Heuristics for Connecting Heterogeneous Knowledge via FrameBase

Jacobo Rouces[1], Gerard de Melo[2], and Katja Hose[1]

[1] Aalborg University, Denmark
jrg@es.aau.dk, khose@cs.aau.dk
[2] Tsinghua University, China
gdm@demelo.org

Abstract. With recent advances in information extraction techniques, various large-scale knowledge bases covering a broad range of knowledge have become publicly available. As no single knowledge base covers all information, many applications require access to integrated knowledge from multiple knowledge bases. Achieving this, however, is challenging due to differences in knowledge representation. To address this problem, this paper proposes to use linguistic frames as a common representation and maps heterogeneous knowledge bases to the FrameBase schema, which is formed by a large inventory of these frames. We develop several methods to create complex mappings from external knowledge bases to this schema, using text similarity measures, machine learning, and different heuristics. We test them with different widely used large-scale knowledge bases, YAGO2s, Freebase and WikiData. The resulting integrated knowledge can then be queried in a homogeneous way.

1 Introduction

In the past decades, numerous large-scale knowledge bases (KBs) have become available and are now essential both in research and in the commercial world, e.g., for IBM's Jeopardy!-winning question answering system Watson [16] and for Google's Knowledge Graph-driven search results. The Web of Linked Data has grown to the point that the numerous different KBs that have been published can no longer easily be visualized in a single cloud image.

Since numerous stakeholders are publishing separate KBs focusing on different domains and sources, a given application often needs to combine knowledge from multiple KBs. Hence, there is a clear need for methods to *integrate* such knowledge. A substantial body of work has aimed to address this problem by automatically aligning individual entries across KBs, both at the schema level [9] and at the level of entity instances [6]. These methods often produce a list of binary links using properties such as `owl:sameAs`. Unfortunately, different KBs often model the world in quite distinct ways. Despite the adoption of standards such as the use of subject-predicate-object triples in RDF [15], the same piece of information can be represented in ways such that a one-to-one alignment is no longer possible.

Consider, for instance, a marriage between two people. The YAGO KB [25] captures this using a binary predicate (`isMarriedTo`) between two persons.

The Freebase KB [1], in contrast, relies on a special entity called a mediator or Compound Value Type (CVT) to describe the marriage, as well as several subject-predicate-object triples to list properties of the marriage, such as involved people, time, location, etc. In cases like this, which are not uncommon, neither `owl:sameAs`, `rdfs:subClassOf`, `owl:equivalentProperty`, nor any other individual property or binary relation can fully express the complex n-ary relationships between these resources.

In this paper, we propose to address this problem by integrating heterogeneous data into the FrameBase schema [21], which consists of a large inventory of *frames* that homogeneously represent n-ary relations. Frame structures are used in linguistics to describe the meaning of a sentence as scenarios with multiple participants and properties filling specific semantic roles. A marriage frame involves two partners, a time and a place, among other things. This is similar to Freebase's CVTs. However, in contrast to the few hundreds of CVTs in Freebase, FrameBase uses a larger number of frames (∼20,000) organized in a dense hierarchy [11].

While FrameBase offers a flexible system for representing knowledge from existing knowledge sources [21, 22], there has not been any research showing how to automatically or semi-automatically integrate heterogeneous knowledge under its schema. In this paper, we develop a generic algorithm to create complex integration rules from external KBs into this schema. These rules go beyond existing alignment mechanisms designed for binary mappings between elements of different KBs. In our experiments, we show results on three particularly heterogeneous sources: Freebase [1] and WikiData [8] are KBs with an especially large schema. YAGO2s [25], in contrast, uses only a small number of properties, but relies heavily on reification to describe phenomena such as time and locations.

2 Related Work

Connecting knowledge sources is a long-standing problem. At the level of individual records in databases, this has variously been addressed as record linkage, entity resolution, and data de-duplication [7]. In KBs, this roughly corresponds to the problems of ontology alignment, data linking [26], and instance matching [6].

For KBs, there has been substantial work on ontology alignment [9] to identify matching classes from different sources, and in some cases also instances and properties across different sources [24]. A closely related task is that of canonicalizing or reconciliating knowledge from open information extraction [12, 18], which focuses on aligning names of entities and predicates by clustering synonymous entries. To achieve this, the knowledge extracted from each text source has to be reconciled, sometimes using complex graph matching algorithms [18]. But as the same extraction tool is used for each text source, the resulting graphs are constructed in similar ways and therefore follow a common model. Hence, the applied techniques for reconciliation are different from the ones necessary to reconcile ontologies created by completely independent parties and tools.

Only very little work has considered scenarios in which the same type of ontological knowledge is modeled in entirely different ways. In these cases, align-

ment by means of binary properties such as equivalence or subsumption is no longer sufficient, because a KB may not have a direct counterpart for an element of another KB. The EDOAL (Expressive and Declarative Ontology Alignment Language) format [3] has been proposed to express complex relationships between properties. It defines a way to describe complex correspondences but it does not address how to create them. Similarly, complex correspondence patterns between ontologies – or ontologies and databases – have been described and classified in an ontology [23]. However, this approach does not provide any method to create the correspondence patterns, neither fully nor semi-automatically. The iMAP tool [4] explores a search space of possible complex relationships between the values of entries in two databases, e.g., `room-price = room-rate * (1 + tax-rate)`, but these are limited to specific types of attribute combinations. The S-Match tool [13] uses formal ontological reasoning to prove possible matches between ontology classes, involving union and intersection operators, but does not address complex matching of properties beyond this. Ritze et al. [20] use a rule-based approach to detect specific kinds of complex alignment patterns between entries in small ontologies.

Unlike previous work, the approach presented in this paper does not focus on matching individual entities but provides techniques to match knowledge that can also be expressed with complex patterns involving multiple entities.

3 Frames for Data Integration

FrameBase [21] relies on the concept of linguistic frames as provided by FrameNet [11]. Such frames represent events or situations with characteristics denoted as Frame Elements (FEs). As FrameNet's original purpose is semantic annotation of natural language, many frames have associated Lexical Units (LUs), i.e., terms that, when appearing in a text, may evoke a frame, which may be connected via FEs to some other parts of the text.

FrameBase represents the information about "*John's 7-year marriage to Mary*" by creating an entity e that is an instance of FrameNet's `Personal_relationship` frame (or a more specific one for marriages, as we describe later on). Relevant FEs such as the marriage partners and the duration are then captured by adding triples with e as subject. For instance, properties `Partner_1` and `Partner_2` connect e to entities representing John and Mary, respectively, while the property `Duration` is used for the time their marriage lasted.

FrameBase thus repurposes FrameNet frames, originally intended to represent natural language semantics, for knowledge representation with subject-predicate-object triples, using what is also called *neo-Davidsonian representation*: One first introduces an entity e that is an instance of a frame class, and hence represents a particular event or situation. This entity is then connected to other entities (for example other frame instances, literals, or named entities) by means of properties representing the frame elements.

To adapt FrameNet for knowledge representation, FrameBase extends the inventory of frames defined by FrameNet in a hierarchy consisting of the following levels (Figure 1):

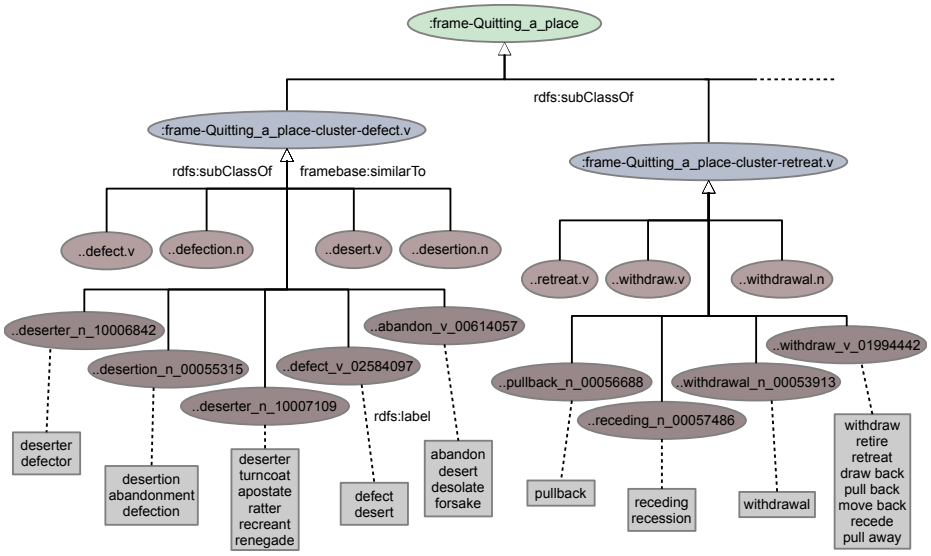

Fig. 1: Example of a hierarchy with a macroframe `:frame-Quitting_a_place`, two cluster-microframes that are direct subclasses of the macroframe, and several LU- and synset-microframes that are direct subclasses of the cluster-microframe. All the microframes under a given synset-microframe are also connected via the symmetric property `framebase:similarTo` (for clarity, the transitive closure is omitted). The synset-microframes also have labels extracted from WordNet. The microframe identifiers have a shared prefix that has been abbreviated.

- *Macroframes* are very coarse-grained and correspond to regular frames in FrameNet. The `Personal_relationship` frame class, for example, subsumes `spouse`, `marriage`, `girlfriend`, and `divorced`.
- *Microframes* inherit the general semantics and FE properties from their parent macroframes. They can be classified into 3 types:
  1. *LU-micoframes* are based on a frame's LUs and are represented as subframes in FrameNet, and therefore as subclasses in FrameBase. `Personal_relationship-married.a` and `Personal_relationship-divorced.a`, for example, are subclasses of `Personal_relationship`.
  2. *Synset-microframes* are created for synsets (sense-disambiguated synonymous words) in WordNet [10] that LUs can be mapped to. For instance, the two LU-microframes `Personal_relationship-suitor.n` and ... `Personal_relationship-court.v` are connected to each other by means of synset-microframes.
  3. *Cluster-microframes* are created to cluster sets of LU- and synset-microframes with similar meaning. `Personal_relationship`, for instance, clusters (encoded as subclasses) `Personal_relationship-married.a`, `Personal_relationship-divorced.a`, `Personal_relationship-suitor.n`, and `Personal_relationship-court.v`.

To enable more efficient querying without involving frame instances, FrameBase also provides Direct Binary Predicates (DBPs) that directly connect pairs of FEs. For instance, the two partners involved in a marriage are directly connected by a triple with `marriedTo` as property. The schema provides both reification and dereification (ReDer) rules to convert knowledge between the two representations (frame and DBPs). Two example ReDer rules are presented in Figure 2.

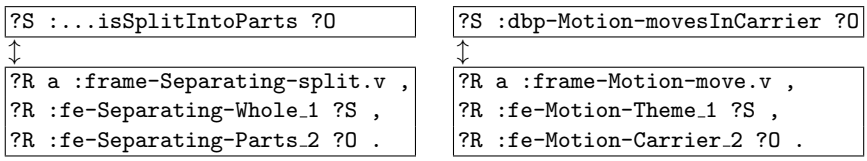

Fig. 2: Two example ReDer rules. The direct binary predicate is the property in the dereified pattern, on the top. The reified pattern is at the bottom.

Overall, the FrameBase RDFS schema currently contains 19,376 frames, including 11,939 frames for specific lexical units and 6,418 frames for WordNet's sets of synonyms. In addition to ReDer rules, the schema uses efficient RDFS+ inference (RDFS extended with a transitive, symmetrical, and reciprocal property used to link elements of a cluster).

4 Knowledge Base Integration

We now outline our approach for integrating heterogeneous knowledge bases using the FrameBase schema. Although the techniques can be applied to a wide range of KBs, we focus in particular on YAGO2s [25], Freebase [1], and WikiData [8].

Our integration algorithm produces integration rules describing how to transform knowledge from a KB into FrameBase. These rules do not connect individual instances but are defined at the schema level and therefore resemble Global-As-View mappings in relational database systems [5]. Formally speaking, the produced integration rules can be expressed in first-order logic – with triples represented as 3-ary predicates (Figure 3). Nevertheless, we implement these rules using SPARQL CONSTRUCT queries [14] because SPARQL is a widely supported standard for KBs available in RDF format. Non-RDF KBs can also be integrated by either using an alternative rule formalism or invoking off-the-shelf or custom-purpose RDF converters[3].

Algorithm 1 sketches our approach, which relies on three mapping functions that are discussed in Section 4.3 and three rule instantiation functions given in Figure 3. The mapping functions relate entities from the source KB with entities from FrameBase into which they can be translated, but they do not provide the structure of the integration rules. The structure is specified by the instantiation functions, which take elements from the source KB and FrameBase, and return structured integration rules.

[3] `http://www.w3.org/wiki/ConverterToRdf`

Algorithm 1 FrameBase Integration Algorithm.

Require: K ▷ input knowledge base
1: $R \leftarrow \emptyset$ ▷ set of SPARQL CONSTRUCT rules
2: **for all** classes C in K **do** ▷ create class-frame rules
3: **for all** frames $F \in \text{mappings}_{\text{C-F}}(C)$ **do**
4: $M \leftarrow \emptyset$ ▷ property mappings
5: **for all** properties P such that $\exists\ s, o:\ \langle s\ P\ o\rangle \in K$ **and** $s \in C$ **do**
6: **for all** $E \in \text{mappings}_{\text{PF-E}}(P, F)$: E is not in R **do**
7: $M \leftarrow M\ \cup (P, E)$
8: $R \leftarrow R \cup \text{ClassFrameRule}(C, F, M)$
9: **for all** properties P in K **do** ▷ create core property-frame rules
10: **if** the domain of P is not `rdf:Statement` **then**
11: **for all** $(F, E_s, E_o) \in \text{mappings}_{\text{P-FEE}}(P)$ **do**
12: $R \leftarrow R \cup \text{PropertyFrameRule}(P, F, E_s, E_o)$
13: **for all** properties P' in K **do** ▷ extend property-frame rules
14: **if** the domain(P')=`rdf:Statement` **then**
15: **for all** properties P in K satisfying $\langle$`P ^rdf:property/P' y`$\rangle$ **do**
16: **for all** property-frame rules r in R **do**
17: **if** r matches $\text{PropertyFrameRule}(P, F, E_s, E_o)$ **then**
18: **for all** frame elements $E_{P'} \in \text{mappings}_{\text{PF-E}}(P', F)$ **do**
19: $\text{Extend}(r, P', E_{P'})$
20: return R ▷ final set of integration rules

The instantiation functions are used to create two kinds of integration rules: (i) class-frame rules, which convert classes and properties from the original KB into similar elements in FrameBase (Section 4.1) and (ii) property-frame rules, which convert properties from the source KB into frames (Section 4.2).

4.1 Class-Frame Rules

The process of creating class-frame rules starts in line 2 in Algorithm 1, relying on mapping functions $\text{mappings}_{\text{C-F}}$ and $\text{mappings}_{\text{PF-E}}$. Class-frame rules are produced by the rule instantiation function $\text{ClassFrameRule}(C, F, M)$ from Figure 3. They convert a class C into a frame F that represents an event, situation or state of affairs, given $M = \{(P_1, E_1), ...(P_n, E_n)\}$ maping properties P_i for C to frame elements E_i of F. Figure 4 provides an example of a class-frame rule automatically generated for integrating Freebase.

4.2 Property-Frame Rules

In general, the purpose of a property-frame rule is to translate a property in a source KB as an instance of a frame with at least two properties. These rules are built in two steps.

Creation of core property-frame rules. The process of creating core property-frame rules starts in line 9 in Algorithm 1, relying on the mapping

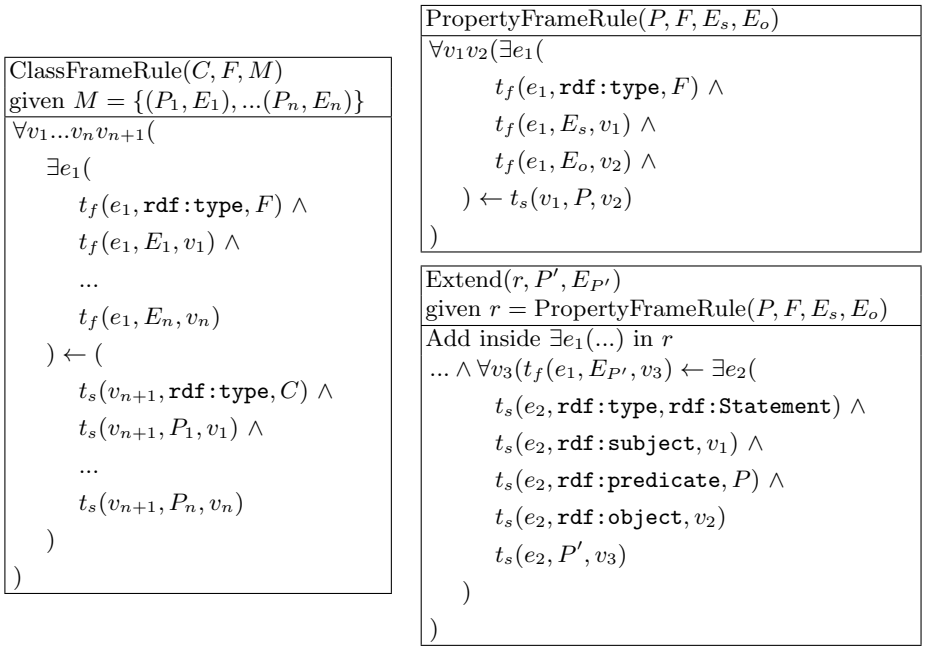

Fig. 3: Instantiation functions for the integration rules used by Algorithm 1. $t_s(s, p, o)$ stands for a triple in a source KB and $t_f(s, p, o)$ for a triple in FrameBase. v_i and e_i are variables (universally and existentially quantified, respectively) over entities in the source KB.

function mappings$_{\mathrm{P-FEE}}$. Core property-frame rules are produced by the instantiation function PropertyFrameRule(P, F, E_s, E_o) from Figure 3. Each RDF triple in the source KB matching pattern `?x` P `?y`, is transformed into a frame instance of type F with two frame-element properties E_s and E_o whose values are `?x` and `?y`, respectively. Figure 5 provides an example of a core property-frame rule automatically generated for integrating Wikidata.

Extending core property-frame rules to capture RDF reification. Additional clauses may be added by Algorithm 1 in the loop starting in line 13. This process relies on the mapping function mappings$_{\mathrm{PF-E}}$. It uses the instantiation function Extend(r, P', E) from Figure 3, which takes a property-frame rule r = PropertyFrameRule(P, F, E_s, E_o) as argument and returns an extended version of it to capture knowledge attached to triples by means of RDF reification [21]. KBs such as YAGO use this to represent n-ary relationships, but the FrameBase model is more efficient for this purpose. Figure 6 provides an example of an extended property-frame rule generated for integrating YAGO.

4.3 Mapping Functions

The mapping functions use an automatic general technique meant to be used with big and dynamic source KBs, extended with heuristics that apply for common patterns across large source KBs or cover most small source KBs.

```
CONSTRUCT {
  _:e a :frame-Win_prize-win.v    ; :fe-Win_prize-Time ?y
  ; :fe-Win_prize-Prize ?a        ; :fe-Win_prize-Competitor ?aw
  ; :fe-Win_prize-Explanation ?hf ; :fe-Win_prize-Competition ?c
  ; :fe-Win_prize-Rank ?al        ; :fe-Win_prize-Event_description ?ed .
} WHERE {
  ?m a fb:award.award_honor .
  OPTIONAL { ?m fb:award.award_honor.year ?y }
   ...honor.award ?a }                  ...honor.award_winner ?aw }
   ...honor.honored_for ?hf }           ...honor.ceremony ?c }
   ...honor.achievement_level ?al }     ...honor.notes_description ?ed } }
```

Fig. 4: Class-Frame rule, automatically generated rule for integrating Freebase.

```
#SOURCE_PROPERTY_NAME='depicts'
#SOURCE_PROPERTY_DESCR='depicted person, place, object or event'
CONSTRUCT {
 _:r a :frame-Communicate_categorization-depict.v .
 _:r :fe-Communicate_categorization-Speaker ?S .
 _:r :fe-Communicate_categorization-Item ?O .
} WHERE { ?S <http://www.wikidata.org/entity/P180> ?O }
```

Fig. 5: Property-Frame rule, automatically generated for integrating Wikidata.

P-FEE Mapping Function Given property P from the source KB, $\text{mappings}_{\text{P-FEE}}(P)$ returns 3-tuples of a frame F, and frame element properties E_s, E_o associated with P. Informally, it means that property `P` from the source KB can be substituted with a path $\hat{}E_s/E_o$ in FrameBase.

General Method. For the general variant of $\text{mappings}_{\text{P-FEE}}(P, F)$, we exploit the fact that the direct binary predicates built into FrameBase, which allow us to directly connect two frame elements, are directly mappable to external properties that should evoke a frame and two frame elements. Since the direct binary predicates were created with labels that follow the prevailing conventions in other LOD KBs [21], we can use a text similarity measure to find equivalent direct binary predicates, and for those found, use the frame and FEs in the

```
CONSTRUCT { _:event a :frame-Ride_vehicle-flight.n        core
  ; :fe-Ride_vehicle-Source ?s ; :fe-Ride_vehicle-Goal ?o core
  ; :fe-Ride_vehicle-Vehicle ?objTransp .                 extension
} WHERE { ?s yago:isConnectedTo ?o .                      core
  OPTIONAL { ?sid rdf:type rdf:Statement .                extension
    ?sid rdf:subject ?s ; rdf:object ?o                   extension
      ; rdf:predicate yago:isConnectedTo .                extension
    OPTIONAL { ?sid yago:byTransport ?objTransp }}}       extension
```

Fig. 6: Extended property-frame rule, generated for integrating YAGO.

associated reification rule. For example, if a property in a source KB is named "is split in", it turns out to be similar to the direct binary property "is split into parts" from the first example in Figure 2, which can be used to create an integration rule that translates that source KB property into the reified pattern of FrameBase's ReDer rule.

To compare direct binary predicates with external ones, the text similarity we use is cosine distance of bag-of-words vectors. We split predicate names into tokens using capitalization, use proper lemmatization (with Stanford CoreNLP 3.6.0 [17]) instead of stemming, and do not filter stop-words, since in this case certain closed-set parts of speech such as prepositions are very important. The use of this measure significantly improved the results compared to using ADW [19], arguably because the latter is not tuned for our kind of text. Besides, our method was much faster.

For each external KB property, we run the similarity function against all existing DBPs in FrameBase, and we take the best candidate if it has a score higher than a threshold of 0.8. The threshold value was chosen empirically to balance precision and recall.

Additional Heuristics. Our system admits manually crafted heuristics to be added to $\text{mappings}_{\text{P}-\text{FEE}}(P, F)$. When one of the heuristics fire, they take preference over the general method. The vast majority of datasets in the Linked Open Data cloud rely on very small hand-crafted ontologies and vocabularies. In this case, relying on the heuristics is particularly useful, because they can cover most of the elements of the source KB. In particular, we do this for YAGO2s, which is not a small ontology per se (it has a rather big class hierarchy and millions of instances) but uses just 77 different non-metadata properties. The heuristics can be expressed using an RDF ontology that is loaded by the system at startup.

PF-E Mapping Function Given a property P from the source KB and a frame F associated with P, $\text{mappings}_{\text{PF}-\text{E}}(P, F)$ returns frame element properties E with domain F, and associated with P. Informally, this means that property `P` from the source KB can be substituted with property `E` in FrameBase.

General Method. The implementation of $\text{mappings}_{\text{PF}-\text{E}}(P, F)$ computes the text similarity between the name of P concatenated with the name of its range, and the names of the FEs whose domain is F, using the ADW similarity measure [19]. It chooses the candidate with the maximum score for each FE. Note that our algorithm only considers these mappings in restricted settings, e.g. when a frame F has already been chosen. This greatly reduces the set of candidates in practice and enables this approach to deliver good results.

Additional Heuristics. To the general method, we add a heuristic that increases similarity to 1 if the following condition is met: $\text{endsWith}(\text{P}, \text{X}) \wedge \text{endsWith}(\text{FE}, \text{Y})$. The possible values required for X and Y can also be loaded from the heuristic ontology. For Freebase, we use the following two pairs: $(X, Y) \in \{(\texttt{from}, \texttt{time}), (\texttt{place}, \texttt{place})\}$. For YAGO, 4 pairs are required: $(X, Y) \in \{(\texttt{happenedIn}, \texttt{place}), (\texttt{happenedOnDate}, \texttt{time}), (\texttt{endedOnDate}, \texttt{time}), (\texttt{startedOnDate}, \texttt{time})\}$.

C-F Mapping Function Given a class C from the source KB, $\text{mappings}_{\text{C-F}}(C)$ returns frames F associated with C. Informally, this means that class C from the source KB can be substituted with class F in FrameBase.

General Method. We let $F(C)$ denote a candidate set of relevant frames F. In order to filter out noisy and incomplete parts of the source KB, $\text{mappings}_{\text{C-F}}(C)$ returns $\emptyset$ for classes from the origin KB that do not have at least 10 instances and at least 3 outgoing properties with text annotations. Otherwise, $F(C)$ is defined to include all LU-microframes with non-zero lexical overlap (some word in common in the text labels) between C's name and the set of text labels for the synonymous frames from the cluster that F belongs to. Clusters of synonymous frames are formed by LU-microframes that are deemed equivalent via links through synset-microframes. To disambiguate and choose the best frame F among all candidates $F(C)$, we train (and later test, c.f. Section 5.1) logit and SVM classifiers over (C, F) pairs of this form, taken from a gold standard. The (C, F) pairs are considered true when there is a class-frame rule in the gold standard with C in the antecedent and F in the consequent, and false otherwise. Then, for each source KB item, we choose the frame whose pair has the highest score. Although this entails an implicit assumption of functionality, in practice, this results in very significant gains in precision. As input to the model, we use the following four features:

1. The lexical overlap between (i) C's name and (ii) the lexical labels of the cluster of synonymous LU-microframes for F.
2. The lexical overlap between (i) the syntactic head of C's name determined iteratively using the Collins Algorithm [2] and (ii) the lexical labels of the cluster of synonymous LU-microframes for f.
3. The lexical overlap between the descriptions (the longer text labels sometimes identified as comments).
4. If C is a class, the lexical overlap between the union of labels and descriptions of the outgoing properties, upweighting the labels by a factor of 10. When available, the labels and descriptions of the ranges are added too.

For all features, we lemmatize and filter out stop words (closed word classes) and use TF-IDF to compute the feature values (although the second feature is boolean in practice).

In Section 5.1, we test this method using a gold standard manually created for Freebase [1], which is a typical case of a large, open-ended schema, where a fully automatic approach becomes more necessary.

Additional Heuristics. A high-accuracy heuristic can be applied for those source KBs that are linked to WordNet, leveraging that FrameBase includes a significant part of WordNet synset as synset-microframes, which are linked to FrameNet-based LU-microframes.

The heuristic works as follows. If a given source KB class C is associated with a WordNet synset, the synset-microframe based on this synset is looked up in FrameBase. If found, this is the match, and if it is not found, a class C' is selected that is the next most specific WordNet-based parent of C. That is,

$C' \supset C \wedge (C'' \supset C \rightarrow C'' = C')$. Now a synset-microframe is searched for C'. If it is not found, the process is repeated until a match is found, or a maximum number of steps is reached (e.g., 6), in order to avoid overly general rules. With this method, a sound rule can still be created, even if it loses some specificity, and it accounts for the fact that not all synsets are mapped in FrameBase.

This heuristic is particularly relevant for YAGO2s, whose upper class hierarchy is based on WordNet nouns, which makes the mapping obvious. However, it also applies to any other KB for which a mapping to WordNet exists, even if this is an external or a-posteriori one. Since WordNet is a very commonly used linguistic resource, this is reasonably common in LOD KBs.

5 Evaluation

5.1 Integration Rules Created

In this section we present examples of creating integration rules with Algorithm 1 for the test cases of YAGO2s, Freebase (2014-09-21 version), and WikiData (2015-09-28 version).

Creation of Class-Frame Integration Rules

Freebase. To evaluate the results of this method on an arbitrary KB, we produced a manual gold standard consisting of 31 classes and 141 external properties from Freebase [1], paired with their candidate frames or FE properties, respectively. The gold standard is available at `http://framebase.org/data`. Using two independent annotators, we obtained a Cohen's kappa (inter-annotator agreement) $k = 0.69$ for class to macroframe mappings, and $k = 0.38$ for property to frame element mappings. The second is lower because it accumulates the errors from the first, which illustrates how difficult it is to create a gold standard for structured knowledge integration. The classes were randomly chosen from Freebase, disregarding classes whose candidate set did not include a valid match in FrameBase. Freebase was chosen for testing this method because it features Compound Value Types (CVTs), which have a similar role to frames, but we are also able to map some non-CVT classes. Out of a total of 155 outgoing properties for the randomly chosen Freebase classes, 141 could successfully manually be matched to frame elements in the gold standard.

Table 1: Evaluation of mapping external classes to FrameBase classes.

	Baseline-1	Baseline-2	Our Method	
			Logit	SVM
Recall	0.21	0.60	0.50	0.77
Precision	0.12	0.15	0.88	0.77
F1	0.15	0.24	0.63	0.77

Table 2: Evaluation of mapping external properties to FrameBase properties.

Metric	Score
Precision	0.81
Recall	0.30
Accuracy	0.36

Table 1 shows the results for automatic class mappings, averaging over 10 random training/test partitions of ratio 2:1. We compare three different methods.

- Baseline-1 takes the frame class with maximum lexical overlap in names (as in feature 1 of our method) and for which the candidate set $F(x)$ consists of all FrameBase classes (which is a sort of metric that can be configured with the Link Specification language in Silk [26], a state-of-the-art ontology alignment system). However ontology alignment systems alone cannot produce complex mappings.
- Baseline-2 uses the same measure as above, but applying the candidate set $F(C)$ chosen in our method, described in Section 4.3.
- Our method described in Section 4.3, using a logistic regression (logit) classifier and the functional assumption in conjunction with a fixed acceptance threshold of $p > 0.5$, where p is the probability obtained from the logit.
- Our method described in Section 4.3, using a support vector machine (SVM) with radial kernel, selecting, for each source KB class, the candidate frame whose score is highest, given by the distance to the frontier (functional assumption).

Table 2 provides the results obtained by our method for properties, averaging over 10 random training/test partitions of the ground truth data, each of ratio 2:1. Precision and recall are calculated with respect to the gold standard. We obtain higher precision with the logit method because we use the output probabilities to apply a condition that filters out false positives at the cost of a lower recall. Both classifier-based methods outperform the baseline.

Note that in general, word sense disambiguation is considered a hard and yet unsolved problem in natural language processing. This is particular relevant when matching properties that come with little or no metadata. For example, the Freebase classes *education.academic_post* and *base.banned.exiled* must be mapped to the *Employing* and *Residence-reside.v* frames, respectively, for which there is no obvious lexical connection. The same applies when mapping, for example, Freebase properties *education.academic_post.institution* and *geography.river.length* to frame elements *Employing-Employer* and *Natural_features-Descriptor*, respectively. A complete high-precision integration of Freebase into another knowledge base thus requires a larger community effort with additional manual revisions. Our system can be used to automatically propose suggestions to speed up this process.

YAGO. 450 class-frame integration rules were automatically created for YAGO2s. The results are given in Table 3. It shows how the number of matches decreases as n increases and the WordNet-based heuristic for $\text{mappings}_{\text{C-F}}(C)$ moves up the WordNet hierarchy. For $n > 6$ the results are negligible. The ratio of correctly matched entities is 0.789, which is equivalent to the precision of the WordNet-FrameNet mapping used for creating the schema [21] – via clustering of near-equivalent microframes, which uses other links in FrameNet and WordNet that are annotated by experts and therefore expected to be nearly error-free. Figure 7 provides an example of a class-frame integration rule created for YAGO2s.

```
CONSTRUCT { ?s a :frame-Change_of_leadership-revolt.n ;
            :fe-Change_of_leadership-Place ?o .
} WHERE {   ?s a/rdfs:subClassOf yago:wordnet_rebellion_100962129 .
            OPTIONAL { ?s yago:happenedIn ?o }  }
```

Fig. 7: Class-Frame rule, automatically generated for integrating YAGO2s.

Table 3: Number of created class-frame rules for YAGO2s. Matches(n) denotes the number of matches obtained for n being the maximum number of generalization steps. For each column, the left side shows the number of created rules and the right side the number of triples in YAGO2s matching these rules. `endedOnDate` has no significant occurrence in YAGO2s and was therefore omitted.

	`happenedIn`		`happenedOnDate`		`startedOnDate`	
	Rules	Triples	Rules	Triples	Rules	Triples
Matches(0)	38	11,149	86	16,836	4	13
Matches(1)	25	944	83	3,579	5	5
Matches(2)	24	469	58	14,329	1	1
Matches(3)	15	1,232	39	2,315	1	2
Matches(4)	9	540	30	986	0	0
Matches(5)	5	42	14	121	0	0
Matches(6)	2	2	11	39	0	0
All matches	118	14,378	321	38,205	11	21
No match	42	633	148	13,195	0	0
Total	160	15,011	469	51,400	11	21
% Match	73%	95%	68%	74%	100%	100%

Property-Frame Integration Rules

State-of-the-art ontology alignment systems cannot produce something comparable to property-frame integration rules because the binary links produced by these systems (equality, subsumption, etc.) cannot reflect the complex 4-ary nature of property-frame integration rules.

WikiData. We use the general method to automatically extract property-frame rules from WikiData. We evaluate it on YAGO2s, as we can re-use the manually created FrameBase mappings for YAGO2s (described below) as a ground truth. Evaluating the results directly we obtain a precision of 0.80, and using the YAGO ground truth we obtain a recall of 0.21. Figure 5 shows an example of a rule extracted from WikiData.

YAGO. Using the RDF ontology with manually specified heuristics mentioned in Section 4.3, 62 out of the 77 non-metadata properties in YAGO2s (i.e., 81%) could be perfectly integrated into FrameBase using simple property-frame rules.

6 Conclusion

In this paper, we have shown that knowledge base heterogeneity is a problem that goes beyond just the use of different identifiers that need to be aligned. We provide a general analysis of declarative constructs – integration rules – that can also achieve kinds of mappings other than basic entity alignments. We further show that FrameBase is able to incorporate multiple broad-coverage knowledge sources, despite their structural heterogeneity, opening up the possibility for it to serve as a hub for semantic integration of other KBs.

We also provide practical methods to produce these rules, combining general methods with heuristics. The quality of the output is certainly not perfect, but while traditional ontology alignment is already a difficult task, complex mappings have combinatorially more possible candidates and are thus much harder. Our results constitute a first step towards a more comprehensive linking of knowledge.

The total size of the instance data obtained from these source KBs is 40,411,393 statements, which renders it the largest collection of facts linked to FrameNet.

All FrameBase data (schema, ReDer rules, integration rules, instance data, and gold standards) is published under a CC–BY 4.0 International license at `http://framebase.org`.

Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement No. FP7-SEC-2012-312651. Additional funding was received from National Basic Research Program of China Grants 2011CBA00300, 2011CBA00301, NSFC Grants 61033001, 61361136003, 61550110504, as well as from the Danish Council for Independent Research (DFF) under grant agreement No. DFF-4093-00301.

References

1. Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *SIGMOD'08*, pages 1247–1250, 2008.
2. Michael Collins. Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics*, 29(4):589–637, 2003.
3. Jérôme David, Jérôme Euzenat, François Scharffe, and Cássia Trojahn dos Santos. The Alignment API 4.0. *Semantic Web Journal*, 2(1):3–10, 2011.
4. Robin Dhamankar, Yoonkyong Lee, AnHai Doan, Alon Halevy, and Pedro Domingos. iMAP: Discovering Complex Semantic Matches Between Database Schemas. In *SIGMOD 2004*, pages 383–394, 2004.
5. AnHai Doan, Alon Y. Halevy, and Zachary G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.
6. Zlatan Dragisic et al. Results of the Ontology Alignment Evaluation Initiative 2014. In *OM'14*, pages 61–104, 2014.
7. Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate Record Detection: A Survey. *IEEE TKDE*, 19(1):1–16, 2007.

8. Fredo Erxleben et al. Introducing wikidata to the linked data web. In *The Semantic Web – ISWC 2014*, pages 50–65. Springer International Publishing, 2014.
9. Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer, 2007.
10. Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
11. Charles J Fillmore, Christopher R Johnson, and Miriam RL Petruck. Background to Framenet. *International Journal of Lexicography*, 16(3):235–250, 2003.
12. Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M. Suchanek. Canonicalizing Open Knowledge Bases. In *CIKM'14*, pages 1679–1688, 2014.
13. Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. S-Match: an Algorithm and an Implementation of Semantic Matching. In *The Semantic Web: Research and Applications*, pages 61–75. Springer, 2004.
14. Steve Harris and Andy Seaborne. SPARQL 1.1 Query Language. W3C Recommendation, W3C Consortium, March 2013.
15. Patrick Hayes. RDF Semantics. Technical report, W3C Consortium, 2004. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/.
16. A. Kalyanpur et al. Structured data and inference in DeepQA. *IBM Journal of Research and Development*, 56(3.4):10:1–10:14, 2012.
17. Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *ACL'03*, pages 423–430, 2003.
18. Misael Mongiovì, Diego Reforgiato Recupero, Aldo Gangemi, Valentina Presutti, Andrea Giovanni Nuzzolese, and Sergio Consoli. Semantic Reconciliation of Knowledge Extracted from Text Through a Novel Machine Reader. In *K-CAP 2015*, pages 25:1–25:4, 2015.
19. Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. Align, Disambiguate and Walk: A Unified Approach for Measuring Semantic Similarity. In *ACL'13*, pages 1341–1351, 2013.
20. Dominique Ritze, Christian Meilicke, Ondrej Svb-Zamazal, and Heiner Stuckenschmidt. A Pattern-based Ontology Matching Approach for Detecting Complex Correspondences. In *OM'10*, 2008.
21. Jacobo Rouces, Gerard de Melo, and Katja Hose. FrameBase: Representing N-ary Relations Using Semantic Frames. In *ESWC'15*, 2015.
22. Jacobo Rouces, Gerard de Melo, and Katja Hose. Representing Specialized Events with FrameBase. In *DeRiVE'15*, 2015.
23. François Scharffe and Dieter Fensel. Correspondence patterns for ontology alignment. In *Proceedings of the 16th International Conference on Knowledge Engineering: Practice and Patterns*, EKAW '08, pages 83–92, Berlin, Heidelberg, 2008. Springer-Verlag.
24. Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *PVLDB*, 5(3):157–168, 2011.
25. Fabian M Suchanek, Johannes Hoffart, Erdal Kuzey, and Edwin Lewis-Kelham. YAGO2s: Modular High-Quality Information Extraction with an Application to Flight Planning. In *BTW*, pages 515–518, 2013.
26. Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and Maintaining Links on the Web of Data. In *ISWC'09*, 2009.