Tracking Provenance In An Entity-Consolidating RESTful Read/Write Web Service For RDF Video Annotations

Thomas Steiner
Universitat Politécnica de Catalunya
Department LSI
08034 Barcelona, Spain
tsteiner@lsi.upc.edu

Your Name
Affiliation
Address
Address
you@example.org

Joaquim Gabarró
Universitat Politécnica de Catalunya
Department LSI
08034 Barcelona, Spain
gabarro@lsi.upc.edu

ABSTRACT

Using Natural Language Processing or URI Lookup third party Web services, converting legacy unstructured data into Linked Data is a relatively straight-forward task. In this paper we first present an approach to first consolidate entities found by such Web services when being used in parallel, and then describe how one can keep track of provenance at the same time. We have implemented a RESTful Web service for the automatic RDF annotation of YouTube videos, and discuss how in a read/write environment manual changes to automatically generated RDF annotations can be tracked.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: On-line Information Services

General Terms

Experimentation

Keywords

RDF, LOD, Linked Data, Semantic Web, NLP, Video

1. INTRODUCTION

With SemWebVid [9] we introduced a client-side interactive Ajax application for the automatic generation of RDF video annotations. For this paper we have re-implemented and vastly improved the annotation logic on the server-side, resulting in a RESTful read/write-enabled Web service for RDF video annotations. A YouTube video is described by a Google Data Atom feed[1]. In order to semantically annotate the various elements of this feed, we concentrated on the following fields (in XPath syntax): title `/entry/media:group/media:title`, description `/entry/media:group/media:description`, tags `/entry/media:group/media:keywords`. YouTube offers an automatic audio transcription service and users can also upload audio transcriptions on their own. This allows for closed captions in several languages (we differentiate between subtitles and closed captions, where subtitles are hard-encoded into the video, and closed captions separate resources). In addition to the previously mentioned elements of the Google Data Atom feed, we thus use closed captions[2] when they are available.

The remainder of this paper is structured as follows: section 2 introduces Web services that allow unstructured data to be converted into Linked Data, section 3 shows our approach to entity consolidation for URI Lookup and NLP Web services, section 5 contains a description of how we maintain provenance metadata in our Web service, section 6 discusses related and future work, and finally section 7 contains a conclusion.

2. WEB SERVICES FOR CONVERTING UNSTRUCTURED DATA INTO LINKED DATA

We differentiate between Natural Language Processing (NLP) Web services and URI Lookup Web services. The NLP Web services that we use for our experiemtns take a text fragment as an input and perform Named Entity Extraction (NER) on it and link extracted entities back into the Linked Open Data cloud[3]. For NLP Web services, we use OpenCalais, Zemanta, and AlchemyAPI[4]. The URI Lookup Web services take a term as an input, and return the set of URIs that most probably represent this term. For URI Lookup Web services, we use Freebase, DBpedia Lookup, Sindice, and Uberblic[5]. For both families of services, we use these services in parallel, aiming for the emergence effect in the sense of Aristotle[6]. In the next section we describe our strategies for entity consolidation.

[1]E.g., `http://gdata.youtube.com/feeds/api/videos/Rq1dow1vTHY`

[2]E.g., `http://www.youtube.com/watch_ajax?action_get_caption_track_all&v=Rq1dow1vTHY`

[3]`http://lod-cloud.net/`

[4]`http://www.opencalais.com/documentation/calais-web-service-api`, `http://developer.zemanta.com/docs/`, `http://www.alchemyapi.com/api/entity/`

[5]`http://wiki.freebase.com/wiki/Search`, `http://lookup.dbpedia.org`, `http://sindice.com/developers/api#SindicePublicAPI-TermSearch`, `http://uberblic.org/developers/apis/search/`

[6]Aristotle, Metaphysics, Book H 1045a 8-10: "[...] the totality is not, as it were, a mere heap, but the whole is something besides the parts [...]"

3. ENTITY CONSOLIDATION

First, we present our approach how to consolidate entities from URI Lookup Web services.

3.1 Entitiy Consolidation For URI Lookup Web Services

As as first step we have implemented a wrapper for all four URI Lookup services that aligns the particular service's output to a common output format. This format is the least common multiple of the information of all feeds. For our experiments we agreed on the JSON format below (the examples below use the term `Google Translate` to illustrate the approach):

```
[
  {
    "name": "Google Translate",
    "uris": [
      "http://dbpedia.org/resource/Google_Translate"
    ],
    "provenance": "freebase,uberblic,dbpedia",
    "relevance": 0.75
  }
]
```

The corresponding call to our wrapper API that calls all four Web services in the background is at `GET http://localhost:3000/uri-lookup/combined/Google/%20Translate` (the particular results from each service are available at `http://localhost:3000/uri-lookup/{service_name}/Google%20Translate`). As can be seen in the example above, already at the lowest data representation level we maintain provenance metadata (Sindice delivered a different result, and is thus not in the list). In order to agree on a winner entity, a majority-based voting system is used. The problem, however, is that both Freebase and Uberblic return results in their own namespaces (for `Google Translate` the results are `http://freebase.com/en/google_translate`, `http://uberblic.org/resource/67dc7037-6ae9-406c-86ce-997b905badc8#thing`), whereas Sindice and DBpedia Lookup return results from DBpedia (obvious for DBpedia Lookup, and among other results for Sindice). Freebase and Uberblic interlink their results with DBpedia at an `owl:sameAs` level for Freebase, and by referencing the source (`umeta:source_uri`) for Uberblic, so by retrieving the referenced resources in the services' namespaces we can map back to DBpedia URIs and match all four services' results at this level. Each service's result contributes with a relevance of 0.25 to the final result, in the above example when three services agree on the same result, the resulting relevance is thus the sum of the singular relevance scores (0.75 in this case).

3.2 Entitiy Consolidation For NLP Web Services

In analogy to our approach to URI Lookup entity consolidation, we have implemented a wrapper API for the three NLP services. While the original calls to the particular NLP service are all HTTP `POST` based, we have implemented the wrapper `GET` based. The least common multiple of the particular results looks like this (shortened to one entity for the sake of legibility, the original result contained 7 entities, 6 directly relevant, and 1 related, but not directly relevant entity):

```
[
  {
    "name": "Google Translate",
    "relevance": 0.7128319999999999,
    "uris": [
      {
        "uri": "http://dbpedia.org/resource/Google_Translate",
        "provenance": "alchemyapi"
      },
      {
        "uri": "http://rdf.freebase.com/ns/en/google_translate",
        "provenance": "zemanta"
      }
    ],
    "provenance": "alchemyapi,zemanta"
  }
]
```

These results came from a call to our wrapper API at `GET http://localhost:3000/entity-extraction/combined/Google%20Translate`, and as with URI Lookup the particular services' can be obtained at `http://localhost:3000/entity-extraction/{service_name}/Google%20Translate`. While AlchemyAPI and Zemanta return results from DBpedia and other interlinked LOD cloud resources, OpenCalais returns only results in its own namespace (e.g., `http://d.opencalais.com/er/company/ralg-tr1r/ce181d44-1915-3387-83da-0dc4ec01c6da.rdf` for the company Google). While in that particular case retrieving the resource RDF representation and checking for `owl:sameAs` links to DBpedia is successful, in general we found OpenCalais URIs sometimes point to non-existant resources, or to not very rich resources like `http://d.opencalais.com/pershash-1/cfcf1aa2-de05-3939-a7d5-10c9c7b3e87b.html` for President Barack Obama, where the only information is that Barack Obama is of type person. In order to consolidate extracted entities, we use the following approach: we have a look at each of the extracted entities from service one and compare each entity's URIs with each URIs from each extracted entity from service two. To illustrate this, see the example below (shortened for the sake of legibility, the used text fragment contained a reference to the company Google). Results for the text fragment from AlchemyAPI:

```
{
  "name": "google",
  "relevance": 0.496061,
  "uris": [
    {
      "uri": "http://dbpedia.org/resource/Google",
      "provenance": "alchemyapi"
    },
    {
      "uri": "http://rdf.freebase.com/ns/guid.9202a8c04000641f8
      "provenance": "alchemyapi"
    },
    {
      "uri": "http://cb.semsol.org/company/google.rdf",
      "provenance": "alchemyapi"
```

```
    }
  ],
  "provenance": "alchemyapi"
}
```

Results for the text fragment from Zemanta:


```
{
  "name": "google inc.",
  "relevance": 0.563132,
  "uris": [
    {
      "uri": "http://rdf.freebase.com/ns/en/google",
      "provenance": "zemanta"
    },
    {
      "uri": "http://dbpedia.org/resource/Google",
      "provenance": "zemanta"
    },
    {
      "uri": "http://cb.semsol.org/company/google#self",
      "provenance": "zemanta"
    }
  ],
  "provenance": "zemanta"
}
```


As can be seen the entity names mismatch (`google inc.` vs. `google`), however, going down the list of URIs for the entity, one can note a match via `http://dbpedia.org/resource/Google`. In addition to that one can also see two would-be matches (`http://cb.semsol.org/company/google.rdf` vs. `http://cb.semsol.org/company/google#self` and `http://rdf.freebase.com/ns/en/google` vs. `http://rdf.freebase.com/ns/guid.9202a8c04000641f800000000042acea`), however, because of the inconsistent use of URIs when there are more than one URI available for the same entity hinders the match from being made. An additional retrieval of the resources would be necessary to detect that in the latter case `http://rdf.freebase.com/ns/guid.9202a8c04000641f800000000042acea` redirects to `http://rdf.freebase.com/ns/en/google`, whereas the first example seems to be broken. The good thing, however, is that as soon as one match has been detected, one can consolidate the entities from both services. Note how the entity names mismatch (`google inc.` vs. `google`). The consolidated name is an array of all detected synonymous names. The consolidated relevance is then the average relevance of both services. In contrast to URI Lookup where we had to manually assign a relevance of 0.25 to each result because not all URI Lookup services included the concept of relevance in their results, with NLP services each service includes this concept, so we can directly use it. In our code the consolidated entities from service 1 and 2 are then in turn compared to extracted entities from service 3 and so on, in practice, however, due to the not always given interconnectedness of OpenCalais, there are no matches after having compared Zemanta-extracted entities with AlchemyAPI-extrated entities. As above with URI Lookup-detected entity consolidation, also with NLP-detected entity consolidation we maintain provenance metadata for each URI on the lowest data representation level.

It is to be noted that the results from URI Lookup are a subset of NLP in our case, however, while all URI Lookup services accept one-word arguments (e.g., `google`), only AlchemyAPI accepts one-word arguments, the two other services accept only non-trivial text fragments (e.g., `google is a company founded by larry page` works).

3.3 Design Of the Web Service

Currently our Web service supports the following operations:

- Looking up URIs for a given term (allowed service names are "dbpedia", "freebase", "uberblic", "sindice", and "combined"): `GET /uri-lookup/{service_name}/{term}`
- Extracting entities from a given text fragment (allowed service names are "opencalais", "zemanta", "alchemyapi", and "combined"): `GET /entity-extraction/{service_name}/{text_fragment}`
- Get an RDF annotation for a video with a given video ID: `GET /youtube/rdf/{video_id}`
- Update an RDF annotation for a video with a given video ID: `PUT /youtube/rdf/{video_id}`
- Delete an RDF annotation for a video with a given video ID: `DELETE /youtube/rdf/{video_id}`
- Get metadata from YouTube for a video with a given video ID: `GET /youtube/video/{video_id}`
- Get all closed captions from YouTube for a video with a given video ID: `GET /youtube/video/{video_id}/closedcaptions`
- Get closed captions in a given language from YouTube for a video with a given video ID: `GET /youtube/video/{video_id}/closedcaptions/{language_code}`
- Get a plaintext audio transcript from YouTube for a video with a given video ID: `GET /youtube/video/{video_id}/audiotranscription`
- Get a plaintext audio transcript in a given language from YouTube for a video with a given video ID: `GET /youtube/video/{video_id}/audiotranscription/{language_code}`

4. REPRESENTING VIDEOS IN RDF

In the following we present the particular components of the available video metadata and their representation in RDF.

4.1 Basic YouTube Metadata

We decided to use the W3C Ontology for Media Resources[11] as the central vocabulary, mainly because it already has a defined mapping not only for YouTube data, but also for many other existing metadata formats. "The ontology is supposed to foster the interoperability among various kinds of metadata formats currently used to describe media resources on the Web" (sic from the introduction of [11]). From the vocabulary we use the following fields: `ma:title`, `ma:creator`, `ma:createDate`, and `ma:description`, which, as outlined before, have direct mappings to YouTube data.

4.2 YouTube Tags

In order to represent YouTube tags, or rather, semantically annotated YouTube tags, we use the Common Tag[1] vocabulary. A resource is `ctag:tagged` with a `ctag:Tag`, which consists of a textual `ctag:label` pointing to a resource that specifies what the label `ctag:means`.

4.3 Entities In Video Fragments

The current video annotation Web service is a re-implementation of our previous client-side application SemWebVid[9]. Hence we had some experience with modeling video data in RDF. In our first attempt we used the Event Ontology[8] and defined each line in the closed captions track as an `event:Event`. In the current implementation we simplified the annotation model by removing the notion of events, and by introducing the notion of video fragment instead. A video fragment is now defined by a complete sentence in the closed captions track, which much more matches the human perception of a self-contained incident in a video. In consequence a video fragment is a part of the whole video. In order to address a video fragment, we decided to use Media Fragment URIs[10]. Media Fragment URIs are also supported by the Ontology for Media Annotation in form of `ma:fragment`. In particular we use the temporal dimension (e.g., `http://example.org/video.webm#t=10,20`), which is defined by its start time and its end time relative to the entire video play time. In addition to the temporal dimension, we also use the track dimension (e.g., `http://example.com/video.webm#track=closedcaptions`), which allows for addressing only a closed captions track (with timing information), or even the plaintext audio transcript (e.g., `#track=audiotranscript`). The value of the parameter `track` is a free-form string. In the previous SemWebVid implementation we used `event:factor`, `event:product`, and `event:agent` to relate events with factors (extracted entities), products (a plaintext closed caption line), and agents (extracted persons). Now in order to annotate entities in a temporal video fragment, we consistently use the same concept of Common Tag as outlined in section 4.2. We thus have (in Turtle syntax, left out prefixes for the sake of brefity):

```
<http://example.org/video.webm#t=10,20> a ma:fragment ;
  ctag:tagged :tag .
:tag a ctagTag ;
  ctag:label "example" ;
  ctag:means <http://example.org/example#> .
```

5. TRACKING PROVENANCE WITH MULTIPLE DATA SOURCES

As outlined before we use several data sources (Web services) in the background in order to deploy our own video annotation Web service. The simple example fact produced by our service that an `ma:fragment` is `ctag:tagged` with a `ctagTag` with the `ctag:label` in plaintext form `example`, which `ctag:means` an example entity represented by the URI `http://example.org/example#` might in consequence have been the result of up to in the concrete case seven agreeing or disagreeing Web services. In order to track the contributions of the various sources, we decided to use the Provenance Vocabulary[6] by Hartig and Zhao. Even if the direct requests of our Web service were made against our wrappers (as outlined in sections 3.1 and 3.2), we still want to credit back the results to the original calls to the third party Web services. We have two basic cases that affect the RDF that describes the data provenance, requests per HTTP `GET` and requests per HTTP `POST`. All URI Lookup services that we use are `GET`-based, all of our NLP services are `POST`-based. In order to make statements about a bundle of triples, we can put them in a named graph. We use the TriG[2] syntax. The example from above then looks like this:

```
:G = {
  <http://example.org/video.webm#t=10,20> a ma:fragment ;
    ctag:tagged :tag .
  :tag a ctagTag ;
    ctag:label "example" ;
    ctag:means <http://example.org/example#> .
} .
```

5.1 The Provenance Vocabulary

In the following we outline the required steps in order to make statements about the provenance of a bundle of triples contained in a named graph `:G` that were generated using several HTTP `GET` requests to third party Web services. First, we state that `:G` is both a `prv:DataItem` and obviously an `rdfg:Graph`. `:G` is `prv:createdBy` the process of a `prv:DataCreation`. This `prv:DataCreation` is `prv:performedBy` a `prv:NonHumanActor`, a `prvTypes:DataCreatingServic` to be precise. This service is `prv:operatedBy` us (`http://tomayac.com/thomas_steiner.rdf#me`. Time is often important for provenance, so the `prv:performedAt` date of the `prv:DataCreation` needs to be saved. During the process of the `prv:DataCreation` there are `prv:usedData`, which are `prv:retrievedBy` a `prv:DataAcess` that is `prv:performedAt` a certain time, and `prv:performedBy` an actor and `prv:operatedBy` us (`http://tomayac.com/thomas_steiner.rdf#me`. For the `prv:DataAccess` we `prv:accessedService` from a `prv:DataProvidingServ` of which we `prv:accessedResource` at a certain `irw:WebResource`. Therefore we `prvTypes:exchangedHTTPMessage` which is an `http:Request` using `http:httpVersion` "1.1" and the `http:methodName` "GET". See Annex A for an overview of the necessary provenance RDF.

5.2 Tracking Provenance With Human Interaction

Oftentimes completely automatically generated RDF video annotation files will need a little manual fine-tuning. In our RESTful Web service we have thus envisiond that not only a big archive of automatically generated video annotations gets built, but that also people can correct errors in the RDF interactively (or remove completely wrong video annotations). For the correction case this can be tracked using the Provenance Vocabulary as follows (let's assume we wanted to replace an unfriendly Freebase `ctag:means` value of `http://rdf.freebase.com/ns/guid.9202a8c04000641f800000000042acea` with the friendlier variant `http://rdf.freebase.com/rdf/en.google`):

```
:G_corrected = {
  <http://gdata.youtube.com/feeds/api/videos/3PuHGKnboNY> ctag:
  :tag_corrected
    a ctag:Tag ;
    ctag:label "google" ;
```

```
    ctag:means <http://rdf.freebase.com/rdf/en.google> ;
} .
:G_corrected
  a prv:DataItem ;
  a rdfg:Graph ;
  prv:createdBy [
    a prv:DataCreation ;
    prv:performedAt "2011-02-07T12:42:30Z"^^xsd:dateTime ;
    prv:performedBy [
      a prv:HumanActor ;
      <http://tomayac.com/thomas_steiner.rdf#me> ;
    ] ;
  ] .
```

Note how the `prv:DataCreation` no longer contains references to `prv:usedData`. Obviously the shown approach to identify a `prv:HumanActor` with her FOAF profile requires an authentication step, which might not always be wanted. One could think of a "John Doe"[7]-like anonymous `prv:HumanActor`, or in the case of a non-anonymous `prv:HumanActor` authentication mechanisms like WebID[8] could be used.

5.3 The Need For Providing Provenance Metadata

Hartig et al. mention in [5] some reasons that justify the need for provenance metadata, among those linked dataset replication and distribution on the Web with not necessarily always the same namespaces. Based on the same source data, different copies of a linked dataset can be created with different degrees of interconnectedness by different publishers. We add to this list the automatic conversion of legacy unstructured data to Linked Data with heuristics, where, as in our case, extracted entities while still being consolidated and backed up by different data sources, might still be wrong. Especially with our "mash-up"-like approach it is very desirable to be able to track back to the concrete source where a certain piece of information might have come from.

6. RELATED AND FUTURE WORK

Related work includes Popcorn.js from the Mozilla Drumbeat project[4] that with its interactive Butter editor allows for a video to be semantically annotated (completely manually). To a given video annotation, the Popcorn.js script then pulls in multiple data feeds from the APIs of Google News, Wikipedia, Twitter, and Flickr in order to semantically enrich the video. It also provides automatic machine translation from Google Translate, and attribution data from Creative Commons. The focus, however, seems to be on the final video "mash-up", not on the actual annotation. Future work could be to offer an RDF-to-Popcorn.js wrapper service that would allow to profit from the project's great visual experience.

In [13], Waitelonis et al. address the problem of how to deploy exploratory search for video data by using semantic search technology for the yovisto[9] video search engine. They show how exploratory search can be enriched by information from the LOD cloud in order to facilitate navigation in big video archives. Yovisto supports several ways to annotate a video with metadata: video-related, and video-time-related tags. Video-related tags are valid for the entire video and are entered by the initial video uploader. Video-time-related metadata can either be automatically extracted from the video on a certain timestamp (e.g., by analyzing the video images with OCR methods), or can be user-generated tags also on a certain timestamp in the video. In [12] Waitelonis et al. show how using term and term context permutations and detecting paths between entities a legacy keyword-based video search engine can be converted to a semantic video search engine. The approach uses a keyword-to-DBpedia-URI mapping heuristic, however, as far as we can tell provenance metadata is not maintained. Future work will compare the results of the yovisto heuristic with ours using agreed-on benchmarks.

In [3] Choudhury et al. describe a framework for semantic enrichment, ranking, and integration of Web video tags using Semantic Web technologies. In order to enrich the oftentimes sparse user-generated tag space, meta data like the recording time and location, or the video title and video description are used, but also social features such as playlists where a video appears in and related videos. Next, the tags are ranked by their co-occurrence and in a final step interlinked to DBpedia concepts for greater integration with other datasets. Choudhury et al. disambiguate the tags based on WordNet[7] synsets if possible (i.e., if there is only one matching synset in WordNet, the corresponding WordNet URI in DBpedia is selected. If there are more than one matching synsets, the tags and their context tags similarity is computed and thereby tried to decide on an already existing tag URI). For words that are not contained in WordNet, Sindice is used to find the most probable concept. To the best of our knowledge provenance metadata is not maintained.

7. CONCLUSION

We have introduced the domain of research of semantic text-based video annotation for YouTube videos with closed captions. Therefore we presented several URI Lookup and NLP Web services and showed our approach for both classes of Web services to consolidate entities. We then focused on the RDF vocabularies and Media Fragment URIs to annotate video-related and video-time-related entities. Due to their different "mash-up"-like history of origins, we need to track provenance metadata in order to assure the trustworthiness of the generated data. We showed how the Provenance Vocabulary can be used to keep track of even the original third party Web service calls, while it is to be noted that these are to be understood as the identificator of Web resources (i.e., the search result). Finally we drew a bow to related work, and presented directions for future work.

In the paper we have shown how a multi-source Web service can automatically maintain provenance metadata, both for entirely Web service-generated content, but also for partly (or completely) human-generated content. We believe that the origin of a triple is of immense importance, especially given the network effect which is one of the Linked Data pro arguments.

8. ACKNOWLEDGMENTS

[7] http://en.wikipedia.org/wiki/John_Doe
[8] http://www.w3.org/2005/Incubator/webid/charter
[9] http://yovisto.com/

We would like to thank Olaf Hartig from the Hmboldt-Universität zu Berlin for his kind support with the correct use of the Provenance Vocabulary. This work is partly funded by the EU FP7 I-SEARCH project (project reference 248296).

9. REFERENCES

[1] Common Tag Specification, retrieved February 7, 2011. http://commontag.org/Specification.

[2] C. Bizer and R. Cyganiak. The TriG Syntax, July 30, 2007. http://www4.wiwiss.fu-berlin.de/bizer/TriG/.

[3] S. Choudhury, J. Breslin, and A. Passant. Enrichment and Ranking of the YouTube Tag Space and Integration with the Linked Data Cloud. In *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, chapter 47, pages 747–762. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2009.

[4] B. Gaylor. Popcorn.js Semantic Video demo. Mozilla Drumbeat Project, August 16, 2010. http://www.drumbeat.org/content/popcorn-js-semantic-video-demo.

[5] O. Hartig and J. Zhao. Publishing and consuming provenance metadata on the web of linked data. In *Proceedings of The third International Provenance and Annotation Workshop*, Troy, NY, USA, 2010.

[6] O. Hartig and J. Zhao. Provenance Vocabulary Core Ontology Specification, January 25, 2011. http://purl.org/net/provenance/ns.

[7] Princeton. WordNet. http://wordnet.princeton.edu/.

[8] Y. Raimond and S. Abdallah. The Event Ontology, October 25, 2007. http://motools.sourceforge.net/event/event.html.

[9] T. Steiner. Semwebvid - making video a first class semantic web citizen and a first class web bourgeois. In *9th International Semantic Web Conference (ISWC2010)*, November 2010.

[10] W3C. Media Fragments URI. W3C Working Draft, December 8, 2010. http://www.w3.org/2008/WebVideo/Fragments/WD-media-fragments-spec/.

[11] W3C. Ontology for Media Resource. W3C Working Draft, June 8, 2010. http://www.w3.org/TR/2010/WD-mediaont-10-20100608/.

[12] J. Waitelonis, N. Ludwig, and H. Sack. Use what you have: Yovisto video search engine takes a semantic turn. In *5th International Conference on Semantic and Digital Media Technologies*, Saarbrücken, Germany, 2010.

[13] J. Waitelonis, H. Sack, J. Hercher, and Z. Kramer. Semantically enabled exploratory video search. In *Proceedings of the 3rd International Semantic Search Workshop*, SEMSEARCH '10, pages 8:1–8:8, New York, NY, USA, 2010. ACM.

APPENDIX

A. PROVENANCE RDF OVERVIEW

Shortened overview of the provenance RDF in Turtle syntax for a YouTube tag with the label `obama` and the assigned meaning `http://dbpedia.org/resource/Barack_Obama` (only two of the `prv:usedData` sources are mentioned):

```
:G = {
  <http://gdata.youtube.com/feeds/api/videos/3PuHGKnboNY> ctag:
  :tag
    a ctag:Tag ;
    ctag:label "obama" ;
    ctag:means <http://dbpedia.org/resource/Barack_Obama> ;
} .
:G
  a prv:DataItem ;
  a rdfg:Graph ;
  prv:createdBy [
    a prv:DataCreation ;
    prv:performedAt "2011-02-07T12:42:30Z"^^xsd:dateTime ;
    prv:performedBy [
      a prv:NonHumanActor ;
      a prvTypes:DataCreatingService ;
      prv:operatedBy <http://tomayac.com/thomas_steiner.rdf#me>
    ] ;
    prv:usedData [
      prv:retrievedBy [
        a prv:DataAcess ;
        prv:performedAt "2011-02-07T12:42:30Z"^^xsd:dateTime ;
        prv:performedBy [
          prv:operatedBy <http://tomayac.com/thomas_steiner.rdf
        ] ;
        prv:accessedService <http://api.freebase.com/api/servic
        prv:accessedResource <http://api.freebase.com/api/servi
        prvTypes:exchangedHTTPMessage [
          a http:Request ;
          http:httpVersion "1.1" ;
          http:methodName "GET" ;
          http:headers (
            [
              http:fieldName "Host" ;
              http:fieldValue "api.freebase.com" ;
              http:headerName <http://www.w3.org/2008/http-head
            ]
          )
        ] ;
      ] ;
    ] ;
    prv:usedData [
      prv:retrievedBy [
        a prv:DataAcess ;
        prv:performedAt "2011-02-07T12:42:30Z"^^xsd:dateTime ;
        prv:performedBy [
          prv:operatedBy <http://tomayac.com/thomas_steiner.rdf
        ] ;
        prv:accessedService <http://lookup.dbpedia.org/> ;
        prv:accessedResource <http://lookup.dbpedia.org/api/sea
        prvTypes:exchangedHTTPMessage [
          a http:Request ;
          http:httpVersion "1.1" ;
          http:methodName "GET" ;
          http:headers (
            [
              http:fieldName "Host" ;
              http:fieldValue "lookup.dbpedia.org" ;
              http:headerName <http://www.w3.org/2008/http-head
            ]
```

```
          )
        ] ;
      ] ;
    ] ;
  ] .
} .
```