YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages

Johannes Hoffart Fabian M. Suchanek* Klaus Berberich
Edwin Lewis-Kelham Gerard de Melo Gerhard Weikum
Max Planck Institute for Informatics, Germany *INRIA Saclay, France
{jhoffart,kberberi,edwin,gdemelo,weikum} @ mpi-inf.mpg.de fabian@suchanek.name

ABSTRACT

We present YAGO2, an extension of the YAGO knowledge base with focus on temporal and spatial knowledge. It is automatically built from Wikipedia, GeoNames, and WordNet, and contains nearly 10 million entities and events, as well as 80 million facts representing general world knowledge. An enhanced data representation introduces time and location as first-class citizens. The wealth of spatio-temporal information in YAGO can be explored either graphically or through a special time- and space-aware query language.

Categories and Subject Descriptors

H.1 [**Information Systems**]: Models and Principles

General Terms

Algorithms

Keywords

Knowledge Base, Ontology, Temporal, Geo-Spatial, Textual, Multilingual

1. INTRODUCTION

In recent years, the success of Wikipedia and algorithmic advances in information extraction have revived interest in large-scale knowledge bases. Notable endeavors of this kind include DBpedia [1], KnowItAll [2], WikiTaxonomy [8], and YAGO [9], as well as commercial services like freebase.com, trueknowledge.com, and wolframalpha.com. These describe many millions of individual entities, their mappings into semantic classes, and relationships between entities. Most of them represent facts in the form of subject-predicate-object (SPO) triples following the RDF data model.

The RDF model has been enriched with means of storing additional fields [5, 7, 10], however current knowledge bases are still mostly blind to the temporal dimension. They may store birth dates and death dates of people, but remain oblivious of the fact that this yields a time span demarcating the person's existence and the events that this person can participate in. They are also largely unaware of the temporal properties of events. For example, they may store that a person is the president of some country, but heads of state or CEOs of companies change. Similar problems can be observed at the spatial level. Purely entity-centric representations know locations and their `locatedIn` relations, but do not consistently attach a geographical dimension to events and entities. The geographical location is a crucial property not just of physical entities such as countries, mountains, or rivers. Organization headquarters, or events such as battles, fairs, and people's births all have a spatial dimension.

What we need is a comprehensive anchoring of current ontologies along both the geo-spatial and the temporal dimension. Such a new knowledge base would be able to understand the time dimension of a phrase such as "the era of Elizabeth I". It would also be able to understand expressions that have multiple dimensions. The term "Woodstock", for instance, conveys not just a time (1969), but also a place (White Lake) and a duration (a few days). A time- and space-aware knowledge base could correctly locate this event on both dimensions. We could for example ask for "all musicians born in the vicinity of Woodstock".

This paper presents the YAGO2 knowledge base, a major new edition of the YAGO knowledge base, with focus on a built-in space- and time-awareness. We have developed an extensible approach to fact extraction, which gathers and integrates temporal, spatial and semantic information from Wikipedia, WordNet, and GeoNames. Moreover, our extractors also catch keywords associated to entities from the Wikipedia articles and incorporate multilingual information. The resulting knowledge base is publicly available at `www.mpi-inf.mpg.de/yago-naga/yago/`. It contains more than 80 million facts for 9.8 million entities, plus 76 million keywords. Sampling-based manual assessment of over 7000 facts shows that YAGO2 has a precision of 95%. Detailed results are available in [6].

This demo allows querying and visualizing the wealth of data in YAGO2. For this purpose, we have extended the classical subject-predicate-object knowledge representation by three more dimensions, a geographical, a temporal, and a keyword dimension, so as to provide a convenient way of browsing and querying the new data. Users can visualize the events and entities on a map and on a synchronized timeline, allowing them to zoom and scroll on both the temporal and the geographical dimension of our planet.

2. EXTRACTION ARCHITECTURE

The YAGO knowledge base was originally introduced in [9]. It combined entities from Wikipedia with concepts from WordNet by exploiting the category system and the in-

*Work was partially funded by the ERC grant Webdam.

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0637-9/11/03.

foboxes of Wikipedia and joining them with the taxonomy of WordNet. YAGO relies on about 100 manually defined relations. In the first version of YAGO, much of the extraction was done by hard-wired rules in the source code. In order to have a more extensible design, we have completely re-engineered the code, and the new architecture is based on declarative rules. The rules take the form of subject-predicate-object triples, so that they are basically additional YAGO2 facts. Indeed, the rules themselves are a part of the YAGO2 knowledge base. There are different types of rules.

Factual rules are additional facts for the YAGO2 knowledge base. They are declarative translations of all the manually defined exceptions and facts that the previous YAGO code contained. These include the definitions of all relations, their domains and ranges, and the definition of the classes that make up the YAGO hierarchy of literal types.

Implication rules express that if certain facts appear in the knowledge base, then another fact shall be added. Whenever the YAGO2 extractor detects that it can match facts to the templates of the subject, it generates the fact that corresponds to the object and adds it to the knowledge base. Thus, implication rules have the expressive power of domain-restricted Horn rules.

Replacement rules say that if a part of the source text matches a specified regular expression, it should be replaced by a given string. This takes care of interpreting microformats, cleaning up HTML tags, and normalizing numbers.

Extraction rules state that if a part of the source text matches a specified regular expression, a sequence of facts shall be generated. These rules apply primarily to patterns found in the Wikipedia infoboxes, but also to Wikipedia categories, article titles, and even other regular elements in the source such as headings, links, or references.

3. TEMPORAL DIMENSION

The meta-physical characteristics of time and existence have been the subject of intense debate since the beginning of philosophy. For YAGO2, we can choose a more pragmatic approach to time, because we can derive the temporal properties of objects from the data in the knowledge base.

Many entities come into existence at a certain point of time and cease to exist at another point of time. People, for example, are born and pass away. Countries are created and dissolved. Other entities come into existence, but never cease to exist. This applies to abstract creations such as pieces of music, scientific theories, or literature works.

Instead of manually considering each and every entity type as to whether time spans make sense or not, we focused on the following four major entity types with the relations that indicate their time span: **People** (with `wasBornOnDate` and `diedOnDate`), **groups** (such as music bands, football clubs, universities, or companies; with `wasCreatedOnDate` and `wasDestroyedOnDate`), **artifacts** (such as buildings, songs or cities; with `wasCreatedOnDate` and `wasDestroyedOnDate`) and **events** (such as sports competitions like the Olympics, or named epochs like the "German autumn"; with the relations `startedOnDate` and `endedOnDate`). We believe that these four types cover almost all of the cases where entities have a meaningful time of existence. They make up 76% of all entities in YAGO2. We introduce two generic *entity-time relations*: `startsExistingOnDate` and `endsExistingOnDate`, which define the temporal start point and end point of an entity, respectively. The type-specific relations `wasBornOnDate`, `diedOnDate` etc. are made sub-properties of the two generic relations, so that all time-dependent entities have a generic existence span. Our infrastructure creates time spans for all entities where it can deduce such information from Wikipedia.

Facts, too, can have a temporal dimension. For example, `BobDylan wasBornIn Duluth` is an event that happened in 1941. The fact `BarackObama holdsPoliticalPosition PresidentOfTheUnitedStates` denotes the epoch from the time Obama was elected until either another president is elected or Obama resigns. The YAGO2 extractors can find occurrence times of facts from the Wikipedia infoboxes. For example, spouses are often mentioned with the date of marriage and divorce. The extractors reify the time-dependent fact by giving it an identifier (such as, e.g. `#42`). Then, they create two meta-facts, `#42 occursSince` *`date`* and `#42 occursUntil` *`date`*, which define the start and end time point of the occurrence time, respectively.

In some cases, the entities that appear in a fact may indicate its occurrence time. For example, for `BobDylan wasBornIn Duluth`, it seems natural to use Dylan's birth date as the occurrence time. For `BobDylan created BlondeOnBlonde`, it should be the creation time of the object. To this end, we make all relations that indicate the creation of its subject (such as `wasBornIn`) instances of the new class `subjectStartRelation`. Likewise, we make all relations that indicate the creation of its object (such as `created`) instances of `objectStartRelation`. Then, it suffices to introduce an implication rule to propagate the start of the existence time of the entity to the occurrence time of the fact.

Thus, the YAGO2 framework assigns begin and/or end of time spans to all entities, to all facts, and to all events, if they have a known start point or a known end point. If no such time points can be inferred from the knowledge base, no assignment is attempted. This conservative approach leaves some time-dependent entities without a time scope, but never assigns an ill-defined time to an entity. Concrete time points are typically denoted with a resolution of days but sometimes with cruder resolution like years.

4. GEO-SPATIAL DIMENSION

All physical objects have a location in space. For YAGO2, we are concerned with entities with a permanent spatial extent on Earth – for example countries, cities, and rivers. In the original YAGO (and in WordNet), such entities have no common super-class. Therefore, we introduce a new class `yagoGeoEntity` that groups together all such *geo-entities*. The position of a geo-entity can be described by geographical coordinates, consisting of latitude and longitude. We introduce a special data type to store geographical coordinates, `yagoGeoCoordinates`, and link each geo-entity to its coordinates by the `hasGeoCoordinates` relation[1].

YAGO2 harvests geo-entities from two sources. The first is Wikipedia, which contains a large number of cities, mountains, lakes, etc., many of which come with associated geographical coordinates. Our extraction framework retrieves coordinates for 176,474 geo-entities.

However, not all geo-entities in Wikipedia are annotated

[1]For locations that have a physical extent, the assignment of coordinates follows the rules from Wikipedia, `http://en.wikipedia.org/wiki/Wikipedia:WikiProject_Geographical_coordinates`

with geographical coordinates, and of course there are many geo-entities not covered by Wikipedia at all. Therefore, we tap into an even richer source of geographical data: GeoNames (`geonames.org`), which describes more than 7 million locations. GeoNames classifies locations, assigning each one a class, e. g. `Berlin` is a "capital of a political entity". Furthermore, GeoNames contains `locatedIn` hierarchies, e. g. `Berlin` is located in `Germany`, which is in `Europe`. These `locatedIn` facts model the notion of one entity being contained within another, something which is not possible with geographical coordinates alone.

All this data is a valuable addition to YAGO, so we make an effort to integrate it as completely as possible. We need to match the individual geo-entities that exist both in Wikipedia and GeoNames in order to avoid duplicates. Our matching algorithm takes into account not only the textual similarity between the entity names, but also the geographic coordinates where such information is available.

We also integrate GeoNames classes, but to avoid duplication, we have to match them to existing YAGO classes. There is prior work that aligns GeoNames classes with WordNet, called GeoWordNet [4], which however relies on manual curation. This approach is time-intensive and fragile when either GeoNames or WordNet change in future releases. To counter this problem, we devised an automated matching algorithm based on shallow noun phrase parsing of the original YAGO to detect and match candidate class names. The matching is further improved by knowledge about which classes in YAGO2 have geographical meaning and are thus valid candidates (those grouped under the newly created `yagoGeoEntity`), and by measuring the similarity of the gloss of both GeoNames and YAGO2 classes.

This matching process augments YAGO2 with nearly 7 million geo-entities and nearly 50 million new facts from GeoNames, in particular adding geographical coordinates that could not be extracted from Wikipedia, which renders more entities accessible by spatial queries.

We deal with the spatial dimension similarly to how we deal with time: a location is assigned to both entities and facts, wherever this is ontologically reasonable and can be deduced from the data. The location of facts and entities is given by a geo-entity, which we can extract from Wikipedia. For example, the location of *Woodstock* is White Lake, NY, which is an instance of `yagoGeoEntity`. We have spatial data in our knowledge base for **events** that took place at a specific location, e.g. battles and sports competitions, **groups** or organizations that have a seat, e.g. company headquarters or university campus locations, and **artifacts** with a physical location, e.g. the Mona Lisa in the Louvre.

Some facts have a spatial dimension as well. For example, the fact that Leonard Cohen was born in 1934 happened in his city of birth, Montreal. Naturally, not all facts have a spatial dimension, e.g. schema-level facts such as `subclassOf` and identifier relations such as `hasISBN` have no location. We introduce the relation `occursIn`, which holds between a (reified) fact and a geo-entity. For example, if we have the fact `#1: LeonardCohen bornOnDate 1934`, we would write `#1 occursIn Montreal`. Again, the key to a semantically clean treatment of the spatial dimension lies in the relations.

Some facts occur in a place that is indicated by their subject or object. For example, the fact that Jimi Hendrix was born in Seattle happened in Seattle. We introduce two new classes to describe such relations, `relationLocatedByObject` and `relationLocatedBySubject`, which are both subclasses of `yagoRelation`. The first class combines relations whose location is given by the location of their object. These include for example `wasBornIn`, `diedIn`, `worksAt`, and `participatedIn`. The second class groups relations whose location is given by the subject. These include, for example, `hasMayor`. We can transfer the location of the fact argument to the fact itself by an implication rule.

Some relations occur in tandem: One relation determines the location of the other. For example, `wasBornOnDate` defines the time of the corresponding `wasBornIn` fact, and the latter defines the location of the former. We express this situation by the relation `timeToLocation`, which holds between two relations. The first relation specifies the time of the event while the second specifies the location. Other examples include the pairs `diedOnDate`/`diedIn` and `happenedOnDate`/`happenedIn`. We introduce implication rules that transfer the location from one relation to the other.

Together, these rules derive a location for a fact whenever this is semantically meaningful.

5. TEXTUAL DIMENSION

For each entity, YAGO2 contains *contextual information*, which is gathered by our extractors from Wikipedia. They include the relations `hasWikipediaAnchorText` (linking an entity to a string that occurs as anchor text in the entity's article), `hasWikipediaCategory` (holds the name of a category in which Wikipedia places the article), `hasCitationTitle` (holds the title of a reference on the Wikipedia page). All of these are sub-properties of the relation `hasContext`, thus providing a wealth of keywords associated to the entity.

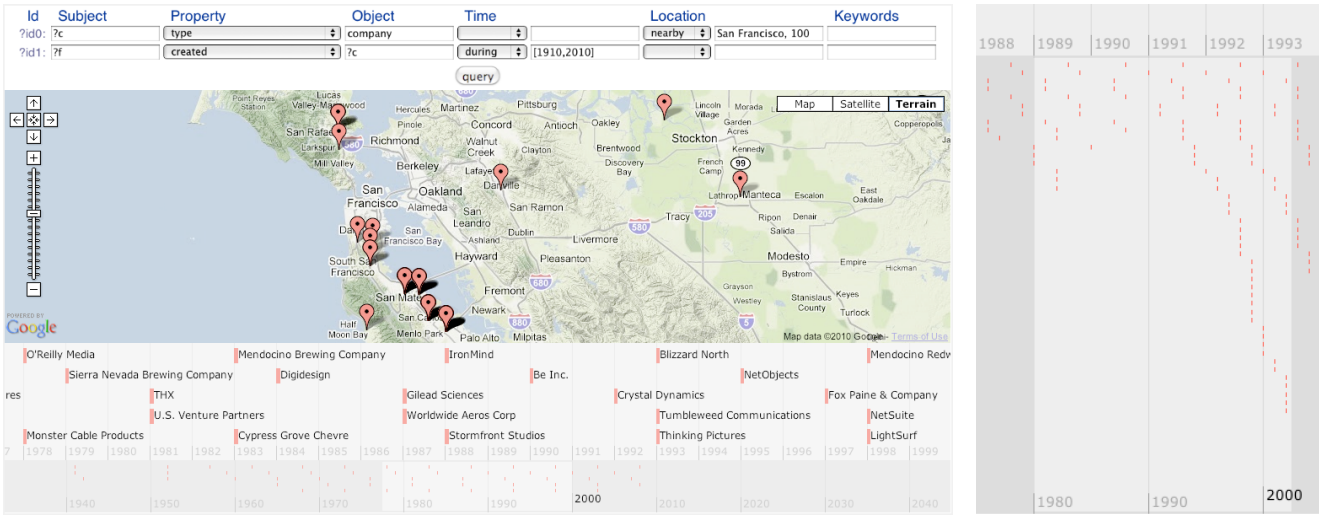
For individual entities, we extract multilingual translations from inter-language links in Wikipedia, allowing us to query individuals using non-English names. Multilingual class names are obtained by integrating UWN [3] into YAGO. UWN maps words and word senses of WordNet to their counterparts in other languages, providing over 1.5 million translations and sense assignments for 800,000 words in over 200 languages at a precision of 90%. Overall, this gives us multilingual names for most entities and classes in YAGO and enables multilingual querying.

6. DEMO

To facilitate the querying of YAGO, we have added to each triple of **S**ubject, **P**redicate, and **O**bject in YAGO2 three more dimensions: **T**ime, **L**ocation, and conte**X**t. This yields a 6-tuple representation, which we call *SPOTLX*. SPOTLX is obtained by joining in the occurrence spans of facts, the locations of facts, and the context of the involved entities. Our interface shows a SPARQL-like query form for SPOTLX tuples. If results contain events or entities with an associated timespan, they are visualized on an interactive map with a time line. When the query changes, the information is updated. Users may also zoom and scroll in the map or timeline, which updates the query. For the visualization, we use TimeMap (`http://timemap.googlecode.com`). Combining the new dimensions, we can solve different kinds of queries.

Entities in Time and Space. Adding the time dimension, we can ask for companies founded in the last 100 years:

```
?c type company .
?f created ?c during [1910,2010]
```



(a) Visualized on timeline and map (b) Companies of the web era

Figure 1: Our demo showing some companies founded in the Bay Area during the last century

We can refine this query by adding a restriction on the location where the company was founded, e. g. `nearby San-Franciso`.

Our interface visualizes the retrieved entities on a timeline accompanied by a map (Figure 1(a)). In the screenshot, we have zoomed in on the Bay Area, and the markers show locations where companies have been founded between 1978 and 2000. The visualization shows that the late 80s and early 90s saw a slowdown in the founding of companies, followed by a sharp rise in the wake of the web era (Figure 1(b)). Instead of specifying the time directly with an interval in the form of `[1910,2010]`, the name of an event can be used, e. g. `World War II`, which will automatically be resolved to the entity's occurrence time.

Relationships among Entities. Relationships change over time – an example is the continuously increasing rate of divorces. YAGO2 captures time intervals for marriages and knows, for example, that Nicolas Sarkozy was married to Cécilia Attias from 1996 to 2007, and to Carla Bruni from 2008. The timeline visualization shows his date of divorce coinciding with his election as President of France.

Multilingual Queries. The French terms "marquis" and "Guerres de la Révolution française" can be used to find marquesses born during the French Revolutionary Wars. This is particularly helpful when non-English words like "comte" are less ambiguous than their English counterparts ("count").

Combining Facts and Text. Sometimes it is convenient to use keywords associated with entities to make a query more specific. Take the above query about companies founded in San Francisco: Despite the restriction of time and location, it is still quite broad, as a large number of companies have been founded there. We might be interested in those that are *not* involved in the web business. We can specify this by adding the negated keyword `!web` to make the query more specific. Using keywords is necessary if the knowledge base doesn't contain the specific fact one is looking for, or simply to make the querying more convenient.

7. CONCLUSION

We have developed a methodology for enriching large knowledge bases of entity-relationship-oriented facts along the dimensions of time and space, and have demonstrated its practical viability by presenting YAGO2, which comprises more than 80 million facts of near-human quality. This makes YAGO2 a valuable gazetteer, not just for geographical, but also for temporal and semantic data in general. We believe that such spatio-temporal knowledge is a crucial asset for many applications including entity linkage across independent sources and Semantic Search.

8. REFERENCES

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

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

[3] G. de Melo and G. Weikum. Towards a Universal Wordnet by Learning from Combined Evidence. In *CIKM*, 2009.

[4] F. Giunchiglia, V. Maltese, F. Farazi, and B. Dutta. GeoWordNet: A Resource for Geo-spatial Applications. In *ESWC*, 2010.

[5] C. Gutierrez, C. A. Hurtado, and A. Vaisman. Introducing Time into RDF. *IEEE TKDE*, 19:207–218, 2007.

[6] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. Research report, Max-Planck-Institut für Informatik, 2010.

[7] M. Koubarakis and K. Kyzirakos. Modeling and Querying Metadata in the Semantic Sensor Web: The Model stRDF and the Query Language stSPARQL. In *The Semantic Web: Research and Applications*. 2010.

[8] S. P. Ponzetto and M. Strube. Deriving a Large-Scale Taxonomy from Wikipedia. In *AAAI*, 2007.

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

[10] O. Udrea, D. Reforgiato, and V. Subrahmanian. Annotated RDF. *ACM Tr. Comp. Log.*, 11(2), 2010.