YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia

Johannes Hoffart[a], Fabian M. Suchanek[b], Klaus Berberich[a], Gerhard Weikum[a]

[a]*Max Planck Institute for Informatics, Germany*
[b]*INRIA Saclay, France*

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 447 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

---

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 [1] and WordNet [2] 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).

Email address: `jhoffart@mpi-inf.mpg.de` (Johannes Hoffart)

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 [3], KnowItAll [4, 5], Omega [6], WikiTaxonomy [7, 8], and YAGO [9, 10], 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 [11]. 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) [12]. 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. In addition, YAGO2 incorporates carefully selected keywords and keyphrases that characterize entities; these are automatically gathered from the contexts where facts are extracted from. As no knowledge base can ever be complete, the conteXtual annotations further enhance the capabilities for querying and interactive exploration. The full YAGO2 interface provides SPOTLX tuples to this end.

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(X) representation* of spatio-temporally enhanced facts, with expressive and easy-to-use *querying*;

- exemplary demonstrations of the added value obtained by the spatio-temporal knowledge in YAGO2, by showing how this aids in extrinsic tasks like question answering and named entity disambiguation.

The result is YAGO2, available at `http://www.yago-knowledge.org`. It contains more than 447 million facts for 9.8 million entities (if GeoNames entities are included). Without GeoNames entities, it still contains 124 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 30 million facts are associated with their occurrence time, and more than 17 million with the location of their occurrence. The time of existence is known for 47% of all entities, the location for 30%. Sampling-based manual assessment shows that YAGO2 has a precision (i.e., absence of false positives) of 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(X) model and querying. Section 8 presents the evaluation of YAGO2. Section 9 presents the exemplary extrinsic tasks that we carried out to demonstrate the added value of YAGO2. Section 10 reviews related work, before Section 11 concludes.

2. The YAGO Knowledge Base

YAGO was originally introduced in [9]. 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 [2] (e.g., `canadian poet` becomes a subclass of the WordNet synset `poet`). This linkage allowed YAGO to create a strong taxonomy, which is key not only to the applications of the knowledge base, but also to the consistency checks that YAGO can perform on its data.

The linkage algorithm (which we still use in YAGO2) proceeds as follows: For each category of a page, it determines the head word of the category name through shallow noun phrase parsing. In the example of `Canadian poets`, the head word is `poets`. It checks whether the head word is in plural. If so, it proposes the category as a class and the article entity as an instance. This process effectively distinguishes thematic categories (such as `Canadian poetry`) from conceptual ones, by the simple observation that only countable nouns can appear in plural form and only countable nouns can be ontological classes. The class is linked to the WordNet taxonomy by choosing the most frequent sense of the head word in WordNet. This simple disambiguation strategy proved surprisingly accurate. In addition, YAGO contains a list of a handful of manually compiled exceptions. These are head words that are not conceptual even though they appear in plural (such as `stubs` in `Canadian poetry stubs`). A second list of exceptions contains words that do not map to their most frequent sense, but to a different sense. The word `capital`, e.g., refers to the main city of a country in the majority of cases and not to the financial amount, which is the most frequent sense in WordNet. We have slightly extended these lists over time and provide a new evaluation of the results in this paper.

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. These instances are called facts: triples of an entity, a relation, and another entity. 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 [9].

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 extend the list of exceptions for linking the Wikipedia categories to the WordNet synsets (explained in Section 2). The list of category head words that should not be mapped to their primary sense in WordNet grew to 60. 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.

In our current implementation, the extraction rules cover some 200 infobox patterns, some 90 category patterns, and around a dozen patterns for dealing with disambiguation pages. Our patterns map to around 100 relations. They aim to cover the 200 most frequent infobox attributes in Wikipedia. They exclude infobox attributes that are used inconsistently in Wikipedia. One example is the attribute *history*, which contains sometimes date information and sometimes full text descriptions. They also exclude attributes that contain natural language text, or that contain mostly entities that are not in Wikipedia, because YAGO could not type check these facts.

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. 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 the 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 two thirds of all entities (not including the GeoNames locations).

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 Existence times of entities are associated with relations that have an identifying character, e.g. `hasISBN` or `isCalled`, but also with other relations that imply permanent association to an entity. An example here is the `type` relation: although it might change over time (`BobDylan` was not always a `singer`), they are mostly permanent (`BobDylan` was, is, and always will be a `person`). We call all these relations permanent relations. 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:

$$\frac{\begin{array}{c}\texttt{\$id: \$s \$p \$o;}\\ \texttt{\$p type objectStartRelation;}\\ \texttt{\$o startsExistingOn \$b}\end{array}}{\begin{array}{c}\texttt{\$id occursSince \$b;}\\ \texttt{\$id occursUntil \$b}\end{array}}$$

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:

$$\begin{array}{c}
\texttt{\$id: \$s \$p \$o;} \\
\texttt{\$p type subjectEndRelation;} \\
\texttt{\$s endsExistingOn \$e} \\
\hline
\texttt{\$id occursSince \$e;} \\
\texttt{\$id occursUntil \$e}
\end{array}$$

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 2011-06-15`. 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 (both 9335240 and 9334396). 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].

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 191,200 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, as well as neighboring countries. 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. Otherwise, we do not match them.

[1]Guidelines from `http://en.wikipedia.org/wiki/Wikipedia:WikiProject_Geographical_coordinates`, last accessed on 2011-06-30

3. In the end, we add all the unmatched GeoNames entities as new individual entities to YAGO2, 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 120,281 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 [13]. 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 YAGO2 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 tokenized, 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 `yagoGeoEntity` class, making the mapping as general as possible.

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.

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)
begin
    Cand ← {y ∈ YAGO | y or syn(y) = geo_class}
    if Cand = ∅ then
        Cand ← {y ∈ YAGO | y or syn(y) = head(geo_class)}
        if Cand = ∅ then
            return no match
    GeoCand ← Cand ∩ YagoGeo
    if |GeoCand = 1| then
        return g ∈ GeoCand
    else if |GeoCand| > 1 then
        Cand ← GeoCand
        /* Cand contains original set or only classes with geo meaning */
    best ← $\mathrm{argmax}_{c \in \mathsf{Cand}}(jacc\_sim(\mathsf{gloss(g)}, \mathsf{gloss(c)}))$
    if jacc_sim(g,best) > 0.0 then
        return best
    else
        return yagoGeoEntity
        /* as default, map to general yagoGeoEntity class */
```

This matching process augments YAGO2 with over 7 million geo-entities and over 320 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 neighboring countries, as well as 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 are those relations that imply a direct association with the entity. 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:

$$\frac{\begin{array}{c}\texttt{\$id: \$s \$p \$o;}\\ \texttt{\$p type permanentRelation;}\\ \texttt{\$s placedIn \$l}\end{array}}{\texttt{\$id occursIn \$l}}$$

$$\frac{\begin{array}{c}\texttt{\$id: \$s \$p \$o;}\\ \texttt{\$p type permanentRelation;}\\ \texttt{\$s type yagoGeoEntity}\end{array}}{\texttt{\$id occursIn \$s}}$$

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:

$$\frac{\begin{array}{c}\texttt{\$id: \$s \$p \$o;}\\ \texttt{\$p type relationLocatedByObject;}\\ \texttt{\$o type yagoGeoEntity}\end{array}}{\texttt{\$id occursIn \$o}}$$

$$\frac{\begin{array}{c}\texttt{\$id: \$s \$p \$o;}\\ \texttt{\$p type relationLocatedByObject;}\\ \texttt{\$o placedIn \$l}\end{array}}{\texttt{\$id occursIn \$l}}$$

(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.

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 and keyphrases associated with the entity. We extract more than 82 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) [14] 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% [14]. 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 [9], 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

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

joining facts from R with their associated information from T, L, and C. Here, $\bowtie$ 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 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.

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 1: SPOTL(X)-View Example: Grateful Dead performing “The Closing of Winterland” in San Francisco on New Year’s Eve of 1978

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 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 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 1. Our time predicates are a subset of those identified by Allen [16]. 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 referencing entities by noun phrases that refer to their canonical name, 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" .
```

More elaborate examples are available in [17], which also discusses the query interface in more detail. 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 [18]. 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. For the space dimension we use the freely available PostGIS extension (http://postgis.refractions.net) to build a spatial index (based on GiST [19]). 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. Factual Evaluation and Numbers

Our main goal for the construction of the YAGO2 ontology was near-human accuracy. This section presents an evaluation of the knowledge base quality. 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 WordNet synsets. 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 conducted 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 only considers the "semantic" relations (such as `wasBornOnDate`) and not the "technical" relations (such as `hasWikipediaURL`). In our methodology [9], 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 is presented next to the fact. Thus, the judges evaluate the correctness of YAGO2 with respect to the content of Wikipedia. We do not assess the factual correctness of Wikipedia itself. We used the Wikipedia dump from 2010-08-17 for the YAGO2 extraction and evaluation.

Relation	#Total Facts	#Evaluated	Accuracy
`actedIn`	126,636	69	97.36% ± 2.64%
`created`	225,563	94	98.04% ± 1.96%
`exports`	522	113	93.22% ± 4.32%
`graduatedFrom`	15,583	57	96.84% ± 3.16%
`hasExport`	161	61	95.50% ± 4.21%
`hasGender`	804,747	50	94.58% ± 5.07%
`hasGivenName`	746,492	134	97.16% ± 2.43%
`hasLatitude`[2]	311,481	47	96.22% ± 3.78%
`holdsPoliticalPosition`	3,550	81	94.20% ± 4.53%
`influences`	18,653	58	95.28% ± 4.42%
`isInterestedIn`	296	93	92.85% ± 4.83%
`isMarriedTo`	27,708	58	96.89% ± 3.11%
`subclassOf`	367,040	339	93.42% ± 2.67%
`type`	8,414,398	208	97.68% ± 1.83%

Table 2: Evaluation of non-temporal, non-spatial facts extracted from Wikipedia

For 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.

26 judges participated in our evaluation. Over the course of a week, they evaluated a total number of 5864 facts. This gave us an accuracy value for each sample. We estimate the accuracy of the entire pool by the fraction of samples that were assessed as true, and we compute a Wilson confidence interval [20] for each pool. We kept on evaluating until the confidence interval was smaller than ±5%. This ensures that our findings are statistically significant.

Table 2 describes the results for some of the important non-temporal, non-spatial relations. Table 3 shows three relations with best and worst accuracy, respectively. Table 4 finally shows the results for temporal and spatial relations. Results for all relations are available at `http://www.yago-knowledge.org`.

The evaluation shows the very high accuracy of our extractors. The vast majority of facts, 97.80%, were judged correct. This results in an overall Wilson center (weighted average over all relations) of 95.40% with a width of ±3.69%.

The crucial taxonomic relations are `type` (categorizing the individuals into classes) and `subclassOf` (linking a subclass to a super-class). Both relations

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

Relation	#Total Facts	#Evaluated	Accuracy
`created`	225,563	94	98.04% ± 1.96%
`diedIn`	28,834	88	97.91% ± 2.09%
`happenedOnDate`	27,563	94	97.86% ± 2.14%
	⋮		
`hasHeight`	26,477	120	91.99% ± 4.59%
`hasBudget`	547	95	90.97% ± 5.41%
`hasGDP`	175	93	90.79% ± 5.52%

Table 3: Evaluation of best and worst relations

Relation	#Total Facts	#Evaluated	Accuracy
`diedIn`	28,834	88	97.91% ± 2.09%
`diedOnDate`	315,659	79	97.68% ± 2.32%
`happenedIn`	11,694	51	96.50% ± 3.50%
`happenedOnDate`	27,563	94	97.86% ± 2.14%
`isLocatedIn`	436,184	51	96.50% ± 3.50%
`livesIn`	20,882	56	96.79% ± 3.21%
`wasBornIn`	90,181	49	96.36% ± 3.64%
`wasBornOnDate`	686,053	56	96.79% ± 3.21%
`wasCreatedOnDate`	507,733	110	97.43% ± 2.41%
`wasDestroyedOnDate`	23,617	72	96.15% ± 3.61%

Table 4: Evaluation of temporal and spatial relations

have a Wilson center of about 95%, demonstrating the highly 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. This is done by the same algorithm that links a Wikipedia category head noun to the corresponding WordNet synset (see Section 2). 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". Another 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 90%.

To estimate inter-annotator agreement, we had a random sample of 10% of the facts evaluated by 2 judges instead of one. We computed Fleiss' Kappa [21] as a measure of the agreement. The Kappa value is 0.37, which is generally regarded as fair agreement. In general, Fleiss' Kappa tends to have lower values if the distribution of the assessment labels is skewed [22]. In our case, the distribution is highly skewed, as more than 97% of the assessments have the label "true". Using the test procedure and the variance estimator described by Fleiss [21], we find that we can reject the null hypothesis that there is no agreement among annotators beyond chance, at any significance level larger than 7%. Speaking in absolute numbers, the judges disagreed on only 10 out of the 586 sample facts, a fraction of 1.7%.

8.2. GeoNames Matching

We evaluated the automated class matching (Section 5.1.2) with the GeoNames-WordNet matches of GeoWordNet [13] 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. Consider "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 number of facts from Wikipedia. The number of locations we integrate from GeoNames, as well as the multilingual class names imported from Universal WordNet (UWN) [14] further increase this number. We give numbers for the core of YAGO2 (without entities from GeoNames or facts from UWN), as well as for the full YAGO extension with everything included, in Table 5. Note that even when not including all GeoNames entities, we still extract the facts (e. g. to augment the `isLocatedIn` hierarchy), associating them with Wikipedia entities we could match to GeoNames. Table 6 breaks down the numbers by interesting classes of entities. Table 7 gives the number of time/location meta-facts, broken down by single relations in Table 8. Finally, Table 9 gives the numbers of base-facts (facts between entities, such as `wasBornIn`, `interestedIn, type, or subclassOf` and of semantic meta-facts, which are either extracted from Wikipedia (facts about facts, such as `occursSince`), or deduced by our rules in Section 4 and 5. Furthermore, there are one or more provenance facts for each of these semantic facts, which capture where, when, and how a fact has been extracted. Without GeoNames, there are 480 million provenance facts, and more than 1.6 billion when including GeoNames.

Type	# in YAGO2	# incl. GeoNames
Classes	365,372	365,372
Entities	2,648,387	9,756,178
Facts	124,333,521	447,470,256
Relations	104	114

Table 5: YAGO size for core and full extension

Class	#Entities	% existence time	% existence location
People	872,155	80.46	-
Groups	316,699	38.46	24.03
Artifacts	212,003	58.91	1.78
Events	187,392	60.16	16.01
Locations	687,414	13.34	100
Other	372,724	24.99	2.25
Total	2,648,387	47.05	30.62

Table 6: Number of entities by class, percentage associated with existence time/location

Relation	#Facts
occursSince/Until	30,820,228
occursIn	17,148,596

Table 7: Number of time and location meta-facts

relation	#Total Facts	% occur. times	% occur. locations
created	225,563	89.26	-
diedIn	28,834	99.56	100
diedOnDate	315,659	100	9.09
directed	38,184	98.81	-
endedOnDate	23,546	100	18.94
happenedOnDate	27,563	100	18.73
participatedIn	15,932	-	87.66
produced	23,769	99.17	-
startedOnDate	28,862	100	20.29
wasBornIn	90,181	97.14	100
wasBornOnDate	686,053	100	12.77
wasCreatedOnDate	507,733	100	<0.01
wasDestroyedOnDate	23,617	100	<0.01
worksAt	2,954	-	86.46

Table 8: Ratio of facts associated with occurrence times and locations per relation

Type	#Total Facts	# incl. GeoNames and Context
base facts	35,642,122	185,459,298
semantic meta-facts	88,691,399	262,010,958
total	124,333,521	447,470,256

Table 9: Number of base-facts and meta-facts in YAGO2

9. Task-Based Evaluation

The time, location, and context data in YAGO2 allow new tasks to be supported by a knowledge base that were previously infeasible or very cumbersome. We present two exemplary tasks making use of the newly available data and querying capabilities.

First, we use the new YAGO2 features to formulate (in a structured query format) and *answer questions* of temporal or spatial nature. This task demonstrates the usefulness and conciseness of the SPOTLX query language as well as the availability of temporal, spatial, and contextual data to answer advanced questions.

In the second task the new features are used for enhancing the task of *disambiguating* mentions of named entities in natural language text, mapping mentions to their corresponding canonical entities in the knowledge base.

9.1. Answering Spatio-Temporal Questions

Temporal or spatial relations play a big role in many question-answering settings. In this task we focus on two existing collections of such questions: the 15 questions of the GeoCLEF 2008 GiKiP Pilot[3], and a sample of temporal and spatial questions blocks from Jeopardy, available on J! Archive[4]. We first formulate the questions in the new SPOTLX style in a way that we find natural, then run these queries, and check the results for correctness. For GeoCLEF GiKiP, the original questions did not come with a set of answers, so we judged a query result against YAGO2 correct if it contained at least one correct answer.

Note that this task is merely an exemplary study, not a comprehensive evaluation. It serves to demonstrate the potential value of the spatio-temporal knowledge in YAGO2. For full-fledged natural-language QA, we would need an automatic mapping from questions to structured queries (see, e.g., [23] for work along these lines). For systematic evaluation, we would need a broader set of questions and comparison to state-of-the-art systems.

9.1.1. GeoCLEF GiKiP

The original intent of the GeoCLEF GiKiP Pilot is: "Find Wikipedia entries / articles that answer a particular information need which requires geo- graphical reasoning of some sort." [24]. The geographical reasoning part makes the 15 questions good candidates for mapping onto SPOTLX queries. All questions, their formulations and their results from YAGO2 are listed in Appendix A. As an example, consider question *GP13* (the original question is followed by our

[3]`http://www.linguateca.pt/GikiP/`
[4]`http://www.j-archive.com/`

formulation as a SPOTLX query):

Relevant documents describe navigable Afghan rivers whose length is greater than one thousand kilometers.

```
?x isA river locatedIn Afghanistan .
?x hasLength ?l .
?l isGreaterThan 1000km
```

The result is `Amu Darya`, a major river in Central Asia, part of which flows through Afghanistan. As this is a correct answer to our formulation, we consider this query as successful. There are three more questions, *GP5*, *GP7*, and *GP10*, which we formulated as straightforward SPOTLX queries and could use to obtain correct results.

In other cases, we needed some creativity in formulating the query, including the use of keywords in the conteXt part of a SPOTLX query. An example is the question *GP1*:

Name the waterfalls that have been employed in any of the several adaptations of Fenimore Cooper's book "The Last of the Mohicans" to cinema.

```
?x isA waterfall matches (+last +mohicans)
```

YAGO2 does not contain any relations about scene locations in movies, but reformulating the constraint on the movie as a keyword condition yields two correct results. This demonstrates the usefulness of the context part of SPOTLX when structured data is not available. Other questions that returned correct results using keywords in their query are *GP4* and *GP8*. Note that although we manually identified the keywords for the conteXt part of these queries, this relaxation could easily be automated by generating keyword counterparts to structured conditions, one at a time, whenever a more structured formulation does not return the desired results. Further note that this approach is quite different from conventional methods for natural-language QA where the entire question is mapped into a keywords-only query for a search engine or on specific text corpus. Our approach still harnesses the rich type system for entities and aims to preserve as many structural conditions as possible. A pure keywords query like "waterfalls last mohicans cinema fenimore cooper" would usually not work (depending on the underlying corpus).

There are also questions which we can formulate perfectly, but we do not have any correct entity or fact in the knowledge base; an example is question *GP15*:

Find articles about bridges in France whose construction started, continued or ended in or between 1980 and 1990.

```
?x isA bridge nearby Bourges,400km .
?x wasCreatedOnDate ?d during 1980,1990
```

We used `Bourges` as a "geographical center" of France. Unfortunately, all the bridges that have coordinates associated were not constructed in that time interval. *GP6*, *GP9*, *GP11*, *GP12*, and *GP14* are further questions of this kind. Obviously, no knowledge base can ever be complete, and these advanced questions happened to hit some blank spots in YAGO2.

Out of the 15 questions, there were only two that we could not formulate at all: *GP2* "Find documents about people who have belonged to or are considered affiliated with the Vienna circle but who are not Austrian or German". This question contains a negation predicate, which is not supported in our query language. The other one is *GP3*: "Relevant documents describe rivers in Portugal that have cities with a population higher than 150,000 people along their banks", where we lack the relation of rivers flowing through cities, something that cannot be captured appropriately using keywords.

Altogether we observed:

- 4 questions working perfectly;
- 3 questions working when relaxing a geographical condition from structural to keyword conditions – resulting in a less precise but still useful result set;
- 6 questions that could be well formulated as SPOTLX queries but did not return any good result for the limited coverage of the knowledge base;
- 2 questions that could not be properly formulated at all.

Note that this result, albeit far from perfect, could not have been achieved with the original YAGO. The geographic and temporal knowledge of YAGO2, the keyword context of entities, and the SPOTLX capabilities were crucial for successful query formulation.

The original competition at CLEF 2008 from which we adopted our queries, had three participating systems [24, 25]. One of them was a semi-automatic system and critically relied on human guidance in a multi-stage procedure. Among the other two systems, one performed poorly and could find answers to only 4 of the 15 queries, and had generally low precision over all retrieved answers (about 10 percent). The third system, WikipediaQAList, achieved very good results. It found good answers to 14 of the 15 queries, and had an overall precision of about 63 percent. In comparison to our YAGO2-based results, this is much better. However, one has to consider that WikipediaQAList is highly tailored to the task at hand: list questions, formulated in a particular style, and evaluated by a carefully designed procedure over Wikipedia categories with link-based filters. YAGO2, in contrast, is a general purpose ontology. It was not designed in any way with the CLEF competition in mind. Our study merely demonstrates the off-the-shelf usefulness of YAGO2 in combination with SPOTL(X) querying and is meant to provide background information. An apples-vs.-oranges comparison between our case study and the actual participants in the CLEF 2008 task is meaningless, as they have very different goals and assumptions.

Type	Category	Correct	Nearly Correct	NA
Temporal	Name the Decade	3 (2)	0 (0)	2 (3)
	Died on the same day	4 (1)	1 (1)	0 (3)
	The 19th Century	2 (0)	2 (1)	1 (4)
Spatial	Canadian Geography	4 (1)	1 (1)	0 (3)
	Urban	1 (0)	4 (0)	0 (5)
	American Towns & Cities	3 (1)	2 (0)	0 (4)
Total		17 (5)	10 (3)	3 (22)
Percent		57% (17%)	33% (10%)	10% (73%)

Table 10: Jeopardy questions answered with SPOTLX queries (numbers in parentheses are answered with the original YAGO data and SPO-only queries)

9.1.2. Jeopardy

The Jeopardy quiz show recently obtained attention in the computer science field, because IBM's Watson system [26] participated in one of the shows and won against two human champions. Jeopardy questions are grouped in categories, each comprising 5 questions. We chose three such blocks with temporal questions, and three blocks with spatial questions, a total of 30 questions. All questions could be formulated using SPOTLX queries. An overview of the question categories and how many questions could be correctly answered by YAGO2 is given in Table 10.

In the case of Jeopardy we counted a question as *correctly answered* if the SPOTLX query gave exactly the correct answer and no other results. An example is this question:

In June 1876 George Custer made his last stand at the Battle of this river.

```
?x isa battle overlaps 1876-06 matches (+George +Custer) .
?x happendIn ?r .
?r isa river
```

It returns the correct result `Battle of the Little Bighorn`.

Nearly correct cases are questions for which we obtained the correct result among other results, or where we could not formulate the geographical or temporal condition with structured conditions only but needed to include keyword conteXt conditions. An example is:

Montana State University has a branch in this city named for frontiersman John.

```
?k means ?x locatedIn Montana matches (+Montana +State
+University) .
?s hasFamilyName ?k .
?s hasGivenName John
```

We could not formulate that the university is located in the city, but added this condition as a keyword associated with a permanent relation of the city in question. Also, the correct result `John Bozeman` was not the only result returned.

We classified questions as *not expressible (NA)* if we could not formulate them appropriately as SPOTLX queries, using the available relations and predicates, and thus could not obtain any correct result.

In some cases, we found that we were missing specific relations necessary to answering the question, e. g. `hasLength`, which we then added. Other useful relations we identified were `namedAfter` and `flowsThrough`, but the semi-structured data in Wikipedia is too sparse to extract enough facts for these.

As with the GeoCLEF GiKiP questions, the spatio-temporal and contextual extensions of YAGO2 are crucial assets to answer most of the Jeopardy questions. Table 10 shows the improvement over the original YAGO knowledge base: 73% of the questions could not be expressed with the original YAGO, either due to lack of querying capabilities or due to lack of data. With YAGO2, only 10% of the questions were left unanswered.

9.2. Improving Named Entity Disambiguation by Spatio-Temporal Knowledge

The task of mapping mentions of named entities, such as persons, locations or organizations, in natural language text onto canonical entities registered in a knowledge base is called named entity disambiguation. It is a necessary step when extracting facts from natural language text (see, e.g., [27]), but also useful in itself to annotate Web pages, news articles, or any other text with embedded entities.

Named entity disambiguation for Wikipedia entities dates back to Bunescu and Pasca [28], with substantial improvements by Cucerzan [29], Milne and Witten [30], and Kulkarni et al [31]. The most basic measure for disambiguation is the prior probability of a mention pointing to a certain entity, which can be harvested from the Wikipedia link structure. When a mention is then encountered in a text, for example, "Joey", we can exploit the knowledge (from Wikipedia) that it links to `Joey (TV Series)` in 49% of all cases, and with a 3% probability to `Joey (Bob Dylan song)`. This approach always chooses the most prominent entity for a given mention string. Key to improving this prior is to consider the context of a mention. For example, consider the sentence "Dylan performed Hurricane about the black fighter Carter, from his album Desire. That album also contains a duet with Harris in the song Joey." Here, the tokens "song", "album", and "performed" are strong cues for `Joey (Bob Dylan song)` instead of the TV series.

To score entities based on their overlap with the context of a mention, each entity is associated with keyphrases gathered in the YAGO2 `hasContext` relation: link anchor texts, category names, and titles of works in the reference section. These keyphrases are matched against the context surrounding a mention in the text. In the above example, the keyphrase "Bob Dylan songs" is associated with `Joey (Bob Dylan song)`. It matches multiple parts of the context, e. g. "Dylan" and "song". The score for each keyphrase q is calculated as follows:

$$score(q) = z \left(\frac{\sum_{w \in cover} weight(w)}{\sum_{w \in q} weight(w)} \right)^2$$

where $z = \frac{\#\ matching\ words}{length\ of\ cover(q)}$, and the cover is the shortest span of tokens in the context where all matching keywords occur. The weight of a keyword w is a combination of *mutual information* between the keyword and the entity it is associated to, as well as the standard *idf* weight. The score of an entity is then calculated by summing up the scores of all its keyphrases. More details about our keyphrase-based context similarity method are given in [32, 33].

Further improvements can be obtained by mapping mentions *jointly* rather than one at a time. In the above example, if "Joey" is mapped to the song, "Carter" should be mapped to `Rubin Carter`, the boxer also known as "Hurricane", and not `Jimmy Carter`, the president, overruling the most prominent meanings for both mentions. Cucerzan [29] was the first to introduce this notion of joint disambiguation of all entity mentions in a text. Kulkarni et al. [31] cast the joint mapping approach into a factor-graph probabilistic model, approximated by linear programming. Our recent approach AIDA [32] casts the joint mapping into a graph-theoretic problem, solving it with a greedy algorithm.

All of the previous approaches use the Wikipedia link structure as a measure of *coherence* among entities. The measure introduced by [30] and also used by [31] rates the relatedness of two entities based on the overlap of the set of incoming links in Wikipedia, and is defined as follows:

$$inlink\_coh(e_1, e_2) = 1 - \frac{\log(\max(|IN_{e_1}|, |IN_{e_2}|)) - \log(|IN_{e_1} \cap IN_{e_2}|)}{\log(|N|) - \log(\min(|IN_{e_1}|, |IN_{e_2}|))}$$

if > 0 and else set to 0. N is the total number of entities in the knowledge base. The more similar the set of entities linking to two given entities, the higher the relatedness between these two. Other possibilities for measuring coherence among entities include the distance of two entities in the type hierarchy. In the task presented here, we make use of the temporal and spatial knowledge available in YAGO2, to measure coherence in the following ways:

Spatial Coherence is defined between two entities $e_1, e_2 \in E$ with geo-coordinates, where E is the set of all candidates for mapping mentions in a text to canonical entities:

$$geo\_coh(e_1, e_2) = \frac{\text{great\_circle\_distance}(coord(e_1), coord(e_2))}{\text{length\_of\_equator}/2}$$

This postulates that two entities that are geographically close to each other are a coherent pair, based on the intuition that texts or text passages (news, blog postings, etc.) usually talk about a single geographic region.

Temporal Coherence is defined between two entities $e_1, e_2 \in E$ with existence time:

$$temp\_coh(e_1, e_2) = \frac{|cet(e_1) - cet(e_2)|}{max_{e_i,e_j \in E}(|cet(e_i) - cet(e_j)|)}$$

where $cet(\,)$ is the center of an entity's existence time interval, and the denominator normalizes the distance by the maximum distance of any two entities in the current set of entity candidates, $e_i, e_j \in E$. The intuition here is that a text usually mentions entities that are clustered around a single or a few points in time (e.g., the date of an event in which several people participated, thus posing the requirement that these entities must have overlapping life spans).

Using these coherence measures, the input graph for the AIDA graph algorithm is constructed by two kinds of nodes. One type of nodes represents the entity *mentions* occurring in the input text, the other nodes are the candidate *entities* in the knowledge base. The edges between the mentions and their respective entity candidates are weighted with a combination of the prior and the similarity between the mention context and the entity context. The edges between the entities are weighted either by the *spatial coherence* or the *temporal coherence* measure, which indicates how strongly two entities are related. The objective of the algorithm is to find a dense (highly weighted) subgraph in this input graph that has only one mention-entity edge per mention, solving the disambiguation problem. This problem is computationally hard. We approximate the solution using an efficient greedy algorithm, which aims to maximize the minimum weighted degree of entity nodes in two phases: 1) iteratively remove the node with the lowest weighted degree, and 2) run a local-search optimization algorithm on the (usually much smaller) graph with the highest minimum weighted degree found in any of the iterations.

9.2.1. Experiments

We experimentally evaluated the usefulness of spatial and temporal coherence for named entity disambiguation on two datasets that we created from Wikipedia samples: *WikipediaLocation* and *WikipediaEvent*. Each of these is a set of 50 randomly selected Wikipedia articles that contain a hyperlink with an anchor text matching one the following terms:

WikipediaLocation: San Jose, Victoria, Springfield, Columbia, Georgia

WikipediaEvent: battle, attack, revolution, election, invasion

The terms were selected so as to construct articles with high ambiguity among their entity mentions. There are many cities or states that share the same name, and the common nouns for events may point to a wide variety of concrete events, depending on the context that they are used in. A restriction for an article to be selected was that the entity the hyperlink points to

	WikipediaLocation	WikipediaEvent
Number of Documents	50	50
Avg. number of words per article	2571	4303
Number of evaluated mentions	64	52
Avg. number of entities per mention	180.4	126.4

Table 11: Datasets for entity disambiguation task

	prior	sim	sim + coherence
Accuracy on *WikipediaLocation*	40.00%	41.00%	61.00%
Accuracy on *WikipediaEvent*	13.33%	20.00%	53.33%

Table 12: Results of the Named Entity Disambiguation

must be a `yagoGeoEntity` with coordinates for the *WikipediaLocation* or an *event* (`wordnet_event_100029378`) with an occurrence time for the *Wikipedia-Event*. Otherwise the aspects of spatial or temporal coherence would not apply. More details about the datasets are given in Table 11. We evaluated only the mention-entity mappings for the anchor texts matching our *WikipediaLocation* or *WikipediaEvent* term lists. Thus the focus was on the most difficult mentions.

We ran different configurations of our disambiguation framework on the datasets, shown in Table 12. The accuracy is the fraction of the given mentions correctly disambiguated onto the entity that the hyperlink really points to, averaged over all 50 documents. The baseline is the *prior*, which chooses the entity that links with this anchor text point to most often, e. g. for "Victoria" it chooses `Victoria, Australia`, because 71% of all links with "Victoria" as anchor text point to the Australian state. The accuracy that *prior* achieves is low, especially for *WikipediaEvent*, where only 13% of the mappings are correct. In general, however, the prior has been shown to be a fairly good baseline [32], especially for Wikipedia link prediction [30]. The poor results here reflect the difficulty of our choice of mentions in this evaluation task. The more powerful method *sim* combines the prior with a keyphrase based similarity measure, slightly improving the results for *WikipediaEvent* and *WikipediaLocation*, but not significantly. Including spatial coherence on the *WikipediaLocation* dataset and temporal coherence on the *WikipediaEvent* dataset improved the accuracy by 20 and 33 percent points, respectively. This is a significant improvement over both the prior and the keyphrase based similarity measure, with a p-value of a paired t-test < 0.01.

For comparison, we also ran the experiments with the general-purpose coherence measure used in [32], based on Wikipedia in-link overlap between entities. This approach achieved even better results with an accuracy of 83.33% on *WikipediaEvent* and 79% on *WikipediaLocation*. This is not surprising, as the Wikipedia link structure is very rich, and provides a strong asset for fine-grained coherence measures. However, once we address situations where not all entity candidates in a knowledge base are covered and richly featured in Wikipedia, the link structure is not available for coherence measures. This is the case for all entities in YAGO2 that come from GeoNames and are not in Wikipedia. GeoNames provides coordinates for every entity, though; so using the spatial coherence is feasible and allows us to significantly improve the accuracy of named entity disambiguation. The same argument holds for adding events to the knowledge base: acquiring the existence time of an event from a news page or event calendars on sports or concerts is not that difficult and may become a standard case in maintaining knowledge bases. In contrast, Wikipedia is manually maintained and curated; so adding such facts and appropriate hyperlinks would require much higher effort. Moreover, it is unlikely that Wikipedia will ever cover the "long tail" of named entities that appear in news, blogs, online communities, and other Web pages (e.g., songs, artists, concerts, small-town landmarks, etc.).

10. Related Work

10.1. Taxonomy Construction

YAGO2 constructs a taxonomy from Wikipedia and WordNet. Our method involves two stages: First, it links Wikipedia entities by a *type* (instanceOf) relationship to suitable Wikipedia leaf category classes. Second, it links these Wikipedia category classes by a *subclassOf* relationship to suitable WordNet classes [2]. For this purpose, the system employs the algorithm that was presented in the original YAGO paper from 2007 [9], described in Section 2. Quite a number of works have addressed similar tasks.

Wikipedia Taxomonies. Some projects have constructed a taxonomy from the Wikipedia category system alone [34, 8, 35, 7, 36]. WikiTaxonomy [34] (with improvements in [8]) introduced the idea of arranging the categories of Wikipedia into a hypernymy hierarchy. The approach restricts itself to the Wikipedia categories only. Therefore, WikiTaxonomy's goal is different from YAGO's, which aims to establish consistent links between Wikipedia categories and WordNet as WordNet is the most widely used computational lexicon of English in Natural Language [37]. Furthermore, WikiTaxonomy does not distinguish between classes and instances, which is crucial in YAGO and adds substantial value. Finally, the noisy and inconsistent nature of Wikipedia's non-leaf categories leads WikiTaxonomy to construct a many-rooted taxonomy (with several thousand unrelated roots), as opposed to the consistent and fully connected semantic graph of YAGO.

A number of works have followed up on the WikiTaxonomy project. Zirn et al. [35] take the WikiTaxonomy as input and decide whether a leaf node is an instance or a class. In YAGO, we select only those Wikipedia categories that are classes and take the instances from the Wikipedia articles instead. Among other techniques, the approach of [35] uses the head word plural detection previously described in the YAGO paper [9].

Similarly to WikiTaxonomy, the WikiNet [36] project extracts a concept tree from Wikipedia categories, as opposed to YAGO's goal of interlinking Wikipedia and WordNet. Unlike YAGO, WikiNet does not distinguish between classes and instances. WikiNet also extracts a rich set of relationships between entities.

Mapping WordNet and Wikipedia. Other projects are concerned with mapping Wikipedia categories to WordNet senses [7, 37, 38].

Ponzetto et al. [7] map the Wikipedia categories of WikiTaxonomy to WordNet concepts. Among other techniques, their approach employs head word plural detection (as in the original YAGO). The authors find that the most frequent names heuristic has a precision of 75%, while their techniques improve precision to 80%. The paper does not provide any discussion of how this relates to YAGO's [9] precision of 97%. We believe that the striking difference in performance is due to the fact that the method of [7] aims to map *all* categories of Wikipedia to WordNet, while YAGO is concerned only with the leaf categories. YAGO limits itself to mapping leaf categories of Wikipedia, because it aims to link with, not replace, WordNet. Non-leaf categories in Wikipedia carry substantial noise and inconsistencies.

In a similar spirit, Toral et al. [38] map Wikipedia categories to WordNet nouns. They report a precision of 77%, without any comparison to YAGO. Again, we conjecture that the difference is due to the method's attempt to map *all* Wikipedia categories, instead of just the leaf categories. Furthermore, the paper provides a fully automated approach, while YAGO employs a small number of manually defined mapping rules (to enhance the most-frequent sense information provided by WordNet). The strategy in YAGO was to invest a small amount of manual effort in order to achieve very high precision.

WordNet++ [37] binds Wikipedia pages about common nouns (such as *soda drink*) to the corresponding WordNet concepts. YAGO ignores Wikipedia pages about common nouns. In contrast, YAGO contains the full set of individual entities from Wikipedia and their leaf categories. Thus, WordNet++ and YAGO pursue complementary goals.

BabelNet [39] also maps Wikipedia articles to WordNet, but enhances them with multilingual concepts. YAGO maps only Wikipedia *categories* to WordNet. BabelNet does not contain facts about entities (other than lexical, taxonomic, and unspecified unlabeled relations). UWN and MENTA [14, 40] have added a multilingual dimension to entity and concept names, and also the class system. All these recent projects have been carried out in parallel to the construction of YAGO2 and are complementary to YAGO2 in their structure and contents. For the multilingual dimension, we have integrated UWN into YAGO2.

Taxonomies from the Web. Cyc [1] has attempted to populate its semantic classes by instances gathered from the Web [41]. However, that work reported only very small coverage; the commercial products of CyCorp Inc. may have higher coverage, but there are no details published. Freebase (`freebase.com`) and Trueknowledge (`trueknowledge.com`) are more recent endeavors to build large-scale knowledge bases, tapping into Wikipedia as well as other sources. Both of them are also of commercial nature.

Newer work [42] has addressed the issue of taxonomy generation from the Web on a larger scale. This work differs in its goal from YAGO, which aimed very specifically at connecting Wikipedia instances to WordNet classes.

10.2. Ontologies

Ontologies have been either hand-crafted or constructed in an (semi-) automated manner; see [43] for an overview. Prominent examples of hand-crafted knowledge resources are Cyc [1], WordNet [2], and SUMO [44], and also more recent ontologies such as GeoWordNet [13]. While these hand-crafted approaches have near-perfect precision, they cannot achieve the large-scale coverage of automatically constructed ontologies.

Most automated approaches have drawn from semistructured elements in Wikipedia and other Web sources: infoboxes, category names, tables, lists, etc. [45, 46, 47] and the references given there provide an overview of recent work along these lines. Commercial endeavors include Freebase, Trueknowledge, and Wolframalpha. None of these approaches has addressed the specific dimensions of temporal and geospatial knowledge.

There is a variety of academic projects for constructing large knowledge collections, using information extraction techniques on Web sources. These include KnowItAll and its successor TextRunner [4, 5], DBpedia [3], the Omnivore system [48], work on distilling Web tables and lists into facts [49, 50, 51], the ReadTheWeb project [52] the StatSnowball methods used for building EntityCube [53] and its follow-up project Probase [54], WikiNet [36], our own work on SOFIE [27] and Prospera [55], and others. None of these approaches has specifically considered the temporal and geographical dimension. Moreover, most of them produce outputs in non-canonical form, with surface names and textual patterns rather than canonicalized entities and typed relations. DBpedia [3] will be discussed in detail below.

The Kylin/KOG project [56] has developed learning-based methods for automatically typing Wikipedia entities and generating infobox-style facts. However, this project has not (yet) led to a publicly available knowledge base. Omega [6] integrated WordNet with separate upper-level ontologies and populated various classes with instance collections, including locations from geo gazetteers. Sweto [57] is a tool suite for building knowledge bases in a semi-automatic manner. Predating the advent of Wikipedia harvesting, the sizes of the Omega and Sweto resources are much smaller than that of YAGO2.

10.3. DBpedia

Closest to YAGO in spirit is the DBpedia project [3, 11], which also extracts an ontological knowledge base from Wikipedia. DBpedia and YAGO have different class systems. While YAGO re-uses WordNet and enriches it with the leaf categories from Wikipedia, the DBpedia project has manually developed its own taxonomy. YAGO's compatibility with WordNet allows easy linkage and integration with other resources such as Universal WordNet [14], which we have exploited for YAGO2. DBpedia's taxonomy has merely 272 classes, while YAGO2 contains about 350,000.

For extracting relational facts from infoboxes, YAGO2 uses carefully handcrafted patterns, and reconciliates duplicate infobox attributes (such as *birthdate* and *dateofbirth*), mapping them to the same canonical relation. DBpedia outsourced the task of pattern definition to its community and uses a much larger number of more diverse extraction patterns, but ends up with redundancies and even inconsistencies. Overall, DBpedia contains about 1100 relations, versus YAGO2 having about 100. The following key differences explain this big quantitative gap, and put the comparison in the perspective of data quality.

- Many relations in DBpedia are very special. As an example, take *aircraftHelicopterAttack*, which links a military unit to a means of transportation. Half of DBpedia's relations have less than 500 facts.

- YAGO2's relations have more coarse-grained type signatures than DBpedia's. For example, DBpedia knows the relations *Writer, Composer*, and *Singer*, while YAGO2 expresses all of them by *hasCreated.* On the other hand, it is easy for YAGO2 to infer the exact relationship (*Writer* vs. *Composer*) from the types of the entities (*Book* vs. *Song*). So the same information is present.

- YAGO2 represents years as incomplete dates, so that there is a single unified way of expressing a birth date, no matter whether this date is given as a calendar date or as the year only. DBpedia has different relations for complete dates and for years. This yields a number of relations that are semantic duplicates, but are not synchronized with each other. Not every entity with a birth date also has a birth year.

- YAGO2 does not contain inverse relationships. A relationship between two entities is stored only once, in one direction. DBpedia, in contrast, has several relations that are the inverses of other relations (e.g., *hasChild*/*hasParent*). This increases the number of relation names without adding information.

- YAGO2 has a sophisticated time and space model, which represents time and space as facts about facts. DBpedia closely follows the infobox attributes in Wikipedia. This leads to relations such as *populationAsOf*, which contain the validity year for another fact. A similar observation holds for geospatial facts, with relations such as *distanceToCardiff.*

Overall, DBpedia and YAGO share the same goal and use many similar ideas. At the same time, both projects have also developed complementary techniques and foci. Therefore, the two projects generally inspire, enrich, and help each other. For example, while DBpedia uses YAGO's taxonomy (for its `yago:type` triples), YAGO relies on DBpedia as an entry point to the Web of Linked Data [58].

10.4. Geographical and Temporal Knowledge

The first geographical gazetteers have been created centuries ago to collect information associated with geographical locations. Today, the most comprehensive collection of this kind is GeoNames (`geonames.org`), providing geo-coordinates for about 7 millions of entities and a geo-specific type system with several hundred classes. GeoNames is curated by integrating a suite of structured data collections. By itself it is not connected to any other universal knowledge base. To our knowledge, YAGO2 is the first collection that has fully integrated all GeoNames entities along with proper mappings into the rich class system of WordNet.

More recently, the idea of gazetteers has been expanded, e. g., by Feinberg et al. [12], to encompass named periods, such as "The French Revolution" or "Renaissance", with their corresponding time periods. Such a temporal directory was created using Library of Congress subject headings by Petras et al. [59]. YAGO2 takes this idea further, by combining the temporal and geographical data with semantic information. It knows not only the periods of named events or lifetimes of persons, but also semantic relationships for connecting all these entities.

10.5. Temporal Fact Extraction

Several approaches have targeted the extraction of temporal facts from text sources. The most prominent work along these lines is TARSQI [60]. TARSQI captures not only explicit dates, but also phrases such as "a week ago" or "last year", mapping them into explicit dates. The NLP community has had event extraction tasks in its TempEval workshop series [61], using representations such as TimeML and reference corpora such as Timebank [62]. More recent work in this area is from Strötgen and Gertz [63]. There is no attempt, though, to connect these dates to a large knowledge base of entity-relationship facts.

Temporal knowledge as a first–class citizen in richly populated knowledge bases has been addressed by only few prior papers: the TOB framework of [64], our own preliminary attempt towards T-YAGO [65], and the TIE approach of [66]. TOB [64] 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 [65] 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. [67] focused on logics-based querying over uncertain t-facts, but did not address the extraction and fact harvesting process. Finally, the TIE approach [66] used training data with fine-grained annotations to learn an inference model based on Markov Logic. However, it did not aim to create a full knowledge base with time and space.

There is also recent awareness of temporal IR: ranking of search results for keyword queries with temporal phrases [68, 69, 70]. This work is orthogonal to ours.

10.6. Knowledge Representation for Time

The general theme of temporal knowledge is an old AI topic [71]. The standard textbook by Russel and Norvig [72] refers to temporal facts as *fluents*: instances of relations whose validity is a function of time.

There are different approaches to translate this notion of fluents into the Semantic Web world. The earliest approach of the W3C has been to favor *event entities*. For example, the birth of a person can be represented by an event entity such as *birth42*; Entities that participate in this event – person, date, location, etc. – are then linked by relations to this event entity. The drawback of this approach is that one has to decide a priori which relations are represented as binary relations with standard RDF triples, and which relations should be cast into event entities with additional annotations.

A second approach to time representation is reification. Reification creates a fact identifier for every fact. Then, it links the fact identifier to the subject, predicate and object of the original fact, using additional RDF triples. This leads to a substantial blow-up of storage space, and makes browsing and querying inconvenient. In contrast, the specific form of reification in YAGO2 has fact identifiers built in. Most facts are represented as standard triples, fact identifiers are only used when really needed. Moreover, our SPOTL view on this data makes exploring and querying easy.

A third approach for the representation of meta-information are named graphs [73], which will soon become a W3C standard. A named graph is a set of RDF statements. Named graphs allow focusing on specific knowledge bases, making statements about certain knowledge bases, and annotating entire knowledge bases by trust. To represent time and space annotations by named graphs, every RDF statement would have to form its own named graph. This seems like a huge overkill, and is not in the spirit of the original intention behind named graphs.

A fourth approach [74, 75] extends RDF triples into quadruples, *quads* for short. The fourth component primarily serves to represent the provenance of a triple, but could also be used for other kinds of meta-facts like validity time of fluents. The SPOTL model in YAGO2 is compatible with the quads approach, and extends to more than 4 dimensions so as to simultaneously capture time, space, provenance, and context.

Other RDF extensions for temporal and/or spatial knowledge include work by [76, 77, 78, 79]. Gutierrez et al. [76] 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. Pugliese et al. [78] propose an a time index supporting queries on this kind of enhanced RDF data. Koubarakis and Kyzirakos [79] combine 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. [77] proposed an ontological model for a time- and space-aware ontology, together with a set of temporal and spatial query operators. Our contribution, relative to these related works, lies in a unified simple representation of temporal knowledge and systematically propagating the available data to all relevant facts. For an overview on the field of spatio-temporal databases, refer to [80].

11. Conclusions

11.1. Choice of Sources and Generalization Beyond

YAGO2 is built on Wikipedia, WordNet, and GeoNames. We chose these sources because of four specific reasons.
1) Coverage and quality: Wikipedia comprises millions of entities of common interest, WordNet aims to cover all words of the English language, and GeoNames is the largest free geo-gazetteer available. All three sources are manually curated, with excellent accuracy of their contents.
2) Extraction accuracy: All three sources exhibit a high degree of structure, so that we can extract data with near-human accuracy.
3) Contents licensing: All three sources have permissive licenses, so that we can integrate their data into YAGO2 and make it publicly available.
4) Standard references: Wikipedia, WordNet and GeoNames are by far the largest and most popular references of their kinds.
These characteristics make Wikipedia, WordNet and GeoNames unique. By bringing these sources together, YAGO2 amplifies their usefulness.

While the YAGO2 extraction system is tailored to the specific sources, much of the methodology is of more general purpose. Our extraction system combines syntactic information (regular-expression patterns) with semantic information (data types). This allows a limited but highly effective form of semantically checking fact candidates at extraction time, which may be applicable independently of the source. We have also devised specific techniques for dealing with the space and time dimensions. Temporal and geospatial information can be propagated from the base-facts to the meta-facts and back. This method works universally regardless of where the base-facts originate. It makes YAGO2 the first large-scale knowledge base that is consistently anchored in time and space. The original YAGO paper [9] introduced the idea of type checking. In the current paper, we have substantially extended this methodology to a framework of Horn rules. Horn rules allow extracted facts to generate new facts. This technique demonstrates that a high extraction quality allows not just plausibility checks, but also fact-generating rules. These can fruitfully interact with the extraction process to further enhance the knowledge base.

11.2. Summary and Outlook

We have developed a 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 120 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 [11]) 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, Freebase, WikiTaxonomy, YAGO, 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 [27]. 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 [65, 66]. 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.

12. Download

YAGO2 is publicly available on our project page `http://www.yago-knowledge.org`. The content is licensed under a Creative Commons License[5]. All data is available in multiple formats including RDF.

13. 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.

References

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

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

[5]`http://creativecommons.org`

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, 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, pp. 722–735.

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

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

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

[7] S. P. Ponzetto, R. Navigli, Large-Scale Taxonomy Mapping for Restructuring and Integrating Wikipedia, in: Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009, Pasadena, California, USA, pp. 2083–2088.

[8] S. P. Ponzetto, M. Strube, Taxonomy induction based on a collaboratively built knowledge repository, Artificial Intelligence 175 (2011) 1737–1756.

[9] F. M. Suchanek, G. Kasneci, G. Weikum, YAGO: A Core of Semantic Knowledge, in: Proceedings of the 16th international conference on World Wide Web, WWW 2007, Banff, Canada, pp. 697–706.

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

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

[12] M. Feinberg, R. Mostern, S. Stone, M. Buckland, Application of Geographical Gazetteer Sdards to Named Time Periods, Technical Report LG-02-02-0035-02, Institute of Museum and Library Services National Leadership, Berkeley, 2003.

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

[14] G. de Melo, 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, Hong Kong, China, pp. 513–522.

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

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

[17] J. Hoffart, F. M. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, G. Weikum, YAGO2: Exploring and Querying World Knowledge in Space, Context, and Many Languages (Demo Paper), in: Proceedings of the 20th international conference companion on World Wide Web, WWW 2011, Hyderabad, India, pp. 229–232.

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

[19] J. M. Hellerstein, J. F. Naughton, A. Pfeffer, Generalized Search Trees for Database Systems, in: Proceedings of 21th International Conference on Very Large Data Bases, VLDB 1995, Zurich, Switzerland, pp. 562–573.

[20] L. D. Brown, T. T. Cai, A. Dasgupta, Interval Estimation for a Binomial Proportion, Statistical Science 16 (2001) 101–133.

[21] J. L. Fleiss, Measuring nominal scale agreement among many raters, Psychological Bulletin 76 (1971) 378–382.

[22] K. L. Gwet, Computing inter-rater reliability and its variance in the presence of high agreement, British Journal of Mathematical and Statistical Psychology 61 (2008) 29–48.

[23] A. Frank, H.-U. Krieger, F. Xu, H. Uszkoreit, B. Crysmann, B. Jörg, U. Schäfer, Question answering from structured knowledge sources, J. Applied Logic 5 (2007) 20–48.

[24] D. Santos, N. Cardoso, GikiP: evaluating geographical answers from wikipedia, in: Proceeding of the 2nd international workshop on Geographic information retrieval, GIR 2008, Napa Valley, California, USA, pp. 59–60.

[25] D. Santos, N. Cardoso, P. Carvalho, I. Dornescu, S. Hartrumpf, J. Leveling, Y. Skalban, Gikip at geoclef 2008: Joining gir and qa forces for querying wikipedia, in: Evaluating Systems for Multilingual and Multimodal Information Access, volume 5706 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2009, pp. 894–905.

[26] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, C. A. Welty, Building Watson: An Overview of the DeepQA Project, AI Magazine 31 (2010) 59–79.

[27] F. M. Suchanek, M. Sozio, G. Weikum, SOFIE: A Self-Organizing Framework for Information Extraction, in: Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, pp. 631–640.

[28] R. Bunescu, M. Pasca, Using Encyclopedic Knowledge for Named Entity Disambiguation, in: Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2006, Trento, Italy, pp. 9–16.

[29] S. Cucerzan, Large-scale named entity disambiguation based on Wikipedia data, in: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2007, Prague, Czech Republic, pp. 708–716.

[30] D. Milne, I. H. Witten, Learning to Link with Wikipedia, in: Proceedings of the 17th ACM Conference on Information and Knowledge Mining, CIKM 2008, Napa Valley, California, USA, pp. 509–518.

[31] S. Kulkarni, A. Singh, G. Ramakrishnan, S. Chakrabarti, Collective Annotation of Wikipedia Entities in Web Text, in: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, KDD 2009, Paris, France, pp. 457–466.

[32] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, G. Weikum, Robust Disambiguation of Named Entities in Text, in: Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, Edinburgh, 2011, pp. 782–792.

[33] B. Taneva, M. Kacimi, G. Weikum, Finding Images of Rare and Ambiguous Entities, Technical Report MPI-I-2011-5-002, Max Planck Institute for Informatics, 2011.

[34] S. P. Ponzetto, M. Strube, Deriving a Large-Scale Taxonomy from Wikipedia, in: Proceedings of the 22nd AAAI Conference on Artificial Intelligence, AAAI 2007, Vancouver, British Columbia, Canada, 2007, pp. 1440–1445.

[35] C. Zirn, V. Nastase, M. Strube, Distinguishing between instances and classes in the wikipedia taxonomy, in: The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, pp. 376–387.

[36] V. Nastase, M. Strube, B. Boerschinger, C. Zirn, A. Elghafari, WikiNet: A Very Large Scale Multi-Lingual Concept Network, in: Proceedings of the 7th International Conference on Language Resources and Evaluation, LREC 2010, La Valetta, Malta.

[37] S. P. Ponzetto, R. Navigli, Knowledge-rich Word Sense Disambiguation rivaling supervised system, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL 2010, Uppsala, Sweden, pp. 1522–1531.

[38] A. Toral, O. Ferrández, E. Agirre, R. Muñoz, A study on linking wikipedia categories to wordnet synsets using text similarity, in: Recent Advances in Natural Language Processing, RANLP 2009, Borovets, Bulgaria, Association for Computational Linguistics, 2009, pp. 449–454.

[39] R. Navigli, S. P. Ponzetto, BabelNet: Building a very large multilingual semantic network, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL 2010, Uppsala, Sweden, pp. 216–225.

[40] G. de Melo, G. Weikum, MENTA: Inducing Multilingual Taxonomies from Wikipedia, in: Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Canada, pp. 1099–1108.

[41] P. Shah, D. Schneider, C. Matuszek, R. C. Kahlert, B. Aldag, D. Baxter, J. Cabral, M. J. Witbrock, J. Curtis, Automated Population of Cyc: Extracting Information about Named-entities from the Web, in: Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2006, Melbourne Beach, Florida, USA, pp. 153–158.

[42] R. Navigli, P. Velardi, S. Faralli, A graph-based algorithm for inducing lexical taxonomies from scratch, in: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011, Barcelona, Spain, pp. 1872–1877.

[43] S. Staab, R. Studer, Handbook on Ontologies, Springer, 2. edition, 2009.

[44] I. Niles, A. Pease, Towards a standard upper ontology, in: Proceedings of the 2nd International Conference on Formal Ontology in Information Systems, FOIS 2001, Ogunquit, Maine, pp. 2–9.

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

[46] A. Doan, L. Gravano, R. Ramakrishnan, S. Vaithyanathan (Eds.), SIGMOD Rec. Special Section on Managing Information Extraction, volume 37(4), 2008.

[47] G. Weikum, M. Theobald, From Information to Knowledge: Harvesting Entities and Relationships from Web Sources, in: Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems of data, Indianapolis, PODS 2010, Indiana, USA, pp. 65–76.

[48] M. J. Cafarella, Extracting and Querying a Comprehensive Web Database, in: 4th Biennial Conference on Innovative Data Systems Research, CIDR 2009, Asilomar, CA, USA.

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

[50] G. Limaye, S. Sarawagi, S. Chakrabarti, Annotating and searching web tables using entities, types and relationships, Proc. VLDB Endow. 3 (2010) 1338–1347.

[51] P. Venetis, A. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, C. Wu, Recovering Semantics of Tables on the Web, in: Proc. VLDB Endow. 4 (2011), pp. 528–538.

[52] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., T. M. Mitchell, Toward an Architecture for Never-Ending Language Learning, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, US, pp. 1306–1313.

[53] J. Zhu, Z. Nie, X. Liu, B. Zhang, J.-R. Wen, StatSnowball: A Statistical Approach to Extracting Entity Relationships, in: Proceedings of the 18th international conference on World Wide Web, WWW 2001, Madrid, Spain, pp. 101–110.

[54] W. Wu, H. Li, H. Wang, K. Zhu, Towards a Probabilistic Taxonomy of Many Concepts, Technical Report MSR-TR-2011-25, Microsoft Research Beijing, 2011.

[55] N. Nakashole, M. Theobald, G. Weikum, Scalable knowledge harvesting with high precision and high recall, in: Proceedings of the fourth ACM international conference on Web search and data mining, WSDM 2011, Hong Kong, China, pp. 227–236.

[56] F. Wu, D. S. Weld, Automatically Refining the Wikipedia Infobox Ontology, in: Proceeding of the 17th international conference on World Wide Web, WWW 2008, Beijing, China, ACM, 2008, pp. 635–644.

[57] B. Aleman-Meza, C. Halaschek, A. Sheth, I. B. Arpinar, G. Sannapareddy, SWETO: Large-Scale Semantic Web Test-bed, in: In 16th International Conference on Software Engineering and Knowledge Engineering, SEKE 2004: Workshop on Ontology in Action, Banff, Canada, pp. 21–24.

[58] C. Bizer, T. Heath, K. Idehen, T. Berners-Lee, Linked data on the web (LDOW2008), in: Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, pp. 1265–1266.

[59] V. Petras, R. R. Larson, 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, pp. 151–160.

[60] M. Verhagen, I. Mani, R. Sauri, R. Knippen, S. B. Jang, J. Littman, A. Rumshisky, J. Phillips, J. Pustejovsky, Automating Temporal Annotation with TARSQI, in: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, ACL 2005, University of Michigan, USA, pp. 81–84.

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

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

[63] J. Strötgen, M. Gertz, HeidelTime: High Quality Rule-Based Extraction and Normalization of Temporal Expressions, in: Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval 2010, Los Angeles, California, pp. 321–324.

[64] Q. Zhang, F. M. Suchanek, L. Yue, G. Weikum, TOB: Timely Ontologies for Business Relations, in: 11th International Workshop on Web and Databases 2008, WebDB 2008, Vancouver, Canada.

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

[66] X. Ling, D. S. Weld, Temporal Information Extraction, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, pp. 1385–1390.

[67] Y. Wang, M. Yahya, M. Theobald, Time-aware Reasoning in Uncertain Knowledge Bases, in: Proceedings of the Fourth International VLDB workshop on Management of Uncertain Data (MUD 2010) in conjunction with VLDB 2010, Singapore, pp. 51–65.

[68] O. Alonso, M. Gertz, R. Baeza-Yates, On the Value of Temporal Information in Information Retrieval, SIGIR Forum 41 (2007) 35–41.

[69] K. Berberich, S. J. Bedathur, O. Alonso, G. Weikum, A Language Modeling Approach for Temporal Information Needs, in: Proceedings of the 32nd European Conference on Information Retrieval, ECIR 2010, Milton Keynes, UK, pp. 13–25.

[70] M. Pasca, Towards Temporal Web Search, in: Proceedings of the 2008 ACM Symposium on Applied Computing, SAC 2008, Fortaleza, Ceara, Brazil, pp. 1117–1121.

[71] M. Fisher, D. Gabbay, L. Vila, Handbook of Temporal Reasoning in Artificial Intelligence, Elsevier Science Inc., 2005.

[72] S. J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Pearson Education, 3rd edition, 2010.

[73] J. J. Carroll, C. Bizer, P. J. Hayes, P. Stickler, Named graphs, provenance and trust, in: Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, pp. 613–622.

[74] C. A. Welty, R. Fikes, A reusable ontology for fluents in owl, in: B. Bennett, C. Fellbaum (Eds.), Formal Ontology in Information Systems, Proceedings of the Fourth International Conference, FOIS 2006, Baltimore, Maryland, USA, Frontiers in Artificial Intelligence and Applications, IOS Press, 2006, pp. 226–236.

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

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

[77] M. Perry, A. P. Sheth, F. Hakimpour, P. Jain, Supporting Complex Thematic, Spatial and Temporal Queries over Semantic Web Data, in: Proceedings of the 2nd international conference on GeoSpatial semantics, GeoS 2007, Mexico City, Mexico, pp. 228–246.

[78] A. Pugliese, O. Udrea, V. S. Subrahmanian, Scaling RDF with Time, in: Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, pp. 605–614.

[79] M. Koubarakis, 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, 2010, pp. 425–439.

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

Appendix A. Question Answering with YAGO2

All questions of the task-based evaluation, Section 9.1, and their answers from YAGO2 are listed here.

GP1: Name the waterfalls that have been employed in any of the several adaptations of Fenimore Cooper's book "The Last of the Mohicans" to cinema.

```
?x isA waterfall matches (+last +mohicans)
```

Result: Nearly correct. Two waterfalls are found; according to the last-mohicans-movie page there are more than five.

GP2: Find documents about people who have belonged to or are considered affiliated with the Vienna circle but who are not Austrian or German

```
NA
```

Result: Could not be formulated, as there is no support for 'not'.

GP3: Relevant documents describe rivers in Portugal that have cities with a population higher than 150,000 people along their banks.

```
NA
```

Result: Could not be formulated, YAGO2 does not contain any relation to express 'flows through'

GP4: Find which cantons are on the border of Switzerland with Germany.

```
?x isa state matches (+canton +Germany) .
?x isLocatedIn Switzerland
```

Result: Nearly correct. Two cantons are correctly identified, some are missing, some additional results are wrong (but these could easily dismissed by other criteria).

GP5: Wars that took place in (ancient or modern) Greece are relevant.

```
?x isa war .
?x happenedIn Greece
```

Result: Correct.

GP6: Relevant documents are about mountains, ranges or peaks in Australia whose altitude is greater than two thousand meters.

```
?x isa mountain .
?x locatedIn Australia .
?x hasHeight ?h .
?h isGreaterThan 2000
```

Result: Could be formulated, but did not return any results, as YAGO2 does not contain any mountains in Australia.

GP7: Relevant documents describe capital cities in the African continent with a population greater or equal to 2,000,000 people.

```
?x isa capital locatedIn Africa .
?x hasPopulation ?p .
?p isGreaterThan 2,000,000
```

Result: Correct.

GP8: Find Brazilian suspension bridges.

```
?x isa suspension bridge matches (+Brazil)
```

Result: Nearly correct. Returns one correct and one incorrect result.

GP9: Which Renaissance composers have German origin?

```
?x isA composer during [1400,1700] .
?x wasBornIn ?b .
?b isLocatedIn Germany
```

Result: Could be formulated, but did not return any result, for lack of facts. The earliest composer in YAGO2 born in Germany is from 1709.

GP10: Find islands in Polynesia whose population is higher than five thousand inhabitants.

```
?x isA island nearby "Cook Islands",4000 .
?x hasPopulation ?p .
?p isGreaterThan 5000
```

Result: Correct (when taking Cook Islands as center).

GP11: Find the plays of William Shakespeare which occur wholly or partially in Italy.

```
WilliamShakespeare created ?p matches (+Italy)
```

Result: Could be formulated, but did not return any result. None of the original Shakespeare works seem to have an infobox (e. g. for Hamlet, Othello, Romeo & Juliet), so YAGO2 does not have the `created` facts.

GP12: Relevant results are cities or other places where Johann Wolfgang von Goethe lived or stayed for some time.

```
JohannWolfgangvonGoethe livesIn ?p
```

Result: Could be formulated, but did not return any result for lack of facts. Wikipedia does not contain this information in semi-structured form (infobox, category, list), so YAGO2 could not extract any facts.

GP13: Relevant documents describe navigable Afghan rivers whose length is greater than one thousand kilometers.

```
?x isA river locatedIn Afghanistan .
?x hasLength ?l .
?l isGreaterThan 1000km
```

Result: Correct.

GP14: Relevant documents describe the life and works of architects from Brazil who have created works in Europe.

```
?x isA architect .
?x created ?a locatedIn Europe .
?x wasBornIn ?p locatedIn Brazil
```

Result: Could be formulated, but did not return any result for lack of facts.

GP15: Find articles about bridges in France whose construction started, continued or ended in or between 1980 and 1990.

```
?x isA bridge nearby Bourges,400km .
?x wasCreatedOnDate ?d during [1980,1990]
```

Result: Could be formulated, but did not return any results for lack of facts. YAGO2 does not contain any bridges in France with geo-coordinates.

Appendix A.2. Jeopardy Questions/Queries and Results

Appendix A.2.1. Name the Decade

Q: Disneyland opens & the peace symbol is created
A: 1950s

`PeaceSymbol wasCreatedOnDate ?x . Disneyland wasCreatedOnDate ?y`

Result: Correct.

Q: The Empire State Building opens & the "War of the Worlds" radio broadcast causes a panic
A: 1930s

`EmpireStateBuilding wasCreatedOnDate ?x`

Result: Correct.

Q: Klaus Barbie is sentenced to life in prison & DNA is first used to convict a criminal
A: 1980s

`NA`

Result: Not expressible.

Q: The first flight takes place at Kitty Hawk & baseball's first World Series is played
A: 1900s

`BaseballWorldSeries wasCreatedOnDate ?x`

Result: Correct.

Q: The first modern crossword puzzle is published & Oreo cookies are introduced
A: 1910s

`Oreo wasCreatedOnDate ?x`

Result: Could not be answered, as `Oreo` is not in YAGO2.

Appendix A.2.2. Died on the Same Day

Q: C.S. Lewis & Aldous Huxley's deaths on Nov. 22, 1963 were overshadowed by this man's death in Dallas
A: John Kennedy

```
C.S.Lewis diedOnDate ?d . ?p diedOnDate ?d . ?p diedIn Dallas
```

Result: Correct.

Q: Just hours before Michael Jackson's death, Hollywood lost this TV "Angel"
A: Farrah Fawcett

```
?p diedOnDate ?d matches (+angel) . MichaelJackson diedOnDate
?d
```

Result: Correct.

Q: On April 25,1995 first "Jeopardy!" host Art Fleming passed away & the dance was over for this partner of Fred
A: Ginger Rogers

```
ArtFleming diedOnDate ?d . ?p diedOnDate ?d matches (+fred)
```

Result: Correct.

Q: This famed aviator outlived his brother by 35 years, passing away in 1948 on the same day Gandhi was assassinated
A: Orville Wright

```
Gandhi diedOnDate ?d .
?p diedOnDate ?d .
?p type aviator
```

Result: Nearly correct; the YAGO2 result is `WrightBrothers` instead of Orville Wright.

Q: Italian filmmaker Michelangelo Antonioni died in 2007 at age 94 on the same day as this 89-year-old Swedish director
A: Ingmar Bergmann

```
MichelangeloAntonioni diedOnDate ?d .
?p diedOnDate ?d .
?p type director matches (+Swedish)
```

Result: Correct.

Appendix A.2.3. The 19th Century

Q: In the 1840s he began reaping fame & fortune from the sale of his reaping machines
A: Cyrus McCormick

```
?x isa person overlaps [1840,1850] matches (+reaper)
```

Result: Nearly correct. The query returns the correct result and several incorrect ones.

Q: In June 1876 George Custer made his last stand at the Battle of this river
A: Little Bighorn

```
?x isa battle overlaps 1876-06 matches (+George +Custer) .
?x happendIn ?r .
?r isa river
```

Result: Correct.

Q: It was the largest & most powerful state of the German Empire in the 1800s
A: Prussia

```
NA
```

Result: Not expressible.

Q: Much of the fighting in this war, 1853 to 1856, took place on a peninsula in the Black Sea
A: Crimean War

```
?x isa war during [1853,1856] .
?x happenedIn BlackSea
```

Result: Correct.

Q: In 1889 this South American country's last emperor, Pedro II, was forced to abdicate
A: Brazil

```
?x isa country matches (+Pedro +II) .
?x isLocatedIn SouthAmerica
```

Result: Nearly correct. YAGO2 does not know any connection between Pedro II and Brazil, but the query still gives Brazil as one result.

Appendix A.2.4. Canadian Geography

Q: The name of this Ontario capital means “place of meeting” in the Huron Indian language
A: Toronto

```
?x isa capital matches (+Huron) .
?x isLocatedIn Ontario
```

Result: Correct.

Q: Canada’s most densely populated province, it’s known to locals just as “The Island”
A: Prince Edward Island

```
?x isa island .
?x isa province .
?x isLocatedIn Canada
```

Result: Correct.

Q: One of the two largest lakes solely within Canada; both are “Great”
A: Great Bear Lake / Great Slave Lake

```
?x isa lake matches (+largest) .
?x isLocatedIn Canada .
Great% means ?x
```

Result: Correct.

Q: North America’s second-longest river, it flows into the sea in the Northwest Territories
A: The McKenzie

```
?x isa river .
?x islocatedin NorthwestTerritories
```

Result: Nearly correct. Returns 19 rivers, including the correct `Mackenzie`.

Q: Canada’s northernmost point lies on this large island a “mere” 6 degrees, 92 minutes from the North Pole
A: Ellesmere Island

```
?x islocatedin NorthernCanada . ?x isa island matches
(+northernmost)
```

Result: Correct.

Appendix A.2.5. Urban

Q: This Australian city was founded in 1788 as a penal colony
A: Sydney

```
?x isa city locatedIn Australia .
?x wasCreatedOnDate 1788-##-##
```

Result: Correct.

Q: A downtown area of this major city is named for "the loop" formed by its elevated train tracks
A: Chicago

```
?x isa city matches (+loop)
```

Result: Nearly correct. The query returns about 100 cities, including the correct answer `Chicago`.

Q: In 930 A.D. Karmathian Muslim rebels stormed & destroyed this holy city, carrying off the sacred black stone
A: Mecca

```
?x isa Holycities matches (+muslim)
```

Result: Nearly correct. The query returns many results, including the correct one.

Q: By the end of 1999 the transfer of the Bundestag back to Berlin from this city was largely complete
A: Bonn

```
?x isa city matches (+Bundestag)
```

Result: Nearly correct. The query returns 7 cities, one of which is the correct answer `Bonn`.

Q: Site of a famous commando raid, it was the capital of Uganda until 1962
A: Entebbe

```
?x isA city locatedIn Uganda matches (+terrorist) .
```

Result: Nearly correct. The results include the correct answer.

Q: The National Earthquake Information Center is in this “colorful” city just west of Denver
A: Golden, Colorado

```
?x isa city westOf Denver matches (+National +Earthquake
+Information +Center)
```

Result: Correct.

Q: This New Mexico city was founded in 1706 & named for a viceroy of New Spain
A: Albuquerque

```
?x isA city .
?x isLocatedIn NewMexico .
?x wasCreatedOnDate 1706-##-##
```

Result: Correct.

Q: The growth of this state capital in Eagle Valley was stimulated by the discovery of the Comstock Lode in 1859
A: Carson City

```
?x isA city matches (+Comstock +Lode)
```

Result: Nearly correct. The query returns 5 cities, including the correct answer.

Q: This raisin center of more than 400,000 people in California’s San Joaquin Valley has grapes on its seal
A: Fresno

```
?x isLocatedIn California matches (+raisin +center) .
?x hasPopulation ?n .
?n isGreaterThan 400000
```

Result: Correct.

Q: Montana State University has a branch in this city named for frontiersman John
A: Bozeman

```
?k means ?x matches (+Montana +State +University) .
?x isLocatedIn Montana .
?s hasFamilyName ?k .
?s hasGivenName John
```

Result: Nearly correct. The query returns many results, which include the correct answer `Bozeman`.