KEYSTONE - Short scientific report for STSM visit in TU Delft

Visitor: Georgia Kapitsaki
Host: TU Delft

1. Introduction

This report covers the activities performed during my STSM at Delft University of Technology (TU Delft) in the framework of the Cost Action KEYSTONE IC1302 (semantic KEYword-based Search on sTructured data sOurcEs). The activities were relevant to the design and implementation of a tool on Web API usage from JavaScript in web pages, and on a preliminary study of the literature on Context information extraction from social media structured sources.

2. Purpose of the STSM

The initial concept of the STSM as described in the STSM proposal was to focus on the area of Context information extraction from social media structured sources. In that framework, the purpose was to investigate techniques and methods that can lead to the extraction of context-relevant data from user models/profiles in social media along with the understanding of social media-based user context metamodel that is vital for the proper information extraction in that context.

However, in the beginning of the STSM along with the host institution we decided to perform during the visit only a preliminary study on the above and focus on another activity relevant to information extraction from web pages. Specifically, the purpose was to design and implement a tool on Web API usage from JavaScript in web pages motivated by the announcement of the Norvig Web Data Science Award[1] that provides access to the Common Crawl dataset. The main aim of the award is to promote the study of the web giving to participants the possibility to investigate and draw useful conclusions on web usage from the existing dataset. In that sense we decided to investigate the use of web APIs in scripts integrated in web pages. Specifically, we wanted to see how we can extract information on calling a web API from JavaScript with a focus on the invocation of web services, but not only limited to that.

3. Main activities performed

As aforementioned the main activities were centered on the two main areas relevant with the purpose of the STSM.

3.1. Web API usage from JavaScript in web pages

During the visit I designed and implemented an initial prototype version for a tool for the analysis of scripts in Web pages for the invocation of Web APIs called ***webanalifer***. The motivation was as aforementioned the Norvig Web Data Science Award that would provide access to the dataset of Common Crawl and to the Dutch Hadoop cluster, where experiments could be run. During the duration

[1] http://norvigaward.github.io/

of the STSM (covering the period of 3 weeks out of the 4 weeks visit) the experiments were mainly run locally on a subset of the dataset that was provided for download by Common Crawl.

Description of the ***webanalifer***:
HTML pages contain numerous scripts most of which are in JavaScript and contain invocations of popular Web APIs (e.g., Facebook, LinkedIn) and Web services (either SOAP-based services or RESTful services). In order to see which web APIs are most popular and draw conclusions on their usage and the existing types of invocation, we decided to analyze the existing scripts in web pages. For this purpose we followed the process with the activities described next.

1) *Study of the structure of Web ARChive (WARC) files*: WARC format is a revision of the Internet Archive's ARC File Format that has traditionally been used to store "web crawls" as sequences of content blocks harvested from the World Wide Web. The files contained in the Common Crawl dataset followed the WARC format and for this reason it was necessary to understand their structure and extract the HTML pages contained in WARC files.

2) *Determination of the possible patterns used to call a web API from a web page using JavaScript*. By studying pages on the web, examples in developer resources (e.g., StackOverflow) and the existing literature on provision of web APIS, such as [3], we identified the following ways:

A. Using an XML HTTP request: in this case a new XMLHttpRequest or a new ActiveXObject object - depending on the type and version of the browser - is created before the request is sent to a specific URL. An example is provided below:

```
var myReq = new XMLHttpRequest();
if (window.XMLHttpRequest) {
      var url = "http://localhost:49216/WebXmlHttpDemo/books.xml"
      myReq.open("GET", url, false)
      myReq.send();
      alert(myReq.responseXML.xml);
}
```

B. Through jQuery requests (using either HTTP GET or POST). An example is provided below:

```
$.get("http://mysite.com/mywebservice", {
      paramOne : 1,
      paramX : 'abc'
}, function(data) {
      alert('page content: ' + data);
});
```

C. Through jQuery requests with AJAX (using either HTTP GET or POST). An example is provided below:

```
jQuery.ajax({
      url :
'http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/mov
```

```
iefunctions.asmx',
      async : true,
      type : 'POST',
      dataType : 'json',
      success : function(response) {
            var employeeDetails = '<b>Employee Details Table</b><br
/><table>';
            for (i = 0; i < response.length; i++) {
                  employeeDetails = employeeDetails + '<tr><td>'
                  + response[i]['Id'] + '</td><td>'
                  + response[i]['Name'] + '</td><td>'
                  + response[i]['Designation'] + '</td></tr>';
            }
            $('#divEmployeeDetails').append(
                        employeeDetails + '</table>');
            }
});
```

D. Through AJAX request with the Prototype JavaScript framework[2] (using either HTTP GET or POST). An example is provided below:

```
new Ajax.Request(testUrl, {
      method : 'post',
      onSuccess : function(transport) {
      var response = transport.responseText || "no response text";
      alert("Success! \n\n" + response);
      },
      onFailure : function() {
            alert('Something went wrong...');
      }
});
```

Currently it seems that the most employed ones are AJAX requests with JQuery followed by XML HTTP requests.

3) Design and implementation of the parser

I implemented a parser that performs the following actions: 1. Reads a WARC file and extracts the HTML code of the page (body of the HTML), 2. Identifies the JavaScript sections of the page (either inline JavaScript, or from a local or remote location), 3. Parses the script based on the patterns specified previously, and 4. Gathers information on the use of web APIs for all the scripts of the web page.

4) Statistics

For each page different information is gathered, such as the number of scripts contained in the page, the type of each script (e.g., JavaScript, VBScript), the type of requests performed in each web API request (e.g., AJAX, JQuery), the type of response format for the web request – if available (e.g., XML, JSON, plain text).

[2] http://prototypejs.org/

3.2. Context information extraction from social media structured sources

Since the Web Information Systems (WIS) group at the host already had experience on user modeling from social media I came in contact with members of the group and studied the relevant literature from existing publications of some former members of the group ([1, 2] among others). This activity was performed during the first week of the visit as indicated in the proposal submitted covering the WP1 (Preparation and background study) of the work plan. At that stage it was realized that it was better to work on a smaller task for the duration of the STSM and continue working in this area in the future (section 3.1).

3.3. Side activities

During my visit I also came in contact with other members of the group:

- Prof. Geert-Jan Houben (Full Professor)
- Dr. Claudia Hauff (Assistant Professor)
- Ke Tao (PhD student)
- Jie Yang (PhD student)
- Dr. Alessandro Bozzon (Assistant Professor)

and discussed with them the research areas they are working on.

4. Main results obtained

The above activities (described mainly in 3.1) resulted in a code that consists of 15 Java classes and uses different existing libraries in order to carry out the activities of the tool (e.g., Rhino, JSoup). The code is currently available locally, but in the future the respective project will be uploaded on GitHub.

At the current state of the prototype implementation of the tool, the webanalifer was run on a very small subset of the Common Crawl dataset provided as example provided online[3] (its size is 943MB). Using this test set we extracted information on each web page. We observed that in many cases the invocations were performed using more complex constructs as the commonly used patterns identified (for instance, the request URL would be constructed using a concatenation of the values of many variables of the script) that would give as output many results that were not relevant to the invocation itself or that would not allow the drawing of useful conclusions on which web APIs are actually used. Note that the invocations contained also a lot of noise (e.g., actions relevant to the presentation of the page).

5. Future work and collaboration

Since the period of the visit was not sufficient to complete all the activities of the foreseen research work, I will continue the work in collaboration with the host institution. Specifically, with the help of the members of the group of Web Information Systems (WIS) - mainly along with Dr. Claudia Hauff – we will investigate the following:

- *On Web API usage*:

[3] http://beehub.nl/surfsara-hadoop/public/CC-TEST-2014-10-segment-1394678706211.tar.gz

completion of the work and execution on a larger dataset along with analysis of the obtained results (e.g., providers; distribution, most popular APIs, most common combinations of web APIs etc.)

- *On context information extraction*:
  activities relevant to WP3 and WP4 and follow-up activities

References

[1] Ilina, E., Hauff, C., Celik, I., Abel, F., & Houben, G. J. (2012). Social event detection on twitter. In Web Engineering, pp. 169-176, Springer Berlin Heidelberg.

[2] Abel, F., Herder, E. , Houben, G.-J., Henze, N., Krause, D. (2013). Cross-system user modeling and personalization on the Social Web. User Model. User-Adapt. Interact. 23(2-3), pp. 169-209.

[3] Maleshkova, M., Pedrinaci, C., Domingue, J., (2010). Investigating Web APIs on the World Wide Web, *Web Services* (ECOWS), 2010 IEEE 8th European Conference on , vol., no., pp.107,114, 1-3.