published as: Waitelonis, J.; Ludwig, N. & Sack, H.: Use What You Have -- Yovisto Video Search Engine Takes a Semantic Turn. In Proceedings of 5th Int. Conf. on Semantic and Digital Media (SAMT 2010), December 1-3, 2010, Saarbrücken, DFKI Saarbrücken, 2010.

Use What You Have: Yovisto Video Search Engine Takes a Semantic Turn

Jörg Waitelonis, Nadine Ludwig, and Harald Sack

Hasso-Plattner-Institute Potsdam,
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
{joerg.waitelonis,nadine.ludwig,harald.sack}@hpi.uni-potsdam.de
http://www.hpi.uni-potsdam.de/

Abstract. The phenomenal increase of online video content confronts the consuming user with an immeasurable amount of data which can only be accessed with sophisticated multimedia search and management technologies. Usual video search engines provide a keyword-based search, where lexical ambiguity of natural language often leads to imprecise and incomplete results. Semantics of keywords and metadata has to be determined to overcome these shortcomings to provide high precision and high recall. In this work, we show how to gradually transform the video search engine Yovisto from a simple keyword-based search engine to a fully-fledged semantic video search engine simply by using the existing search engine infrastructure based on Lucene augmented by simple semantic metadata.

1 Introduction

There is a continuous increasing demand for online videos in recent years. With more than 14.6 billion video downloads from YouTube[1] in May 2010 only in the USA[2], video and multimedia is becoming the predominant part of internet traffic. Besides, the entire World Wide Web (WWW) with a growing amount of realtime data has risen beyond any expectations. For the user, the only way to orientate oneself in this huge amount of data and to find the needle in the haystack is making use of search engines.

But, also web search engines are reaching their limits. With the sheer quantity of retrieved documents the user is faced with an almost insolvable task of deciding, whether the desired document really is among the result list or if the result set is complete at all. A simple Google[3] search often returns millions of documents ranked by link popularity. But, what if the desired result is not among the first few result list pages? Traditional web search engines are based on keywords, i.e. keywords are extracted from web documents for distinctive representation.

[1] http://www.youtube.com/
[2] http://www.comscore.com/Press_Events/Press_Releases/2010/6/comScore_Releases_May_2010_U.S._Online_Video_Rankings
[3] http://www.google.com/

The user tries to identify the document(s) she is looking for with a query phrase consisting out of one or more keywords, which she supposes to match the keywords being extracted and indexed by the search engine. Unfortunately, besides different user context and pragmatics, natural language suffers from inherent ambiguities that cause deficits in search engine recall and precision.

One way to cope with these ambiguities is to enable a better understanding of the meaning of the web document's content as well as of the user's query string. Semantic Web technologies intend to make implicit meaning of content explicit by providing suitable metadata annotations based on formal knowledge representations. Instead of extracting keywords from documents the document content is mapped to distinct entities and classes to avoid ambiguities and to enable higher recall and precision of search results. Content-based search based on Semantic Web technologies often is referred to as 'Semantic Search'. Besides, the problem of different contexts and pragmatics remains, but is not taken into consideration for this paper.

But, do we need an entire new search engine architecture for enabling semantic search? We will show how an existing keyword-based search engine can be augmented by simple means to become a fully-fledged semantic search engine capable of content-based retrieval. To achieve this, we will show how to enable a mapping of index keywords, user provided tags as well as the terms in the user's query string to Semantic Web entities and classes with the help of Linked Open Data, linguistic and lexical resources. Entities and classes in the Semantic Web are identified by URIs (Uniform Resource Identifiers) that will be utilized instead of former text-based keywords within an existing open source enterprise search platform to enable semantic search. The paper will give a step-by-step description on how to turn a keyword-based search engine into a semantic search engine demonstrated at the video search engine Yovisto[4].

The rest of the paper is structured as follows: Section 2 will summarize related work on keyword-based search, multimedia search, and semantic search with a focus on entity mapping and disambiguation. In Section 3, we will give a step-by-step description on how we achieved the turning into a semantic search engine with onboard means. Section 4 concludes the paper with a short summary and an outlook on open problems and future work.

2 Related Work

2.1 Keyword-based Search

Since Google's conquest of the web search engine universe we know that the quality of keyword-based search mainly depends on the ranking of search results, but also lacks on problems coming with the peculiarities of natural language. Synonyms and homonyms cause incomplete and inaccurate search results. Keyword-based search presumes that the user knows the exact keyword for describing the document she is looking for. If the user does not know the appropriate keyword

[4] `http://www.yovisto.com`

or if she is trying to find the right documents to answer more complex query tasks, the search objective can become unattainable. Query expansion and suggestions help the user to refine the search results and enable a step-by-step convergence [3]. Filtering methods such as facetted search categorize search results and enable to limit the results to subsets for better overview [14]. Beyond the simple keyword lookup, the user can reiterate a sequence of queries while refining and/or extending the query depending on the content of the obtained results. Search scenarios sophisticated and complex like this are referred to as exploratory search [7].

2.2 Multimedia Search

The same inadequacies can be found in the more and more emerging multimedia search. General video search engines such as YouTube support a keyword-based search within the textual metadata provided by the users, accepting all the shortcomings caused by e.g. synonyms and homonyms. For example, a search for "history of golf" will result in documents containing a variety of outcomes for 'Golf' (e.g. sports, car) and by refining the search phrase to "history of golf car" it is astonishing that the top ranked videos are about the 'golf cart', which is obviously not a product of Volkswagen, but has at least four wheels. The ranking of the correct results is thwarted by the automated query completion. However, multimedia search lacks the same problems as keyword-based text search. Compared to textual search, multimedia search allows varied visualizations [15, 4].

The video search engine Yovisto is specialized on lecture recordings and scientific presentations and provides a time-dependent video index. With sophisticated video analysis techniques (such as automated scene detection, intelligent character recognition, etc.) in combination with collaborative user annotation Yovisto provides scene accurate access to more than 10.000 videos with pinpoint accuracy [5, 12]. To overcome some of the the shortcomings of keyword-based search, Yovisto deploys an exploratory search navigation to allow the user to browse the repository beyond simple fact-finding and page-turning [18–20].

2.3 Linked Data and Semantic Search

Yovisto enables the exploratory search by applying *Linked Open Data*[5] (LOD). Linked Data means to expose, share and connect pieces of data, information, and knowledge on the Semantic Web using URIs for identification of resources and RDF[6] as structured data format [1]. The Linked Open Data project aims to publish and connect open but heterogeneous databases by applying the Linked Data principles. One of the most important interlinking LOD hubs is *DBpedia* [2], which publishes encyclopedic information from the famous Wikipedia[7]. DBpedia currently provides information about more than 3.4 million "things" with over 1

[5] `http://linkeddata.org/`
[6] `http://www.w3.org/RDF/`
[7] `http://www.wikipedia.org/`

billion "facts"[8]. Furthermore, *Wortschatz Leipzig* in an automatically compiled thesaurus, which collects large volumes of natural language text from the WWW and applies sophisticated linguistic and statistic analysis in large scale to provide thesaurus information. Wortschatz Leipzig supports more than 17 languages and is also publicly available as RDF Linked Data [13]. The aggregation of all LOD data sets is denoted as *LOD-Cloud*.

To access and to make use of the Semantic Web information, search engines need to index semantic data. Commonly, this can be achieved by storing the data in RDF databases (triple stores). This allows to query the data with structured languages such as SPARQL [11]. Search engines such as *sindice* [8, 17] or *swoogle*[9] are crawling the Semantic Web to obtain a data set as large as possible and provide a label and entity based search. Those search engines are already operating on the existing Semantic Web, while this work focusses on how to turn a non-semantic web search engine into a semantic web search engine. Therefore, the already existing keyword-based textual metadata has to be mapped to Semantic Web entities.

The most challenging problem on mapping data to Semantic Web entities is the existence of ambiguous names and thus resulting in a set of entities, which have to be disambiguated. Related work fields are word-sense disambiguation in text documents, named entity (reference) resolution, and feature based entity matching amongst others. The presence of assumed same named entities in different triple stores of the LOD-Cloud necessitates similarity based comparison of the entities and their respective features or properties [16]. In the context of named entity resolution in text documents the semantic information needed for disambiguation of potential entities has to be extracted automatically and compared to adequate knowledge resources [9]. Further research approaches are using the LOD-Cloud itself as RDF graph to find relations between entities co-occurring in a text maintaining the hypothesis that disambiguation of co-occurring elements in a text can be obtained by finding connected elements in an RDF graph [6].

3 How to turn Yovisto into a Semantic Video Search Engine

Turning Yovisto from a keyword-based search engine to a semantic video search engine needs several steps. Fig. 1 shows an overview of the process sequence to enable semantic search with Yovisto. Yovisto's time-dependent as well as time-independent metadata is mapped to entities of LOD, respectively DBpedia (a). The Wortschatz Leipzig co-occurence network is applied to support entity disambiguation and ranking of entity candidates (b). Within the indexation process the old keyword index is extended by additional semantic information (c). On the users hand, the query is disambiguated with an auto-suggestion select box,

[8] `http://dbpedia.org`
[9] `http://swoogle.umbc.edu/`

providing the URIs of the requested entity (d). Finally, the index lookup returns a list of documents containing the entity (e).

According to the Linked Data principles, Yovisto's metadata has been linked to the LOD-Cloud by connecting organizations, categories, and persons (speaker) to DBpedia entities including a mapping of Yovisto's metadata structure to an appropriate ontology[10] representation [19]. To ensure interoperability, mappings and references to existing ontologies have been used whenever possible. All metadata has been published through a SPARQL-endpoint[11] and embedded via RDFa in Yovisto's web pages.

The mapping of video content to LOD entities is the subject of this paper. The video content is described as well by metadata referring to the entire video (e.g. title) as also by information assigned to a distinct time position within the video. Among these time-dependent metadata are collaboratively generated user tags, user comments, and text extracted from the simple video frames. The following section deals with the entity mapping of both kinds of metadata.

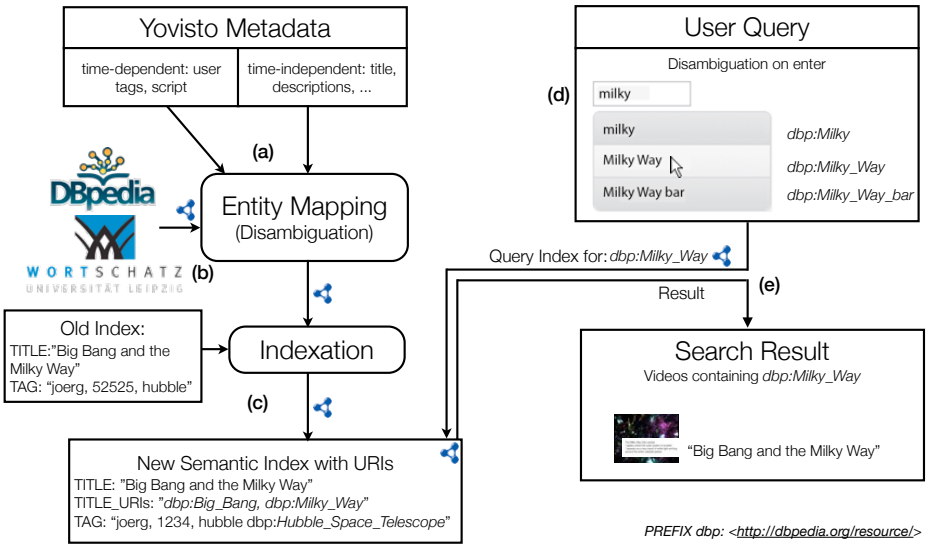

Fig. 1. Overview of the processes of semantic search with Yovisto.

3.1 Entity-Mapping

Currently, user-generated tags and automatically extracted data are provided as text elements essentially for the keyword-based search. To transform the

[10] http://www.yovisto.com/ontology/
[11] http://sparql.yovisto.com/

keyword-based search engine to a semantic video search engine the text elements need to be mapped to semantic web entities to set a certain segment respectively a whole video in the context of the LOD-Cloud. The following section describes how to use the existing tags and keywords to create a semantic context mapped to entities of the LOD-Cloud. An overview of all steps of the entity mapping process are shown in Fig. 2.

Term and Group Mapping Yovisto provides several methods to equip a video with metadata: video-related keywords and video-time-related keywords and tags. Video-related keywords are supplied by the user who uploaded a video to the portal and this metadata is applied to the whole video. Video-time-related metadata are on the one hand keywords that are automatically extracted from the video on a certain timestamp (e.g. by OCR methods) and on the other hand user-generated tags also on a certain timestamp in the video.

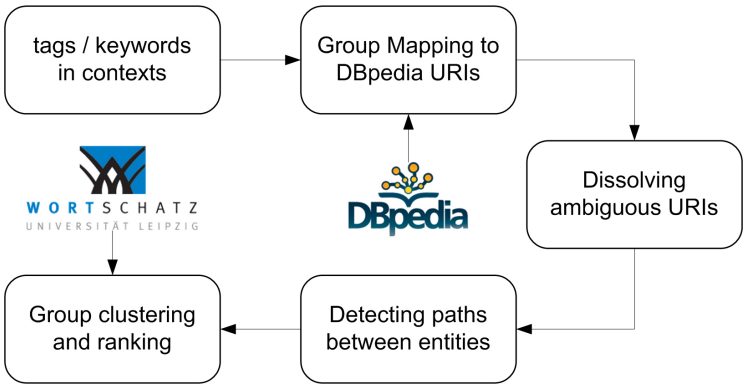

Fig. 2. Overview of Steps in the Entity Mapping Process

Metadata exist in a certain context and have to be mapped to Semantic Web entities within this context. We define the context of a term (tag or keyword) in a video as the tags and keywords at and around the timestamp in the video the term is occurring at. All tags and keywords are stored separately with timestamp and video ID in the database.

In a preprocessing step we generated a "term mapping table" by using labels, normalized URI-suffix, and labels of redirects and assigning them to the corresponding DBpedia URI. These terms are single words, as well as composed groups of words e.g., names ("Albert Einstein").

The first step to map the metadata of a video to Semantic Web entities is to gather all tags/keywords of a context and build groups to be able to also map groups of tags to entities. Let us consider the following example: a user tagged "hubble", "deep", "space", "exploration", "field" in that order. In that context it might be useful to allocate the DBpedia entities for "deep space exploration",

"hubble" (space telescope), and "hubble deep field". Therefore a variation without repetition (to avoid combinations of a tag with itself) on this set of all tags in this context has to be constructed. In this example the set contains $n = 5$ elements and groups of $k = 1, 2, 3$ elements have to be built.

$$(n)_k = \frac{n!}{(n-k)!}$$

For our example, this leads to $(n)_k = 85$ combinations in total. However not all combinations are needed, only the groups that match to a term (from the term mapping table) are stored with their respective URI. This term matching adds up to 13 URIs. As every matching group is stored, there are also URIs for single terms like "field". Such single terms are often assigned to ambiguous URIs and after resolving these ambiguous URIs (also for the terms "deep", "field", "deep space", "hubble", "space") the result set of possible applicable URIs at this timestamp in this video contains 152 URIs. This enormous amount of matching URIs to a set of only five tags in a context leads to the inevitable problem of disambiguation and ranking of the matching URIs.

Problems of ambiguous URIs in DBpedia: In the example mentioned above we only disambiguated the resulting URIs in one "lap". The 152 URIs still contain three ambiguous URIs leading to 24 additional URIs. That means the algorithm for resolving ambiguous URIs has to be run repeatedly taking into account that DBpedia contains "circles" of disambiguations referencing back to already resolved URIs and these circles have to be detected and erased. For performance reasons we constrain the process of resolving ambiguous URIs to two runs.

Currently, Yovisto maintains approximately 22.936 user tags in 9271 contexts (timestamps) in 517 videos. The tags were mapped as single words and groups of 2 and 3 words – 30.417 groups were mapped to DBpedia URIs. The ambiguous URIs were resolved in a first run resulting in 162.523 URIs, after a second run of resolving 192.056 URIs were found.

Disambiguation of URIs The result set of URIs mapped to the matching groups of tags contains duplicate URIs with different matching groups - e.g., the matching groups "deep field" and "hubble deep field" were both assigned to the URI `<http://dbpedia.org/resource/Hubble_Deep_Field>`. So these duplicate records can be filtered by deleting the records with synonyms as matching groups. In the example, the URI contains as label the term "hubble deep field", so that the record with the matching group "deep field" can be removed from the result set.

For the further disambiguation steps of the diverse URIs in one matching group we assume that entities that are related to each other are also referenced over few properties in the LOD-Cloud (sample relationships of the example entities "Hubble Space Telescope", "Hubble Deep Field", and "Deep Space Exploration" are shown in Fig.3. Therefore, the applicable URIs from the set of URIs

for one matching group should be linked to one or more other URIs of the context. We consider an algorithm, which is detecting paths (with properties and other entities as nodes) between the entities of a context. The result set of this path-detecting algorithm is one or more sub-graphs of the LOD-Cloud. Because the amount of properties in the LOD-Cloud is quite large, we need to identify the most meaningful properties that link related entities in a certain context. Also, the degree of relationship - conveyed by the length of the path between two entities - has to figured to meet the right measurement to find both entities from the same context and filter out relations that are too distant to be considered from the same context [6].

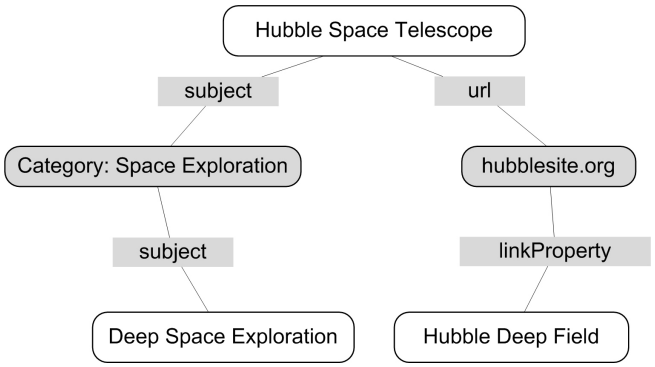

Fig. 3. Relation of relevant URIs for the examples tag set {hubble, deep, space, exploration, field}

Special Case – Simply one Ambiguous Tag in a Context: In case there is only one tag / keyword at a certain timestamp we have to create a context by comprising adjacent contexts.

The algorithm for detecting related entities generates a graph of entities. Related entities should be linked to each other and entities not applicable for the concerning context should stand alone. The set of entities derived from resolving ambiguous entities is disambiguated by detecting a path between the applicable URI from the set and other URIs from the context.

Ranking of URIs The path-detecting algorithm produces n graphs of size $k = 1...m$. Graphs of size $k = 1$ will be deleted in case the concerning entity is retrieved from resolving an ambiguous URI, because there is no relation to any other element from the context. In case of an URI building an own graph of size $k = 1$ and not being retrieved from resolving the URI will be ranked very low in this context, because the URI is probably of low relevance for this context. The elements of graphs of size $k > 1$ will be ranked higher with increasing k. Within a graph the entities are ranked by the amount of in-going / out-going edges.

Additionally, the already disambiguated URIs are ranked by their co-occurrences. Wortschatz Leipzig provides amounts of co-occurrences for single words as well as composite word groups. The co-occurences of a term contain weightings that we are using to apply an additional score to our clusters of URIs. The ranking of URIs is then weighted regarding their co-occurrences and their relations in the LOD-Cloud.

Special Case – Name Components of Person Names: The group mapping algorithm maps single words and groups of 2 and 3 words to entities. Thus, in case of the context {peter, goddard} the algorithm will find mappings for "peter", "goddard", and "peter goddard". Certainly, the mapped URI for the whole name should be ranked higher than the URIs forname components. But, besides the whole name the URIs for the male name "Peter" and the family name "Goddard" should also be included in the set of assigned URIs to a video. In case a user searches for videos involving male persons in it or all persons with the family name "Goddard" these URIs need to be included rather than deleting them from the result set. This special case also affects the disambiguation process as the URI `<http://dbpedia.org/resource/Peter>` disambiguates to 31 further URIs, but in this case (and any other case the URI is referenced to a part of a person's name) only the URI `<http://dbpedia.org/resource/Peter_(first_name)>` is needed.

3.2 Indexation of Semantic Data

Entity-mapping and disambiguation are key problems in enabling a semantic search. Nevertheless, sophisticated indexing of the generated semantic data cannot be neglected to facilitate efficient search. This section shows that it is rather simple to extend a state-of-the art keyword-based search index with additional semantic information to enable an entity-based search.

The Yovisto Index The Yovisto index is based on Solr[12], a Java open source enterprise search platform from the Apache Lucene[13] project. Major features are full-text search, hit highlighting, facetted search, and dynamic clustering. The indexing of documents requires the configuration of index fields, which make up the index schema. The schema specifies, which fields a document can contain and how those fields should be handled, when adding new documents to the index, or when querying those fields.

Corresponding to the structure of Yovisto's metadata, there are three different kinds of metadata:

(1) information referring to the entire resource such as the video title, subtitle, description, etc.,
(2) user generated tags assigned to a specific point in time of the video [5],

[12] http://lucene.apache.org/solr/
[13] http://lucene.apache.org/

(3) time related text/keywords from automated analysis of the video (such as OCR on video frames).

In accordance with the type of metadata the indexing scheme has to be designed. For simple metadata (e.g. title) a single index field is sufficient to enable proper indexation. Time-dependent metadata carries the temporal information and is indexed either in one single field as (time, data)-tuple or in two dynamic fields with a mutual identifier, such as TIME_i, DATA_i. In case of user generated tags, the username has to be taken into account. Therefore, a (username, time, text)-triple is stored in one single field.

```
(a)
TITLE:  "The Future of the Web"
DESCRIPTION: "We're approaching the end of 2010, and many people are
              wondering what the future will bring."
AUTHOR: "Cameron Chapman"

(b)
TAG: "[joerg, 31234] future"
TAG: "[joerg, 51231] internet"
TAG: "[nadine, 31215] future "

(c)
SEG_1_TEXT: "iPhone, MySpace, Facebook in 2025"
SEG_1_TIME: "121000"
SEG_2_TEXT: "No Print Magazines Have to Die"
SEG_2_TIME: "127000"
```

Fig. 4. The indexing scheme for simple metadata (a), time-dependent user tags (b), time-dependent text/keywords (c). The numbers represent the time point in milliseconds of the metadata within the video.

Fig. 4 shows the indexing structure of simple metadata, time-dependent user tags, and text/keywords in Yovisto. For every index document *single valued fields* are used for simple metadata (e.g. TITLE), *multi valued fields* are used for user generated tags (e.g. TAG), and *dynamic fields* are used to index more complex time-dependent data (e.g. SEG_*_TEXT). When the fields are indexed, analyzing and tokenizing filters are applied to transform and normalize the field values. For example the *solr.WhitespaceTokenizer* creates tokens of characters separated by splitting on whitespace, and the *solr.PorterStemFilterFactory* applies the Porter stemming algorithm [10] on the tokens to enable to search regardless of inflection forms. The stemming filter is *not* applied to the tag fields, because an exact match is desired here.

Extending the Index with Semantic Data As explained in section 3.1 for every metadata keyword a list of URIs with a corresponding rank is created. The rank indicates how confidently the URI was mapped to the keyword. To supplementarily index the corresponding URIs besides the original metadata, the idea is, to simply create additional fields. These additional fields comprise the URIs with corresponding rank and the corresponding position within the text data. In some cases it is even possible to extend the field instead of creating a new one. Note that adding new fields makes it very easy to adapt a regular search index scheme and extend it with the semantic information. Therefore, it is not necessary to modify the indexer implementation to enable a semantic supported search.

For every single valued field an additional URI field (TITLE_URI) is defined (cf. Fig. 5a). The user tag fields can be extended with the URIs. The text position can be omitted, because it would always be 0 (cf. Fig. 5b). The dynamic fields also require an additional field for the URIs, similar to the single value fields (cf. Fig. 5c).

```
(a)
TITLE: "The Future of the Web"
TITLE_URI: "<http://dbpedia.org/resource/Future> 0.9432 1 <http://dbpedia
           .org/resource/Future_tense> 0.487 1 <http://dbpedia.org/resou
           rce/Web> 0.9342 4"

(b)
TAG: "[joerg, 31234] future <http://dbpedia.org/resource/Future> 0.9432
                      <http://dbpedia.org/resource/Future_tense> 0.487"

(c)
SEG_1_TEXT: "iPhone, MySpace, Facebook in 2025"
SEG_1_TEXT_URI: "<http://dbpedia.org/resource/IPhone> 1.0 0 <http://dbpe
          dia.org/resource/MySpace> 1.0 1 <http://dbpedia.org/resource/
          Facebook> 1.0 2 "
SEG_1_TIME: "121000"
```

Fig. 5. The indexing scheme extended with URIs, ranks, and text positions. Note, that the resource "Future_tense" has a lower rank than "Future" because the disambiguation algorithm considers "Future" more related to the resource.

To index the new URI fields properly, only the *solr.WhitespaceTokenizer* should be used. This is the reason, why new URI fields have to be created for origin fields with natural language values. In that case, usually other tokenizer and normalizers are used, e.g. word stemming. Since the user tag fields are only using the whitespace tokenizer, the URIs can be appended to the tag field values instead of creating a new field.

Querying the Semantic Data Besides storing the data in an index structure the querying for an entity (URI) can be done by simply querying on the additional URI fields instead on the origin text fields. The result comprises all documents containing the requested URI. The document rank in the result is based on the Lucene document score. To map the user's query string to an entity an auto-suggestion widget can be used to support the user to disambiguate the query string by herself (c.f. Fig 6). The auto-suggestion list for URIs is generated from the entities labels, names, and titles. If the user wants to search for more than one entity at one time, she can select more entities simply by continuing typing. This would provoke another suggestion list for the new query term.

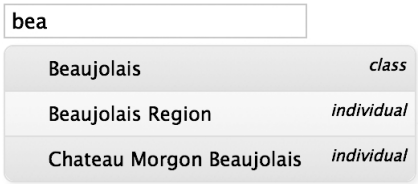

Fig. 6. Automated suggestion of entities while typing the user query. When the user selects an entity a query for its URI is issued, e.g. for "Beaujolais Region": `http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#BeaujolaisRegion` is issued. If the user does not select an entity, a regular keyword-based search is issued.

4 Conclusion

In this paper, we showed how to transform a keyword-based video search engine to a semantic video search engine. We extended the existing Yovisto keyword/tag index by semantic information, which we determined by mapping the video content to semantic entities. Furthermore, we have explained how to store the newly-acquired semantic information in an efficient state-of-the-art search index without modifying its implementation or using special Semantic Web technologies. Finally, we provided the user a semantic search engine, that enables to search for LOD entities in the entire video repository of Yovisto without the typical shortcomings on keyword-based search.

The mapping process of keywords respectively user tags to DBpedia entities is a core problem in this research field and the challenging details of our approach are still under further development. To estimate how a user can be supported by providing a fully-fledged semantic search instead of a keyword-based search future work will in first place include a comparative evaluation of both search types. An evaluation of the entity mapping method with state-of-the-art benchmarks will be addressed in future work.

References

1. Berners-Lee, T.: Linked Data. World wide web design issues (July 2006), http://www.w3.org/DesignIssues/LinkedData.html
2. Bizer, C., Heath, T., Idehen, K., Berners-Lee, T.: Linked data on the web. In: Proc. of the 17th Int. Conf. on World Wide Web. pp. 1265–1266. ACM (2008)
3. Carpineto, C., de Mori, R., Romano, G., Bigi, B.: An information-theoretic approach to automatic query expansion. ACM Trans. Inf. Syst. 19(1), 1–27 (2001)
4. Christel, M.G.: Supporting video library exploratory search: when storyboards are not enough. In: Proc. of the Int. Conf. on Content-based image and video retrieval. pp. 447–456. ACM, New York, NY, USA (2008)
5. H. Sack and J. Waitelonis: Integrating Social Tagging and Document Annotation for Content-Based Search in Multimedia Data. In: Proc. of the 1st Semantic Authoring and Annotation Workshop. Athens (GA), USA (2006)
6. Kleb, J., Abecker, A.: Entity reference resolution via spreading activation on rdf-graphs. The Semantic Web: Research and Applications pp. 152–166 (2010)
7. Marchionini, G.: Exploratory search: from finding to understanding. Commun. ACM 49(4), 41–46 (2006)
8. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com: a document-oriented lookup index for open linked data. IJMSO 3(1), 37–52 (2008)
9. Pilz, A., Paa, G.: Named Entity Resolution Using Automatically Extracted Semantic Information. In: Workshop on Knowledge Discovery, Data Mining, and Machine Learning. pp. 84–91 (2009)
10. Porter, M.F.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
11. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C (January 2008)
12. Repp, S., Waitelonis, J., Sack, H., Meinel, C.: Segmentation and Annotation of Audiovisual Recordings based on Automated Speech Recognition. In: in Proc. of 8th Int. Conf. on Intelligent Data Engineering and Automated Learning (2007)
13. Richert, M., Quasthoff, U., Hallensteinsdottir, E., Biemann, C.: Exploiting the Leipzig Corpora Collection. In: Proceedings of the IS-LTC 2006. Ljubliana, Slovenia (2006), http://wortschatz.uni-leipzig.de/
14. Schraefel, Wilson, M., Russell, A., Smith, D.A.: mSpace: improving information access to multimedia domains with multimodal exploratory search. Commun. ACM 49(4), 47–49 (April 2006)
15. Snoek, C., Sande, K.v.d., Rooij, O.d., Huurnink, B., Gemert, J.v., Uijlings, J., He, J., Li, X., Everts, I., Nedovic, V., Liempt, M.v., Balen, R.v., Yan, F., Tahir, M., Mikolajczyk, K., Kittler, J., Rijke, M.d., Geusebroek, J., Gevers, T., Worring, M., Smeulders, A., Koelma, D.: The MediaMill TRECVID 2008 semantic video search engine. National Institute of Standards and Technology (NIST) (2009)
16. Stoermer, H., Rassadko, N.: Results of OKKAM Feature based Entity Matching Algorithm for Instance Matching Contest of OAEI 2009. In: OM (2009)
17. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the Open Linked Data. The Semantic Web pp. 552–565 (2008)
18. Waitelonis, J., Sack, H., Kramer, Z., Hercher, J.: Semantically Enabled Exploratory Video Search. In: Proc. of Semantic Search Workshop at the 19th Int. World Wide Web Conference. Raleigh, NC, USA (2010)
19. Waitelonis, J., Sack, H.: Augmenting Video Search with Linked Open Data. In: Proc. of Int. Conf. on Semantic Systems 2009 (2009)

20. Waitelonis, J., Sack, H.: Towards Exploratory Video Search Using Linked Data. In: Proc. of the 11th IEEE Int. Symp. on Multimedia. pp. 540–545. IEEE Computer Society, Washington, DC, USA (2009)