ROXXI: Reviving witness dOcuments to eXplore eXtracted Information

Shady Elbassuoni
Max-Planck Institute for Informatics
Saarbrücken, Germany
elbass@mpi-inf.mpg.de

Katja Hose
Max-Planck Institute for Informatics
Saarbrücken, Germany
hose@mpi-inf.mpg.de

Steffen Metzger
Max-Planck Institute for Informatics
Saarbrücken, Germany
smetzger@mpi-inf.mpg.de

Ralf Schenkel
Saarland University and MPI for Informatics
Saarbrücken, Germany
schenkel@mmci.uni-saarland.de

ABSTRACT

In recent years, there has been considerable research on information extraction and constructing RDF knowledge bases. In general, the goal is to extract all relevant information from a corpus of documents, store it into an ontology, and answer future queries based only on the created knowledge base. Thus, the original documents become dispensable. On the one hand, an ontology is a convenient and non-redundant structured source of information, based on which specific queries can be answered efficiently. On the other hand, many users doubt the correctness of facts and ontology subgraphs presented to them as query results without proof. Instead, users often wish to verify the obtained facts or subgraphs by reading about them in context, i.e., in a document relating the facts and providing background information. In this demo, we present ROXXI, a system operating on top of an existing knowledge base and reviving the abandoned witness documents. In doing so, it goes the opposite way of information extraction approaches – starting with ontological facts and tracing their way back to the documents they were extracted from. ROXXI offers interfaces for expert users (SPARQL) as well as for non-experts (ontology browser) and provides a ranked list of documents each associated with a content snippet highlighting the queried facts in context. At the demonstration site, we will show the advantages of this novel approach towards document retrieval and illustrate the benefits of reviving the documents that information extraction approaches neglect.

1. INTRODUCTION

The success of knowledge-sharing communities like Wikipedia, IMDB, etc. and the advances in automatic information extraction from textual and Web sources have made it possible to build large "knowledge repositories" such as DBpedia [1], Freebase [5], and YAGO [8]. Information is extracted as facts from structured parts such as infoboxes in Wikipedia but also from unstructured text using textual patterns [3, 4, 9]. A textual pattern is a sequence of words in natural language text that expresses a certain relation between entities, e.g., "Quentin Tarantino was one of the producers of 'From Dusk till Dawn'" matches the pattern "`X` *was one of the producers of* `Y`" and corresponds to the abstract fact `<Quentin_Tarantino produced From_Dusk_till_Dawn>`, where `Quentin_Tarantino` and `From_Dusk_till_Dawn` refer to entities and `produced` is a binary predicate representing a relation.

Knowledge bases enable users to search for explicit knowledge by exploiting semantic structures based on entities and relations; a task often impossible to achieve with standard keyword-based search engines. As an example, consider the following query: "movies created and acted in by the same person". First, it is nearly impossible to formulate this query using keywords only. And second, it might be necessary to combine information from different pages to answer it. In fact, posing this query to any of the major search engines returns no relevant answer in the first ten results. Such queries, however, can be answered precisely using knowledge bases and referring to binary predicates. For example, the following two facts would answer the above query:
`<Quentin_Tarantino created From_Dusk_till_Dawn>` and
`<Quentin_Tarantino actedIn From_Dusk_till_Dawn>`.

Still, this approach is limited in several ways. First, a user cannot look up information that has not been extracted before, i.e., available knowledge is restricted to predicates and patterns that have been identified as being relevant beforehand. Second, users tend to doubt the correctness of facts obtained from a knowledge base and presented out of context, i.e., users prefer to verify presented facts by accessing the documents they were extracted from. Moreover, users often like to learn more details and background knowledge about extracted facts.

Knowledge bases do not only lack background information about facts but are also restricted to the most common relations and patterns expressible with binary relations. For example, when looking for detailed information about the movie 'From Dusk till Dawn', we can query a knowledge base and find the information that Quentin Tarantino wrote, produced, and acted in this movie. But we cannot find information such as the movie plot, tag lines, or user ratings, because they cannot be expressed with binary relations. Another piece of information that we will not find in a knowledge base, even though available in the original documents, is that Quentin Tarantino was originally set to direct the movie, but in the end decided against it so that he could focus more on his tasks as actor and screenplay writer; knowledge bases do not consider "potential" facts and explanations.

Proceedings of the VLDB Endowment, Vol. 3, No. 2
Copyright 2010 VLDB Endowment 2150-8097/10/09... $ 10.00.

With ROXXI we present a system that enables information exploration on top of RDF knowledge bases. It does not rely on entities only [2] but instead combines the benefits of document retrieval and structured search in knowledge bases by relating automatically extracted information (facts) to its witnesses, i.e., the documents the information originates from. The information about the connections between facts and documents can either be collected during the information extraction process or added later on. In addition, ROXXI also preserves information about which textual pattern was involved in a particular fact occurrence in a witness document.

ROXXI presents the extracted information in context giving the user the opportunity to verify extracted facts as well as to relate and connect different pieces of information. Furthermore, this provides the user with additional information that cannot be represented in a knowledge base, information that the user was not sure she was looking for when issuing the query, or information for which no extraction rules in the form of patterns and relations exist.

In particular, ROXXI's main contributions are:

- fact-based document retrieval – retrieving documents based on a set of facts,
- document ranking utilizing fact occurrences – applying a ranking model based on statistical language models,
- content snippet generation – generating meaningful content snippets for each document in the result set, and
- providing context for extracted facts – a document browser allows the user to verify extracted facts and to read about non-extracted background information.

Section 2 provides more details on using the system while Section 3 illustrates the system architecture and implementation issues. Finally, Section 4 outlines what we will present at the demonstration site.

2. WORKING WITH ROXXI

As mentioned in the introduction, ROXXI operates on top of an RDF knowledge base. RDF knowledge bases consist of facts in the form of subject-property-object (SPO) triples of the Semantic Web data model RDF. For example, the fact `<Quentin_Tarantino actedIn From_Dusk_Till_Dawn>` is an RDF triple with subject `Quentin_Tarantino`, predicate `actedIn` and object `From_Dusk_till_Dawn`. An RDF knowledge base can be conceptually viewed as a large graph with nodes corresponding to entities (subjects and objects) and edges denoting relationships or predicates, which we refer to as an RDF graph.

To start information exploration with ROXXI, a user first identifies a set of facts she is interested in, which we refer to as an RDF subgraph. Note that this subgraph does *not* need to be connected. ROXXI offers two ways to provide RDF subgraphs, one for users acquainted with SPARQL and the other for unexperienced users. The first allows users to retrieve RDF subgraphs via SPARQL queries whereas the second allows them to search for entities in the knowledge base. A SPARQL query and a corresponding result for the example about Quentin Tarantino writing and acting in the same movie are shown as step 1b in Figure 1. Step 1a in Figure 1 shows the results when performing an entity search for "Tarantino". Once the user clicks on either "send result to ROXXI" below a result in the SPARQL interface or on an entity, the *Exploration Page* opens.

The *Exploration Page* offers two interconnected ways to learn more about a certain RDF triple or subgraph, a graphical *Ontology Browser* and the *Witness List*, the latter being a textual interface that lists the witnesses for a chosen RDF subgraph.

The *Ontology Browser* allows users to easily navigate through the RDF graph by selecting or deselecting facts in the underlying graph and thus defining the RDF subgraph to be explored. The *Witness List* shows a ranked list of witnesses containing (some of) the knowledge expressed by the chosen subgraph. For each listed document, a snippet from the document's content is shown. The snippet shows occurrences for some of the facts in the RDF subgraph inside the document. Step 2 in Figure 1 illustrates this for the RDF subgraph representing the facts that Quentin Tarantino wrote and acted in From Dusk Till Dawn (see highlighted text in *Witness List* frame in Figure 1).

Once witnesses are listed, the user can view a local copy of the document taken at extraction time by clicking on its title. Alternatively, the current version of the document can be viewed by following the URI shown below the snippet. When exploring the local copy, word combinations which the facts of interest have been extracted from are highlighted using the same colors as used in the *Witness List*. Furthermore, word combinations associated with any other fact are highlighted in a different color (green) as shown in step 3 in Figure 1.

In addition to exploring witnesses from which facts were extracted, ROXXI enables users to find complementary witness documents that were not identified during information extraction. When exploring an RDF subgraph, a keyword search for further witnesses can be issued. In particular, for each fact independent keyword queries are generated based on the textual patterns used to extract the corresponding fact. For example, consider the pattern "`X` *was one of the producers of* `Y`", which is used to extract facts for the relation *produced*. This pattern can be used to generate a keyword query to retrieve additional witnesses for the fact `<Quentin_Tarantino produced From_Dusk_Till_Dawn>`. Precisely, the issued query would be "Quentin Tarantino was one of the producers of From Dusk Till Dawn". By using each pattern known to the extraction system in this fashion and aggregating the retrieved results, ROXXI can spare users from formulating these (or similar) queries manually. This is also a powerful tool to expand or augment the set of source documents for the facts present in the knowledge base.

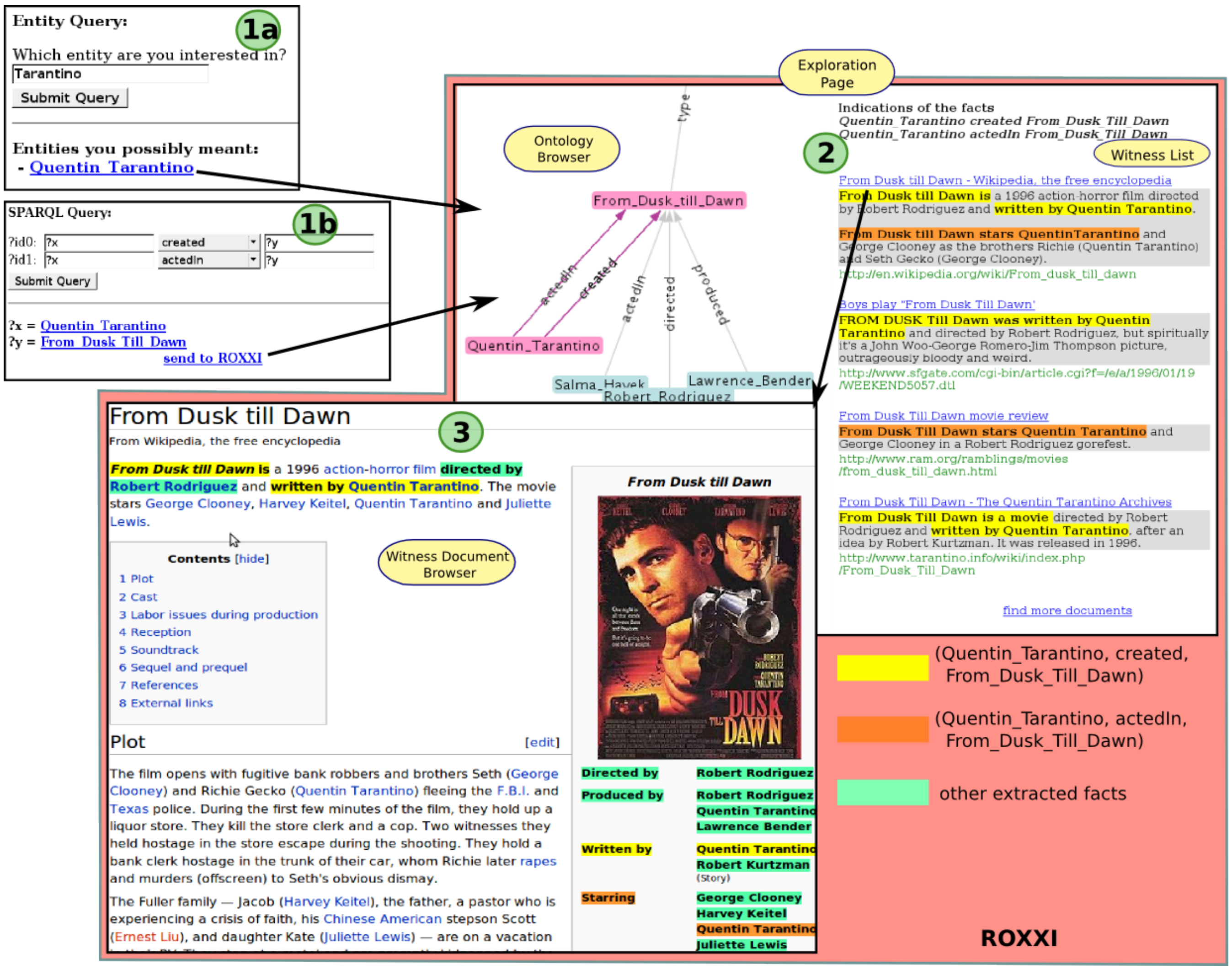
3. ARCHITECTURE

ROXXI's system architecture is depicted in Figure 2. It consists of 3 main components: the *Data Manager*, the *Query Engine* and a *User Interface*. We now describe each component in more detail in the following subsections.

3.1 Data Manager

The *Data Manager* manages the data ROXXI operates on. This data consists of the knowledge base stored as an *RDF Database*. As mentioned earlier, facts were automatically extracted from Web sources by exploiting different knowledge acquisition and information extraction techniques. Most of these information extraction systems are based on patterns (e.g. SOFIE [9]). These patterns are either textual patterns that usually appear in text, like "`X` *was one of the producers of* `Y`" or mappings for structured data, e.g. property tables in Wikipedia. A pattern is associated with a confidence value reflecting its accuracy. For example, Wikipedia property tables should have a higher confidence value than textual patterns.

We store the patterns and their confidence values as *Extraction Metadata*. In addition, for each one of these patterns, we store the set of witness documents it occurred in and the start and end position of the pattern in the document. This is later used to retrieve the witnesses for a given RDF subgraph, which we annotate with the corresponding pattern occurrences before presenting it to the user.



Entity Query:

Which entity are you interested in?

Tarantino

Submit Query

Entities you possibly meant:

- Quentin Tarantino

SPARQL Query:

?id0:	?x	created	?y
?id1:	?x	actedIn	?y

Submit Query

?x = Quentin Tarantino
?y = From Dusk Till Dawn
send to ROXXI

Indications of the facts
Quentin_Tarantino created From_Dusk_Till_Dawn
Quentin_Tarantino actedIn From_Dusk_Till_Dawn

From Dusk till Dawn - Wikipedia, the free encyclopedia
From Dusk till Dawn is a 1996 action-horror film directed by Robert Rodriguez and **written by Quentin Tarantino**.
From Dusk till Dawn stars QuentinTarantino and George Clooney as the brothers Richie (Quentin Tarantino) and Seth Gecko (George Clooney).
http://en.wikipedia.org/wiki/From_dusk_till_dawn

Boys play "From Dusk Till Dawn'
FROM DUSK Till Dawn was written by Quentin Tarantino and directed by Robert Rodriguez, but spiritually it's a John Woo-George Romero-Jim Thompson picture, outrageously bloody and weird.
http://www.sfgate.com/cgi-bin/article.cgi?f=/e/a/1996/01/19/WEEKEND5057.dtl

From Dusk Till Dawn movie review
From Dusk Till Dawn stars Quentin Tarantino and George Clooney in a Robert Rodriguez gorefest.
http://www.ram.org/ramblings/movies/from_dusk_till_dawn.html

From Dusk Till Dawn - The Quentin Tarantino Archives
From Dusk Till Dawn is a movie directed by Robert Rodriguez and **written by Quentin Tarantino**, after an idea by Robert Kurtzman. It was released in 1996.
http://www.tarantino.info/wiki/index.php/From_Dusk_Till_Dawn

find more documents

🟨	(Quentin_Tarantino, created, From_Dusk_Till_Dawn)
🟧	(Quentin_Tarantino, actedIn, From_Dusk_Till_Dawn)
🟩	other extracted facts

From Dusk till Dawn

From Wikipedia, the free encyclopedia

***From Dusk till Dawn* is** a 1996 action-horror film **directed by Robert Rodriguez** and **written by Quentin Tarantino**. The movie stars George Clooney, Harvey Keitel, Quentin Tarantino and Juliette Lewis.

Contents [hide]

1 Plot
2 Cast
3 Labor issues during production
4 Reception
5 Soundtrack
6 Sequel and prequel
7 References
8 External links

Plot [edit]

The film opens with fugitive bank robbers and brothers Seth (George Clooney) and Richie Gecko (Quentin Tarantino) fleeing the F.B.I. and Texas police. During the first few minutes of the film, they hold up a liquor store. They kill the store clerk and a cop. Two witnesses they held hostage in the store escape during the shooting. They hold a bank clerk hostage in the trunk of their car, whom Richie later rapes and murders (offscreen) to Seth's obvious dismay.

The Fuller family — Jacob (Harvey Keitel), the father, a pastor who is experiencing a crisis of faith, his Chinese American stepson Scott (Ernest Liu), and daughter Kate (Juliette Lewis) — are on a vacation

From Dusk till Dawn

Directed by	**Robert Rodriguez**
Produced by	**Robert Rodriguez** **Quentin Tarantino** **Lawrence Bender**
Written by	**Quentin Tarantino** **Robert Kurtzman** (Story)
Starring	**George Clooney** **Harvey Keitel** **Quentin Tarantino** **Juliette Lewis**

Figure 1: Example Workflow

3.2 Query Engine

The query engine consists of 4 subcomponents, which we describe separately in the following.

Graph Explorer. The graph explorer takes an RDF subgraph as input. An RDF subgraph can either be an entity, an RDF triple, or a set of RDF triples. The graph explorer expands this subgraph by retrieving additional neighboring triples from the RDF database and returns an extended graph that can be explored by the user using the *Ontology Browser*.

SPARQL Query Engine. It takes a SPARQL query as input and returns a ranked list of results matching the given SPARQL query. We rely on the NAGA search engine [6] as the SPARQL query engine in our system.

Witness Retrieval Engine. It is responsible for retrieving and ranking the witnesses for a given RDF subgraph. Our ranking model is based on statistical language models [7]. The witnesses are ranked based on the probability of being the source of the given RDF subgraph $g = \{t_1, t_2, ..., t_n\}$, which we denote as $P(d|g)$. Applying Bayes' rule and ignoring $P(g)$ as it is witness independent, we have:

$$P(d|g) \propto P(g|d)P(d)$$

$P(d)$ is a prior probability that a witness d is the source of any RDF subgraph. This probability can be estimated in various ways, and in our case we estimate it using the static authority of the page or *pagerank*.

The probability $P(g|d)$ is the probability that the given RDF subgraph $g = \{t_1, t_2, ..., t_n\}$ was extracted from witness d. We assume independence between the triples or facts in g for computational tractability (in-line with most traditional keyword ranking models). In addition, we apply smoothing with the collection background model (Col) to avoid overfitting. Thus,

$$P(g|d) = \prod_{i=1}^{n} [\alpha P(t_i|d) + (1-\alpha)P(t_i|Col)]$$

Recall that each triple t_i was extracted from witnesses using a set of patterns $r_1, r_2, ..., r_m$. Each pattern r_j is associated with a certain confidence value $conf(r_j)$. Thus, the probabilities $P(t_i|d)$ and $P(t_i|col)$ are estimated as follows

$$P(t_i|x) = \Sigma_{j=1}^{m} p(t_i|r_j)P(r_j|x)$$

The first component $P(t_i|r_j)$ is the probability of extracting triple t_i using pattern r_j. It is estimated using the confidence value $conf(r_j)$. The second component $P(r_j|x)$ is the probability of pattern r_j occurring in x where $x \in \{d, Col\}$. It is estimated using

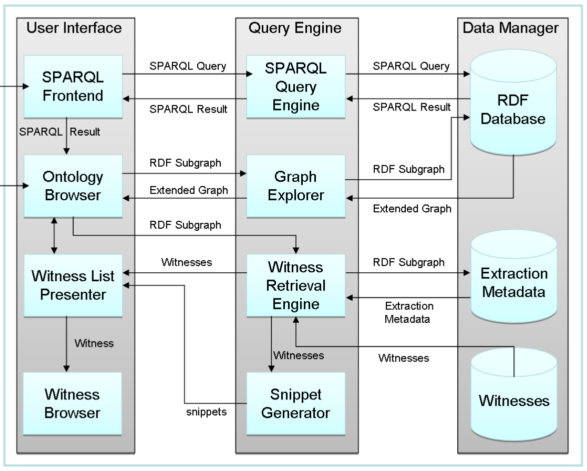


Figure 2: Architecture

a Maximum-Likelihood estimator as follows:

$$P(r_j|x) = \frac{c(r_j;x)}{\Sigma_r c(r;x)}$$

where $c(r_j;x)$ denotes the frequency of pattern r_j in x.

Snippet Generator. The snippet generator is responsible for generating a snippet for each retrieved witness. Similar to most state-of-the-art search engines, we generate a query-biased snippet. The snippet contains (some of) the pattern occurrences in the witnesses retrieved by the *Witness Retrieval Engine*. For snippet generation in ROXXI, we use the simple technique of presenting the sentences that highly match the user query from [10]. Adapting the aforementioned technique to work with RDF subgraphs and presenting patterns instead of sentences is similar in spirit to our witness retrieval model described earlier, and we omit it due to space limitations.

3.3 User Interface

The user interface contains facilities for both the casual as well as the expert user. The expert users can utilize the *SPARQL Frontend* to issue SPARQL queries and retrieve matching RDF subgraphs. On the other hand, the less advanced users can navigate directly to the exploration page described in Section 2 by performing entity search.

The knowledge base can be browsed using the *Ontology Browser* which renders a hyperbolic visualization of the RDF graph. Our ontology browser is based on the Prefuse visualization tool [1]. Furthermore, the user can navigate through the RDF graph as desired and select or deselect facts to retrieve witnesses.

The *Witness List Presenter* displays a set of ranked witnesses for the currently selected RDF subgraph in the ontology browser. Finally, the *Witness Browser* utilizes a Web Browser plug-in to highlight any extracted facts in the currently viewed witness.

4. DEMO OUTLINE

We will demonstrate the features of ROXXI using a knowledge base with facts from the movie domain, based on information from YAGO [8] and IMDB. Additional facts will be extracted using the SOFIE extraction framework [9] from movie-related documents in Wikipedia and in the ClueWeb09 dataset[2]. This provides a huge knowledge graph that can be explored by attendees. To further demonstrate the document ranking capabilities of ROXXI, we will preselect a set of entities and facts, for which we will retrieve further documents from the Web using ROXXI's query generation feature (Section 2); these documents will then be run through SOFIE to find fact occurrences in them.

At the demonstration site, we will give conference attendees the opportunity to experience first-hand and interactively ROXXI's benefits. We will show attendees how to use ROXXI to explore all the interesting details and background stories about their favorite movies. Furthermore, attendees are given the opportunity to explore documents relating interesting sets of facts and to learn how these facts relate. Finally, attendees can also observe how the ranked list of relevant documents changes when a single fact is added to or removed from the input.

The demo will run on a notebook with all data stored locally. Internet connectivity is needed only to demonstrate ROXXI's query generation feature.

5. ACKNOWLEDGMENTS

This work was partially supported by the BMBF project 'WisNetGrid', FKZ 01 IG 09008E, and by the Cluster of Excellence „Multimodal Computing and Interaction“ (funded by the German Science Foundation).

6. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In *ISWC/ASWC*, pages 11–15, 2007.
[2] F. Brauer, W. Barczynski, G. Hackenbroich, M. Schramm, A. Mocan, and F. Förster. RankIE: document retrieval on ranked entity graphs. *Proc. VLDB Endow.*, 2(2):1578–1581, 2009.
[3] S. Brin. Extracting Patterns and Relations from the World Wide Web. In *WebDB*, pages 172–183, 1999.
[4] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-Scale Information Extraction in KnowItAll: (Preliminary Results). In *WWW*, pages 100–110, 2004.
[5] Freebase: A social database about things you know and love. www.w3.org/RDF/.
[6] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. NAGA: Searching and Ranking Knowledge. In *ICDE*, pages 953–962, 2008.
[7] X. Liu and W. B. Croft. Statistical language modeling for information retrieval. *The Annual Review of Information Science and Technology*, 39:3–31, 2004.
[8] F. Suchanek, G. Kasneci, and G. Weikum. YAGO – A Large Ontology from Wikipedia and WordNet. *Journal of Web Semantics*, 6(3):203–217, 2008.
[9] F. M. Suchanek, M. Sozio, and G. Weikum. SOFIE: A Self-Organizing Framework for Information Extraction. In *WWW*, pages 631–640, 2009.
[10] R. W. White, I. Ruthven, and J. M. Jose. Finding relevant documents using top ranking sentences: an evaluation of two alternative schemes. In *SIGIR*, pages 57–64, 2002.

[1]http://prefuse.org

[2]`http://boston.lti.cs.cmu.edu/Data/clueweb09/`