YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia

Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, Gerhard Weikum

MPI–I–2010–5-007 November 2010

Authors' Addresses

Johannes Hoffart, Klaus Berberich, Gerhard Weikum
Max-Planck-Institut für Informatik
Campus E 1 4

D-66123 Saarbrücken

Fabian M. Suchanek
INRIA Saclay

Acknowledgements

We are grateful for input from various people's work: Edwin Lewis-Kelham for implementing the YAGO2 user interface, Gerard de Melo for his help on integrating his Universal WordNet, and Erdal Kuzey for his work on named events and time facts in Wikipedia. We would also like to thank the people who helped evaluate the quality of YAGO2 by manual assessment, most notably, Ndapandula Nakashole, Stephan Seufert, Erdal Kuzey, and Anh Tuan Tran.

Abstract

We present YAGO2, an extension of the YAGO knowledge base, in which entities, facts, and events are anchored in both time and space. YAGO2 is built automatically from Wikipedia, GeoNames, and WordNet. It contains 80 million facts about 9.8 million entities. Human evaluation confirmed an accuracy of 95% of the facts in YAGO2. In this paper, we present the extraction methodology, the integration of the spatio-temporal dimension, and our knowledge representation SPOTL, an extension of the original SPO-triple model to time and space.

Keywords

ontologies, knowledge bases, spatio-temporal facts, information extraction

Contents

1 Introduction

1.1 Motivation

Comprehensive knowledge bases in machine-readable representations have been an elusive goal of AI for decades. Seminal projects such as Cyc [26] and WordNet [18] manually compiled common-sense and lexical (word-sense) knowledge, yielding high-quality repositories on intensional knowledge: general concepts, semantic classes, and relationships like hyponymy (subclass-of) and meronymy (part-of). These early forms of knowledge bases contain logical statements that songwriters are musicians, that musicians are humans and that they cannot be any other species, or that Canada is part of North America and belongs to the British Commonwealth. However, they do not know that Bob Dylan and Leonard Cohen are songwriters, that Cohen is born in Montreal, that Montreal is a Canadian city, or that both Dylan and Cohen have won the Grammy Award. Early resources like the original Cyc and WordNet lacked extensional knowledge about individual entities of this world and their relationships (or had only very sparse coverage of such facts).

In the last few years, the great success of Wikipedia and algorithmic advances in information extraction have revived interest in large-scale knowledge bases and enabled new approaches that could overcome the prior limitations. Notable endeavors of this kind include DBpedia [4], KnowItAll [16, 5], Omega [34], Wikitaxonomy [36, 35], and YAGO [41, 42], and meanwhile there are also commercial services such as freebase.com, trueknowledge.com, or wolframalpha.com. These contain many millions of individual entities, their mappings into semantic classes, and relationships between entities. DBpedia has harvested facts from Wikipedia infoboxes at large scale, and also interlinks its entities to other sources in the Linked-Data cloud [7]. YAGO has paid attention to inferring class memberships from Wikipedia category names, and has integrated this information with the taxonomic backbone of WordNet. Most of these knowledge bases represent facts in the form of subject-property-object triples (SPO triples) according to the RDF data

model, and some provide convenient query interfaces based on languages like SPARQL.

However, current state-of-the-art knowledge bases are mostly blind to the temporal dimension. They may store birth dates and death dates of people, but they are unaware of the fact that this creates a time span that demarcates the person's existence and her achievements in life. They are also largely unaware of the temporal properties of events. For example, they may store that a certain person is the president of a certain country, but presidents of countries or CEOs of companies change. Even capitals of countries or spouses are not necessarily forever. Therefore, it is crucial to capture the time periods during which facts of this kind actually happened. However, this kind of temporal knowledge has not yet been treated systematically in state-of-the-art work. A similar problem of insufficient scope can be observed for the spatial dimension. Purely entity-centric representations know locations and their located-in relations, but they do not consistently attach a geographical location to events and entities. The geographical location is a crucial property not just of physical entities such as countries, mountains, or rivers, but also of organization headquarters, or events such as battles, fairs, or people's births. All of these entities have a spatial dimension.

If it were possible to consistently integrate the spatial and the temporal dimension into today's knowledge bases, this would catapult the knowledge bases to a new level of usefulness. The knowledge base would be fully time and space aware, knowing not only that a fact is true, but also when and where it was true. The most obvious application is that it would become possible to ask for distances between places, such as organization headquarters and cities (already possible today), or even between places of events (mostly not supported today). The time-awareness would allow asking temporal queries, such as "Give me all songs that Leonard Cohen wrote after 'Suzanne' ". Another, perhaps less obvious application is the ability to spatially and temporally locate practically any entity that occurs in a natural language discourse. Simple examples are sentences such as "I am going to Berlin", which could be automatically annotated with the coordinates of Berlin. We may even want to refer to locations by informal and vague phrases such as "the midwest" or "the corn belt". Likewise, the new knowledge base would be able to assign a time dimension to a sentence such as "During the era of Elizabeth I, the English waged war against the Spanish", so that this event could be temporally anchored. More subtle examples are expressions that have both a temporal and a spatial dimension. Take "Summer of Love" as example. This term conveys more than just a time (1967). It also conveys a place (San Francisco) and a duration (a few months) [17]. 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 the Summer of Love".

1.2 Contribution

What we need is a comprehensive anchoring of current ontologies along both the spatial and the temporal dimension. This paper presents such an endeavor: YAGO2. As the name suggests, this is a new edition of the YAGO knowledge base. However, in contrast to the original YAGO, the methodology for building YAGO2 (and also maintaining it) is systematically designed top-down with the goal of integrating entity-relationship-oriented facts with the spatial and temporal dimensions. To this end, we have developed an extensible approach to fact extraction from Wikipedia and other sources, and we have tapped on specific inputs that contribute to the goal of enhancing facts with spatio-temporal scope. Moreover, we have developed a new representation model, coined SPOTL tuples (SPO + *T*ime + *L*ocation), which can co-exist with SPO triples, but provide a much more convenient way of browsing and querying the YAGO2 knowledge base.

Along these lines, the paper makes the following novel contributions:

- an *extensible framework for fact extraction* that can tap on infoboxes, lists, tables, categories, and regular patterns in free text, and allows fast and easy specification of new extraction rules;
- an extension of the knowledge representation model tailored to *capture time and space*, as well as rules for *propagating time and location* information to all relevant facts;
- methods for gathering *temporal facts* from Wikipedia and for seamlessly integrating *spatial types and facts* from GeoNames (http://geonames.org), in an ontologically clean manner with high accuracy;
- a new *SPOTL representation* of spatio-temporally enhanced facts, with expressive and easy-to-use *querying*.

The result is the *YAGO2 knowledge base*, which is publicly available at `www.mpi-inf.mpg.de/yago-naga/yago2/`. It contains more than 80 million facts for 9.8 million entities (if GeoNames data is included). Without GeoNames, it still contains more than 33 million facts for 2.6 million entities, extracted from Wikipedia and WordNet. Both facts and entities are properly placed on their temporal and geographical dimension, thus making YAGO2 a truly

time- and space-aware ontology. More than 13 million facts are associated with their occurrence time, and more than 1.5 million with the location of their occurrence. Sampling-based manual assessment shows that YAGO2 has a precision (i.e., absence of false positives) of more than 95 percent (with statistical significance tests).

The rest of the paper is organized as follows. Section 2 gives a brief overview of the original YAGO knowledge base. Section 3 presents our extraction architecture. Section 4 introduces the temporal dimension in YAGO2. Section 5 introduces the spatial dimension. Section 6 explains additional context data in YAGO2. Section 7 describes our SPOTL model and querying. Section 8 presents the evaluation of YAGO2. Section 9 reviews related work, before Section 10 concludes.

2 The YAGO Knowledge Base

YAGO was originally introduced in [41]. The YAGO knowledge base is automatically constructed from Wikipedia. Each article in Wikipedia becomes an entity in the knowledge base (e.g., since Leonard Cohen has an article in Wikipedia, `LeonardCohen` becomes an entity in YAGO). Certain categories in Wikipedia can be exploited to deliver type information (e.g., the article about Leonard Cohen is in the category `Canadian poets`, so he becomes a `canadian poet`). YAGO links this type information to the taxonomy of WordNet [18] (e.g., `canadian poet` becomes a subclass of the WordNet synset `poet`). YAGO has about 100 manually defined relations, such as `wasBornOnDate`, `locatedIn` and `hasPopulation`. Categories and infoboxes can be exploited to deliver instances of these relations. For this purpose, YAGO has manually defined patterns that map categories and infobox attributes to fact templates (e.g., Leonard Cohen has the infobox attribute `born=Montreal`, which gives us the fact `wasBornIn(LeonardCohen, Montreal)`). This resulted in 2 million extracted entities and 20 million facts. On top of these extractions, the YAGO algorithms performed extensive consistency checks, eliminating facts that do not conform to type or functionality constraints. A manual evaluation confirmed an overall precision of YAGO of 95%. The key to such a high precision on such a large set of facts were the manually defined relations, which gave the facts a well-defined semantics and thus enabled YAGO to self-check its consistency.

YAGO represents facts as triples of subject (S), predicate (P), and object (O), in compatibility with the RDF data model. YAGO makes extensive use of *reification*: every fact (SPO triple) is given an identifier, and this identifier can become the subject or the object of other facts. For example, to say that a fact with id `#42` was extracted from Wikipedia, YAGO can contain the fact `wasFoundIn(#42, Wikipedia)`. This fact has itself an id. Unlike RDF, YAGO can only reify facts that are already part of the knowledge base. Thereby, YAGO avoids problems of undecidability. The consistency of a YAGO knowledge base can still be decided in polynomial time [41].

During its young life, YAGO has found many applications and is part of or contributes to numerous other knowledge base endeavors (such as DBpedia or SUMO). The present paper embarks to take YAGO to the next level of a temporally and spatially enhanced ontology.

3 Extensible Extraction Architecture

In the first version of YAGO, much of the extraction was done by hard-wired rules in the source code. As this design does not allow easy extension, we have completely re-engineered the code. The new YAGO2 architecture is based on declarative rules that are stored in text files. This reduces the hard-wired extraction code to a method that interprets the 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 simply 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 YAGO2 hierarchy of literal types (`yagoInteger` etc.). Each literal type comes with a regular expression that can be used to check whether a string is part of the lexical space of the type. The factual rules also add 3 new classes to the taxonomy: `yagoLegalActor` (which combines legal actors such as organizations and people), `yagoLegalActorGeo` (the union of `yagoLegalActor` and geopolitical entities) and `yagoGeoEntity` (which groups geographical locations such as mountains and cities). The factual rules also help with the extraction of classes. In principle, every Wikipedia category (such as "American Rock n'Roll singers") becomes a class in YAGO2 and gets linked to the corresponding WordNet class ("singer"). Not all Wikipedia categories shall be treated this way ("1935 births", e.g., is not a subclass of "births") and so the class rules define a handful of words that identify non-conceptual categories. The rules also include a list of 60 words with their primary meaning, if the primary meaning as defined in WordNet is not suitable. For example, the word "capital" has as primary meaning in WordNet the financial amount, whereas the categories use the word in the

sense of a city. Such a factual rule is represented as a simple YAGO fact:

```
"capital"    hasPreferredMeaning   wordnet_capital_108518505
```

Implication rules say that if certain facts appear in the knowledge base, then another fact shall be added. Thus, implication rules serve to deduce new knowledge from the existing knowledge. An implication rule is also expressed as a YAGO fact, i.e., as a triple. The subject of the fact states the premise of the implication, and the object of the fact holds the conclusion. Both the subject and the object are strings that contain fact templates. 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. For example, one of the implication rules states that if a relation is a sub-property of another relation, then all instances of the first relation are also instances of the second relation. Implication rules use the relation `implies`, with strings as arguments:

```
"$1 $2 $3; $2 subpropertyOf $4;"    implies   "$1 $4 $3"
```

Replacement rules say that if a part of the source text matches a specified regular expression, it should be replaced by a certain string. This takes care of interpreting micro-formats, cleaning up HTML tags, and normalizing numbers. It also takes care of eliminating administrative Wikipedia categories (such as "Articles to be cleaned up") and articles that we do not want to process (such as articles entitled "Comparison of...") – simply by replacing this material by the empty string. Replacement rules use `replace`, with strings as arguments:

```
"\{\{USA\}\}"    replace   "[[United States]]"
```

Extraction rules say 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. The regular expression (Syntax as in `java.util.regex`) contains capturing groups that single out the parts that contain the entities. The capturing groups are used in the templates that generate the facts. The templates also define the syntactic type of the entity, e.g. "Wikipedia Link" or "Class" or one of the YAGO literal types. This allows the extractor to seek and check the entities in the captured group, thus making sure that no syntactically wrong information is extracted.

```
"\[\[Category:(.+) births\]\]"   pattern   "$0 wasBornOnDate Date($1)"
```

This architecture for extraction rules is highly versatile and easily extensible. It allows accommodating new infoboxes, new exceptions, new fact types, and new preprocessing by simply modifying the text files of rules.

4 Giving YAGO a Temporal Dimension

The meta-physical characteristics of time and existence have been the subject of intense philosophical debate ever since the inception of philosophy. In fact, long before it came to mean "semantic knowledge base", the word "ontology" itself referred foremost to the philosophical study of existence. For YAGO2, we can choose a more pragmatic approach to time, because we can derive the temporal properties of objects from the data we have in the knowledge base.

YAGO2 contains a data type `yagoDate` that denotes time points, typically with a resolution of days but sometimes with cruder resolution like years. Dates are denoted in the standard format `YYYY-MM-DD` (ISO 8601). If only the year is known, we write dates in the form `YYYY-##-##` with `#` as a wildcard symbol. In YAGO2, facts can only hold at time points; time spans are represented by two relations that together form a time interval (e. g. `wasBornOnDate` and `diedOnDate`). We consider temporal information for both entities and facts:

- *Entities* are assigned a time span to denote their existence in time. For example, Elvis Presley is associated with `1935-01-08` as his birthdate and `1977-08-16` as his time of death. Bob Dylan (who is still alive), is associated only with the time of birth, `1941-05-24`. The relevant relations are discussed below in Section 4.1.

- *Facts* are assigned a time point if they are instantaneous events, or a time span if they have an extended duration with known begin and end. For example, the fact `BobDylan created BlondeOnBlonde` is associated with time point `1966-05-16` (the release date of this album). The fact `BobDylan isMarriedTo SaraLowndes` is associated with the time span from `1965-##-##` to `1977-##-##`. The time of facts is discussed in Section 4.2.

Thus, YAGO2 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, it does not attempt any assignment. Thereby, YAGO2 chooses a conservative approach, leaving some time-dependent entities without a time scope, but never assigning an ill-defined time.

4.1 Entities and Time

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 die. Countries are created and dissolved. Buildings are built and possibly destroyed. We capture this by the notion of an entity's *existence time*, the span between the creation and destruction of the entity.

Some entities come into existence, but never cease to exist. This applies to abstract creations such as pieces of music, scientific theories, or literature works. These entities have not existed prior to their inception, but they will never cease to exist. Thus, they have an unbounded end point of their existence time. Other entities have neither well-defined begin nor end, or we lack information about these points in the knowledge base. Examples are numbers, mythological figures, or virus strains (for which we do not have any information about their existence – which is different from their discovery). In these cases, YAGO2 does not assign any existence time.

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:

People where the relations `wasBornOnDate` and `diedOnDate` demarcate their existence times;

Groups such as music bands, football clubs, universities, or companies, where the relations `wasCreatedOnDate` and `wasDestroyedOnDate` demarcate their existence times;

Artifacts such as buildings, paintings, books, music songs or albums, where the relations `wasCreatedOnDate` and `wasDestroyedOnDate` (e.g., for buildings or sculptures) demarcate their existence times;

Events such as wars, sports competitions like Olympics or world championship tournaments, or named epochs like the "German autumn", where the relations `startedOnDate` and `endedOnDate` demarcate their

existence times. This includes events that last only one day (e.g., the fall of the Berlin wall). Here, the start date and the end date of the event coincide. We use the relation `happenedOnDate` for these cases.

We believe that these four types cover almost all of the cases where entities have a meaningful existence time. Note that the entities are already captured in richly populated types within YAGO2, covering 76% of all entities.

Rather than dealing with each of the above four types in a separate manner, we unify these cases by introducing two generic *entity-time relations*: `startsExistingOnDate` and `endsExistingOnDate`. Both are an instance of the general `yagoRelation`, and hold between an entity and an instance of `yagoDate`. They define the temporal start point and end point of an entity, respectively. We then specify that certain relations are sub-properties of the generic ones: `wasBornOnDate subpropertyOf startsExistingOnDate`, `diedOnDate subpropertyOf endsExistingOnDate`, `wasCreatedOnDate subpropertyOf startsExistingOnDate`, and so on. For events that last only day, we specify that `happenedOnDate` is a sub-property of both `startsExistingOnDate` and `endsExistingOnDate`. Declaring relations `subpropertyOf` other relations serves on the one hand as grouping, on the other hand we use the YAGO2 implication rule infrastructure to automatically deduce a second fact for the parent relation. For example, for the fact `BobDylan wasBornOnDate 1941-05-24`, an implication rule creates the second fact `BobDylan startsExistingOnDate 1941-05-24`.

The YAGO2 extractors can obtain a lot of temporal information about entities from Wikipedia infoboxes. Our extractors also find temporal information in the categories. For example, the article about the 82nd Academy Awards Ceremony is in the category "2009 Film Awards", which gives us the temporal dimension for the award: the year 2009.

Our infrastructure generates existence times for all entities where YAGO can deduce such information from its data.

4.2 Facts and Time

4.2.1 Facts with an Extracted Time

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 an epoch from the time Obama was elected until either another president is elected or Obama resigns. When we can extract time information for these kinds of facts from Wikipedia, we associate it as *occurrence time*: the time span

when the fact occurred. To capture this knowledge, we introduce two new relations, `occursSince` and `occursUntil`, each with a (reified) fact and an instance of `yagoDate` as arguments. For example, if the above fact had the fact id `#1`, we would indicate its time by `#1 occursSince 2009-01-20`.

For facts that last only one day (or one year if this is the relevant granularity, e.g., for awards), we use a short-hand notation by the automatically deduced relation `occursOnDate`. For example, for `BarackObama wasInauguratedAs PresidentOfTheUnitedStates` with fact id `#2`, we write `#2 occursOnDate 2009-01-20`, as short-hand for two separate facts `#2 occursSince 2009-01-20` and `#2 occursUntil 2009-01-20`.

If the same fact occurs more than once, then YAGO2 will contain it multiple times with different ids. For example, since Bob Dylan has won two Grammy awards, we would have `#1: BobDylan hasWonPrize GrammyAward` with `#1 occursOnDate 1973`, and a second `#2: BobDylan hasWonPrize GrammyAward` (with a different id) and the associated fact `#2 occursOnDate 1979`.

The YAGO2 extractors can find occurrence times of facts from the Wikipedia infoboxes. For example, awards are often mentioned with the year they were awarded. Spouses are often mentioned with the date of marriage and divorce. Our extractors can detect these annotations and attach the corresponding `occursSince` and `occursUntil` facts directly to the target fact.

4.2.2 Facts with a Deduced Time

In some cases, the entities that appear in a fact may indicate the occurrence time of the fact. For example, for `BobDylan wasBornIn Duluth`, it seems most natural to use Dylan's birth date as the fact's occurrence time. For `ElvisPresley diedIn Memphis` we would want the death date of the subject as the occurrence time, and for `BobDylan created BlondeOnBlonde`, it should be the creation time of the object.

The principle for handling these situations is to use rules that propagate the begin or end of *an entity's existence time* to the *occurrence time of a fact*, where the entity occurs as a subject or object. To avoid a large number of rules for many specific situations, we categorize relations into several major cases. Each of these has an ontological interpretation, and each can be handled by a straightforward propagation rule. More precisely, we consider a fact of the form `$id: $s $p $o` where `$id`, `$s`, `$p`, `$o` are placeholders for identifier, subject, property, and object of the fact, respectively. We want to deduce an ontologically meaningful occurrence time for this fact, i.e., facts with the relations `occursSince` or `occursUntil`, based on the ontological

nature of the relation $p.

Permanent relations Entities can have permanent properties that are non-modifiable; we call these permanent relations. Examples are hasISBN, isCalled, or hasNativeName for events. The occurrence time of facts for such relations coincides with the existence time of the subject entity. We group all these relations into a new relation class permanentRelation, by stating that hasISBN type permanentRelation, isCalled type permanentRelation, and so on. permanentRelation is in turn a subclass of yagoRelation. Note that here we use type as means of categorizing relations meaningfully, and not subpropertyOf, which would automatically deduce new facts. The propagation of the existence time of $s to the time of the entire fact is specified as an implication rule (see Section 3), written here in logical deduction notation, with the premises above the bar and conclusion below:

$$\frac{\begin{array}{c}\texttt{\$id: \$s \$p \$o};\\ \texttt{\$p type permanentRelation};\\ \texttt{\$s startsExistingOnDate \$b};\\ \texttt{\$s endsExistingOnDate \$e}\end{array}}{\begin{array}{c}\texttt{\$id occursSince \$b};\\ \texttt{\$id occursUntil \$e}\end{array}}$$

Creation relations Some facts indicate the creation of an entity. For example, a wasBornIn fact indicates the birth of a person. A fact with such a relation has as its occurrence time the beginning of the existence time of the created entity. For example, the fact ElvisPresley wasBornIn Tupelo has as its occurrence time the birth date of Elvis Presley. Therefore, we introduce a class subjectStartRelation, which groups all relations that indicate the creation of a new entity in their subject position. Some relations indicate the creation of an entity in their object position. For example, the relation created indicates the creation of an artifact, which appears in the object position of the relation. Consider, e.g., the fact LeonardCohen created Suzanne(song), which indicates the creation of the song Suzanne(song). We make these relations instances of the class objectStartRelation.

Now, it suffices to transfer the starting point of the existence time of the new entity to the occurrence time of the creation fact. This can be

done by an implication rule:

$$\frac{\texttt{\$id: \$s \$p \$o;}\quad \texttt{\$p type objectStartRelation;}\quad \texttt{\$o startsExistingOn \$b}}{\texttt{\$id occursSince \$b;}\quad \texttt{\$id occursUntil \$b}}$$

Consider again the example fact `#1: LeonardCohen created Suzanne(song)`. Knowing that the song Suzanne came into existence in 1967, we would deduce two new facts: `#1 occursSince 1967-##-##` and `#1 occursUntil 1967-##-##`. An analogous rule transfers the start point of the existence of the subject to the fact, if the relation is an instance of `subjectStartRelation`.

Destruction relations Other facts indicate the destruction of an entity. These are, e.g., `diedIn` or `destroyed`. Analogously to the creation relations, we define two new classes of relations, `subjectEndRelation` and `objectEndRelation`. The first class contains all relations that indicate that the subject of the fact ceases to exist (such as `diedIn`). The second class contains all relations that indicate that the object of the fact ceases to exist (such as `destroyed` in `Taliban destroyed BuddhasOfBamyan`). The time point of the destruction coincides with the end of the existence time of the destroyed entity.

This can be expressed by a simple implication rule:

$$\frac{\texttt{\$id: \$s \$p \$o;}\quad \texttt{\$p type subjectEndRelation;}\quad \texttt{\$s endsExistingOn \$e}}{\texttt{\$id occursSince \$e;}\quad \texttt{\$id occursUntil \$e}}$$

An analogous rule transfers the end point of the existence of the object to the fact, if the relation is an instance of `objectEndRelation`.

Unless a relation is explicitly of one of these types, we do not use any propagation of this kind. For example, we do not attempt to propagate entity existence times into fact occurrence times for relations such as `subclassOf` or `hasDomain`. Even relations such as `hasWonPrize` or `isCapitalOf` will not receive an occurrence time, unless it is explicitly specified in the Wikipedia infoboxes. This is a conservative approach, but avoids non-sensical deduction of occurrence times.

4.3 Extraction Time of Facts

In addition to the occurrence times, each fact also has a time point of its extraction and insertion into the knowledge base. For example, assume that the fact `LeonardCohen created Suzanne` has identifier `#42`. This fact `#42` was found in Wikipedia and, therefore, we have a (meta-)fact `#43: #42 wasFoundIn Wikipedia`. The fact `#43` happened on October 15, 2010, when we ran the extractor, and therefore, we have a fact `#43 extractedOn 2010-10-15`. Each fact is adorned with this meta-information. This information is independent of the semantic aspects of the fact, and rather captures provenance. Still, such meta-facts are useful, as they allow reasoners to include or exclude facts from certain sources or from certain points of time.

5 Giving YAGO a Spatial Dimension

All physical objects have a location in space. For YAGO2, we are concerned with entities that have a permanent spatial extent on Earth – for example countries, cities, mountains, and rivers. In the original YAGO type hierarchy (and in WordNet), such entities have no common super-class. Therefore, we introduce a new class `yagoGeoEntity`, which groups together all *geo-entities*, i. e. all entities with a permanent physical location on Earth. The subclasses of `yagoGeoEntity` are (given by preferred name and WordNet 3.0 synset id): location (27167), body of water (9225146), geological formation (9287968), real property (13246475), facility (3315023), excavation (3302121), structure (4341686), track (4463983), way (4564698), and land (9335240). 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`. An instance of `yagoGeoCoordinates` is a pair of a latitude and a longitude value. Each instance of `yagoGeoEntity` is directly connected to its geographical coordinates by the `hasGeoCoordinates` relation.

YAGO2 only knows about coordinates, not polygons, so even locations that have a physical extent are represented by a single geo-coordinate pair. As we extract these coordinates from Wikipedia, the assignment of coordinates to larger geo-entities follows the rules given there: for a settlement like a city, it represents the center, for military and industrial establishments the main gate, and for administrative districts it represents the head office[1].

[1]Guidelines from `http://en.wikipedia.org/wiki/Wikipedia:WikiProject_Geographical_coordinates`, last accessed 2010-10-27

5.1 Harvesting Geo-Entities

YAGO2 harvests geo-entities from two sources. The first source is Wikipedia. Wikipedia contains a large number of cities, regions, mountains, rivers, lakes, etc. Many of them also come with associated geographical coordinates. We harvest these with our extraction framework and retrieve coordinates for 176,474 geo-entities.

However, not all geo-entities in Wikipedia are annotated with geographical coordinates. Furthermore, there are many more geo-entities than are known to Wikipedia. Therefore, we tap into an even richer source of freely available geographical data: GeoNames (`http://www.geonames.org`), which contains data on more than 7 million locations. GeoNames classifies locations in a flat category structure, and each location is assigned only one class, e. g. `Berlin` is a "capital of a political entity". Furthermore, GeoNames contains information on location hierarchies (partOf), e. g. `Berlin` is located in `Germany` is located in `Europe`. GeoNames also provides alternate names for each location. All this data is a valuable addition to YAGO, so we make an effort to integrate it as completely as possible. This means that we need to match the individual geo-entities that exist both in Wikipedia and GeoNames, so that we do not duplicate theses entities when extracting them from the respective repositories.

5.1.1 Matching Locations

When processing Wikipedia articles, we try to match individual geo-entities, proceeding as follows:

1. If the Wikipedia entity has the type `yagoGeoEntity` and shares its name with exactly one entity in GeoNames, we match them.

2. If the Wikipedia entity has the type `yagoGeoEntity` and shares its name with more than one entity in GeoNames, and we have coordinates for the Wikipedia entity, we match it to the geographically closest GeoNames entity – if its distance does not exceed 5km.

3. In the end, we add all the unmatched GeoNames entities as new individual entities to YAGO, together with all the facts about them given in GeoNames.

Taking `Berlin` in Germany as an example, we find multiple geo-entities in GeoNames that have the name "Berlin". From Berlin's Wikipedia article we extract the coordinates $52°30'2''N$, $13°23'56''E$, which is less than

3km distance to the coordinates we find for one of the Berlin locations in GeoNames ($52°31'27''N$, $13°24'37''E$). We unify the two entities and add all further data extracted from GeoNames — like alternate names and where Berlin is located — to the existing YAGO2 entity `Berlin`. Following this approach for all Wikipedia articles, we unify 84,349 geo-entities. The rest of the GeoNames locations are imported as they are.

5.1.2 Matching Classes

Matching individual locations is not enough to fully integrate GeoNames into YAGO2, as in YAGO2 each individual needs to be typed. Fortunately, GeoNames assigns a class to each location, which we can use as type. Again, to avoid duplication of classes, we have to match them to existing classes. There is prior work that aligns all GeoNames classes with WordNet classes (the backbone of the YAGO2 class hierarchy), most notably, GeoWordNet [20]. However, GeoWordNet relies on manual curation to accomplish correct matchings. This approach is both time-intensive and fragile when either GeoNames or WordNet changes, something that will definitely happen in future releases of either resource.

To counter this problem, we devised an automated matching algorithm. This algorithm uses solely data that is readily available, namely the YAGO2 class hierarchy, as well as textual descriptions for both YAGO2 classes and GeoNames categories.

The automated matching works as follows.

1. For every class from GeoNames, we identify a set of WordNet classes from YAGO that have the same name as the GeoNames class (including synonymous alternative names).

2. If there are no such classes, we do a shallow noun phrase parsing of the GeoNames class name in order to determine the head noun (this is, e.g., "mine" for "gold mine"). We search for classes in YAGO2 that carry the head noun as their name.

3. From the resulting YAGO2 classes, we remove the ones that are not subclasses of `yagoGeoEntity`, as we know that GeoNames contains only geographical classes.

4. If only a single class remains, we return this one as the matching class.

5. If more than one class remains, we use the glosses describing the GeoNames class and the YAGO classes, respectively. The glosses are tok-

enized, and the Jaccard Similarity of the resulting bag-of-words is calculated between the GeoNames-class gloss and each candidate's gloss. The class with the highest overlap is returned as best match.

6. If there is no overlap between the glosses at all, we return the YAGO2 class that is most often denoted by the name of the GeoNames class - this information is taken from WordNet, which sorts senses for each word in order of most common use.

Algorithm 1 shows pseudo-code for this method.

Matched classes are added to YAGO2 as subclass of the matched class, unmatched classes are added as subclass of `yagoGeoEntity`, so we do not lose them.

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. Furthermore, GeoNames augments the `isLocatedIn` hierarchy in YAGO2. Last, it also yields alternative names for geographic entities. We use this information for entities that do not exist in Wikipedia, but also augment entities extracted from Wikipedia with alternate or foreign language names. For example, the information that the "Peru-Chile Trench" is also called "Arica Trench" is not present in Wikipedia.

5.2 Assigning a Location

We deal with the spatial dimension in a manner similar to the way we deal with time, as described in Section 4: we assign a location to both entities and facts, wherever this is ontologically reasonable and wherever this can be deduced from the data. The location of facts and entities is given by a geo-entity. For example, the location of the *Summer of Love* is San Francisco, which is an instance of `yagoGeoEntity`.

5.2.1 Entities and Location

Many entities are associated with a location. For example, events take place at a specific place, organizations have their headquarters in a specific city, and works of art are displayed in a museum. We have such spatial data in our knowledge base for the following types of entities:

Events that took place at a specific location, such as battles or sports competitions, where the relation `happenedIn` holds the place where it happened.

Groups or organizations that have a venue, such as the headquarters of a company or the campus of a university. The location for such entities is given by the `isLocatedIn` relation.

Artifacts that are physically located somewhere, like the Mona Lisa in the Louvre, where the location is again given by `isLocatedIn`.

The semantics of such relations varies, but instead of treating each case separately, we define a new relation to treat all entities in a uniform way: `placedIn`. Both `isLocatedIn` and `happenedIn` are defined as sub-properties of this new relation, and the YAGO2 infrastructure generates the `placedIn` facts for each entity type where it can be deduced from the knowledge base.

5.2.2 Facts and Location

Some facts also have a spatial dimension. 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: for example, schema-level facts such as `subclassOf` or identifier relations such as `hasISBN` have no location on Earth. We introduce the relation `occursIn`, which holds between a (reified) fact and a geo-entity. For example, if we have the fact `#1: LeonardCohen wasBornOnDate 1934`, we would write its location as `#1 occursIn Montreal`. Again, the key to a semantically clean treatment of the spatial dimension of facts lies in the relations. We distinguish three cases where we can deduce an ontologically meaningful location.

Permanent Relations. As defined in Section 4.2, permanent relations describe properties of entities that are immutable. If the described entity has a permanent location, so has the fact that describes it. We use the following two implication rules, where the first transfers the location of the entity to the fact, and the second transfers the entity itself if it is

a geo-entity:

```
$id: $s $p $o;
$p type permanentRelation;
$s placedIn $l
-----------------------------
$id occursIn $l
```

```
$id: $s $p $o;
$p type permanentRelation;
$s type yagoGeoEntity
-----------------------------
$id occursIn $s
```

Take for example the `2006FIFAWorldCup`. Assume that we extracted from the Wikipedia infobox that `2006FIFAWorldCup happenedIn Germany`. We want to propagate this location to all associated facts with a `permanentRelation`. For example, for `id: 2006FIFAWorldCup isCalled FootballWorldCup2006`, we associate the meta-fact `id occursIn Germany`.

Space-Bound Relations. Some facts occur in a place that is indicated by their subject or object. For example, the fact that Bob Dylan was born in Duluth happened in Duluth. 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, e. g. `hasMayor`. Then, we can transfer the location of the fact argument to the fact itself by the following two rules:

```
$id: $s $p $o;
$p type relationLocatedByObject;
$o placedIn $l
-----------------------------
$id occursIn $l
```

```
$id: $s $p $o;
$p type relationLocatedByObject;
$o type yagoGeoEntity
-----------------------------
$id occursIn $o
```

(correspondingly for `relationLocatedBySubject`)

The first rule fires for facts that directly concern geo-entities. For example, it would infer the (trivial but correct) meta-fact `#1 occursIn Duluth` for the fact `#1: BobDylan wasBornIn Duluth`. The second rule fires for entities that are not geo-entities but do have a physical location. For example, the second rule will infer that the location of the fact `FrenchEmpire participatedIn BattleOfWaterloo` is Waterloo, assuming that we know that `BattleOfWaterloo` is located in Waterloo. Note that these rules will only fire if the subject or object indeed has a known location.

Tandem Relations. Some relations occur in tandem: One relation determines the location of the other. For example, the relation `wasBornOnDate` defines the time of the corresponding `wasBornIn` fact, and the latter defines the location of the former. We express this tandem 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. Examples for such pairs are `wasBornOnDate`/`wasBornIn`, `diedOnDate`/`diedIn` and `happenedOnDate`/`happenedIn`. The following rule can transfer the location from one relation to the other

$$\frac{\begin{array}{c}\texttt{\$id1: \$s \$p \$t;}\\ \texttt{\$p timeToLocation \$r;}\\ \texttt{\$id2: \$s \$r \$l;}\\ \texttt{\$id2 occursIn \$l;}\end{array}}{\texttt{\$id1 occursIn \$l}}$$

For example, given the facts `#1: BobDylan wasBornOnDate 1941-05-24` and `#2: BobDylan wasBornIn Duluth`, the space-bound relation `wasBornIn` will first deduce `#2 occursIn Duluth`. The tandem pair `wasBornOnDate`/`wasBornIn` will then deduce `#1 occursIn Duluth`.

These rules derive a location for a fact whenever this is semantically meaningful.

Algorithm 1: Matching GeoNames to YAGO2 class

Input:
geo_class: GeoNames class with gloss
YAGO: set of YAGO classes, each class with synonyms syn, preferred_meaning, and gloss
YagoGeo: set of YAGO classes with geographical meaning (manually defined)
Output:
yago_class ∈ YAGO (best match for geo_class)

```
1  begin
2      Cand ← {y ∈ YAGO | y or syn(y) = geo_class}
3      if Cand = ∅ then
4          Cand ← {y ∈ YAGO | y or syn(y) = head(geo_class)}
5          if Cand = ∅ then
6              return no match
7      GeoCand ← Cand ∩ YagoGeo
8      if |GeoCand = 1| then
9          return g ∈ GeoCand
10     else if |GeoCand| > 1 then
11         Cand ← GeoCand
12         /* Cand contains original set or only classes with geo meaning
               */
13     best ← argmax_{c∈Cand}(jacc_sim(gloss(g), gloss(c)))
14     if jacc_sim(g,best) > 0.0 then
15         return best
16     else
17         return preferred_meaning(geo_class)
18         /* preferred_meaning is the meaning (class) the string is most
               often used for */
```

6 (Con-)Textual Data in YAGO2

YAGO2 does not just contain a time and a location for facts and entities, but also meta information about the entities. This includes non-ontological data from Wikipedia as well as multilingual data.

6.1 Non-Ontological Data from Wikipedia

For each entity, YAGO2 contains *contextual information*. This context is gathered by our extractors from Wikipedia. They include the following relations, with an entity and a string as arguments:

hasWikipediaAnchorText links an entity to a string that occurs as anchor text in the entity's article.

hasWikipediaCategory links an entity to the name of a category in which Wikipedia places the article. These include not just the conceptual categories that form the YAGO taxonomy, but also all other categories.

hasCitationTitle links an entity to a title of a reference on the Wikipedia page. Wikipedia often references external works for reasons of verifiability. The titles of these cited references form another source of contextual information.

All of these relations are sub-properties of the relation `hasContext`. This relation provides a wealth of keywords associated to the entity, we extract more than 76 million context facts for the YAGO2 entities in total. We will see in Section 7 how the context can be used as an additional means for searching knowledge in YAGO2.

6.2 Multilingual Information

For individual entities, we extract multilingual translations from inter-language links in Wikipedia articles. This allows us to refer to and query for YAGO2 individuals in foreign languages. YAGO2 represents these non-English entity names through reified facts. For example, we have the reified fact `#1: BattleAtWaterloo isCalled SchlachtBeiWaterloo` with the associated fact `#1 inLanguage German`.

This technique works for the individuals in YAGO2, but not for the classes, because the taxonomy of YAGO2 is taken from WordNet, which is in English. To fill this gap, we integrate the Universal WordNet (UWN) [12] into YAGO2. UWN maps words and word senses of WordNet to their proper translations and counterparts in other languages. For example, the French word "école" is mapped to its English translation "school" at the word level, but only to specific meanings of school at the word-sense level, as the French word does never denote, e.g., a school of fish or a school of thought. UWN contains about 1.5 million translations and sense assignments for 800,000 words in over 200 languages at a precision of over 90% [12]. Overall, this gives us multilingual names for most entities and classes in YAGO2.

7 SPOTL(X) Representation

7.1 Drawbacks of Reification-based Models

In YAGO2, as in YAGO [41], we represent the time and location of facts through reification. Each *base-fact* has an *identifier*, which in turn can be used in the S or O role in another fact, a *meta-fact*. For example, suppose we know the base-fact `#1: GratefulDead performed TheClosingOfWinterland` about the rock band Grateful Dead. Adding knowledge about the place and time of this concert is expressed by two meta-facts `#2: #1 occursIn SanFrancisco` and `#3: #1 occursOnDate 1978-12-31`.

The YAGO query language allows writing SPARQL-like queries that include fact identifiers. However, already a simple query for a location requires a large number of joins. For example, if we want to find concerts that took place near San Francisco, we need a rather convoluted query, consisting of five triple patterns (separated by dots, the syntax of the SPARQL Where clause):

```
?id: ?s performed ?o .
?id occursIn ?l .
?l hasGeoCoordinates ?g .
SanFrancisco hasGeoCoordinates ?sf .
?g near ?sf .
```

Here, `near` is a proximity predicate (with a predefined distance of say 50 km) and `?id` is a fact-identifier variable; we specify a join between the identifier variable and the S component of another (meta-fact) triple. In the following, we refer to such identifier-based joins as *de-reification joins*. To make this notion more precise, consider a set of RDF triples with identifiers that can be used in other facts using reification. These triples can be viewed as quadruples of the form (id, s, p, o). A *de-reification join* is then a conjunctive query (in the relational Datalog sense) with the same variable `?x` appearing in the *id* role of one sub-query and either the s or the o role of

another sub-query. If we cast all reified triples into a (virtual) relational table with schema $R(Id, S, P, O)$, then a de-reification join can be algebraically written as an equi-join of the form $R \bowtie_{[Id=S]} R$ or $R \bowtie_{[Id=O]} R$. The semantics of de-reification joins are thus well-defined in terms of query results for relational calculus (Datalog) or relational algebra.

For a non-expert, it is not easy to come up with these five joins and the proper use of location names, coordinates, etc. Conceptually, the query seems to require only a single spatial join between concerts and places, but the tedious SPARQL formulation has four joins between five triple patterns. In addition, the lack of genuine support for data types for space and time makes it difficult to express proximity conditions or temporal comparisons. Note that we already helped ourselves by liberally introducing the `near` predicate, which is not really available in our knowledge base and not supported by SPARQL.

7.2 SPOTL(X)-View Model

The key idea for making browsing and querying more convenient is to provide users and programmers with a de-reification-join view. Instead of seeing only SPO triples and thus having to perform an explicit de-reification join for associated meta-facts, the user should see extended 5-tuples where each fact already includes its associated temporal and spatial information. We refer to this view of the data as the SPOTL view: SPO triples augmented by ***T****ime* and ***L****ocation*. We also discuss a further optional extension into SPOTLX 6-tuples where the last component offers keywords or key phrases from the *conte****X****t* of sources where the original SPO fact occurs. The context component caters to those cases where users have a good intuition about their information need, but have problems casting it into triple patterns (e.g., because they lack proficiency with the knowledge base and its relations), or, are faced with too large a query result that they need to narrow down. In such situations, being able to query both fact triples and associated text in a combined manner often proves to be very useful [15]. For example, we may desire augmenting a triple pattern like `?s performed ?o` with a keyword condition like `"psychedelic rock jam session"` which cannot be cast into a crisp ontological fact.

The situation that our knowledge base now contains well-defined temporal and spatial information for base-facts, as described in Sections 4 and 5, simplifies the construction of the SPOTL(X) view. In detail, it is composed of the following – virtual – relations:

$R(Id, S, P, O)$ – all (id, s, p, o)-tuples in the knowledge base.

$T(Id, TB, TE)$ – all (id, t_b, t_e)-tuples that associate the time interval $[t_b, t_e]$ with the fact identified by id. The t_b-component is set using the `occursSince` relation; the t_e-component is set using the `occursUntil` relation. Our definitions in Section 4 guarantee that this can be done unambiguously and consistently. The t_b- or t_e-component might not be set, if there is no corresponding meta-fact in our knowledge base. In that case, the respective component assumes a NULL value whose appropriate interpretation is deferred until query-processing time.

$L(Id, LAT, LON)$ – all (id, lat, lon)-tuples that associate the location $< lat, lon >$ (i.e., a pair of latitude and longitude) with the fact identified by id. The l-component is set using the `occursIn` relation to retrieve the location and `hasGeoCoordinates` to retrieve its coordinates.

$X(Id, C)$ – all (id, c)-tuples that associate a context c with the fact identified by id. The c-component is based on the `hasContext` relation, applied to both the subject and the object of the fact. The `hasContext` relation was introduced in Section 6.1. The range of the c-component is a set of words or phrases by forming the union of the strings from the various relations that underlie `hasContext` (or alternatively, a bag of words or phrases if we want to consider frequencies of repeated strings).

Based on these building blocks we define the *SPOTL(X) view* as

$$\pi_{[R.Id,\,[TB,TE],\,<LAT,LON>,\,C]}(((R \,⟕_{[Id=Id]}\, T) \,⟕_{[Id=Id]}\, L) \,⟕_{[Id=Id]}\, X)\ ,$$

joining facts from R with their associated information from T, L, and C. Here, ⟕ denotes an outer join, to avoid losing triples that do not have spatio-temporal or contextual facts and instead producing NULL values in the respective fields. Figure 7.1 shows a SPOTL(X) view as it could be determined for our introductory example. Note that, in the figure, we employ the short-hand notation `[1978-12-31]` to denote the time interval `[1978-12-31, 1978-12-31]` and present content excerpts that are not mentioned in our introductory example.

7.3 SPOTL(X) Querying

The SPOTL(X) view defined above associates facts with canonical time and space information and, as we describe now, avoids most de-reification joins. Beyond that, time and space are special dimensions with inherent semantics that remain hidden to standard triple-pattern queries. Finding all actors who

Id	S	P	O	T	L	X
id1	GD	performed	TCOW	1978-12-31	-37.5, 122.3	*"Wall of Sound..."*
id2	id1	occursIn	SF			*"Golden Gate..."*
id3	id1	occursOnDate	1978-12-31			

Figure 7.1: SPOTL(X)-View Example: Grateful Dead performing "The Closing of Winterland" in San Francisco on New Year's Eve of 1978

were born *near* Berlin *after* the German reunification, for instance, is hard to express. The lack of genuine support for data types time and space forces users to "paraphrase" the query (e.g., by asking for birth places located in the same federal state as Berlin). Second, *Berlin* and *German reunification*, in our example, refer to a specific location (i.e., `<48.52, 2.20>`) and time (i.e., `[1990-10-03]`), respectively. When using standard triple-pattern queries, though, getting to this referred time and space would again require (de-reification) joins and a deep comprehension of the knowledge base and its relations. Our SPOTL(X) query interface, which we describe now, addresses these issues and is designed to operate directly on the SPOTL(X) view.

Dimension	Predicate	Valid Examples
Time	`overlaps`	`[1967, 1994] [1979, 2010]`
	`during`	`[1967, 1994] [1915, 2009]`
	`before`	`[1967, 1994] [2000, 2008]`
	`after`	`[1967, 1994] [1939, 1945]`
Space	`westOf`	`<48.52, 2.20> <52.31, 13.24>`
	`northOf`	`<48.52, 2.20> <41.54, 12.29>`
	`eastOf`	`<48.52, 2.20> <51.30,  0.70>`
	`southOf`	`<48.52, 2.20> <59.20, 18.30>`
	`nearby`	`<48.52, 2.20> <48.48,  2.80> 25.00`
conteXt	`matches`	`''...cowboys in Mexico...'' (+cowboys)`
		`''...her debut album...'' (+debut -live)`

Table 7.1: Predicates supported for Querying the SPOTL(X)-View

To deal with the important dimensions of time, space, and context and to make their inherent semantics accessible to users, we introduce the predicates given in Table 7.1. Our time predicates are a subset of those identified by Allen [2]. We include spatial predicates that reflect the relative position of two locations, as well as `nearby` which tests whether the geographic distance between the two locations is below a given threshold (e.g., `25.0` km).

The `matches` predicate for the context dimension tests whether the context matches a given keyword query that consists of mandatory and forbidden terms (e.g., `+debut -live`).

Queries can add one predicate from each dimension to every triple pattern. Patterns may thus be of arity up to six. Consider, as an example, the query

```
?p directed ?m after [1970] matches (+cowboys +mexico) .
```

that finds directors of movies made after 1970 having something to do with cowboys in Mexico (as captured by the context condition).

Often, the time or location of interest (e.g., `[1970]` above) would not be known explicitly, but be associated with an entity. When using standard triple-pattern queries, this is a frequent cause of (de-reification) joins, as explained above. In our SPOTL(X) query interface, time and space can be specified implicitly through an associated entity – a major improvement in query convenience. For example, the query

```
GeorgeHarrison created ?s after JohnLennon .
```

identifies songs written by George Harrison after John Lennon's death. When processing the query, the entity `JohnLennon` is transparently replaced by its associated time interval `[1940-10-09, 1980-12-08]` that is determined as described in Section 4. Here, we compare time intervals with the semantics that $[b_1, e_1]$ precedes $[b_2, e_2]$ if $e_1 < b_2$. This condition is satisfied for the creation times (intervals that span only one day, or month or year if this is the best known resolution) following the existence time of John Lennon. To see how this improves querying convenience, consider the following, much more tedious, triple-pattern formulation for the same information need:

```
GeorgeHarrison created ?s .
?s wasCreatedOn ?t1 .
JohnLennon diedOn ?t2 .
?t1 after ?t2 .
```

The possibility to specify time and space implicitly through an entity name, in combination with our context dimension, allows for intuitive and powerful queries, such as

```
?p isA Guitarist matches (+left +handed) .
?p wasBornIn ?c nearby Seattle 25.0 .
```

that identifies left handed guitarists who were born in the vicinity of (i.e., at most 25 km away from) Seattle. Good results should include Jimi Hendrix.

Our query interface, as an additional feature, supports natural language phrases to reference entities, which is particularly useful if the specific entity name is unknown to the user. Thus, the query

```
"Bobby Dylan" created ?s before "Knocking on Heaven's Door" .
```

identifies all songs that Bob Dylan wrote before Knocking on Heaven's Door. To this end, we leverage the `means` relation to map the phrases "Bobby Dylan" and "Knocking on Heaven's Door" to the entities named `BobDylan` and `Knockin'OnHeaven'sDoor`, thus also retrieving the time span `[1973-07-13, ####-##-##]` associated with the song. Note the subtle differences between input phrases and official entity names. Here we exploit the `means` relation that provides a rich repertoire of alternate names including multilingual ones. The pseudo-constant `####-##-##` indicates that the end boundary of the time interval is unknown. This has no effect when evaluating the query at hand, given that the `before` predicate only considers the begin boundary of the time interval.

Putting all features together, our initial information need related to actors can be satisfied by issuing the query

```
?p isA actor .
?p wasBornIn ?l nearby Berlin 10.0 .
?p wasBornOnDate ?d after "German reunification" .
```

Our concrete implementation of the SPOTL(X) query interface builds on PostgreSQL as a relational database system. The SPOTL(X) view is materialized into a single table of 7-tuples (SPOTLX plus ids). To achieve good response times, we adopt ideas put forward in recent work on the efficient triple store RDF-3X [28]. We build auxiliary B^+-Tree indexes for all six permutations of the `SPO` columns. For the additional columns, corresponding to the time, space, and context dimension, we build additional indexes specifically suited to the respective data type. In detail, for the space dimension we use the freely available PostGIS extension (http://postgis.refractions.net) to build a spatial index (based on GiST [23]). We build two additional B^+-Tree indexes to deal with the time dimension. Finally, to support efficient evaluation of our `matches` predicate on the `X` column, we employ PostgreSQL's built-in text-indexing functionality.

8 Evaluation

Our main goal for the construction of the YAGO2 ontology was near-human accuracy. Therefore, this section presents an exhaustive evaluation of our knowledge base. In the ideal case, we would compare the data in YAGO2 to some prior ground truth. Such ground truth, however, is only available for a small subset of YAGO2, namely the GeoWordNet matching of GeoNames classes onto YAGO2 classes. We will describe this evaluation in Section 8.2. For the rest of the facts in YAGO2, there is no pre-existing ground truth, so we had to rely on human judgement for sampled facts.

8.1 Facts from Wikipedia

We devised an extensive evaluation of the facts extracted from Wikipedia. Our evaluation concerns only the base facts of YAGO2, not the facts derived by implication rules. It also concerns exclusively the "semantic" relations (such as `wasBornOnDate`) and not the "technical" relations (such as `hasWikipediaURL`). In our methodology [41], human judges are presented with randomly selected facts, for which they have to assess the correctness. Since the judges might not have enough knowledge to assess each fact, the Wikipedia page from which the fact was extracted was presented next to the fact. Thus, the judges evaluated the correctness of YAGO2 with respect to the content of Wikipedia. We did not assess the factual correctness of Wikipedia itself. We used the Wikipedia dump from 2010-08-17 for the YAGO2 extraction and evaluation.

To get a detailed picture of the accuracy of YAGO2, we formed pools of facts. We formed one pool for each relation, i.e., one pool with all `wasBornIn` facts, one pool with all `wasBornOnDate` facts, etc. For each pool, we drew random samples of facts. Then, we had the judges evaluate the correctness of the facts in the sample. This allowed us to estimate the overall correctness of the facts in the pool. One pool may contain facts extracted by different

extraction patterns. Since samples were randomly drawn, we expect the distribution of extraction patterns in the sample to represent the distribution of patterns in the pool.

Two extraction methods deserve special attention:

Concept Linker: The heuristics that matches Wikipedia categories against WordNet synsets. The categories that could be matched to WordNet serve as `type` facts for entities.

Infobox Typer: The heuristics that matches the types of Wikipedia infobox templates against WordNet synsets, thus contributing `type` facts for the entities.

These techniques build the link from the Wikipedia data to the WordNet taxonomy. Since this link is crucial for domain and range checking and for the taxonomic coherence overall, we evaluated these two methods in two separate pools.

26 judges participated in our evaluation. Over the course of a week, they evaluated an overall number of 7465 facts. This gave us a precision value for each sample. We generalize the precision on the sample to the precision of the pool by help of the Wilson confidence interval [9], and get a center of about 95% at an interval width of less than ±5%. This ensures that our findings are statistically significant. Table 8.1 shows the results for the techniques described above, Table 8.2 the results for some of the important non-temporal, non-spatial relations. Table 8.3 shows the three relations with best and worst accuracy. Table 8.4 finally shows the results for temporal and spatial relations. Results for all relations are available at `http://www.mpi-inf.mpg.de/yago-naga/yago2/`.

Technique	#Evaluated	Accuracy
`Concept Linker`	259	$97.71\% \pm 2.29\%$
`Infobox Template Type`	79	$97.68\% \pm 2.32\%$

Table 8.1: Results of evaluation of extraction techniques

The evaluation shows the very high accuracy of our extractors. The vast majority of facts, 97.33%, were judged correct. This results in an overall

[1] `hasLatitude` is extracted from Wikipedia only, hasGeoCoordinates combines Wikipedia coordinates and GeoNames coordinates

[2] For `occursSince/Until` we evaluated extracted facts only, not the ones created automatically using the methods described in Section 4

[3] Excluding 239,119 facts imported from GeoNames without evaluating them

Relation	#Total Facts	#Evaluated	Accuracy
actedIn	107,409	80	$95.75\% \pm 2.53\%$
created	206,860	79	$95.48\% \pm 3.54\%$
exports	522	145	$93.33\% \pm 3.84\%$
graduatedFrom	11,889	70	$97.40\% \pm 2.60\%$
hasExport	161	76	$96.34\% \pm 3.43\%$
hasGivenName	692,800	148	$95.44\% \pm 3.11\%$
hasLatitude[1]	190,224	80	$97.71\% \pm 2.29\%$
holdsPoliticalPosition	2,233	106	$94.61\% \pm 3.91\%$
influences	15,633	74	$96.25\% \pm 3.51\%$
isInterestedIn	184	141	$93.15\% \pm 3.95\%$
isMarriedTo	12,076	67	$97.29\% \pm 2.71\%$
subclassOf	571,641	339	$95.64\% \pm 2.09\%$
type	8,082,256	173	$96.65\% \pm 2.45\%$

Table 8.2: Results of evaluation of non-temporal, non-spatial relations with facts extracted from Wikipedia

Wilson center (weighted average over all relations) of 95.36%, with an average width of ±3.30%.

The crucial taxonomic relations are `type` (categorizing the individuals into classes) and `subclassOf` (linking a subclass to a super-class). The latter are extracted by the `Concept Linker`, which connects Wikipedia categories to WordNet synsets. Both relations have a Wilson center of more than 95%, demonstrating the very accurate integration of both resources. Relations between individuals, such as `graduatedFrom`, `influences`, or `isMarriedTo` are of even higher accuracy, as they are based on Wikipedia links between articles, which are of very good quality.

The relations that link individuals to classes, such as `isInterestedIn` or `exports/imports`, are of lower accuracy. The problem is that the extractors do not only have to extract the class name correctly, but they also have to disambiguate the class to the correct WordNet class (which is done using the `Concept Linker` technique). For example, the fact `UnitedStates imports medicine` is wrong if `medicine` is matched to the WordNet class "the branches of medical science that deal with nonsurgical techniques", instead of the correct "something that treats or prevents or alleviates the symptoms of disease". A last source of errors are incorrectly formatted literals in Wikipedia — handling all possible ways of formatting e. g. a date is nearly impossible. Still, even these difficult extractions show an accuracy of at least 93%.

Relation	#Total Facts	#Evaluated	Accuracy
isCitizenOf	24,190	100	98.15% ± 1.85%
diedIn	22,274	99	98.13% ± 1.87%
livesIn	16,405	89	97.93% ± 2.07%
	⋮		
hasGDP	160	122	92.91% ± 4.28%
hasFamilyName	694,146	130	92.59% ± 4.26%
hasHeight	23,893	133	92.02% ± 4.38%

Table 8.3: Results of evaluation of best and worst relations

Relation	#Total Facts	#Evaluated	Accuracy
diedIn	22,274	99	98.13% ± 1.87%
diedOnDate	315,528	88	97.91% ± 2.09%
happenedIn	5,192	68	95.93% ± 3.81%
happenedOnDate	22,039	106	97.34% ± 2.49%
occursSince/Until[2]	9,840	179	97.86% ± 1.84%
isLocatedIn	95,327[3]	68	95.93% ± 3.81%
livesIn	16,405	89	97.93% ± 2.07%
wasBornIn	56,415	59	96.94% ± 3.06%
wasBornOnDate	685,746	60	96.99% ± 3.01%
wasCreatedOnDate	467,194	137	95.59% ± 1.94%
wasDestroyedOnDate	24,218	85	95.59% ± 3.77%

Table 8.4: Results of evaluation of temporal and spatial relations

8.2 GeoNames Matching

We evaluated the automated class matching (Section 5.1.2) with the GeoNames-WordNet matches of GeoWordNet [20] as ground truth. We found that we match 86.7% of GeoNames to YAGO2 classes. This match has a very high precision of 94.1% – similar to the accuracy of our YAGO2 extractors. As WordNet's sense inventory is very fine-grained, some of the wrong matches are actually still valid. Take "library" as an example: GeoWordNet matches this to the WordNet "library" sense described by "a building that houses a collection of books and other materials". Our automated approach matches it to "library" described by "a depository built to contain books and other materials for reading and study". We count this mapping as error, so the precision is in fact even higher than the 94.1% we find by comparing against

GeoWordNet.

8.3 Size of YAGO2

YAGO2 contains a huge amount of facts from Wikipedia. The number of locations we integrate from GeoNames, as well as the multilingual class names imported from Universal WordNet [12] further increases this number. We give numbers for the core of YAGO2 (without facts GeoNames or UWN), as well as for the full YAGO with everything included, in Table 8.5. Table 8.6 breaks these numbers down by interesting classes of entities. Table 8.7 gives the number of time/location meta-facts. Finally, Table 8.8 gives the numbers of base-facts (facts between entities, such as `wasBornIn`, `interestedIn, type, or subclassOf`, semantic meta-facts extracted from Wikipedia (facts about facts, such as `occursSince` or `hasSuccessor`), and semantic meta-facts deduced by our rules in Section 4 and 5. Another type of meta-fact are provenance facts, keeping book of where, when, and how a fact has been extracted.

Type	Number in YAGO2	YAGO2 incl. GeoNames
Classes	562,312	562,954
Entities	2,661,594	9,849,496
Facts	253,213,842	451,726,172
Relations	99	101

Table 8.5: YAGO core and full numbers

Class	#Entities
People	882,534
Organizations	240,047
Locations	695,712 (7,569,708 incl. GeoNames)
Events	212,236
Other	631,065

Table 8.6: Number of entities by class

Relation	#Facts
occursSince/Until	27,169,970
occursIn	3,066,757

Table 8.7: Number of time and location meta-facts

Type	Number of Facts
base facts	27,550,575
semantic (extracted) meta-facts	5,715,127
semantic (deduced) meta-facts	30,236,727
provenance meta-facts	189,711,413

Table 8.8: Number of base and meta-facts in YAGO2 (without GeoNames)

9 Related Work

Going beyond the classical knowledge sources Cyc [26] and WordNet [18] and recently hand-crafted but small ontologies such as GeoWordNet [20], most projects on automatic knowledge base construction have drawn from semistructured elements in Wikipedia and other Web sources: infoboxes, category names, tables, lists. [1, 14, 50] and the references given there provide an overview of recent work along these lines. Commercial endeavors include Cyc, Freebase, Trueknowledge, and Wolframalpha. Academic work of comprehensive scale and ontological rigor includes DBpedia [4, 7], Omega [34], WikiTaxonomy [36, 35], and YAGO [41, 42]. WikiTaxonomy and YAGO have emphasized high precision from Wikipedia categories, and aligning this with WordNet into a much richer class system. YAGO has additionally harvested infoboxes. DBpedia has placed emphasis on high recall from the infoboxes, and makes use of the YAGO taxonomy. Omega integrated WordNet with separate upper-level ontologies and populated various classes with instance collections, including locations from geo gazetteers. Predating the advent of Wikipedia harvesting, Omega's size is considerably smaller than that of DBpedia or YAGO. None of these approaches has placed emphasis on attaching time and space consistently to its facts and entities.

The Kylin/KOG project [51] has developed learning-based methods for automatically typing Wikipedia entities and generating infobox-style facts; however, this project has not yet led to the construction of large-scale knowledge bases of near-human precision. Very recently, UWN and MENTA [12, 13] have added a multilingual dimension to entity and concept names, and also the class system. This work is complementary to YAGO and YAGO2; we have integrated UWN into YAGO2.

In addition to this line of work on ontologically rigorous knowledge, a suite of recent projects have been working on large-scale gathering of entity-relationship-oriented knowledge from arbitrary Web sources and natural-language documents. These include KnowItAll and its successor TextRunner [16, 5], the Omnivore system [11], work on distilling Web tables and lists into

facts [10], the ReadTheWeb project [47], the StatSnowball methods used for building EntityCube [53], our own work on SOFIE [43], and others. These approaches have not focused on the temporal and geographical dimension so far.

The most prominent work on extracting temporal facts is TARSQI [46]. However, it is limited to capturing explicit dates and phrases such as "a week ago" or "last year" (and mapping them into an explicit date relative to a reference date like the date of a news article). The NLP community has had event extraction tasks in its TempEval workshop series [45], using representations such as TimeML and reference corpora such as Timebank [8]. More recent work in this area is from Strötgen and Gertz [40], done as part of the TempEval track in the SemEval workshop. There is no attempt, though, to connect these dates to corresponding entity-relationship facts.

Temporal knowledge as a first-class citizen in richly populated knowledge bases has been addressed by only a few prior papers: the TOB framework of [52], our own preliminary attempt towards T-YAGO [49], and the TIE approach of [27]. TOB [52] focused on extracting business-related temporal facts such as terms of CEOs. It used a heavy NLP machinery, with deep parsing of every sentence, and machine-learning methods for classifiers for specifically interesting relations. It worked well, but was computationally expensive, required extensive training, and could not easily generalize to other relations. The work on T-YAGO [49] focused on extracting relevant timepoints and intervals from semistructured data in Wikipedia: dates in category names, lists, tables, infoboxes. It was rather preliminary and did not aim at the exhaustive anchoring of an ontology in time and space. There was no support for processing free text. Finally, the TIE approach [27] uses training data with fine-grained annotations to learn an inference model based on Markov Logic. This involves using consistency constraints on the relative ordering of events.

The general theme of temporal knowledge is an old AI topic [19], but prior work concentrated on representational models without any attempts at populated knowledge bases. The standard textbook by Russel and Norvig [38] refers to t-facts as *fluents*: instances of relations whose validity is a function of time. There is also recent awareness of temporal IR: ranking of search results for keyword queries with temporal phrases [3, 6, 29]. [48] focused on logics-based querying over uncertain t-facts, but did not address the extraction and fact harvesting process.

Prior literature on RDF-based data models [44] has proposed to extend SPO triples into quadruples, *quads* for short, where the fourth component, sometimes referred to as "color", primarily serves to represent the provenance of the triple, but can also be used for other kinds of meta-facts. The work

that extends ontologies along the temporal and/or geographical dimension includes Gutierrez et al. [22]. They have introduced a temporal semantics for RDF, coined *Temporal RDF*, where time is modeled as a label on RDF triples, giving each triple a validity time. For querying, the time interval can be specified as an annotation to standard RDF query. Pugliese et al. [37] propose an efficient implementation of a time index supporting such queries. A very recent approach by Koubarakis and Kyzirakos [25] combines the semantics of spatial and temporal constraint databases to create a time- and space-aware extension of RDF called stRDF, as well as an equivalent extension to SPARQL. Perry et al. [32] proposed an ontological model for a time- and space-aware ontology, together with a set of temporal and spatial query operators. They also implemented their system in a relational DBMS, similar to our approach. We do not allow time queries as complex as [22] or [37], where a time interval can be combined with a natural number denoting the time of occurrences of a fact in the given interval. For our approach, simply specifying the interval is enough. We also do not allow more complex spatial representations such as polygons, as [32] and [25] do. Our main contribution is in how to deal with time when it is not only present as annotation of thematic facts, but present as a base fact itself, and how to propagate the available data to all relevant facts. For a more general overview on the field of spatio-temporal databases, there is an overview paper of the field by Pelekis et al. [30].

The field of geographical gazetteers is a very old one, the first geographical gazetteers have been created hundreds of years ago to collect information associated with geographical locations. More recently, the idea of gazetteers has been expanded, e. g. by Feinberg et al. [17], to encompass named periods, such as "The French Revolution" or "Renaissance". In contrast to geo-gazetteers, which store the geographical coordinates for entities, these temporal gazetteers store the time periods for the named events. One such temporal directory was created using Library of Congress subject headings by Petras et al. [33]. YAGO2 takes this idea further, by combining the temporal and geographical data and semantic information. It knows the periods of named events or lifetimes of persons and extends them by the semantic knowledge that connect these periods – something that increases the value of YAGO2 beyond that of a gazetteers, as the new knowledge can be used, e.g., for disambiguating strings occurring in texts to time periods in the dictionary.

10 Conclusions

We have developed methodology for enriching large knowledge bases of entity-relationship-oriented facts along the dimensions of time and space, and we have demonstrated the practical viability of this approach by the YAGO2 ontology comprising more than 80 million facts of near-human quality. We believe that such spatio-temporal knowledge is a crucial asset for many applications including entity linkage across independent sources (e.g., in the Linked-Data cloud [7]) and semantic search. Along the latter lines, we think that the combined availability of ontological facts and contextual keywords makes querying and knowledge discovery much more convenient and effective.

Regardless of the impressive extent and great success of Wikipedia-centric knowledge bases in the style of DBpedia, YAGO, Freebase, or YAGO2, there is a wealth of latent knowledge beyond Wikipedia in the form of natural-language text. This includes biographies and homepages of people or organizations, scientific publications, daily news, digests of contemporary events and trends, and more. Tapping on these kinds of sources requires learning- and reasoning-based forms of information extraction, as pursued, for example, by our prior work on SOFIE [43]. In this context, too, considering the temporal and spatial dimensions would be of utmost importance, but here the complexity of natural language poses major obstacles. Early work along these lines include [49, 27]. Much more refined and intensive efforts are needed, though. Our future work aims at this open challenge of extracting, reconciling, and integrating spatio-temporal knowledge from free-text sources.

Bibliography

[1] First workshop on automated knowledge base construction. `http://akbc.xrce.xerox.com/`, 2010.

[2] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Commun. ACM*, 26(11):832–843, 1983.

[3] O. Alonso, M. Gertz, and R. Baeza-Yates. On the Value of Temporal Information in Information Retrieval. *SIGIR Forum*, 41:35–41, December 2007.

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea*, pages 722–735, 2007.

[5] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open Information Extraction from the Web. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India*, pages 2670–2676, 2007.

[6] K. Berberich, S. J. Bedathur, O. Alonso, and G. Weikum. A Language Modeling Approach for Temporal Information Needs. In *Proceedings of the 32nd European Conference on Information Retrieval (ECIR 2010)*, pages 13–25, 2010.

[7] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A Crystallization Point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154 – 165, 2009. The Web of Data.

[8] B. Boguraev, J. Pustejovsky, R. Ando, and M. Verhagen. TimeBank Evolution as a Community Resource for TimeML Parsing. *Language Resources and Evaluation*, 41:91–115, 2007.

[9] L. D. Brown, T. T. Cai, and A. Dasgupta. Interval Estimation for a Binomial Proportion. *Statistical Science*, 16(2):101–133, May 2001.

[10] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: Exploring the Power of Tables on the Web. *Proc. VLDB Endow.*, 1(1):538–549, 2008.

[11] Cafarella, Michael. Extracting and Querying a Comprehensive Web Database. In *CIDR, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, Online Proceedings, 2009*, 2009.

[12] G. de Melo and G. 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.

[13] G. de Melo and G. Weikum. MENTA: Inducing Multilingual Taxonomies from Wikipedia. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM 2010)*, Toronto, Canada, October 2010. ACM.

[14] A. Doan, L. Gravano, R. Ramakrishnan, and S. Vaithyanathan, editors. *Special Section on Managing Information Extraction*, volume 37(4), 2008.

[15] S. Elbassuoni, M. Ramanath, R. Schenkel, and G. Weikum. Searching RDF Graphs with SPARQL and Keywords. *IEEE Data Eng. Bull.*, 33(1):16–24, 2010.

[16] O. Etzioni, M. J. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised Named-Entity Extraction from the Web: An Experimental Study. *Artif. Intell.*, 165(1):91–134, 2005.

[17] M. Feinberg, R. Mostern, S. Stone, and M. Buckland. Application of Geographical Gazetteer Standards to Named Time Periods. Technical report, Berkeley, 2003.

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

[19] M. Fisher, D. Gabbay, and L. Vila. *Handbook of Temporal Reasoning in Artificial Intelligence (Foundations of Artificial Intelligence (Elsevier))*. Elsevier Science Inc., New York, NY, USA, 2005.

[20] F. Giunchiglia, V. Maltese, F. Farazi, and B. Dutta. GeoWordNet: A Resource for Geo-spatial Applications. In *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC, Heraklion, Crete, Greece, 2010*, pages 121–136, 2010.

[21] C. Gutierrez, C. Hurtado, and A. Vaisman. Temporal RDF. In *The Semantic Web: Research and Applications*, volume 3532 of *Lecture Notes in Computer Science*, pages 93–107. Springer Berlin / Heidelberg, 2005.

[22] C. Gutierrez, C. A. Hurtado, and A. Vaisman. Introducing Time into RDF. *IEEE Transactions on Knowledge and Data Engineering*, 19:207–218, 2007.

[23] J. M. Hellerstein, J. F. Naughton, and A. Pfeffer. Generalized Search Trees for Database Systems. In U. Dayal, P. M. D. Gray, and S. Nishio, editors, *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, Zurich, Switzerland*, pages 562–573. Morgan Kaufmann, 1995.

[24] D. Kolas and T. Self. Spatially-Augmented Knowledgebase. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea*, pages 792–801, 2007.

[25] 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*, volume 6088 of *Lecture Notes in Computer Science*, pages 425–439. Springer Berlin / Heidelberg, 2010.

[26] D. B. Lenat. CYC: A Large-Scale Investment in Knowledge Infrastructure. *Commun. ACM*, 38(11):32–38, 1995.

[27] X. Ling and D. S. Weld. Temporal Information Extraction. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA*, 2010.

[28] T. Neumann and G. Weikum. The RDF-3X Engine for Scalable Management of RDF Data. *VLDB J.*, 19(1):91–113, 2010.

[29] M. Pasca. Towards Temporal Web Search. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1117–1121, 2008.

[30] N. Pelekis, B. Theodoulidis, I. Kopanakis, and Y. Theodoridis. Literature Review of Spatio-Temporal Database Models. *Knowl. Eng. Rev.*, 19(3):235–274, 2004.

[31] M. Perry, F. Hakimpour, and A. Sheth. Analyzing Theme, Space, and Time: An Ontology-Based Approach. In *GIS '06: Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, pages 147–154, New York, NY, USA, 2006. ACM.

[32] M. Perry, A. P. Sheth, F. Hakimpour, and P. Jain. Supporting Complex Thematic, Spatial and Temporal Queries over Semantic Web Data. In *GeoS'07: Proceedings of the 2nd international conference on GeoSpatial semantics*, pages 228–246, Berlin, Heidelberg, 2007. Springer-Verlag.

[33] V. Petras, R. R. Larson, and M. K. Buckland. Time Period Directories: A Metadata Infrastructure for Placing Events in Temporal and Geographic Context. In *ACM/IEEE Joint Conference on Digital Libraries, JCDL 2006, Chapel Hill, NC, USA Proceedings*, pages 151–160, 2006.

[34] A. Philpot, E. H. Hovy, and P. Pantel. *Ontology and the Lexicon*, chapter The Omega Ontology. Cambridge University Press, 2008, 2008.

[35] S. P. Ponzetto and R. Navigli. Large-Scale Taxonomy Mapping for Restructuring and Integrating Wikipedia. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 2083–2088, 2009.

[36] S. P. Ponzetto and M. Strube. Deriving a Large-Scale Taxonomy from Wikipedia. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, Vancouver, British Columbia, Canada, 2007*, pages 1440–1445, 2007.

[37] A. Pugliese, O. Udrea, and V. S. Subrahmanian. Scaling RDF with Time. In *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China*, pages 605–614, 2008.

[38] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[39] A. Sheth and M. Perry. Traveling the Semantic Web through Space, Time, and Theme. *IEEE Internet Computing*, 12:81–86, 2008.

[40] J. Strötgen and M. Gertz. HeidelTime: High Quality Rule-Based Extraction and Normalization of Temporal Expressions. In *SemEval '10: Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324, Morristown, NJ, USA, 2010. Association for Computational Linguistics.

[41] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A Core of Semantic Knowledge. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 697–706, New York, NY, USA, 2007. ACM.

[42] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A Large Ontology from Wikipedia and WordNet. *J. Web Sem.*, 6(3):203–217, 2008.

[43] F. M. Suchanek, M. Sozio, and G. Weikum. SOFIE: A Self-Organizing Framework for Information Extraction. In *International World Wide Web conference (WWW 2009)*, New York, NY, USA, 2009. ACM Press.

[44] O. Udrea, D. R. Recupero, and V. S. Subrahmanian. Annotated RDF. *ACM Trans. Comput. Log.*, 11(2), 2010.

[45] M. Verhagen, R. J. Gaizauskas, F. Schilder, M. Hepple, J. Moszkowicz, and J. Pustejovsky. The TempEval Challenge: Identifying Temporal Relations in Text. *Language Resources and Evaluation*, 43(2):161–179, 2009.

[46] M. Verhagen, I. Mani, R. Sauri, R. Knippen, S. B. Jang, J. Littman, A. Rumshisky, J. Phillips, and J. Pustejovsky. Automating Temporal Annotation with TARSQI. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, ACL '05, pages 81–84, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[47] R. C. Wang and W. W. Cohen. Iterative Set Expansion of Named Entities Using the Web. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 1091–1096, Washington, DC, USA, 2008. IEEE Computer Society.

[48] Y. Wang, M. Yahya, and M. Theobald. Time-aware Reasoning in Uncertain Knowledge Bases. In *Proceedings of Management of Uncertain Data (MUD), Singapore 2010*, 2010.

[49] Y. Wang, M. Zhu, L. Qu, M. Spaniol, and G. Weikum. Timely YAGO: Harvesting, Querying, and Visualizing Temporal Knowledge from Wikipedia. In *EDBT 2010, 13th International Conference on Extending Database Technology, Lausanne, Switzerland*, pages 697–700, 2010.

[50] G. Weikum and M. Theobald. From Information to Knowledge: Harvesting Entities and Relationships from Web Sources. In *PODS '10: Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems of data*, pages 65–76, New York, NY, USA, 2010. ACM.

[51] F. Wu and D. S. Weld. Automatically Refining the Wikipedia Infobox Ontology. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 635–644, New York, NY, USA, 2008. ACM.

[52] Q. Zhang, F. M. Suchanek, L. Yue, and G. Weikum. TOB: Timely Ontologies for Business Relations. In *11th International Workshop on Web and Databases 2008 (WebDB 2008)*. ACM, 2008.

[53] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J.-R. Wen. StatSnowball: A Statistical Approach to Extracting Entity Relationships. In *WWW '09: Proceedings of the 18th international conference on World Wide Web*, pages 101–110, New York, NY, USA, 2009. ACM.

Below you find a list of the most recent research reports of the Max-Planck-Institut für Informatik. Most of them are accessible via WWW using the URL `http://www.mpi-inf.mpg.de/reports`. 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 and Publications –
Campus E 1 4

D-66123 Saarbrücken

E-mail: `library@mpi-inf.mpg.de`

MPI-I-2010-RG1-001	M. Suda, C. Weidenbach, P. Wischnewski	On the saturation of YAGO
MPI-I-2010-5-006	A. Broschart, R. Schenkel	Real-time text queries with tunable term pair indexes
MPI-I-2010-5-002	M. Theobald, M. Sozio, F. Suchanek, N. Nakashole	URDF: Efficient Reasoning in Uncertain RDF Knowledge Bases with Soft and Hard Rules
MPI-I-2010-5-001	K. Berberich, S. Bedathur, O. Alonso, G. Weikum	A language modeling approach for temporal information needs
MPI-I-2009-RG1-005	M. Horbach, C. Weidenbach	Superposition for fixed domains
MPI-I-2009-RG1-004	M. Horbach, C. Weidenbach	Decidability results for saturation-based model building
MPI-I-2009-RG1-002	P. Wischnewski, C. Weidenbach	Contextual rewriting
MPI-I-2009-RG1-001	M. Horbach, C. Weidenbach	Deciding the inductive validity of $\forall\exists^*$ queries
MPI-I-2009-5-007	G. Kasneci, G. Weikum, S. Elbassuoni	MING: Mining Informative Entity-Relationship Subgraphs
MPI-I-2009-5-006	S. Bedathur, K. Berberich, J. Dittrich, N. Mamoulis, G. Weikum	Scalable phrase mining for ad-hoc text analytics
MPI-I-2009-5-005	G. de Melo, G. Weikum	Towards a Universal Wordnet by learning from combined evidenc
MPI-I-2009-5-004	N. Preda, F.M. Suchanek, G. Kasneci, T. Neumann, G. Weikum	Coupling knowledge bases and web services for active knowledge
MPI-I-2009-5-003	T. Neumann, G. Weikum	The RDF-3X engine for scalable management of RDF data
MPI-I-2009-5-002	M. Ramanath, K.S. Kumar, G. Ifrim	Generating concise and readable summaries of XML documents
MPI-I-2009-4-006	C. Stoll	Optical reconstruction of detailed animatable human body models
MPI-I-2009-4-005	A. Berner, M. Bokeloh, M. Wand, A. Schilling, H. Seidel	Generalized intrinsic symmetry detection
MPI-I-2009-4-004	V. Havran, J. Zajac, J. Drahokoupil, H. Seidel	MPI Informatics building model as data for your research
MPI-I-2009-4-003	M. Fuchs, T. Chen, O. Wang, R. Raskar, H.P.A. Lensch, H. Seidel	A shaped temporal filter camera
MPI-I-2009-4-002	A. Tevs, M. Wand, I. Ihrke, H. Seidel	A Bayesian approach to manifold topology reconstruction
MPI-I-2009-4-001	M.B. Hullin, B. Ajdin, J. Hanika, H. Seidel, J. Kautz, H.P.A. Lensch	Acquisition and analysis of bispectral bidirectional reflectance distribution functions
MPI-I-2008-RG1-001	A. Fietzke, C. Weidenbach	Labelled splitting
MPI-I-2008-5-004	F. Suchanek, M. Sozio, G. Weikum	SOFI: a self-organizing framework for information extraction
MPI-I-2008-5-003	G. de Melo, F.M. Suchanek, A. Pease	Integrating Yago into the suggested upper merged ontology
MPI-I-2008-5-002	T. Neumann, G. Moerkotte	Single phase construction of optimal DAG-structured QEPs
MPI-I-2008-5-001	G. Kasneci, M. Ramanath, M. Sozio, F.M. Suchanek, G. Weikum	STAR: Steiner tree approximation in relationship-graphs

MPI-I-2008-4-003	T. Schultz, H. Theisel, H. Seidel	Crease surfaces: from theory to extraction and application to diffusion tensor MRI
MPI-I-2008-4-002	D. Wang, A. Belyaev, W. Saleem, H. Seidel	Estimating complexity of 3D shapes using view similarity
MPI-I-2008-1-001	D. Ajwani, I. Malinger, U. Meyer, S. Toledo	Characterizing the performance of Flash memory storage devices and its impact on algorithm design
MPI-I-2007-RG1-002	T. Hillenbrand, C. Weidenbach	Superposition for finite domains
MPI-I-2007-5-003	F.M. Suchanek, G. Kasneci, G. Weikum	Yago : a large ontology from Wikipedia and WordNet
MPI-I-2007-5-002	K. Berberich, S. Bedathur, T. Neumann, G. Weikum	A time machine for text search
MPI-I-2007-5-001	G. Kasneci, F.M. Suchanek, G. Ifrim, M. Ramanath, G. Weikum	NAGA: searching and ranking knowledge
MPI-I-2007-4-008	J. Gall, T. Brox, B. Rosenhahn, H. Seidel	Global stochastic optimization for robust and accurate human motion capture
MPI-I-2007-4-007	R. Herzog, V. Havran, K. Myszkowski, H. Seidel	Global illumination using photon ray splatting
MPI-I-2007-4-006	C. Dyken, G. Ziegler, C. Theobalt, H. Seidel	GPU marching cubes on shader model 3.0 and 4.0
MPI-I-2007-4-005	T. Schultz, J. Weickert, H. Seidel	A higher-order structure tensor
MPI-I-2007-4-004	C. Stoll, E. de Aguiar, C. Theobalt, H. Seidel	A volumetric approach to interactive shape editing
MPI-I-2007-4-003	R. Bargmann, V. Blanz, H. Seidel	A nonlinear viseme model for triphone-based speech synthesis
MPI-I-2007-4-002	T. Langer, H. Seidel	Construction of smooth maps with mean value coordinates
MPI-I-2007-4-001	J. Gall, B. Rosenhahn, H. Seidel	Clustered stochastic optimization for object recognition and pose estimation
MPI-I-2007-2-001	A. Podelski, S. Wagner	A method and a tool for automatic veriication of region stability for hybrid systems
MPI-I-2007-1-003	A. Gidenstam, M. Papatriantafilou	LFthreads: a lock-free thread library
MPI-I-2007-1-002	E. Althaus, S. Canzar	A Lagrangian relaxation approach for the multiple sequence alignment problem
MPI-I-2007-1-001	E. Berberich, L. Kettner	Linear-time reordering in a sweep-line algorithm for algebraic curves intersecting in a common point
MPI-I-2006-5-006	G. Kasnec, 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