YAGO2s: Modular High-Quality Information Extraction with an Application to Flight Planning

Fabian M. Suchanek, Johannes Hoffart, Erdal Kuzey, Edwin Lewis-Kelham
Max Planck Institute for Informatics, Germany

Abstract: In this paper, we present YAGO2s, the new edition of the YAGO ontology [SKW07, HSBW12]. The software architecture has been refactored from scratch, yielding a design that modularizes both code and data. This modularization enables us to add in new data sources more easily, while still maintaining the high accuracy and coherence of the ontology. Thus, we believe that YAGO2s occupies a sweetspot between a centralized design and a completely distributed design.

In this demo, we present an application of this design to the task of planning a flight. Our proposed system finds flights between all airports close to the departure city to all airports close to the destination city.

1 Knowledge Base Construction

In recent years, many projects have successfully created large-scale knowledge bases (KBs) in an automated fashion. The KBs contain millions of entities (such as rivers, universities, people, and movies), and millions of facts about them (such as who acted in which movie, which river is located in which country, etc.). There are several strategies to build such KBs. One strategy is to accumulate and reconcile as much data as possible. Projects such as TextRunner [BCS$^+$07], and NELL [CBK$^+$10] follow this strategy, as did YAGO [SKW07, HSBW12]. The advantage of this method is that the knowledge is largely coherent, because it is curated by a single provider. The drawback is that the project becomes more and more difficult to maintain as more resources are integrated, and as more people work on it. An alternative approach is to furnish only small pieces of data, and to interlink these. This is the approach favored by the Semantic Web community as *Linked Open Data*. The advantage of this strategy is that individual datasets are curated by their respective owners, thus modularizing the data and distributing the work. The drawback is that it is challenging to establish links between the data sets. Thus, the quality of the cross-ontology data is often not perfect [HHM$^+$10, DSSM10]. The DBpedia ontology [ABK$^+$07] is pursuing another approach: It relies on information extraction, but outsources some of the manual tasks to a community. This has the advantage of distributing the work. At the same time, it can lead to slight incoherences [HSBW12], and nothing is known about the accuracy of the data in DBpedia.

The YAGO project started in 2007 by combining WordNet [Fel98] and Wikipedia to a coherent general-purpose KB. Thus, YAGO was in the camp of the centralized KBs. This allowed it to enforce accuracy and coherence on its data. Every entity in YAGO and every relationship is unique. Manual evaluations proved [SKW07, HSBW12] that the probability that a given statement in YAGO is correct stands at 95%. This focus on precision is the main characteristics of YAGO in the realm of automatically constructed KBs.

In recent years, more and more people have joined the YAGO team, and more and more resources have been integrated into the KB. This yielded YAGO2 [HSBW12]. Today, YAGO is driven by a core team of 5 people, with around a dozen more people working on directly related projects. With more people joining, and now a small research group dedicated to ontology development, the centralized mode was no longer sustainable. However, a distributed mode in the spirit of the Semantic Web or a community-based approach makes it harder to achieve the data quality or the coherence of YAGO2. Therefore, we have opted for a middle course, coined YAGO2s ("YAGO 2 star"). This new framework is based on the data sources and the extraction mechanisms of YAGO2 [HSBW12]. However, the entire software architecture for the new YAGO2s has been reengineered from scratch.

2 The YAGO2s Architecture

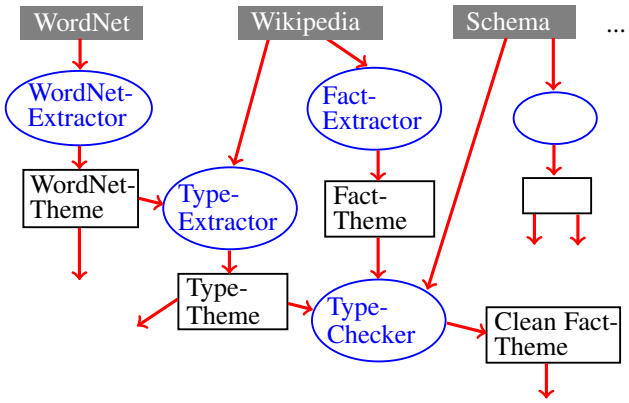

Figure 1: The YAGO2s Architecture

We have completely refactored the ontology extraction framework into a modular structure (Figure 1). There are 39 extractor modules. Each module receives a data source as input. Input sources are WordNet [Fel98], Wikipedia, WordNet Domains [BFMP04], the Universal WordNet [dMW09], and Geonames. Others can be added. Each extractor produces one or multiple *themes* as output. A theme is a set of RDF triples. For example, the module "WordNet-Extractor" receives WordNet as an input source, and produces an output theme called "WordNet-Theme". This output theme contains RDF triples extracted from WordNet. The data sources can also be files that contain handwritten data. The schema of YAGO, e.g., which defines the relations with their domains and ranges, is defined manually in YAGO. Such data can be supplied to the modules as input sources. This ensures that all modules operate on the same predefined schema. Modules can also take other themes as input. In the figure, the module "Type-Extractor" requires the theme "WordNet-Theme" as input theme. The module uses data from WordNet and Wikipedia to build the YAGO type system ("Type-Theme"). This theme can become again the input of other modules. This yields a dependency graph of extraction modules.

Modules can be added or removed ad libitum, as long as the dependencies are respected. In order to guarantee the data quality, we provide some modules that check the output themes

of other modules. The "Type-Checker", e.g., filters out statements that do not conform to the domain and range constraints. The output theme of the type checker is a cleaned theme, which following modules can use as input. A Deduplicator Module takes a similar role, deduplicating statements and entities. A Rule Module applies the deduction rules of YAGO2 to deduce implied facts (such as facts induced by symmetric relations). These modules ensure that every statement that makes it into the final YAGO themes has been deduplicated, and checked for type coherence. A revision checker signals if statements were extracted in a previous run of the system on a previous version of the data sources, but went missing in the current run. A sample of these statements can then be checked manually to see, e.g., if Wikipedia infobox attribute names have changed.

This architecture has a number of distinct advantages: First, it modularizes the information extraction process. Every team member can be responsible for one or multiple modules. Second, it modularizes the data. YAGO2s will be made available in theme slices, so that users can download just the themes they desire. Third, this architecture allows for efficient parallelization. Our scheduling system runs modules that do not depend on each other in parallel. Fourth, different from a truly distributed approach, our architecture helps keeping the data coherent by allowing the easy re-use of data-cleaning components across various sources. This is achieved by a predefined schema as input, and the checker modules that verify the output. This way, our architecture implements a controlled trade-off between a centralized approach and a distributed approach.

The native data format of YAGO2s is now Turtle. The fact identifiers of YAGO, which are used to attach time and space information to facts, are stored in the files in a commented line before the fact. This allows standard RDF engines to consume YAGO themes without the fact identifiers. We have also made YAGO's terminology fully RDF and OWL compliant. In addition, the new data set contains a theme with WordNet Domains [BFMP04], which give a topic structure to YAGO. Thus, it is now possible to ask for all entities related to, e.g., "geography". We have re-evaluated the ontology by manual sampling, in the same way as described in [HSBW12]. Overall, we evaluated over 3700 facts. Our evaluation confirmed again a fact accuracy of 95%. YAGO2s is available at `http://yago-knowledge.org`.

3 An Application: Flight Planning

The new architecture of YAGO2s makes it easier to integrate new data sources in a controlled environment. To demonstrate this, we added a new module to the framework that provides information about flights. This information can be extracted from Wikipedia pages about airports. These pages contain tables with the names of the flight companies that operate at the airport, together with their destination airports. We designed a simple information extractor that reads out these tables from Wikipedia. This piece of code acts as a module in our framework. Its output is checked by the type checker and the other checker modules, thus ensuring smooth integration with the rest of YAGO2s.

In this demo, we show how useful the new YAGO2s is with this module. Many commercial Web sites allow searching and booking flights. However, the user usually has to specify the departure airport and the destination airport. If the user lives close to several airports, this can lead to a combinatorial problem. As an example, take a user based in Saarbrücken,

a small city in the West of Germany. Assume that he wishes to go to the Ligurian cost, in Italy. Assume also that he is willing to make a trip of up to 3 hours to the airport. Then there are at least 11 airports that could serve as departure airports.[1] There are also at least 6 airports that are close to the Ligurian cost.[2] This yields a total of 66 possible connections from departure airports to destination airports. With current services, the user has to try out all of them until there is a suitable match. This process can easily take several days (as one author of this paper experienced).

With YAGO2s, this task can be simplified. YAGO2 contained already many airports and cities, together with their coordinates. With the new flight information module, YAGO2s knows also which companies offer flights between which airports. This allowed us to build a system that can suggest flight connections to the user. The user simply enters the city of departure and the city of arrival. Our system seeks all airports within a predefined distance of the departure city, and finds all direct flights to airports in the vicinity of the target city. Then the system displays all connections, along with their trajectory on a map. A search for "Saarbrücken to Genova", e.g., yields 4 possible flight connections, with 3 different airlines. We link these to the Web page of a travel company, so that the user can check flight availability and book the flight. This way, YAGO2s acts as an entrance portal to the commercial service, giving a true added value to the user. Our demo is available at `http://www.mpi-inf.mpg.de/yago-naga/yago/flights.html`.

References

[ABK+07] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In *ISWC*, 2007.

[BCS+07] M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open Information Extraction from the Web. In *IJCAI*, 2007.

[BFMP04] L. Bentivogli, P. Forner, B. Magnini, and E. Pianta. Revising WordNet Domains Hierarchy. In *COLING Workshop on Multilingual Linguistic Resources*, 2004.

[CBK+10] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka Jr., and T. Mitchell. Toward an Architecture for Never-Ending Language Learning. In *AAAI*, 2010.

[dMW09] Gerard de Melo and Gerhard Weikum. Towards a Universal Wordnet by Learning from Combined Evidence. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009)*, pages 513–522, New York, NY, USA, 2009. ACM.

[DSSM10] L. Ding, J. Shinavier, Z. Shangguan, and D. McGuinness. SameAs Networks and beyond: Analyzing deployment status and implications of owl:sameAs in Linked Data. In *ISWC*, 2010.

[Fel98] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[HHM+10] H. Halpin, P. Hayes, J. McCusker, D. McGuinness, and H. Thompson. When owl:sameAs isn't the Same: An Analysis of Identity in Linked Data. In *ISWC*, 2010.

[HSBW12] J. Hoffart, F. Suchanek, K. Berberich, and G. Weikum. YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence Journal*, 2012.

[SKW07] F. Suchanek, G. Kasneci, and G. Weikum. YAGO: A Core of Semantic Knowledge. In *WWW*, 2007.

[1] Ensheim, Zweibrücken, Hahn, Karlsruhe, Frankfurt, Stuttgart, Luxembourg, Strasbourg, 3 airports in Paris.
[2] Nice, Genova, Turin, Milan, Bergamo, Pisa.