NAGA: Searching and Ranking Knowledge

Gjergji Kasneci, Fabian M. Suchanek, Georgiana Ifrim, Maya Ramanath, and Gerhard Weikum

MPI–I–2007–5–001 March 2007

Authors' Addresses

Gjergji Kasneci
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Fabian M. Suchanek
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Georgiana Ifrim
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Maya Ramanath
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Gerhard Weikum
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Abstract

The Web has the potential to become the world's largest knowledge base. In order to unleash this potential, the wealth of information available on the web needs to be extracted and organized. There is a need for new querying techniques that are simple yet more expressive than those provided by standard keyword-based search engines. Search for knowledge rather than Web pages needs to consider inherent semantic structures like entities (person, organization, etc.) and relationships (`isA`, `locatedIn`, etc.).

In this paper, we propose NAGA, a new semantic search engine. NAGA's knowledge base, which is organized as a graph with typed edges, consists of millions of entities and relationships automatically extracted from Web-based corpora. A query language capable of expressing keyword search for the casual user as well as graph queries with regular expressions for the expert, enables the formulation of queries with additional semantic information. We introduce a novel scoring model, based on the principles of generative language models, which formalizes several notions like confidence, informativeness and compactness and uses them to rank query results. We demonstrate NAGA's superior result quality over current search engines by conducting a comprehensive evaluation, including user assessments, for advanced queries.

Keywords

Semantic Search, Entities, Relationships, Ranking

Contents

1 Introduction

1.1 Motivation

The World Wide Web bears the potential of being the world's most comprehensive knowledge base, but we are far from exploiting this potential. The Web includes a wild mixture of valuable scientific and cultural content, news and entertainment, community opinions, advertisements, as well as spam and junk. Unfortunately, all this is coiled up into an amorphous pile of hyperlinked pages, and keyword-oriented search engines merely provide best-effort heuristics to find relevant "needles" in this humongous "haystack".

As a concrete example, suppose we want to learn about physicists who were born in the same year as Max Planck. First, it is close to impossible to formulate this query in terms of keywords. Second, the answer to this question is probably distributed across multiple pages, so that no state-of-the-art search engine will be able to find it, and third, the keywords "Max Planck" could stand for different world entities (e.g., the physicist Max Planck, the Max-Planck Society, etc.). In fact, posing this query to Google (by using the keywords "physicist born in the same year as Max Planck") yields only pages about Max Planck himself, along with pages about the Max-Planck Society.

This example highlights the need for more explicit, unifying structures for the information of the Web. For example, a knowledge base that could understand binary predicates, such as `Max_Planck isA physicist` or `Max_Planck bornInYear 1858`, would go a long way in addressing information needs such as the above. Combined with an appropriate query language and ranking strategies, users would be able to express queries with semantics and retrieve precise information in return.

There are several ways of addressing the envisioned functionality, and there are several research avenues that aim at this direction in a broader sense. Large-scale information extraction from semistructured corpora or unstructured text sources has made great progress in recent years [2], but it is not addressing the *querying* of the acquired knowledge. Graph querying such as RDF-based languages or data mining on biological networks is a direction that is gaining momentum [32], but does not consider the potential

uncertainty of the data and disregards the need for a ranking model. Finally, entity-oriented Web search and other forms of "semantic" information retrieval [11] provide ranking but have rather simple query models such as keyword search. Ranked retrieval on XML data like XQuery Full-Text [5] are more expressive but focus on trees and do not carry over to richer knowledge graphs. Our work positions itself at the confluence of these research avenues and creates added value by combining techniques from all of them and further extending this synergetic approach by various novel building blocks.

1.2 Our Approach and Contributions

In this paper, we describe NAGA, our new semantic search engine. NAGA's data model is a graph, in which the nodes represent entities and the edges represent relationships between the entities. We call an edge in the graph with its two end-nodes a *fact*. Facts are extracted from various Web-based data sources. To each fact, we attach a *confidence measure*, which reflects the authority of the source of extraction as well as the certainty of the extraction process. Furthermore, we maintain all URLs of Web pages in which a certain fact occurred. NAGA's knowledge base currently consists of 34 million facts extracted from semi-structured Web-based sources such as Wikipedia and IMDB as well as hand-crafted ontologies such as WordNet [28]. Additionally, we utilize state-of-the-art extraction tools such as LEILA[47] in order to extract facts from unstructured Web-pages containing natural language text.

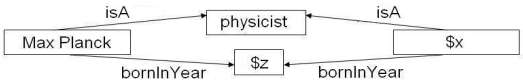

Figure 1.1: **Query**: Physicists born in the same year as Max Planck

In order to query the knowledge-graph, NAGA provides a graph-based query language. The query language allows the formulation of queries with semantic information. For example, Figure 1.1 shows how the preceding query about physicists born in the same year as Max Planck can be formulated with the explicit semantic information that "Max Planck" is a physicist. NAGA also allows more complex graph queries with regular expressions over relationships on edge labels.

NAGA returns multiple answers for some queries. In order to rank these answers, we propose a novel scoring mechanism based on the principles of generative language models for document-level information retrieval [41, 52]. Language models have been very successful in IR, but have so far been restricted to entire documents or passages as retrieval units, using generative models for bags of words or n-grams. We introduce a new scoring model for the specific and unexplored setting of weighted, labeled graphs. Our scoring

model is extensible and tunable and takes into consideration several intuitive notions such as compactness, informativeness and confidence of the results.

Our major contributions in this paper are the following:

- A graph-based knowledge representation model and tools for knowledge extraction from Web-based corpora, building on and extending our earlier work on the YAGO ontology [48]
- A novel, expressive yet concise query language for searching a Web-derived knowledge base
- A novel scoring and ranking model based on a generative language model for queries on weighted and labeled graphs
- An evaluation of the search-result quality provided by NAGA, based on user assessments and in comparison to state-of-the-art search engines like Google and Yahoo! Answers.

The rest of the paper is organized as follows. Chapter 2 discusses related work. Chapter 3 describes NAGA's knowledge base and introduces the graph-based data model of NAGA. Chapter 4 explains the query language with several examples of its use. In Chapter 5, we present our novel scoring model. Chapter 6 discusses the query processing algorithms and Chapter 7 presents experiments in comparison to conventional Web search engines. Finally, we conclude in Chapter 8.

2 Related Work

The *Semantic Web* community has advocated OWL and RDF as the models for Web scale knowledge representation [46, 7]. However, these models cannot express uncertainties and different confidence levels. RDFS, which can be seen as a schema for annotated graph structures, has recently been extended by means for annotations [49], but this has not yet found applications in the community. There have also been recent proposals, to create "semantic" annotations for hyperlinks (e.g. "Semantic Wikipedia" [50]), Web pages and images (e.g., del.icio.us and flickr.com) by means of *social tagging* (e.g., [26]). The rationale is that the "wisdom of crowds" may lead to "folksonomies" with expressive tags and latent structure (e.g., [23]). However, all these proposals require humans to create, annotate, and maintain the knowledge representation, which is in contrast to NAGA's approach of automated knowledge extraction and maintenance.

Information extraction (IE) methods have been studied for about a decade now; [17, 19, 24, 2] provide excellent overviews of the state of the art. Recently, significant advances on the scalability and robustness of IE techniques have been made (e.g., [1, 37, 8, 47]). The most prominent work that has leveraged IE for casting the Web into relational knowledge is probably the KnowItAll project [27] and recent projects such as ExDBMS [10] and AVATAR [39]. NAGA uses state-of-the-art IE and captures confidence levels of imperfect extraction. In addition and in contrast to the above work, NAGA considers this uncertainty aspect at query time by means of an elaborate ranking method for complex queries based on principles of statistical language models, the latter being one of the cornerstones of modern information retrieval (IR) for text documents [41, 52].

Our query language is akin to the RDF query language SPARQL [18] and XML query languages such as XPath or XQuery. However, all these languages disregard the issue of uncertainty. More related to our work is the research on XML IR for ranked retrieval (see [4, 5] and the references given there). This line of work, however, does not consider graph structures that reflect Web connectivity or arbitrary relations that could be viewed as typed edges in a graph. SPARQL-style languages are geared for graphs (see,

e.g., [6, 29] for recent, powerful models), but do not consider uncertainty and treat ranking as a second-class citizen.

Deep-Web search, vertical search and entity search on the Web, and semantic desktop search [12, 13, 14, 15, 22, 25, 44, 43, 42] enhance keyword-based querying by typed attributes, but none of these approaches is sufficiently complete for effectively searching a richly structured knowledge base. Finally, there is prior work on *graph-oriented text search* in various forms. Schema-oblivious keyword search in relational databases operates on a graph that is created by the foreign-key relationships in the database. BANKS [9], DBXplorer [3], and DISCOVER [35] are the most prominent, early systems of this kind. More recent work along these lines has focused on efficiency issues [21, 33, 38, 40, 45] and did not explore richer functionality. These kinds of data graphs can be generalized into networks of entities and relationships [11], and similar graph structures also arise when considering XML data with XLinks and other cross-references within and across document boundaries [31, 16]. However, these methods operate on text nodes in a graph, but without using explicit relation types in advanced queries. In contrast, NAGA has a notion of typed edges that correspond to binary relations between entities, and can utilize this additional knowledge for more precise querying (along with a principled ranking model).

NAGA's querying power is most comparable to, but goes beyond, the search capabilities of Libra [43], EntitySearch [14], and ExDBMS [10], all of which also operate on relations extracted from Web data. Each of them has salient properties that are not matched by NAGA, but also significant shortcomings compared to the rich functionality of NAGA. Libra focuses on entities and their attributes, using a novel record-level language model. However, it does not address general relations between different entities, and its query model is keyword-centric. EntitySearch facilitates search that can combine keywords and structured attributes in a convenient and powerful manner, and it has an elaborate ranking model for result entities. However, it does not address typed relations between entities and SPARQL-style path expressions on knowledge graphs, and its ranking model is very different from ours. ExDBMS uses powerful IE tools to capture entities and typed relations, and its query model supports this full suite as well. It uses a probabilistic form of Datalog for search [20]. In contrast, NAGA uses a graph-based search paradigm that is more expressive by supporting regular expressions on paths and a very general form of relatedness queries, with joins along edges as a special case. Also, the ranking model of ExDBMS is not nearly as comprehensive as NAGA's language-model-based approach, and the experimental results in [10] are very preliminary.

3 The Knowledge Base

3.1 The Data Model

Formally, our data model is a directed, weighted, labeled multi-graph (V, E, L_V, L_E). V is a set of nodes, $E \subseteq V \times V$ is a multi-set of edges, L_V is a set of node labels and L_E is a set of edge labels. Each node $v \in V$ is assigned a label $l(v) \in L_V$ and each edge $e \in E$ is assigned a label $l(e) \in L_E$.

Each node represents an entity. For example, a node v with label $l(v) =$ `Max_Planck(physicist)` represents the physicist Max Planck. Each edge stands for a relationship between two entities. For example, if w is a node with label $l(w) =$ `1858`, then the fact that Max Planck was born in 1858 is represented by an edge $e = (v, w)$ with the edge label $l(e) =$ `bornInYear`. We call the edges of the knowledge graph *facts* and identify them by their edge label and the two node labels. For example, we write simply `Max_Planck(physicist) bornInYear 1858`.

In our setting, facts are collected by information extraction from Web sources. Since the sources may be unreliable, we have to estimate the *confidence* for each fact. This confidence depends on the trust we have in the Web source s, $tr(s)$, from which the fact was extracted. $tr(s)$ can be approximated, e.g., by the Page Rank of the Web page. Furthermore, the confidence depends on the estimated accuracy $acc(f, s)$ with which the fact f was extracted from the Web source s. This accuracy value is usually provided by the extraction mechanism. Suppose that a fact f was extracted from the Web sources $s_1, \ldots, s_n$. We add f only once to the knowledge graph and compute its confidence value as

$$c(f) = \frac{1}{n} \sum_{i=1}^{n} acc(f, s_i) \cdot tr(s_i) \tag{3.1}$$

When a fact was extracted form a Web source, we call that source a *witness* for the fact. With each fact f, we store its set of witnesses $W(f)$. The rationale behind the above formula is that the confidence in a fact f is computed as the mean value of extraction accuracy and trust accumulated across all the witnesses of f.

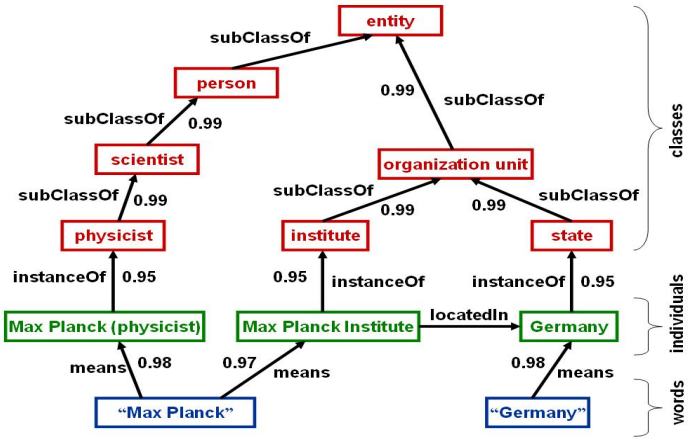

Figure 3.1: Example fragment of the knowledge graph

From an ontological point of view, we distinguish the following three kinds of entities in the knowledge graph:

1. A *word* refers to an entity via the `means` relation. For example, the string "Dr. Planck" may be used to refer to the entity `Max_Planck(physicist)`. This is reflected in our data model as "Dr. Planck" `means` `Max_Planck(physicist)`. Different words may refer to the same entity (synonymy) and the same word may refer to different entities (ambiguity).

2. An *individual* is a real-world object. For example, `Max_Planck(physicist)` or `Germany`, are individuals.

3. A *class* is an entity that represents a group of similar individuals. For example, the class `physicist` represents the group of all physicists. We use the `instanceOf` relation to state that an individual belongs to a class: `Max_Planck(physicist) instanceOf physicist`

The classes are arranged in a hierarchy by means of the `subClassOf` relation. The root of this hierarchy is the class `entity`.

3.2 Building the Knowledge Graph

Currently, NAGA makes use of three basic sources of information. The first is WordNet [28], a comprehensive thesaurus of the English language. Each word sense in WordNet (i.e. each set of synonymous words) becomes a class

in NAGA's knowledge graph (we exclude the proper names from WordNet). WordNet arranges the word senses in a hierarchy, where a more general sense is above the more specific senses (e.g. "scientist" is above "physicist"). This hierarchy gives us the `subclassOf` relation of classes.

The second source for the knowledge graph is Wikipedia, the large online encyclopedia. If a Wikipedia page describes exactly one individual, we make the title of the page an individual in our knowledge graph. For example, the page with the title "Paris" describes the capital of France, so we make `Paris` an individual in the knowledge graph. We connect each individual by the `instanceOf` relation to the corresponding WordNet class. We can derive other relationships from Wikipedia by use of its category system. For example, the page about Paris is in the category "Cities in France". This tells us that Paris is `locatedIn` France. Wikipedia articles are highly interlinked. From this link structure, we derive a `context` relation between individuals. For a more detailed explanation of these techniques, please refer to our prior work in [48].

The third source for the knowledge graph is the Internet Movie Database IMDB. The IMDB contains data about movies, actors, movie directors, producers, etc. These entities may also occur in Wikipedia, possibly with a slightly different name. Hence we use disambiguation heuristics to avoid duplicate entities in the database. Last, we connect the IMDB movies and persons to their corresponding WordNet classes by the `instanceOf` relation.

IE techniques can be used to extract facts from other Web pages. We use the state-of-the-art tool Leila [47], which can extract facts from natural language Web pages. For example, Leila can extract the fact `AlbertEinstein bornIn 1879` from the sentence "Albert Einstein, the great physicist, was born in 1879". For each extracted fact, Leila provides an accuracy estimation.

As of now, since our facts have been extracted from reliable sources, we set the trust for each page to 1 and compute the confidence of each fact according to the formula (3.1) by taking only the extraction accuracy of our extraction process into account [48].

All in all, our knowledge graph contains 3 million entities and 34 million facts about them. The relations include the taxonomy (`instanceOf` and `subclassOf`) and the `context` relation between individuals, as well as general relations like e.g. `bornInYear`, `locatedIn`, `actedInFilm` or `invented`.

4 Query Language

Our query language borrows some concepts from the RDF query language SPARQL [18]. A query is a graph that may contain unlabeled vertices and edges. An answer to a query is a subgraph of the knowledge graph that matches the query graph. Our query language goes beyond SPARQL by allowing also edges labeled with regular expressions.

4.1 Formal Query Model

Given a set S of labels (e.g. relation labels), we denote by $\textsc{RegEx}(S)$ the set of regular expressions over S. We write $\mathcal{L}(r)$ ($\subseteq S^*$) to denote the language of some $r \in \textsc{RegEx}(S)$.

Definition 4.1.1 (Query) *A* query *is a connected directed graph* $Q = (V, E, L_V, L_E, U)$*, where* V *is a set of vertices,* $E \subseteq V \times V$ *is a set of edges,* L_V *is a set of word, individual and class labels, and* L_E *is a set of relation labels.* U *is a set of variables.*
Each vertex $v \in V$ *is assigned a label*

$$l(v) \in L_V \cup U$$

Each edge $e \in E$ *is assigned a label*

$$l(e) \in \textsc{RegEx}(L_E) \cup U \cup \{\texttt{connect}\}$$

If an edge or vertex is labeled with a variable, we call that edge or vertex unbound.

We disallow unbound edges between two unbound vertices.

As in the knowledge graph, the vertex labels L_V denote entities and the edge labels L_E denote relations. Edges can be labeled by the special keyword `connect` or by a regular expressions over L_E. We say that an edge is labeled by a *simple relation* if its label is contained in L_E. In addition, the labels of nodes and edges can be variables. We call an edge of a query graph

a *fact template* and denote it by its egde label and the two node labels. For example, `Albert_Einstein friendOf* $x` is a fact template. Here, `$x` denotes a variable.

NAGA's query model is based on graph matching. Given a query, NAGA aims to find subgraphs of the knowledge graph that match the query graph. Before defining matches to our queries, we first define matches to fact templates. We say that a vertex v from a knowledge graph *matches* a vertex from a query graph with label l, if $l(v) = l$ or if $l(v)$ is a variable. Furthermore, we say that a query vertex v' is *bound* by a vertex v of the knowledge graph if v matches v'.

Definition 4.1.2 (Matching Path) *A* matching path *for a fact template x r y is a sequence of edges $m_1, \ldots, m_n$ from the knowledge graph, such that the following conditions hold:*

- *If r is a variable, then $n = 1$ and the start node of m_1 matches x and the end node of m_1 matches y.*
- *If r is a regular expression, then $m_1, \ldots, m_n$ forms a directed path and $l(m_1) \ldots l(m_n) \in \mathcal{L}(r)$. Furthermore, the start node of m_1 matches x and the end node of m_n matches y*[1]*.*
- *If r=`connect`, then $m_1, \ldots, m_n$ forms an undirected path, such that its start node matches x and its end node matches y.*

Given a query q and an answer graph g, we denote the matching path of a query template q_i from q by $match(q_i, g)$. Now we generalize this definition to queries:

Definition 4.1.3 (Answer Graph) *An* answer graph *to a query Q is a subgraph A of the knowledge graph, such that*

- *for each fact template in the query there is a matching path in A,*
- *each fact in A is part of one matching path,*
- *each vertex of Q is bound to exactly one vertex of A.*

We will occasionally use the label `isA` as a shorthand for the regular expression `instanceOf subclassOf*`. `isA` connects an individual via one `instanceOf`-edge to its immediate class and by several `subclassOf`-edges to more general superclasses.

[1]For the special case $r \in L_E \subseteq \text{RegEx}(L_E)$ (in which r is a simple relation) the match is an edge and $n = 1$.

4.2 Query types

We provide a taxonomy of four query types in ascending order of expressiveness:

Evidence Queries

Evidence Queries allow the user to search for evidence for a hypothesis. For example, Figure 4.1 shows an evidence query that asks for evidence of the hypothesis *Max Planck was a physicist who was born in Kiel*. NAGA's answer graph is a subgraph of the knowledge graph matching the query with confidence values for each fact.

Figure 4.1: Evidence query example

Formally, an evidence query is a query in which all fact templates have only bound components.

Discovery Queries

Discovery Queries are queries that supply the user with pieces of missing information. For example, Figure 4.2 asks for *physicists who were born in the same year as Max Planck*. NAGA attempts to fill in the blanks by finding a subgraph in the knowledge graph that matches the query and thus binds the two variables. Note that there are multiple answers to this query and NAGA returns a ranked list of answers. One possible answer is shown in Figure 4.3.

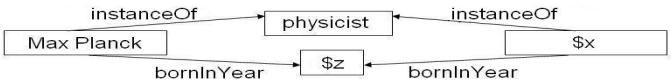

Figure 4.2: Discovery query example

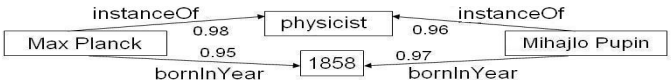

Figure 4.3: Answer graph to the preceding query

Formally, a discovery query is a query in which at least one fact template has an unbound component.

Regular Expression Queries

Regular Expression Queries enable users to specify more flexible matchings by allowing *regex labels* on query edges. Figure 4.4 shows two queries, the first of which uses the regular expression shorthand `isA` to ask *Which rivers are located in Africa?* Here, an answer such as `Nile instanceOf 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.*

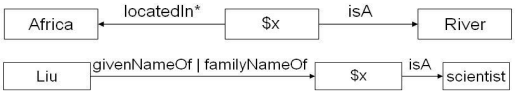

Figure 4.4: Regular expression query examples

We say that a regular expression query is a query in which at least one edge is labeled by a regular expression that is not a simple relation.

Relatedness Queries

Relatedness Queries discover "broad" connections between pieces of information. For example, Figure 4.5 asks the question *How are Margaret Thatcher and Indira Gandhi related?*. 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".

Figure 4.5: Relatedness Query example

Formally, a relatedness query is a query in which at least one edge is labeled by the `connect` label.

5 Ranking Answer Graphs

A good ranking model for answer graphs should satisfy the following desiderata:

1. The uncertainty of extracted facts should be taken into account. That is, answers containing facts with higher *confidence* should be ranked higher.

2. *Informative* answers should be ranked higher. For example, when asking a 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.

3. *Compact* answers should be preferred, i.e. tightly related blocks of information should be ranked higher than loosely related ones. For example, for the query *How are Einstein and Bohr related?* the answer about both having won the Nobel Prize in physics should rank higher than the answer that Tom Cruise connects Einstein and Bohr by being a vegetarian like Einstein, and by being born when Bohr died. This compactness criterion also avoids overly general answers, such as `Albert_Einstein isA entity`, as compared to more specific answers such as `Albert_Einstein isA physicist`, because the path from `Albert_Einstein` to `entity` is longer than the path to `physicist`[1].

We propose a novel scoring model that integrates all the above desiderata in a unified framework. Our approach is inspired by existing work on language models (LM) for information retrieval on document collections [51, 34], but it is adapted and extended to the new domain of knowledge graphs. In this setting, the basic units are not *words*, but *facts* or *fact templates*. Our

[1]Recall from Chapter 4 that `isA` is a shorthand for the regular expression `instanceOf subclassOf*`

graphs and queries can be seen as sets of facts or fact templates respectively. A candidate result graph in our setting corresponds to a document in the standard IR setting.

The language model we propose is much more challenging than the traditional language models for two reasons:

1. By considering facts and fact templates as IR units rather than words-in-documents, our queries include both bound and unbound arguments – a situation that is very different from what we encounter in multi-term queries on documents.

2. Our corpus, the knowledge graph, is virtually free of redundancy - unlike a document-level corpus. This makes reasoning about background models and idf-style aspects [51] more subtle and difficult.

5.1 Language Model

The basic idea of our language modeling approach is in line with IR models [34, 51]. We assume that a query q is generated by a probabilistic model based on a candidate result graph g. Given a query q, we denote its fact templates by $q = q_1 q_2 \cdots q_n$. Analogously, we denote the facts of a candidate answer g by $g = g_1 g_2 \cdots g_n$. We are interested in estimating the conditional probability $P(g|q)$, i.e. the probability that g generates the observed q [51].

After applying Bayes formula and dropping a graph-independent constant (since we are only interested in ranking graphs), we have

$$P(g|q) \sim P(q|g)P(g)$$

where $P(g)$ can reflect a prior belief that g is relevant to any query. $P(q|g)$ is the query likelihood given the graph g which captures how well the graph fits the particular query q. In our setting we assume $P(g)$ to be uniform and thus we are interested in computing $P(q|g)$. In the spirit of IR models [51], we assume probabilistic independence between the query's fact templates, which results in

$$P(q|g) = \prod_{i=1}^{n} P(q_i|g).$$

Following the approach in [34] adapted to our setting, we write the likelihood of a query fact given an answer graph as a mixture of two distributions, $\tilde{P}(q_i|g)$ and $\tilde{P}(q_i)$ as follows:

$$P(q_i|g) \quad = \quad \alpha \cdot \tilde{P}(q_i|g) + (1-\alpha) \cdot \tilde{P}(q_i),\ 0 \leq \alpha \leq 1 \tag{5.1}$$

$\tilde{P}(q_i|g)$ is the probability of drawing q_i randomly from an answer graph, $\tilde{P}(q_i)$ is the probability of drawing q_i randomly from the total knowledge graph and

α is either automatically learned (when relevance information about answer graphs is available [34]) or set to an empirically calibrated global value.

We first show the connection between this probabilistic formulation and $tf \cdot idf$ style retrieval models (following [34]), and then explain how each distribution is modeled and estimated.

Dividing formula (5.1) by $(1-\alpha)\cdot\tilde{P}(q_i)$ does not affect the ranking because α and $\tilde{P}(q_i)$ have the same value for each answer graph. This leads us to the following formulation, which can be interpreted as a probabilistic justification for the popular $tf * idf$ heuristics for standard IR [34] (with the numerator capturing the tf part and the denominator capturing the df part):

$$P(q|g) \sim \prod_{i=1}^{n} \left(1 + \frac{\alpha}{1-\alpha} \cdot \frac{\tilde{P}(q_i|g)}{\tilde{P}(q_i)}\right) \tag{5.2}$$

As mentioned in the scoring model requirements, we want to capture *confidence*, *informativeness*, and *compactness*. We first explain how we formulate the confidence and informativeness components and then explain how our formulation automatically deals with compactness as well. We describe $\tilde{P}(q_i|g)$ using a mixture model which puts different weights on confidence and informativeness.

$$\begin{aligned} \tilde{P}(q_i|g) &= \beta \cdot P_{conf}(q_i|g) + (1-\beta) \cdot P_{info}(q_i|g), \\ 0 \leq \beta \leq 1 \end{aligned} \tag{5.3}$$

Note that confidence and informativeness are indeed independent criteria. For example, we can be very confident that Albert Einstein was both a physicist and a politician, but the former fact is more informative than the latter, because Einstein was a physicist to a larger extent than he was a politician. The next subsections will explain how we estimate confidence, informativeness and compactness.

Estimating Confidence

The maximum likelihood estimator for $P_{conf}(q_i|g)$ is given by:

$$P_{conf}(q_i|g) = \prod_{f \in match(q_i,g)} P(f \text{ holds}) \tag{5.4}$$

where $P(f$ holds$)$ can be approximated by $c(f)$. If q_i is labeled by a simple relation, then $match(q_i, g)$ contains just one fact and $P_{conf}(q_i|g)$ is the confidence of that fact. If q_i is labeled with `connect` or with a regular expression over relations, then $match(q_i, g)$ contains the sequence of facts that together match q_i. The probability of that sequence being true is the product of the confidences of the single facts, assuming that the confidences of the single facts are independent.

Estimating Informativeness

The *informativeness* of a query template q_i given the answer graph g depends on the informativeness of each matching fact in g:

$$P_{info}(q_i|g) = \prod_{f \in match(q_i,g)} P_{finfo}(f|q_i) \tag{5.5}$$

Note that the same fact f may have different informativeness values, depending on the query formulation. For example, the fact `Bob_Unknown instanceOf physicist` is less informative if the query asked for (famous) physicists (`$x instanceOf physicist`), but could be very informative if the query asked about the occupation of `Bob_Unknown` (`Bob_Unknown instanceOf $x`). Thus the informativeness of the fact f depends on the unbound arguments of the query template q_i.

Let $f = (x, r, y)$ be an observation drawn from the joint distribution of three random variables X, R and Y. X and Y take values from the set of knowledge graph nodes and R takes values from the set of edges (i.e. relations). Given a query template $q_i = (x', r', y')$, if $f = (x, r, y)$ is a match for q_i, we define the informativeness of f as follows:

$$P_{finfo}(f|q_i) = \begin{cases} P(x|r,y), & \text{if } x' \text{ unbound in } q_i \\ P(y|r,x), & \text{if } y' \text{ unbound in } q_i \\ P(r|x,y), & \text{if } r' \text{ unbound in } q_i \\ P(x,y|r), & \text{if } x', y' \text{ unbound in } q_i \\ P(x,r|y), & \text{if } x', r' \text{ unbound in } q_i \\ P(r,y|x), & \text{if } r', y' \text{ unbound in } q_i \\ P(x,r,y), & \text{else} \end{cases} \tag{5.6}$$

We show how to estimate these probabilities by the example of $P(x|r, y)$. $P(x|r, y)$ can be written as follows:

$$P(x|r,y) = \frac{P(x,r,y)}{P(r,y)} = \frac{P(x,r,y)}{\sum_{x'} P(x',r,y)} \tag{5.7}$$

We estimate $P(x, r, y)$ using the number of witness pages for the fact (x, r, y) [2]:

$$P(x,r,y) \approx \frac{|W(x,r,y)|}{\sum_{x',r',y'} |W(x',r',y')|} \tag{5.8}$$

Intuitively speaking, informativeness computes the following: If $x\ r\ y$ is an answer to a template $x'\ r'\ y'$ where y' is unbound, then the informativeness computes how often $x\ r\ y$ is mentioned on the Web, normalized by the

[2]The witnesses could also be weighted by their authority, e.g. Page Rank.

summation over the number of times that $x\ r\ y'$ is mentioned on the Web for corresponding y' from the set of entities of the knowledge graph. To see why this formulation captures the intuitive understanding of informativeness, consider some examples.

Let q be the query $q =$ `Albert_Einstein instanceOf $x`, which consists of just one fact template. Let f be a possible answer $f =$ `Albert_Einstein instanceOf physicist`. Here, the informativeness measure for f computes how often Einstein is mentioned as a physicist as compared to how often he is mentioned with some other `instanceOf` fact. Thus, f = `Albert_Einstein instanceOf physicist` will rank higher than $f' =$ `Albert_Einstein instanceOf politician`. In this case, informativeness measures the *degree* to which Einstein was a physicist.

Now consider the query q=`$x instanceOf physicist` and consider again the answer f=`Albert_Einstein instanceOf physicist`. The informativeness measure for f will compute how often Einstein is mentioned as a physicist compared to how often other people are mentioned as physicists. Since Einstein is an important individual among the physicists, `Albert_Einstein instanceOf physicist` will rank higher than `Bob_Unknown instanceOf physicist`. In this case, informativeness measures the *importance* of Einstein in the world of physicists.

Other examples could be: when asking for prizes that Einstein won, our informativeness will favor the prizes he is most known for. When asking for people born in some year, our informativeness favors famous people. When asking for the relationship between two individuals, informativeness favors the most prominent relation among them.

For now the number of witnesses for each fact in our knowledge graph is not statistically significant, because our facts are extracted only from a limited number of Web-based corpora, and many facts appear only on one page. For this reason we approximated the $P(x, r, y)$ values by a heuristic. We transformed the facts into keyword queries and used a search engine to retrieve the number of pages in the Web that contain the corresponding keywords. For example, to estimate $P(\texttt{Albert\_Einstein}|\texttt{instanceOf}, \texttt{physicist})$, we formulated the query "Albert Einstein" + "physicist" and retrieved the number of hits for this query. We retrieved the number of hits for the query "physicist" as well and estimated the probability as follows:

$$P(Albert\_Einstein|instanceOf, physicist) \tag{5.9}$$

$$\sim P(Albert\_Einstein|physicist) \tag{5.10}$$

$$= \frac{P(Albert\_Einstein, physicist)}{P(physicist)} \tag{5.11}$$

$$\sim \frac{\#hits(Albert\ Einstein\ physicist)}{\#hits(physicist)} \tag{5.12}$$

In summary, confidence and informativeness are two complementary components of our model. The *confidence* expresses how certain we are about a specific fact – independent of the query and independent of how popular the fact is on the Web. The *informativeness* captures how useful the fact is for a given query. This depends also on how visible the fact is on the Web.

Estimating Compactness

The third desideratum, *compactness* of results, is implicitly captured by this model. This is because the likelihood of an answer is the product over the likelihoods of its component facts. Thus, it is inversely correlated to the path length. This means that, given two answers with equal informativeness and equal confidence of the component facts, our ranking will always prefer the more compact answer.

For example, for the query `Margaret_Thatcher connect Indira_Ghandi` we get as top results that they are both `Female heads of government`, `Women in war`, and `Former students of Somerville College Oxford`, while less compact (i.e. longer paths) answers that connect them via other persons which have similar characteristics (for example `Jane Fonda` connects them by being as well a `Woman in war`) are ranked lower.

5.2 The Background Model

We turn to estimating $\tilde{P}(q_i)$. $\tilde{P}(q_i)$ plays the role of giving different weights to different fact templates in the query. This is similar in spirit to the idf-style weights for weighting different query terms in traditional language models. For a single-term query the *idf* part would just be a constant shift or scaling, which does not influence the ranking. But for multi-term queries, the *idf* weights give different importance to different query terms. For example, consider the query with two fact templates $q1$=`$y bornIn Ulm` and $q2$=`$y isA scientist`. If matches to this query are only partial, i.e. answers in which only one of the fact templates is matched are allowed, then the more important template should get higher weight. Traditionally, the more important condition is the more specific one – the one that is expected to have fewer matches, i.e., higher *idf*. If there are many people born in Ulm, but there are only few scientists overall, this suggests giving a higher weight to $q2$. By counting edges of the form x `bornIn Ulm` and x `isA scientist` in the overall corpus (knowledge graph), we get corresponding frequency and thus inverse frequency weights, in the *idf* spirit. $\tilde{P}(q_i)$ can be estimated by the frequency in the corpus. This type of background model is heavily used in standard IR [34, 51].

6 Query Processing

Algorithm 1 Query Processing Algorithm

1: **Function: queryResults**(Q)
2: **Input:** A query graph $Q = (V_Q, E_Q, L_{E_Q}, L_{V_Q}, U)$
3: **Output:** A set of answer graphs
4: normalize Q into $Q' = (V_{Q'}, E_{Q'}, L_{E_{Q'}}, L_{V_{Q'}}, U)$
5: return templateResults($Q', E_{Q'}$)

1: **Function: templateResults**(C,E)
2: **Input:** A query graph $C = (V_C, E_C, L_{E_C}, L_{V_C}, U)$
3: **Input:** A set of templates E
4: **Output:** A set of answer graphs
5: If $E = \emptyset$, return $\{C\}$
6: $Results = \emptyset$
7: Pick some template $e \in E$
8: **for all** matches e' of e in the knowledge graph **do**
9: $r_{e'}$ =templateResults($(V_C, E_C - e + e', L_{E_C}, L_{V_C}, U), E - e$)
10: If $r_{e'} \neq \emptyset$, $Result = Result + r_{e'}$
11: **end for**
12: return *Results*

NAGA stores the knowledge graph in a database. A high-level overview of NAGA's query processing algorithm is shown in Algorithm 1.[1] We first pre-process the given query into a normalized form (line 4 in the Function `queryResults`) by applying the following rewritings: First, we add an additional edge with the regular expression `means|familyNameOf` for each bound vertex. For example, the query

```
Einstein hasWonPrize $x
```

becomes

```
"Einstein" means|familyNameOf $Einstein
$Einstein hasWonPrize $x
```

[1] We write $E + e$ for $E \cup \{e\}$.

This allows the user to simply use the word "Einstein" in the query to refer to the entity `Albert_Einstein`. Second, we translate the pseudo-relation `isA` to its explicit form `(instanceOf subclassOf*)`. For example, the query

```
$x isA $y
```

becomes

```
$x (instanceOf subclassOf*) $y.
```

This allows the user to ask for instances of classes without the need to know about regular expressions.

The main function of the query processing algorithm is `templateResults`. It is given a preprocessed query graph and a list of templates to be processed. Initially, the templates are edges of the query graph. Some edge is picked (line 7) and all possible matches of this edge in the query graph are identified. For each possible match, we construct a refined query graph by replacing the query edge by the match. Then, the function is called recursively with the refined query graph. Once no more query edges need to be processed, the refined query graph constitutes a result.

We identify matches for templates as follows. If the label of the edge in the fact template is a **simple relation** or a variable, we translate the template directly to an SQL statement. This applies to templates like "Einstein" `means $z`, "Einstein" `$y Ulm`, or `$x invented $z`, which can be translated into SELECT statements. This gives us a set of matching edges for the template.

If the edge of the template is labeled with a **regular expression** over relations, we construct an automaton for the regular expression. That automaton might be nondeterministic. We identify one vertex v_0 of the edge that is already bound[2]. If v_0 is not the start vertex of the edge, but the target vertex, we invert the automaton. Now, we start a kind of breadth-first-search in the knowledge graph from v_0. The visited vertices of the search are stored in a queue. To each vertex in the queue, we attach (1) a set of states of the automaton and (2) a predecessor vertex. Initially, the queue contains just v_0 with the initial states of the automaton and no predecessor node attached. Whenever we poll a vertex v from the queue, we examine the automaton states attached to v. For each state s, we find all possible vertices v' in the knowledge graph with $v\ r\ v'$, where r is the relation label of the regular expression that is being read by the automaton. To each such v', we attach v as a predecessor vertex and the successor states of s as its states. If s is an exit state of the automaton, we obtain a matching path for the regular expression edge by following the predecessor vertices of v'. Then,

[2]In this case, if none of the vertices in the query template is bound at query time then we bind one of the vertices to some node in the knowledge graph.

v' is enqueued in the queue. This process continues until the queue is empty. This gives us a set of matching paths for the template.

If the template is labeled with `connect`, we search a chain of facts from the first template argument to the second. This is implemented by two breadth-first-searches, which start from the two vertices and grow until they meet. This process can deliver multiple paths between the arguments, if desired. This gives us a set of matching paths for the `connect` template.

We incorporate some simple query optimizations: First, fact templates in which the edge as well as both vertices are not labeled by a variable are processed separately, so that they do not need to be computed in each recursive call. Second, we coalesce subsequent non-regular expression edges to one single SQL statement whenever possible. Furthermore, certain trivial relations (such as e.g. `smallerThan` for numbers or `before` and `after` for dates) are not stored in the database, but are computed at query time.

7 Evaluation

In this chapter we evaluate NAGA's search and ranking behavior. First, we look at some some sample queries and analyze the influence of the various factors of the ranking model, i.e. *confidence*, *informativeness*, and *compactness*, on NAGA's ranking. Second, we show sample queries for each of the query classes (i.e. discovery, regular expression, and relatedness queries) and a subset of the ranked list of results for these queries for both Google and NAGA. Third, we present an extensive user study that compares NAGA's ranking performance to the search results and rankings produced by Google and Yahoo! Answers.

7.1 Influence of ranking desiderata

As explained in Chapter 5 the parameters of the ranking model allow emphasizing the *confidence* or the *informativeness* of the results, while at the same time the *compactness* of answers is implicitly promoted. We study the influence of these factors for two simple queries `$y isa physicist` and `Einstein isa $x`. We set $\alpha = 1$. This disables the *idf* component of our model, i.e., the background model.

We first analyze the query `$y isA physicist`. For this query, we expect answers about famous physicists at the top of the ranked list. If we choose to rank by confidence alone, e.g. we set $\beta = 1$ we get as the top results less known physicists, while the famous ones, e.g. *Albert Einstein, Niels Bohr*, etc., are ranked lower in the list. This happens because we can be equally confident that a less known physicist is a physicist, as we are for a famous one, but this does not promote the famous physicists.

If we enable the informativeness component by setting $\beta = 0.5$ (which gives equal weight to confidence and informativeness), the top three results are about the famous physicists *Albert Einstein, Niels Bohr* and *Max Planck*, followed by *Marie Curie* and *Blaise Pascal*. Thus our informativeness aspect plays a very important role in satisfying the information demand latent in the query.

We can observe the same effect for the query `Einstein isA $x`. Note

that *Einstein* is an ambiguous word that can refer to multiple entities in our knowledge graph. We expect the answers to this query to be ranked based on two aspects of the information need: first, answers about famous entities which the word *Einstein* refers to in the knowledge graph should be preferred, (e.g. Einstein refers among many other things to the famous physicist Albert Einstein), and second, for the chosen entities, the extent to which an entity has a certain property should be considered (e.g. Albert Einstein was a physicist to a higher extent than he was a politician). If we choose to rank by confidence alone, i.e., we set $\beta = 1$, the top results for this query are *Bob Einstein* who is an actor, *Arik Einstein* who is a singer; a crater and an asteroid which bear the name Einstein follow, and only very low in the ranking the famous physicist *Albert Einstein* appears. As soon as informativeness comes into play, with the setting $\beta = 0.5$, the top results are about the famous physicist *Albert Einstein*, namely, that he was a scientist, a physicist, a philosopher, etc., while the less informative entities that bear the name Einstein are at the bottom of the list. Simultaneously, the compactness property of our ranking ensures that answer graphs with long paths are at the bottom of the ranked list, e.g., the answer graph entailing that *Albert Einstein* is an *entity* is ranked very low.

The parameter β can be used to formulate a more flexible scoring, in which either confidence or informativeness could be given a higher emphasis. For example, if we search for a drug that heals malaria, we would want to emphasize confidence more than informativeness, i.e. we would not be interested in famous drugs for malaria, but in drugs that have high associated confidence for healing the disease. If we want to find out new meanings associated with a word, we may emphasize informativeness more than confidence. This would promote information that appears in many possibly low confidence sources, e.g. revealing that the word `Kleenex` (which is a trademark) is used by many people with the meaning of `tissues`.

The simple examples above show the power of the ranking strategy provided by the combined components of our ranking model. For the rest of the paper, we set β to the balanced value 0.5 and maintain for α the value 1.

In Table 2 we show sample queries for each query class and the *top*-3 ranked result lists for these queries, for both Google and NAGA. For Google, we show the result snippets as returned by the search engine, while for NAGA the corresponding result graphs. As we can observe from the table, NAGA successfully manages to deal with the ambiguity of the names `Hitchcock` and `Pulitzer`, returning as top answers relevant information about the famous movie director and the famous journalist. Google's third hit is relevant for the first query, while for the second query Google does not deliver any relevant result in *top*-3. For the connection query Google returns pages discussing certain problems between the ethnics of the two countries. NAGA returns results about both being `European countries` and about entities that are co-located in Albania and Greece. These results already give an impression

of NAGA's ranking capabilities.

7.2 User Study

7.2.1 Benchmarks

We evaluated NAGA on three sets of queries.

- TREC 2005 and TREC 2006 provide standard benchmarks for question answering systems. Out of this set, we determined the questions that can be expressed by the current set of NAGA relations. We obtained a set of 55 questions (query set TREC). Note that although NAGA knows the relations used in the questions, the knowledge graph does not necessarily contain the data to answer them.

- The work on SphereSearch [30] provides a set of 50 natural language questions for the evaluation of a search engine. Again, we determined the 12 questions that can be expressed in NAGA relations (query set SphereSearch).

- Since, to the best of our knowledge, we are the first to utilize regular expressions over relations and relatedness queries, we had to provide these queries by ourselves. We constructed 18 natural language questions for this purpose(query set OWN).

For NAGA, we translated the questions to graph queries. Table 1 shows some sample questions with their translations.

7.2.2 Competitors

We were interested in comparing NAGA to other systems that common Internet users would use to answer questions. The most obvious established competitors in this domain are Google and Yahoo! Answers. It is clear that these systems are considerably different. Google is designed to find Web pages, not to answer questions. Still, it is a reasonable competitor, because it is the best-known portal to the knowledge of the Web. It is also tuned to answer specific types of questions (like *When was Einstein born?*) directly by its built-in question answering system. Yahoo! Answers has its own corpus of questions together with their answers (provided by humans in the social-Web spirit). When given a question, it tries to match it to a question in its corpus and retrieves the answer. Unlike NAGA, Yahoo! Answers can answer questions directly in natural language. It is probably the closest established real-world competitor.

All the questions were posed to Google, Yahoo! Answers, and NAGA. While for Google and Yahoo! Answers the queries were posed in their original natural language form, for NAGA the queries were posed in their graph form.

Corpus	Question with NAGA translation

TREC	When was Shakespeare born?
	`Shakespeare bornInYear $x`
	In what country is Luxor?
	`Luxor locatedIn $x`
	`$x isA country`

SphereSearch	In which movies did a governor act?
	`$y isA governor`
	`$y actedIn $z`
	What was discovered in the 20th century?
	`$x discoveredInYear $y`
	`$y after 1900`
	`$y before 2000`

OWN	Who produced or directed the movie "Around the World in 80 Days"?
	`$x produced|directed`
	`Around_the_World_in_80_Days`
	What do Albert Einstein and Niels Bohr have in common?
	`Albert_Einstein connect Niels_Bohr`

Table 7.1: Sample queries

This type of comparison is influenced by several aspects: First, the performance of a search engine in this evaluation depends on how precise the question can be formulated for the search engine. Second, it will depend on the size of the knowledge base that the search engine uses. Last, the comparison measures the quality of the ranking of results produced by the search engine. Clearly, NAGA has an advantage over Google and Yahoo! Answers, because the questions are already translated into the graph query language. At the same time, Google and Yahoo! Answers have a massive advantage over NAGA, because they are commercially operated systems that can search the whole Web (Google) or a huge corpus of several million predefined questions (Yahoo! Answers). Thus, our evaluation is in effect a stress test of real-world applicability for all three search engines.

7.2.3 Measurements

For each question, the top-ten results of all search engines were shown to human judges. On average, every result was assessed by 4 human judges, who were students not working on this project. For each result of each search engine, the judges had to decide on a scale from 2 to 0 whether the result is highly relevant (2), correct but less relevant (1), or irrelevant (0).

Google	NAGA
When was Hitchcock born?	`Hitchcock bornInYear $x`

HITCHCOCK Family History — Cousin's Corner — 1689 Spouse: Hannah CHAPIN Married: 27-Sep-1666 in: Springfield, HampdenCo,MA F Child 2 Hannah HITCHCOCK Born: 1645 in: Died: 31-Aug-1733 in: Hadley	`"Hitchcock" familyNameOf Alfred_Hitchcock`
	`Alfred_Hitchcock bornInYear 1899`
	{$@Hitchcock=Alfred_Hitchcock, $X=1899}
	Score: 6.180527667244928E-5

Russell Hitchcock, born in Melbourne, Australia, rock vocalist ... Web brainyhistory.com. June 11, 1952 in History. Born: Russell Hitchcock, born in Melbourne, Australia, rock vocalist, Air Supply, Related Topics:...	`Robyn_Hitchcock bornInYear 1953`
	`"Hitchcock" familyNameOf Robyn_Hitchcock`
	{$@Hitchcock=Robyn_Hitchcock, $X=1953}
	Score: 2.521802368090589E-7

Alfred Hitchcock, born in London, director, Psycho, Birds, Rear ... Born: Alfred Hitchcock, born in London, director, Psycho, Birds, Rear Window, Related Topics: Alfred Birds director Hitchcock ...	`Ken_Hitchcock bornInYear 1951`
	`"Hitchcock" familyNameOf Ken_Hitchcock`
	{$@Hitchcock=Ken_Hitchcock, $X=1951}
	Score: 9.800236818104399E-8

After whom is the Pulitzer prize named?	`Pulitzer givennameOf|familyNameOf $x; $x instanceOf $y`

Book results for After whom is the Pulitzer prize named? Doctors and Discoveries - 459 pages San Francisco. - by Richard Sterling, Tom Downs - 352 pages	`"Pulitzer" familyNameOf Joseph_Pulitzer`
	`Joseph_Pulitzer instanceOf Publisher_19th_century`
	{$X=Joseph_Pulitzer, $Y=Publisher_19th_century}
	Score: 2.377569870331447E-6

TicketWeb Serafina retreats to the safe haven of memories after the death of her husband, ... The Rose Tattoo by Pulitzer Prize winning dramatist, Tennessee Williams.	`"Pulitzer" familyNameOf Joseph_Pulitzer`
	`Joseph_Pulitzer instanceOf Publisher_20th_century`
	{$X=Joseph_Pulitzer, $Y=Publisher_20th_century}
	Score: 2.377569870331447E-6

Hemingway - Wikipedia, the free encyclopedia Hemingway received the Pulitzer Prize in 1953 for The Old Man and the Sea.... Hemingway's first novel after For Whom the Bell Tolls was ...	`"Pulitzer" familyNameOf Joseph_Pulitzer`
	`Joseph_Pulitzer instanceOf American_journalist`
	{$X=Joseph_Pulitzer, $Y=journalist}
	Score: 1.7510477120329342E-6

What do Albania and Greece have in common?	`Albania connect Greece`

Presidenti i Republikes se Shqiperise You issued a message for the Greece-Albania match, but do you have guarantee from President Papulias for a righteous treatment of the Albanians, ...	`"Greece" means Greece`
	`"Albania" means Albania`
	`Albania instanceOf European_country`
	`Greece instanceOf European_country`
	Score: 1.623389355769744E-7

The Bridge - When Christos met Gazi And we have in common many ideas on matters related to the subject of identity. ... for a bilingual country, with both Greek and Albanian inhabitants. ...	`"Greece" means Greece`
	`"Albania" means Albania`
	`Ionian_Islands locatedIn Albania`
	`Ionian_Islands locatedIn Greece`
	Score: 5.635796257200772E-8

We are not Greek, but...: Dealing with the Greek-Albanian Border.... do the Lunxhotes speak Greek in addition to Albanian, or not?... have been common among the Muslims, especially after 1976 and the ban on religion ...	`"Greece" means Greece`
	`"Albania" means Albania`
	`Lake_Prespa locatedIn Albania`
	`Lake_Prespa locatedIn Greece`
	Score: 1.6873181761967196E-11

Table 7.2: Sample Queries and Rankings

Corpus	#Q	#A	Measure	Google	Yahoo	NAGA
TREC	55	180	NDCG	75.64% ± 6.31%	25.15% ± 6.38%	**87.64%** ± 4.79%
			P@1	67.41% ± 6.91%	16.55% ± 5.44%	**76.38%** ± 6.24%
SphereSearch	12	68	NDCG	38.22% ± 11.85%	17.20% ± 9.02%	**84.68%** ± 8.57%
			P@1	19.38% ± 9.49%	6.15% ± 5.29%	**79.23%** ± 9.76%
OWN	18	70	NDCG	54.18% ± 11.99%	18.23% ± 9.12%	**92.88%** ± 5.69%
			P@1	28.33% ± 10.77%	6.66% ± 5.48%	**86.56%** ± 7.94%

#Q – Number of questions
#A – Total number of assessments for all questions

Table 7.3: Results

NAGA tries to answer queries precisely by finding matchings to the query graphs in the knowledge graph. Since the knowledge graph is free of redundancy and bound to the semantics of the relations it builds on, the number of results returned for some queries could be smaller then ten. For example, for a query such as `Albert Einstein bornInYear $x`, NAGA returns only the result `Albert Einstein bornInYear 1879`. Hence the direct comparison with the other search engines in terms of the well known precision-at-top-10 ($P@10$) measure would be misleading. Therefore we chose an evaluation measure that is not dependent on the number of results returned by the system for a given query. Additionally, this measure can exploit the grading of relevant results.

This is the *Normalized Discounted Cumulative Gain (NDCG)* introduced by [36] and intensively used in IR benchmarking (e.g. TREC). It computes the cumulative gain the user obtains by examining the retrieval results up to a fixed rank position. The NDCG takes into account that highly relevant documents are more valuable than marginally relevant documents, and that the lower a relevant result is ranked, the less valuable it is for the user, because the less likely it is that the user will examine the result. Thus this measure not only estimates the number of relevant results in the ranked list, but also incures a penalty for relevant results that are ranked low in the list. Given a query and a ranked list of results $r = r_1, \ldots, r_n$, the *gain* G_i of the result at rank i is the judgment of the user (on the scale from irrelevant (0) to highly relevant (2)). From $G_1, \ldots, G_n$, one derives the *Discounted Cumulative Gain vector* $\overrightarrow{DCG_r}$, which is defined recursively as follows:

$$\overrightarrow{DCG_r}[i] = \begin{cases} G[1] & \text{if i =1;} \\ \overrightarrow{DCG_r}[i-1] + \frac{G[i]}{\log i} & \text{otherwise.} \end{cases}$$

The value $DCG_r = \overrightarrow{DCG_r}[n]$ is the *Discounted Cumulative Gain* of the ranking r. Now, one constructs the ideal ranking $r' = r'_1, \ldots, r'_n$, in which a more-relevant result always precedes a less-relevant result. The Discounted

Cumulative Gain DCG_r is then normalized by this maximum value $DCG_{r'}$, yielding the NDCG for r:

$$NDCG_r = \frac{DCG_r}{DCG_{r'}}$$

We average the NDCG for one query over all user evaluations for that query and average these values over all queries.

Furthermore, we provide the precision at one, P@1. P@1 is the number of times that a search engine provided a relevant result in the first position of the ranking, weighted by the relevance score (0 to 2) and normalized by the total number of queries. Thus, P@1 measures how satisfied the user was on average with the first answer of the search engine.

To be sure that our findings are statistically significant, we compute the Wilson interval for the estimates of NDCG and P@1.

7.3 Results

Table 3 shows the results of our evaluation. For the TREC query set, Google performs very well. It has a high NDCG and in the majority of cases, the first hit in its result ranking was already a satisfactory answer. The reason for this is that the TREC questions are mostly of a very basic nature (see Table 1) and Google can answer a major part of them directly by its highly precise built-in question answering system. In contrast, Yahoo! Answers performs less well. Very often, it retrieves answers to questions that have only the stop-words in common with the question we posed. In many cases, it does not deliver an answer at all. NAGA, in contrast, is very strong on the TREC questions, because they translate mostly to just one fact template. However, NAGA does not always have the answer to the question in its knowledge graph. Still, it performs better than both Yahoo! Answers and Google.

The SphereSearch questions are of a more sophisticated nature. They ask for a non-trivial combination of different pieces of information. Consequently, both Google and Yahoo! Answers perform worse than for the TREC questions. NAGA, in contrast, excels on these questions, because they make full use of its graph-based query language.

For our OWN corpus, Google again performs quite well. This is because the questions mostly ask for a broad relationship between two individuals. Google can answer these questions by retrieving Web documents that contain the two names. In most cases, these answers were satisfactory. Yahoo! Answers has again difficulties, whereas NAGA delivers good results for the majority of questions and clearly outperforms Google.

In summary, our user study shows that NAGA can deliver the right answers to benchmark questions in the vast majority of cases, if the questions

can be expressed with NAGA relations. Even if one leaves aside the questions in our OWN query set, the user study shows that NAGA outperforms the established competitors Yahoo! Answers and Google on this task by a considerable margin.

8 Conclusions

In this paper, we presented the NAGA search engine which facilitates advanced querying for knowledge rather than merely retrieving Web pages. NAGA's large knowledge base, organized as a directed graph, consists of millions of facts automatically extracted from Web-based corpora. We introduced a graph-based query language with several distinctive features. We proposed a novel scoring mechanism based on generative language models, incorporating the notions of confidence, informativeness, and compactness in a principled manner. We compared our system to both Google and Yahoo! Answers by conducting a comprehensive user study on a variety of simple and complex queries. Our query set consisted of our own queries as well as queries from standard benchmarks like TREC question answering. The results demonstrated that NAGA returns answers that are superior in quality to both competitors.

Our future work will aim to further increase the scale and scope of our knowledge base by incorporating more sources.

A second aspect for future work is query-processing efficiency. Although, for almost all queries, the first result is returned in less then a second, the efficiency for relatedness queries can still be improved.

We plan to adopt and adapt recently proposed algorithms for keyword search in relational graphs for this purpose. NAGA is already available online at `http://www.mpii.mpg.de/~kasneci/naga`.

Bibliography

[1] E. Agichtein. Scaling information extraction to large document collections. *IEEE Data Eng. Bull.*, 28(4), 2005.

[2] E. Agichtein, S. Sarawagi. Scalable information extraction and integration (tutorial). *KDD*, 2006.

[3] S. Agrawal, S. Chaudhuri, G. Das. DBXplorer: A system for keyword-based search over relational databases. *ICDE*, 2002.

[4] S. Amer-Yahia, P. Case, T. Rölleke, J. Shanmugasundaram, G. Weikum. Report on the DB/IR panel at SIGMOD 2005. *SIGMOD Record*, 34(4), 2005.

[5] S. Amer-Yahia, J. Shanmugasundaram. XML Full-Text search: challenges and opportunities (tutorial). *VLDB*, 2005.

[6] K. Anyanwu, A. Maduko, A. Sheth. Sparq2l:towards support for subgraph extraction queries in rdf databases. *WWW*, 2007.

[7] F. Baader, I. Horrocks, U. Sattler. Description logics as ontology languages for the semantic web. In *Mechanizing Mathematical Reasoning*, Springer, 2005.

[8] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, O. Etzioni. Open information extraction from the web. *IJCAI*, 2007.

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

[10] M. Cafarella, C. Re, D. Suciu, O. Etzioni. Structured querying of web text data: A technical challenge. *CIDR*, 2007.

[11] S. Chakrabarti. Breaking through the syntax barrier: Searching with entities and relations. *ECML*, 2004.

[12] S. Chakrabarti. Dynamic personalized pagerank in entity-relation graphs. *WWW*, 2007.

[13] K. C-C. Chang, B. He, Z. Zhang. Toward large scale integration: Building a metaquerier over databases on the web. *CIDR*, 2005.

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

[15] P.-A. Chirita, S. Ghita, W. Nejdl, R. Paiu. Beagle++: Semantically Enhanced Searching and Ranking on the Desktop. *ESWC*, 2006.

[16] S. Cohen, Y. Kanza, B. Kimelfeld, Y. Sagiv. Interconnection semantics for keyword search in XML. *CIKM*, 2005.

[17] W. Cohen. Information extraction (tutorial). http://www.cs.cmu.edu/ wcohen/ie-survey.ppt, 2004.

[18] W. W. W. Consortium. SPARQL. http://www.w3.org/TR/rdf-sparql-query/, 2005.

[19] H. Cunningham. *An Introduction to Information Extraction.* In *Encyclopedia of Language and Linguistics*, Elsevier, 2005.

[20] N.N. Dalvi, D. Suciu. Efficient Query Evaluation on Probabilistic Databases. *VLDB*, 2004.

[21] B. Ding, J.X. Yu, S. Wang, L. Qin, X. Zhang, X. Lin. Finding Top-k Min-Cost Connected Trees in Databases. *ICDE*, 2007.

[22] J.-P. Dittrich, M. A. V. Salles. idm: A unified and versatile data model for personal dataspace management. *VLDB*, 2006.

[23] A. Doan et al., Community information management. *IEEE Data Eng. Bull.*, 29(1), 2006.

[24] A. Doan, R. Ramakrishnan, S. Vaithyanathan. Managing information extraction: state of the art and research directions (tutorial). *SIGMOD*, 2006.

[25] X. Dong, A. Y. Halevy. A platform for personal information management and integration. *CIDR*, 2005.

[26] M. Dubinko, R. Kumar, J. Magnani, J. Novak, P. Raghavan, A. Tomkins. Visualizing tags over time. *WWW*, 2006.

[27] O. Etzioni et al. Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165(1), 2005.

[28] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[29] T. Furche, B. Linse, F. Bry, D. Plexousakis, G. Gottlob. RDF Querying: Language Constructs and Evaluation Methods Compared. In *Reasoning Web 2006*, Springer, 2006.

[30] J. Graupmann. *The SphereSearch Engine for Graph-based Search on heterogeneous semi-structured data (in German)*. Dissertation, Saarland University, May 2006.

[31] J. Graupmann, R. Schenkel, G. Weikum. The spheresearch engine for unified ranked retrieval of heterogeneous XML and web documents. *VLDB*, 2005.

[32] J. Han, X. Yan, P.S. Yu. Mining and searching graphs and structures (tutorial). *KDD*, 2006.

[33] H. He, H. Wang, J. Yang, P. Yu. BLINKS: Ranked keyword searches on graphs. *SIGMOD*, 2007.

[34] D. Hiemstra, A. de Vries. Relating the new language model of information retrieval to the traditional retrieval models. Technical Report TR-CTIT-00-09, University of Twente, 2000.

[35] V. Hristidis, Y. Papakonstantinou. DISCOVER: Keyword search in relational databases. *VLDB*, 2002.

[36] K. Järvelin J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. *SIGIR*, 2000.

[37] T. S. Jayram, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, H. Zhu. Avatar information extraction system. *IEEE Data Eng. Bull.*, 29(1), 2006.

[38] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, H. Karambelkar. Bidirectional expansion for keyword search on graph databases. *VLDB*, 2005.

[39] E. Kandogan, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, H. Zhu. Avatar semantic search: a database approach to information retrieval. *SIGMOD*, 2006.

[40] B. Kimelfeld, Y. Sagiv. Finding and approximating top-k answers in keyword proximity search. *PODS*, 2006.

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

[42] J. Madhavan, S. Cohen, X. Dong, A. Halevy, S. Jeffery, D. Ko, C. Yu. Navigating the seas of structured web data. *CIDR*, 2007.

[43] Z. Nie, S. S. Y. Ma, J.-R. Wen, W. Ma. Web object retrieval. *WWW*, 2007.

[44] Z. Nie, Y. Zhang, J.-R. Wen, W.-Y. Ma. Object-level ranking: bringing order to web objects. *WWW*, 2005.

[45] M. Sayyadian, H. LeKhac, A. Doan, L. Gravano. Efficient Keyword Search Across Heterogeneous Relational Databases. *ICDE*, 2007.

[46] S. Staab, R. Studer. *Handbook on Ontologies*. Springer, 2004.

[47] F. M. Suchanek, G. Ifrim, G. Weikum. Combining Linguistic and Statistical Analysis to Extract Relations from Web Documents. *KDD*, 2006.

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

[49] O. Udrea, D. Recupero, V. S. Subrahmanian. Annotated rdf. *ESWC*, 2006.

[50] M. Voelkel, M. Kroetzsch, D. Vrandecic, H. Haller, R. Studer. Semantic wikipedia. *WWW*, 2006.

[51] C. Zhai. *Risk Minimization and Language Modeling in Text Retrieval*. PhD thesis, Carnegie Mellon University, 2003.

[52] C. Zhai and J. Lafferty. A risk minimization framework for information retrieval. *Information Processing and Management 42*, 2006.

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Stuhlsatzenhausweg 85
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-2007-5-001 G. Kasneci, F.M. Suchanek, G. Ifrim, M. Ramanath, G. Weikum

NAGA: Searching and Ranking Knowledge

MPI-I-2007-2-001 A. Podelski, S. Wagner

A Method and a Tool for Automatic Verification of Region Stability for Hybrid Systems

MPI-I-2006-5-006 G. Kasneci, F.M. Suchanek, G. Weikum

Yago - A Core of Semantic Knowledge

MPI-I-2006-5-005 R. Angelova, S. Siersdorfer

A Neighborhood-Based Approach for Clustering of Linked Document Collections

MPI-I-2006-5-004 F. Suchanek, G. Ifrim, G. Weikum

Combining Linguistic and Statistical Analysis to Extract Relations from Web Documents

MPI-I-2006-5-003 V. Scholz, M. Magnor

Garment Texture Editing in Monocular Video Sequences based on Color-Coded Printing Patterns

MPI-I-2006-5-002 H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum

IO-Top-k: Index-access Optimized Top-k Query Processing

MPI-I-2006-5-001 M. Bender, S. Michel, G. Weikum, P. Triantafilou

Overlap-Aware Global df Estimation in Distributed Information Retrieval Systems

MPI-I-2006-4-010 A. Belyaev, T. Langer, H. Seidel

Mean Value Coordinates for Arbitrary Spherical Polygons and Polyhedra in $\mathbb{R}^3$

MPI-I-2006-4-009 J. Gall, J. Potthoff, B. Rosenhahn, C. Schnoerr, H. Seidel

Interacting and Annealing Particle Filters: Mathematics and a Recipe for Applications

MPI-I-2006-4-008 I. Albrecht, M. Kipp, M. Neff, H. Seidel

Gesture Modeling and Animation by Imitation

MPI-I-2006-4-007 O. Schall, A. Belyaev, H. Seidel

Feature-preserving Non-local Denoising of Static and Time-varying Range Data

MPI-I-2006-4-006 C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, H. Seidel

Enhanced Dynamic Reflectometry for Relightable Free-Viewpoint Video

MPI-I-2006-4-005 A. Belyaev, H. Seidel, S. Yoshizawa

Skeleton-driven Laplacian Mesh Deformations

MPI-I-2006-4-004 V. Havran, R. Herzog, H. Seidel

On Fast Construction of Spatial Hierarchies for Ray Tracing

MPI-I-2006-4-003 E. de Aguiar, R. Zayer, C. Theobalt, M. Magnor, H. Seidel

A Framework for Natural Animation of Digitized Models

MPI-I-2006-4-002 G. Ziegler, A. Tevs, C. Theobalt, H. Seidel

GPU Point List Generation through Histogram Pyramids

MPI-I-2006-4-001 A. Efremov, R. Mantiuk, K. Myszkowski, H. Seidel

Design and Evaluation of Backward Compatible High Dynamic Range Video Compression

MPI-I-2006-2-001 T. Wies, V. Kuncak, K. Zee, A. Podelski, M. Rinard

On Verifying Complex Properties using Symbolic Shape Analysis

MPI-I-2006-1-007 H. Bast, I. Weber, C.W. Mortensen

Output-Sensitive Autocompletion Search

MPI-I-2006-1-006 M. Kerber

Division-Free Computation of Subresultants Using Bezout Matrices

MPI-I-2006-1-005 A. Eigenwillig, L. Kettner, N. Wolpert

Snap Rounding of Bzier Curves

MPI-I-2006-1-004 S. Funke, S. Laue, R. Naujoks, L. Zvi

Power Assignment Problems in Wireless Communication

MPI-I-2005-5-002 S. Siersdorfer, G. Weikum

Automated Retraining Methods for Document Classification and their Parameter Tuning

MPI-I-2005-4-006 C. Fuchs, M. Goesele, T. Chen, H. Seidel

An Emperical Model for Heterogeneous Translucent Objects

MPI-I-2005-4-005 G. Krawczyk, M. Goesele, H. Seidel

Photometric Calibration of High Dynamic Range Cameras

MPI-I-2005-4-005 M. Goesele

?

MPI-I-2005-4-004 C. Theobalt, N. Ahmed, E. De Aguiar, G. Ziegler, H. Lensch, M.A. Magnor, H. Seidel

Joint Motion and Reflectance Capture for Creating Relightable 3D Videos

MPI-I-2005-4-003 T. Langer, A.G. Belyaev, H. Seidel

Analysis and Design of Discrete Normals and Curvatures

MPI-I-2005-4-002 O. Schall, A. Belyaev, H. Seidel

Sparse Meshing of Uncertain and Noisy Surface Scattered Data

MPI-I-2005-4-001 M. Fuchs, V. Blanz, H. Lensch, H. Seidel

Reflectance from Images: A Model-Based Approach for Human Faces

MPI-I-2005-2-004 Y. Kazakov

A Framework of Refutational Theorem Proving for Saturation-Based Decision Procedures

MPI-I-2005-2-003 H.d. Nivelle

Using Resolution as a Decision Procedure

MPI-I-2005-2-002 P. Maier, W. Charatonik, L. Georgieva

Bounded Model Checking of Pointer Programs

MPI-I-2005-2-001 J. Hoffmann, C. Gomes, B. Selman

Bottleneck Behavior in CNF Formulas

MPI-I-2005-1-008 C. Gotsman, K. Kaligosi, K. Mehlhorn, D. Michail, E. Pyrga

Cycle Bases of Graphs and Sampled Manifolds

MPI-I-2005-1-007 I. Katriel, M. Kutz

A Faster Algorithm for Computing a Longest Common Increasing Subsequence

MPI-I-2005-1-003 S. Baswana, K. Telikepalli

Improved Algorithms for All-Pairs Approximate Shortest Paths in Weighted Graphs

MPI-I-2005-1-002 I. Katriel, M. Kutz, M. Skutella

Reachability Substitutes for Planar Digraphs

MPI-I-2005-1-001 D. Michail

Rank-Maximal through Maximum Weight Matchings

MPI-I-2004-NWG3-001 M. Magnor

Axisymmetric Reconstruction and 3D Visualization of Bipolar Planetary Nebulae

MPI-I-2004-NWG1-001 B. Blanchet

Automatic Proof of Strong Secrecy for Security Protocols

MPI-I-2004-5-001 S. Siersdorfer, S. Sizov, G. Weikum

Goal-oriented Methods and Meta Methods for Document Classification and their Parameter Tuning

MPI-I-2004-4-006 K. Dmitriev, V. Havran, H. Seidel

Faster Ray Tracing with SIMD Shaft Culling

MPI-I-2004-4-005 I.P. Ivrissimtzis, W.-. Jeong, S. Lee, Y.a. Lee, H.-. Seidel

Neural Meshes: Surface Reconstruction with a Learning Algorithm

MPI-I-2004-4-004 R. Zayer, C. Rssl, H. Seidel

r-Adaptive Parameterization of Surfaces

MPI-I-2004-4-003 Y. Ohtake, A. Belyaev, H. Seidel

3D Scattered Data Interpolation and Approximation with Multilevel Compactly Supported RBFs

MPI-I-2004-4-002 Y. Ohtake, A. Belyaev, H. Seidel

Quadric-Based Mesh Reconstruction from Scattered Data

MPI-I-2004-4-001 J. Haber, C. Schmitt, M. Koster, H. Seidel

Modeling Hair using a Wisp Hair Model

MPI-I-2004-2-007 S. Wagner

Summaries for While Programs with Recursion

MPI-I-2004-2-002 P. Maier

Intuitionistic LTL and a New Characterization of Safety and Liveness

MPI-I-2004-2-001 H. de Nivelle, Y. Kazakov

Resolution Decision Procedures for the Guarded Fragment with Transitive Guards

MPI-I-2004-1-006 L.S. Chandran, N. Sivadasan

On the Hadwiger's Conjecture for Graph Products

MPI-I-2004-1-005 S. Schmitt, L. Fousse

A comparison of polynomial evaluation schemes

MPI-I-2004-1-004 N. Sivadasan, P. Sanders, M. Skutella

Online Scheduling with Bounded Migration

MPI-I-2004-1-003 I. Katriel

On Algorithms for Online Topological Ordering and Sorting