School of
Computer Science

Information
Management Group

HCW— HuCEL Technical Report 3, June 2009

HuCEL: Links Experiment Manual

Paul Waring
Human Centred Web Lab
School of Computer Science
University of Manchester
UK

The World Wide Web contains a vast corpus of information describing a variety of events, but this information is poorly interconnected. The aim of the HuCEL project is to provide a solution to this problem by automatically generating associative links between related events. This manual describes how to re-run an investigation into how users perceive the relatedness of links generated by keywords from humans against those from an automated parser.

HCW
Human Centred Web

HuCEL

The aim of the HuCEL project is to investigate how related events can be connected on the Web, in order to improve navigation of the hypertext space and enable users to serendipitously discover new information. The HuCEL Web pages may be found at: `http://hcw.cs.manchester.ac.uk/research/hucel/`.

HuCEL Reports

This report is in the series of HCW HuCEL technical reports. Other reports in this series may be found in our data repository, at `http://hcw-eprints.cs.man.ac.uk/view/subjects/hucel.html`. Reports from other Human Centred Web projects are also available at `http://hcw-eprints.cs.manchester.ac.uk/`.

Contents

1 Introduction **1**

2 System Requirements **1**

3 Data Storage **1**

4 Installation **5**

5 Flow Control **5**

6 Common Errors **6**

7 Summary **6**

8 Associated Files **7**

Human Centred Web Lab
School of Computer Science
University of Manchester
Kilburn Building
Oxford Road
Manchester
M13 9PL
UK

tel: +44 161 275 7821
http://hcw.cs.manchester.ac.uk/

Corresponding author:
Paul Waring
tel: +44 (161) 275 6239
pwaring@cs.man.ac.uk

1 Introduction

This manual explains how to re-run the links experiment of the HuCEL project, which is an investigation into how users perceive the relatedness of links generated by keywords from humans against those from an automated parser.

The experiment can be re-run without any changes to the code supplied with this manual. However, some work will be required on the part of the investigator, both to set up the experiment and to recruit participants. Furthermore, as this experiment is part of a series, it requires the keywords obtained in the previous two HuCEL experiments [2, 1] in order to generate all the links required. These keywords can be obtained either by utilising the results from the initial investigations or by re-running the two experiments again.

It is also possible to re-analyse the data collected during the first run of the experiment, conducted between 27 October 2008 and 6 December 2008.

This report assumes that the reader is familiar with the PHP programming language and basic SQL queries, and is capable of uploading files to a publicly accessible Web server using a mechanism such as FTP.

2 System Requirements

In order to run the experiment code, certain minimum software requirements must be met. The experiment may run under older versions of software, but no guarantees are made as to whether this will be the case, and users are strongly recommended to use the latest versions where possible. The minimum requirements include:

- Web server capable of supporting PHP 5 – Apache 2.2 is recommended.
- PHP 5.2.0[1]
- SQLite 2.1.0[2]

The experiment code should run on any operating system and platform which is capable of meeting the above system requirements. However, all instructions in this report assume a Unix-like operating system such as Linux, FreeBSD or Mac OS X.

3 Data Storage

All of the data collected in the course of the experiment is stored in an SQLite[3] database. To ensure maximum backwards compatibility, SQLite version 2 is used.

The majority of configuration options are also stored in the database, so it is possible to re-run the experiment with different Web pages and keywords without changing the underlying PHP code. Furthermore, the HTML output is generated

[1] Support for accessing SQLite databases in built in to PHP 5 by default, so no action should be required to activate this functionality unless it has been explicitly disabled.

[2] The database used for this experiment is created in SQLite 2. Attempting to use a database created in SQLite 3 will result in error messages suggesting a 'corrupt database'.

[3] `http://www.sqlite.org/`

from templates, so the look and feel of the experiment can also be altered without changing the PHP code. It should only be necessary to edit `experiment.php` if changes to the logic of the experiment are required – for example, if users were to be shown more than one link from each group.

The schemas for the various database tables are described in the following sections.

Table 1: Schema for `participants` table

Field Name	**Data Type**	**Additional**	**Notes**
id	INTEGER	PRIMARY KEY	Unique number for identifying individual participants
starttime	INTEGER	NOT NULL	Timestamp for beginning of experiment
endtime	INTEGER		Timestamp for end of experiment (0 if user does not complete)
gender	CHAR(1)	NOT NULL	Set to 'm' or 'f'
age	VARCHAR(10)	NOT NULL	Age range of the user
time_web	VARCHAR(10)	NOT NULL	Time user spends on Web each week
english_native	CHAR(1)	NOT NULL	'y' if user is a native English speaker, 'n' if not

Table 2: Schema for `webpages_config` table

Field Name	**Data Type**	**Additional**	**Notes**
id	INTEGER	PRIMARY KEY	Unique number for identifying individual Web pages
original_url	VARCHAR(255)	NOT NULL	Original URL of the Web page
local_path	VARCHAR(255)		Path to the local copy of the Web page

Table 3: Schema for `describe_links` table

Field Name	**Data Type**	**Additional**	**Notes**

id	INTEGER	PRIMARY KEY	Unique number for identifying individual links
webpage	INTEGER	NOT NULL	Foreign key on `webpages_config.id`
url	VARCHAR(255)	NOT NULL	URL of the page the link leads to
title	VARCHAR(255)	NOT NULL	Title of the page the link leads to
summary	VARCHAR(255)	NOT NULL	Summary of the page the link leads to
ratings_count	INTEGER	NOT NULL	Number of ratings this link has received

Table 4: Schema for `related_links` table

Field Name	**Data Type**	**Additional**	**Notes**
id	INTEGER	PRIMARY KEY	Unique number for identifying individual links
webpage	INTEGER	NOT NULL	Foreign key on `webpages_config.id`
url	VARCHAR(255)	NOT NULL	URL of the page the link leads to
title	VARCHAR(255)	NOT NULL	Title of the page the link leads to
summary	VARCHAR(255)	NOT NULL	Summary of the page the link leads to
ratings_count	INTEGER	NOT NULL	Number of ratings this link has received

Table 5: Schema for `parser_links` table

Field Name	**Data Type**	**Additional**	**Notes**
id	INTEGER	PRIMARY KEY	Unique number for identifying individual links
webpage	INTEGER	NOT NULL	Foreign key on `webpages_config.id`

url	VARCHAR(255)	NOT NULL	URL of the page the link leads to
title	VARCHAR(255)	NOT NULL	Title of the page the link leads to
summary	VARCHAR(255)	NOT NULL	Summary of the page the link leads to
ratings_count	INTEGER	NOT NULL	Number of ratings this link has received

Table 6: Schema for `link_ratings` table

Field Name	**Data Type**	**Additional**	**Notes**
participant	INTEGER	NOT NULL	Foreign key on `participants.id`
starttime	INTEGER	NOT NULL	Timestamp for when user began looking at this page
endtime	INTEGER	NOT NULL	Timestamp for when user finished looking at this page (0 if user quit experiment at this point)
webpage	INTEGER	NOT NULL	Foreign key on `webpages_config.id`
describe_link_id	INTEGER	NOT NULL	Foreign key on `describe_links.id`
describe_link_rating	INTEGER	NOT NULL	Rating given to `describe_link_id`
related_link_id	INTEGER	NOT NULL	Foreign key on `related_links.id`
related_link_rating	INTEGER	NOT NULL	Rating given to `related_link_id`
parser_link_id	INTEGER	NOT NULL	Foreign key on `parser_links.id`
parser_link_rating	INTEGER	NOT NULL	Rating given to `parser_link_id`
link_order	CHAR(3)	NOT NULL	Order in which the links were displayed on the page
link_preference	CHAR(1)	NOT NULL	Link which user preferred

qualitative_feedback	TEXT		Feedback left by the user on the links (optional)

For the fields `url`, `title` and `summary` in the tables `describe_links`, `related_links` and `parser_links` are taken directly from the results of a Yahoo! search API query.

4 Installation

Once all of the system requirements have been fulfilled, installing the code is a simple case of uploading the files to a publicly accessible directory (e.g. `public_html`) via a mechanism such as FTP or the `scp` command. After uploading the files, the correct permissions must be set in order for the experiment to run correctly.

The majority of files within the experiment will only require read access for all users (`chmod 644`) and directories will require read and execute access (`chmod 755`). However, there are some exceptions where files or directories require different permissions:

- `templates_c` – all users should have read, write and execute access (`chmod 777`).
- `data.db` – all users should have read, write and execute access (`chmod 777`).

In addition to the above requirements, the directory which contains all of these files (e.g. `public_html`) needs to allow all users read, write and execute access.

5 Flow Control

The flow of the experiment code follows a simple process:

1. Find out which page the user has just submitted.
2. Process the form data from this page and save it to the database.
3. Decide what the next page to display should be – at most there will be two options to choose from.
4. Fetch any information from the database which is required to display the next page (e.g. Web page configuration details).
5. Display the next page.

There are two ways for the participant to break this process. Before the experiment begins, the user is shown an example page and asked if he understands what is required during the experiment. If the user selects 'no' at this point, he will

be redirected to the final page and no useful data will be recorded. Alternatively, the user can finish the experiment by closing the browser window. In this case, any information entered on the current page will not be saved, but all data entered on previous pages will be retained. Finally, if the participant ends the experiment before the final page, no end time will be recorded.

6 Common Errors

Some common error messages which may occur when installing the experiment code include:

- 'Unable to open database file': Usually indicates that the permissions are not set correctly on the directory which contains the SQLite database.
- 'Corrupt database': Indicates that the database is in SQLite 3 format, as opposed to SQLite 2.
- Blank page displaying: Suggests that the permissions on the `templates_c` directory are not set correctly.

7 Summary

Using the instructions contained in this manual, readers should be able to re-run the HuCEL links experiment or extend the experiment to capture additional data. The additional files included with this manual also allow readers to re-analyse the data collected in the original run of the experiment.

References

[1] Paul Waring. HuCEL: Keywords Experiment II Manual. Technical Report, University of Manchester, `http://hcw-eprints.cs.man.ac.uk/89/`, 2009.

[2] Paul Waring. HuCEL: Keywords Experiment Manual. Technical Report, University of Manchester, `http://hcw-eprints.cs.man.ac.uk/88/`, 2009.

8 Associated Files

In addition to this manual, the repository also includes the following files:

- `experiment-code.zip`: A compressed file containing all of the code necessary to re-run the experiment, including an SQL file for recreating the database.
- `data.db`: An SQLite database containing all of the data obtained from the experiment.

Within the `experiment-code.zip` archive are the following folders and files:

- `experiment.php`: The PHP code which provides an interface between the user and the database, responsible for controlling the flow of the experiment as described in Section 5.
- `webpages`: Directory containing static copies of all the Web pages used in our run of the experiment, so that all participants will see the same page.
- `templates`: Smarty[4] templates for all the pages which will be shown to participants in the course of the experiment.
- `index.html`: The first page of the experiment, which participants should be directed to once the code is up and running.
- `style.css`: Stylesheet for the experiment.
- `validation.js`: Client-side validation routines (JavaScript).
- `create.sql`: SQL commands to recreate the database structure and populate it with initial configuration.
- `empty.db`: An SQLite database set up with all the required tables and configuration (including all the links generated from the two keywords experiments), but no user data.

[4]Smarty is a template engine for PHP. It can be downloaded from `http://www.smarty.net` and is available as a package for numerous platforms.