NAGA: Harvesting, Searching and Ranking Knowledge

Gjergji Kasneci Fabian M. Suchanek Georgiana Ifrim
Shady Elbassuoni Maya Ramanath Gerhard Weikum
Max-Planck Institute for Informatics
{kasneci,suchanek,ifrim,elbass,ramanath,weikum}@mpi-inf.mpg.de

ABSTRACT

The presence of encyclopedic Web sources, such as Wikipedia, the Internet Movie Database (IMDB), World Factbook, etc. calls for new querying techniques that are simple and yet more expressive than those provided by standard keyword-based search engines. Searching for explicit knowledge needs to consider inherent semantic structures involving entities and relationships.

In this demonstration proposal, we describe a semantic search system named NAGA. NAGA operates on a knowledge graph, which contains millions of entities and relationships derived from various encyclopedic Web sources, such as the ones above. NAGA's graph-based query language is geared towards expressing queries with additional semantic information. Its scoring model is based on the principles of generative language models, and formalizes several desiderata such as confidence, informativeness and compactness of answers.

We propose a demonstration of NAGA which will allow users to browse the knowledge base through a user interface, enter queries in NAGA's query language and tune the ranking parameters to test various ranking aspects.

1. INTRODUCTION

The presence of clean and well organized knowledge domains such as Wikipedia, WordNet, World Factbook, IMDB, etc. strengthens our belief that future search technology will have to deal with queries representing more intricate user needs for explicit knowledge. In order to satisfy these needs, novel search techniques need to go beyond simple keyword-based search and exploit semantic structures based on entities and relationships. As a concrete example, consider the following query: *"Which politicians are also scientists?"* First, it is nearly impossible to formulate this query using only keywords. And second, the answer to this query involves extracting entities and relations from different pages. In fact, posing this query to Google returns no relevant answer in the first ten results. A knowledge base that could understand binary predicates, such as <Angela_Merkel ISA politician> and <Angela_Merkel HASDEGREE Doctor_of_Physics> would go a long way in addressing information needs such as the above. Combined with an appropriate query language and an effective scoring strategy, users would be able to express sophisticated information needs and retrieve precise information in return.

Our semantic search system, NAGA [5], builds on a large knowledge base of facts derived from various encyclopedic Web sources. As of now, this knowledge base consists of approximately 14 million facts and understands 90 predefined relationships such as ISA, BORNINYEAR, ESTABLISHEDINYEAR, HASWONPRIZE, LOCATEDIN, POLITICIANOF, MEANS, ACTEDIN, DISCOVEREDINYEAR, DISCOVEREDBY, ISCITIZENOF, etc. NAGA's graph-based query language allows users to pose precise queries with semantic information at the entity level. The query to retrieve politicians who are also scientists can be expressed as depicted in Figure 1.

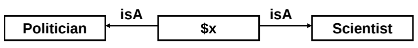

Figure 1: Example Query

For a given query, NAGA may retrieve multiple answers. In order to rank these answers, our scoring mechanism applies the principles of generative language models (already well studied for document-level information retrieval [6]) to the unexplored setting of weighted, labeled graphs. Our scoring model is extensible, tunable, and formalizes several intuitive notions such as compactness, informativeness and confidence of results.

Our approach is in line with recent work on information extraction and entity search [3, 7], graph based keyword search [2] and with RDF querying [1]. However NAGA goes beyond this prior and ongoing work by capturing not just entities but also semantic relations, by an expressive query language beyond keywords and by its powerful ranking model. For a detailed discussion of related work, see [5].

In the rest of this proposal, in Section 2, we describe the main features of NAGA, and list the specific components that we will demonstrate in Section 3.

2. DESCRIPTION OF NAGA

NAGA's system architecture is depicted in Figure 2. The *backend* consists of the knowledge base, YAGO [8], which is updated with facts extracted from Web sources by exploiting different knowledge acquisition and information extraction techniques. The query processing and ranking component takes the user query as input, processes it and ranks the results. The *user interface* contains facilities for both the casual as well as the expert user. The knowledge base can be browsed using a browser which renders a hyperbolic visualization of the knowledge graph. Furthermore, the query results can be highlighted in the knowledge graph and browsed as desired by the user. The expert user can additionally provide her own ranking parameters for the scoring model and issue sophisticated queries based on regular expressions over relation labels. In the rest of this section, we describe NAGA's knowledge base, query language, and the scoring model.

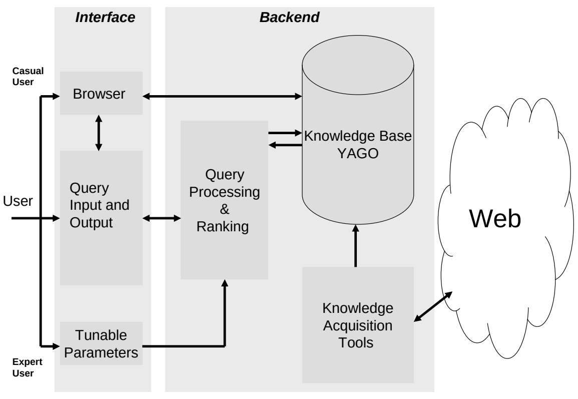

Figure 2: NAGA: System architecture

2.1 The Knowledge Base

NAGA's knowledge base, YAGO [8], contains *facts* derived from a number of semi-structured and unstructured Web sources such as Wikipedia, IMDB, etc. A fact is a triple of the form <*entity*, *relationship*, *entity*>. Examples of the facts known to YAGO are: <Nile LOCATEDIN Egypt>, <Mostly_Harmless WRITTENINYEAR 1992>, etc. The named entities occurring in such facts are attached to the corresponding WordNet concepts. For example, the named entity Nile is attached to the WordNet concept river. Each fact is assigned a *confidence* value denoting a combination of the authority of the page from which the fact was extracted as well as the extraction confidence as returned by the underlying extraction technique. A fragment of YAGO is depicted in Figure 3.

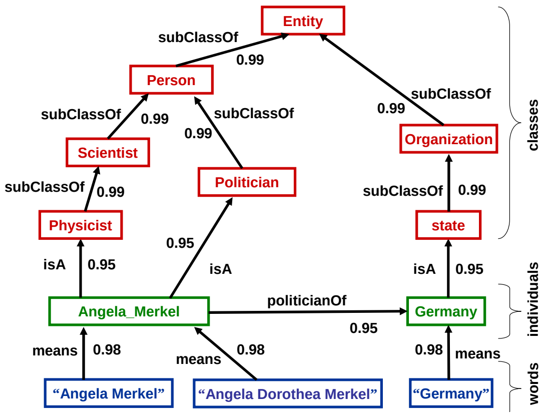

Figure 3: Example fragment of the knowledge graph

YAGO consists of 3 kinds of entities: A *word* refers to an entity via the MEANS relation. For example, the string "Angela Dorothea Merkel" may be used to refer to the entity Angela_Merkel. This is reflected by the MEANS relation as <"Angela Dorothea Merkel" MEANS Angela_Merkel>. Different words may refer to the same entity (synonymy) and the same word may refer to different entities (ambiguity).

An *individual* is a real-world object. For example, Angela_Merkel or Germany are individuals.

A *class* is an entity that represents a group of similar individuals. For example, the class politician represents the group of all politicians. We use the TYPE relation to state that an individual belongs to a class, and the SUBCLASSOF relation to state that a class is subsumed by another class: <Angela_Merkel TYPE Chancellor>, <Chancellor SUBCLASSOF Politician>.

Although we used YAGO, the query model and the ranking model of NAGA can be applied to any knowledge base that exhibits the above structure. In the following, we treat the knowledge base as a black box and refer the interested reader to [8].

2.2 The Query Language

Our graph-based query language is designed to support several different types of querying. We derive much of the syntax and some of the semantics of our query language from SPARQL [4], but extend it with several useful features. We present examples below to show the different styles of querying.

Discovery Queries are queries that supply the user with pieces of missing information. For example, Figure 4 asks for physicists who were born in the same year as Max Planck. NAGA attempts to bind the variables $x and $z by finding a subgraph in the knowledge graph that matches the query graph. Note that there are multiple answers to this query and NAGA returns a ranked list of answers.

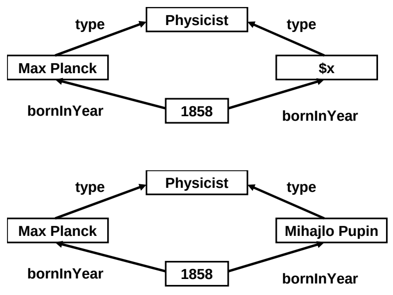

Figure 4: A discovery query with one answer

Regular Expression Queries enable users to specify more flexible matches by allowing regular expressions over relationship labels on query edges. Figure 5 shows two such queries. We use ISA as a shortcut for the regular expression TYPE SUBCLASSOF*. The first query asks for rivers located in Africa. Here, an answer such as <Nile TYPE river>, <Nile LOCATEDIN Egypt>, <Egypt LOCATEDIN Africa> is a valid match. The second query asks for scientists whose first name or last name is Liu.

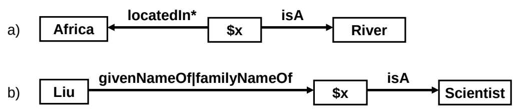

Figure 5: Two regular expression queries

Relatedness Queries discover "broad" connections between pieces of information. For example, we could ask the question *"How are Margaret Thatcher and Indira Gandhi related?"* (expressed as <Margaret_Thatcher CONNECT Indira_Gandhi>). There are several possible answers to this query – including the trivial answer that "they are both people", more informative answers such as "they were both prime-ministers" as well as more complex answers such as "Margaret Thatcher was the prime-minister of England. Indira Gandhi was the prime-minister of India. India and England are both English-speaking countries".

2.3 The Scoring Model

Our scoring model scores answer graphs based on important aspects such as confidence, informativeness and compactness, which are integrated into a unified framework. Our approach is inspired by existing work on language models for information retrieval on document collections, and is adapted and extended to the new domain of labeled and weighted graphs. Here, we describe the above aspects briefly. For more details, we refer the reader to [5].

Confidence: Answers containing facts of high confidence should be ranked higher.

Informativeness: The informativeness of an answer is dependent on the query and is intuitively made clear with the following example. When asking the query <Albert_Einstein ISA $z> the answer <Albert_Einstein ISA physicist> should rank higher than the answer <Albert_Einstein ISA politician>, because Einstein was a physicist to a larger extent than he was a politician. Similarly, for a query such as <$y ISA physicist>, the answers about world class physicists should rank higher than those about hobby physicists.

Compactness: Tight connections between query entities are preferred to loose ones. For example, for the query *"How are Einstein and Bohr related?"* the answer about both having won the Nobel Prize should rank higher than the answer that Tom Cruise connects Einstein and Bohr by being a vegetarian like Einstein, and by being born the year Bohr died.

3. DEMONSTRATION

We have developed NAGA as a stand-alone application as well as a Web-based application using Java. The backend consists of the knowledge base of facts which is stored in an Oracle database. We will demonstrate the following components of our system.

Browsing the Knowledge Base: Our knowledge base consists of millions of facts. Consequently, an important facility that we provide is the exploration of this knowledge base by means of hyperbolic browsing. The user can zoom-in on any entity that occurs in the result set by clicking on it, or by directly typing the name of that entity. Hence, the user can explore the knowledge graph by zooming in on specific parts of interest.

Query Processing: Users can enter queries of their choice in text format through our user interface. The retrieved results are ranked based on our scoring model, and displayed in text format. Users can click on specific entities or facts in the retrieved result graphs and browse their neighborhood in the knowledge base. A snapshot of results for the query *"Which politicians are also scientists?"* in text format is shown in Figure 6.

```
Score: 5.6887E-7
<Benjamin_Franklin TYPE American_scientists>
<American_scientists SUBCLASSOF scientist>
<"scientist" MEANS scientist>
<Benjamin_Franklin TYPE Massachusetts_politicians>
<Massachusetts_politicians SUBCLASSOF politician>
<"politician" MEANS politician>
$X = Benjamin_Franklin
```

```
Score: 5.335E-7
<Paul_Wolfowitz TYPE American_political_scientists>
<American_political_scientists SUBCLASSOF scientist>
<"scientist" MEANS scientist>
<Paul_Wolfowitz TYPE Jewish-American_politicians>
<Jewish-American_politicians SUBCLASSOF politician>
<"politician" MEANS politician>
$X = Paul_Wolfowitz
```

Figure 6: Results for the example query

Figure 7 shows a snapshot of the first result graph (with Benjamin Franklin) highlighted with blue, dotted lines, on a subset of the knowledge base. Figure 8 shows additional facts when the user clicks on the node "American_humanists" to which Benjamin Franklin is connected.

Scoring Model: Users can tune the parameters for confidence and informativeness and see how the tradeoff between the confidence of a result and its informativeness can result in the re-ranking of answers. NAGA's text-only interface is available at `http://www.mpi-inf.mpg.de/~kasneci/naga`.

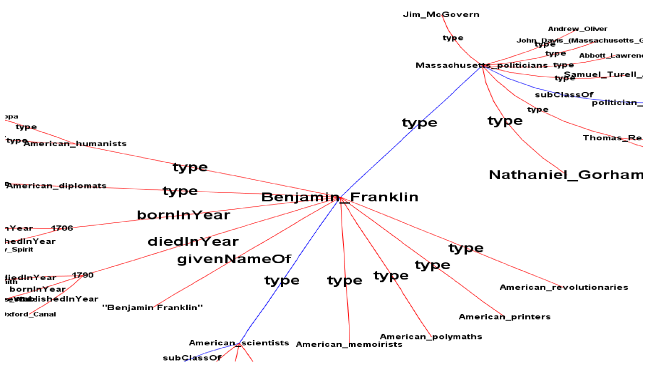

Figure 7: The first result graph for the example query

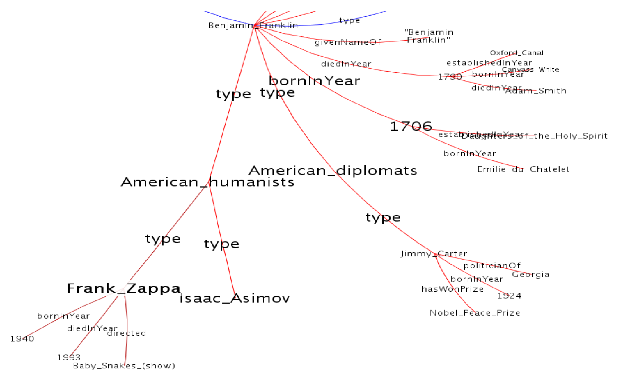

Figure 8: Exploring additional facts from the result

4. REFERENCES

[1] K. Anyanwu, A. Maduko, and A. Sheth. SPARQ2L: Towards support for subgraph extraction queries in rdf databases. In *Proc. of WWW*, 2007.

[2] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *Proc. of ICDE*, 2002.

[3] T. Cheng and K. C.-C. Chang. Entity search engine: Towards agile best-effort information integration over the web. In *CIDR*, 2007.

[4] World Wide Web Consortium. SPARQL. http://www.w3.org/TR/rdf-sparql-query/, 2005.

[5] G. Kasneci, F.M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. NAGA: Searching and Ranking Knowledge. In *Proc. of ICDE*, 2008.

[6] X. Liu and W.B. Croft. Statistical language modeling for information retrieval. In *Annual Review of Information Science and Technology 39*, 2004.

[7] Z. Nie, S. Shi Y. Ma, J.-R. Wen, and W.Y. Ma. Web object retrieval. In *Proc. of WWW*, 2007.

[8] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *Proc. of WWW*, 2007.