A knowledge base from multilingual Wikipedias - YAGO3

Une base de connaissances des Wikipédias plurilingues – YAGO3

Farzaneh Mahdisoltani
Joanna Biega
Fabian M. Suchanek

2014D003

août 2014

Département Informatique et réseaux
Groupe IC2 : Interaction, Cognition et Complexité

A Knowledge Base from Multilingual Wikipedias – YAGO3

Une base de connaissances des Wikipédias plurilingues – YAGO3

Farzaneh Mahdisoltani
Max Planck Institute for Informatics, Germany

Joanna Biega
Max Planck Institute for Informatics, Germany

Fabian M. Suchanek
Télécom ParisTech, France

August 2014

Abstract

We present YAGO3, an extension of the YAGO knowledge base that combines the information from the Wikipedias in multiple languages. Our technique fuses the multilingual information with the English WordNet to build one coherent knowledge base. We make use of the categories, the infoboxes, and Wikidata, and discover the meaning of infobox attributes across languages. We run our method on 10 different languages, and achieve a precision of 95%-100% in the attribute mapping. Our technique enlarges YAGO by 1m new entities and 7m new facts.

Résumé

Nous présentons YAGO3, une extension de la base de connaissances YAGO, qui combine les informations provenant de Wikipédias en plusieurs langues. Notre technique combine l'information plurilingue avec la version anglaise de WordNet afin de créer une base de connaissance cohérente. Nous utilisons les catégories, les infoboxes, et Wikidata, et nous découvrons ainsi la signification des attributs des infoboxes pour chaque langue. Nous utilisons notre méthode sur 10 langues, et nous obtenons une précision de 95%-100% pour la correspondance des attributs entre eux. Notre technique étoffe YAGO avec 1m de nouvelles entités et 7 millions de nouveaux faits.

1 Introduction

Motivation. Wikipedia[1] is one of the most popular online encyclopedias. Several projects construct knowledge bases (KBs) from Wikipedia, with some of the most prominent projects being DBpedia [3], Freebase[2], and YAGO [23]. These KBs contain millions of entities, including people, universities, cities, organizations, or artworks. These entities are structured into a taxonomy of classes, where more general classes (such as *person*) subsume more specific classes (such as *singer*). The KBs also contain hundreds of millions of facts about these entities, such as which person was born where, which singer sang which song, or which city is located in which country. Unlike Wikipedia itself, the KBs store this knowledge in a structured form of subject-predicate-object triples, which allows one to query it like a database.

So far, most extraction approaches have focused on the English version of Wikipedia. With 4.5 million articles, it is the largest Wikipedia. However, there are dozens of other Wikipedias in different languages. Several of these have more than a million articles. If we could tap the knowledge of these Wikipedias, we could gain thousands, if not millions of new entities and facts – e.g., those entities that are too local to be described in the English Wikipedia.

This is the treasure that we want to unearth: Our goal is to construct a KB from the Wikipedias in different languages. Crucially, we want to build not several KBs, but one coherent fact collection from these different sources.

State of the Art and Limitations. Several projects extract information from multilingual Wikipedias. However, these projects either build up one KB per language [3], fuse data across different Wikipedias without building a central KB [21, 25, 20, 5, 1, 26], or exclude the facts from the infoboxes [19, 7, 18]. The infoboxes contain information about the article entity in the form of attribute-value pairs, and are thus a very rich source of knowledge. Despite a lot of progress on several aspects of multilingual extraction, the community still does not have a single coherent KB built from the Wikipedias in different languages.

Challenges. Building a coherent KB from different Wikipedias is not an easy task. The first challenge is extracting knowledge from the infoboxes. The infobox attributes usually appear in a foreign language and are not shared across different Wikipedias. Thus, they have to be mapped to the canonical relations of the central KB. Since there are thousands of infobox attributes, this is very hard to achieve. Furthermore, the extraction from Wikipedia is error-prone, and so the data has to be cleaned in order to be of use for a central KB. Finally, the challenge is creating a taxonomy that reaches across different languages, and that integrates entities from all Wikipedias under the same roof.

Contribution. In this paper, we propose a holistic approach for the creation of a full-fledged KB on top of Wikipedias in different languages. Our approach maps multilingual infobox attributes to canonical relations, merges equivalent entities into canonical entities by help of Wikidata, cleans the data, and arranges all entities into a single taxonomy. The result is YAGO3, the successor of YAGO.

[1] `http://wikipedia.org`
[2] `http://freebase.com`

Our key advantage is that we do not align different noisy extractions from different Wikipedias, but different Wikipedias with a central clean KB. This yields an approach that is remarkably simple and elegant. Yet, it works with European languages as well as with non-European ones, across different scripts, and with an accuracy of 95%-100%. In total, we gain 1m new entities and 7m new facts over the original YAGO.

The rest of this paper is structured as follows. We discuss related work in Section 2. We discuss preliminaries in Section 3, before we present our approach in Section 4. Section 5 shows our experiments and our data, and Section 6 shows some applications of the KB, before Section 7 concludes.

2 Related Work

Several works have harvested the multilingual data from Wikipedia.

Wikidata. The Wikidata project builds a KB through crowd sourcing. The community members add the facts manually to the KB. So far, Wikidata has gathered 14 million entities. However, most entities have only few facts. On the long run, Wikidata aims to incorporate, consolidate, and replace the Wikipedia infoboxes, lists, and categories. Thus, our approach and Wikidata share the same goal of creating a common multilingual KB. While Wikidata is a community effort, our approach is automated. We believe that the two projects can complement each other: Our approach builds on Wikidata, and we believe that Wikidata could benefit from our results in return.

DBpedia. The DBpedia project has launched KB construction projects for Wikipedias in different languages. The community maps the infobox attributes manually to the central ontology. Inspired by this idea, we also aim to construct a single KB from the Wikipedias. However, different from DBpedia's crowd-sourced approach, we aim at an automated approach.

Lexical KBs. Several projects [7, 18, 17, 19] make use of the multilingual data in Wikipedia to construct dictionaries and concept networks. MENTA [7, 10] collects entities from all editions of Wikipedia as well as WordNet into a single coherent taxonomic class hierarchy. BabelNet [19] is built by integrating lexicographic and encyclopedic knowledge from WordNet and Wikipedia. We share the goal of a unified ontology, but want to add to this structure also the consolidated facts from the infoboxes in different Wikipedias. This is an entirely different problem.

[8] straightens the inter-language links in Wikipedia. This task has been addressed also by the Wikidata community, and we make use of the latter.

Cross-Lingual Data Fusion. A large number of works extract information from Wikipedia (see, e.g., [15] for an overview). Of these, several approaches consolidate information across different Wikipedias [21, 25, 20, 5, 1, 26]. We also want to align information across Wikipedias, but our ultimate goal is different: Unlike these approaches, we aim to build a single coherent KB from the Wikipedias, which includes a taxonomy. This goal comes with its own challenges, but it also allows simplifications. Our infobox alignment method is considerably simpler, and requires no similarity functions or machine learning methods. Still, as we show in our experiments, we achieve precision and recall values comparable to previous methods. Second, unlike previous approaches that have been shown to work on 4 or less languages [21, 5, 20, 25, 1, 26], we can show that our method is robust enough to run across 10 different languages, different scripts, and thousands of attributes. In addition, we construct a coherent knowledge base on top of these knowledge sources.

Ontology Alignment. A large number of works have looked into the alignment of entities, relations, and classes across KBs (see, e.g., [22, 27] and references therein for recent works). PARIS [22] is a probabilistic approach for the automatic alignment of ontologies. It aligns instances, relations and classes by measuring degrees of matchings based on probability estimates. We show in our experiments that we achieve comparable precision and recall to this approach in our alignment of infobox attributes. At the same time, we construct an entire unified knowledge base on top of the different knowledge sources. This includes a unified taxonomy, the resolution of attribute names across 10 different languages, and the insertion of the data into one central schema.

3 Preliminaries

RDFS. The Resource Description Framework Schema (RDFS) is a W3C standard for knowledge representation. It is used in most major KBs. RDFS is based on a set $\mathcal{U}$ of *resources*. In most applications, the resources are partitioned into instances $\mathcal{I}$, relations $\mathcal{R}$, literals $\mathcal{L}$, and classes $\mathcal{C}$, with $\mathcal{U} = \mathcal{I} \dot{\cup} \mathcal{R} \dot{\cup} \mathcal{L} \dot{\cup} \mathcal{C}$. An *instance* is any entity of the world, such as a person, a city, or a movie. A *class* (or *type*) is a name for a set of instances. The class `city`, e.g., is the name for the set of all instances that are cities. A *relation* is a name for a relationship between resources, such as `loves`, or `livesIn`. Every relation r comes with a domain $dom(r) \in \mathcal{C}$ and a range $ran(r) \in \mathcal{C}$. A *literal* is number, string, or date. Literals usually take the form `"string"^^ datatype`. Here, `string` is the string representation of a number, date, or other literal. `datatype` is a resource. For YAGO, the datatypes behave exactly like classes: Every literal is considered an instance of its datatype. Usually, instances, classes, and relations are prefixed by a namespace. We omit the namespace in this paper for legibility. In all of the following, we assume fixed sets $\mathcal{U}, \mathcal{I}, \mathcal{R}, \mathcal{L}, \mathcal{C}$. A *statement* (or *fact*) is a triple $s \in (\mathcal{U} \setminus \mathcal{L}) \times \mathcal{R} \times \mathcal{U}$, and usually for most statements s, $s \in \mathcal{I} \times \mathcal{R} \times (\mathcal{I} \cup \mathcal{L})$. The statement says that the first component (the *subject*) stands in the relation given by the second component (the *predicate*) with the third component (the *object*), as in ⟨`Elvis, marriedTo, Priscilla`⟩. We use the statement ⟨`Elvis, type, singer`⟩ to say that `Elvis` is an instance of the class `singer`. We use ⟨`singer, subClassOf, person`⟩ to say that `singer` is a subclass of `person`. The `subClassOf`-relationship is transitive. A *knowledge base* (KB) is set of statements.

Wikipedia. The online encyclopedia Wikipedia is written by a community of volunteers. It is available in 287 languages, and 9 of them have more than

1m articles. The English edition currently has 4.5m articles. The articles are written in the Wiki markup language. Each article usually describes one concept or entity. Most articles are members of one or several *categories*. The article about Elvis Presley, e.g., is in the categories "American baritones", and "1935 births". Furthermore, many articles have an *infobox*. An infobox is a set of attribute-value pairs with information about the article entity, such as {*birthplace = Tupelo, birthdate = 8 January 1935, ...*}. The infoboxes are grouped into *templates*, which often carry the name of the class of the article entity. For example, the infobox for Elvis belongs to the template *singer*. The templates define which attributes may be used. However, the templates are not used consistently, and the attributes vary widely across articles.

Wikidata. The Wikidata project aims to be a structured version of Wikipedia. It is a KB that is fed with facts by volunteers. Wikidata provides central abstract identifiers for entities and links them to the articles in the Wikipedias in different languages. For example, Elvis Presley has the identifier `Q303`, and has links to the Wikipedia articles in 147 languages. Wikidata provides similar data for infobox templates and category names.

WordNet. The online dictionary WordNet [16] aims to cover all words of the English language. WordNet groups synonymous nouns together into *synsets*, which can be interpreted as classes. For example, `person` and `human` are in the same synset, which is the class of people. WordNet structures the synsets into an `subClassOf` hierarchy (a *taxonomy*), where, e.g., the synset of people is below the synset of animals.

YAGO. YAGO [23, 24] is a large KB constructed from Wikipedia and WordNet. In its new version, YAGO 2 [13, 4], several new sources were added, including geonames[3] and the Universal WordNet [9]. For this paper, we exclude the newer sources from the construction process, and stay with WordNet and Wikipedia. This subset of YAGO contains 3.4m entities, 17m facts, and 77 manually defined relations.

YAGO Architecture. The architecture of the extraction system [4] is based on *extractors* and *themes*. A theme is a set of facts stored in a file. An extractor is a software module, which takes as input a set of themes and other data, and produces a set of output themes. Extractors can extract data from Wikipedia, WordNet, or other sources. They can also postprocess themes produced by other extractors, and perform deduplication, verification, or constraint checking. These dependencies yield a bipartite graph of themes and extractors, where every extractor is connected to the themes it consumes and the themes it produces. A scheduling module calls the extractors in parallel so that an extractor starts as soon as all its input themes are available. Figure 1 shows an excerpt from this graph; the full graph is available at `http://yago-knowledge.org`. Some extractors exist in several instantiations, because they perform the same task on different input data. All in all, YAGO uses 40 extractor instantiations.

[3] `http://geonames.org`

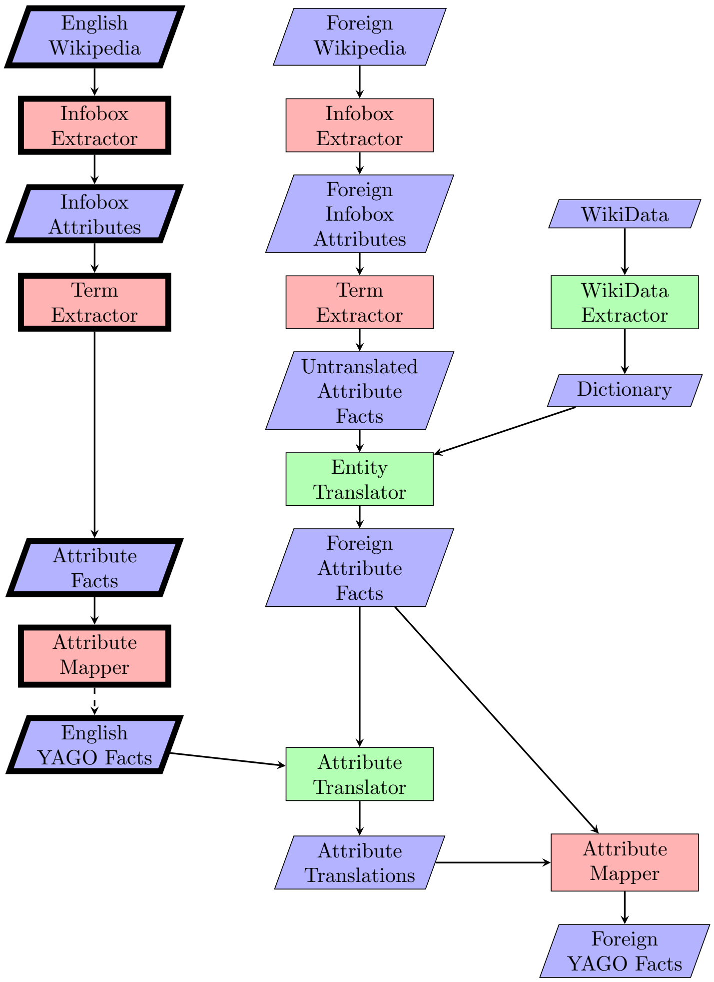


Figure 1: **Extraction from infoboxes**
In bold: English extraction
In light: multilingual extraction
In green: newly designed extractor modules

4 Approach

Our input is a list of Wikipedia languages, and our goal is to create a KB from these multilingual Wikipedias. This comes with 3 challenges: (1) We have to determine the set of entities, (2) we have to extract facts about these entities from Wikipedia, and (3) we have to create a taxonomy.

We will now show how these 3 challenges can be overcome holistically. Our key advantage is that we can leverage the existing English YAGO as a reference

KB and as a taxonomic backbone. Furthermore, the YAGO architecture is modular enough to allow an organic extension beyond the English Wikipedia. By adding extractors in the right places, we arrive at an elegant, yet effective solution for the creation of a full-fledged KB from multilingual Wikipedias.

4.1 Set of Entities

In YAGO, every Wikipedia article becomes an entity. In YAGO3, we have to take care not to duplicate entities, because the same entity can be described in different Wikipedias. We use Wikidata for this purpose. Wikidata maintains its own repository of entity identifiers, and maps them to articles in Wikipedias in different languages. For example, Elvis Presley has the identifier `Q303`, and has links to the Wikipedia articles in 147 languages. Wikidata provides similar data for the category names. We designed a *Wikidata extractor* (Figure 1 top right), which takes Wikidata as input, and produces a theme *Dictionary* as output. This theme contains facts that map every foreign category name and entity name to its English counterpart:

```
"de/Amerikanische Sänger" hasTranslation
      "American Singer"
de/Elvis hasTranslation Elvis
```

We prefix entities and categories with the language code of their Wikipedia edition. Some foreign entities have no English counterpart. In these cases, our method chooses the first language in the input list of languages in which the entity appears. For example if our list of languages is ⟨English, French, Italian, German⟩, and if the Italian Wikipedia contains the village of *Loano*, then we will seek this entity first in English and then in French before defaulting to Italian. This gives us a file that maps every foreign and English article name to a *unique entity name*. The set of these unique names is our set of entities.

4.2 Fact Extraction

In this section, we explain how we extract facts from infoboxes. We first treat the existing extractors for the English YAGO and then explain the new extractors for the multilingual YAGO.

4.2.1 English Extraction

Extraction from Infoboxes. The YAGO system extracts facts about the article entity from infoboxes, such as birth places, authored books, or physical attributes. We first explain the existing extractors (Figure 1, left). The *Infobox Extractor* (top left) parses out attribute-value pairs and produces the theme *Infobox Attributes*. These are raw pairs as they appear in the article, and take, e.g., the following form:

```
Elvis infobox-spouse "[Pricilla Presley], 1967"
```

The *Term Extractor* extracts all possible terms from the attribute value. These are instance names, and literals such as dates, numbers, and quantities. The resulting theme, *Attribute Facts*, contains, e.g.:

```
Elvis infobox-spouse Priscilla_Presley
Elvis infobox-spouse "1967-##-##"^^xsd:date
```

YAGO uses wildcards if components of a date are unknown. The *Attribute Mapper* uses manually defined mappings between the English infobox attributes and the YAGO relations to produce facts in YAGO's schema. This yields, e.g.,

```
Elvis marriedTo Priscilla_Presley
Elvis marriedTo "1967-##-##"^^xsd:date
```

Type Checking. YAGO performs several checks on its facts. One of them is the *type check*. A fact $\langle x, r, y\rangle$ is accepted, if $\langle x,$ `type,` $dom(r)\rangle$ and $\langle y,$ `type,` $ran(r)\rangle$ are part of the KB. YAGO defines classes not just for instances, but also for literals. These include dates, strings, and numbers with their subclasses of rational numbers, integers, positive integers, and so forth. Every such class comes with a manually defined regular expression that identifies literals of this class. For example, the class *integer* comes with the regular expression "`[+-]?[0-9]+`". The type check $y \in ran(r)$ is performed by matching y to the regular expression of $ran(r)$. This way, the type check can filter out non-conforming facts for both entities and literals. In the example, the type checker will reduce the facts to

```
Elvis marriedTo Priscilla_Presley
```

Funclash Checking. Some of YAGO's relations have been defined as *functional*. If r is a functional relation, this imposes $\langle x, r, y\rangle \wedge \langle x, r, y'\rangle \Rightarrow y = y' \quad \forall x \in \mathcal{I}, y \in \mathcal{I} \cup \mathcal{L}$. If two fact candidates $\langle x, r, y\rangle, \langle x, r, y'\rangle$ appear with $y \neq y'$ for some functional relation r, we have a *functional clash* (*funclash* for short). In this case, either one or both of the candidates have to be discarded. In the current YAGO implementation, the input themes are sorted in such a way that the more precise information (from the infoboxes) precedes the more coarse information (from the categories). In case of a funclash, the fact from the first theme is kept, and all potentially following clashing facts are discarded.

Type checking and funclash checking are performed by extractors that read one or several themes as input, and produce a theme as output that contains only the facts that pass the check. Finally, the themes are merged together. In this process, duplicates are removed. Furthermore, less specific facts (such as ⟨`Elvis, wasBorn, "1935-##-##"`⟩) are removed if more specific facts are available (such as ⟨`Elvis, wasBorn, "1935-01-08"`⟩).

4.2.2 Multilingual Extraction

Foreign Infoboxes. The extraction from multilingual Wikipedias proceeds similar to the English one. The center branch of Figure 1 uses the same extractors, and is replicated for every input language. For example, for German, the theme *Foreign Infobox Attributes* may contain:

```
de/Elvis de/heirat "[Priscilla Presley],
      1967 in [Las Vegas (Stadt)]"
```

This fact states that Elvis married Priscilla in 1967 in the city of Las Vegas. As for the English Wikipedia, the *Term Extractor* extracts all possible terms:

```
de/Elvis de/heirat de/Priscilla_Presley
de/Elvis de/heirat "1967-##-##"^^xsd:date
de/Elvis de/heirat de/Las_Vegas_(Stadt)
```

Different languages have different ways to express numbers and dates (this is true in particular for Farsi). We adopt a conservative approach: We run our standard term extractor, and extract only dates and numbers that follow the English convention. Our date extraction is designed in such a way that it only extracts dates in which the order of day, month, and year can be unambiguously established. We leave the adaptation of the term extractor to different languages for future work. After the term extraction, the *Entity Translator* uses the dictionary to translate the entities to their unique names:

```
Elvis de/heirat Priscilla_Presley
Elvis de/heirat "1967-##-##"^^xsd:date
Elvis de/heirat Las_Vegas
```

We call these facts *foreign attribute facts*.

Attribute Mapping. While the entities of the foreign attribute facts have been mapped to their language-independent unique names, the attributes have still to be mapped to the YAGO schema. In the example, we want to deduce that the German infobox attribute *de/heirat* corresponds to the YAGO relation *marriedTo*. Let F_a be the set of foreign infobox attribute facts with a given attribute a (e.g., $a =$*de/heirat*), and let E_r be the set of English YAGO facts with a given relation r (e.g., $r =$*marriedTo*). We want to determine whether a maps to r.

In principle, we could deduce this mapping from the fact that a F_a and E_r will share many subject-object pairs. In practice, this is challenging for several reasons: First, the term extractor produces dozens of terms per attribute value, and only very few of them are the intended objects. Second, the foreign Wikipedia may contain facts that YAGO does not contain. Vice versa, YAGO may contain facts that the foreign Wikipedia does not contain. Third, there may be *spurious matches*. The foundation year of a village, e.g., may coincide with its number of inhabitants.

Matches. Our Term Extractor can extract several objects of several types from a single infobox attribute-value pair. Since the majority of them are usually of the wrong type, we may not hope to find *all* of them in the English YAGO facts. Vice versa, the English YAGO facts may be more detailed than the foreign attribute facts. For example, they may know several albums of a singer, while the foreign Wikipedia may know only a few. Thus, we may not hope to find *all* English YAGO objects in the foreign Wikipedia. Therefore, we count as a *match* between a and r any subject that has a common object for a and r in F_a and E_r, respectively:

$$matches(F_a, E_r) = \pi_{subj}(F_a \cap E_r)$$

Clashes. If a YAGO relation r is a functional relation, and a foreign attribute a contains a different object for the same subject x, then a cannot map to r. We call x a *clash*. In practice, F_a will contain several objects of different types, and only few of them will actually match with the English YAGO. Therefore, we relax the definition of a clash as follows: A subject is a *clash* for a and r, if it has objects in F_a and in E_r, and if these objects are disjoint:

$$clashes(F_a, E_r) = \pi_{subj}(F_a) \cap \pi_{subj}(E_r) \setminus \pi_{subj}(F_a \cap E_r)$$

We call this definition of clashes and matches the *object set semantics*, because, for a given subject and a given relation, the objects are considered as a set. We look only at disjointness or non-empty intersections for the definition of clashes and matches.

Contributions. The foreign Wikipedias may bring entities that are unknown to the English YAGO. They may also bring facts that the English YAGO does not know. Thus, for a given foreign attribute, not every subject is a clash or a match. It may also just be a new fact. To quantify this phenomenon, we introduce the *contributions* of a foreign attribute a as the number of distinct subjects:

$$contrib(F_a) = \pi_{subj}(F_a)$$

The total number of facts that a contributes in the end may be larger than this number (if most subjects have several objects), or smaller (if most objects are removed by type checks).

Given F_a and E_r, our goal is to determine whether a maps to r. Several measures can be envisaged to this end.

Support. The *support* is simply the number of matches:

$$support(F_a, E_r) = |matches(F_a, E_r)|$$

This measure corresponds to the support in association rule mining [2]. It can be used to map an attribute to a relation if the number of subject-object pairs exceeds a certain threshold. This measure might be to restrictive for attributes with a small number of contributions. Vice versa, it may be misleading if there is a large number of contributions and a large number of spurious matches.

Confidence. The other measure of association rule mining is the *confidence*. In our setting, it corresponds to the ratio of matches out of the total number of contributions:

$$confidence(F_a, E_r) = |matches(F_a, E_r)| \times |contrib(F_a)|^{-1}$$

This measure is rather conservative, because it will punish mappings that have only few matches, and potentially many new facts that are unknown to the English YAGO.

PCA Confidence. The PCA-confidence [12] measures the number of matches out of the total number of matches and clashes:

$$pca(F_a, E_r) = |matches(F_a, E_r)| \times (|matches(F_a, E_r)| + |clashes(F_a, E_r)|)^{-1}$$

It was developed for association rule mining under the open world assumption, and is used in [11] to align relations across KBs. The PCA confidence admits that a fact that cannot be found in the English YAGO is not necessarily wrong. It is merely unknown. Therefore, the measure considers only facts that are confirmed or rejected explicitly (the matches and clashes), and ignores other facts.

Probabilistic Measure. Most pairs of a and r will exhibit some proportion of spurious matches. Before mapping a to r, we want to make sure that the proportion of matches exceeds that proportion of spurious matches, say 1%. The problem is that if our sample is small (say, 5 elements), then already one spurious match will fulfill that condition. Hence, we use a measure that models the mapping problem as a Bernoulli experiment. We observe that all non-English Wikipedias are smaller than the English Wikipedia. Therefore, we assume that F_a is only a subset of the future, yet-unknown set of infobox attribute facts F_a^* that the foreign Wikipedia will eventually comprise. We want to know the proportion of these infobox attribute facts that match the English YAGO facts with r. In particular, we want to know whether this proportion exceeds a threshold θ of spurious matches.

We make the following simplifying assumptions: We assume that F_a is a uniformly randomly drawn sample from F_a^*. We also assume that E_r is fixed. We want to estimate the proportion of matches, $confidence(F_a^*, E_r)$. In particular, we want to know whether $confidence(F^*a, E_r)$ exceeds θ. We do not have access to F_a^*, and cannot compute $confidence(F_a^*, E_r)$, but only $confidence(F_a, E_r)$. Thus, we aim to estimate a lower bound for a Bernoulli parameter from the observed parameter in the sample.

There are several ways to estimate a lower bound for a Bernoulli parameter. Here, we opt for the Wilson Score Interval [6], because it is particularly well-suited for small samples. The interval takes the form of a center c and a width δ. These values are computed from the size of the sample, $|F_a|$, and the confidence on the sample, $confidence(F_a, E_r)$. The interval guarantees that with a given probability (set a priori, usually to $\alpha = 95\%$), the value $confidence(F_a^*, E_r)$ falls into $[c - \delta, c + \delta]$. For small samples, the interval width δ will be very large. With growing sample size, δ shrinks and the center c converges towards $confidence(F_a, E_r)$.

The properties of the Wilson interval imply that $confidence(F_a^*, E_r) > c - \delta$ with $\alpha = 95\%$. Therefore, we define our measure for the matching of a to r as

$$wilson(F_a, E_r) := c - \delta$$

This measure allows us to judge whether the proportion of matches is large enough, even if the sample is small.
Mapping. Given any of the measures, $m \in \{\text{support, confidence, pca, wilson}\}$, and given a threshold $\theta \in \mathbb{R}$, we can define an approximate mapping of foreign infobox attributes to YAGO relations:

$$\widehat{map}(a) = \begin{cases} argmax_r \;\; m(F_a, E_r), & \text{if } max_r \;\; m(F_a, E_r) > \theta \\ \text{undefined}, & \text{else} \end{cases}$$

We designed an extractor for the YAGO system that performs this mapping, the *Attribute Matcher* (Figure 1). It takes as input a measure m and a threshold θ, and maps the foreign attributes in to YAGO relations. In the example, this will yield:

```
de/heirat hasTranslation marriedTo
```

These mappings are then used by an Attribute Mapper, just as for the English Wikipedia, to produce foreign YAGO facts from the attribute facts. In the example, we get:

```
Elvis marriedTo Priscilla_Presley
Elvis marriedTo "1967-##-##"^^xsd:date
Elvis marriedTo Las_Vegas
```

These facts will undergo the same checking and filtering as the other YAGO facts. A type check, e.g., will leave us with

```
Elvis marriedTo Priscilla_Presley
```

In this example, we just learned a fact that was in the English YAGO anyway. However, other foreign infobox attribute facts will give rise to new YAGO facts that were not in the English Wikipedia. Likewise, in the example, both Elvis and Priscilla were in the English YAGO. However, we extracted one million entities from the other Wikipedias that were not in the English YAGO. These give rise to new nodes in the YAGO knowledge base.
Further Processing. The facts will also undergo a funclash checking. We redesigned the funclash checking in such a way that preference is given to facts that were extracted from the infoboxes over facts from the categories. Within each group, preference is given to the English Wikipedia, followed by the other Wikipedias in our input list of languages. Within the infobox facts, preference is given to the first values. Within an infobox value string, preference is given to the left-most extracted value. This is just the order in which the Term Extractor produces the terms. We justify this choice by our manual analysis in Section 5.1.

4.3 Taxonomy Construction

In this section, we explain how we construct a unique taxonomy for YAGO. We first explain the existing architecture for the monolingual case before explaining our extension to the multilingual case.

English Extraction. The taxonomy of YAGO comes mainly from the categories of Wikipedia. Again, the process is driven by a sequence of extractors that each perform one particular transformation of data. The *Category Extractor* (not shown in the figure) extracts category memberships. It creates a theme that contains facts such as

```
Elvis inCategory "Rock Music"
Elvis inCategory "American Singers"
```

A subsequent extractor will filter out categories that do not correspond to class names (such as `Rock Music`, see [23]), and produce a theme that contains statements such as

```
Elvis type American_Singer
```

A follow-up extractor will use noun phrase parsing to find out that this class is most likely a subclass of the class `Singer` of WordNet, thus producing

```
American_Singer subclassOf Singer
```

Another extractor will transform the entire WordNet taxonomy into triples, which yields, e.g.,

```
Singer subclassOf Person
Person subclassOf LivingBeing
etc.
```

This way, the entity Elvis Presley is linked via the subclass of American Singers into the WordNet taxonomy.

Multilingual Extraction. In the English Wikipedia, the categories of articles are identified by the keyword *Category:*. Other languages use other keywords. To find out these keywords, we made use of Wikidata. For example, in our *Dictionary* from Wikidata, *Category: American singers* is mapped to the German equivalent *Kategorie: Amerikanische Sänger*. This tells us that the categories in the German Wikipedia are introduced by the keyword *Kategorie:*. We extracted all of these translations from Wikidata. We could then modify the existing category extractor of YAGO to extract category memberships also from foreign Wikipedias. For German, the *Category Extractor* extracts:

```
de/Elvis inCategory "de/Amerikanische Sänger"
```

A follow-up extractor uses the dictionary to translate these foreign entity and category names to their unique names:

```
Elvis inCategory "American singers"
```

From this point on, the standard YAGO extractors can do their work. The categories will be filtered and connected to the WordNet taxonomy.

Other Processing. Categories such as `1935 births` can also be a source of facts about the article entity. YAGO uses manually defined patterns to extract these facts. In the multilingual case, we first translate the categories to English through the dictionary, and then use the very same patterns to extract facts. If an infobox template (such as "singer") is used in an article, this can be indication that the article entity belongs to a particular class. Hence, we extract infobox templates in the same way as categories, and use them to create `type` facts.

4.4 Overall Integration

Taxonomy. Every entity is mapped to a language-independent unique identifier (Section 4.1), so that facts about the same entity will use the same entity identifier. Each Wikipedia article is in one or more categories. These categories are translated to their English counterparts, and then give rise to classes (Section 4.3). If we cannot establish a class for an entity, the entity is abandoned. Hence, every entity, foreign or English, is a member of at least one class. The classes are linked into the common WordNet taxonomy. All in all, this process creates a KB in which English entities and foreign entities live together in the same taxonomy.
Schema. All foreign language attributes have been either abandoned or mapped to one of the 77 English YAGO relations. This ensures that every fact has its place in the common schema. Final extractors will remove any duplicate facts, so that each fact in YAGO is unique, even if contributed from several sources.
Manual Work. We note that the only manual effort in the construction of YAGO3 is the mapping of English infobox attributes to YAGO relations – which exists already in the original YAGO. The foreign infobox attributes are mapped automatically to YAGO relations. The translation of entities and categories is done with the data from Wikidata.

5 Experiments

We ran the YAGO extraction system on 10 languages: English, German, French, Dutch, Italian, Spanish, Romanian, Polish, Arabic, and Farsi. The choice of languages was determined by the proficiency of the authors, so as to facilitate manual evaluation. We cover some of the largest Wikipedias, and both European and non-European languages.

5.1 Funclashes

As discussed in Section 4.2.1, funclashes occur when a fact candidate $\langle x, r, y \rangle$ with a functional relation r encounters a YAGO fact $\langle x, r, y' \rangle$ with $y \neq y'$. Such clashes are often used to spot contradictions in semantic data. We wanted to know whether funclashes have this role in our setting, too. In our setting, facts are collected from the Wikipedias in the given order of languages (Section 4.2.2). Whenever an incoming fact clashes with an existing fact from a higher ranked language, the incoming fact is discarded. Table 1 shows the relations that produce most funclashes. The vast majority of clashes stem from date relations, followed by numeric relations.

Funclashes	Relation
552,693	wasCreatedOnDate
63,473	diedOnDate
50,588	wasBornOnDate
18,437	wasBornIn
15,927	happenedOnDate
12,308	hasHeight
11,185	hasDuration
9,980	hasWeight
4,300	diedIn
3,418	wasDestroyedOnDate

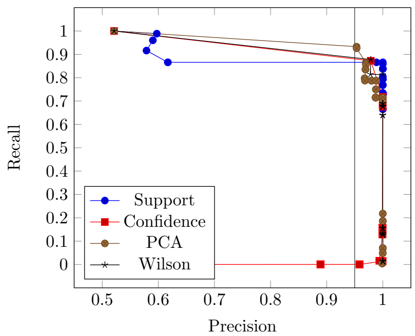
Table 1: Funclashes per relation

We were interested in whether funclashes are really an indication for contradictions. Therefore, we sampled a set of 100 funclashes randomly, and analyzed them manually. In only 5 cases, the rejected object was clearly more accurate than the existing object. The other cases were as follows: (1) A Wikipedia article describes several variants of the entity with different properties. For example, a movie may exist in original and abridged form, where the latter has a shorter duration. (2) In nearly half of the cases, the rejected object was wrong or less accurate than the existing object. This is mainly because Wikipedia often contains complementary information in the infobox value string after the targeted value. For example, numerical attributes (such as page numbers or population numbers) are often followed by years in the infobox attribute value. The funclash gives preference to the first number and thus discards the (incorrect) second number. (3) In one forth of the cases, different values can be justified. For example, numerical values in different units give rise to slightly different objects. The year of foundation of a city can be either when it was first populated, or when it was incorporated. The height of a tower may or may not include the antenna. In these cases, the funclash just makes an arbitrary choice. It would be confusing for applications if a tower had two height values, because the additional textual information that is present in Wikipedia is projected away in the KB.

While we could justify abandoning the functional constraints on this basis, they are invaluable to avoid mixing up different versions of the same entity (case (1)), to avoid extracting years as page numbers (case (2)), and to ensure overall consistency (case (3)). Therefore, we decided to keep functional constraints. At the same time, they are not always an indication for semantic inconsistencies, which makes them less useful for attribute mapping, as we shall see.

5.2 Choice of Parameters

Gold Standard. We wanted to measure the precision and recall of the mapping of infobox attributes to YAGO relations under different measures and thresholds (Section 4.2.2). We created a near-exhaustive mapping, which maps every



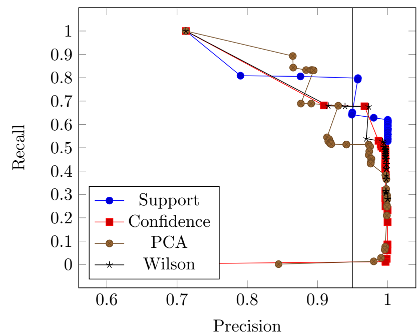


Figure 2: Precision/Recall for French (left) and Farsi (right)

infobox attribute a of a particular Wikipedia edition l to every YAGO relation r with $matches(F_a, E_r) > 0$. This mapping is arguably a superset of the desired mapping. For every language l, we randomly sampled 150 attribute-relation pairs from this mapping, and evaluated them manually. A pair $\langle a, r\rangle$ was evaluated to true, if the attribute a implies that every entity of type $ran(r)$ in its value will yield an r fact. Thus, our manual gold standard will map the German attribute `geboren` (*born*) to both `bornInPlace` and `bornOnDate`, because the attribute value can contain both a birth place and a birth date.

Evaluation. We produced attribute mappings by each of the measures from Section 4.2, for different threshold values, and for all languages under consideration. The threshold θ was varied between 0 and 1 for the confidences, between 0 and 500 for support, and between 0 and 50% for the Wilson score. Values outside these ranges decreased recall without increasing precision. By varying θ, we can trade off precision and recall. Figure 2 exemplifies this for French and Farsi. Since YAGO has an accuracy of 95% [23], we have to choose a threshold that achieves at least this precision.

We find that only for the Wilson score and the confidence there exist thresholds that achieve a precision of 95% and a recall of more than 5% across all languages. This is because the support is not robust to scaling: languages with many facts (such as German) need a high threshold, while languages with few facts (such as Spanish) produce no mappings if the threshold is too high. The PCA confidence is misguided by the clashes, and thus behaves in a very erratic way across languages. The Wilson score consistently achieves the highest recall across all languages, at comparable precision to the confidence (Table 2). This is because in order to achieve a high precision, the confidence has to use a high threshold. This, in turn, hurts recall. The Wilson score, in contrast, can stay with a low threshold and thus a high recall, because it cautions automatically against too small sample sizes. Hence, we chose the Wilson score as our measure. To achieve a precision of 95%, we chose $\theta = 4\%$.

Discussion. Most erroneous mappings come from two sources. First, unit-less numbers (such as the number of inhabitants) produce a disproportionally high number of spurious matches, coinciding, e.g., with longitude values or years.

	Confidence 16%			Wilson 4%		
	Prec	Rec	F1	Prec	Rec	F1
ar	100	73	85	100	82	90
de	100	37	54	98	56	72
es	96	19	32	95	29	45
fa	100	49	66	97	54	69
fr	100	16	27	100	69	82
it	100	7	12	98	23	37
nl	100	19	32	100	22	36
pl	95	10	19	97	64	77
ro	96	52	67	95	70	81

Table 2: Precision and Recall per Language in %

Second, there are a number of relations that are strongly correlated in life, but that are strictly speaking not matches (such as `wasBornIn`/`isCitizenOf` or `wasBuriedIn`/`diedIn`), and so we did not count them as such. Still, we achieve very good precision and recall values overall.

If the Wikipedias of two different languages share no knowledge, then our method will not propose any mapping. However, it is improbable that this happens. There has been a rapid globalization of Wikipedia, which triggered thousands of volunteers to create, modify, and link millions of articles. Together with manual translation of articles, this led to a substantial overlap of the Wikipedias in a lot of common concepts such as countries, important people, and prominent events.

Our KB is built on YAGO, and tries to match foreign attributes to YAGO relations. Our method does not consider attributes that have no mapping to the existing YAGO relations. We leave the introduction of new YAGO relations for future work.

Comparison. With respect to attribute matching, the work of [21] reports a recall of 85% at the precision of 95% that is required for YAGO. This is a combination that our method cannot achieve. The focus in YAGO3, however, is on broad language coverage. Our method achieves an extraordinary weighted precision of 98% at a recall of 56% for German, a precision of 99.98% at a recall of 69% for French, and a precision of 99.99% at a recall of 22% for Dutch. These are the languages that [21] considered. In addition, our system produces alignments for Italian, which the method of [21] could not run on due to poor infobox naming conventions, and 5 other languages. These include languages of non-Latin script, which [21] explicitly excludes. Our method thus provides a robust alternative to [21] with much higher language coverage.

[1] solves the slightly different problem of merging infoboxes across Wikipedia languages. They report a precision of 86% for English, Spanish, French, and German. Our method compares favorably with this precision. Beyond that, we can show that it works for 6 more languages.

[26] aligns the English Wikipedia with 2 Chinese online encyclopedias. They

Language	Entities	Facts	Type Facts	Labels
en	3,420,126	6,587,175	10,280,369	477,628
de	349,352	984,830	2,563,246	125,575
fr	255,063	549,321	920,014	361,932
nl	204,566	249,905	398,719	208,521
it	67,330	148,268	160,777	1,424
es	118,467	43,271	512,024	213
ro	11,024	12,871	44,175	946
pl	103,440	235,357	296,398	215,470
ar	50,295	98,285	314,495	2,575
fa	16,243	27,041	121,492	4,553
total	4,595,906	8,936,324	15,611,709	1,398,837

Table 3: Number of entities and facts

report a precision of 86% at a recall of 88%. Their work concerns non-Wikipedia data sources, and so we cannot compare directly to them.

[20] aligns infobox attributes between the English and the Portuguese Wikipedia, and the English and the Vietnamese Wikipedia. For Portuguese, the average weighted precision on 14 infobox templates is 93%, and the recall is 75%. For Vietnamese, the values are 100% and 75%, respectively. These values are comparable to ours.

Different from all of these approaches [20, 1, 21], our alignment method is considerably simpler. It requires no similarity functions or machine learning methods – while achieving comparable results. This is because we can build on the existing YAGO infrastructure which is able to yield a high-precision KB. We also show that our method is robust enough to treat twice as many languages as previous work. Finally, our method goes beyond previous work by constructing a unified KB on top of these sources, which includes `type` facts and a taxonomy. We illustrate this in the next section.

5.3 Size

Facts. Table 3 shows the total number of distinct new entities that each language contributed to our unified KB. Every entity is counted only for the first language in our preference list in which it appears, even if it is contributed by several other languages as well. In total, we gain 1m new entities. The number of new entities does not scale linearly with the number of languages, because most languages treat the same global concepts before venturing into local entities. The table also shows the number of facts contributed by every language. Again, every fact is counted only once. The “facts” column shows ordinary facts extracted from the infoboxes and categories. We gain 2.5m facts over the English-only extraction. The next column shows `type` facts extracted from the template names and the categories. We gain 5m new facts about types. Not all

de/Kirdorf_(Bedburg), hasNumberOfPeople, "1204"^^xsd:integer
fr/Château_de_Montcony, isLocatedIn, Burgundy
pl/Henryk_Pietras, wasBornIn de/Debiensko
fa/امیر خان امیراعلم , wasBornIn, Teheran

Table 4: Some sample facts

of these are necessarily about the new entities that a language contributed; a language can also contribute a type fact about an English entity. The last column shows the `label` facts. We gain 1m labels. This number is complemented by 355k labels that we extracted from the English disambiguation pages, 11m labels that come from the redirect pages, 1.5m person names (given name and family name), and 729k new labels from Wikidata, and 1.5m facts extracted by other extractors from the English Wikipedia, bringing the total number of all facts to 40m.

Examples. Our facts include truly hybrid facts, where the subject or object do not exist in the English Wikipedia. Table 4 shows some examples. In the first example, the subject exists only in the German Wikipedia. In the second example, the subject exists only in French, but the object is contributed by English. In the third example, neither the subject nor the object exists in English. The subject exists only in Polish, and the object exists in German and Polish (and German is chosen over Polish). We also have a large number of facts in non-Latin script. We show a fact about Amir Alam, a former member of the Iranian parliament.

Schema. All foreign language attributes have been either abandoned or mapped to one of the 77 English YAGO relations (see Section 4.2.2). Every entity has at least one `type` fact (see Section 4.3). The types are connected to the common WordNet taxonomy. This ensures that every entity has its place in the common schema. All in all, YAGO contains 488,469 classes.

6 Applications

6.1 DBpedia

Multilingual DBpedia. Our method can also be used to align other KBs. We downloaded the version of the English DBpedia that contains ontological relations. We used our methodology to align it with the foreign attribute facts of German. As in the other scenarios, we produced a gold standard and evaluated the precision and recall. Our vanilla setting of the Wilson score interval with $\theta = 4\%$ achieves a precision of 95% and a recall of 81%. The values are not as high as for the YAGO-internal alignments, because DBpedia uses different data types (such as `xsd:gYear`) that our term extractor does not produce.

On the other hand, our system was run off the shelf, and DBpedia-specific adjustments could increase the performance further. For example, by using the support measure with a dataset-specific threshold of 100, we achieve a precision of 96% and a recall of 94%. Thus, our methodology could help map the infobox attributes of different languages to the common DBpedia properties – a task that is currently achieved manually by the community.

Comparison. Our method can also be used to align the relations of DBpedia and YAGO. We took again the English DBpedia with ontological relations, and matched them with the YAGO facts. We generated a gold standard, and evaluated the mappings. Our vanilla setting of a Wilson score threshold of $\theta = 4\%$ achieves a weighted precision of 100% and a weighted recall of 76%. This is in line with the weighted precision of 100% that [22] reports for this alignment, while they report no recall.

6.2 Le Monde

In [14], the authors analyze the text of the French newspaper *Le Monde* by mapping all mentions of entities in the text to YAGO entities. This allows, e.g., statistics on the prevalence of foreign companies in different countries, or an analysis of the changing age structure in certain professions over time. Since YAGO was not available in French, the approach could use only those YAGO entities that had a French name and that had enough facts in the English Wikipedia, resulting in 3.4m mentions. With YAGO3, we can boost that number by 824k new mentions, referring to 32k unique new entities. 112k of the mentions are people, and of these, 20k are politicians and 8k are musicians. These can contribute more data points to the statistics. For example, we can add 5 more countries to the analysis of the prevalence of foreign companies, because we now have enough data for these countries.

7 Conclusion

In this paper, we have shown how to construct the first full-fledged KB from Wikipedias in multiple languages. By using YAGO as a central reference KB, and by smartly extending its existing architecture, we arrive at a simple and elegant, yet very effective method. Our approach works across 10 languages and different scripts, and achieves a precision of 95%-100% in the attribute mapping. The new KB gains 1m new entities and 7m new facts over the English-only YAGO.

On the technical side, we have compared several measures for the mapping of infobox attributes, and presented a measure that is robust to scale and language. We have shown that our measure can be applied to different relation alignment problems at high precision and recall. For future work, we envisage extending our methods to more languages, and study new applications of the knowledge base.

References

[1] E. Adar, M. Skinner, and D. S. Weld. Information arbitrage across multi-lingual wikipedia. In *WSDM*, 2009.

[2] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2), June 1993.

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

[4] J. Biega, E. Kuzey, and F. M. Suchanek. Inside yago2s: A transparent information extraction architecture. In *Proc. WWW (Comp.)*, 2013.

[5] G. Bouma, S. Duarte, and Z. Islam. Cross-lingual alignment and completion of wikipedia templates. In *Workshop on Cross Lingual Information Access*, 2009.

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

[7] G. de Melo and G. Weikum. MENTA: Inducing multilingual taxonomies from Wikipedia. In *CIKM*, 2010.

[8] G. de Melo and G. Weikum. Untangling the cross-lingual link structure of wikipedia. In *ACL*, 2010.

[9] G. de Melo and G. Weikum. UWN: A large multilingual lexical knowledge base. In *ACL*, 2012.

[10] G. de Melo and G. Weikum. Taxonomic data integration from multilingual wikipedia editions. *Knowl. Inf. Syst.*, 39(1):1–39, 2014.

[11] L. Galárraga, N. Preda, and F. M. Suchanek. Mining rules to align knowledge bases. In *AKBC*, 2013.

[12] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Amie: association rule mining under incomplete evidence. In *WWW*, 2013.

[13] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194, 2013.

[14] T. Huet, J. Biega, and F. M. Suchanek. Mining history with le monde. In *AKBC*, 2013.

[15] O. Medelyan, D. Milne, C. Legg, and I. H. Witten. Mining meaning from wikipedia. *Int. J. Hum.-Comput. Stud.*, 67(9):716–754, Sept. 2009.

[16] G. Miller. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

[17] V. Nastase and M. Strube. Transforming wikipedia into a large scale multilingual concept network. *Artif. Intell.*, 194, 2013.

[18] V. Nastase, M. Strube, B. Boerschinger, C. Zirn, and A. Elghafari. Wikinet: A very large scale multi-lingual concept network. In *LREC*, 2010.

[19] R. Navigli and S. P. Ponzetto. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.

[20] T. H. Nguyen, V. Moreira, H. Nguyen, H. Nguyen, and J. Freire. Multilingual schema matching for wikipedia infoboxes. *PVLDB*, 5(2), 2011.

[21] D. Rinser, D. Lange, and F. Naumann. Cross-lingual entity matching and infobox alignment in wikipedia. *Inf. Syst.*, 38(6):887–907, Sep. 2013.

[22] F. M. Suchanek, S. Abiteboul, and P. Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *PVLDB*, 5(3), 2011.

[23] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A core of semantic knowledge - unifying WordNet and Wikipedia. In *WWW*, 2007.

[24] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO - A Large Ontology from Wikipedia and WordNet. *J. of Web Semantics*, 6(3), September 2008.

[25] E. Tacchini, A. Schultz, and C. Bizer. Experiments with wikipedia cross-language data fusion. In *Workshop on Scripting and Development*, 2009.

[26] Z. Wang, J. Li, Z. Wang, and J. Tang. Cross-lingual knowledge linking across wiki knowledge bases. In *WWW*, 2012.

[27] D. T. Wijaya, P. P. Talukdar, and T. M. Mitchell. Pidgin: ontology alignment using web text as interlingua. In *CIKM*, 2013.

Dépôt légal : 2014 – 3e trimestre
Imprimé à Télécom ParisTech – Paris
ISSN 0751-1345 ENST D (Paris) (France 1983-9999)

Télécom ParisTech

Institut Mines-Télécom - membre de ParisTech

46, rue Barrault - 75634 Paris Cedex 13 - Tél. + 33 (0)1 45 81 77 77 - www.telecom-paristech.fr

Département INFRES