that is used to log in. Conceptually, every database entity, that could be directly visible to the user in the game, can have a 'hidden' flag. If the object has a 'hidden' key at the root level of the JSON description, no matter what the content of the value is, the object will not be visible in the game. Please note, that a 'hidden':false would also hide the object then.

A.0.11 Administrating Users

Users that be suspended via the BaasBox Console directly. The respective users will then not be able to log in anymore.

A.0.12 Developing Locally, Deploying on the Server

The IonicFramework, and the underlying Cordova framework greatly facilitate the develop process. The app has been written in *Atom.io* with plug-ins that provide snippets for the IonicFramework. While developing, the Ionic CLI has been used to serve the project via a local HTTP server. Since the BaasBox is running on a different port, the Cross Origin Policy of the browser disallow accessing the BaasBox API by default for security reasons. The Ionic project however can be set up to instantiate a proxy for the BaasBox server on the same port as the local HTTP server (see the Ionic.project file in the arguegame project root folder).

A.0.13 Running ArgueGame Locally

If you plan to improve the app or to extend its capabilities, it is recommended to develop and run the project locally. Once you are done, you can deploy the project on a server with a few command lines. To develop locally, you need to run the HTML based front-end (client) using the Ionic CLI, and the BaasBox back-end as the server.

Running the Game Front-end Locally

Therefore, install the Ionic Command Line Interface, which is part of the IonicFramework bundle. Check out the project 'arguegame' from the repository and open the folder in the Terminal to run the following command:

```
ionic serve
```

This will start the local HTTP server and open your default browser automatically. Alternatively, you can run

```
ionic serve --lab
```

which will simulate to run the app both in Android and iOS in the browser side-by-side. Now whenever you save a file within the /www folder, the HTTP server will reload the web page to immediately reflect the changes you made.

Running the BaasBox Back-end

It is recommended to let BaasBox concurrently run on localhost with the default port set up (9000). Make sure your BaasBox configuration is set up correctly and that your respective *config.js* file in your app contains the right settings to successfully connect to the local BaasBox instance. Starting BaasBox means running the start.sh file, which resides in the root folder of the BaasBox instance. Unless you specify different values, the BaasBox will use 'admin', 'admin' and '1234567890' as default username, password and app code. After starting BaasBox, make sure to test the connection by opening the following URL in your browser:

```
http://localhost:9000/
```

BaasBox can be configured in two ways: Some of the settings, such as the app code, can only be defined via a dedicated configuration file, whose path must be provided as a parameter while starting the BaasBox. Other settings can be configured in the BaasBox Console while actually running the BaasBox instance. However for the local development environment, it is not required to change any of the default configuration settings. Note that, since BaasBox is built on top of the Play! framework, BaasBox provides many other inherited configuration options, than those listed in the BaasBox documentation. Therefore, make sure to also study the Play! framework settings, if required.

However with the first start of a local BaasBox instance, you need to make sure all plug-ins are installed correctly. If you restore the BaasBox database from a database back-up, the plug-ins will be installed automatically. However, if you would like to start with a clean install with no contained data, you would have to copy and paste each of the plug-ins into a corresponding plug-in instance. Make sure to keep the names of the plug-ins, since their name is part of the API endpoints they define. While installing the arg.lib plug-in, which acts as a library for the other plug-ins, all necessary Collections will be created automatically. Be sure to enable the 'Plugins Log' at the bottom of the Plug-Ins page before installing the first plug-in by clicking the small eye icon on the right side. This will generate valuable output during setting up the plug-ins.

While developing the app, every once in a while, API calls towards the custom plug-ins failed with an error message similar to "Script is in an invalid state". This bug has been reported, but so far has not been fixed yet. As a work-around, copy and paste the contents of the affected script, remove it, create a new one with the same name and paste the contents again. Be sure to actually have copied the contents before removing the plug-in, since there is no Undo operation for the removal.

Also, make sure to make use of the underscore JavaScript library, which provides several useful methods, that facilitate and accelerate the development of plug-ins tremendously.

Linking the Front-end and the Back-end

There is one configuration file for the front-end, that defines how to connect to the BaasBox instance, and what the collections are called like. Since there are three possible deployment targets, that is, the browser version on the production server, the local development environment, and the wrapper instance, which runs the game in an iOS or Android application, there also three different configuration files required. Therefore, the index.html file of the front-end attempts to read connection configuration settings from the following three files:

```
config-device-prod.js
../config-web-prod.js
../.dev/config-web-dev.js
```

Notice the ordering of the files, which implies the priority of the *../.dev/config-web-dev.js* file: The *../.dev/config-we-dev.js* file will be read successful only by the local development version of the arguegame, which is the only instance where the hidden ./.dev folder should exist. The contained *config-web-dev.js* file should contain the connection settings to connect to your locally running BaasBox instance. If the arguegame is able to load this file, any previously loaded settings from the two other configuration files will be overwritten. Accordingly, please ensure to not deploy the hidden */www./.dev/* folder on the production server.

The *../config-web-prod.js* file contains connection settings to connect to the production BaasBox instance on the production server. By putting the file in the parent folder, the configuration can be shared with other related projects, e.g., ArgueAdmin, and furthermore, it is ensured, that the file is not overwritten or lost if the /www project subfolder is replaced or deleted.

Last but not least, the *config-device-prod.js* file is read by instances where the app is running within an application wrapper, as on iOS and Android.

Generally the configuration file must contain at least the following lines:

```
kBaseURL = '[The Base URL of the application]';
kBaasBoxURL = kBaseURL+'[API Suffix]';
kBaasBoxAppCode = '[App Code]';

kCollectionDomains = "";
kCollectionArguments = "";
kCollectionArticles = "";
kCollectionTopics = "";
kCollectionFeedback = "";
kCollectionEvents = "";
kCollectionVotings = "";
kCollectionRankingOverTime = "";
kCollectionPointsOverTime = "";
kCollectionSpamReports = "";
```

As an example, the first version of the game required the following configuration for the two production configurations:

```
kBaseURL = 'https://argue.ukp.informatik.tu-darmstadt.de';
kBaasBoxURL = kBaseURL+'/argueapi/v1';
kBaasBoxAppCode = 'Argue3252';

kCollectionDomains = "domains";
kCollectionArguments = "arguments";
kCollectionArticles = "articles";
kCollectionTopics = "topics";
kCollectionFeedback = "feedback";
kCollectionEvents = "events";
kCollectionVotings = "votings";
kCollectionRankingOverTime = "rankingOverTime";
kCollectionPointsOverTime = "pointsOverTime";
kCollectionSpamReports = "spamReports";
```

The local development target required the following settings:

```
kBaseURL = 'http://localhost:8100';
kBaasBoxURL = 'http://localhost:8100/v1';
kBaasBoxAppCode = '1234567890';
console.warn("Using config-web-dev.js during development.");
```

Notice how the BaasBox URL is set to be ':8100/v1'. This end-point is a proxy that is served by the local Ionic HTTP server, which redirects every call to the BaasBox running on port 9000.

To test the configuration locally, run the app in the browser and check the JavaScript Console output. If the configuration file for the local environment could be loaded, Console will contain a warning *'Using config-web-dev.js during development.'*, which is exactly what we need. Furthermore, the app will post an alert if it could not successfully connect to the BaasBox. Check the Console then to see which URL the app attempts to connect to.

To deploy the game as bundled package of files, the Cordova target 'browser' has been used. This target, similarly to iOS and Android, bundles all required files, and runs compression and obfuscation steps. The output is a ZIP file, which can be decompressed on any HTTP compatible server.

Note that the *gulpfile.js* file at the *www* subfolder of the project contains a few automation configurations, that compress all JavaScript files into either *App.js*, *Shared.js* and *Vendor.js*, based on in which directory the original file is placed. The gulpfile.js will perform these bundling operations automatically while developing, if the *ionic serve* command has previously been run on the command line. Consequently, since these three bundled files are overwritten by gulp, make sure to always modify the original JavaScript files, instead of the, e.g., App.js file directly.

B Database Collections Used by the Server

The following list names and describes the *Collections* used by the BaasBox server software.

Arguments stores all arguments that have been generated by the users, and that are shown to the users as part of the game. They consist of an array of components, which have a certain component type, and furthermore, the stance of the argument, the author, the language of the argument and additional administrative flags.

Articles stores the articles, which topics can optionally link to. Articles consist of the URL of the article on the web and additional publisher information, such as the base URL and the name.

Domains stores the domains, which topics belong to. A domain consists of a dictionary, that maps a language short code to the title of the domain.

Events holds all events that are generated while using the app to track the user behavior for identifying bottlenecks and future improvements. Next to a timestamp and a session identifier, which is arbitrary, but must be unique between launching and leaving the application, any arbitrary data can be contained.

Feedback stores the feedback submission, that can be sent from within the app. They optionally contain the author's email address, the actual feedback text, and the kind of feedback.

spamReports stores reports for malicious or poor data, which can be sent by the users during a game round. A report can contain any arbitrary data, that provides relevant information.

Topics stores the the topics, which argument entities refer to. A topic entity must define the language, and the actual topic itself.

Voting holds the votes which are submitted by completing any kind of analyzing game round. A voting entity contains a key path and a value, whereas the value embodies the voted value the related object should store at the specified key path, according to the voter.

C Database Link Labels used by the Server

The following list contains the names and the descriptions of the database internal links used by the BaasBox server software.

refersTo is the link between an argument and the topic it has been composed for. An argument links to exactly one topic.

belongsTo links a topic to a specific domain. A topic can link multiple domains at once.

references is the link between a topic and an article, which the topic is mainly about. Every topic references one article at maximum.

votingFor is the link between a vote entity and any arbitrary database entity, which this voting is meant for. There may, in fact should, be many votes to each entity that is votable.

D Questionnaire

The following content represents the questions and statements provided in the questionnaire used for the user study. The response possibilities for the participants have been added in brackets.

```
Argotario User Study

To finish the second and last step of the study, continue with the questions below!

Please answer the following questions unbiasedly and consider each of the provided
options carefully.
/ Für die folgende Befragung würden wir Sie bitten, stets sämtliche
Antwortmöglichkeiten abzuwägen und unbefangen zu antworten.

1. On a scale from 1 to 5, how would you rate the following questions?
Auf einer Skala von 1 bis 5, wie würden Sie die folgenden Fragen beantworten?
  [I strongly agree / Stimme voll zu (1), I agree / Stimme zu (2),
  Neither agree nor disagree / Teils teils (3), I disagree / Stimme nicht zu (4),
  I strongly disagree / Stimme gar nicht zu (5), Not specified / Keine Angabe (0)]

  The game was fun in general.
  Das Spiel hat mir insgesamt SpaSS bereitet.

  I can imagine to play the game again once it is finished.
  Ich kann mir vorstellen, das Spiel im fertigen Zustand erneut zu spielen.

  I can imagine to frequently play the game once it is finished.
  Ich kann mir vorstellen, das Spiel im fertigen Zustand regelmäSSig zu spielen.

  I trust the rating system of the game.
  Ich habe Vertrauen in das Bewertungssystem des Spiels.

  I temporarily considered leaving the game.
  Ich hatte zwischenzeitlich überlegt, das Spiel vorzeitig zu verlassen.

  I consider myself to be an experienced user of online offerings.
  Ich halte mich für einen erfahrenen Nutzer von Online-Angeboten.

  I could imagine to play this game on a smartphone.
  Ich könnte mir vorstellen, das Spiel auf einem Smartphone zu spielen.

  In the finished game I would like to be able to compare myself with acquaintances
  and friends.
  Ich würde mich gerne im fertigen Spiel mit Bekannten und Freunden vergleichen
  können.
```

Other Feedback (optional)
Sonstige Rückmeldungen (optional)
[Free text]

2. What is the highest educational level you have achieved?
Was ist der höchste Bildungsgrad den Sie bisher erlangt haben?
[Select one of the options]

Less than elementary school / Weniger als Grundschule

Elementary school / Grundschule

Secondary Modern School Qualification / Hauptschulabschluss

Secondary School Certificate or O-level
/ Realschulabschluss oder Mittlere Reife

Advanced Technical Certificate or High school graduation
/ Fachhochschulreife oder Abitur

Bachelor Degree / Bachelor-Abschluss

Master Degree / Master-Abschluss

Dimploma / Diplom

Doctoral Studies or PhD / Promotion

3. If you could change one thing in the game, what would it be, and why?
Wenn Sie eine Sache am Spiel ändern könnten, was wäre es, und weshalb?
[Free text]

4. Experience / Erfahrung
How would you rate your experiences with computers / IT?
Wie würden Sie Ihre Erfahrung mit Computern / IT einschätzen?
[Very Poor / Sehr schlecht (1), Poor / schlecht (2), Fair / MittelmäSSig (3),
Good / Gut (4), Very good / Sehr gut (5), Not specified / keine Angabe (0)]

Very Poor / Sehr schlecht
Poor / schlecht
Fair / MittelmäSSig
Good / Gut
Very good / Sehr gut
Not specified / keine Angabe

5. What is your age? / Wie alt sind Sie?
[Select one of the options]

Under 18 / Unter 18

18-20
21-29
30-39
40-49
50-59
Over 60 / Über 60

6. Many thanks for participating in our user study for our game "Argotario". If you would like to be notified once the full game becomes available you can provide your email address.
Wir bedanken uns herzlich für Ihre Teilnahme an der Benutzerstudie zum Spiel "Argotario". Sollten Sie wünschen, benachrichtigt zu werden, sobald das fertige Spiel verfügbar ist, können Sie Ihre Emailadresse hinterlassen.
[Free text]

E Questionnaire User Data

The following data represents the unaltered response data resulting from the online questionnaire based interrogation for the user study.

E.0.14 Statements

Statement	**1**	**1**	**1**	**2**	**2**	**2**	**3**	**3**	**3**	**4**	**4**	**4**
Group	$\bar{x}$	A	B	$\bar{x}$	A	B	$\bar{x}$	A	B	$\bar{x}$	A	B
Responses	36	20	16	35	20	15	35	19	16	34	19	15
I Strongly Agree	1.11	2	0	1.00	1	1	0.00	0	0	2.56	3	2
I Agree	8.67	10	7	5.57	6	5	0.46	0	1	9.68	11	8
Neither	4.44	4	5	5.14	6	4	3.54	4	3	2.68	4	1
I Disagree	3.00	3	3	4.00	4	4	7.00	7	7	1.88	1	3
I Strongly Disagree	1.00	1	1	2.14	3	1	6.63	8	5	0.44	0	1
Statement	**5**	**5**	**5**	**6**	**6**	**6**	**7**	**7**	**7**	**8**	**8**	**8**
Group	$\bar{x}$	A	B	$\bar{x}$	A	B	$\bar{x}$	A	B	$\bar{x}$	A	B
Responses	35	19	16	36	20	16	35	19	16	33	19	14
I Strongly Agree	2.00	2	2	3.67	5	2	2.46	2	3	3.00	3	3
I Agree	6.00	6	6	8.11	9	7	7.17	9	5	5.12	3	8
Neither	2.09	3	1	3.33	2	5	3.37	2	5	4.30	6	2
I Disagree	4.46	4	5	3.11	4	2	2.09	3	1	1.73	3	0
I Strongly Disagree	3.09	4	2	0.00	0	0	2.54	3	2	2.73	4	1

Table E.1: Responses to the questionnaire statements 1 to 8 of the user study, by group and by weighted averages.

Statement	**1**	**1**	**1**	**2**	**2**	**2**	**3**	**3**	**3**	**4**	**4**	**4**
Group	$\bar{x}$	A	B	$\bar{x}$	A	B	$\bar{x}$	A	B	$\bar{x}$	A	B
Responses	36	20	16	35	20	15	35	19	16	34	19	15
I Strongly Agree	5.6	10.0	0.0	5.7	5.0	6.7	0.0	0.0	0.0	14.7	15.8	13.3
I Agree	47.2	50.0	43.8	31.4	30.0	33.3	2.9	0.0	6.3	55.9	57.9	53.3
Neither	25.0	20.0	31.3	28.6	30.0	26.7	20.0	21.1	18.8	14.7	21.1	6.7
I Disagree	16.7	15.0	18.8	22.9	20.0	26.7	40.0	36.8	43.8	11.8	5.3	20.0
I Strongly Disagree	5.6	5.0	6.3	11.4	15.0	6.7	37.1	42.1	31.3	2.9	0.0	6.7
Statement	**5**	**5**	**5**	**6**	**6**	**6**	**7**	**7**	**7**	**8**	**8**	**8**
Group	$\bar{x}$	A	B	$\bar{x}$	A	B	$\bar{x}$	A	B	$\bar{x}$	A	B
Responses	35	19	16	36	20	16	35	19	16	33	19	14
I Strongly Agree	11.4	10.5	12.5	19.4	25.0	12.5	14.3	10.5	18.8	18.2	15.8	21.4
I Agree	34.3	31.6	37.5	44.4	45.0	43.8	40.0	47.4	31.3	33.3	15.8	57.1
Neither	11.4	15.8	6.3	19.4	10.0	31.3	20.0	10.5	31.3	24.2	31.6	14.3
I Disagree	25.7	21.1	31.3	16.7	20.0	12.5	11.4	15.8	6.3	9.1	15.8	0.0
I Strongly Disagree	17.1	21.1	12.5	0.0	0.0	0.0	14.3	15.8	12.5	15.2	21.1	7.1

Table E.2: Proportions (%) of responses to the questionnaire statements 1 to 8 of the user study, by group and by weighted averages.

F Questionnaire Response Data Normality Tests

The following tables represent the results of the Shapiro-Wilk tests, that have been performed for each of the questionnaire statement response data, and for each of the two groups A and B respectively.

Statement	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
Group	A	B	A	B	A	B	A	B
n	20	16	20	15	19	16	19	15
Mean	2.55	2.88	3.10	2.93	4.21	4.00	2.16	2.53
SD	1.05	0.96	1.17	1.10	0.79	0.89	0.76	1.19
W	0.87	0.83	0.91	0.92	0.79	0.86	0.84	0.84
$W_t(p = .05)$	0.90	0.89	0.90	0.88	0.90	0.89	0.90	0.88
Normality	No	No	Yes	Yes	No	No	No	No
Statement	**5**	**5**	**6**	**6**	**7**	**7**	**8**	**8**
Group	A	B	A	B	A	B	A	B
n	19	16	20	16	19	16	19	14
Mean	3.11	2.94	2.25	2.44	2.79	2.63	3.11	2.14
SD	1.37	1.34	1.07	0.89	1.32	1.26	1.37	1.03
W	0.89	0.88	0.83	0.89	0.85	0.90	0.91	0.77
$W_t(p = .05)$	0.90	0.89	0.90	0.89	0.90	0.89	0.90	0.87
Normality	No	No	No	Yes	No	Yes	Yes	No

Table F.1: The results of the Shapiro-Wilk tests on the statement response data. Per test, the Null-hypothesis claims that the given data is normally distributed. The Null-hypothesis has to be rejected, if W is smaller than the critical value W_t. This is the case for 5 of the 16 distributions. In order to perform a t Test, both compared samples must be normally distributed. For simplicity reasons, statement 2 has been performed as all others, using a Mann-Whitney-U test.

G Composed Arguments During User Study

The following list contains all of the unmodified arguments generated by study participants.

CLAIM Yes we should do it. PREMISE It helps to do mistakes on other animals before people get hurt

CLAIM We should use animals as it is necessary to test. PREMISE It's better to test on animals than humans

CLAIM Animal testing is something totally unethical and should be banned. PREMISE Animal testing is something extremely cruel and human beings do not have the right of torturing animals for medical purposes. PREMISE Medicaments can be tested nowadays using technology for simulating what would actually happen if the medicament was used.

CLAIM Animals do not have souls PREMISE God has only given souls to mankind PREMISE The soul is the most important element of life, it allows communion with God.

CLAIM Animal testing can have serious impacts on researching new technologies for curing diseases in humans. PREMISE Testing new medications and techniques on animals first reduces the risk for humans.

CLAIM Animal testing for medical purposes should be allowed. PREMISE Animal testing is crucial for the development of life-saving medications.

CLAIM It is the only way to get reliable results PREMISE All other known methods such as testing on human giving them small amount of the medicine or computer simulation either can't test what happens with big dose or need initial data to train the algorithm

CLAIM Animal lives are less important than human lives PREMISE We kill animals for food, why not for healing humans?

CLAIM Tests can be also conducted by numerical simulations. PREMISE State-of-the-art CPU power enables complex simulations of bio chemical processes, thus replacing test on live animals.

CLAIM We should allow it PREMISE It helps improving human medicin

CLAIM Animal testing helps to improve medical care for everybody. PREMISE Without testing them, new health care devices cannot be introduced.

CLAIM It helps to sacrifice animals. PREMISE Tests are needed.

CLAIM If a serious desease can be cured with the help of animal testing, we shpuldn't hesitate. PREMISE Every child with a genealogical desease needs to actually experience a future.

CLAIM It allows to get useful data for human research PREMISE It is as close to do human testing as possible

CLAIM Testing animals should be approved. PREMISE It can provide mankind new ways to for healing ourselves.

CLAIM a a PREMISE b b b, bbb ba

CLAIM Animal testing is a good alternative to human testing. PREMISE Most negative effects of products can be discovered. PREMISE Without animal testing humans would have to suffer from these side effects.

CLAIM Animal testing for medical purposes should be allowed. PREMISE If there is no alternative instead of testing on humans immediately, this should be a possible fall-back option. However, animals should be treated with all due respect.

CLAIM We should allow animal testing! PREMISE To procede with technology and progress

CLAIM Yes PREMISE Yes, better than testing with human

CLAIM Yes PREMISE Because that will allow us to quickly test the new medicine and therefore more people could be saved

CLAIM Yes we should allow but without harming the animals ! PREMISE Before publishing a medicine for humans, it should be testet on animals, to see how they react towards the medicine. e.g. rats

CLAIM better testing new medical products on animals than on humans PREMISE animals react similar to medical products, if animals die from a product -> dangerous medical product detected, no humans harmed

CLAIM No PREMISE Because they don't know what we do with them.

CLAIM It has to be balanced, but for live saving drugs It should be allowed on non-endangered animal species. PREMISE Testing on a living human will result in more complications of civil nature beside usual health problems. PREMISE For a life saving drug it is better to have tested on some living being other than humans. PREMISE But used animal must not be endangered species.

CLAIM Humans are no Animals PREMISE They have a different organism

CLAIM Animal testing leads to medicaments with fewer harmful side effects. PREMISE Humans have many similar parts, like cells, organs and other body system. Thus, side effects will likely afflict animals as well as humans and can be detected before medicaments are tested with humans.

CLAIM We should allow animal testing for medical purposes. PREMISE Before testing medication on humans, possible contraceptions should be well understood.

CLAIM Animal testing helps saving lifes. PREMISE It is often possible to use relatively unintelligent animals as specimen, so that proper laboratory conditions can be set up and according processes can be followed in drug creation without harming humans or intelligent animals or exposing them to uncontrollable risks.

CLAIM For a first steps in basic research, animal testing for medical reasons is more efficient than human testing. PREMISE Lab animals, such as rats and mice have a reproductivity cicle far shorter than humans, so you can examine effects over lifetime and on following generations in a shorter period of time.

CLAIM We need animal testing, even tough it is horrible for the animals. PREMISE Although it is nothing I personally would wanna do, I think animal testing are a crucial part of testing modern drugs and medical treatments.

CLAIM Of course we need it! PREMISE Many of the current medical drugs are based on animal testing based research.

H Events Tracked by the Game During User Study

The following list names the events that are recorded while using the software.

OpenedApp is fired whenever the game is visited. At that point, the visitor is not necessarily authenticated, so the user that created the event may be unknown.

OpenedAuthentication is fired when the authentication window is shown.

SuccessfulAuthentication is fired when the user could successfully be authenticated.

OpenedLevel is fired when a game level window has been opened.

OpenedRound is fired when a game round has started.

SkippedRound is fired when the user decided to skip a game round.

FinishedRound is fired when the user finished a game round, no matter if it has been skipped or not.

RoundDuration is fired when the users have just submitted their response data.

QuitApp is fired when the user leaves the game website. It is important to note that the reliability of this event weak, since the web page may have been closed before the asynchronous communication with the back-end to store the event has succeeded.

QuitRound is fired when the user decided to quit the level the current active game round is part of.

QuitLevel is fired when the user decided to quit the current level.

OpenedUserProfile is fired when a user profile window has been opened.

OpenedAccount is fired when the account window has been opened.

OpenedWorld is fired when a game world window has been opened.

OpenedForeignProfile is fired when a profile window of a another user has been opened.

FinishedLevel is fired when a level has finished, no matter if the level has completely or successfully finished.

FollowedUser is fired when the user followed another user.

FailedAuthentication is fired when the visitor could not be authenticated properly.

AbortedAuthentication is fired when the visitor aborted the authentication.