Let Google Index Your Media Fragments

Yunjia Li,Mike Wald and Gary Wills
School of Electronics and Computer Science
University of Southampton
UK
{yl2,mw,gbw}@ecs.soton.ac.uk

ABSTRACT

Current multimedia applications in Web 2.0 have generated a massive amount of multimedia resources, but most search results for multimedia resources still focus on the whole resource level. Media fragments expose the inside content of multimedia resources for annotations, but they are yet fully explored and indexed by major search engines. W3C has published Media Fragment 1.0 as a standard way to describe media fragments on the Web. In this proposal, we make use of Google's Ajax Application Crawler to index media fragments represented by Media Fragment URIs. Each media fragment with related annotations will have an individual snapshot page, which could be indexed by the crawler. Initial evaluation has shown that the snapshot pages are successfully fetched by Googlebot and we are expecting more media fragments to be indexed using this method, so that the search for multimedia resources would be more efficient.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: Audio, Video and Hypertext Interactive Systems

General Terms

Web,Search,Design

Keywords

media fragment,annotation,microdata,Google search

1. INTRODUCTION

The term "media fragment" refers to the inside content of multimedia objects, such as a certain area within an image, or a five minutes segment within a one-hour video. With the rapid development of multimedia applications, such as YouTube and Flickr, end users could easily upload, share and tag the multimedia resources online. However, most search results for multimedia resources still focus on the whole resource level and media fragments are not fully indexed yet. Multimedia resources are seldom displayed alone on the Web page. Usually, a multimedia player is embedded in the replay page (also called "landing page") together with all media fragments, metadata and annotations. Search engines thus will not be able to distinguish which annotation is related to which media fragment and it leads to the difficulty of searching and reusing the inside content of multimedia.

The W3C Media Fragment Working Group has proposed Media Fragment URI 1.0 (MFURI) [6], which defines hash URI syntax to address media fragments from temporal, spatial, track and named section dimensions. It has been stated by Troncy et al. [6] that "*enabling the addressing of media fragments ultimately creates a means to attach annotations to media fragments*". In this proposal, we introduce a model to enable Google to index media fragments using related annotations, so that they could be found by traditional keyword search. This model uses MFURI syntax and Google's crawling infrastructure for Ajax applications. A demo is implemented on top of the Synote system [3]. Synote is a Web 2.0 application, which allows users to embed audio-visual resources from other domains and make synchronised annotations. We mainly implement the temporal dimension for audio-visual resources, but the concept could be easily extended to spatial and other dimensions defined in MFURI. Some initial evaluation results have shown that the annotations related to media fragments are indexed by Google and users could play the media fragment directly from the links provided by Google search results.

2. PROBLEM ANALYSIS

The online presence of media fragments is very poor at the moment. This is partially because a large portion of applications do not provide media fragments at all. The tags, descriptions and other forms of annotations are on the whole multimedia level. Another important reason is that even though some applications provide synchronised annotations, they are loaded together with the whole multimedia resource on the same logical page. This is reasonable as it provides an interactive experience for users. For example, TED Talks [1] and YouTube's interactive transcript[2] allow users to click on the transcript block and the media player embedded on the page will start playing from that time point.

This user-friendly function is not search-engine-friendly for two reasons. Firstly, most search engines only fetch pages as a direct response from the server. So any dynamically generated content on the client-side is ignored. YouTube load interactive transcript by Ajax, so the transcript will not be indexed by Google from the replay page. TED does not have this problem because the transcript is generated

WWW2012 Developer Track, April 18-20, 2012, Lyon, France.

[1] http://www.ted.com/talks
[2] http://goo.gl/t1nMj

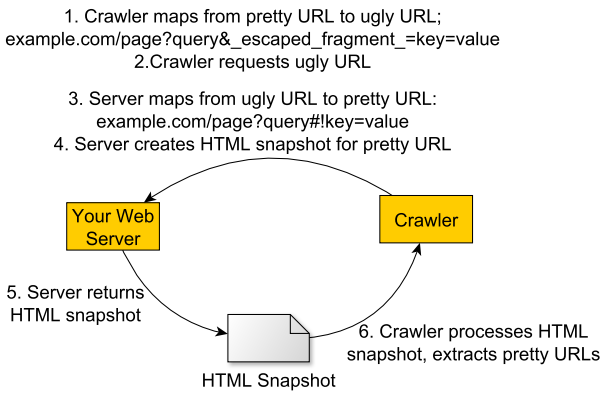


Figure 1: Google Ajax Application Crawler

by server-side script. Secondly, as media fragments and annotations share the same HTML page with the whole multimedia resource and annotations, search engines cannot find a unique logical page for each media fragment. When a keyword is searched, the search engine can only return the whole page and users will lose track of which media fragment is related to the keyword. This also causes accessibility problems for devices with low bandwidth, because much traffic is wasted on downloading unnecessary data.

One solution to these two problems is slicing the whole page into different pages according to media fragments, but the interactive experience will be lost in that users cannot watch different media fragments on the same page. The acceptable solution for both multimedia applications and end users must satisfy the following criteria:

- Keyword search should only return the documents directly related to the media fragments, i.e. all the content in the document should annotate this media fragment. It is better if the media fragment could be highlighted in the search results.
- The interactive experience should be kept. But when users click the link in the search result, the media fragment corresponding to the link should be highlighted
- Few changes to the server are required

3. IMPLEMENTATION

Google has developed a framework to crawl Ajax applications (Figure 1). If the "hashbang" token ("#!") is included in the original URL[3], Google crawler will know that this page contains Ajax content. Then the crawler will request "ugly URL". On receiving this "Ugly URL" request, the server can return the snapshot page after the dynamic information is fully generated by javascript. The content in the snapshot page will be indexed in for the original "pretty URL".

In this framework, the crawler does not care how the server generates the snapshot page, so we make use of this framework by keeping two sets of pages. We do not discard the existing replay pages, but we duplicate this page by cutting it into many snapshot pages based on the time span defined in "_escaped_fragment_" parameter in the "ugly URL". Each snapshot page contains metadata and annotations only related to the media fragments. The hash in MFURI syntax, however, is replaced by hashbang as "#!" is the required token by the crawler. When the request URL containing "_escaped_fragment_", the server returns the snapshot page. If not, the normal replay page will be returned.

Figure 2 explains how this model could be used to index media fragments. As a pre-condition, the time information must be available for the developers so that the MFURI can be constructed. The returned page in step 4 only contains keywords related to fragment "t=3,7". In the Google index, the "pretty media fragment URLs" are associated with the snapshot page. So what Google actually indexed is the URL of the replay page with hashbang and MFURI syntax attached. Step 8 still returns the whole page, but in step 9, the fragment will be passed to the URL representing the real location or the service which delivers the multimedia file. For example, if the request URL is *example/replay/1#!t=3,7*, the fragment "#t=3,7" will be attached at the back of *example2/1.ogv*, which is the video embedded in the replay page. Hashbang is not a valid syntax in MFURI specification, so developers need to parse the information in the hashbang URL before attaching the fragment to the the URL of the actual multimedia file. Then we control the embedded player to play the fragment from 3s to 7s using javascript and the corresponding annotations are highlighted straight away. In this case, step 6 will return the URL of the media fragment instead of the replay page. This design not only makes sure media fragments are indexed precisely with the keywords related to it, but also preserves the existing user interface and the interactive experience.

On the server side, necessary changes needs to be made. The first one is the programme to detect "_escaped_fragment_" parameter in the request URL from Google and redirect it to the snapshot generation programme, which is the second programme we need to add to the server. The snapshot page does not need to be user-friendly, but the metadata and annotations related to the media fragment should be presented in a well-structured manner. We also embed Microdata [2] defined in schema.org[4], into the page. Each media fragment is defined as either "Audio" or "Video", and the annotations are defined as "keywords" and other properties in schema.org.

The third component developers need to write is the programme to highlight the corresponding media fragment when the page is opened. For this demo, we developed Synote Media Fragment Player[5] (SMFP) to fulfil the visual output defined in Media Fragment User Agent (UA) Test Cases[6] on the temporal dimension of MFURI. MFURI should be processed via HTTP protocol with the help of "smart user agents", "smart servers" and proxy caches [4]. Even though some UAs support part of the functions[7] defined in MFURI, the general support for media fragment highlighting is very limited. So we developed SMFP to play the temporal di-

[2] http://goo.gl/dPc81

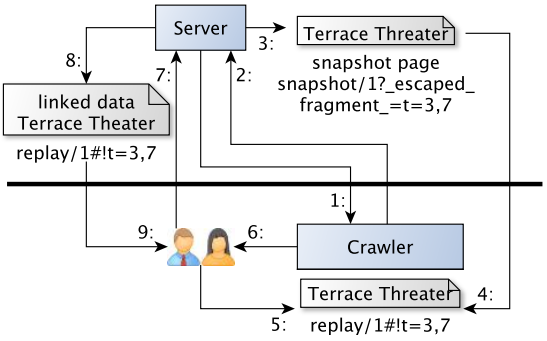
[3] The HTTP protocol and domain names will be ignored in the URLs for short in this paper

[4] http://schema.org

[5] http://goo.gl/NO0t7

[6] http://goo.gl/yexpy

[7] https://bugzilla.mozilla.org/show_bug.cgi?id=648595



1: Submit pretty URL "replay/1#!t=3,7" to the crawler
2: Crawler asks the server for "replay/1?_escaped_fragment_=t=3,7"
3: Redirect the request to "snapshot/1?_escaped_fragment_=t=3,7" and the server generates the snapshot page containing annotations and Microdata for "#t=3,7" only. For example, the page only contains keyword "Terrace Threater"
4: The snapshot page is returned by the crawler
5: A user searches "Terrace Threater"
6: Google includes "replay/1#!t=3,7" in the search results
7: The user clicks the link and asks the server for "replay/1#!t=3,7"
8: The server returns the replay page containing all the annotations i.e. both keywords "Terrace Theater" and "linked data"
9: The replay page highlights the media fragment and "Terrace Theater" by playing the fragment from 3s to 7s

Figure 2: The model to improve media fragment presence based on Google crawler

mension of media fragments for different formats of video and audio resources. In the future, if major browsers could natively support the media fragment highlighting and retrieval defined in MFURI specification, it would not be necessary for developers to implement this component. Except for these changes, we also need to include the newly created hashbang URLs in the sitemaps so that they could be easily found out by Google.

4. EVALUATION AND DISCUSSION

We implemented the model demonstrated in Figure 2 in Synote and pre-defined some media fragments and annotations. Sitemaps containing URIs like *replay/1#!t=3,7* have been submitted to Google for indexing. To have a quick evaluation of what Googlebot fetches from these URIs, we submit several URIs into Google Web Master Tools[8]. The result shows that on fetching this URL:

```
http://linkeddata.synote.org/synote/recording/replay/
36513#!t=00:00:01.000,00:00:14.000
```

The snapshot page is returned with annotations only related to fragment "#t=00:00:01.000,00:00:14.000". We also put some Microdata in the snapshot page and the semantic information can be successfully recognised by Live Microdata[9] and Linter Structured Data[10]. If the keyword "Terrace Theater", for example, is searched in Google, the snapshot page can be successfully found in the search results instead of the whole replay page. When we click the link in search results, the Synote Player page in Figure 3 will be opened and the button in top-left corner indicates that a media fragment is requested. On opening the page, the video will start playing from 1s to 14s and the related annotation on the right column will be highlighted.

As another example for evaluation, we search the sentence "*All kinds of conceptual things, they have names now that start with HTTP*" in Google. This sentence is included in the transcript of the video resources for both TED Talks and Synote. The first result in Figure 4 is from TED Talks. Clicking on the link will open the replay page of the talk, but you still need to manually find the sentence in the interactive transcript. So even though the sentence logically

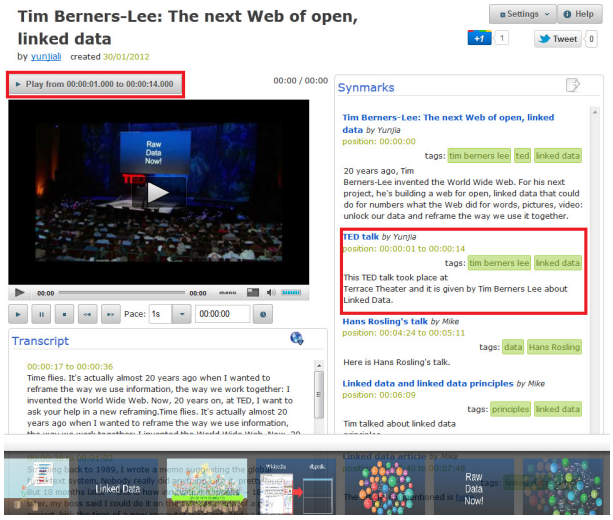


Figure 3: Screenshot of Synote replay page

annotates part of the video, the search engine still associates it with the whole document, i.e. the replay page. The second result in Figure 4 indicates that this sentence in Synote is directly related to the fragment defined in the replay page. On clicking this search result, the video embedded in the replay page will start playing from the start time of this media fragment.

The main idea of this model is generating a set of search-engine-friendly snapshot pages on the fly according to the time span defined in the fragment of request URL. For applications heavily relied on Ajax or Flash, it is useful to keep two sets of pages for both rich user interaction and search engine optimisation (SEO). Considering Web Content Accessibility Guidelines[11], it is also a good practice to provide such snapshot pages because the interactive feature usually is not accessible to screen readers and keyboard users since it depends much on javascript.

This model relies on hashbang URLs, which has been widely used in Facebook and Twitter to allow the indexing of Ajax content. But hashbang is claimed not to be a

[8] http://www.google.com/webmasters/tools/
[9] http://foolip.org/microdatajs/live/
[10] http://linter.structured-data.org/
[11] http://www.w3.org/TR/WCAG20/

Figure 4: Search results comparison between TED Talks and Synote

good practice of URL design[12] and it should be replaced by HTML5 PushState[13]. Twitter recently also announced that they would remove all the hashbang URLs[14]. In order to use Google Ajax Application Crawler, we have to implement the hashbang URLs, but we also need to consider the disadvantages of such URLs in the future work.

Another limitation of this model is that it only works for Google and therefore further work is required to investigate similar solutions for other search engines, such as Yahoo! and Bing. The general solution of media fragment indexing might be embedding Rich Snippet [5], such as Microdata and RDFa [1], into the page so that search engines can highlight them in the search results. But currently, major search engines are still far from reaching an agreement on the vocabularies to describe media fragments.

5. CONCLUSION

Media fragment is important on the Web as it describes the inside content of multimedia resource. However, major search engines currently are not able to index media fragments using the related annotations. We analysed this problem from both multimedia application and search engine's point of views. The key problem is that media fragments and annotations do not have their own page on the Web and they share the same page as the parent resource. We developed a model, which uses MFURI syntax and Google Ajax Application Crawler, to let Google index snapshot pages for each media fragment. In this way, media fragments will have their own links in the search results and on following the links, users still visit the same replay pages as before. We have evaluated the implementation and the result shows that the media fragments could be indexed by Google. We envision that by using this model, more and more media fragments and annotations will be indexed and the search for multimedia resources will be more efficient. The screencast[15] of this evaluation and the live demo of Synote[16] are available online.

6. REFERENCES

[1] B. Adida and M. Birbeck. RDFa Primer, Oct. 2008. http://www.w3.org/TR/xhtml-rdfa-primer/.

[2] I. Hickson. HTML Microdata, Feb. 2012. http://dev.w3.org/html5/md/.

[3] Y. Li, M. Wald, G. Wills, S. Khoja, D. Millard, J. Kajaba, P. Singh, and L. Gilbert. Synote: development of a web-based tool for synchronized annotations. *New Review of Hypermedia and Multimedia*, 17(3):295–312, 2011.

[4] E. Mannens, D. Van Deursen, R. Troncy, S. Pfeiffer, C. Parker, Y. Lafon, J. Jansen, M. Hausenblas, and R. Van de Walle. A uri-based approach for addressing fragments of media resources on the web. *Multimedia Tools and Applications*, pages 1–25. 10.1007/s11042-010-0683-z.

[5] T. Steiner, R. Troncy, and M. Hausenblas. How Google is using Linked Data Today and Vision For Tomorrow. In S. Auer, S. Decker, and M. Hauswirth, editors, *Linked Data in the Future Internet 2010*, Ghent,Belgium, 2010.

[6] R. Troncy, E. Mannens, S. Pfeiffer, and D. V. Deursen. Media fragments URI 1.0 (basic), Mar. 2012. http://www.w3.org/TR/media-frags/.

[12] http://goo.gl/xfvwT
[13] http://diveintohtml5.info/history.html
[14] http://storify.com/timhaines/hashbang-conversation
[15] http://goo.gl/4zl1V
[16] http://linkeddata.synote.org/synote/