



QLectives – Socially Intelligent Systems for Quality

Project no. 231200

Instrument: Large-scale integrating project (IP)

Programme: FP7-ICT

Deliverable D4.4.4

Test report on HQ-MD implementation

Submission date: 2013-08-31

Start date of project: 2009-03-01

Duration: 54 months

Organisation name of lead contractor for this deliverable: IRT

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	x
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document information

1.1 Authors

Author	Organisation	E-mail
M. Guelbahar	IRT	guelbahar@irt.de
R. Ormándi	USZ	ormandi@inf.u-szeged.hu
A. Zistler	IRT	zistler@irt.de
I. Hegedűs	USZ	ihegedus@inf.u-szeged.hu
Johan Pouwelse	TUD	peer2peer@gmail.com

1.2 Other contributors

Name	Organisation	E-mail
Niels Zeilemaker	TUD	niels@zeilemaker.nl

1.3 Document history

Version#	Date	Change
V0.1		Starting version, template
V0.2	22-08-2012	Definition of ToC
V0.3	23-01-2013	First draft version
V0.4	31-01-2013	Draft version
V0.8	26-02-2013	Intermediate version to be submitted to the EU
V1.0	31-08-2013	Final version to be submitted to the EU (M54)

1.4 Document data

Keywords	QLectives, Metadata, Quality, QMedia	
Editor address data	Name:	Alois Zistler
	Partner:	IRT
	Address:	Floriansmuehlstrasse 60, D-80939 Muenchen, Germany
	Phone:	+49 89 323 99 234
	Fax:	+49 89 323 99 200
	E-mail:	zistler@irt.de
Delivery date	August 2013	

1.5 Distribution list

Date	Issue	E-mail
	Consortium members	QLECTIVES@LIST.SURREY.AC.UK
	Project officer Roumen Borissov	Roumen.BORISSOV@ec.europa.eu
	EC archive	INFSO-ICT-231200@ec.europa.eu

QLectives Consortium

This document is part of a research project funded by the ICT Programme of the Commission of the European Communities as grant number ICT-2009-231200.

University of Surrey (Coordinator)

Department of Sociology/Centre for
Research in Social Simulation

Guildford GU2 7XH

Surrey

United Kingdom

Contact person: Prof. Nigel Gilbert

E-mail: n.gilbert@surrey.ac.uk

ETH Zurich

Chair of Sociology, in particular
Modelling and Simulation,

Zurich, CH-8092

Switzerland

Contact person: Prof. Dirk Helbing

E-mail: dhelbing@ethz.ch

Technical University of Delft

Department of Software Technology

Delft, 2628 CN

Netherlands

Contact Person: Dr Johan Pouwelse

E-mail: j.a.pouwelse@tudelft.nl

University of Szeged

MTA-SZTE Research Group on

Artificial Intelligence

Szeged 6720, Hungary

Contact person: Dr Mark Jelasity

E-mail: jelasity@inf.u-szeged.hu

University of Fribourg

Department of Physics

Fribourg 1700

Switzerland

Contact person: Prof. Yi-Cheng Zhang

E-mail: yi-cheng.zhang@unifr.ch

Centre National de la Recherche

Scientifique, CNRS

Paris 75006,

France

Contact person: Dr. Camille ROTH

E-mail: camille.roth@polytechnique.edu

University of Warsaw

Faculty of Psychology

Warsaw 00927, Poland

Contact Person: Prof. Andrzej Nowak

E-mail: nowak@fau.edu

Institut für Rundfunktechnik GmbH

Munich 80939

Germany

Contact person: Christoph Dosch

E-mail: dosch@irt.de

QLectives introduction

QLectives is a project bringing together top social modellers, peer-to-peer engineers and physicists to design and deploy next generation self-organising socially intelligent information systems. The project aims to combine three recent trends within information systems:

- **Social networks** - in which people link to others over the Internet to gain value and facilitate collaboration
- **Peer production** - in which people collectively produce informational products and experiences without traditional hierarchies or market incentives
- **Peer-to-Peer systems** - in which software clients running on user machines distribute media and other information without a central server or administrative control

QLectives aims to bring these together to form Quality Collectives, e.g. functional decentralised communities that self-organise and self-maintain for the benefit of the people who comprise them. We aim to generate theory at the social level, design algorithms and deploy prototypes targeted towards two application domains:

- **QMedia** - an interactive peer-to-peer media distribution system (including live streaming), providing fully distributed social filtering and recommendation for quality
- **QScience** - a distributed platform for scientists allowing them to locate or form new communities and quality reviewing mechanisms, which are transparent and promote

The approach of the QLectives project is unique in that it brings together a highly interdisciplinary team applied to specific real world problems. The project applies a scientific approach to research by formulating theories, applying them to real systems and then performing detailed measurements of system and user behaviour to validate or modify our theories if necessary. The two applications will be based on two existing user communities comprising several thousand people - so-called "Living labs", media sharing community tribler.org; and the scientific collaboration forum EconoPhysics.

Executive Summary

This document describes final results of WP4.4, which had the objective to realise mechanisms for detection and analysis of High-Quality (HQ) and Spam Metadata within fully decentralised Peer-to-Peer (P2P) networks, respectively the QLectives QMedia implementation.

Within this deliverable, a test report on the integration of the metadata solution will be presented that has been performed based on the final version of QMedia, which concludes the work within this work package.

Chapter 1 gives a short introduction on the background and the objectives of this work package, thereby outlining the relevance of the presence of high quality metadata for efficient search and discovery of media content, as well as means for *detection and handling* of HQ and spam metadata.

The following chapter provides background information about QMedia, the Gossip Learning Framework (GoLF), WP4.4 results as well as the process of integration of it within QMedia.

Within chapter 3, details concerning evaluation settings as well as the processing of evaluations with benchmark datasets will be presented.

Chapter 4 describes the process of the tests performed and presents the results on user tests. Focus will also be given on feedback from users concerning improvability of the QMedia implementation.

The conclusions and references are presented in chapter 5 and chapter 6 respectively.

Contents

1	Introduction	4
2	Background	6
2.1	Distributed System Model.....	6
2.2	Tribler & QMedia.....	6
2.3	Gossip Learning Framework	8
2.3.1	Machine learning and Classification	8
2.3.2	Data Distribution	9
2.3.3	GoLF.....	9
2.4	Integration of the GoLF into Tribler	10
2.4.1	GoLF Community	10
2.4.2	Classification Algorithm.....	11
3	Evaluation	13
3.1	Benchmark Evaluation Settings.....	13
3.2	Results on Benchmark Data.....	13
4	Results on user-tests.....	15
4.1	Set-up of user-tests	15
4.1.1	Testing environment	15
4.1.2	Testing: Background & procedure.....	16
4.2	Results from user-tests.....	23
4.2.1	Dynamic GoLF model implementation.....	23
4.2.2	Static GoLF model implementation.....	24
4.2.3	Feedback from participants.....	30
5	Conclusions	32
6	References	34
7	Annex A – Questionnaire	36

1 Introduction

In real life social and scientific portals, administrators or moderators need to put a lot of effort into trying to maintain a high level and quality of the information they host. Particularly when information is of dynamic nature, e.g. generated or editable by users, the existence of spam or invalid data obviously is unavoidable. Since the amount of data is growing rapidly and the users purposely or inadvertently generate spam content (no matter whether intentionally or not), the *validation of any data* is an essential step towards the maintenance of High Quality (HQ) services of any kind.

Manual checking of data for low-quality or spam is a quite expensive task; therefore, *self-learning algorithms* (as known from email Junk filters) promise to be of help when it comes to detection and classification of data - as already outlined in deliverable D4.4.2 [1], where machine learning techniques have been applied to the Wikipedia Vandalism Detection task.

The adoption of machine learning techniques on P2P platforms in a *fully distributed way* is much more challenging than the work carried out within the Wiki Vandalism Detection task, since *no single point of control* (and also trust) exists. Every node is equivalent to every other node, so knowledge about any information inside the network, as well as the exchange and evolution of this knowledge are essential. Especially the latter is of high importance, since, as known from email spam, *ways of spamming change*. Solutions need to be flexible and able to *learn and adapt to these changes* in order to effectively maintain high quality filtering results.

Therefore, the decision fell onto a wedding of both, the Gossip Learning Framework (GoLF), and self-learning algorithms to detect HQ and spam metadata – as further described in chapter 2.

For user evaluations covering SPAM detection functionality, a common understanding of the definition of SPAM is essential to come to evaluable results. The definition applied in QMedia was “any kind of information not being related to the torrent file itself, thereby aiming to relocate the user to a website offering content advertisements”

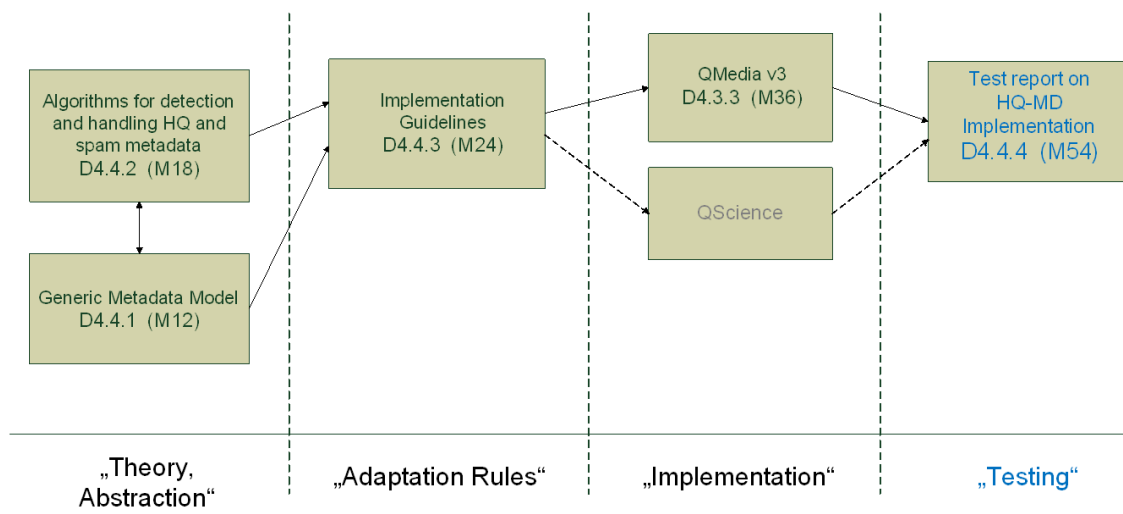


Figure 1: Scheduling & Interdependencies of tasks in WP4.4

This document concludes the work of work package 4.4, which is depicted in *Figure 1*.

2 Background

This section provides background information about QMedia, the Gossip Learning Framework (GoLF), WP4.4 results as well as the process of integration of it within QMedia.

2.1 *Distributed System Model*

As the basic system model within QMedia, a network of computers where each node can communicate with any other node by messages if the address of the target node is locally available is assumed. Additionally, it is being assumed that a peer sampling service exists which can provide addresses of uniform randomly selected nodes from the network – given through the Bit-Torrent and therefore P2P-based QMedia implementation.

2.2 *Tribler & QMedia*

In this section, a real-world example of a peer-to-peer network which the WP4.4 protocols build on will be discussed. *Tribler* (and therefore also QMedia, which is based on Tribler) is a social Peer-to-Peer content sharing platform. It can be seen as a standard BitTorrent [2] client, but enhanced by a set of additional features. One main advantage of using QMedia is that it is *fully decentralised*, meaning that there is no single point of failure, whereas for traditional BitTorrent, the tracker is a single point of failure [3].

Bootstrapping is done by the use of “superpeers”. Everyone can become a bootstrap peer, see [4]. Tribler has gained a lot of attention lately due to its distributed nature. For detailed statistics, see [5]. Tribler offers great searching functionalities, where the search is also performed in a distributed way; there is no need for a tracker. Content is structured into channels or communities.

Tribler is the product of many years of scientific works done at the Delft University of Technology, and is funded by the European Union 7th Framework Research Programme (P2P-Next, QLectives).

QMedia uses its own distributed permission system, called *Dispersy*. Dispersy provides a platform to build up *communities*, where a community basically consists of a protocol over a set of nodes. This includes the permission, distribution, and gossiping subsystems. A community *shares a set of messages known to all of its members, and a set of permissions* implemented by the members. Community members communicate through messages, where some of these messages are signed by the sender, and propagated throughout the network. There are multiple signing and propagating policies. Within the community, it is possible to send messages to random peers.

A special type of community provides *channels*. These are groups of torrents with attached metadata and comments. People can subscribe to channels, can discuss torrents, vote, mark as spam, and mark as favourite. The users can create their own channels, either by collecting torrents, or via an RSS feed. Additionally, there's the option to make a channel available to the public, so that everyone can edit metadata, as well as add and delete torrents. This is similar to how *Wikipedia* works, and has been shown to be a powerful tool for collaboration.

However, people can use this platform *to spam and do malicious edits*. Within QMedia, these edits can be reverted, essentially marking them as spam. This is where *WP4.4 results will be applied*, allowing the use of machine learning for spam filtering and vandalism detection.

Each peer has a local view of channels they subscribe to and it is not feasible to subscribe to all channels. The aim is to build models that are available for every peer and capture the global structure of the network so they can use that for filtering spam comments.

2.3 Gossip Learning Framework

Before describing the Gossip Learning Framework (GoLF) in detail, the central part of machine learning - namely the classification - and the distributed type of these algorithms will be introduced.

2.3.1 Machine learning and Classification

The classification algorithms, which represent a large part of the machine learning theory, are one kind of method that can be applicable for automation of the previously mentioned problem (i.e. decision that an edited item or comment is spam or not).

The problem of spam detection can be seen as a binary classification problem, where the *binary classification* can be defined as follows:

Given a training data set in the form of a set of training examples, each training instance consists of a feature vector (that describes or represents the instance) and the corresponding class label (i.e. it is a spam or not).

This training database can be denoted with:

$$S = (x_1, y_1), \dots, (x_n, y_n) \subset \mathbb{R}^d \times \{-1, +1\}$$

where d represents the dimension of the problem (the number of features); the sample corresponds to the positive class when its label is +1, and to the negative class when its label is -1 respectively.

In the QMedia use case the spam is denoted the positive class, and the negative class represents the non-spam content. The main goal of the classification problem is to find a function

$$f: \mathbb{R}^d \rightarrow \{-1, +1\}$$

using the observations from S that can classify any samples including those that are not part of the training set (this property is known as “generalisation”).

Function f is called *the model of the data*. In the case of training samples being available as a stream, the training process is known as “online learning”.

2.3.2 Data Distribution

In the QMedia data model, the training database is horizontally distributed over the complete network. Additionally it can be assumed that each peer in the network has just a few data records directly available or accessible, which excludes local statistical processing. Another key assumption is that the record never leaves the node; collecting the data to a central server is not allowed due to privacy considerations.

2.3.3 GoLF

The Gossip Learning Framework [6] is a machine learning framework designed to allow *a fully distributed learning in large scale networks*. The basic idea behind GoLF is that a large number of online models take *random walks in the network while improving themselves*, using the training samples contained at the individual nodes.

In *Listing 1*, the skeleton of GoLF is shown. At each peer in the network the same protocol is executed, consisting of an active loop of periodic activity, and the message handler method `ONRECEIVEMODEL` to process incoming models.

This method updates the received model using the locally stored training example and stores it as `currentModel`. In the active loop, the *stored model is sent to a randomly selected neighbour* (proposed by the peer sampling service).

At any time the *freshest model* (`currentModel`) is used for performing predictions. This means that as an advantage based on the algorithm design, *no communication is necessary for performing predictions*.

The concrete learning algorithm is implemented in the method `updateModel`.

Algorithm 1 *GoLF*

```
1: initModel()
2: loop
3:   wait( $\Delta$ )
4:    $p \leftarrow$  selectPeer()
5:   send currentModel to  $p$ 
6: procedure ONRECEIVEMODEL ( $m$ )
7:    $m \leftarrow$  updateModel( $m$ )
8:   currentModel  $\leftarrow$   $m$ 
```

Listing 1: Basic algorithm for decentralised exchange of knowledge
(in pseudo code)

2.4 Integration of the GoLF into Tribler

In this section, a detailed discussion of the core of the Gossip Learning Framework - as implemented in QMedia as a community – is presented, as well as a widely used learning algorithm that works on top of the core.

2.4.1 GoLF Community

Within QLectives, a new community called `GossipLearningCommunity` has been implemented, providing a *generic framework for learning how to use gossiping*.

Each community must define a number of message types that it handles. In the QMedia use case, there is only one message type called `modeldata`. This message has been designed to contain one object of the type `GossipMessage`, which builds the generic base class of the learning models.

The complete community uses this abstract message class, which can later be defined to consist of *any learning algorithm* – e.g. the one that detects spam metadata.

```
def active_thread(self):
    while True:
        self.send_messages([self._model_queue[-1]])
        yield DELAY

def on_receive_model(self, messages):
    for message in messages:
        msg = message.payload.message

        assert isinstance(msg, GossipMessage)

        if self._x == None or self._y == None:
            continue

        self._model_queue.append(self.create_model_mu(msg,
            self._model_queue[-1]))

def update(self, model):
    for x, y in zip(self._x, self._y):
        model.update(x, y)

def predict(self, x):
    return self._model_queue[-1].predict(x)
```

Listing 2: Algorithm of active loop and message handler process (in Python)

2.4.2 Classification Algorithm

As a next step, the support of vector machines (SVM) will be outlined, which represent the learning algorithms applied in the experiments performed [7].

In its simplest form, the SVM approach works with the *space of linear models to solve the binary classification problem*. Assuming a d dimensional problem, we want to find a $d-1$ dimensional separating hyperplane that maximises the margin that separates examples of the two classes.

The margin is defined by the hyperplane as the sum of the minimal perpendicular distances from both classes. The *Pegasos* algorithm – depicted in *Listing 3* - is an SVM training algorithm, based on a stochastic gradient descent approach [8].

```
class P2PegasosModel(GossipLearningModel):
    def __init__(self):
        super(P2PegasosModel, self). init ()
        # Initial model
        self.age = 0

    def update(self, x, y):
        label = -1.0 if y == 0 else 1.0
        self.age = self.age + 1
        lam = 0.0001
        rate = 1.0 / (self.age * lam)
        is_sv = label * sum([self.w[i] * x
                             [for i in range(len(self.w))]) < 1.0
        max_dim = max(len(self.w), len(x))

    for i in range(max_dim):
        if is_sv:
            self.w[i] = (1.0 - 1.0 / self.age) * self.w[i] +
                rate * label * x[i]
        else:
            self.w[i] = (1.0 - 1.0 / self.age) * self.w[i]

    def predict(self, x):
        inner_product = sum([self.w[i] * x[i] for i in
                             range(len(self.w))])
        return 1.0 if inner_product > 0.0 else 0.0
```

Listing 3: The source code of the *Pegasos* learning model (in Python)

For the implementation of the distributed QMedia metadata algorithms, the Pegasos algorithm has been used as the basis.

3 Evaluation

In this section, the evaluation methodology as well as background and results of the evaluation will be presented.

3.1 Benchmark Evaluation Settings

QMedia uses *public-key cryptography with elliptic curves* [9] for authentication and authorisation. By generating a master public/private key-pair, a community has been created, as well as public/private key-pairs for each peer. In the case of the emulation experiments, peers have been started simultaneously using different port and member ID settings, initialising each of them with different local labelled training examples. These were not full-blown QMedia instances, but so-called "scripts" – allowing the performance of the experiments without the need to start the QMedia GUI on every instance. Each peer's script redirects the standard output and the standard error channels into log files. They also periodically (every 10 seconds) log the *timestamp and the model prediction error over the complete testing dataset* as well as the number of messages received and the model parameters.

The error function that has been applied was the *average 0-1 error*, which means "averaging the ratio of incorrect predictions of the test instances over the whole network".

3.2 Results on Benchmark Data

Since there were not enough samples from the real QMedia client available at the time when the simulations took place, the method has been evaluated on the commonly used *SpamBase* classification dataset, which comes from UCI machine learning repository [10]. This data set represents an email database, where the emails are described by 57 features (e.g. word frequencies, character frequencies, length of the email etc.) and classified to spam or not spam classes. This type of database is widely used for testing spam detection methods, typically for email Junk-Filter development.

This data set has been divided into training and test sets; within the preparation of the

evaluation (the “training phase”), only samples coming from the training database have been delivered to the peers, for performing the update of the model with them. The test sets have then been used for the evaluation of the prediction performance.

Due to hardware limitations, a total of 400 peers, each having 10-11 local training examples, have been used for evaluation.

The result of the classification algorithm in QMedia is illustrated in *Figure 2*. The y-axis depicts the errors (normalized) over the time delay from the beginning of the simulation in seconds (x-axis). The gossiping period (called ‘Delta’ in *Listing 1* and ‘DELAY’ in the *Listing 2*) was set to 1 second.

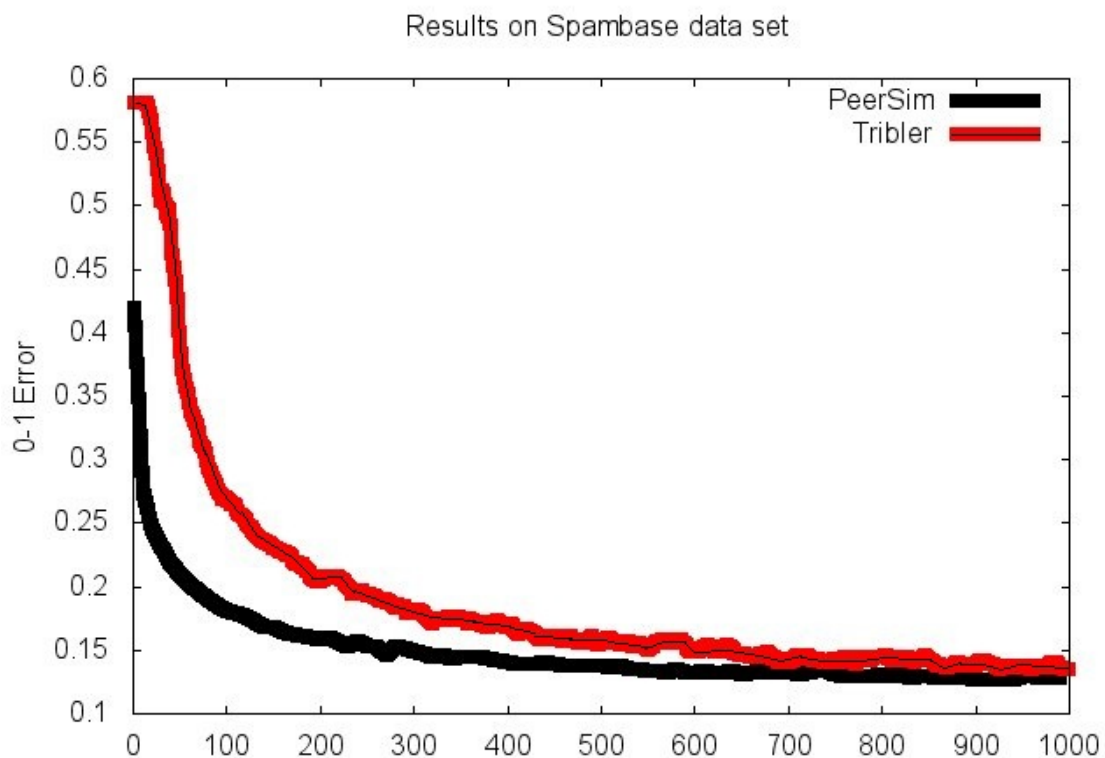


Figure 2: Experimental results on the *SpamBase* data set

Besides this emulation, a simulation within the PeerSim [11] simulation environment has been performed, thereby modelling similar conditions to those in QMedia. PeerSim is an extremely scalable simulation environment that supports dynamic scenarios such as churn and other failure models. The figure shows that the learning method, as

implemented in QMedia, can achieve about the same prediction performance as the simulation, but has a slightly slower convergence rate (since within PeerSim, Newscast has been used as overlay manager, whereas within QMedia, BuddyCast [12] is being used).

4 Results on user-tests

This section describes the set-up of the user tests and the testing methodology, provides insights on the processing of the tests, and finally evaluates their results. The final goal of the user evaluation was to gain further insights into the added value of HQ and SPAM metadata detection within the QMedia system. In addition to studying the successful integration of WP4.4 results and GoLF within QMedia, the evaluation also investigated the usability of the UI (User Interface), how the prototype was used by the participants, and the overall value attributed to the application by the participants.

The methodology used for evaluating the QMedia system was a Living Lab study, performed within the live network of QMedia, from within IRTs Experience lab.

4.1 Set-up of user-tests

This section describes the testing environment as well as the testing preparation and procedure.

4.1.1 Testing environment

Within IRTs Experience Lab, a 200 Mbit/s Telekom V-DSL connection [13] was used for connection of all QMedia clients. This ensured that no company-internal firewall influenced the P2P-related communication behaviour, and reflected the “real life” situation of users at home.

To perform the user tests, six QMedia clients were prepared on different types of PCs, with different hardware as well as different operating systems (MS Windows XP, MS Windows 7).



Figure 3: User testing at IRTs experience lab

4.1.2 Testing: Background & procedure

Background

The participants were 20 male and female employees and students at IRT, covering scientists, engineers and administrative officers (see *Figure 3*). Their mean age was 35.2 years. 70% of the participants are native speakers of German. On a 7-point scale (see Annex A), they indicated their English language proficiency to be 5.6 (70%), which corresponds to “fairly well” within the Questionnaire.

Within the channel testing community, a set of 40 torrents was created as the testing basis for the participants. In the tests, users were asked to:

- Get familiar with the QMedia system
- Create, analyse and modify metadata for entries within the community
- Analyse and eventually revert predictions given by the system
- Give reasons for any modification that was made by them
- Check reaction time, correctness and plausibility of the QMedia system

Figures 4, 5 and 6 (below) show screenshots from the QMedia User Interface. Selected channel is “IRT newtestingchannel” on which (besides another Favourite channel) torrents for the tests have been allocated.

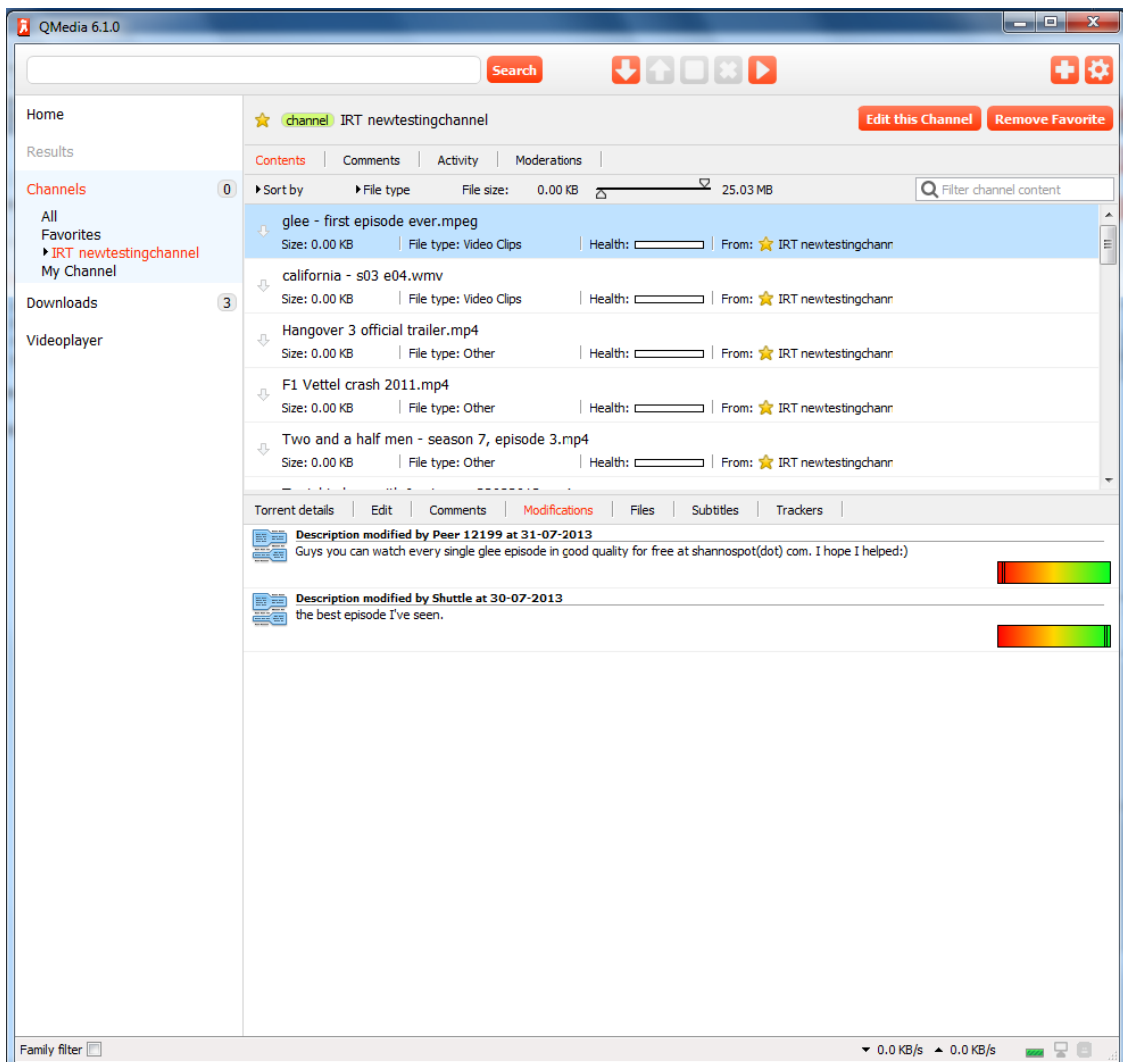


Figure 4: A SPAM and a NO_SPAM entry at the channel “IRT newtestingchannel”

Figure 4 depicts two descriptions for the selected torrent “glee – first episode ever.mpeg”, one classified by the system as a SPAM entry (represented by a vertical bar within the red area) , whereas the second one has been identified as NO-SPAM (marked as a vertical bar within the green area). Note that the current implementation does not display values in-between both conditions (e.g. probability of SPAM).

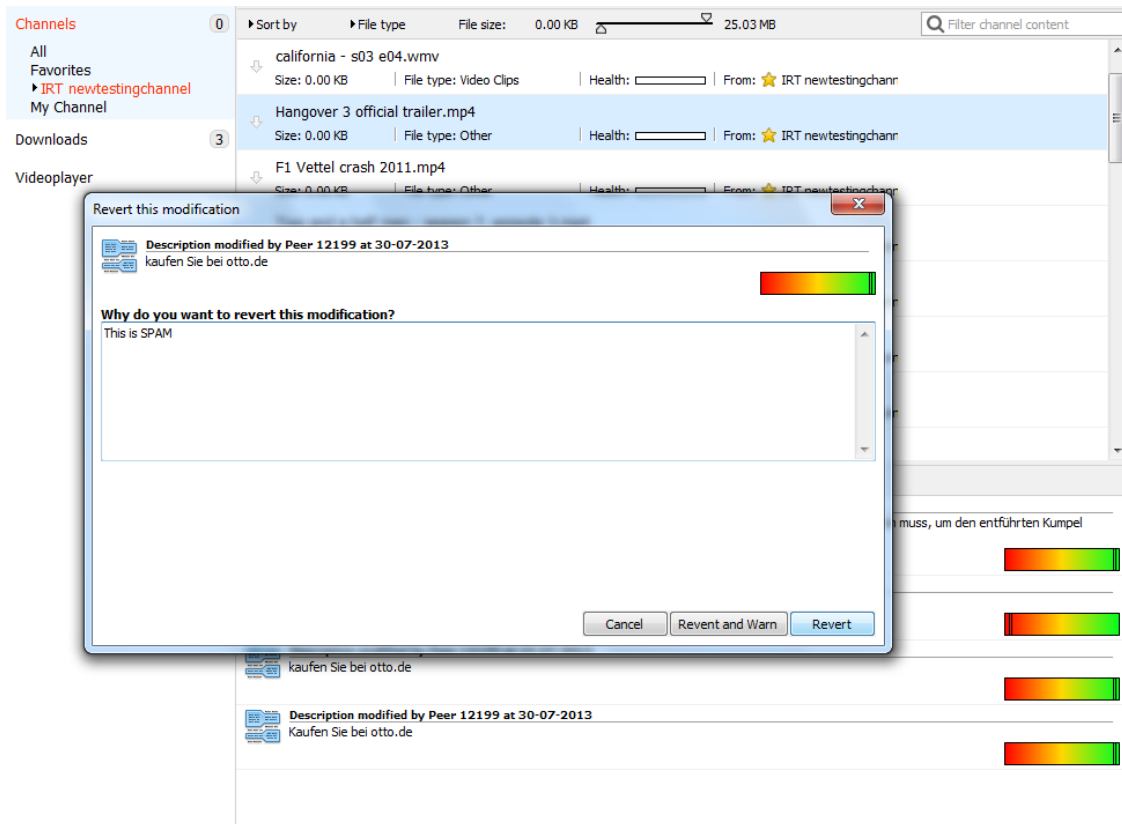


Figure 5: Reverting & reasoning for an entry

Figure 5 above depicts the interaction window to manually revert an entry. A text field offers the user to add a reason for the reverting, allowing others to better understand explicit ratings.

Besides the typical content-based listing of entries within a community, the QMedia system also allows the user to get an overview of all activities within the selected community by clicking onto the “Activity”-tab as shown in the screenshot below (Figure 6).

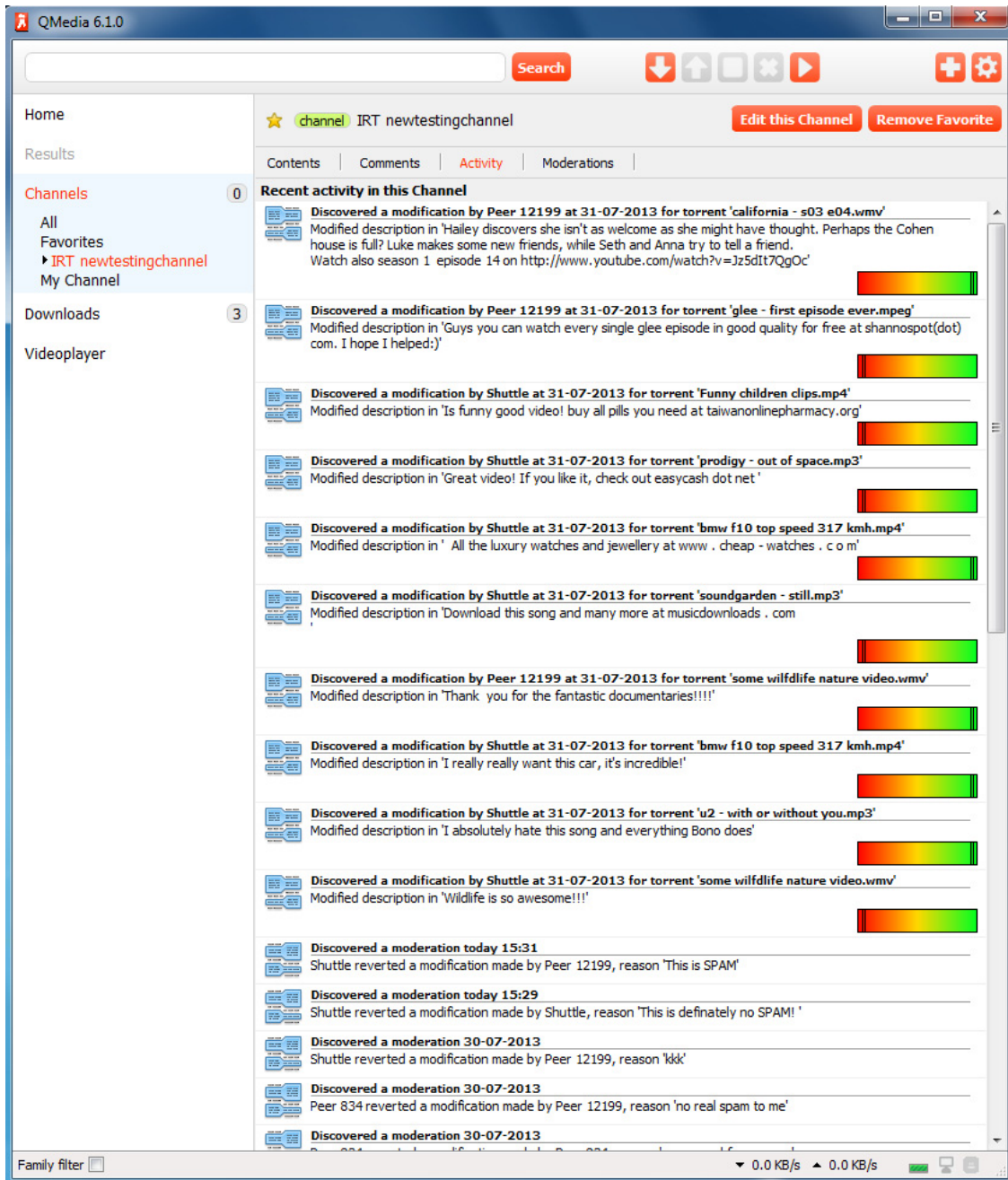


Figure 6: Activity-history at the channel “IRT newtestingchannel”

An overview of all moderations been performed within the community will be given by selecting the “Moderation” tab (see *Figure 7*).

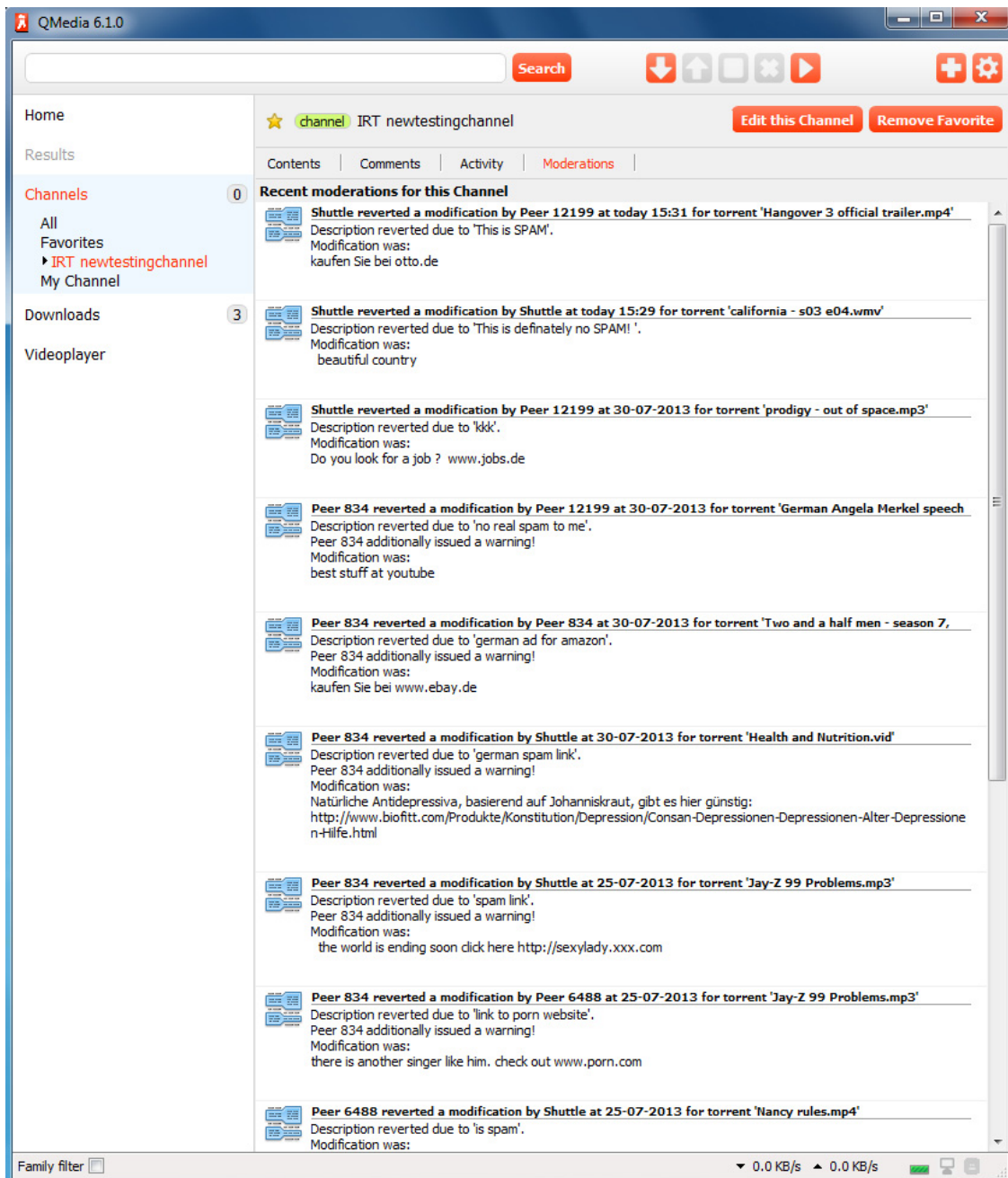


Figure 7: Moderations-history at the channel “IRT newtestingchannel”

For all measures under evaluation, questionnaires were used. These questionnaires were filled in by the participants before, during, and after the experiment. Appendix A

shows the contents of the questionnaire. In summary, the questionnaire contained the following sub-sections:

To be filled in before the experiment

- (An informed consent form)
- Demographic information about gender, age, education, first language, language proficiency
- Background information on how familiar users are with File-Sharing tools

Procedure

Before the evaluation

- Participants had to show up at the lab as agreed. They were offered snacks and drinks.
- The evaluators explained the procedure to the participants, especially everything closely related to the metadata prediction functionality of QMedia.
- Participants received instructions on how to perform the evaluation. Focus was laid on testing the GoLF integration for metadata quality prediction, since that was the key activity within WP4.4. Hence, users were encouraged to test the limit of the quality prediction system by also entering “SPAM” comments into the system.
- A common definition of “what SPAM is” was explained.
- Participants were requested to fill in questionnaires, including an informed consent form, demographic data, data regarding background information, and the integration aspects
- Participants received instructions on how to use the QMedia implementation

During the evaluation

- Behaviour of participants at the lab was observed (by an observer)
- Users were asked to take notes about wrong predictions of SPAM (metadata quality) and also to enter both SPAM and NO_SPAM descriptions.

After the evaluation

- Participants were asked to complete the questionnaire about QMedia and the integrated metadata quality prediction
- Within the lab, the QMedia tool and results were discussed in a group evaluation

4.2 Results from user-tests

This section describes the results from the evaluation of the user-tests, including suggestions for improvements as received from the participants. Evaluations on two types of implementation took place – one using the fully dynamic GoLF implementation for metadata quality prediction, and a second one using a pre-trained but static GoLF model.

The reason for the twofold-approach is described in the following sections.

4.2.1 Dynamic GoLF model implementation

The initial QMedia-tests with the integrated GoLF metadata prediction solution looked very promising; the system behaved as planned, so there was one client who “learned” a pre-trained model from within a specific training-community. After this, the client joined another community – the “testing-community” - where 5 other, untrained clients (un-trained GoLF-models) were already in place. GoLF worked as designed, and the untrained clients automatically started to learn via the network from the trained client, deploying the exchange of Gossip messages (as described in *section 2.4.2*). Further pre-tests were also promising, and also reverting and reasoning worked fine.

Therefore, the planned tests with users have been initiated.

Unfortunately, during user tests the quality prediction system became unstable after some time. This resulted in wrong labelling of the comments. Also, no reaction on metadata modifications was observed, manual reverts were not possible anymore and the prediction system stalled

A deep technical analysis by the partners from TU Delft and University of Szeged took place, resulting in several cycles of bug fixes / updates and pre-tests at IRT. Finally, the main hurdle identified was that the metadata entered by users and the metadata used within the training data set differed significantly: the weight of SPAM metadata compared to NO_SPAM entered by the users was much higher than within the training data set. This originates from the fact that users tried to force the GoLF system in terms of finding the limitations of the SPAM detection capabilities, and the GoLF system had to adapt to the environmental change compared to the training community. Models about SPAM

entries were exchanged intensively, which finally lead to a “too rude environment” for the algorithms that were trained for a “not so rude environment”. So, the need of a mechanism that can handle the continuous accumulation of the wrongly labelled data in the system was identified; for solving this issue, discussions about the implementation of a 'staging' mechanism took place. ‘Staging’ in this case would mean ‘holding back newly received inputs from participating in the training process until a certain time frame or while the sample is not labelled as SPAM’, which would result in maintaining a better training dataset and therefore more accurate and stable GoLF models. To do so, a time frame would need to be estimated for which newly generated data (without any label) would not be used for learning, but learning would start immediately once someone marks it as SPAM, or after a certain time without label, if no one marked it as SPAM. This time frame would certainly need to be estimated and optimised through intensive tests, which would imply an active community maintaining this mechanism.

However, this option (implementation of ‘staging’ in combination with an active community) could not be realised within the given time frame. The only possibility to finally still be able to perform user tests was to switch off the learning feature of the integrated GoLF subsystem after initial teaching of the models has been finished, and making use of a static GoLF model approach during the testing phase (as described in *section 4.2.2*).

4.2.2 Static GoLF model implementation

At first, general results concerning QMedia are being presented:

Many of the participants had used File sharing-tools before, but most of them “sometimes” on a five-point-scale ranging including “never”, “rarely”, “sometimes”, “often” and “always”, resulting in the following distribution:

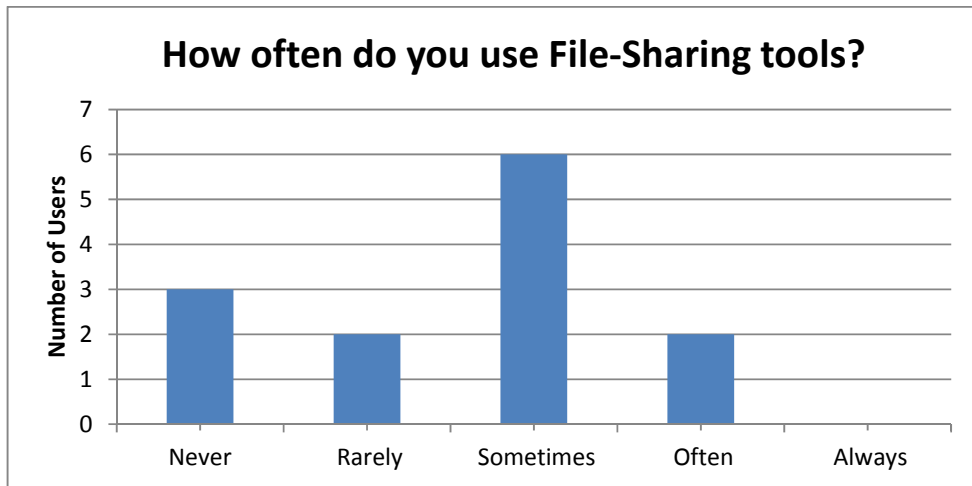


Diagram 1: How often have participants used File sharing-tools before?

However, most of the participants had never used QMedia/Tribler before, as depicted in *Diagram 2*.

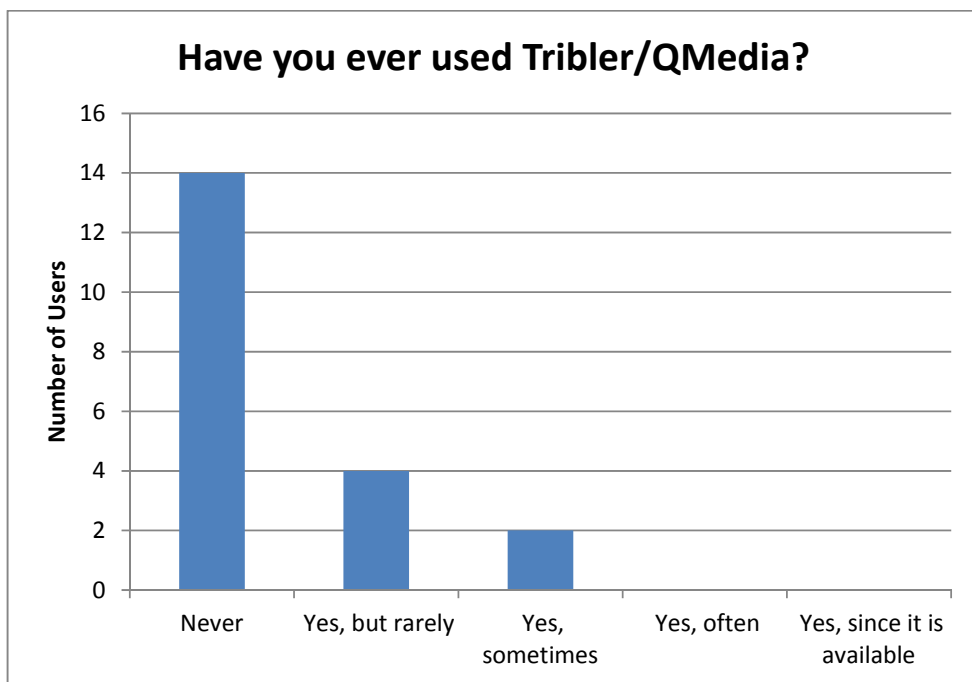


Diagram 2: How often have participants used Tribler before?

Although most users had never used QMedia before, they felt the User Interface (UI) to be rather intuitive to use, as depicted in *Diagram 3*. Most participants stated that everything was placed where expected, and titles, text and descriptions were easy to read. Feedback such as comments, ideas for improvement etc. on the UI is presented in *section 4.2.3*.

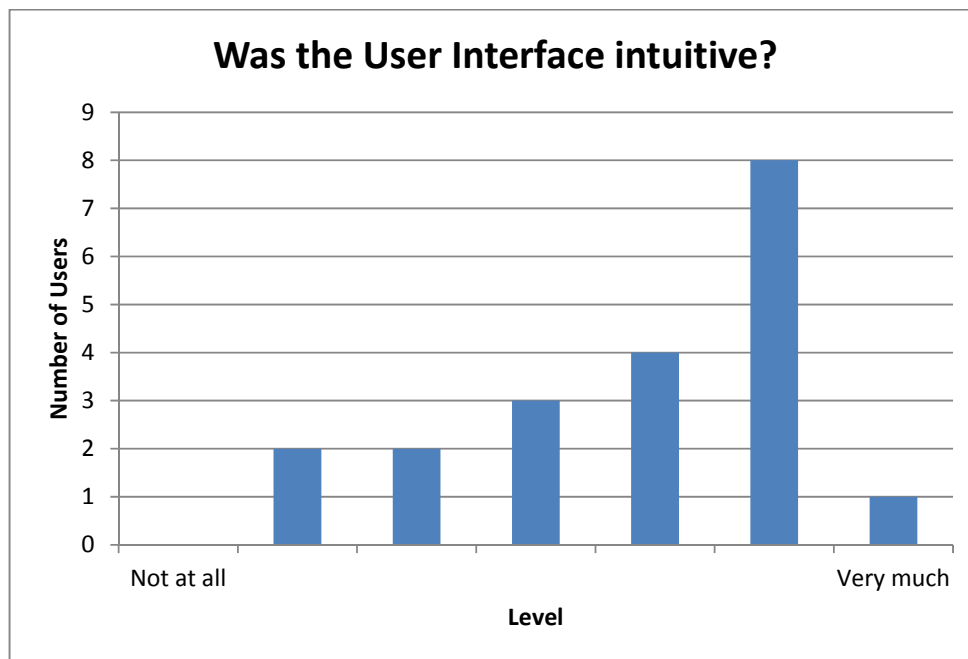


Diagram 3: How intuitive was the UI perceived?

Now, results concerning the metadata prediction integration are presented.

As participants had to at first analyse existing metadata that was created by other users within a separate community, they provided their feedback on how accurate they perceived the prediction of the metadata rating given by QMedia:

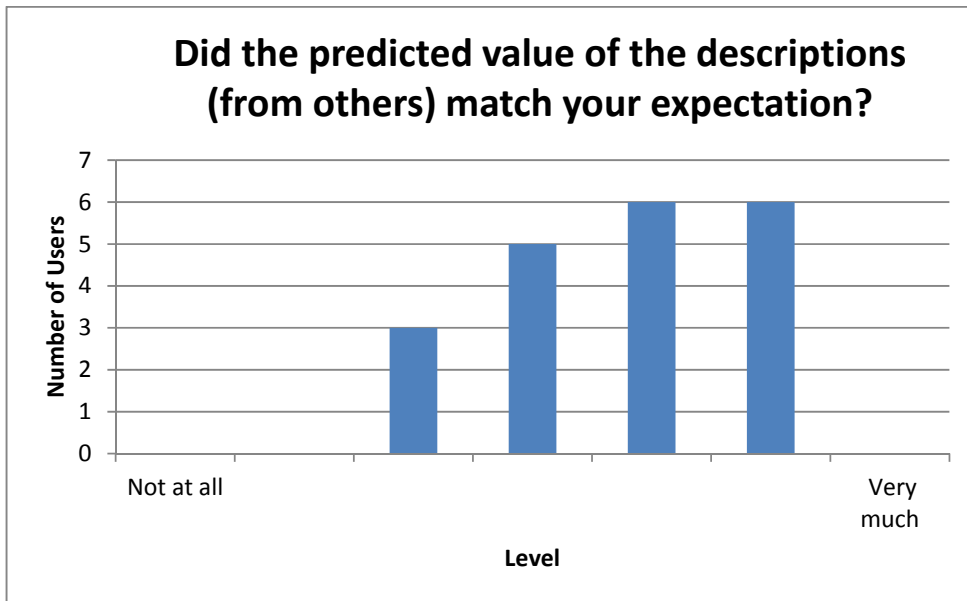


Diagram 4: How good was the metadata prediction for data from others?

Results were given on a 9-point scale, ranging from “*not at all*” to “*very much*”, and the average result over all participants was calculated to 59%, corresponding to “*adequate*”.

In the second part of the evaluation, participants had to enter another channel which was already filled with various different types of content (video, audio, e-books), but for which no metadata existed yet, except a filename / torrent-name, as usual within commonly used file sharing tools. There, participants were asked to randomly pick around 8 torrents, and describe those with metadata text they had in mind, taking into account both cases, consciously creating HQ-metadata as well as SPAM-metadata.

The results of this activity – as shown in diagram 5 below - were significantly higher, with an average of 68%, corresponding to “*fairly well*”.

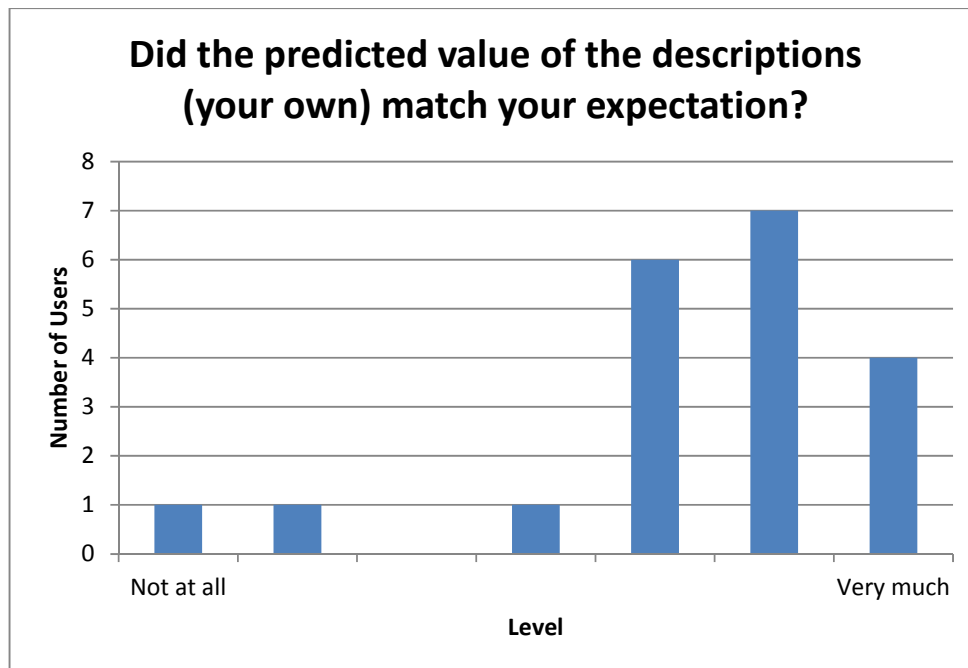


Diagram 5: How good was the metadata prediction for own metadata?

The number of metadata descriptions analysed in total (part 1 and 2) was 726, with an average of 36.3 items per participant. 117 predictions have been marked as wrongly classified by QMedia by the participants, resulting in a subjective error rate of 16 %, or in other words, a correct interpretation of SPAM / HQ metadata of 84%.

Participants were also encouraged to “revert” metadata predictions given by QMedia that they perceived to be wrong (as depicted in *Figure 5*). In total, 22 ratings (3%) were reverted and associated with a reason. The reasoning provided by users was felt to be of high value (to understand why something has been reverted) for 63% of the participants, as depicted in the following *Diagram 6*:

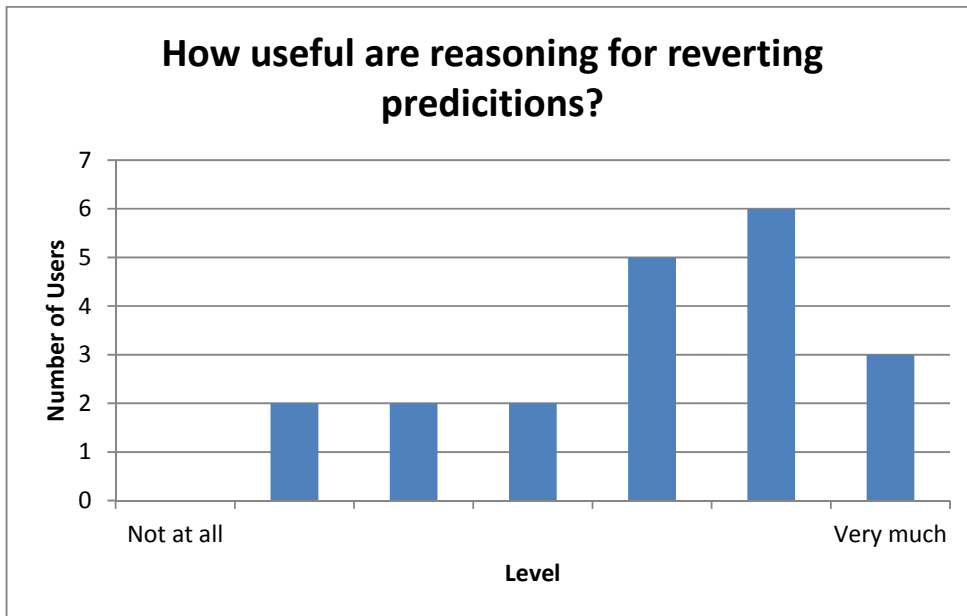


Diagram 6: How useful are the reasons for reverting predictions?

The purpose of reverting was to provide feedback to the metadata prediction mechanism and thereby allow dynamic adaptation and improvement of the models integrated into GoLF. The improvement of the prediction mechanism was perceived to have worked with 32 % of the users, depicted in *Diagram 7*.

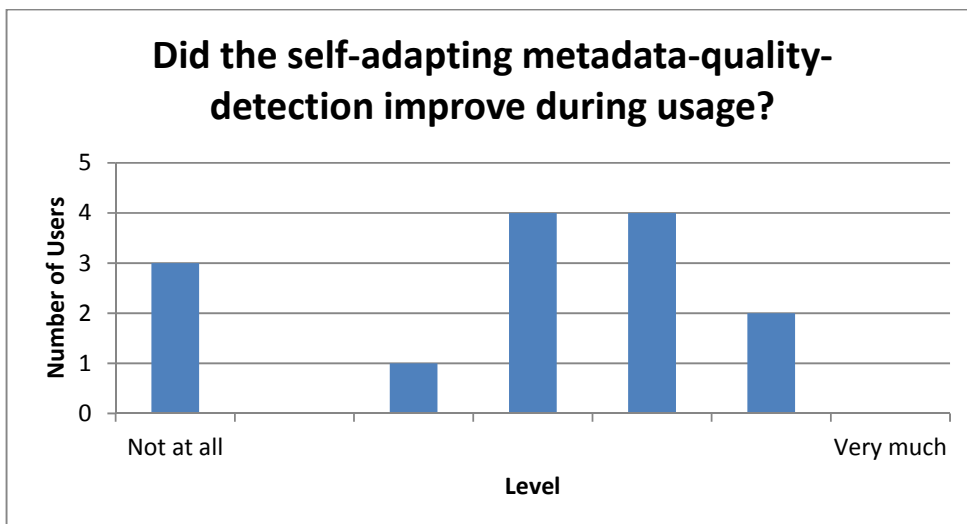


Diagram 7: Did the metadata prediction improve during usage?

4.2.3 Feedback from participants

"How would you propose to improve the UI?"	Text-input on "other"	Total votes
<input type="radio"/> More buttons		2
<input type="radio"/> Less buttons		1
<input type="radio"/> Just a search-bar (like inside a browser navigation bar)		
<input type="radio"/> Better explanations of the functionality ("Help")		4
<input type="radio"/> A demo video for explanation of the functionality		
<input type="radio"/> A multilingual UI		3
<input type="radio"/> Social Network integration		2
<input type="radio"/> Chat functionality		1
<input type="radio"/> Video chat support		
<input type="radio"/> A "private mode" (like in Mozilla Firefox)		8
<input type="radio"/> (other, please name)	Adult-filter	1
<input type="radio"/> (other, please name)	rating for content	1
<input type="radio"/> (other, please name)	detection value (instead of light)	1
<input type="radio"/> (other, please name)	bug-free software	1
<input type="radio"/> (other, please name)	less text	1
<input type="radio"/> (other, please name)	no Navi-Tree	1
<input type="radio"/> (other, please name)	large buttons / grid-layout	1
<input type="radio"/> (other, please name)	video thumbnails	1
Comments		
SPAM indication not clear		1
tooltips (added to topic "Help")		2
mouse-over explanations (added to topic "Help")		2

Table 1: Feedback from users

Analysing the feedback concerning UI improvements, three types of feedback have been given several times and are therefore to be considered as useful:

- "A private mode (like in Mozilla Firefox)"
- "Better explanations of the functionality"
- "A multilingual UI"

The first one is, from a technical point of view, rather hard to implement, since Internet Service Providers usually are forced to keep connection data (i.e. IP addresses) of all their customers for a certain time period [14].

According to the second feedback item, the functionality of QMedia is proposed to be more explained. In addition, two kinds of possibilities for the implementation of this have been given, “Tooltips” and “Explanations on Mouse-Over”.

A multilingual UI was proposed by the (few) non-native German speaking users.

5 Conclusions

This section draws conclusions on WP4 results and its integration within QMedia.

In QMedia, a software solution allowing for the evaluation of additional information (metadata) and the prediction of its quality was realised and tested through the form of user tests at IRT. A SPAM detection mechanism was used to analyse descriptive text of torrent files, which was generated and edited by users through the UI of QMedia (“Open2Edit”: see [15]).

Throughout the user tests, the participants entered descriptions for the torrent files being available within the testing channel. They were asked to analyse the quality given through the system in terms of predictions and compare their own predictions with the ones given by QMedia. Moreover, users created own metadata and analysed the predicted label given for those entries. A common understanding of the definition of SPAM is essential to come to evaluable results. The definition applied in QMedia was “any kind of information not being related to the torrent file itself, thereby aiming to relocate the user to a website offering content advertisements”.

The test results show that the system basically works well. Some of the participants were really surprised how reliably QMedia correctly detected the SPAM metadata that was entered by them by intention – i.e. to find the borders of what types of SPAM QMedia can detect. However, the system could still be improved and achieve a higher detection rate if the learning mechanism of GoLF would allow different clients to learn from each other (refer to *section 4.2.1*). To do so, ‘staging’ was identified as one solution for managing vandalism / overweighed positive or negative entries. A broader set of teaching data, covering SPAM and NO_SPAM labelled text from additional sources besides YouTube (which has been used for training solely) could also help improve the initial pre-teaching of the GoLF models. Moreover, it should be noted that, as with every self-learning algorithm, accuracy improves with larger training sets.

It is also worth mentioning that the existing implementation could easily be enhanced in a way to support quality prediction support for all kinds of text fields, e.g. comments in a

chat discussion, a social network plugin, genre entries etc. Since the underlying GoLF implementation is flexible with respect to the implemented features its models support, additional types of functionality could be added to QMedia, e.g. a self-learning implementation of an Adult-Text filter.

Finally it can be concluded that QMedia with integrated GoLF adds value to the users, especially when it comes to search and retrieval of content based on metadata, which is not supported within other common Bittorrent clients.

6 References

- [1] QLectives “Algorithms for detecting and handling High Quality Metadata (HQ-MD) and Spam Metadata” (D4.4.2):
<http://www.qlectives.eu/docs/D4.4.2.pdf>
- [2] Bittorrent definition: <http://en.wikipedia.org/wiki/BitTorrent>
- [3] B. Cohen. Incentives build robustness in bittorrent. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, USA, 2003.
- [4] Bootstrapping peers in Tribler: <http://www.tribler.org/trac/wiki/BootstrapTribler>
- [5] Tribler statistics: <http://statistics.tribler.org>
- [6] R. Ormándi, I. Hegedüs, and M. Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, pages n/a–n/a, 2012.
- [7] N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.
- [8] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for SVM. *Mathematical Programming B*, 2010.
- [9] I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic curves in cryptography*. Cambridge University Press, New York, NY, USA, 1999.
- [10] A. Frank and A. Asuncion. UCI machine learning repository, 2010.
- [11] A. Montresor and M. Jelasity. Peersim: A scalable P2P simulator. In *Proc. 9th IEEE Intl. Conf. on Peer-to-Peer Computing (P2P 2009)*, pages 99–100. IEEE, September 2009. Extended abstract.
- [12] BuddyCast protocol: QLectives Deliverable D4.3.1:
<http://www.qlectives.eu/publications/public-deliverables>
- [13] Telekom V-DSL:
http://www.telekom.de/is-bin/INTERSHOP.enfinity/WFS/EKI-PK-Site/de_DE/-/EUR/ViewProductDetails-Start?ProductRefID=0100051000227%40EKI-PK&KeywordPath=katalog%2Fsurfen%2Fkomplettpakete%2Fcomfortpakete%2Fcomfort-speed&StageProductRefID=0100051000227_0003%40EKI-PK&CatalogCategoryID=qPQFC7IUXGsAAAE5wxxXlos3

[14] Telecommunications data retention:

http://en.wikipedia.org/wiki/Telecommunications_data_retention

[15] QLectives Deliverable D4.3.3, chapter 2 “Open2Edit”:

<http://www.qlectives.eu/publications/public-deliverables>

7 Annex A – Questionnaire

In this Annex, the Questionnaire used within the user tests as well as background information on preparation and execution of the tests is presented.

Demographic information

<i>Gender</i>	Male / female
<i>Age</i>	____ year
<i>Job / Education</i>	
<i>First language</i>	English / other,
<i>Language skills</i>	very bad <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> very good

Background information

How often do you use File-Sharing tools ?

Never Rarely Sometimes Often Always

Have you ever used Tribler / QMedia ?

Never Yes, but rarely Yes, sometimes Yes, often Yes, since it is available

Metadata Quality Prediction information

Was the User Interface (UI) intuitive ?

Not at all Very much no answer

Were the search results matching your search criteria ?

Not at all Very much no answer

Were the comments on the videos easy to read ?

Not at all Very much no answer

Did the "SPAM-Metadata" detection work properly ?

Not at all Very much no answer

Did the predicted value of the description (from others) match your expectation?

Not at all Very much no answer

Did the predicted value of the description (your own) match your expectation?

Not at all Very much no answer

Did the self-adapting metadata-quality-detection improve during usage ?

Not at all Very much no answer

How many descriptions did you analyse ?

How many ratings of descriptions did you revert ?

How useful have reasonings for changes of ratings given by other users been to you ?

Not at all Very much no answer

How many wrong predictions for "SPAM" text content did the system give ?

 out of

Will you use File-Sharing tools in the future ?

- Never Rarely Sometimes Often Always

Will you use Tribler / QMedia in the future ?

- Never Rarely Sometimes Often Always

How would you propose to improve the UI ?

- More buttons
- Less buttons
- Just a search-bar (like inside a browser navigation bar)
- Better explanations of the functionality
- A demo video for explanation of the functionality
- A multilingual UI
- Social Network integration
- Chat functionality
- Video chat support
- A "private mode" (like in Mozilla Firefox)
- (other, please name)
- (other, please name)
- (other, please name)

Room for comments:

For persons supervising tests on QMedia:

Make sure to have:

- Prepared copies of questionnaires (Lab version) for each participant (including informed consent forms)
- Instructed users to put their name on informed consent form
- Writing materials (pens) for every user available
- contact information from participants

Actions:

Welcome

Order food & drinks (do first as it might take a while before the food is delivered!)

Explain evaluation to participants

- Planning of the experiment
- Procedures and instructions

Let participants fill in first part of questionnaires:

- First page
- Informed consent form

Instructions:

- Instruction for and practicing with QMedia
- Instruction for editing metadata
- Instruction for analysing ratings, reverting and reasoning