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

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

Raphaël Troncy
EURECOM
Sophia Antipolis
France
raphael.troncy@eurecom.fr

Davy Van Deursen
Ghent University - IBBT
ELIS - Multimedia Lab
Ghent, Belgium
davy.vandeursen@ugent.be

Joaquim Gabarró
Univ. 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 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 on-the-fly text-based RDF annotation of YouTube videos that illustrates how provenance metadata can be automatically added to the Web service output, and discuss how in our read/write-enabled Web service 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 [**?**] we introduced a client-side interactive Ajax application for the automatic generation of RDF video annotations. For this paper we have re-implemented and 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 concentrate on the following fields (in XPath syntax): title (`/entry/media:group/media:title`), description (`/entry/media:group/media:description`), and tags (`/entry/media:group/media:keywords`). YouTube offers an automatic audio transcription service and video owners can also upload audio transcriptions and/or closed captions files on their own. We differentiate between subtitles and closed captions, where subtitles are hard-encoded into the video, and closed captions are separate resources. Audio transcriptions consist of a plaintext representation of speech, however, without timing information. In the case of audio transcriptions, YouTube automatically adds the timing information and tries to convert them to closed captions. YouTube offers support for closed captions in several languages. Hence, in addition to the previously mentioned elements of the Google Data Atom feed, we thus also use closed captions[2] when they are available.

The remainder of this paper is structured as follows: Sect. 2 introduces two classes of Web services that allow for unstructured data to be converted into Linked Data, Sect. 3 explains our approach to entity consolidation for URI Lookup and NLP Web services, Sect. 5 contains a description of how we automatically maintain provenance metadata in our Web service, Sect. 6 discusses related and future work, and finally Sect. 7 finalizes the paper with 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 experiments take a text fragment as an input, perform Named Entity Extraction (NER) on it and then link the extracted entities back into the Linked Open Data (LOD) cloud[3]. For NLP Web services we use

[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/`

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. We use Freebase, DBpedia Lookup, Sindice, and Uberblic[5]. For both classes of services we use all 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 in both cases.

3. ENTITY CONSOLIDATION

We define the process of entity consolidation as the merge of entities, i.e., if two services extract the same entity from the same input text fragment or term, we say that the entity is consolidated.

3.1 Interplay Of the Web Services

As outlined before the metadata for a YouTube video are its title, its description, its tags, and its closed captions tracks. We analyze the tags one-by-one with URI Lookup Web services through our wrapper Web service, and afterwards we analyze the title combined with the description in a first, and the closed captions track in a second separate run through our wrapper Web service to the NLP Web services.

In Sect. 3.2 we present our approach for how to consolidate entities from URI Lookup Web services, followed by Sect. 3.3 where we show our approach for NLP Web services.

3.2 Entity Consolidation For URI Lookup Web Services

As as first step we have implemented a wrapper for all four URI Lookup services that assimilates the particular service's output to a common output format. This format is the least common multiple of the information of all Web service results. 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 request to our wrapper API that calls all four URI Lookup Web services in the background is via `GET /uri-lookup/combined/Google%20Translate` (the particular results from each service are available at `/uri-lookup/{service_name}/Google%20Translate`). As can be seen in the example above, already at the lowest data representation level (JSON) we maintain provenance metadata (Sindice in the concrete case of the example delivers a different result, and is thus not in the provenance 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 (e.g., for "Google Translate" the results are `http://freebase.com/en/google_translate`, and `http://uberblic.org/resource/67dc7037-6ae9-406c-86ce-997b905badc8#thing`), whereas Sindice and DBpedia Lookup return results from DBpedia (obvious for DBpedia Lookup, and from DBpedia among also other results for Sindice). Freebase and Uberblic interlink their results with DBpedia at an `owl:sameAs` level in the case of Freebase, and by referencing the source (`umeta:source_uri`) for Uberblic. So by retrieving and parsing the referenced resources in the services' namespaces we can map back to DBpedia URIs and thus match all four services' results on the DBpedia 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).

3.3 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`- and `POST`-based. The least common multiple of the particular results can be seen below (shortened to one entity with just two URIs for the sake of legibility, the original result contained seven entities, thereof six directly relevant, and one 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 come from a request to our wrapper API via `GET /entity-extraction/combined/Google%20Translate`, and as with URI Lookup the particular services' results can be obtained at `/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

[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 [...]"

resource RDF representation and parsing 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 the current US 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 examples below (shortened for the sake of legibility, the used text fragment contains 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.9202
          a8c04000641f800000000042acea",
      "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`. Additionally, 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 is 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 (`http://cb.semsol.org/company/google#self` returns status code 404). The good thing, however, is that as soon as one match has been detected, one can consolidate the entities from both services.

Again note how the two entity names mismatch ("google inc." vs. "google"). The consolidated name is then an array of all detected synonymous names. The consolidated relevance is 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 include the concept of relevance in their results, with NLP services each service already includes the relevance concept for all services on a scale from 0 (irrelevant) to 1 (relevant), so we can directly use it. In our approach the consolidated and merged entities from service one and two are then in turn compared to extracted entities from service three (and so on, if we used even more services). In practice, however, due to the not always given interconnectedness of OpenCalais, there are no matches after having compared Zemanta-extracted entities with AlchemyAPI-extracted 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 (JSON) on both a per URI basis and an entity basis.

It is to be noted that the results from URI Lookup are a subset of the results from NLP in our case. However, while all URI Lookup services accept one-word arguments (e.g., "google" works), from the NLP services 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).

3.4 Design of the Web Service

Our Web service is designed with RESTful design principles in mind: properly named resources, use of the adequate HTTP verbs, and implementation of Hypermedia Controls (also known as HATEOAS[7]). 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 | POST /entity-extraction/{service_name}/{text_fragment}`[8]
- Getting an RDF annotation for a video with a given video ID: `GET /youtube/rdf/{video_id}`

[7] `http://martinfowler.com/articles/richardsonMaturityModel.html#level3`

[8] In the case of `POST`, the `{text_fragment}` has to be sent in the body of the HTTP message.

- Modifying or manually creating an RDF annotation for a video with a given video ID: `PUT /youtube/rdf/{video_id}`
- Deleting an RDF annotation for a video with a given video ID: `DELETE /youtube/rdf/{video_id}`
- Getting metadata from YouTube for a video with a given video ID: `GET /youtube/video/{video_id}`
- Getting all closed captions from YouTube for a video with a given video ID: `GET /youtube/video/{video_id}/closedcaptions`
- Getting closed captions in a given language from YouTube for a video with a given video ID: `GET /youtube/video/{video_id}/closedcaptions/{language_code}`
- Getting a plaintext audio transcription from YouTube for a video with a given video ID: `GET /youtube/video/{video_id}/audiotranscription`
- Getting a plaintext audio transcription 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 [**?**] as the central vocabulary, mainly because it already has a defined mapping not only for YouTube metadata, 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" (quoted from the introduction of [**?**]). 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 metadata.

4.2 YouTube Tags

In order to represent YouTube tags, or rather, semantically annotated YouTube tags, we use the Common Tag vocabulary [**?**]. A resource is `ctag:tagged` with a `ctag:Tag`, which consists of a textual `ctag:label` and a pointer to a resource that specifies what the label `ctag:means`. The Common Tag vocabulary is well-established and developed by both industry and academic partners.

4.3 Entities In Video Fragments

As stated before the current video annotation Web service is a re-implementation of our previous client-side application SemWebVid [**?**]. Hence we already could collect some experience with modeling video data in RDF on our own, and were also inspired by the semantic video search engine yovisto[9] [**?**, **?**]. In our first attempt we used the Event Ontology[**?**] 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 fragments instead. A video fragment stretches now over a complete sentence which usually contains more than just one line in the closed captions track, which much more matches the human perception of a self-contained incident in a video.

In order to address a video fragment, we decided to use Media Fragment URIs [**?**]. Media Fragment URIs are also supported by the Ontology for Media Annotation in form of `ma:MediaFragment`. In particular we use the temporal dimension (e.g., `http://example.org/video.webm#t=10,20`), which is defined by its start and 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.org/video.webm#track=closedcaptions`), which allows for addressing only a closed captions track (i.e., speech with time information), or even the plaintext audio transcription without time information (e.g., `#track=audiotranscription`). The value of the parameter `track` is a free-form string, so we are flexible with regards to its usage.

In the previous SemWebVid implementation we used `event:factor`, `event:product`, and `event:agent` to relate events with factors (extracted non-person entities), products (the particular plaintext closed captions line), and agents (extracted persons). Now, in order to annotate entities in a temporal video fragment, we consistently use the same vocabulary of Common Tag as outlined in Sect. 4.2. We thus have (in Turtle[10] syntax, left out prefixes for the sake of brefity):

```
<http://example.org/video.webm#t=10,20>
  a ma:MediaFragment ;
  ctag:tagged
    [ 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 a `ma:MediaFragment` is `ctag:tagged` with a `ctagTag` with the `ctag:label` in plaintext form `example`, where what this `ctag:label ctag:means` is represented by an example entity with 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 [**?**] by Hartig and Zhao. Even if the direct requests of our Web service were made against our wrappers (as outlined in Sect. 3.2 and Sect. 3.3), 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 describing 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

[9] `http://yovisto.com/`

[10] `http://www.w3.org/TeamSubmission/turtle/`

make statements about a bundle of triples, we group them in a named graph. We use the TriG [?] syntax. The example from above then looks like this:

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

5.1 The Provenance Vocabulary

In the next paragraph we outline the required steps in order to make statements about the provenance of a group of triples contained in a named graph `:G` that were generated using several HTTP `GET` requests to third party Web services. The text below can be best understood by following the triples in Appendix A.

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:DataCreatingService` to be precise. This service is `prv:operatedBy` a human (`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` a non-human actor (our Web service) that is `prv:operatedBy` a human (`http://tomayac.com/thomas_steiner.rdf#me`. For the `prv:DataAccess` (there is one for each third party Web service involved) we `prv:accessedService` from a `prv:DataProvidingService` 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".

5.2 Tracking Provenance With Human Interaction

Oftentimes completely automatically generated RDF video annotation files will need a bit of 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 us assume we wanted to replace an unfriendly Freebase `ctag:means` resource 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/3
      PuHGKnboNY>
  ctag:tagged [
    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"[11]-like anonymous pseudo-`prv:HumanActor`, or in the case of an intendedly non-anonymous `prv:HumanActor`, authentication methods like WebID[12] could be used.

5.3 The Need For Providing Provenance Metadata

Hartig et al. mention in [?] 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 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. This a) in order to correct the error at the root of our Web service (fighting the cause), b) in order to correct the concrete error in an RDF annotation (fighting the symptom), or c) probably the most important reason, to judge the trustworthiness and quality of a dataset.

6. RELATED AND FUTURE WORK

Related work includes Popcorn.js from the Mozilla Drumbeat project [?] that with its interactive Butter editor[13] allows for a video to be semantically annotated (a completely manual process). Based on 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 viewing experience. It also provides automatic machine translation from Google Translate, and attribution data from Creative Commons. The focus, however, is on the final visual video "mash-up", not on the actual annotation. Future work could be to offer an RDF-to-Popcorn.js wrapper service that would allow us to profit from the project's HTML5 video framework.

[11] `http://en.wikipedia.org/wiki/John_Doe`
[12] `http://www.w3.org/2005/Incubator/webid/charter`
[13] `http://popcornjs.org/butter/`

In [?], Waitelonis et al. address the problem of how to deploy exploratory search for video data by using semantic search technology for the yovisto 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 applied to the entire video and are entered by the initial video uploader, whereas video-time-related tags only apply to a certain point in the video. They 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 a different paper [?] Waitelonis et al. show how using permutations of a term and this term's surrounding context and by detecting paths between entities, a legacy keyword-based video search engine can be converted into 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 [?] 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 that 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[14] synsets if possible. That means 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 a Web service for 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 necessary 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 that led to consolidated results. It is to be noted that these references to the original calls are to be understood as the identificator of Web resources (i.e., the results of a request). Finally we positioned our work to related work, and presented directions for future work.

[14] `http://wordnet.princeton.edu/`

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

8. ACKNOWLEDGMENTS

We would like to thank Olaf Hartig from the Humboldt-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).

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` (for the sake of brefity only two of the `prv:usedData` sources are mentioned):

```
:G = {
  <http://gdata.youtube.com/feeds/api/videos/3
      PuHGKnboNY> ctag:tagged :tag .
  :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#me> .
        ] ;
        prv:accessedService <http://api.freebase.com
            /api/service/search> ;
        prv:accessedResource <http://api.freebase.
            com/api/service/search?format=json&query
            =obama> ;
        prvTypes:exchangedHTTPMessage [
          a http:Request ;
          http:httpVersion "1.1" ;
          http:methodName "GET" ;
          http:mthd <http://www.w3.org/2008/http-
              methods#GET> ;
          http:headers (
            [
              http:fieldName "Host" ;
              http:fieldValue "api.freebase.com" ;
              http:hdrName <http://www.w3.org/2008/
                  http-header#host> ;
            ]
          )
        ] ;
      ] ;
```

```
    ] ;
    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#me> .
        ] ;
        prv:accessedService <http://lookup.dbpedia.
            org/> ;
        prv:accessedResource <http://lookup.dbpedia.
            org/api/search.asmx/KeywordSearch?
            QueryString=obama> ;
        prvTypes:exchangedHTTPMessage [
          a http:Request ;
          http:httpVersion "1.1" ;
          http:methodName "GET" ;
          http:mthd <http://www.w3.org/2008/http-
              methods#GET> ;
          http:headers (
            [
              http:fieldName "Host" ;
              http:fieldValue "lookup.dbpedia.org" ;
              http:hdrName <http://www.w3.org/2008/
                  http-header#host> ;
            ]
          )
        ] ;
      ] ;
    ] ;
  ] .
} .
```