The YAGO-NAGA Approach to Knowledge Discovery

Gjergji Kasneci, Maya Ramanath, Fabian Suchanek, Gerhard Weikum
Max Planck Institute for Informatics
D-66123 Saarbruecken, Germany
kasneci,ramanath,suchanek,weikum@mpi-inf.mpg.de

ABSTRACT

This paper gives an overview on the YAGO-NAGA approach to information extraction for building a conveniently searchable, large-scale, highly accurate knowledge base of common facts. YAGO harvests infoboxes and category names of Wikipedia for facts about individual entities, and it reconciles these with the taxonomic backbone of WordNet in order to ensure that all entities have proper classes and the class system is consistent. Currently, the YAGO knowledge base contains about 19 million instances of binary relations for about 1.95 million entities. Based on intensive sampling, its accuracy is estimated to be above 95 percent. The paper presents the architecture of the YAGO extractor toolkit, its distinctive approach to consistency checking, its provisions for maintenance and further growth, and the query engine for YAGO, coined NAGA. It also discusses ongoing work on extensions towards integrating fact candidates extracted from natural-language text sources.

1. INTRODUCTION

Universal, comprehensive knowledge bases have been an elusive AI goal for many years. Ontologies and thesauri such as OpenCyc, SUMO, WordNet, or UMLS (for the biomedical domain) are achievements along this route. But they are typically focused on intensional knowledge about semantic classes. For example, they would know that mathematicians are scientists, that scientists are humans (and mammals and vertebrates, etc.); and they may also know that humans are either male or female, cannot fly (without tools) but can compose and play music, and so on. However, the currently available ontologies typically disregard the extensional knowledge about individual entities: instances of the semantic classes that are captured and interconnected in the ontology. For example, none of the above mentioned ontologies knows more than a handful of concrete mathematicians (or famous biologists etc.). Today, the best source for extensional knowledge is probably Wikipedia, providing a wealth of knowledge about individual entities and their relationships. But most of this knowledge is only latent, by being embedded in the natural-language text of Wikipedia articles or, in the best case, reflected in the semistructured components of Wikipedia: infoboxes and the category system.

A comprehensive knowledge base should know all individual entities of this world (e.g., Nicolas Sarkozy), their semantic classes (e.g., Sarkozy isa Politician), relationships between entities (e.g., Sarkozy presidentOf France), as well as validity times and confidence values for the correctness of such facts. Moreover, it should come with logical reasoning capabilities and rich support for querying. The benefits from solving this grand challenge would be enormous. Potential applications include but would not be limited to:

1. a machine-readable, formalized encyclopedia that can be queried with high precision like a semantic database;

2. an enabler for semantic search on the Web, for detecting entities and relations in Web pages and reasoning about them in expressive (probabilistic) logics;

3. a backbone for natural-language question answering that would aid in dealing with entities and their relationships in answering who/where/when/ etc. questions;

4. a key asset for machine translation (e.g., English to German) and interpretation of spoken dialogs, where world knowledge provides essential context for disambiguation;

5. a catalyst for acquisition of further knowledge and largely automated maintenance and growth of the knowledge base.

To illustrate the first two points, consider the following difficult "knowledge queries" that a student, journalist, or researcher may pose to the Web:

Q1: Which Grammy winners were born in Europe?
Q2: Which French politicans are married to singers?
Q3: Which Nobel prize winners had an academic advisor who graduated from the same university?
Q4: Give me a comprehensive list of HIV drugs that inhibit proteases (a specific family of enzymes).

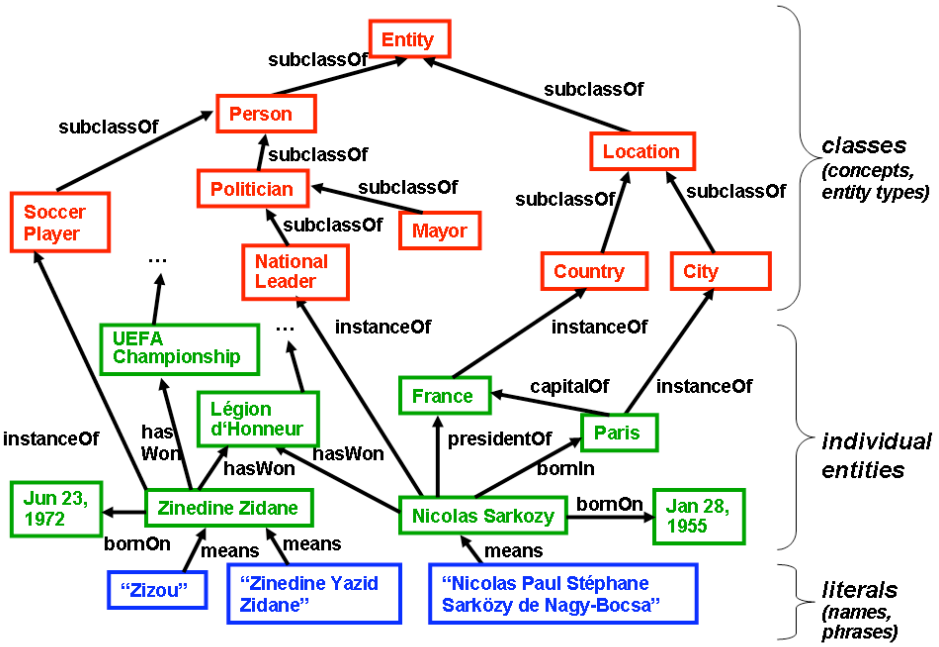

Figure 1: Excerpt of the YAGO Knowledge Base

Regardless of how well these information needs are formulated as keyword queries or question phrases, current search engines would hardly produce good answers. In the best case, the user would have to browse through many potentially relevant pages and manually compile the bits and pieces of the desired answers. A large knowledge base would support precise query formulation (not necessarily in natural language) that explicitly indicates entities and relations, and would provide the best answers in a concise manner.

This paper gives an overview of the YAGO-NAGA[1] approach to automatically building and maintaining a conveniently searchable, large and highly accurate knowledge base, by applying information-extraction (IE) methods to Wikipedia and other sources of latent knowledge. The project started in summer 2006 at the Max Planck Institute for Informatics, with continuous enhancements and extensions.

The YAGO knowledge base represents all facts in the form of unary and binary relations: classes of individual entities, and pairs of entities connected by specific relationship types. This data model can be seen as a typed graph with entities and classes corresponding to nodes and relations corresponding to edges. It can also be interpreted as a collection of RDF triples with two adjacent nodes and their connecting edge denoting a (subject, predicate, object) triple. Figure 1 shows an excerpt of the knowledge base. The knowledge base is publicly available at `http://www.mpi-inf.mpg.de/~suchanek/yago`. It can be queried for knowledge discovery by the NAGA search engine. An online demo is accessible at `http://www.mpi-inf.mpg.de/~kasneci/naga`.

Section 2 outlines the architecture of YAGO. Section 3 presents the extractor toolkit for building the YAGO core knowledge base. Section 4 presents the consistency checking methods, which ensure the high accuracy of YAGO's facts. Section 5 discusses our ongoing work on how YAGO can be automatically maintained and further grown. Section 6 presents the NAGA model and system for querying YAGO and ranking search results.

2. SYSTEM ARCHITECTURE

The YAGO architecture is depicted in Figure 2. In contrast to many other IE projects, YAGO emphasizes high accuracy and the consistency of the harvested knowledge rather than aiming for high recall (coverage) of facts. YAGO primarily gathers its knowledge by integrating information from Wikipedia and WordNet. It performs rule-based IE on the infoboxes and category system of Wikipedia, and reconciles the resulting facts with WordNet's taxonomical class system. This is done by performing consistency checks whenever a new fact is considered for addition to the knowledge base. This approach resulted in the *YAGO core knowledge base* [18, 19], currently containing 249,015 classes, 1,941,578 in-

[1]YAGO = Yet Another Great Ontology
NAGA = Not Another Google Answer

dividual entities, and about 19 million facts (instances of binary relations) for 93 different relations. Extensive sampling showed that the accuracy is at least 95 percent [19], and many of the remaining errors (false positives) are due to entries in Wikipedia itself (which we considered as ground truth).

As the rule-based core extractors are limited in coverage, YAGO can also employ pattern-, NLP- and learning-based IE techniques [1, 5, 15] on text sources such as Wikipedia texts, news articles, research literature, or Web pages of interest and clear style. These techniques, in combination with the diversity and mixed quality of the sources, introduce a higher risk of degrading in precision, and are computationally much more expensive. Therefore, the text-oriented harvesting of YAGO is carried out in two phases. The *gathering phase* employs recall-oriented IE methods, and aims at high throughput. The output is interpreted as a set of fact hypotheses. Subsequently, the *scrutinizing phase* assesses the hypotheses against the existing knowledge base, in a batched manner, and filters out facts that show high indication of being inconsistent with essential invariants and prior knowledge (e.g., that a person's birth place is unique and that certain cities are located in Europe so that an American-born person cannot be born in such a city).

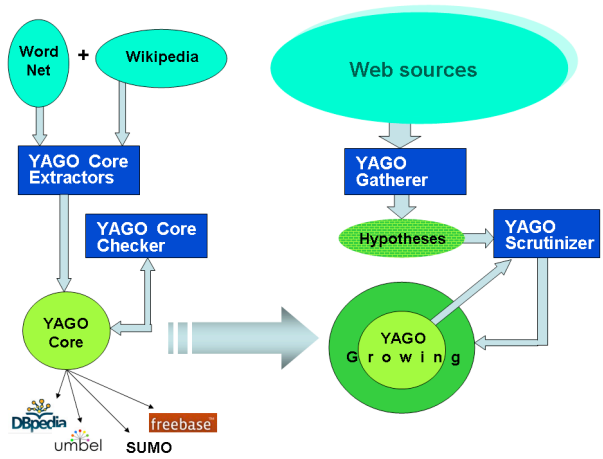

Figure 2: The YAGO Architecture

3. YAGO CORE EXTRACTORS

Wikipedia Infoboxes. Wikipedia has two kinds of - almost structured - information components on article pages that can be effectively harvested: the *infoboxes* and the *category system*. Infoboxes are collections of attribute name-value pairs. They are often based on templates and then reused for important types of entities such as countries, companies, scientists, music bands, sports teams, etc. For example, the infobox for Nicolas Sarkozy gives us data such as *birth_date = 28 January 1955*, *birth_place = Paris*, *occupation = lawyer*, and *alma_mater = University of Paris X: Nanterre*. YAGO uses a suite of rules for frequently used infobox attributes to extract and normalize the corresponding values. For example, the *spouse* attribute is mapped to the *marriedTo* relation, and the extracted fact then is *Nicolas Sarkozy marriedTo Carla Bruni*. YAGO does not attempt to extract all infobox attributes, as their "long tail" has a lot of naming diversity and noise (see [20, 21] for a more recent, universal attempt at harvesting infoboxes).

Wikipedia Categories. As for the category system, the Wikipedia community has manually placed (the article about) Nicolas Sarkozy into categories such as: *Presidents_of_France*, *Légion_d'honneur_recipients*, or *Alumni_of_Sciences_Po* (the Paris Institute of Political Studies). These give YAGO clues about instanceOf relations, and we can infer that the entity Nicolas Sarkozy is an instance of the classes PresidentsOfFrance, LégionD'HonneurRecipients, and AlumniOfSciencesPo. Occasionally, YAGO encounters a misleading category name that does not indicate a class membership (instanceOf). An example is the category *Hollywood_Walk_of_Fame*, which includes many actors and musicians. Semantically, however, these people are not instances of a class *Walk* (which would be a subclass of *Motion*), nor are they instances of some superclass *Awards* but rather awards winners which in turn would be a subclass of humans. YAGO employs linguistic processing (noun phrase parsing) and also some heuristics (e.g., the head word in the noun phrase should be in plural form), in order to cope with these pitfalls and achieve high accuracy in harvesting the category system.

Ongoing Work: Temporal Validity. The world is dynamic, and facts change with time. When we ran YAGO on Wikipedia in spring 2007, we extracted the fact *Jacques Chirac presidentOf France*, seemingly contradicting our current result *Nicolas Sarkozy presidentOf France*. To resolve this situation, we need temporal annotations for facts. To ensure that our knowledge model is not inflated with ad-hoc features, we decided to adapt the reification technique of RDF and use our standard binary-relation approach to represent validity times. This works as follows. Both of the above facts have identifiers, say *Id1* (for the fact about Chirac) and *Id2* (for the fact about Sarkozy), to which we can refer in other facts. We create additional facts like *Id1 ValidSince 17 May 1995*, *Id1 ValidUntil 16 May 2007*, and *Id2 ValidSince 17 May 2007*. This technique is general, it can also be used to represent arbitrary ternary and higher-arity relations in our model. Likewise, meta-relations such as *witnesses* for a fact - remembering sources that support a fact's validity, use the same representation.

In principle, YAGO could also handle divorces and identify spouses for specific time periods, but often this kind of information is not available in the semistructured parts of Wikipedia. Therefore, we have starting working on specialized text-oriented extractors with the specific target of detecting validity times. One problem in this task is that temporal statements often refer to relative timepoints (e.g., last Monday), have different

granularities such as "May 2007" vs. "17 May 2007", or are incomplete (e.g., showing either an "until" or "since" statement but not both, even for terminated intervals). To deal with these situations, we represent every timepoint as a pair of (earliest, latest) intervals and use plus/minus infinity for missing information in such an interval. As we find more accurate or complete time statements or gather more evidence for certain facts in the IE process, we can refine the temporal facts in the knowledge base [23].

4. YAGO CONSISTENCY CHECKERS

Rationale. YAGO pays particular attention to the consistency of its knowledge base. Solely performing IE on infoboxes and categories of Wikipedia may result in a large but incoherent collection of facts: redundant and potentially inconsistent, overly specific in some parts and left blank in others in an arbitrary way. For example, we may know that Nicolas Sarkozy is an instance of PresidentsOfFrance, but we may not be able to automatically infer that he is also an instance of Presidents and most likely also an instance of FrenchPeople. Likewise, the fact that he is a president (or minister of the interior or mayor to also include former positions) does not automatically tell us that he is a politician. To overcome these problems, YAGO intensively uses the WordNet thesaurus and integrates the facts from Wikipedia with the taxonomic backbone provided by WordNet, using techniques outlined in the rest of this section.

Class Hierarchy. YAGO initializes its class system by importing all WordNet classes and their hypernymy/hyponymy (superclass/subclass) relations. Each individual entity that YAGO discovers in Wikipedia needs to be mapped into at least one of the existing YAGO classes. If this fails, the entity and its related facts are not admitted to the knowledge base. Analogously, classes that are derived from Wikipedia category names such as PresidentsOfFrance need to be mapped, with a subclassOf relationship, to one or more proper superclasses such as Presidents or FrenchPeople. These procedures ensure that we can maintain a consistent knowledge base, where consistency eliminates dangling entities or classes and also guarantees that the subclassOf relation is acyclic. The empirically assessed accuracy of the subclassOf relation is around 97 percent.

Type Conditions. As all entities are typed by their assignment to one or more classes, binary relations have a type signature as well. For example, the isCEOof relation can be specified to have a domain BusinessPerson or simply Person and a range Company (where we denote relations as powerset functions). When the extractor produces a candidate fact like *Nicolas Sarkozy isCEOof France*, we can reject it because the second argument, France, is not a company. Similarly, the hypothesis *Nicolas Sarkozy marriedTo Élysée Palace* which may, perhaps, be incorrectly inferred from sentences such as "Nicolas Sarkozy loves his work with the Élysée Palace" (or from an incorrect interpretation of the infobox attribute Residence), is falsified by the type invariant that marriages are between persons.

Ongoing Work: Constraints on Relations. A very powerful constraint for the consistency of IE results is declaring a relation to be transitive and acyclic (or requiring that its transitive closure is acyclic). The class hierarchy is one important usage case; the locatedIn relation between geographic entities is another one. Going even further, we are considering support for functional dependencies, inclusion dependencies, and inverse relations. For example, a person can have only one birth place, which would allow us to prune out many spurious candidates for alternative birth places that we may see in Web pages. As for inclusion dependencies, we can derive *Paris locatedIn France* from the fact *Paris capitalOf France*, without necessarily seeing it in explicit form. Finally, the presidentOf fact would be an example for exploiting inverse relations. Once we accept that *Nicolas Sarkozy presidentOf France*, we could automatically infer that *France hasPresident Nicolas Sarkozy* (for the same time interval).

5. GROWING YAGO

Keeping YAGO Up-To-Date. We can maintain the YAGO knowledge base, with its current scope, by periodically re-running the extractors on Wikipedia and WordNet. It takes a few hours to build the entire knowledge base. When validity times or intervals can be properly inferred, this would retain previously compiled facts (e.g., former CEOs) as long as they do not cause any inconsistencies.

Adding Natural-Language Text Sources. Further growth of YAGO, beyond the harvesting of infoboxes and category systems, calls for text-oriented IE with more or less powerful NLP capabilities. Our own tool LEILA [17] can be employed for this purpose. It uses a dependency-grammar parser for deep parsing of natural-language sentences, with heuristics for anaphora resolution (e.g., pronouns referring to subjects or objects in a preceding sentence). This produces a tagged graph representation, whose properties can be encoded as features for a statistical learner (e.g., an SVM) that classifies fact candidates into acceptable facts vs. false hypotheses. LEILA provides reasonably good accuracy, but requires about a minute to process a long Wikipedia article on a standard PC, and works for one specific relation at a time. Simpler NLP methods, such as part-of-speech tagging (for word categories: nouns, verbs, etc.), are much faster but would have significantly lower precision. Statistics, like frequencies of witnesses, can be used to improve precision, but proper tuning of statistical thresholds is all but easy. The YAGO architecture supports all these approaches. However, with the open issues in understanding the three-way tradeoffs between precision, recall, and efficiency, we do not yet employ these techniques at large scale.

Ongoing Work. For growing YAGO we can leverage the existing YAGO core in several ways. We believe that the core contains more or less all entities of inter-

est and their most important classes. For example, all notable politicians, athletes, pop stars, movies, cities, rivers, etc. should have a Wikipedia entry, and thus are included in YAGO, too. Certainly, this does not hold for classes like computer scientists, medical drugs, etc. But we could incorporate other sources such as DBLP or UMLS, and adapt the YAGO extractors to them. With the YAGO core as semantic backbone, we can quickly test sentences, paragraphs, or entire Web pages as to whether they contain one or two interesting entities. And when aiming at a particular binary relation, we can exploit our type system: a sentence or paragraph is promising only if it contains two entities of the proper types. For example, for hypothesizing a fact of the isCEOof relation, a sentence must contain a person and a company to be worth undergoing deeper analysis. Testing the type of an entity is a fast lookup in the core knowledge base.

To preserve the consistency of YAGO when adding new facts gathered with "riskier" IE methods, we can utilize the type and constraint checkers that YAGO already has. For efficiency, we batch newly acquired facts and run the consistency checking procedures for several thousand hypotheses together, dropping those that violate vital checks or too many "soft constraints".

6. QUERYING YAGO BY NAGA

Query Language. For querying the YAGO knowledge base, we have designed a query language that builds on the concepts of SPARQL (the W3C standard for querying RDF data), but extends these capabilities by more expressive pattern matching. Our prototype system, NAGA (Not Another Google Answer) [10], implements this query language and provides a statistical ranking model for query results.

A query is a conjunction of *fact templates*, where each template would have to be matched by an edge and its incident nodes in the knowledge graph. For example, the first two example queries of Section 1 can be expressed as follows:

Q1: $x hasWonPrize GrammyAward,
$x bornIn $y,
$y locatedIn Europe
Q2: $x isa politician,
$x citizenOf France,
$x marriedTo $y,
$y isa singer

where $x and $y are variables for which we are seeking bindings so that all query patterns are matched together.

The relation names in the query can also be regular expressions, which have to be matched by an entire path in the knowledge graph. This is a powerful way of dealing with transitive relations and variants of relations where the user may not exactly know by which relations the entities of interest are connected. For example, if the *bornIn* relation actually refers to cities and the *locatedIn* relation captures a city-county-state-country hierarchy, we should replace the last condition in Q1 by the fact template *$y (locatedIn)* Europe*. And if we do not care whether the persons that we are looking for are born in Europe or are citizens of a European country, we may use the template *$y (citizenOf | bornIn | originatesFrom).(locatedIn)* Europe* instead of the last two conditions of Q1. Regular expressions are also helpful in dealing with the class hierarchy in a flexible way. In Q2 the relation label *isa* is actually a shorthand notation for the regular expression *instanceOf.(subclassOf)**, thus enabling ministers or mayors, which are subclasses of politicians, to be returned as query results.

The query language also provides support for formulating temporal conditions on the validity of the facts of interest. For example, if we want to retrieve all French presidents whose terms started in this millenium, we can phrase this as:

f: ($x presidentOf France),
(f since $t),
($t after 31 December 1999)

We are working on enhancing the query language to provide more elaborate capabilities for temporal queries. NAGA has further advanced features, most notably, for specifying relatedness queries among a set of entities [10, 11]. For example, the query:

connect (Nicolas Sarkozy, Zinedine Zidane,
Gerard Depardieu, Miles Davis)

asks for commonalities or other relationships among Sarkozy, the soccer player Zidane, the actor Depardieu, and the trumpet player Miles Davis. A possible answer (technically, a Steiner tree in the underlying knowledge graph) could be that all four are recipients of the French Légion_d'honneur order.

Ranking. Whenever queries return many results, e.g., hundreds of (mostly unimportant) politicians, we need ranking. NAGA employs a novel kind of *statistical language model (LM)* [12, 22] for this purpose, capturing the *informativeness* of a query result [10]: users prefer salient facts or interesting facts, e.g., Nicolas Sarkozy and not the mayor of a small provincial town. In addition, we need to consider the *confidence* that the result facts are indeed correct. Our IE methods assign a confidence weight to each fact f in the knowledge base based on the empirically assessed goodness of the extractor and the extraction target (e.g., rule-based for birthdates vs. linguistic for birth places) and the trustworthiness of the fact's witnesses s (i.e., sources from which it was extracted). One possible way of combining these aspects (among various options) would be:

$$confidence(f) = max\ \{accuracy(f,s) \times trust(s) \mid s \in witnesses(f)\}$$

where trustworthiness could be based on PageRank-style authority or on empirical assessment by experts (e.g., high for Wikipedia, low for blogs). The confidence in a query-result graph that consists of multiple facts is the product of the individual facts' confidence values, postulating statistical independence among the facts.

For informativeness, NAGA employs an LM for graph-structured data. In the following we give a simplified

explanation of the model introduced in [10]. Conceptually, we construct a statistical model for each possible result graph g with connected edges (facts) g_i, and consider the probability that the query q, consisting of fact templates q_i, was generated from g:

$$P[q|g] = \prod_i \lambda P[q_i|g_i] + (1-\lambda)P[q_i]$$

where we factorize over edges for tractability and use $P[q_i]$ for smoothing with parameter λ (analogously to standard LM's). Applying Bayes' rule, simplifying the resulting expression and omitting sub-expressions that do not influence the ranking of results, we obtain:

$$P[q|g] \sim \prod_i \frac{P[q_i|g_i]}{P[q_i]}$$

We can interpret $1/P[q_i]$ as an idf-style weighting of the individual subqueries (emphasizing the more selective patterns in the query). The main parameters to be estimated are the $P[q_i|g_i]$ values, which reflect informativeness for the given query. We use a "background corpus" for this purpose, either a large Web sample or the entirety of Wikipedia texts. We compute the number of witnesses for g_i, that is, the frequency of the two entities (or classes) in g_i co-occurring (in the same sentence, paragraph, or Web page). Analogously, the number of witnesses for q_i is the frequency of the non-variable parts of q_i occurring together. Our current implementation precomputes these statistics based on the Wikipedia corpus. With these ingredients we can finally set

$$P[q_i|g_i] \approx \frac{\#witnesses(g_i)}{\#witnesses(q_i)}$$

For example, as partial results to query Q1, famous Grammy winners such as Eric Clapton, Phil Collins, or Enya should be among the highest ranked results.

For the overall scoring of query results, NAGA uses a weighted linear combination of informativeness and confidence:

$$score(q,g) = \alpha \prod_i \frac{P[q_i|g_i]}{P[q_i]} + (1-\alpha) \prod_i confidence(g_i)$$

Ongoing Work: Personalization. The notion of informativeness is, strictly speaking, a subjective measure: an individual user wants to see a salient result that is also interesting to her. This calls for a *personalized ranking model* or at least a user-group-specific model. An elegant property of the LM approach pursued in NAGA is that we can easily compose multiple LM's using a probabilistic mixture model. We can estimate parameters of a user- or community-specific LM and combine this with a global LM, both models using the same mathematical structure but different parameters.

For the personalized LM, we monitor the history of queries and browsing interactions on the online knowledge base. A click on a fact is interpreted as positive feedback that the fact is interesting to the user, and this evidence is spread to the graph neighborhood, with exponential decay and attention to the edge types along which propagation is meaningful [7]. As an example, assume that a user has intensively explored epic movies and orchestral music, and then poses query Q1. The personalized ranking would prioritize European film-music composers such as Ennio Morricone, Hans Zimmer, or Javier Navarrete.

7. CONCLUSION

The YAGO core is publicly available and has been imported into and integrated with various other knowledge-management projects including DBpedia (`dbpedia.org`), SUMO (`www.ontologyportal.org`), UMBEL (`umbel.org`), and Freebase (`www.freebase.com`). Our ongoing work to improve YAGO mostly centers around making it larger while retaining its high accuracy. This entails deeper considerations on scalability issues, for example, by utilizing database-style query processing and optimization techniques, along the lines of [9].

YAGO shares many of its goals and methodologies with parallel projects along related lines. These include Avatar [14], Cimple/DBlife [6, 16], DBpedia [2], KnowItAll/TextRunner [8, 3, 4], Kylin/KOG [20, 21], and the Libra technology [13, 24] (and probably more). Together they form an exciting trend of leading research towards the elusive goal of machine-readable, comprehensive knowledge bases.

8. REFERENCES

[1] Eugene Agichtein: Scaling Information Extraction to Large Document Collections. IEEE Data Eng. Bull. 28(4), 2005

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, Zachary G. Ives: DBpedia: A Nucleus for a Web of Open Data. ISWC/ASWC 2007

[3] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, Oren Etzioni: Open Information Extraction from the Web. IJCAI 2007

[4] Michael J. Cafarella, Christopher Re, Dan Suciu, Oren Etzioni: Structured Querying of Web Text Data: A Technical Challenge. CIDR 2007

[5] Hamish Cunningham: An Introduction to Information Extraction. In: Encyclopedia of Language and Linguistics, 2nd Edition, Elsevier, 2005

[6] Pedro DeRose, Warren Shen, Fei Chen, AnHai Doan, Raghu Ramakrishnan: Building Structured Web Community Portals: A Top-Down, Compositional, and Incremental Approach. VLDB 2007

[7] Minko Dudev, Shady Elbassuoni, Julia Luxenburger, Maya Ramanath, Gerhard Weikum: Personalizing the Search for Knowledge. PersDB 2008.

[8] Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen

Soderland, Daniel S. Weld, Alexander Yates: Unsupervised Named-Entity Extraction from the Web: An Experimental Study. Artif. Intell. 165(1), 2005

[9] Panagiotis G. Ipeirotis, Eugene Agichtein, Pranay Jain, Luis Gravano: Towards a Query Optimizer for Text-Centric Tasks. ACM Trans. Database Syst. 32(4), 2007

[10] Gjergji Kasneci, Fabian M. Suchanek, Georgiana Ifrim, Maya Ramanath, Gerhard Weikum: NAGA: Searching and Ranking Knowledge. ICDE 2008

[11] Gjergji Kasneci, Maya Ramanath, Mauro Sozio, Fabian M. Suchanek, Gerhard Weikum: STAR: Steiner Tree Approximation in Relationship-Graphs. ICDE 2009

[12] Xiaoyong Liu, W. Bruce Croft: Statistical Language Modeling for Information Retrieval. Annual Review of Information Science and Technology 39, 2004

[13] Zaiqing Nie, Yunxiao Ma, Shuming Shi, Ji-Rong Wen, Wei-Ying Ma: Web Object Retrieval. WWW 2007

[14] Frederick Reiss, Sriram Raghavan, Rajasekar Krishnamurthy, Huaiyu Zhu, Shivakumar Vaithyanathan: An Algebraic Approach to Rule-Based Information Extraction. ICDE 2008

[15] Sunita Sarawagi: Information Extraction. Foundations and Trends in Databases 2(1), 2008

[16] Warren Shen, AnHai Doan, Jeffrey F. Naughton, Raghu Ramakrishnan: Declarative Information Extraction Using Datalog with Embedded Extraction Predicates. VLDB 2007

[17] Fabian M. Suchanek, Georgiana Ifrim, Gerhard Weikum: Combining Linguistic and Statistical Analysis to Extract Relations from Web Documents. KDD 2006

[18] Fabian M. Suchanek, Gjergji Kasneci, Gerhard Weikum: YAGO: a Core of Semantic Knowledge. WWW 2007

[19] Fabian Suchanek, Gjergji Kasneci, Gerhard Weikum: YAGO: A Large Ontology from Wikipedia and WordNet. Journal of Web Semantics, 2008

[20] Fei Wu, Daniel S. Weld: Autonomously Semantifying Wikipedia. CIKM 2007

[21] Fei Wu, Daniel S. Weld: Automatically Refining the wikipedia Infobox Ontology. WWW 2008

[22] ChengXiang Zhai, John D. Lafferty: A risk minimization framework for information retrieval. Inf. Process. Manage. 42(1), 2006

[23] Qi Zhang, Fabian M. Suchanek, Lihua Yue, Gerhard Weikum: TOB: Timely Ontologies for Business Relations. WebDB 2008

[24] Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, Wei-Ying Ma: Simultaneous Record Detection and Attribute Labeling in Web Data Extraction. KDD 2006