Recent Topics of Research around the YAGO Knowledge Base

Antoine Amarilli[1], Luis Galárraga[1], Nicoleta Preda[2], and Fabian M. Suchanek[1]

[1] Télécom ParisTech, Paris, France
[2] University of Versailles, France

Abstract. A knowledge base (KB) is a formal collection of knowledge about the world. In this paper, we explain how the YAGO KB is constructed. We also summarize our contributions to different aspects of KB management in general. One of these aspects is rule mining, i.e., the identification of patterns such as $spouse(x, y) \wedge livesIn(x, z) \Rightarrow livesIn(y, z)$. Another aspect is the incompleteness of KBs. We propose to integrate data from Web Services into the KB in order to fill the gaps. Further, we show how the overlap between existing KBs can be used to align them, both in terms of instances and in terms of the schema. Finally, we show how KBs can be protected by watermarking.

1 Introduction

Recent advances in information extraction have led to the creation of large knowledge bases (KBs). These KBs provide information about a great variety of entities, such as people, countries, rivers, cities, universities, movies, animals, etc. Among the most prominent academic projects are Cyc [12], DBpedia [2], Freebase[3], and our own YAGO [21]. Most of these projects are linked together in the Semantic Web [5]. KBs find numerous applications in the industry. The Knowledge Graph released by Google is an example of a large commercial KB project. It contains linked information about millions of people, places, and organizations, and helps Google deliver more semantic search results. Facebook is also building a KB from the information of its users and their interests, and Microsoft, too, is experimenting with a KB to enhance its search results. These projects show not just the advances in technology and the growth of semantic data, but also the rising commercial interest in KBs.

Our work investigates models and algorithms for the automated construction, maintenance, and application of large-scale KBs. The main project is the YAGO knowledge base, which we develop jointly at the Télécom ParisTech Institute in Paris and the Max Planck Institute for Informatics in Germany. YAGO was extracted automatically from Web sources, and contains around 10 million entities and 120 million facts. We use YAGO as an example to study different aspects of KB management in general: how new information can be added automatically

[3] `http://freebase.com`

to a KB, how we can protect KBs from plagiarism, how KBs can be integrated with other KBs, and how we can mine patterns from KBs.

In this paper, we summarize 5 main directions of research. Section 2 describes our latest efforts in the construction of the YAGO KB. In Section 3, we discuss our work on automated matching of one KB to another KB. Our system matches not just instances, but also classes and relations at the same time. In Section 4 we introduce AMIE, a system for mining semantic rules in KBs. Section 5 shows models and algorithms for the integration of Web services into KBs. Section 6 discusses algorithms to protect KBs against plagiarism. We conclude in Section 7 with an outlook.

2 YAGO: Knowledge à la Carte

The YAGO KB. YAGO [21,10,3] is one of the largest public knowledge bases. It contains more than 10 million entities (such as people, cities, rivers, or movies), and more than 120 million facts about them. YAGO knows, e.g., which actors acted in which movies, which cities are located in which countries, and which person is married to which other person. YAGO is constructed by extracting information automatically from Web sources such as Wikipedia. Unlike other such projects, YAGO has a manually confirmed accuracy of 95%.

Achieving such accuracy is no simple task, because YAGO draws from few, but very different sources. The system extracts and merges information from Wikipedia, WordNet, Geonames, the Universal WordNet, and WordNet Domains. Facts have to be extracted from the infoboxes, the categories, and the full text of Wikipedia, and reconciled with conflicting, duplicate, or complementary facts from the other sources. Entities have to be mapped and deduplicated, and class hierarchies have to be merged and combined. In addition, we have to apply a suite of verifications to make sure that domain and range constraints are respected, that functional relations have no more than one object for any given subject, and that the types of an entity are consistent with each other. This entire process takes several days to run. Furthermore, the YAGO team has steadily grown, which requires a careful distribution of responsibilities. Apart from this, more than a dozen researchers work directly or indirectly on the knowledge base. To adapt to these conditions, we have recently taken a radical step, and refactored the YAGO system from scratch into a transparent and modular architecture [3].

The YAGO2s Architecture. The refactored version of YAGO is called YAGO2s. The main ingredients of the new architecture are *themes* and *extractors.* A theme is a collection of facts, such as all facts extracted from Wikipedia infoboxes, all facts derived from WordNet, or all facts that concern people. A theme is stored in a file in the RDF Turtle format.

An extractor is a module of code, which takes a number of themes as input, and produces a number of themes as output. For example, one extractor is the *deduplicator*, which takes a number of themes as input, and produces one theme with the deduplicated facts as output. Other extractors check types, verify functional constraints, or merge information. Some extractors also extract information from an external data source. These extractors take a raw data file as an additional input. The *Wikipedia category extractor*, e.g., takes as input the XML dump of Wikipedia and produces a theme with facts extracted from Wikipedia categories. Similar extractors exist for WordNet [13], UWN [6], and Geonames[4]. We also added an extractor for WordNet domains [11]. The WordNet domains give YAGO a thematic structure of topics, such as "music", "law", and "emotions". Therefore, it is now possible to ask for all entities related to, e.g., "music". An extractor can only be run once its input themes have been produced. This constraint yields a dependency graph, in which some extractors have to run before others, and some can run in parallel.

We have designed a scheduler that respects the dependencies of extractors and themes. Of the 40 extractors, up to 16 run in parallel, producing around 80 themes in 4 days on a 8-core machine. The interplay of data extractors and verification extractors ensures that all facts that make it into the final layer of the architecture have been checked for consistency and uniqueness. Together, the themes of the final layer constitute the YAGO KB.

Applications. The new architecture allows us to add new extractors easily. To exemplify this, we have added a new module to YAGO, which extracts flight information from Wikipedia. Thanks to this module, YAGO now knows which airports are connected by direct flights to which other airports. Since YAGO also has vast data on geographic entities, users can now ask YAGO for flights between any two cities. Our interface will determine all airports in the vicinity of the departure city, all airports in the vicinity of the arrival city, and all direct flights between them [20]. YAGO also finds applications elsewhere. Two of the most prominent applications are the DBpedia KB [2] and the IBM Watson system [7]. DBpedia uses the taxonomy of YAGO to structure its entities into a hierarchy of classes. The IBM Watson system uses YAGO (and other KBs) to answer natural language questions. It has recently beaten the human champion at the US quizz show *Jeopardy!*.

YAGO can be downloaded for free from the Web site[5]. Thanks to the new architecture, facts about entities, literals, or multilingual labels all appear in different themes. The themes can be downloaded separately, so that users can download just what they need. With this concept, called "YAGO à la carte", we hope to facilitate further applications of our KB.

3 PARIS: Aligning Instances and Schemas

KB Alignment. The Semantic Web is a large collection of publicly available knowledge bases (KBs). YAGO is only one of them: there are other KBs, such as DBpedia, Freebase, or domain-specific KBs, which cover music, movies, geographical data, scientific publications, and medical or government data. Many of these KBs are complementary. For instance, a general KB may know who

[4] `http://geonames.org`
[5] `http://yago-knowledge.org`

discovered a certain enzyme, whereas a biological database may know its function and properties. However, since the KBs often use different identifiers for an entity, their information cannot be joined or queried across KBs. In the example, we cannot ask who discovered which enzyme with which properties. In addition, the KBs generally use different relations. For example, YAGO will say that Elvis Presley *wasBornIn* Tupelo, whereas another KB could say that Tupelo is the *placeOfBirth* of Elvis.

We propose an approach, PARIS [17], that solves both of these alignment problems. PARIS can match not just the equivalent entities, but also equivalent classes and relations across two KBs. Since PARIS considers all problems at the same time, we can benefit from a fruitful interplay between schema and instance matching, where the alignment of relations helps the alignment of instances, and the alignment of instances may lead to the alignment of relations.

Model. Our insight is that equalities between instances and relations determine each other. This link is achieved by using *functional relations*. A functional relation is a relation that has at most one second argument for each first argument, and conversely an inverse functional relation has at most one first argument for each second argument. Thus, if two instances x and x' share the same second argument of an inverse functional relation, then they must be equal:

$$\exists r, y, y' : r(x, y) \wedge r(x', y') \wedge y \equiv y' \wedge r \text{ inv. functional} \Rightarrow x \equiv x'$$

For example, if two instances share an email address, and if each email address can belong to only one instance, then the two instances must be equal.

However, real-world KBs are never free from noise. They may contain erroneous statements, and functional constraints may not always be respected. This is why we designed a *probabilistic model* to relax the hard logical rules. We call an *alignment fact* a statement of the form $x \equiv x'$, where x and x' are two entities in the first and second ontology, respectively. We also consider alignment facts of the form $r \subseteq r'$, where r and r' are two relations, and likewise $c \subseteq c'$, where c and c' are classes. For the purposes of our model, we also consider the statement *invfun*(r) an alignment fact. It states that r is an inverse functional relation. A possible world is a set of alignment facts, and we call it simply an *alignment*. Our universe Ω is the set of all possible alignments.

The probability function of our model associates a probability $P(A)$ to each alignment $A \in \Omega$, with the constraint that $\sum_{A \in \Omega} P(A) = 1$. We do not know what this probability distribution is, but we will never try to manipulate it directly. Instead, we will study it through the *marginal probabilities* of the various alignment facts. Formally, for each alignment fact a, we define a random variable X_a such that $X_a(A) = 1$ if $a \in A$, and $X_a(A) = 0$ otherwise. The *marginal probability* of X_a is the total probability of the alignments of Ω where a holds: $P(X_a) = \sum_{A \in \Omega} X_a(A)P(A)$. For brevity we will write the marginal probability of X_a as $P(a)$. We impose constraints on the marginal probabilities, for instance $P(x \equiv x) = 1$ for every entity x. With the *product measure*, we can always construct a probability distribution that respects these constraints: $P(A) = \prod_{a \in A} P(a) \prod_{a \notin A}(1 - P(a))$. The product measure results in de-facto independence between the events of all alignment facts. Therefore, we make the assumption that all the X_a are independent. We can now replace the hard implication rule above by an equation which relates the marginal probabilities for various alignment facts. Namely, for every two instances x and x':

$$P(x \equiv x') = 1 - \prod_{r(r,x), r(x',y')} (1 - P(\mathit{invfun}(r)) \times P(y \equiv y')$$

The equality of instances can help us determine whether a class c of one KB is a subclass of a class c' of the other KB. We estimate this probability as the ratio of instances of c that are instances of c'. Since the instances of c belong to one KB, and the instances of c' belong to the other KB, we must count the overlap of the classes by taking into account the equality of instances that we have already estimated:

$$P(c \subseteq c') = \frac{\sum_{x \in c}(1 - \prod_{y \in c'}(1 - P(x \equiv y)))}{|c|}$$

The probability that one relation subsumes another relation can be estimated in a similar manner. This probability is then factored back into the probability for the equality of instances. This yields a system of equations in which the probability for the equality of instances, the subsumption of relations, and the subsumption of classes depend on each other.

Implementation. To bootstrap the dependencies of the probabilities, we make use of literals. Literals are strings, numbers, and dates. Two identical literals are always considered equal, so the probability that they are aligned is always 1. Starting from these probabilities, we implemented an iterative algorithm, which computes the equalities and relation subsumptions of the current step from the values of the previous step. Once this process has converged, we output the marginal probability scores as estimations for the equalities of instances, subsumptions of classes, and subsumptions of relations.

The large number of instances in today's KBs implies a prohibitively high number of potential matches. A naive implementation would need a quadratic number of comparisons per iteration, which would be practically infeasible. We therefore impose more conditions on the alignment, such as requiring that each entity is matched to at most one other entity. Since the original publication of PARIS [17], we have considerably improved the implementation of the system. We store the KB facts in main memory, and we use a new method to update probability scores: We simultaneously compute entity and relation alignments, and run this in parallel across multiple threads. To match YAGO [21] and DBpedia [2], two of the largest public KBs on the Semantic Web with several dozen million facts each, PARIS needs less than 30 minutes.

We have also experimented with refinements of the approach. For instance, we can align a single relation of one KB with a join of two relations in the other one, or we can use approximate matches between literals rather than exact matches. These additional features may help in specific situations. However, for the scenarios that we considered, our experiments show that the default approach of PARIS is both simpler and more efficient. What is more, it is parameter-free: Unlike its competitors, the system has no thresholds to tune, no similarity functions to design, and no settings to be tried out. In our experiments, the system was run on all different datasets with the default settings.

Results. To show the practical viability of our approach, we have run PARIS on the benchmark matching problems of the Ontology Alignment Evaluation Initiative (OAEI). PARIS outperformed the previously leading system on the instance alignment test data. In addition, it also computed the alignment between relations and classes, which was not even requested by the task.

We also conducted large-scale experiments with real KBs on the Semantic Web. PARIS is able to align YAGO and DBpedia with a precision of 94% on the instances, 84 % on the classes, and 100% on the relations (weighted by their number of occurrences). This alignment revealed interesting correspondences between the KBs, such as different naming policies, different design decisions, or redundancies within one KB. PARIS was also able to align YAGO with an ontology built from IMDb (the Internet Movie Database).

All data, alignments, and results, as well as our implementation, are available on the Web site of the project[6]. Our alignments have become part of the Semantic Web, and thus contribute to the vision of a large Web of linked data, where the KBs truly complement each other.

4 AMIE: Mining Logical Rules

Rules. Knowledge bases can show us which facts are typically true about entities. For example, we could find:

$$\mathit{motherOf}(m,c) \land \mathit{marriedTo}(m,f) \Rightarrow \mathit{fatherOf}(f,c)$$

This rule says that the husband of a mother is often the father of her children. Such a rule does not always hold. Still, such rules can be interesting for several reasons. First, by applying such rules on the data, new facts can be derived that make the KB more complete. For example, if we know the mother of a child and her husband, we can infer the father. Second, such rules can identify potential errors in the knowledge base. If, for instance, the KB claims that a totally unrelated person is the father of a child, then maybe this statement is wrong. Finally, rules about general tendencies can help us understand the data better. We can, e.g., find out that countries often trade with countries speaking the same language, that marriage is a symmetric relationship, that musicians who influence each other often play the same instrument, and so on.

While mining logical rules has been well studied in the Inductive Logic Programming (ILP) community, mining logical rules in KBs is different in two aspects: First, current rule mining systems are easily overwhelmed by the amount of data (state-of-the art systems cannot even run on today's KBs). Second, ILP usually requires counterexamples. KBs, however, implement the open world

[6] `http://webdam.inria.fr/paris`

assumption (OWA), meaning that absent data cannot be used as counterexamples. On that ground, we developed the AMIE approach [9]. AMIE learns a set of meaningful logical rules from a KB. She uses a new mining model and a confidence metric, suitable for potentially incomplete KBs under the OWA.
Model. Technically, a Horn rule takes the form

$$r_1(x_1, y_1) \wedge ... \wedge r_n(x_n, y_n) \Rightarrow r(x, y)$$

Here, all $r_i(x_i, y_i)$ and $r(x, y)$ are binary atoms, each containing a relation name r_i, and constants or variables x_i, y_i. The left hand side is called the *body* and we abbreviate it by $\mathcal{B}$. $r(x, y)$ is the called the *head*. AMIE mines only closed rules, i.e., each variable must appear at least twice in the rule. This constraint ensures the rule can make concrete predictions as follows: Whenever some facts of the KB match the body of the rule, the rule predicts the instantiated head atom as a new fact. For instance, if the KB knows *motherOf(Priscilla,Lisa)* and *marriedTo(Priscilla,Elvis)*, then our example rule will predict *fatherOf(Elvis,Lisa)*. Our goal is to find rules that are true for many instances in the KB. We measure this by the *support* of the rule:

$$supp(\mathcal{B} \Rightarrow r(x, y)) := \#(x, y) : \exists z_1, ..., z_m : \mathcal{B} \wedge r(x, y)$$

Here, $\#(x, y)$ is the number of pairs x, y that fulfill the condition, and the $z_1, ..., z_n$ are the variables of $\mathcal{B}$. We want to count not just the cases where the rule makes a correct prediction, but also where it makes a wrong prediction. However, KBs usually do not contain negative information, and so we cannot say that a prediction is wrong. We can only count the predictions that are not in the KB. The *standard confidence* is a measure borrowed from association rule mining [1], which computes the ratio of predictions that are in the KB:

$$conf(\mathcal{B} \Rightarrow r(x, y)) := \frac{supp(\mathcal{B} \Rightarrow r(x, y))}{\#(x, y) : \exists z_1, ..., z_m : \mathcal{B}}$$

This measure punishes a rule if it makes many predictions that are not in the KB. However, due to the Open World Assumption, not all absent facts are wrong. Furthermore, such punishment is even counter-productive, because we want to use the rule to predict new facts.

To address this problem, AMIE resorts to the *Partial Completeness Assumption* (PCA). The PCA is the assumption that if the database knows $r(x, y)$ for some x and r, then it knows all facts $r(x, y')$. This assumption is sound for functional relations (such as *wasBornOnDate*, *hasCapital*). If we reverse all inverse functional relations (such as *creates* and *owns*), the assumption holds also for them. Even for non-functional relations, the PCA is still reasonable for KBs that have been extracted from a single source (such as DBpedia and YAGO). These usually contain either all objects or none for a given entity and a given relation.

The PCA can be used to infer negative information. For instance, if the database knows the citizenship of a person, then any other statement about her citizenship is assumed to be false. The *PCA confidence* normalizes the support of the rule only against the facts that are known to be true or false according to the PCA.

$$pcaconf(\mathcal{B} \Rightarrow r(x,y)) := \frac{supp(\mathcal{B} \Rightarrow r(x,y))}{\#(x,y) : \exists z_1, ..., z_m, y' : \mathcal{B} \wedge r(x,y')}$$

The denominator of the PCA confidence expression includes all pairs x, y for which the r values of x are in the KB. If the relation is incomplete, those gaps are not used as counter-evidence.

Results. Our experiments compare the usability and running times of AMIE against two state-of-the-art systems: WARMR and ALEPH[7]. Since these approaches are designed to serve more general purposes, they require additional input such as a language bias and, in some cases, output post-processing. In contrast, AMIE runs out of the box with her default setting. Furthermore, AMIE runs several orders of magnitude faster. For example, AMIE can mine rules on the entity-entity facts of the YAGO2 KB (approx. 1M facts) in only 4 minutes. The other systems did not terminate or failed to start on this KB due to its size. Since the original publication [9], we have further improved our implementation by means of fine-grained optimizations such as query rewriting and smarter query plans. This has improved the running time significantly. The latest version can run on YAGO2 in approximately 30 seconds.

We used the rules mined by AMIE on YAGO2 to predict new facts, and evaluated the facts by comparing to the newer version of the KB (YAGO2s), or by manually comparing to Wikipedia. Our results show that the PCA confidence outperforms the standard confidence in terms of precision and recall. The top rules ranked by PCA confidence produce many more predictions than the top rules ranked by standard confidence, while at the same time the aggregated precision of the results is higher. Our predictions beyond YAGO2 have an overall precision of 40%. While the rules cannot be used directly to infer new facts for KBs, they still provide a signal that can be used, e.g., in conjunction with other rules or with other prediction mechanisms. Furthermore, we show that the PCA confidence is a better estimate of the actual precision of rules. All results are available on the Web site of the project[8].

Ontology Alignment. Rule mining can be used not just to mine Horn rules, but also to align two KBs. In Section 3, we have already seen an approach that can align the instances, classes, and relations of two KBs. Unfortunately, there are cases where a simple 1:1 alignment of relations is not enough. For instance, KB $\mathcal{K}$ may express the relation between a person and their native land by the relationship *k:wasBornInCountry*, while $\mathcal{K}'$ may require a join of the relationships *k':wasBornInCity* and *k':locatedInCountry*. Here, a "one-hop" relation of one KB has to be aligned with a "two-hop" relation in the other. In [8], we propose to discover such alignments by rule mining. As input, the approach requires two KBs whose instances have already been aligned. We coalesce the KBs into a single KB, where facts about the same entity use the same entity

[7] `http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph_toc.html`
[8] `http://mpi-inf.mpg.de/departments/ontologies/projects/amie`

identifier. This coalesced KB mixes the facts from one KB with the facts of the other. Then we use AMIE to mine rules in which the body atoms are restricted to relation names from the first KB, and the head atom must use a relation from the second KB. In the example, we could mine

$$\textit{k':wasBornInCity}(x, z) \wedge \textit{k':locatedInCountry}(y, z) \Rightarrow \textit{k:wasBornInCountry}(x, z)$$

We call these expressions *rules for ontological schema alignment*, ROSA rules for short. They express common structural mappings between two ontologies, such as relation subsumptions, class subsumptions, relation equivalences, two-hop subsumptions, and predicate-object translations, among others. However, ROSA rules define just a subset of all the possible mappings required for the alignment of ontologies in the Semantic Web, and thus give us room for further research.

5 SUSIE: Integrating Web Services

Web Services. A growing number of data providers let us access their data through Web services. There are Web services about books (`isbndb.org`, `librarything.com`, Amazon, AbeBooks), about movies (`api.internetvideoarchive.com`), about music (`musicbrainz.org`, `lastfm.com`), and about a large variety of other topics. We have studied Web services under a variety of aspects [14,4,16].

The API of a Web service restricts the types of queries that the service can answer. For example, a Web service might provide a method that returns the songs of a given singer, but it might not provide a method that returns the singers of a given song. If the user asks for the singer of some specific song, then the Web service cannot be called – even though the underlying database might have the desired piece of information. This problem is particularly pronounced if multiple Web services have to be combined in order to deliver the answer to the user query. In this case, it may happen that the API restrictions force the query answering system into an infinite series of attempts to orchestrate the services to no avail.

The Web as an Oracle. With the SUSIE project [15], we propose to use Web-based information extraction (IE) on the fly to determine the right input values for asymmetric Web services. For example, assume that we have a Web service *getSongsBySinger*, which returns the songs of a given singer. Now assume that the user wishes to find the singer of the song *Hallelujah*. This query cannot be answered directly with the service *getSongsBySinger*. Therefore, we issue a keyword query "singers Hallelujah" to a search engine. We extract promising candidates from the result pages, say, *Leonard Cohen*, *Lady Gaga*, and *Elvis Presley*. Next, we use the existing Web service to validate these candidates. In the example, we would call *getSongsBySinger* for every candidate, and see whether the result contains *Hallelujah*. This confirms the first singer and discards the others. This way, we can use an asymmetric Web service as if it allowed querying for an argument that its API does not support.

We show how such functions can be integrated into a Web orchestration system, and how they can be prioritized over infinite chains of calls. For this purpose, we define the notion of *smart service calls*. These are those calls for which we can guarantee an answer under certain conditions. We have implemented our system, and shown in experiments with real-world Web services that SUSIE can answer queries on which standard approaches fail.

With SUSIE, we have opened the door to an interesting suite of research questions: How can promising calls be prioritized over less promising calls? Under which assumptions can we give guarantees that a call composition will be successful? We plan to investigate these questions in future work.

6 Watermarking

Licensing. Most KBs on the Internet are available for free. However, in most cases, their use is governed by a license: If a user re-publishes the data or part of the data, he has to give credit to the creators of the original KB. If he does not, then this constitutes plagiarism. In some cases, re-publication may be prohibited completely (e.g., for commercially licensed KBs).

This raises the question of how we can prove if someone re-published the data. Since ontological statements are usually world knowledge, there is no way we can show that someone took the data from us. The other person might as well have taken the data from a different source. He might even claim that we took the data from him. We propose to address this problem through watermarking. We developed two approaches: Additive Watermarking and Subtractive Watermarking.

Additive Watermarking. Additive Watermarking works by adding a small number of wrong statements to the KB ("fake facts"). If these fake facts appear in another KB, then the other KB most likely took the data from our KB. The fake facts have to be plausible enough in order not to be spotted by a machine or by a human. At the same time, they may not be so plausible that they are correct. We provide a theoretical analysis of how many facts we have to add in order to ensure plausibility and security at the same time.

The main objection to this approach is that it compromises the data quality of the KB. It is true that watermarking is always a trade-off between data quality and the ability to prove provenance. However, our technique has to add only very few fake facts, usually a handful or a dozen. Large, automatically constructed KBs contain anyway several thousands of wrong facts. YAGO, for example, one of the KBs with a particularly rigorous quality assessment, has a guaranteed correctness of 95%. Since YAGO contains millions of facts, thousands are wrong. Adding a few more might be a valuable trade-off.

We show in a system demonstration of our approach how fake facts can be generated in such a way that most of them go undetected by a human [18].

Subtractive Watermarking. Subtractive Watermarking works by removing a small number of statements from the KB. The KB is then published without these statements. This creates a pattern of "holes" in the KB, which we can imagine like holes in a cheese. If this pattern of holes appears in another KB, then the data has likely been taken from the source KB.

The main advantage of this approach is that it does not compromise the precision of the data. It just removes statements. The Semantic Web is governed by the Open World Assumption, which states that the absence of a statement implies neither its truth nor its falsehood. Thus, the removal of a statement does not influence the correctness of the data. It does influence its completeness, though. As always, watermarking remains a trade-off between the quality of the data and the ability to prove provenance.

We show in theoretical analyses that only a few hundred facts have to be removed in order to protect the KB effectively from plagiarism. If this number is slightly increased, then the method can work even if only part of the KB is plagiarized [19]. Further information on watermarking KBs can be found on the Web page of our project[9].

7 Outlook

In this paper, we have summarized approaches that address challenges in the mining, linking, and extension of knowledge bases (KBs). Research in these areas has made huge progress during the last decade. However, many challenges remain. One issue is making those methods run at Web scale. Computers become ever more powerful, but we also produce ever more data. Currently, the growth rate of data outpaces the advancement rate of computers. Therefore, new methods will have to be developed to extract information at scale, to integrate it with existing information, and also to make use of large-scale semantic data. It is not just the size of the data that poses a challenge, but also the different types of information that we encounter. Social media, such as Twitter, Blogs, or Facebook, have seen a rise in recent years. The public parts of these sources could be harvested for KBs. Finally, new applications for semantic data will be explored. This includes its use in search, translation, decision making, or education. Ultimately, our goal is to make computers ever more useful for mankind.

References

1. Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2), June 1993.
2. Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A Nucleus for a Web of Open Data. In *ISWC*, 2007.
3. Joanna Biega, Erdal Kuzey, and Fabian M. Suchanek. Inside YAGO2s: A transparent information extraction architecture. In *Proc. WWW (Companion volume)*, 2013.

[9] `http://mpi-inf.mpg.de/departments/ontologies/projects/watermarking`

4. Meghyn Bienvenu, Daniel Deutch, Davide Martinenghi, Pierre Senellart, and Fabian Suchanek. Dealing with the deep web and all its quirks. In *VLDS workshop*, 2012.
5. Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee. Linked data on the Web. In *WWW*, 2008.
6. Gerard de Melo and Gerhard Weikum. Towards a universal wordnet by learning from combined evidence. In *CIKM*. ACM, 2009.
7. David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3), 2010.
8. Luis Galárraga, Nicoleta Preda, and Fabian M. Suchanek. Mining rules to align knowledge bases. In *AKBC workshop*, 2013.
9. Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. AMIE: Association Rule Mining under Incomplete Evidence in Ontological Knowledge Bases. In *WWW*, 2013.
10. Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194, 2013.
11. Bernardo Magnini and Gabriela Cavaglia. Integrating subject field codes into wordnet. In *LREC*, 2000.
12. Cynthia Matuszek, John Cabral, Michael Witbrock, and John Deoliveira. An introduction to the syntax and content of cyc. In *AAAI Spring Symposium*, 2006.
13. George A. Miller. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
14. N. Preda, G. Kasneci, F. M. Suchanek, T. Neumann, W. Yuan, and G. Weikum. Active Knowledge : Dynamically Enriching RDF Knowledge Bases by Web Services. (ANGIE). In *SIGMOD*, 2010.
15. Nicoleta Preda, Fabian M. Suchanek, Wenjun Yuan, and Gerhard Weikum. Susie: Search using services and information extraction. In *ICDE*, 2013.
16. Fabian Suchanek, Alessandro Bozzon, Emanuele Della Valle, Alessandro Campi, and Stefania Ronchi. *Towards an Ontological Representation of Services in Search Computing*. LNCS vol. 6585. Springer, April 2011.
17. Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *PVLDB*, 5(3), 2011.
18. Fabian M. Suchanek and David Gross-Amblard. Adding fake facts to ontologies. In *WWW (Companion volume)*, 2010.
19. Fabian M. Suchanek, David Gross-Amblard, and Serge Abiteboul. Watermarking for ontologies. In *ISWC*, 2011.
20. Fabian M. Suchanek, Johannes Hoffart, Erdal Kuzey, and Edwin Lewis-Kelham. YAGO2s: Modular High-Quality Information Extraction with an Application to Flight Planning. In *German Database Symposium (BTW 2013)*, 2013.
21. Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A core of semantic knowledge - unifying WordNet and Wikipedia. In *WWW*, 2007.