Improving Web Accessibility Using A Computer Game

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

in

Information Technology

under the Supervision of

Mr. Arvind Kumar

By

Arushi Sharma

111404

to

Jaypee University of Information Technology

Waknaghat, Solan – 173234, Himachal Pradesh

CERTIFICATE

This is to certify that project report entitle **Improving web accessibility using a computer game** , submitted by **Arushi Sharma** in partial fulfillment for the award of degree of Bachelor of Technology in Information Technology to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Supervisor's Signature :
Supervisor's Name : Mr Arvind Kumar
Date :

ACKNOWLEDGEMENT

I would like to express my gratitude to all those who gave us the possibility to complete this project. I want to thank the Department of CSE & IT in JUIT for giving us the permission to commence this project in the first instance, to do the necessary research work.

I am deeply indebted to my project guide Mr Arvind Kumar, whose help, stimulating suggestions and encouragement helped me in all the time of research on this project. I feel motivated and encouraged every time I get his encouragement. For his coherent guidance throughout the tenure of the project, I feel fortunate to be taught by him, who gave me his unwavering support.

We are also grateful to **Mr.Amit Singh(CSE Project lab**) for his practical help and guidance

Arushi Sharma

Table of Content

List of Figures

S.NO. TITLE. . .. . PAGE NO.

List of Tables

Abstract

Images on the Web present a major accessibility issue for the visually impaired, mainly because the majority of them do not have proper captions. This paper addresses the problem of attaching proper explanatory text descriptions to arbitrary images on the Web. To this end, we introduce a new game, an enjoyable computer game that collects explanatory descriptions of images. People play the game because it is fun, and as a side effect of game play we collect valuable information. Given any image from the World Wide Web, this game can output a correct annotation for it. The collected data can be applied towards significantly improving Web accessibility. In addition to improving accessibility, this is an example of a new class of games that provide entertainment in exchange for human processing power. In essence, we solve a typical computer vision problem with HCI tools alone. The Web is not built for the blind. Only a small fraction of major corporate websites are fully accessible to the disabled, let alone those of smaller organizations or individuals [5]. However, millions of blind people surf the Web every day, and Internet use by those with disabilities grows at twice the rate of the non-disabled .One of the major accessibility problems is the lack of descriptive captions for images. Visually impaired individuals commonly surf the Web using "screen readers" .We set our goal to assign proper descriptions to arbitrary images. A "proper" description is correct if it makes sense with respect to the image, and sufficient if it gives enough information about its contents. Rather than designing a computer vision algorithm that generates natural language descriptions for arbitrary images (a feat still far from attainable), we opt for harnessing humans. It is common knowledge that humans have little difficulty in describing the contents of images, although typically they do not find this task particularly engaging. On the other hand, many people would spend a considerable amount of time involved in an activity they consider "fun." Thus, like the ESP Game we achieve our goal by working around the problem, and creating a fun game that produces the data we aim to collect .We therefore introduce a game which, as a side effect, generates explanatory sentences for randomly chosen images.

Problem Statement

To create an online game in order to improve the web accessibility for visually impaired people.

Chapter 1: Introduction

1. Introduction

In the last few decades, computers have turned from academic curiosities into industrial machines and finally into ubiquitous devices that penetrate nearly all aspects of modern life. Many day-to-day things we take for granted would simply be impossible without the immense computational capabilities of today's computers. There are airplanes that would not be able to sustain flight [4], if not for the constant meticulous corrections of highly complex control systems. Computers can combine and analyze data from thousands of measurement stations all over the world to create increasingly reliable weather forecasts Many trains today even run without human drivers [34].This list could be extended to fill a thesis of its own. Suffice it to say that overall computers are faster, more accurate and less error-prone than human intuitively in solving a wide variety of problems. However, the more difficult and diverse these solved problems are, the more apparent the unsolved ones become: Understanding the meaning of a spoken sentence, distinguishing a cat from a dog or describing the contents of a picture is no challenge at all for an average five-year-old, but is currently challenging for even the most advanced computer systems.

One way to approach these somewhat surprising deficiencies is to try and find ways in which computers can "think" in a more human way. Examples of this include neuronal networks and other machine learning approaches. Similarly, one can try to transform these problems to be more accessible for computers and find better algorithms.

A different approach – the one taken in this work – is to simply accept the different "skill sets" of humans and computers. The goal is then to try and find ways to harness the special capabilities of the human mind and combine them with the skills of machines.

1.1. Early Experiments in Human Computation

Von Ahn calls this process of harnessing the users' intelligence Human Computation [24]. A large number of alternative terms are used to describe collaborative problem-solving, including Crowdsourcing, Peer Production, Collective Intelligence and Crowd Wisdom. The first experiments in utilizing human intelligence presented tasks to their users directly. Only later these tasks were turned into the games which are the focus of this thesis. Nevertheless,the early systems form the base from which these games evolved.

1.1.1. Open Mind Initiative

One of the first attempts to use the mental capabilities of laypersons to solve scientific problems was the Open Mind Initiative. The Open Mind Initiative was founded by Stork to improve intelligent systems such as text- or pattern recognition. Most of these systems heavily rely on large sets of training data, the acquisition of which can be difficult – after all, one cannot use computers to create them.

Stork proposes a three-tiered structure for the creation of intelligent systems. Firstly, domain experts provide the fundamental models and algorithms. The creation of training data for these systems is too labor intensive to be performed by a small group of specialists.

Stork therefore propose to allow large groups of laypersons to provide the necessary training data as the second tier. Finally, the Open Mind Initiative provides a framework for recruiting and motivating volunteers and collecting the data using the Web. Since training datasets should usually be as large as possible, it is desirable to recruit as many volunteers as possible. Stork names a number of incentives for motivating participants. These incentives include altruism and public recognition. Stork furthermore proposes to design the interfaces of the data collection systems as games. The experience of playing these games should be pleasurable, thus motivating users to participate. This is the first mention of what would later be called Games With A Purpose by von Ahn. However, Stork and other members of the Open Mind Initiative did not focus on the creation of games in their further research. The Open Mind Initiative will therefore not be discussed in more detail in this work.

1.2. Image Labeling Games

After the creation of the CAPTCHA, von Ahn recognized another important fact. Aside from the time wasted through necessary evils like CAPTCHAs, people love to waste time voluntarily. Von Ahn estimates that several billion human-hours are spent each year playing solitaire2. If properly channeled, this energy could be used to tackle previously unsolved large scale computation problems. Von Ahn proposes to do this by creating so called Games With A Purpose (for better readability, the abbreviations "GWAP" and "GWAPs" will be used to distinguish "Game With A Purpose" and "Games With A Purpose" respectively).

The first application of a GWAP demonstrated by von Ahn and Dabbish was image labeling. Image labeling is the process of creating textual meta-data describing the content of an image. These meta-data are usually divided into single attributes called labels or tags (both terms will be used synonymously from now on). These attributes can include both abstract information, such as the type of the image (e.g., "photograph", "drawing"or "painting"), as well as concrete descriptions of the contents depicted in it (e.g., "small dog" or "tree"). Automating this process is a highly researched subject in the field of machine learning, but current automatic solutions cannot rival the accuracy of human created descriptions.

Manually annotating images can be feasible for a private collection of photographs, but current public image databases often contain millions of different images. Paying workers to manually label these images is usually not financially feasible and relying solely on volunteers raises the issue of how to motivate the participants. To solve this problem, von Ahn and Dabbish created the so-called ESP Game in which two players collaboratively create textual descriptions of images.

1.2.1. Applications of Image Labeling

There are several reasons for creating accurate textual descriptions of images. Arguably the most important application for image labeling is image retrieval. Selecting stored images in a systematic fashion is one of the fundamental functions of any image database. As a special case, the Web can be considered a very large image database. Usually, queries to such a database refer to the content of the stored images. For example, a user might want to access an image depicting a specific object. The following

paragraph lists a number of image retrieval methods and explains why image labeling is important.

One possibility for handling such requests is query by example. The user provides a sample image and the database returns images that are similar to this image by some metric. An implementation of this technique is the TinEye image search engine [tin]. One issue of this method is that the user has no method for specifying which trait of the sample image is important (images contain colors, shapes and specific objects, all of which could be, but do not need to be important in a query). Furthermore, this approach requires that the user is already in possession of a suitable example image. Lastly, query by example does not allow searching for generic classes of images (for example, images containing any kind of "building").

Another approach for creating visual queries is through *sketches*. Users can create simple drawings and the image database should return objects that match this sketch in shape and/or color. For this approach to return accurate results, users need to be able to create detailed and accurate drawings. Existing implementations of this query approach (see the works of Eitz et al., Chen et al., Gavilan et al.) are designed for casual applications such as creating collages from personal images. This suggests that sketching alone is not sufficient for accurate selection of images. Chen et al. circumvent this problem, by combining querying by shape with textual queries [3].

Using textual queries is arguably the most common image retrieval technique. Many image database systems (e.g., Google Images [goob] or Prometheus [pro]) primarily rely on textual descriptions for querying images. This means that each image is assigned a symbolic description against which any entered query is matched. Since the content of the images does not need to be considered while processing the queries, database requests can be performed using fast and proven algorithms for document retrieval. Furthermore, users can describe the image they are looking for without having to provide examples or needing to be able to draw. However, the results of textual queries will only be accurate and reliable if the descriptions of the images reflect their actual content.

In some cases, image descriptions can be created automatically without analyzing the content of the images. On the Web, images are often accompanied by text. By considering spatial proximity between images and texts, potentially relevant keywords for images can be extracted from Web-sites. Nevertheless, this extraction is error-prone and fails completely for images without text. Thus, image labeling is highly important for image retrieval.

In turn, an important application for image retrieval is the filtering of content. In some cases, restricting the access of users to certain materials is desirable. For example, parents might want to allow their children access to the Web, but protect them from pornography and violence. Appropriately labeled images could solve this issue reliably.

1.2.2. Goals of Image Labeling

As the arguably most important application, image retrieval defines the desirable properties of image labeling. Therefore, the labels for an image should reflect the structure of common queries. Therefore, the existence of a sufficient number of generic labels for each image is crucial. On the Web, a large number of generically labeled images already exist – a search for "car" on Bing (formerly Live) Image Search [bin] returns 150 million results.

In contrast, queries using highly specific labels such as a car make and color return too few or incorrect results. Therefore, it is also important that images are labeled with as many specific tags as possible.

In combination, the above statements lead to the following conclusion: The best results in image retrieval will be achieved if images are labeled with a comprehensive set of labels, including both generic and specific ones.

1.3. Motivation

Tag Diversity. As discussed above, the goal of image labeling is to create a comprehensive set of descriptions for any given image.

The basic mechanics of existing games are designed to enforce correct labels, but not necessarily comprehensive ones. Probably the most important example for this is the ESP game. It is important because the problem is prominent in the ESP Game and because it is one of the few implementations that are used productive systems.The ESP game pairs two random players who have to agree on a description of an image. Thus, each player does not know anything about her partner and thus has to assume she is not an expert in a given domain. Therefore, players are more likely to achieve a match if they enter a generic term as opposed to a specific ones. This has been proven using game theory by

Jain and Parkes. The basic game design of the ESP game would thus yield specific descriptions only with a small likelihood.

Current Solution. As a solution,we propose the use of so called taboo words. Any tag entered for a given image is added to a list (such a list is kept for each image). If a tag is already on the list, the number of times it has been assigned is counted. Once this number of occurrences of a tag exceeds a given threshold, the tag is added to the list of taboo words for the given image. This list is shown to players in later rounds and the players cannot enter the displayed labels any longer. Taboo words are meant to force players to use different and eventually more specific terms.

Since the results of this game are not publicly available, the authors extracted the displayed taboo words, which reflect the labels assigned to an image.

Despite their restriction to taboo words, we found a large number of redundant and generic labels in the extracted data. Furthermore, many tags where highly correlated. As an example, the authors mention that 68% of all images labeled with “clouds” were also labeled with “sky”. Weber et al. therefore argue that this kind of data does not necessarily need to be created by human players. To prove their point, they implemented a software that successfully plays the ESP Game. This is somewhat paradoxical, since the main objective of the game (i.e., labeling images) cannot yet be achieved reliably by computers.

1.4. Proposed New Game and Related Approaches

Aside from the use of taboo words, a number of image labeling games have been created using various strategies for ensuring tag diversity.

Like most of the games mentioned above, the proposed game is designed to be collaboratively played by two players. A grid of images is shown to both players. The order of the images is randomized for each player. In the grid of the first player, one image is highlighted. Their goal is to describe this image to their partner. The second player must then select the right image from their grid. By selecting the images in the grid by tag similarity, it is possible to force players to contribute new information. The basic mechanism of game is quite similar to that of Verbosity, created by von Ahn et al.However, Verbosity is not an image labeling game and lacks the concept of using input similarity to increase tag diversity.

Scope of this Thesis.

Following the arguments mentioned above, image labeling games should be designed with both correctness and diversity of tags as their fundamental goals. The aim of this thesis is to present and evaluate a new design for an image labeling game.

Chapter 2: Related work and the state of the art

2. GWAP

The following Section contains a short description of the history of Games With A Purpose and an overview of the state of the art of GWAPs. The discussed related works include both games used image labeling as well as games designed for other purposes.

2.1. The Evolution of Games With A Purpose

As described in the previous Section, GWAPs evolved from data gathering systems. Their main goal is to motivate players to contribute time and effort for solving scientific problems.These games should thus be designed to be fun to play to incite as many users as possible.At the same time, GWAPs must perform some kind of calculation while being played. This *purpose* of the game does necessarily have to be communicated to the player, but is the essential characteristic that separates GWAPs from games that only aim to entertain. Instead of being executed by computer systems, these algorithms are being "run" on the highly distributed minds of the players. Other than that, the same considerations as for any other algorithm apply. The game should be correct, meaning that irrespective of the individual players' intentions, the relationship between input and output of the game must conform to a given specification. Von Ahn suggests to achieve this by making correctness of the output the winning condition for a given game. At first, this sounds like a paradox. The problems tackled by Games With A Purpose are those to which the solution cannot be computed. This usually means that the correctness of a solution cannot be verified automatically either.However, in many instances the players themselves can verify a solution. In all cases, the players are required to be independent. Depending on the design of the game and the types of information involved, the players may also need to be unable to communicate outside of the game. If these preconditions apply, there are several possibilities of enforcing correct output. Von Ahn proposes two general verification schemes [24] and later added a third one [26].

• In *Output-agreement* games, all players involved assume the same role. Their output will be accepted only if both players independently agree on it.

• In *Inversion-problem* games, the players assume different roles. The first player transforms a given input into an intermediate output. This output is passed to the second player. The second player's task is to reconstruct the first player's input. If she succeeds, the intermediate output is accepted

• In *Input-agreement* games, both players transform a given input into an intermediate output. In contrast to inversion-problem games, both players can see the intermediate output of their partner. They then try to agree on whether their respective inputs are identical

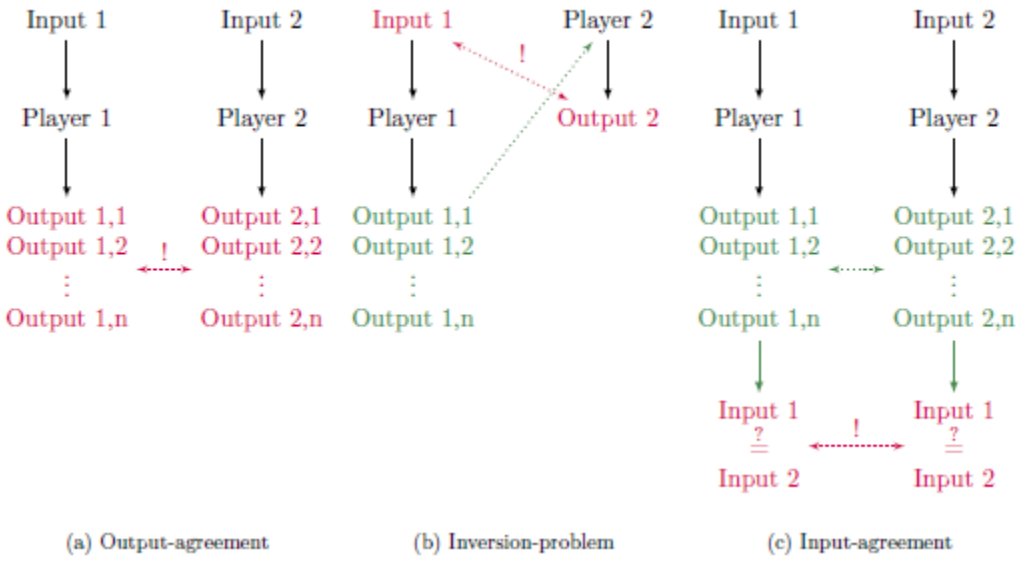


Figure 2.1.1

Output verification schemes for GWAPs
Any data shared by both player is green; Data that is used for verification and has to be identical for the players to succeed is magenta

Despite the diversity of existing Games With A Purpose, almost all of them use variations of the described methods. The correctness of the output cannot be guaranteed using any of the described mechanisms. For example, both players could make a mistake and still agree on the erroneous output. Alternatively, malicious users could circumvent the communication barrier and deliberately create false .These and other scenarios which lead to incorrect data cannot be avoided entirely. However, if the aforementioned

preconditions apply and a verification method is used, the game design can be adjusted to return correct solutions with an arbitrarily high probability.

Aside from being correct as described above, an algorithm should also be efficient. For computer algorithms, this means that the time and memory required for finding a solution to a given problem should be as low as possible. This requirement is applicable to the implementation of a game, but not necessarily to the game itself. To evaluate the efficiency of Games With A Purpose, von Ahn proposes a combination of two measures

• The *throughput* of a game measures the average number of problems solved per player in a given time span. This is an indication of how effective players are when they are playing the game. However, a game with a high throughput can still be inefficient if no one is willing to play it. The enjoyability of a game is a very important aspect of its quality and is not captured by the measure of throughput.

• The *Average Lifetime Play* (ALP) of a game is defined as the overall average time one single player spends playing the game. Since players are less likely to quit playing a fun game, this measure is indicative of the enjoyability of a game.

Verification \ Game type	Image Labeling	Other
Output-agreement	ESP Game KissKissBan Matchin Picture This Magic Bullet	People Watcher OntoGame
Inversion-problem	Peekaboom Phetch Karido	Verbosity
Input-agreement		TagATune Intentions
Hybrid	PhotoSlap	Foldit CityExplorer Page Hunt

Table 2.1.1

Games With A Purpose presented in this thesis

2.2. Image Labeling Games

Although image labeling appears to be a relatively narrow field of application for GWAPs, a number of different image labeling games have been designed. The earliest

and most popular image labeling game is the ESP Game (see next Section) developed by von Ahn and Dabbish. Several other games have been developed (by the group around von Ahn as well as other groups) as extensions or improvements to the ESP Game. Additionally, there are a number of games that use radically different game mechanics. The following Section gives an overview of these existing implementations and their respective gameplay.

2.3 Secure Distributed Human Computation

Gentry et al. propose a framework for what they call *secure distributed human computation* (SDHC). An SDHC system is constituted by four parties:

• A set of *problem suppliers*, who have a set of problems they need solved. This is usually a company.
• A set of (human) *problem solvers*, who are capable of solving such problems. These are the primary users of the system – the players, in case of a game.
• A *distributor* who assigns problems to solvers who are willing to process them.
• A set of *store fronts*, which are venues (usuallyWeb-sites), at which solvers can contact the distributor to acquire tasks. The store fronts usually provide some kind of service to the solvers as a reward for processed problems.Gentry et al. divide the interaction between these entities into a registration and an operation phase. The authors furthermore propose a threat model for both phases. In the registration phase, the primary attack lies in the (possibly automated) creation of multiple user accounts. In the operation phase, the primary attack lies in the introduction of false answers into the system by malicious users From this, the authors derive a probabilistic analysis of the reliability of a SDHC system.
To achieve any reliability in human computation, each problem should be processed by multiple users. Under the assumption that there is only one correct answer for each problem,the answers of the users must be combined to reach an overall conclusion. Gentry et al.compare majority voting and Bayesian inference for achieving this. In a majority vote, the answer given by the highest number of users is elected to be the correct one. In contrast, Bayesian inference takes into account the previous results of a user, thus putting more trust into answers by users that have given right answers before. Gentry et al. prove that majority voting yields the most reliable results in the presence of malicious users. Assuming that malicious users only provide false answers and the other users are

correct 75% of the time, only two thirds of all players need to be honest for majority voting to succeed. If these conditions are met or exceeded, the reliability of a solution increases exponentially with number of users assigned to a given problem.

Chapter 3:Algorithms

3.1 Optimal puzzle selection algorithm

The idea is there are optimal r labels per labeled image in system

We group images into 3 sets

1. Contains all the images that have not been played
2. Contains all the images that have been played at least once, but less than r rounds
3. $P_2 = P - P_0 - P_1$

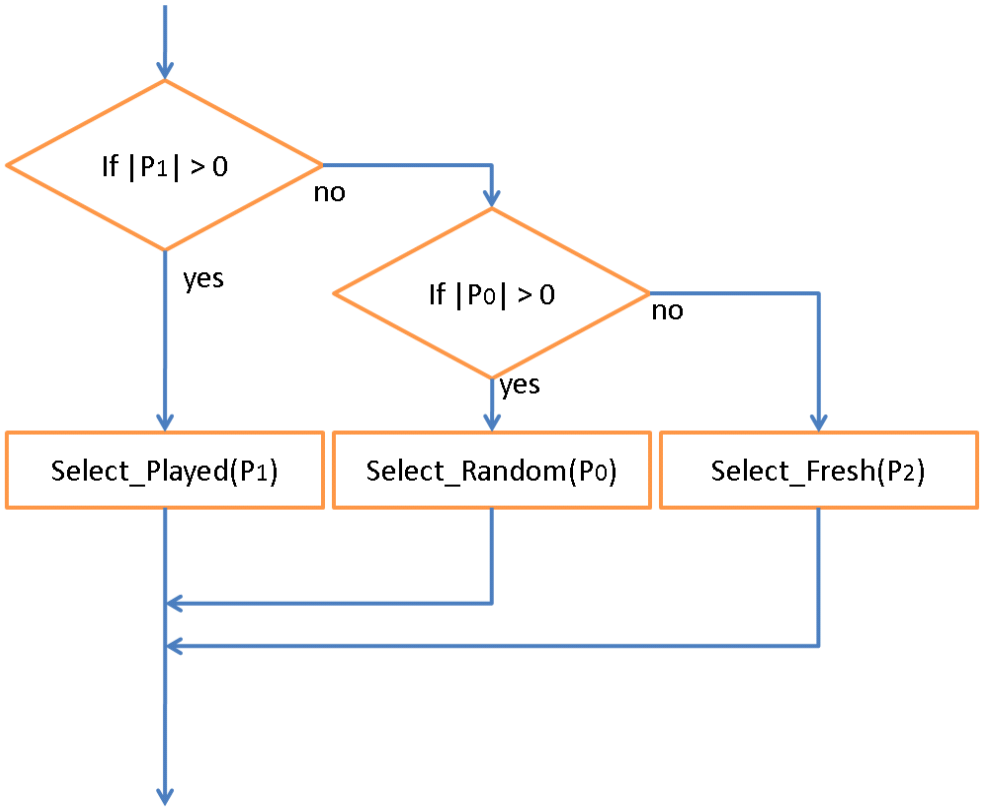


Figure3.1.1 Flowchart of OPSA

3.1.1 Simulation of algorithm

Observation

T vs. r → OPSA

T vs. System gain → 3 strategies

Graphs:

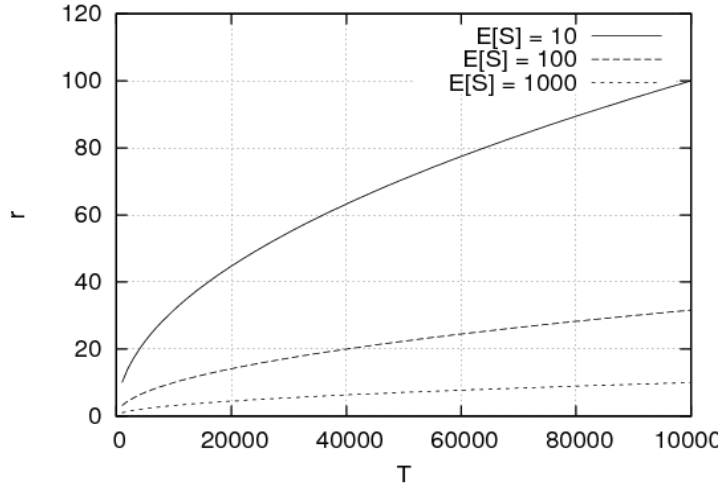


Figure 3.1.1.1 The above graph is T vs. r

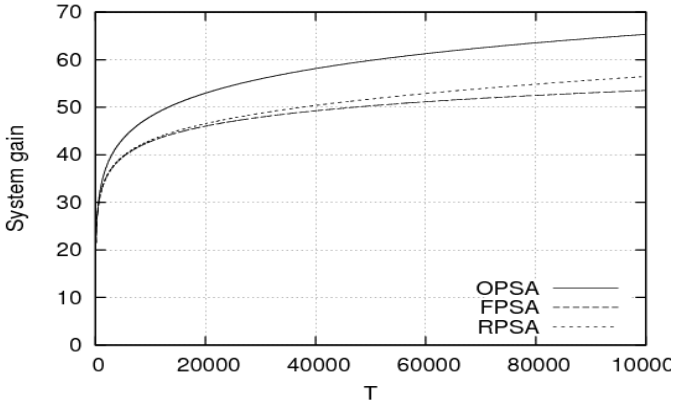


Figure3.1.1.2 The above graph is T vs. System gain

3.1.2 Discussion

OPSA is superior to RPSA & FPSA in the simulation

A systematically & thorough study to verify the purposed strategies in real systems is highly desirableTo this end, we decide to implement the game system

3.2 Flow Chart

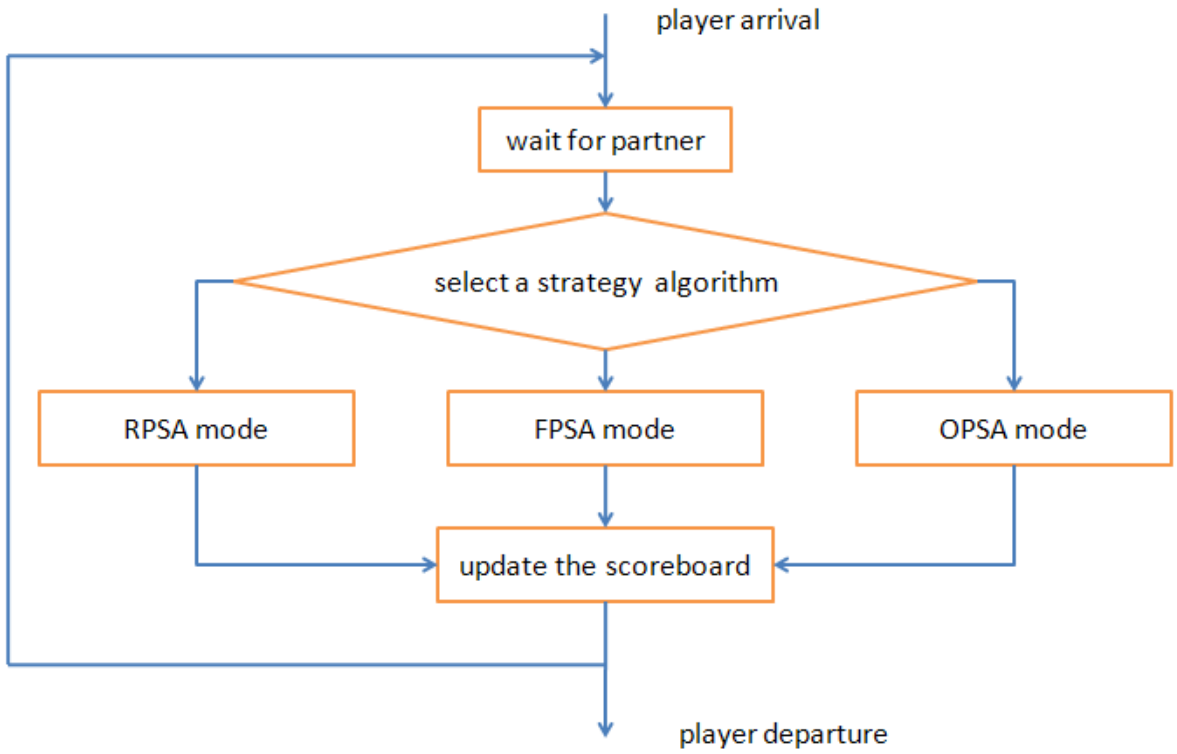


Figure 3.2.1 The strategy selection process gives priority to the strategy that has been used least in terms of **the # of rounds played previously**

3.3 Score System

The score system is used to measure the quality of the agreed words

The quality of each the agreed word should depends on its popularity

high frequency → low quality → low score

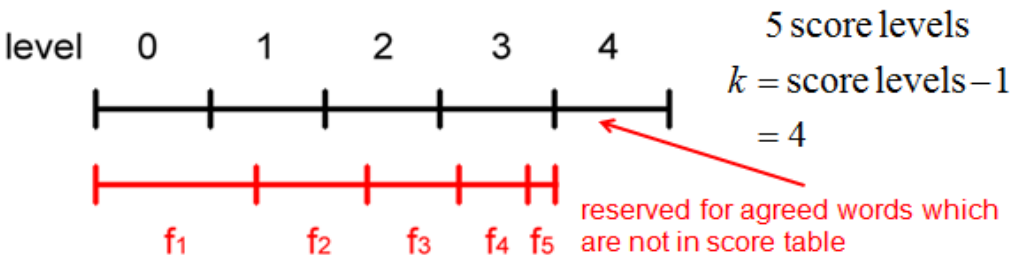
low frequency → high quality → high score

$w_1, w_2, \cdots, w_n$ n words in the score table

$f_1 \geq f_2 \geq \cdots \geq f_n$ frequency of the word

$score(w_i) = L_i \, S_{offset} + S_{base}$ score of w_i

i-th level of the w_i



$$L_i = \left\lfloor \frac{\sum_{j=0}^{i-1} f_j}{\sum_{j=0}^{n} f_j} \cdot k \right\rfloor, \; f_0 = 0$$

Figure3.3.1 The score System

3.3.1 Porter stemming algorithm

Apply the Porter Stemming Algorithm to remove common morphological and inflectional endings of English words

Prevent words with the same root, but receiving different scores (e.g. determinant and determine)Reduce the plural form to the singular form (e.g. experiments and experiment)

Chapter 4:Implementation

4.1Tools and Technologies:

4.1.1 Java 1.6 Version:

4.1.1.1 Characteristics:

JAVA is a programming language, developed by Sun Microsystems and first released in 1995 (release 1.0). Since that time, it gained a large popularity mainly due to two characteristics:

A JAVA programme is hardware and operating system independant. If well written (!), the same JAVA programme, compiled once, will run identically on a SUN/solaris workstation, a PC/windows computer or a Macintosh computer. Not mentioning other Unix flavors, including Linux, and every Web browser, with some restrictions described below. This universal executability is made possible because a JAVA programme is run through a JAVA Virtual Machine. it is an object oriented language. This feature is mainly of interest for software developers.

4.1.1.2 JAVA Virtual Machine (JVM):

A JAVA programme is build by a JAVA compiler which generates its own binary code. This binary code is independant from any hardware and operating system. To be executed, it needs a *virtual machine*, which is a programme analyzing this binary code and executing the instructions it contains.

Of course, this Java Virtual Machine (JVM) is hardware and operating system dependant. Two types of Virtual Machines exist: those included in every Web Browser, and those running as an independent programme, like the Java RunTime Environment (JRE) from Sun Microsystems. These programmes need to be downloaded for your particular platform. As seen in the next paragraph, these two types of Virtual Machines do not behave exactly the same.

4.1.1.3 Applet and Standalone Application:

A JVM in a web browser runs a JAVA programme as an Applet. The applet is embedded in a web page and downloaded from a web server like any other HTML page or image

when requested. An independent JVM runs a JAVA programme as a Standalone Application.

4.1.2 Eclipse Galileo Version 3.5.1

Eclipse is an Integrated development environment(IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications in Java.

4.2 Classes Provided by the Framework

The interaction framework provides three base classes that contain the basic functionality necessary for implementing real-time games. All three classes are *generic* and *abstract*. This means that they must be extended to create actual games. The subclasses which extend the base classes contain all additional code that describes the behavior of the game. As only the core functionality is provided by the base classes, the framework makes it easy to implement a wide range of different game designs. This Section gives an overview of the functionality provided by the base classes.

PlayerMatcher The most important class in the multi-player interaction framework is the PlayerMatcher. It lies within the application context and represents the global entry point for all users wanting to participate in a game. The most important method provided by the PlayerMatcher is match. The method returns a SharedGame and a Player object. It can be called without parameters. In this case, it creates a Player and a SharedGame using default parameters. Alternatively, a Player object and a SharedGame can be provided by the caller. This makes it possible to customize the parameters of the SharedGame as opposed to relying on the default value. For example, the two variants of Karido rely on the same set of classes, but customize the parameters of the SharedGame
to implement the different game mechanics.In the match method, the PlayerMatcher searches the maintained list of running gamesto ensure that no player is able to participate in multiple games at the same time. If anexisting game is found and it is *compatible* the game provided by the caller, the existing SharedGame and the existing Player object are

returned. This means that the player is redirected to the existing game. In contrast, if the existing game is not compatible with the requested new game, the existing game is terminated and the new SharedGame is used to find a partner for the player. As the next step, the PlayerMatcher searches the list of waiting players

SharedGame class

The SharedGame class encapsulates all data that is shared between the players of a game. In the base class, this data primarily consists of information necessary for running a game – such as the index of the current round, the total number of rounds, the duration of each round or the score achieved by the players. The methods provided by the SharedGame base class consists primarily of accessors for the shared properties. Two abstractmethods must be implemented by the subclasses created for a game. The startNewRoundmethod is called each time a new round of the game has begun and can be used to initializeany game specific properties.

Player class

The Player class represents a player of a game. Each Player contains a Person attribute. The Person class is provided by the Artigo framework and represents a person that is logged into the system or plays anonymously. The Player class also holds a GameRound object, which represents a game round as defined by the Artigo framework. This object can, for example, be used to append custom actions to the game round. Each Player is assigned a list of *notifiers*. These notifiers are text strings, which can be added using the addNotifier and signal methods. The notifiers are used by the Player object as well as the user interface to react to events in a game. For example, once a player has sent a tag to her partner in Karido, a notifier "tags" is added. This notifier is intercepted by the polling method of the partner's user interface. The user interface responds by displaying the updated list of tags.

Preparations. The first step in the implementation of a new game is to create the necessary folder structure, representing the Java namespace of the new game. In this example, the used namespace is gwap.game.test. Four new classes must be added into this namespace: **Ai.java, Player.java, PlayerMatcher.java, SharedGame.java**. These classes extend the base classes provided by the framework

Furthermore, a GameType for the new game must be created in the database of the system.

For the example, this can be achieved by executing the following SQL command:

INSERT INTO gametype

(id , d e s c r i p t i o n , l abe l , name , playe r s , roundduration , rounds , workflow , plat form)

VALUES

(3 , ’A simple test scenario ’ , ’Test ’ , ’gwapGameTest ’ , 2 , 120 , 1 , **NULL**, **NULL**) ;

Please make sure to choose an id that is not yet taken. Longer values for roundduration can be set if desired.

4.3 CODE

Database

```
import HumanComputation.GWAP.*;

import java.sql.*;
import java.util.concurrent.*;

import java.util.*;

public class Database extends HumanComputation.GWAP.Database{

   //
   //constructors
   //

   public Database(){

      System.out.print("creating database system ... ");

      checkDB();
      connectDB();
      checkTable();

      System.out.println("ok");

   }

   //
   //public methods
   //

   public String getParameter(String s){

      String str = "";

      if(s.equals("GETNEWIMAGE")){

         PconDB();

         //random selection

         String sql;
```

```
        ResultSet rs;

        int picid = 0;
        String filename = "";
        List<String> label = new ArrayList<String>();
        List<String> candidate = new ArrayList<String>();

        //get id and filename
        sql = "SELECT id, filename FROM pic WHERE isplaying = false ORDER BY
RAND() LIMIT 1";
        rs = querySQL(sql);

        int i = 0;

        try{
           while(rs.next()){
              picid = rs.getInt("id");
              filename = rs.getString("filename");
              i++;
           }
        }catch(SQLException sqle){
           System.out.println("" + sqle);
           VconDB();
        }

        if(i != 1){
           System.out.println("error code : 19, i = " + i + ", picid = " + picid + ", filename
= " + filename);
           VconDB();
        }

        //get label
        sql  = "SELECT label.taboo taboo FROM label AS label ";
        sql += "JOIN session AS session ON label.session_id = session.id ";
        sql += "WHERE session.pic_id = " + picid + " ";
        sql += "ORDER BY taboo ASC ";

        rs = querySQL(sql);

        try{
           while(rs.next()){
              label.add(rs.getString("taboo"));
           }
        }catch(SQLException sqle){
           System.out.println("" + sqle);
           VconDB();
        }

        //get candidate
        sql  = "SELECT DISTINCT tag FROM pic_metadata ";
```

```
            sql += "WHERE pic_id = " + picid + " ";

            rs = querySQL(sql);

            try{
               while(rs.next()){
                  candidate.add(rs.getString("tag"));
               }
            }catch(SQLException sqle){
               System.out.println("error code : 21 " + sqle);
               VconDB();
            }

            sql = "UPDATE pic SET isplaying = true WHERE id = " + picid;
            //executeSQL(sql);

            VconDB();

            str += "<IMAGEINFORMATION IMAGEID=\"" + picid + "\" URI=\"" +
imagehost + filename + "\">";
            for(i = 0; i < label.size(); i++){
               str += "<LABEL>" + label.get(i)  + "</LABEL>";
            }
            for(i = 0; i < candidate.size(); i++){
               str += "<CANDIDATE>" + candidate.get(i) + "</CANDIDATE>";
            }
            str += "</IMAGEINFORMATION>";
         }

         return str;
      }

      public void writeRecord(String s){
      }

      //
      //protected methods
      //


      //
      //private methods
      //

      private void checkDB(){

         try{
            Class.forName("com.mysql.jdbc.Driver");
         }catch(ClassNotFoundException cnfe){
            // can not find mysql driver
```

```
            System.out.println(cnfe);
        }

    }

    private void checkTable(){
    }

    private void connectDB(){

        try{
            setConnection(DriverManager.getConnection("jdbc:mysql://" + mysqlhost + "/" +
mysqldatabase, mysqlaccount, mysqlpassword));
        }catch(SQLException sqle){
            // can not connect to the database
            System.out.println(sqle);
        }

    }

    private void executeSQL(String sql){

        try{
            Statement statement = getConnection().createStatement();
            statement.executeUpdate(sql);
        }catch(SQLException sqle){
            if(sqle.getSQLState().equals("08S01") || sqle.getSQLState().equals("41000")){
                connectDB();
                executeSQL(sql);
            }else{
                System.out.println(sqle);
            }
        }

    }

    private ResultSet querySQL(String sql){

        try{
            Statement statement = getConnection().createStatement();
            ResultSet rs = statement.executeQuery(sql);
            return rs;
        }catch(SQLException sqle){
            if(sqle.getSQLState().equals("08S01") || sqle.getSQLState().equals("41000")){
                connectDB();
                return querySQL(sql);
            }else{
                System.out.println(sqle);
                return null;
            }
```

```
        }

    }

    private void PconDB(){

        try{
            conDataBase.acquire();
        }catch(InterruptedException ie){
            System.out.println(ie);
        }

    }

    private void VconDB(){
        conDataBase.release();
    }

    //
    //private attributes
    //

    private Semaphore conDataBase = new Semaphore(1, true);

    private String mysqlaccount = "";  // MySQL account
    private String mysqldatabase = ""; // Database name
    private String mysqlhost = "";        //MySQL host IP or localhost
    private String mysqlpassword = "";  //Password

    private String imagehost = "http:/xxx.xxx.xxx.xxx/.../.../";  //host of images

}
```

Enigma

```
import java.math.*;
import java.security.*;

public class Enigma{

   //
   //constructors
   //

   public Enigma(){

   }

   //
   //public methods
   //

   public String getPublickey(){


      return "";

   }

   public String decrypt(String msg){

      return "";
   }

   public String encrypt(String msg){

      return "";
   }

   //
   //protected methods
   //

   //
   //private methods
   //

   private void generateKeys(){
```

```
        p = BigInteger.probablePrime(N, random);
        q = BigInteger.probablePrime(N, random);



    }

    private String normalize(String val){

        return "";

    }


    //
    //private attributes
    //

    SecureRandom random = new SecureRandom();
    BigInteger phi;
    BigInteger n;
    BigInteger p;
    BigInteger q;

    //block size is 2 * N bits
    int N = 16;

}
```

Game

```
import HumanComputation.GWAP.*;

import java.util.*;
import java.text.*;

import java.util.concurrent.*;

public class Game extends HumanComputation.GWAP.Game{

   //
   //constructors
   //

   public Game(){

   }

   public Game(Player p1, Player p2, HumanComputation.GWAP.Database d,
HumanComputation.GWAP.Log l, HumanComputation.GWAP.Score s){
      addPlayer(p1);
      addPlayer(p2);
      setDatabase(d);
      setLog(l);
      setScore(s);

      p1.setGame(this);
      p2.setGame(this);
   }

   //
   //public methods
   //

   public void close(){

   }

   public void processMessage(HumanComputation.GWAP.Player p,String input){
      PconIP();
      Message msg = new Message(input);
      if(msg.getStringParameter("TYPE").equals("GUESS")){
         System.out.println(1);
         String guess = msg.getStringParameter("VALUE");
         System.out.println(guess);
         if(msg.getStringParameter("ROUNDID").equals("" + getRoundSize())){
```

```
            boolean agreement = false;
            for(int i = 0; i < thisroundlabelpool.size(); i++){
               if(thisroundlabelpool.get(i).toLowerCase().equals(guess.toLowerCase())){
                  agreement = true;
                  break;
               }
            }
            if(!agreement){
               boolean findnew = false;
               for(int i = 0; i < thisroundguesspool.size(); i++){
                  boolean wordequal =
thisroundguesspool.get(i).toLowerCase().equals(guess.toLowerCase());
                  boolean playerequal =
thisroundplayeridguesspool.get(i).equals(p.getInformation("ID"));
                  if(wordequal && !playerequal){
                     findnew = true;
                     break;
                  }
               }
               System.out.println("check point 1");
               thisroundplayertrace += "<GUESS PLAYER=\"" + getPlayerIndex(p) + "\"
TIMESTAMP=\"" + getNowTime() + "\">" + guess + "</GUESS>";
               thisroundguesspool.add(guess);
               System.out.println(thisroundplayertrace);
               thisroundplayeridguesspool.add(p.getInformation("ID"));
               if(findnew){
                  System.out.println(2);
                  thisroundagreement = guess;
                  thisroundscore = getScore().getScore(guess);
                  System.out.println(thisroundscore);
                  totalscore += Integer.parseInt(thisroundscore);

                  saveRound("SUCCESS");
                  createNewRound();
               }else{
                  sendExclusivePlayerMessage(getPlayerIndex(p), "<GUESS
ROUNDID=\"" + getRoundSize() + "\">" + guess + "</GUESS>");
               }

            }
         }
      }else if(msg.getStringParameter("TYPE").equals("PASS")){
         if(!(thisroundispassing.get(getPlayerIndex(p)).booleanValue())){
            thisroundispassing.remove(getPlayerIndex(p));
            thisroundispassing.add(getPlayerIndex(p), new Boolean(true));
            thisroundplayertrace += "<PASS PLAYER=\"" + getPlayerIndex(p) + "\"
timestamp=\"" + getNowTime() + "\" />";
            sendExclusivePlayerMessage(getPlayerIndex(p), "<PASS ROUNDID=\"" +
getRoundSize() + "\" />");
         }
```

```
            if(isAllPassing()){
                saveRound("PASS");
                createNewRound();
            }
        }else if(msg.getStringParameter("TYPE").equals("TIMEISUP")){
        }else if(msg.getStringParameter("TYPE").equals("ERRORMSG")){
        }else if(msg.getStringParameter("TYPE").equals("NAME")){
        }else if(msg.getStringParameter("TYPE").equals("ENCRYPTED")){
        }
        VconIP();
    }

    public void start(){

        setAllPlayerStage(4);

        gametrace += "<GAMETRACE CREATETIME=\"" + getNowTime() + "\"
NUMBEROFPLAYER=\"" + getPlayerSize() + "\">";
        for(int i = 0; i < getPlayerSize(); i++){
            gametrace += "<PLAYER ID=\"" + i + "\" IP=\"" +
getPlayer(i).getInformation("UserIP") + "\" />";
        }

        String str = "server is creating a game,";
        for(int i = 0; i < getPlayerSize(); i++){
            str += " player" + i + " id = " + getPlayer(i).getInformation("ID") + " ip = " +
getPlayer(i).getInformation("UserIP");
        }
        printlnLog(str);

        createNewRound();

    }

    //
    //protected methods
    //



    //
    //private methods
    //

    private void clearThisRoundInformation(){
        thisroundimageid = "";
        thisroundplayertrace = "";
        thisroundresult = "";
        thisroundscore = "";
        thisroundagreement = "";
```

```
        thisroundlabelpool.clear();
        thisroundguesspool.clear();
        thisroundplayeridguesspool.clear();
        thisroundispassing.clear();
        for(int i = 0; i < getPlayerSize(); i++){
            thisroundispassing.add(new Boolean(false));
        }
    }

    private void createNewRound(){

        // get new image
        Message msg = new Message(getNewImage());

        //create match element
        String match = "";
        if(thisroundresult.equals("SUCCESS")){
            match = "<MATCH SCORE =\"" + thisroundscore + "\">" + thisroundagreement
+ "</MATCH>";
        }

        clearThisRoundInformation();
        thisroundimageid = msg.getStringParameter("IMAGEID");

        //create image message
        String str = "<IMAGE URI=\"" + msg.getStringParameter("IMAGEURI") + "\"
ROUNDID=\"" + getRoundSize() + "\">";
        str += match;
        if(msg.getListParameter("LABELLIST") != null){
            for(int i = 0; i < msg.getListParameter("LABELLIST").size(); i++){
                str += "<LABEL>" + msg.getListParameter("LABELLIST").get(i) +
"</LABEL>";
                thisroundlabelpool.add(msg.getListParameter("LABELLIST").get(i));
            }
        }
        str += "</IMAGE>";

        sendAllPlayerMessage(str);

    }

    private String getNewImage(){
        return getDatabase().getParameter("GETNEWIMAGE");
    }

    private String getNowTime(){
        Calendar now = Calendar.getInstance();
        DateFormat dateformat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        return dateformat.format(now.getTime());
    }
```

```
    private boolean isAllPassing(){
        boolean passing = true;
        for(int i = 0; i < thisroundispassing.size(); i++){
            if(!(thisroundispassing.get(i).booleanValue())){
                passing = false;
                break;
            }
        }
        return passing;
    }

    private void PconIP(){
        try{
            conInstuctionProcess.acquire();
        }catch(InterruptedException ie){
            System.out.println(ie);
        }
    }

    private void printLog(String s){
        getLog().write(s);
    }

    private void printlnLog(String s){
        getLog().writeLine(s);
    }

    private void saveRound(String result){
        if(result.equals("SUCCESS")){
            thisroundresult = "SUCCESS";

        }else if(result.equals("PASS")){
            thisroundresult = "PASS";

        }else if(result.equals("INTERRUPT")){
            thisroundresult = "INTERRUPT";

        }

        String thisroundtrace = "<ROUND RESULT=\"" + thisroundresult + "\"
ENDTIME=\"" + getNowTime() + "\">";
        if(result.equals("SUCCESS")){
            thisroundtrace += "<AGREEMENT SCORE=\"" + thisroundscore + "\">" +
thisroundagreement + "</AGREEMENT>";
        }

        thisroundtrace += thisroundplayertrace;
        thisroundtrace += "</ROUND>";
        System.out.println(thisroundtrace);
```

```
    Round temp = new Round();
    temp.setParameter("ROUNDTRACE", thisroundtrace);
    addRound(temp);

}

private void setAllPlayerStage(int stage){
    for(int i = 0; i < getPlayerSize(); i++){
        getPlayer(i).setInformation("NowStage", "" + stage);
    }
}

private void sendAllPlayerMessage(String msg){
    for(int i = 0; i < getPlayerSize(); i++){
        getPlayer(i).sendMessage(msg);
    }
}

private void sendExclusivePlayerMessage(int index, String msg){
    for(int i = 0; i < getPlayerSize(); i++){
        if(i != index){
            getPlayer(i).sendMessage(msg);
        }
    }
}

private void sendPlayerMessage(int index, String msg){
    if(index >= 0 && index < getPlayerSize()){
        getPlayer(index).sendMessage(msg);
    }
}

private void VconIP(){
    conInstuctionProcess.release();
}

//
//private attributes
//

private String gametrace = ""; //game trace
private int totalscore = 0; //total score

//semaphore control instruction process
private Semaphore conInstuctionProcess = new Semaphore(1, true);

private String thisroundimageid = ""; //this round image id
private String thisroundplayertrace = ""; //this round player trace
private String thisroundresult = ""; //this round result
```

```
    private String thisroundscore = ""; //this round score
    private String thisroundagreement = ""; //this round agreement
    private List<Boolean> thisroundispassing = new ArrayList<Boolean>(); //store player
is passing or not
    private List<String> thisroundlabelpool = new ArrayList<String>(); //this round label
pool
    private List<String> thisroundguesspool = new ArrayList<String>(); //this round guess
pool
    private List<String> thisroundplayeridguesspool = new ArrayList<String>(); //this
round player guess pool
}
```

Log

```
import HumanComputation.GWAP.*;

import java.io.*;

import java.text.*;

import java.util.*;

public class Log extends HumanComputation.GWAP.Log
{

   //
   //constructors
   //

   public Log(){

     System.out.print("creating log system ... ");

    createLogFile();

   System.out.println("ok");

  }

   //
   //public methods
   //

   public void write(String s)
{

      Calendar now = Calendar.getInstance();
```

```
  DateFormat dateformat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

     String msg = dateformat.format(now.getTime()) + " " + s;

 System.out.print(msg);

   //output to log file
try{
      getBufferedWriter().write(msg);
getBufferedWriter().flush();
   }catch(IOException ioe){
      //occur error when write data to file
  System.out.println(ioe);
  }

 }

 public void writeLine(String s)
{
       write(s + "\n");
   }

//
   //protected methods
   //

   //
   //private methods
   //

 private void createLogFile()
{
```

```
        Calendar now = Calendar.getInstance();

    DateFormat dateformat = new SimpleDateFormat("yyyyMMddHHmmss");


   FileWriter logfile;


   try{


        logfile =  new FileWriter(dateformat.format(now.getTime()) + ".log");

setBufferedWriter(new BufferedWriter(logfile));


  }
catch(IOException ioe)
{
          //occur error when open file

 System.out.println(ioe);


   }


    }

   //
   //private attributes
   //

}
```

Main

```
public class Main
{


public static void main(String[] args)
{


 new Server(new Database(), new Log(), new Score()).start();


  }


}
```

Message

```
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.*;
import java.io.*;

import java.util.*;

public class Message{

   //
   //constructors
   //

   public Message(){

   }

   public Message(String xmlstr){
      parse(xmlstr);
   }

   //
   //public methods
   //

   public String getStringParameter(String str){
      String output = null;
      if(str.equals("CLIENTNAME")){
         output = clientname;
      }else if(str.equals("CLIENTPUBLICKEY")){
         output = clientpublickey;
      }else if(str.equals("IMAGEID")){
         output = imageid;
      }else if(str.equals("IMAGEURI")){
         output = imageuri;
      }else if(str.equals("ROUNDID")){
         output = roundid;
      }else if(str.equals("TYPE")){
         output = type;
      }else if(str.equals("VALUE")){
         output = value;
      }
      return output;
```

```
    }

    public List<String> getListParameter(String str){
        List<String> l = null;
        if(str.equals("CANDIDATELIST")){
            l = candidatelist;
        }else if(str.equals("LABELLIST")){
            l = labellist;
        }
        return l;
    }

    public void parse(String xmlstr){

        DocumentBuilderFactory dbf = dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = null;
        try{
            db = dbf.newDocumentBuilder();
        }catch(ParserConfigurationException pce){
            System.out.println(pce);
            return;
        }
        Document doc = null;
        try{
            doc = db.parse(new InputSource(new StringReader(xmlstr)));
        }catch(Exception e){
            System.out.println(e);
            return;
        }
        doc.normalize();
        NodeList nl = doc.getChildNodes();

        if(nl.getLength() == 1){
            Node root = nl.item(0);
            Node node;
            NamedNodeMap attrs;
            //check type
            if(root.getNodeName().equals("ECHO")){
                //ESP Lite Protocol
                //<ECHO CLIENTNAME="" CLIENTPUBLICKEY="" />
                //check format
                attrs = root.getAttributes();
                if(!root.hasChildNodes() && attrs != null &&
attrs.getNamedItem("CLIENTNAME") != null &&
attrs.getNamedItem("CLIENTPUBLICKEY") != null){
                    type = "ECHO";
                    clientname = attrs.getNamedItem("CLIENTNAME").getNodeValue();
                    clientpublickey =
attrs.getNamedItem("CLIENTPUBLICKEY").getNodeValue();
                    //System.out.println(type);
```

```
                //System.out.println(clientname);
                //System.out.println(clientpublickey);
            }
        }else if(root.getNodeName().equals("GUESS")){
            //ESP Lite Protocol
            //<GUESS ROUNDID=""></GUESS>
            //check format
            attrs = root.getAttributes();
            if(attrs != null && attrs.getNamedItem("ROUNDID") != null &&
root.getChildNodes().getLength() == 1 && root.getChildNodes().item(0).getNodeType()
== Node.TEXT_NODE){
                type = "GUESS";
                roundid = attrs.getNamedItem("ROUNDID").getNodeValue();
                value = root.getChildNodes().item(0).getNodeValue();
            }
            //check format
        }else if(root.getNodeName().equals("PASS")){
            //ESP Lite Protocol
            //<PASS ROUNDID="" />
            //check format
            attrs = root.getAttributes();
            if(attrs != null && attrs.getNamedItem("ROUNDID") != null &&
!root.hasChildNodes()){
                type = "PASS";
                roundid = attrs.getNamedItem("ROUNDID").getNodeValue();
            }
        }else if(root.getNodeName().equals("TIMEISUP")){
            //ESP Lite Protocol
            //<TIMEISUP />
            //check format
            attrs = root.getAttributes();
            if(attrs == null && !root.hasChildNodes()){
                type = "TIMEISUP";
            }
        }else if(root.getNodeName().equals("ERRORMSG")){
            //ESP Lite Protocol
            //<ERRORMSG></ERRORMSG>
            //check format
            attrs = root.getAttributes();
            if(attrs == null && root.getChildNodes().getLength() == 1 &&
root.getChildNodes().item(0).getNodeType() == Node.TEXT_NODE){
                type = "ERRORMSG";
                value = root.getChildNodes().item(0).getNodeValue();
            }
        }else if(root.getNodeName().equals("NAME")){
            //ESP Lite Protocol
            //<NAME></NAME>
            //check format
            attrs = root.getAttributes();
```

```
            if(attrs == null && root.getChildNodes().getLength() == 1 &&
root.getChildNodes().item(0).getNodeType() == Node.TEXT_NODE){
                type = "ERRORMSG";
                value = root.getChildNodes().item(0).getNodeValue();
            }
         }else if(root.getNodeName().equals("ENCRYPTED")){
            //ESP Lite Protocol
            //<ENCRYPTED></ENCRYPTED>
            //check format
            attrs = root.getAttributes();
            if(attrs == null && root.getChildNodes().getLength() == 1 &&
root.getChildNodes().item(0).getNodeType() == Node.TEXT_NODE){
                type = "ENCRYPTED";
                value = root.getChildNodes().item(0).getNodeValue();
            }
         }else if(root.getNodeName().equals("IMAGEINFORMATION")){
            //check format
            //System.out.println(xmlstr);
            attrs = root.getAttributes();
            if(attrs != null && attrs.getNamedItem("IMAGEID") != null &&
attrs.getNamedItem("URI") != null){
                type = "IMAGEINFORMATION";
                imageid = attrs.getNamedItem("IMAGEID").getNodeValue();
                imageuri = attrs.getNamedItem("URI").getNodeValue();
                //System.out.println(imageid);
                //System.out.println(imageuri);
                NodeList childs = root.getChildNodes();
                for(int i = 0; i < childs.getLength(); i++){
                   if(childs.item(i).getNodeName().equals("LABEL")){
                      //System.out.println(childs.item(i).getFirstChild().getNodeValue());
                      labellist.add(childs.item(i).getFirstChild().getNodeValue());
                   }else if(childs.item(i).getNodeName().equals("CANDIDATE")){
                      //System.out.println(childs.item(i).getFirstChild().getNodeValue());
                      candidatelist.add(childs.item(i).getFirstChild().getNodeValue());
                   }
                }
            }
         //}else if(root.getNodeName().equals("")){
            //check format
         //}else if(root.getNodeName().equals("")){
            //check format
         }else{
            return;
         }
      }else{
         return;
      }

   }
```

```
    //
    //protected methods
    //

    //
    //private methods
    //

    private void clearAllParameter(){
    }

    //
    //private attributes
    //

    private String clientname = "";
    private String clientpublickey = "";
    private String roundid = "";
    private String type = "";
    private String value = "";

    private String imageid = "";
    private String imageuri = "";
    private List<String> labellist = new ArrayList<String>();
    private List<String> candidatelist = new ArrayList<String>();

/*
<ECHO CLIENTNAME="" CLIENTPUBLICKEY="" />


<GUESS SESSIONID=""></GUESS>

<PASS SESSIONID="" />
<TIMEISUP />
<ERRORMSG></ERRORMSG>


<NAME></NAME>
<ENCRYPTED></ENCRYPTED>
*/


}
```

Player

```
import HumanComputation.GWAP.*;

import java.io.*;
import java.net.*;
import java.util.concurrent.*;

import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.*;

public class Player extends HumanComputation.GWAP.Player{

   //
   //constructors
   //

   public Player(){
      //do nothing
   }

   public Player(int i, Socket sc, Server sv, HumanComputation.GWAP.Log l){

      id = i;
      socket = sc;
      server = sv;
      setLog(l);

      userip = socket.getInetAddress().getHostAddress();

   }

   //
   //public methods
   //

   public String getInformation(String str){

      String info = null;

      if(str.equals("ID")){

         info = "" + id;

      }else if(str.equals("UserIP")){
```

```
            info = userip;

        }else if(str.equals("NowStage")){
            PconNS();
            info = "" + nowstage;
            VconNS();
        }
        return info;

    }

    public void run(){

        printlnLog("client id " + id + " ip " + userip + " connected");

        createIO();

        //send hello message
        String hellomsg = "<WELCOME SERVERNAME=\"" + getServerName() + "\"
PROTOCOLVER=\"" + getProtocolVersion();
        hellomsg += "\" GAMEDURATION=\"" + getGameDuration() + "\"
SERVERPUBLICKEY=\"" + getServerPublicKey();
        hellomsg += "\" TRANSMITIONENCRYPTED=\"" +
getIsTransmissionEncrypted() + "\" />";
        sendMessage(hellomsg);

        String receivestr = null;
        while(true){
            try{
                receivestr = input.readLine();
            }catch(IOException ioe){
                receivestr = null;
            }

            if(receivestr == null){
                if(nowstage == 3){
                    printlnLog("client id " + id + " breaks connection");
                    server.removefromWaitPool(this);
                }else if(nowstage == 4){
                    printlnLog("client id " + id + " breaks connection");
                }else if(nowstage == 5){
                }else{
                }
                break;
            }else{
                printlnLog("recvive message from connection id " + id + " : '" + receivestr +
"'");

                Message msg = new Message(receivestr);
```

```
            if(nowstage == 1){
               //flash client auth
            }else if(nowstage == 2){
               //login
            }else if(nowstage == 3){
               //find player
               if(msg.getStringParameter("TYPE").equals("ECHO")){
                  if(server.addtoWaitPool(this) == false){
                     //add failure
                     sendMessage("<ERRORMSG>server is finding your partner
now</ERRORMSG>");
                     closeSocket();
                  }else{
                     sendMessage("<REMAINDER>" + getRemainder() +
"</REMAINDER>");
                  }
               }

            }else if(nowstage == 4){
               //playing

               if(msg.getStringParameter("TYPE").equals("GUESS")){
                  getGame().processMessage(this, receivestr);
               }else if(msg.getStringParameter("TYPE").equals("PASS")){
                  getGame().processMessage(this, receivestr);
               }else if(msg.getStringParameter("TYPE").equals("TIMEISUP")){
                  getGame().processMessage(this, receivestr);
               }else{
               }

            }else if(nowstage == 5){
               //ending
            }else{

            }
         }
      }

   }

   public void sendMessage(String msg){

      PconSM();

      printlnLog("sending message to client id " + id + " : " + msg);

      try{
         output.write(msg + "\n");
         output.flush();
```

```
        }catch(IOException ioe){
            System.out.println(ioe);
        }

        VconSM();

    }

    public void setInformation(String name, String value){
        if(name.equals("NowStage")){
            nowstage = Integer.parseInt(value);
        }
    }

    //
    //protected methods
    //



    //
    //private methods
    //

    private boolean addtoWaitPool(Player p){
        return server.addtoWaitPool(p);
    }

    private void closeSocket(){
        try{
            socket.close();
        }catch(IOException ioe){
            System.out.println(ioe);
        }
    }

    private void createIO(){

        try{
            input = new BufferedReader(new InputStreamReader(socket.getInputStream(),
"UTF-8"));
            output = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream(),
"UTF-8"));
        }catch(IOException ioe){
            //occur error when applying input or output to socket
            System.out.println(ioe);
        }

    }

    private String getGameDuration(){
```

```
    return server.getGameDuration();
}

private int getRemainder(){
    return server.getRemainder();
}

private String getServerName(){
    return server.getName();
}

private String getServerPublicKey(){
    return "";
}

private String getIsTransmissionEncrypted(){
    return server.getIsTransmissionEncrypted();
}

private String getProtocolVersion(){
    return server.getProtocolVersion();
}

private void printLog(String s){
    getLog().write(s);
}

private void printlnLog(String s){
    getLog().writeLine(s);
}

private void PconNS(){
    try{
        conNowStage.acquire();
    }catch(InterruptedException ie){
        System.out.println(ie);
    }
}

private void PconSM(){
    try{
        conSendMessage.acquire();
    }catch(InterruptedException ie){
        System.out.println(ie);
    }
}

private void VconNS(){
    conNowStage.release();
}
```

```
    private void VconSM(){
        conSendMessage.release();
    }

    //
    //private attributes
    //

    private Semaphore conNowStage = new Semaphore(1, true);

    private Semaphore conSendMessage = new Semaphore(1, true);

    private Game g;

    private int id;

    //now stage
    private int nowstage = 3;

    //the socket input bufferedreader
    private BufferedReader input;

    //the socket output bufferedwriter
    private BufferedWriter output;

    private Server server;

    private Socket socket;

    private String userip;

}
```

Round

```
import HumanComputation.GWAP.*;


public class Round extends HumanComputation.GWAP.Round
{


    //
    //constructors
    //

    public Round()
{

   //do nothing

}


 //
    //public methods
    //

    public String getParameter(String str)
{

String temp = "";

      if(str.equals("ROUNDTRACE"))
{

            temp = roundtrace;

   }

        return temp;

}


  public void setParameter(String name, String value){

  if(name.equals("ROUNDTRACE"))
```

```
{
        roundtrace = value;
  }
  }
    //
    //protected methods
    //


    //
    //private methods
    //

    //
    //private attributes
    //
    private String roundtrace = "";

}
```

Score

```
import HumanComputation.GWAP.*;


public class Score extends HumanComputation.GWAP.Score
{


 //
    //constructors
    //

    public Score()
{


   System.out.print("creating scoring system ... ");


     System.out.println("ok");


 }


 //
    //public methods
    //

    public String getScore(String input){

     return "100";

  }


    private void loadScoreTable(){


  }

}
```

Server

```
import HumanComputation.GWAP.*;

import java.io.*;
import java.net.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;

public class Server extends HumanComputation.GWAP.Server{

   //
   //constructors
   //

   public Server(){

   }

   public Server(Database db, Log l, Score sc){
      super(db, l, sc);

   }

   //
   //public methods
   //

   public boolean addtoWaitPool(Player p){
      boolean addsuccess = true;
      PconWP();
      for(int i = 0;i < waitpool.size(); i++){
        if(waitpool.get(i).getInformation("UserIP").equals(p.getInformation("UserIP"))){
           addsuccess = false;
           break;
        }
      }
      addsuccess = true;
      if(addsuccess){
         waitpool.add(p);
      }
      VconWP();
      return addsuccess;
   }
```

```
public String getGameDuration(){
   return "" + gameduration;
}

public String getProtocolVersion(){
   return protocolversion;
}

public String getIsTransmissionEncrypted(){
   if(istransmissionencrypted){
      return "1";
   }else{
      return "0";
   }
}

public String getName(){
   return name + " " + version;
}

public int getRemainder(){
   int temp;
   PconRM();
   temp = remainder;
   VconRM();
   return temp;
}

public void removefromWaitPool(Player p){
   PconWP();
   for(int i = 0; i < waitpool.size(); i++){
      if(waitpool.get(i) == p){
         waitpool.remove(i);
         break;
      }
   }
   VconWP();
}

//start server
public void start(){

   printlnLog(getName());

   printlnLog("opening server port ...");

   //open server port
   ServerSocket serversocket = null;
   Socket socket;
```

```
        try{
            serversocket = new ServerSocket(port, 1024);
        }catch(IOException ioe){
            //occur error when open listing port
            printlnLog("" + ioe);
        }

        //start creating game
        new Timer().schedule(new CreateGame(this), 1 * 1000, 1 * 1000);

        printlnLog("server starting");

        //listening port
        while(true){

            try{
System.out.println("before socket accept");
                socket = serversocket.accept();
System.out.println("socket accepted!!");
            }catch(IOException ioe){
                //occur error
                printlnLog("" + ioe);
                continue;
            }

            try{
                socket.setSoTimeout((gameduration + 30) * 1000);
            }catch(SocketException se){
                //occur error
                printlnLog("" + se);
                continue;
            }

            new Player(idcount, socket, this, getLog()).start();

            idcount++;

        }

    }

    public void triggerRemainder(){
        boolean key = false;

        PconRM();

        remainder--;
        if(remainder == 0){
            remainder = creategameperiod;
            key = true;
```

```
        }

        VconRM();

        if(key){
            createGame();
        }
    }

    //
    //protected methods
    //

    protected void createGame(){

        PconWP();
        int numberofgame = (int)Math.ceil((double)waitpool.size() / 2.0);
        int waitpoolsize = waitpool.size();
        printlnLog("creating game, there are " + waitpoolsize  + " player" + (waitpoolsize >
1 ? "s" : "") + " in waitingpool.");

        if(waitpoolsize % 2 == 0){
            for(int i = 0; i < numberofgame; i++){
                new Game(waitpool.get(i * 2), waitpool.get(i * 2 + 1), getDatabase(), getLog(),
getScore()).start();
            }
            waitpool.clear();
        }


        VconWP();
    }

    //
    //private methods
    //

    //P() control the remainder
    private void PconRM(){
        try{
            conRemainder.acquire();
        }catch(InterruptedException ie){
            System.out.println(ie);
        }
    }

    //P() control the wait pool
    private void PconWP(){
        try{
            conWaitPool.acquire();
```

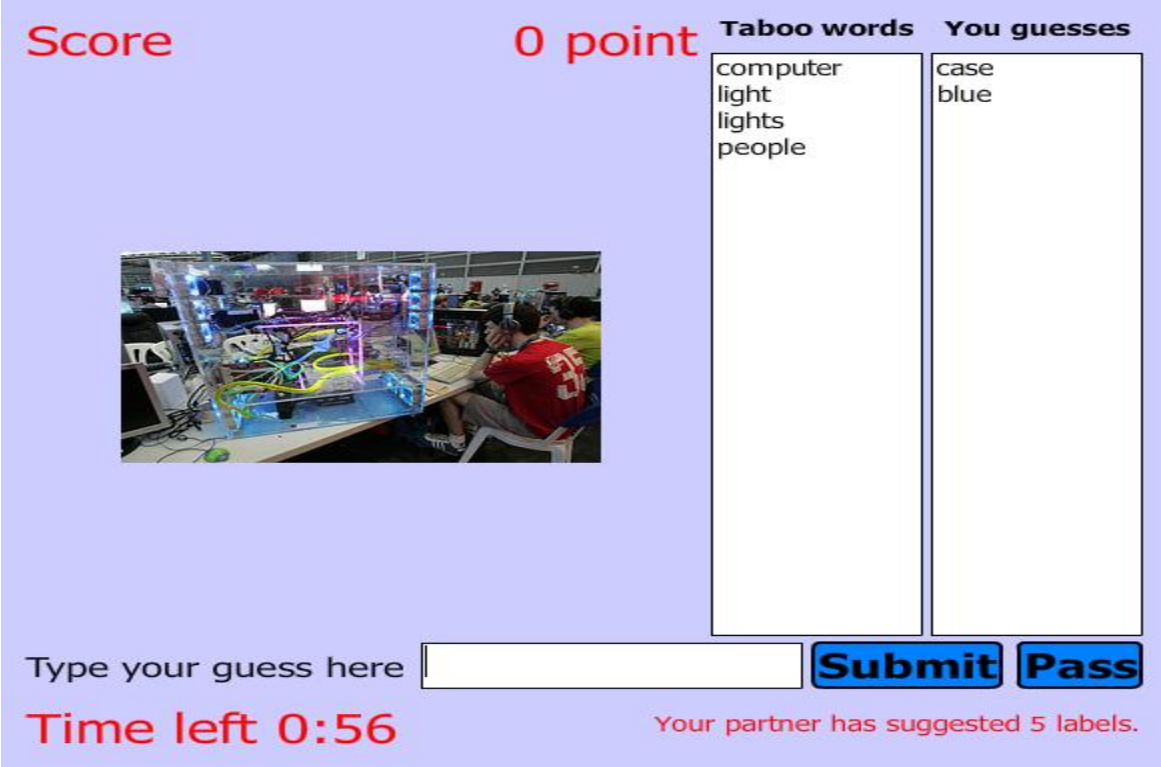
```
    }catch(InterruptedException ie){
        System.out.println(ie);
    }
}

private void printLog(String s){
    getLog().write(s);
}

private void printlnLog(String s){
    getLog().writeLine(s);
}

//V() control the remainder
private void VconRM(){
    conRemainder.release();
}

//V() control the wait pool
private void VconWP(){
    conWaitPool.release();
}

//
//private attributes
//



//control the remainder
private Semaphore conRemainder = new Semaphore(1, true);

//control the wait pool
private Semaphore conWaitPool = new Semaphore(1, true);

//create game period (seconds)
int creategameperiod = 30;

//connection id count
int idcount = 0;

//is transmission encrypted
boolean istransmissionencrypted = false;

//server port
int port = 56779;

//protocol version
String protocolversion = "0.1";
```

```
    //game duration (seconds)
    int gameduration = 120;

    //server name
    String name = "ESP Lite Server";

    //the remainder time of starting game
    private int remainder = 30;

    //the wait pool
    private List<Player> waitpool = new ArrayList<Player>();

    //server version
    String version = "3.1";

}

class CreateGame extends TimerTask{

    //
    //constructors
    //

    public CreateGame(Server sv){
        server = sv;
    }

    //
    //public methods
    //

    public void run(){
        server.triggerRemainder();
    }

    //
    //protected methods
    //

    //
    //private methods
    //

    //
    //private attributes
    //

    //the server reference
    Server server;
```

CHAPTER 5: Results

SNAPSHOTS

Figure4.4.1Client interface1



Explanation

In this screenshot we can see that there are two words which are being guessed by the user

The time left is shown in the bottom of the page.

As soon as the submit button is pressed the time is stopped

Figure4.4.2Client interface 2

Explanation

The second player is now guessing the name .As soon as he presses ok. The time is stopped and the answer is stored in the database

Chapter 6:Conclusion and future work

5.1 Conclusion

The goal of this diploma thesis is to create a novel image labeling game, collecting more comprehensive tag sets than previous implementations. A short evaluation has shown promising results, especially for the time-limited mode of the game. However, more assessments are necessary to conclusively prove the effectiveness of the game.

The scientific contribution of this thesis is a new design of an image labeling game, which can potentially collect more comprehensive tags sets and has been proven to be fun to play in a empirical evaluation. The game is publicly accessible at the time of writing and collects valuable data.

5.2.Future Work

The preliminary evaluation indicates that the game is fun play and collects reasonable data. However, several possible tasks remain to be completed in the future.

5.2.1. Large Scale Evaluation

Game is designed to collect comprehensive sets of tags. An empirical evaluation of this design goal should be performed. However,game's main mechanism of input-similarity only comes into full effect with reasonably sized collections of images (as there must be a sufficient number of similar images). Furthermore, the game strives to balance the number of tags for each image, processing images with few tags first. Thus, a large number of generic tags have to be applied to all images before more specific labels are collected. A proper evaluation of the game therefore requires a large number of participants or a long duration,preferably both.

5.2.2. Improved Data Verification

As described before, game uses a scoring system to assess the validity of any tags entered by the players. Tags with a score exceeding a given threshold can be considered reliable.

However, there is currently no way to distinguish between tags which have a low score because they are wrong and tags which have not yet been validated. The game should not continue to validate tags which seldom or never lead to correct guessing, in order to avoid

annoying the Guessers. Similarly, tags which have been verified to be correct should also be verified no longer, to ensure that new tags can be verified quickly.

The data model of game could be enhanced to keep track of the number of times a given tag has been replayed. The verification could then be stopped after a tag has been replayed a certain number of times. However, this would enable malicious players to remove tags from the verification pool, by deliberately entering wrong tags. Alternative schemes could use median values of the number of times all tags have been replayed to balance the verification process.

Designing and implementing a verification scheduling system for large datasets exceeds the scope of this thesis. However, such a system could be created in the future to improve the data verification process.

5.2.3. Compiling Compound Tags

During the creation and evaluation of game, a new issue for image labeling games emerged. While game ensures that its players provide new labels, it does not impose restrictions on the form in which these labels are applied.

Chapter 6:List of References

[1] P. N. Bennett, D. M. Chickering, and A. Mityagin. Picture this: preferences for image search. In Proceedings of the ACM SIGKDD Workshop on Human Computation,HCOMP '09, pages 25–26, New York, NY, USA, 2009. ACM. ISBN

[2] M. Borella. Source models of network game traffic. Computer Communication 23 403–410, Feb. 2000. ISSN 01403664. doi: 10.1016/S0140-3664(99)00197-8.

[3]T. Chen, M. M. Cheng, P. Tan, A. Shamir, and S. M. Hu. Sketch2Photo: internet image montage. ACM Trans. Graph., 28(5):1–10, December 2009. ISSN 0730-0301

[4] R. Clarke, J. Burken, J. Bosworth, and J. Bauer. X-29 flight control system: lessons learned. International Journal of Control, 59(1):199–219, 1994. ISSN 0020-7179. 1

[5]von Ahn, L., and Dabbish, L. Labeling Images with a Computer Game. In ACM Conference on HumanFactors in Computing Systems (CHI), 2004, pp 319-326.

[6]Barnard, K., and Forsyth, D. A. Learning the Semantics of Words and Pictures. International Conference of Computer Vision, 2001, pages 408-415.