Tracking Provenance in an Entity-Consolidating Read/Write API for RDF YouTube 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 Vallés
Univ. Politécnica de Catalunya
ALBCOM and LSI Dept.
08034 Barcelona, Spain
gabarro@lsi.upc.edu

ABSTRACT

Using Natural Language Processing or URI lookup third party Web services for converting legacy unstructured data into Linked Data is a relatively straightforward task. In this paper, we first present an approach to 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. We discuss how manual changes to automatically generated RDF annotations can be tracked in this read/write-enabled Web service.

Categories and Subject Descriptors

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

General Terms

Experimentation

Keywords

RDF, Linked Data, Semantic Web, NLP, video annotations, provenance

1. INTRODUCTION

With SemWebVid [9], we introduced a client-side interactive Ajax application for the automatic generation of RDF video annotations. In 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. In the background, this Web service calls several enrichment services in parallel and merges their results in order to consolidate entities. We present two linking algorithms for two different kinds of enrichment services: Natural Language Processing (NLP) and URI Lookup Web services. Our objective is to publish legacy data sources as Linked Data on the Web, with the option of a write-enabled back-channel for manual corrections. Upon merging results from different data sources, being able to track back provenance of Linked Data is essential in order to judge its trustworthiness and reliability. In this paper, we propose an approach for maintaining trackable provenance metadata.

A YouTube video is described by a Google Data Atom feed[1]. In order to semantically annotate a YouTube video, we concentrate on the following fields of this feed (in XPath syntax): title (`/entry/media:group/media:title`), description (`/entry/media:group/media:description`), and tags (`/entry/media:group/media:keywords`). Other technical YouTube metadata (e.g. duration, creation date) are also exposed as Linked Data since we benefit from the mappings established by the W3C Ontology for Media Resources (see also the section 4.1).

In the following, we differentiate subtitles from closed captions, where subtitles are hard-encoded into the video frames, while closed captions are separate textual resources. Currently, we only work with closed captions. Closed captions are based on audio transcriptions, which consist of a plain-text representation of speech of a video, which are then time aligned to the video. YouTube offers an automatic audio transcription service and video owners can also upload audio transcriptions and/or closed captions files on their own[2]. Since YouTube supports closed captions in several languages, we also use this data when available for the task of semantic video annotation[3].

[1]See `http://gdata.youtube.com/feeds/api/videos/Rq1dow1vTHY` for a concrete example.

[2]`http://googleblog.blogspot.com/2009/11/automatic-captions-in-youtube.html`

[3]See `http://www.youtube.com/watch_ajax?action_get_caption_track_all&v=Rq1dow1vTHY` for a concrete example.

The remainder of this paper is structured as follows: In Section 2, we introduce two classes of Web services that allow for unstructured data to be converted into Linked Data. In Section 3, we detail our entity consolidation with URI Lookup and NLP Web services approach. In Section 4, we present how video annotations are represented in RDF. In Section 5, we describe how we automatically maintain provenance metadata in our Web service. We discuss related work in Section 6 and finally conclude in Section 7.

2. WEB SERVICES FOR CONVERTING UNSTRUCTURED DATA INTO LINKED DATA

In this section, we describe both Natural Language Processing (NLP) and URI Lookup Web services.

2.1 NLP Web Services

The NLP Web services that we use for our experiments take a text fragment as an input, perform Named Entity Recognition (NER) on it and then link the extracted entities back into the Linked Open Data (LOD) cloud[4]. We use OpenCalais[5], Zemanta[6] and AlchemyAPI[7] as NLP Web services.

2.2 URI Lookup Web Services

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[8], DBpedia Lookup[9], Sindice[10], and Uberblic[11] as URI Lookup Web services.

2.3 Technical Note

For both classes of services, we use them in parallel, aiming for the emergence effect in the sense of Aristotle: "*[...] the totality is not, as it were, a mere heap, but the whole is something besides the parts [...]*"[12]. Our Web service is written in JavaScript and is based on Node[13] and the Express[14] framework. Therefore, we use the third parties' JSON or XML APIs as a means of communication with all services, and not the potential SPARQL endpoints or RDF APIs. 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 several 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 first analyze the tags one-by-one with URI Lookup Web services through our wrapper Web service. We then analyze the title combined with the description (first run) or the closed captions track (second run) through our wrapper Web service to the NLP Web services. We choose this order because we observe that, sometimes, video description and closed captions are not related at all. This happens when the video description gets used not in a way to describe the video content on a high level but, for example, to advertise different videos from the same user. In this case, the NLP gets steered in a completely wrong direction. During our experiments, this occurred often enough for us to separate the NLP analysis of descriptions and closed captions into two independent steps. When all analysis steps are completed, we try to match the extracted entities from all classes of services back in the video. In the following, we present our approach for how to consolidate entities from URI Lookup Web services and 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 assimilate 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. In the example below, we use the term "Google Translate" to illustrate the approach. Three services return back a dbpedia URI, Sindice being the only one returning a different result in a first place.

```
[
  {
    "name": "Google Translate",
    "uris": [
      "http://dbpedia.org/resource/Google_Translate"
    ],
    "source": "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` while the particular results from each service are available at `/uri-lookup/{service_name}/Google%20Translate`. We observe in this example that even in the lowest data representation level (JSON), we maintain provenance metadata in the form of a `source` field.

In order to agree on a winner entity, a majority-based voting system is used. As soon as for two entities only one of these entities' URIs match, we consider the entity consolidated and can merge the results. 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 (Sindice returns other results as well). 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. Therefore, by retrieving and parsing the referenced resources in the services' namespaces, we can map back to DBpedia URIs and thus match all four services' re-

[4] http://lod-cloud.net/
[5] http://www.opencalais.com/documentation/
[6] http://developer.zemanta.com/docs/
[7] http://www.alchemyapi.com/api/entity/
[8] http://wiki.freebase.com/wiki/Search
[9] http://lookup.dbpedia.org/
[10] http://sindice.com/developers/api
[11] http://uberblic.org/developers/apis/search/
[12] Aristotle, Metaphysics, Book H 1045a 8-10.
[13] http://nodejs.org/
[14] http://expressjs.com/index.html

sults on the DBpedia level. Each service's result contributes with a relevance of 0.25 to the final result. In this 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

Similarly to URI Lookup entity consolidation, we have implemented a wrapper API for the three NLP services. While the original calls to each particular NLP service are all HTTP `POST`-based, we have implemented the wrapper `GET`- and `POST`-based. All NLP Web services return entities with their types and/or subtypes, names, relevance, and URIs that link into the LOD cloud. The problem is that each service has implemented its own typing system and providing mappings for all of them would be a relatively time-consuming task. However, as all services provide links into the LOD cloud, the desired typing information can be pulled from there in a true Linked Data manner. We thus have all the information we need if named entities with interlinked URIs are detected. The least common multiple of the results for the query "Google Translate" is depicted below. For the sake of clarity, we just show one entity with two URIs while the original result contained seven entities among which six were relevant and one was just related.

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

These results come from a request to our wrapper API via `GET /entity-extraction/combined/Google%20Translate`, and similarly to the 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). In this particular case, retrieving the resource RDF representation and parsing for `owl:sameAs` return links to DBpedia. However, in the general case, we found OpenCalais URIs sometimes pointing to non-existent resources or to not very rich resources such as `http://d.opencalais.com/pershash-1/cfcf1aa2-de05-3939-a7d5-10c9c7b3e87b.html`, a URI identifying 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. The examples below illustrate this process.

Results for the text fragment from AlchemyAPI only:

```
{
  "name": "Google",
  "relevance": 0.496061,
  "uris": [
    {
      "uri": "http://dbpedia.org/resource/Google",
      "source": "alchemyapi"
    },
    {
      "uri": "http://rdf.freebase.com/ns/guid.9202
          a8c04000641f800000000042acea",
      "source": "alchemyapi"
    },
    {
      "uri": "http://cb.semsol.org/company/google.
          rdf",
      "source": "alchemyapi"
    }
  ],
  "source": "alchemyapi"
}
```

Results for the text fragment from Zemanta only:

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

Results merged from both Zemanta and AlchemyAPI:

```
{
  "name": [
    "Google",
    "Google Inc."
  ],
  "relevance":  0.5295965,
  "uris": [
    {
      "uri": "http://dbpedia.org/resource/Google",
      "source": "alchemyapi"
    },
    {
      "uri": "http://rdf.freebase.com/ns/guid.9202
          a8c04000641f800000000042acea",
      "source": "alchemyapi"
    },
    {
      "uri": "http://umbel.org/umbel/ne/wikipedia/
          Google",
      "source": "alchemyapi"
    },
    {
      "uri": "http://cb.semsol.org/company/google.
          rdf",
      "source": "alchemyapi"
    },
    {
      "uri": "http://rdf.freebase.com/ns/en/google",
      "source": "zemanta"
```

```
    },
    {
      "uri": "http://cb.semsol.org/company/google#
          self",
      "source": "zemanta"
    }
  ],
  "source": "alchemyapi,zemanta"
}
```

In this example, 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, 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 the 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.

Given the two entity names mismatch ("google inc." vs. "google"), the consolidated name is then an array of all detected synonymous. 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 since not all URI Lookup services include the concept of relevance in their results, with NLP services, each service already includes a relevance score ranging 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. Similarly to URI Lookup-detected entity consolidation, we also maintain provenance metadata for each URI on the lowest data representation level (JSON) on both a per URI basis and an entity basis with the NLP-detected entity consolidation.

We note that in our example, the results from URI Lookup are a subset of the results from NLP. However, while all URI Lookup services accept one-word arguments (e.g., "google"), only AlchemyAPI from the NLP services accepts one-word arguments. The two other services accept only non-trivial text fragments (e.g., "google is a company founded by larry page").

3.4 Identity Links On the Semantic Web

In order to tackle the problem of different namespaces in results that we have outlined in the Section 3.2, a straightforward idea is to use a Web service such as <sameAs>[15] to easily find mappings from one namespace into another. In practice, however, while many data sources in the Linked Data world are marked as being equal to each other (e.g., `http://dbpedia.org/resource/Barack_Obama` `owl:sameAs` `http://rdf.freebase.com/rdf/en.barack_obama`), the quality of such equality links is not always excellent. As Halpin et al. show in [5], the problem with `owl:sameAs` is that people tend to use it very differently. The authors of [5] differentiate four separate usage styles, each with its particular implications. Inference is thus problematic, if not impossible, when the sense of the particular use of `owl:sameAs` is unknown.

[15] `http://sameas.org/`

3.5 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[16]). Currently, the 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}`[17]
- Getting an RDF annotation for a video with a given video ID: `GET /youtube/rdf/{video_id}`
- 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 plain text audio transcription from YouTube for a video with a given video ID: `GET /youtube/video/{video_id}/audiotranscription`
- Getting a plain text 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. DESCRIBING VIDEOS IN RDF

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

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

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

4.1 Basic YouTube Metadata

We use the W3C Ontology for Media Resources [11] as the central vocabulary, mainly because it already defines a set of mappings[18] not only for YouTube metadata, but also for many other existing metadata formats. This ontology aims to foster the interoperability among various kinds of metadata formats currently used to describe media resources on the Web. From this vocabulary, we use the following properties: `ma:title`, `ma:creator`, `ma:createDate`, `ma:description` and others technical properties which 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 [3]. 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 [9]. 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[19] [12, 13]. In our first attempt, we used the Event Ontology[8] and defined each line (which not necessarily corresponds to a complete sentence) 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. We split the single lines in the complete closed captions track in complete sentences. Consequently, a video fragment now stretches over a complete sentence which usually contains more than just one line in the closed captions track. This matches much more the human perception of a self-contained incident in a video.

For addressing a video fragment, we use the Media Fragment URIs [10] specification. Media Fragment URIs are also supported by the Ontology for Media Resources via the `ma:MediaFragment` property. 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. The value of the parameter `track` is a free-form UTF-8 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 (non-person entities extracted), products (particular plain text closed captions lines), and agents (persons extracted). In the current implementation, we consistently use the same Common Tag vocabulary to annotate entities in a temporal video fragment (see Section 4.2). We thus have:

```
<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

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 plain text 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 seven agreeing (or disagreeing) Web services. In order to track the contributions of the various sources, we decided to use the Provenance Vocabulary [7] by Hartig and Zhao. Even if the direct requests of our Web service were made against two wrappers (see Section 3.2 and Section 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 make statements about a bundle of triples, we group them in a named graph. We use the TriG [1] syntax:

```
: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 this section, 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 was 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`

[18] `http://www.w3.org/2008/WebVideo/Annotations/drafts/ontology10/CR/test.php?table=YouTube`

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

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

RDF video annotations that are completely automatically generated often need to be manually corrected. In our RESTful Web service, we have thus envisioned that not only a big archive of automatically generated video annotations gets built, but that also people can correct errors (via HTTP PUT or later PATCH) in the RDF interactively (or remove completely wrong video annotations via HTTP DELETE). 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>
    ]
  ] .
```

The prv:DataCreation no longer contains references to prv:usedData. Obviously, this approach to identify a prv:HumanActor with a FOAF profile requires an authentication step which might not always be desired. One could think of a "John Doe"-like anonymous pseudo-prv:HumanActor, or in the case of an intendedly non-anonymous prv:HumanActor, authentication methods like WebID[20] could be used.

5.3 The Need for Providing Provenance Metadata

Hartig et al. mention in [6] 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 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 enables a) to correct the error at the root of our Web service (fighting the cause), b) to correct the concrete error in an RDF annotation (fighting the symptom), or c) to judge the trustworthiness and quality of a dataset which is probably the most important reason.

5.4 Change Of Referenced Datasets Over Time

It is to be noted that a statement such as the triple below refers to the triple object as an identifier for a Web resource (where the Web resource is a representation of the result of the API call at the time where it was prv:performedAt).

```
_:x prv:accessedResource <http://api.freebase.com/
     api/service/search?format=json&query=obama> ;
```

As provenance metadata always refer to the time context in which a certain statement was made, it is essentially unimportant what representation the resource returns at a later time.

6. RELATED AND FUTURE WORK

Related work includes Popcorn.js from the Mozilla Drumbeat project [4] that allows for a video to be semantically annotated using its interactive Butter editor[21] (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.

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 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 [12]. This 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.

[20] http://www.w3.org/2005/Incubator/webid/charter

[21] http://popcornjs.org/butter/

In [2], 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 user-generated tag space which is often sparse, metadata such as the recording time and location, the video title and video description, social features such as playlists that a video appears in and related videos, etc. are used. 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 using WordNet[22] synsets if possible. This means that 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 are 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. 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 the original third party Web service calls that led to the consolidated results. 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 with respect to the state of the art and presented directions for future work.

In this paper, we have also 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 crucial importance, especially given the network effect which is one of the Linked Data benefits.

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 was partly supported by the European Commission under Grant No. 248296 FP7 I-SEARCH project and by the French Ministry of Industry (*Innovative Web* call) under contract 09.2.93.0966, "Collaborative Annotation for Video Accessibility" (ACAV).

8. REFERENCES

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

[2] S. Choudhury, J. Breslin, and A. Passant. Enrichment and Ranking of the YouTube Tag Space and Integration with the Linked Data Cloud. In *8th International Semantic Web Conference (ISWC'09)*, pages 747–762, Chantilly, VA, USA, 2009.

[3] Common Tag. Common Tag Specification, January 11, 2009. http://commontag.org/Specification.

[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] H. Halpin and P. Hayes. When owl:sameas isn't the same: An analysis of identity links on the semantic web. In *3rd Workshop on Linked Data on the Web (LDOW'10)*, Raleih, NC, USA, 2010.

[6] O. Hartig and J. Zhao. Publishing and Consuming Provenance Metadata on the Web of Linked Data. In *3rd International Provenance and Annotation Workshop (IPAW'10)*, Troy, NY, USA, 2010.

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

[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 (ISWC'10)*, Shanghai, China, 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 (SAMT'10*, Saarbrücken, Germany, 2010.

[13] J. Waitelonis, H. Sack, J. Hercher, and Z. Kramer. Semantically enabled exploratory video search. In *3rd International Semantic Search Workshop (SemSearch'10)*, pages 81–88, 2010.

[22] http://wordnet.princeton.edu/

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 brevity, 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> ;
            ]
          )
        ] ;
      ] ;
    ] ;
  ] .
} .
```