Knowledge Translation

Bahar Ghadiri Bashardoost
University of Toronto
ghadiri@cs.toronto.edu

Renée J. Miller
Northeastern University
miller@northeastern.edu

Kelly Lyons
University of Toronto
klyons@cs.toronto.edu

Fatemeh Nargesian
University of Rochester
fnargesian@rochester.edu

ABSTRACT

We introduce Kensho, a tool for generating mapping rules between two Knowledge Bases (KBs). To create the mapping rules, Kensho starts with a set of correspondences and enriches them with additional semantic information automatically identified from the structure and constraints of the KBs. Our approach works in two phases. In the first phase, semantic associations between resources of each KB are captured. In the second phase, mapping rules are generated by interpreting the correspondences in a way that respects the discovered semantic associations among elements of each KB. Kensho's mapping rules are expressed using SPARQL queries and can be used directly to exchange knowledge from source to target. Kensho is able to automatically rank the generated mapping rules using a set of heuristics. We present an experimental evaluation of Kensho and assess our mapping generation and ranking strategies using more than 50 synthesized and real world settings, chosen to showcase some of the most important applications of knowledge translation. In addition, we use three existing benchmarks to demonstrate Kensho's ability to deal with different mapping scenarios.

PVLDB Reference Format:
Bahar Ghadiri Bashardoost, Renée J. Miller, Kelly Lyons, Fatemeh Nargesian. Knowledge Translation. *PVLDB*, 13(11): 2018-2032, 2020.
DOI: https://doi.org/10.14778/3407790.3407806

1. INTRODUCTION

Knowledge bases (KBs) have become building blocks for many knowledge-rich applications. As a result, significant work has been devoted to studying methods for creating and populating KBs. Most of this effort is focused on approaches for general-purpose KBs and less work has considered populating domain-specific KBs. Massive numbers of KBs are available today and their numbers are growing making it now possible and desirable to populate (or augment) new KBs with knowledge already available in others. Currently, the Linked Open Data Cloud (LOD) [1] contains 1255 KBs each of which contains more than 1000 triples. Together, KBs in the LOD contain billions of triples. Many approaches have been developed that facilitate the discovery and recommendation of KBs (see [42] for a recent survey). While these approaches can help discover a desirable source of knowledge, the heterogeneity of vocabulary and structure between KBs makes sharing data between KBs difficult. What is needed is a KB equivalent of schema mapping and data exchange [23] in which data structured under a source KB can be faithfully translated to a target KB.

Data exchange requires the existence of a set of rules (called mapping rules) that specify the relationship between the source and target. It is important to note that even in the relational model, heterogeneity cannot be reconciled with simple rules, called correspondences (or sometimes matches). Correspondences are created in ontology alignment (a.k.a. schema matching) [20] and ontology merging [73] techniques. These rules only represent simple relationships (such as equivalence or containment) between small sets of resources. This is all the more true in KBs, as correspondences (even N:M correspondences) cannot express the complex relationships among many resources that need to be exchanged as a whole in order to preserve their relationships. Mapping in general requires complex logic or a full query language. Duo et al. [19] argue that one of the *biggest obstacles* to performing data exchange in KBs is the difficulty in manually creating mapping rules. In traditional (non-KB) data exchange, there is a large body of literature on systems that reduce the burden of creating these rules manually, by making the rule creation process as automatic as possible. See Bonifati et al. [14] for a survey on mapping generation tools (or MGTs). In comparison, there has been much less work on automatic mapping rule generation *when the exchange is between two KBs.* We refer to these tools as KMGT (knowledge-base mapping generation tools).

Challenges and Contributions. Several languages [6, 13, 19, 55] and frameworks [19, 68] have been proposed to help data engineers write KB mapping rules. In addition, there are a few pioneering KMGTs, including Mosto [60, 63] and a system by Qin et al. [58] that automatically create KB mappings. Here we describe some of the limitations of the current solutions (both KMGTs and MGTs) and describe our contributions.

Proceedings of the VLDB Endowment, Vol. 13, No. 11
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3407790.3407806

Associations: The main difference between Kensho and all other MGTs and KMGTs is in the way that it defines associations. Data sharing tasks generally involve two steps: (1) correspondence creation (a.k.a alignment or matching) and (2) correspondence interpretation (a.k.a mapping creation). Central to step (2) is how to create associations – the main semantic unit for associating resources so that they and their relationships are correctly mapped. In MGTs, tables need to be combined to create associations. Following the first relational MGT, Clio [50, 56], most MGTs use declared (or discovered) constraints (like foreign keys or inclusion dependencies) to create mappings. Using this approach, and excluding cycles, there are typically only a small number of ways of joining any two tables. KBs are much more general graphs and contain large numbers of property paths. In general, these property paths are not indicated as being full (functional) inclusions. This is an important property of KBs, their flexibility in representing information.

Example 1.1. Consider the source KB in Figure 2. In a relational world where `Organization` and `Country` are tables, if we know that if an `Organization` has value for the attribute `country` then it is a valid key for `Country` (i.e., there is a FK from `Organization.country` to `Country`), then when mapping `Organization` and `Country` data to the target, MGTs will be sure to map an `Organization` with its own `country` (and not the country of a different organization). However, if a `Person` has a `has-worked-for` attribute, but this attribute is not declared as having to have an organization (e.g., a FK constraint), then MGTs will not generate an association between Person and Organization. In contrast, KBs often contain properties that hold only for portions of the data. The issue here is that KBs are open-world models and include many relationships that exist only for a portion of the data. Consequently, generating associations for all paths has been considered *infeasible* by existing KMGTs [58, 60, 63]. ■

To solve this complexity problem, Mosto [60, 63] assumes that two concepts are associated only if they are connected via an aligned object property or if one is the ancestor of the other. Because of this assumption, Mosto *relies heavily* on the existence of aligned properties, and if there are no correspondences between properties, it cannot interpret correspondences collectively among concepts which are not subclass/superclass of each other. To mitigate this, Mosto allows users to *manually* add to the KB new constructs `mosto:strongRange` and `mosto:strongDomain` that *mimic* the role of a relational FKs. This permits Mosto to follow a MGT-like approach that also forms associations over paths that include these new, user-provided, annotations. The newer MostoDex [64, 65] can generate mapping rules without relying on any KB constructs, including user-provided annotations, and instead uses user-provided examples. Both Mosto and MostoDex require a user to fully understand the nuances of KB translation in order to provide correct axioms or sufficiently informative examples. In contrast, Kensho does not require user-provided KB constructs or user-provided examples. To the best of our knowledge, Kensho is the first KMGT to consider associations that cover all property paths in both the source and target. It is worth mentioning that the KMGT proposed by Qin et al. [58] uses a similar approach to Mosto but does not allow user intervention.

Structurally Valid Mappings: Considering all associations, however, can lead to an overwhelming number of mappings. To conquer this complexity (without eliminating desired mappings), we define **c2a valid** mappings, that eliminate many mapping alternatives by requiring mappings to respect the internal structure and interconnection of individuals and their attributes. Importantly, mappings that are c2a valid ensure structural consistency of mapped resources even when value invention is required. In addition, Kensho uses correspondences between property paths when available, called r2r mappings, which may be provided by a matcher or correspondence generator, to narrow down a set of good mappings (called **r2r valid** mappings). Note that previous approaches do not accept correspondences among object property *paths* of length greater than one.

Knowledge Translation: The theory of data exchange (i.e. semantics and query answering) between KBs has already been studied [8, 9, 10, 11]. However, the practicalities of generating queries that create valid solutions over real KBs has not received as much attention. Kensho's mapping rules are expressed using SPARQL queries which means they can be used directly to exchange knowledge from source to target. Because we create associations using all paths in a KB, we have systematically considered how to create correct queries over associations that include property paths, without cardinality or functionality restrictions. Our solutions perform the value invention needed (for example, when a target concept does not exist in the source, but its attributes do). Kensho correctly associates data and translates all desired data (using the SPARQL optional command as required). MGTs use sophisticated methods for value invention (using labeled nulls or skolem functions) such that the structure of the target and source are preserved [22, 46]. When dealing with KBs, *blank nodes* fulfill this role. To the best of our knowledge, Kensho is the only KMGT that creates these blank nodes while preserving the grouping and relationships from the source KB.

Example 1.2. Consider the scenario presented in Figure 1. Current KMGTs create mapping rules which exchange data as shown in Box A. They associate the phone number and address of a single source `Office` with two (possibly different) `Contact` resources in the target. This does not capture the semantic information encapsulated in the structure of the source KB, namely, that this phone and address are associated with one another. Kensho on the other hand, creates mapping rules that exchange data as shown in Box B, capturing the original grouping of data in the source KB. Kensho can do this even if the source is incomplete (some offices do not have addresses, some do not have phones, or some have neither). ■

Ranking Heuristics: Kensho is the only tool that takes on the task of generating associations for all paths in the KB. While our structural validity requirements reduce the number of possible mappings, we may still generate many mappings. To reduce the burden of selecting the best set of mapping rules for the exchange task at hand, we have proposed three new ranking heuristics to rank the final set of *valid mappings*. Buhmann et al. [16] observe that, in practice, there is a lack of KBs that contain high quality schema axioms. Keeping this in mind, we have designed these heuristics such that they do not require KBs to both be populated with instances or to be annotated with reach axioms such as cardinality, functionality, or disjointness.

Evaluation: We evaluated the performance of Kensho on several real-world scenarios designed to highlight the role of knowledge translation in different tasks including KB popu-

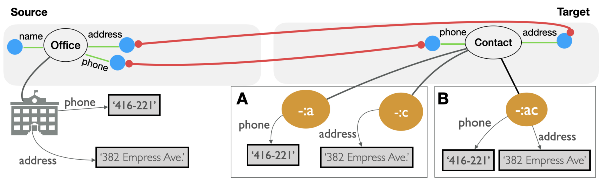

Figure 1: Red lines represent a correspondence.

lation, versioning, and migration. To show the performance of mapping generation in complex scenarios, we have also evaluated Kensho on a large number of synthetic scenarios. Our results show that Kensho scales very well even for KBs that are three times larger than DBpedia in terms of the the number of concepts, with the largest bottleneck being the number of possible interpretations of a correspondence. We have also compared Kensho with existing KMGTs on three existing benchmarks. Finally, we have included a small case study to further investigate the effectiveness of Kensho.

2. METHODOLOGY

Kensho generates executable mapping rules in two steps: semantic association discovery and correspondence interpretation. When interpreted separately, correspondences cannot describe how to translate the resources of KBs in conjunction with each other. To determine a set of executable mapping rules which weave these correspondences together, we must understand what relationships exist between aligned resources *within* each of the two KBs. We call these relationships semantic associations. The goal of the first step is to discover these associations. The goal of the *correspondence interpretation* step is to interpret the set of given correspondences collectively in a way which respects the discovered semantic associations among elements of each KB. We begin by defining correspondences.

2.1 Correspondences

We distinguish between two important types of properties in a KB: 1) a *datatype property* or *attribute* which represents a relationship between an IRI and a Literal and 2) an *object property* which expresses a relationship between two IRIs. A property path is a list of properties in an RDF graph between resources [70]. An attribute or an object property is a property path of length one. Property paths are either data property paths (a path between an IRI and a Literal) or object property paths (a path between two IRIs). Property paths can be expressed using a regular expression grammar. We use the grammar from W3C [70] to represent them.

In this work, we define three types of correspondences. First, a **Concept2Concept** correspondence associates a concept `s` in the source to a concept `t` in the target, represented as $\mathtt{s} \rightsquigarrow_{c2c} \mathtt{t}$. In Figure 2, $C3$ is a Concept2Concept correspondence between the source concept `Person` and the target concept `Person`. Second, a **Rel2Rel** correspondence associates an object property path `P` in the source to an object property path `R` in the target, represented as $\mathtt{P}_{(\mathtt{s}_1,\mathtt{s}_2)} \rightsquigarrow_{r2r} \mathtt{R}_{(\mathtt{t}_1,\mathtt{t}_2)}$ where $\mathtt{s}_1$ and $\mathtt{s}_2$ are connected by `P` and $\mathtt{t}_1$ and $\mathtt{t}_2$ are connected by `R`. In Figure 2, $C4$ is a Rel2Rel correspondence that associates the source object property `employer` with the target object property `works_for`. An **Attr2Attr** correspondence associates a data property path in the source with a data property path in the target, represented as $\mathtt{a}_\mathtt{s} \rightsquigarrow_{a2a} \mathtt{b}_\mathtt{t}$, where `s` and `t` are concepts that are connected by a data path to attribute values of `a` and `b`, respectively. In Figure 2, $C2$ is an Attr2Attr correspondence that associates the source attribute `address` with the target attribute `address_line`. Correspondences can represent various relationships. In this work, we use correspondences that express subset-or-equal relations since correspondences produced by automated tools are often of this type [20]. Following the terminology of Euzenat and Shvaiko [20], we call a **set** of correspondences an *alignment*.

We say a concept is an *aligned concept* if it either directly participates in a Concept2Concept correspondence or if it participates in an Attr2Attr correspondence (meaning for $\mathtt{a}_\mathtt{s} \rightsquigarrow_{a2a} \mathtt{b}_\mathtt{t}$, it is the concept `s` or some concept along the path from `s` to the attribute `a`). Note that we treat Rel2Rel correspondences differently, and use them to refine how concepts are mapped collectively using a notion we call **r2r validity** defined in the next section. Finally, in general we call any KB resource that participates in a correspondence an *aligned element*. It is worth mentioning that Mosto also uses another type of correspondence that matches an attribute value to an instance of a concept. In these situations, we create an Atr2Atr that matches the attribute in the source to the `rdfs:label` attribute of the concept in the target. Note that this type of correspondence is different from metadata-data correspondences [38, 49] which Kensho does not support.

2.2 Semantic Association Discovery

In a KB, semantic associations between aligned elements are represented as property paths between them. First, a concept and its attributes indicate a semantic association (sometimes called the internal structure of the KB [20]). In Figure 2, the fact that `trgt:name` and `trgt:work_phone` are attributes of the concept `trgt:Person` indicates that for each individual of type `trgt:Person`, the value of these attributes are semantically related. Second, two concepts (and their attributes) are associated when there is a property path between them. For instance, information about an organization and its employees may be modeled as a path containing a number of concepts. Of course, in the presence of cycles, the number of associations is infinite so Kensho will only enumerate a finite set of these.

Definition 2.1. (Basic Association) Each aligned concept defines a Basic Association that includes the concept along with all its aligned attributes. We call the aligned concept the *root* of the basic association. ■

In addition to basic associations, aligned concepts can be associated by the relational [20] structure of a KB. To define these associations, we use property paths between the roots of basic associations.

Example 2.2. In Figure 2, the source concepts `Organization` and `Country` are associated through the path `located_in`. The instances of this *association* can be retrieved using the following two queries.

```
SELECT * WHERE {
?o a src:Organization.
OPTIONAL
  {?o  src:located_in  ?ozCountry.
   ?ozCountry  a src:Country.}}
```

```
SELECT * WHERE {
?c a src:Country.
OPTIONAL
 {?c  src:located_in^  ?czOrganization.
  ?czOrganization  a src:Organization.}}
```

The property `src:located_in` models the relationship between the two aligned concepts `Organization` and `Country` which are the roots of basic associations.

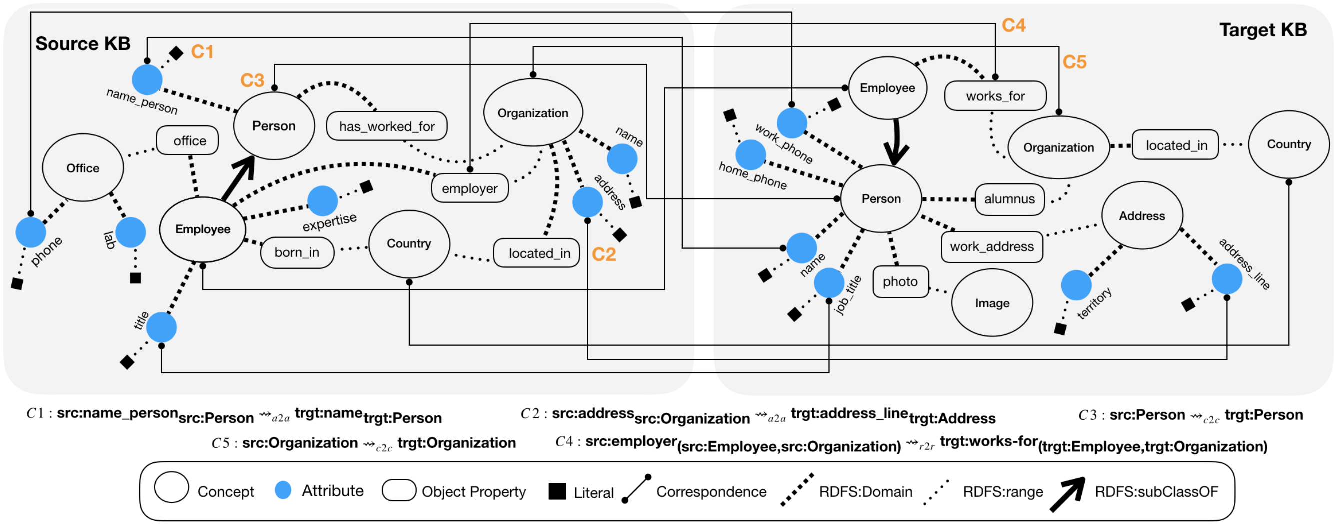

Figure 2: RDFS layer of two KBs and correspondences between them.

Note that in KBs, the source might be incomplete, hence, the query needs to contain the `OPTIONAL` keyword. ■

To directly associate two aligned concepts, we use paths, called *association paths* that do not go through other aligned concepts. We limit the length of these paths to $\mathcal{D}$. To simplify notation, we assume that each instance in our KBs has exactly one most-specific type. This is not an essential assumption. In our TR [31], we briefly discuss how this assumption may be relaxed by generalizing the definition of Association Path.

Definition 2.3. (Association Path) An association path p between aligned concepts u_0 and u_1 is an ordered list of resources on object property path Π which matches: $(\texttt{rdfs:domain}^{\wedge}|\texttt{rdfs:subClassOf})/$ $(\texttt{rdfs:domain}^{\wedge}|$ $\texttt{rdfs:domain}|\texttt{rdfs:range}|\texttt{rdfs:range}^{\wedge}|\texttt{rdfs:subClassOf})^*$ such that (u_0, u_1) is in the evaluation of Π, and there is no aligned concept along Π. The concept u_1 (also called *tail* of the association path) is part of the path, but u_0 (also called the *root* of the association path) is not part of the path. ■

Example 2.4. In Figure 2, there is only one association path that connects the target concept `Organization` to `Country`, namely `[located_in, Country]`. Note there are no association paths from `Country` to `Organization` (because R does not include $\texttt{rdfs:range}^{\wedge}$). This is simply to prohibit enumerating redundant paths. If we consider `Employee` to `Person` (again in the target), then `[Person]` is an association path which is created by going through the `rdfs:subClassOf` property path from `Employee`. If `Organization` were *not* aligned then `[works_for, Organization, alumnus^, Person]` would also be an association path. ■

We use association paths to connect basic associations. Semantic associations start with a basic association for an aligned concept u_0. We add to u_0 all association paths to other aligned concepts u_i together with the basic association for u_i. We repeat this process recursively adding to the semantic association paths from u_i to other aligned concepts u_j together with the basic association of u_j. We limit the length of the longest path in the semantic association to $\mathcal{L}$.

Definition 2.5. (Semantic Association) The semantic association for aligned concept u_0, denoted $\mathcal{A}(u_0)$, contains the basic association for u_0. In addition, if u_i is a concept in $\mathcal{A}(u_0)$ and there is an association path p from u_i to aligned concept u_j and if adding p to $\mathcal{A}(u_0)$ does not create a path longer than length $\mathcal{L}$ in $\mathcal{A}(u_0)$, then the association path p is in $\mathcal{A}(u_0)$ and the basic association of u_j is also in $\mathcal{A}(u_0)$. For each concept in each p added to $\mathcal{A}(u_0)$, a new node will be created (even if a node representing that concept is already in $\mathcal{A}(u_0)$). A semantic association $\mathcal{A}(u_0)$ can be thought of as a tree, where the concepts in $\mathcal{A}(u_0)$ are nodes and the root is a node representing concept u_0. We define this tree such that if u_i and u_j are associated via a `rdfs:subclassOf` path, then there is an edge between them labeled with `a`. Otherwise, the edge is labeled with the corresponding property. Each node $n \in \mathcal{A}(u_0)$ is assigned a variable (denoted *var*(n)) and we use the notation *concept*(n) to denote the concept that n represents. Also, if the basic association of *concept*(n) is in $\mathcal{A}(u_0)$, each attribute a in this basic association is assigned a variable (denoted *var*(n, a)). For a semantic association $\mathcal{A}$, we call the set of all variables assigned to its nodes and attributes, denote $\mathcal{A}_{vars}$, the *variables* of the semantic association. ■

For the KBs of Figure 2, the semantic associations of the source and target `Employee` concepts are depicted in Figure 3. Note that cycles and KBs with multiple property paths between the same concepts can lead to semantic associations containing multiple nodes for the same concept.

2.3 From Correspondence to Mapping

Semantic associations (within the source and target) can be used to understand correspondences collectively.

Example 2.6. The correspondences $C1$ and $C2$ of Figure 2 can be interpreted independently. If we do this, we would get mappings that can be represented by the following queries.

```
construct{
 -:trgtPerson a trgt:Person.
 -:trgtPerson trgt:name ?trgtName.}
where{
 ?srcPerson a src:Person.
 ?srcPerson src:name_person ?srcName.
 bind(?srcName as ?trgtName)}
```

```
construct {
 -:tAddress a trgt:Address.
 -:tAddress trgt:address_line ?trgtAddress.}
where {?srcPerson a src:Person.
 ?srcPerson src:has_worked_for ?srcOrg.
 ?srcOrg src:address ?srcAddress.
 bind(?srcAddress as ?trgtAddress)}
```

These mappings create target `addresses` from source `addresses` (and target `names` from source `names`), but do not associate `names` and `addresses` in the target. To preserve source informa-

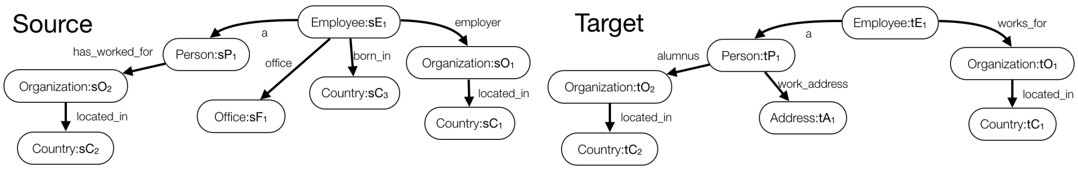

Figure 3: Semantic associations $\mathcal{A}$(`src:Employee`) and $\mathcal{A}$(`trgt:Employee`) with attribute variables omitted.

tion, specifically the association between resources, we need to understand when a set of correspondences can be interpreted collectively. For instance, if $C1, C2$, and $C3$ were the only correspondences in Figure 2, then the following mapping is a better interpretation of the correspondences.

```
construct {?trgtPerson a trgt:Person.
    ?trgtPerson trgt:name ?trgtName.
    ?trgtPerson trgt:work_address -:trgtAddress.
    -:tAddress a trgt:Address.
    -:tAddress trgt:address_line ?trgtAddress.
  }
where {?srcPerson a src:Person,
      OPTIONAL{?srcPerson src:name_person ?srcName}
      OPTIONAL{?srcPerson src:has_worked_for ?srcOrg
          OPTIONAL{?srcOrg src:address ?srcAddress}}
      bind(?srcName as ?trgtName)
      bind(?srcAddress as ?trgtAddress)
      bind(?srcPerson as ?trgtPerson)}
```

Similar to the mappings that interpret each correspondence independently, the above mapping dictates how to create target `addresses`, `names`, and `persons` from the resources of the source. However, this mapping *also* preserves the relationships among the translated elements, and thus is usually more desirable. ■

The first step of query generation is to find correspondences that can be interpreted together. To do this, we consider pairs containing one source semantic association and one target semantic association and define the set of correspondences that are *covered* by this pair (e.g., the pair $\langle\mathcal{A}$(`src:Person`), $\mathcal{A}$(`trgt:Person`)$\rangle$ covers the three correspondences in the above example). Following the terminology used in Clio [56], a *Skeleton* is a pair $\langle S, T\rangle$ where S is a source semantic association and T is a target semantic association. To identify how a correspondence C can be interpreted using a skeleton, or in other words to identify the coverage of correspondence C by a skeleton, it is not enough to check whether the semantic associations include the aligned elements which are participating in the correspondence C. One reason is that the same concept of the KB might be included more than once in a semantic association. Hence, we define a *renaming function* that associates target variables of the target association to variables of the source association. We do this in a way that respects the correspondences. In general, there may be multiple ways to cover a correspondence with respect to a pair of source and target semantic associations and each represents a specific (different) interpretation of the correspondences. We give an example, then formally define coverage.

Example 2.7. Figure 3 depicts a pair of source and target semantic associations. All nine correspondences shown in Figure 2 are covered by this pair of semantic associations. Some of the correspondences such as `src:Country` $\rightsquigarrow_{c2c}$ `trgt:Country` can be covered in multiple ways. Intuitively, there are six possible interpretations of the country correspondence. The country of the organization where an employee works (target variable tC_1) can be populated with the country of an organization which is her employer (sC_1), or the country of an organization for which she has worked (sC_2), or with the country where she was born (sC_3). Similarly, the country of the organization of which a target employee is an alumni (variable tC_2) can be populated with any of these three source options. Note that there are other options, for example, one might decide to map both target countries to source data, or to leave one of these target countries unmapped. Also note that we are mapping each employee with a country (maintaining this association), so some of these options may require value invention for concepts along a target path that do not exist in the source. ■

$\mathfrak{R}_1$

tE1 = sE1	$tE1_{name} = sE1_{name}$
tO2 = sO2	$tE1_{title} = sE1_{title}$
tP1 = sP1	tO2 = sO2
tC2 = sC2	$tE1_{phone} = sF1_{phone}$
tC1 = sC1	$tA1_{address\_line} = sO1_{address}$

$\mathfrak{R}_2$

tE1 = sE1	$tE1_{name} = sE1_{name}$
tO1 = sO1	$tE1_{title} = sE1_{title}$
tP1 = sP1	$tA1_{address\_line} = sO1_{address}$
tC1 = sC1	$tE1_{phone} = sF1_{phone}$

Target elements not mentioned map to ϵ

Figure 4: Two of the possible renamings of $\langle\mathcal{A}$(`src:Employee`)$, \mathcal{A}$(`trgt:Employee`)$\rangle$.

We now define this formally by defining renamings from the variables of T (denoted by T_{vars}), to the variables of S (denoted by S_{vars}). A renaming is a *total function* that maps each variable of T either to a variable of the source or to ϵ which will be an indication that the mapping query *might* need to perform value-invention for this variable (something we discuss in Section 2.4).

Definition 2.8. (Correspondence Coverage) A correspondence C is *covered* by a skeleton $\langle S, T\rangle$ if there is a *renaming* $\Re : T_{vars} \rightarrow S_{vars} \cup \{\epsilon\}$, where:

- (Concept2Concept) If $C : s \rightsquigarrow_{c2c} t$, then $\exists n \in S, m \in T$, $\Re(var(m)) = var(n)$, $concept(n) = s$, $concept(m) = t$.
- (Rel2Rel) If $C : P_{(s_1,s_2)} \rightsquigarrow_{r2r} R_{(t_1,t_2)}$, then $\exists n_1, n_2 \in S$ and $\exists m_1, m_2 \in T$, $\Re(var(m_1)) = var(n_1)$, $\Re(var(m_2)) = var(n_2)$, $concept(n_1) = s_1$, $concept(n_2) = s_2$, $concept(m_1) = t_1$, and $concept(m_2) = t_2$.
- (Attr2Attr) If $C : a_s \rightsquigarrow_{a2a} b_t$, then $\exists n \in S, m \in T$, $concept(n) = s$, $concept(m) = t$, $\Re(var(m, b)) = var(n, a)$.

If a correspondence C is covered by $\langle S, T\rangle$ using a renaming $\Re$, then we call $\Re$ an *interpretation* of C. ■

Notice that a single renaming may be an interpretation for many correspondences.

Definition 2.9. (Skeleton Renaming) Given a skeleton $\langle S, T\rangle$ and a set of correspondences $\mathcal{C}$, then $\Re$ is a renaming for $\mathcal{C}$ if it includes at least one interpretation of each correspondence in $\mathcal{C}$. The set $\bar{\Re}_{S,T}$ denotes all possible $\Re$s over a skeleton $\langle S, T\rangle$. ■

Example 2.10. In Figure 2, let $\mathcal{C}$ contain all correspondences except $C4$. Figure 3 depicts source and target associations for the skeleton $\langle\mathcal{A}$(`src:Employee`)$, \mathcal{A}$(`trgt:Employee`)$\rangle$.

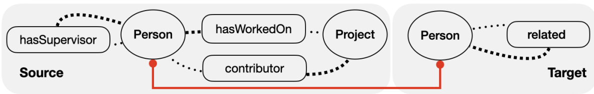

Figure 5: Red line represents a correspondence.

Given $\mathcal{C}$, Figure 4 shows two possible renamings ($\Re_1$ and $\Re_2$) for this skeleton. In comparison with $\Re_1$, $\Re_2$ does not use the source variables `sO2` and `sC2`, and the target variables `tO2` and `tC2` now map to ϵ. The renaming $\Re_2$ might be desirable if a data engineer decides that the relationship `trgt:alumnus` between `trgt:Person` and `trgt:Organization` in the target is not expressed by any property path between `src:Person` and `src:Organization` in the source, and thus should not be part of the translation. ■

To accommodate the modeling flexibility of KBs, we have defined semantic associations to include any paths up to a given length (not just functional, total, or user indicated paths). Thus, there can be a large number of possible renamings for a pair of semantic associations. We define three notions of validity, each of which reduces the number of possible renamings without excluding desirable renamings.

The first is similar to what is used now by existing KMGTs [58, 60, 63] and requires that the renaming only map concepts and attributes for which there is a correspondence (and map them as indicated by the correspondence). The second requires the renaming to respect all correspondences between relationships (property paths). Note that the set of *Rel2Rel* correspondences may be incomplete (due to limitations of current alignment tools), but if present, the renaming must respect them. The third requires the renaming to respect the internal structure of the KB and interconnection between concepts and attributes, and map the attribute values to individuals correctly.

Definition 2.11. (Baseline Validity) Given a set of correspondences $\mathcal{C}$ and skeleton $\langle S, T\rangle$, renaming $\Re$ is baseline valid if:

- $\forall n \in S, m \in T$, if $\Re(var(m)) = var(n)$, $concept(n) = s$, and $concept(m) = t$, then there is a correspondence $s \rightsquigarrow_{c2c} t \in \mathcal{C}$
- $\forall n \in S, m \in T$, if $\Re(var(m,b)) = var(n,a)$, $concept(n) = s$, $concept(m) = t$, then $\exists\ a_s \rightsquigarrow_{a2a} b_t \in \mathcal{C}$ ■

The number of baseline valid renamings can still be large. Kensho use path correspondences (Rel2Rel) to narrow down the set of possible valid renamings.

Definition 2.12. (r2r Validity) Given a Rel2Rel correspondence $C : P_{(s_1,s_2)} \rightsquigarrow_{r2r} R_{(t_1,t_2)}$ and skeleton $\langle S, T\rangle$, a renaming $\Re$ is r2r valid for C if: $\exists m_1, m_2 \in T$, and $\exists n_1, n_2 \in S$, where $concept(m_1) = t_1$ and $concept(m_2) = t_2$ and $concept(n_1) = s_1$ and $concept(n_2) = s_2$, and $\Re(var(m_1)) = var(n_1)$ and $\Re(var(m_2)) = var(n_2)$, and n_1 and n_2 are connected through P in S, and m_1 and m_2 are connected through R in T. ■

Previous KMGTs do not take advantage of Rel2Rel correspondences in which the length of the corresponding property paths is greater than one. For instance in Figure 5, none of the current KMGTs consider a correspondence that represents the fact that two persons who are working on a project in the source are related in the target since this requires mapping the `hasWorkedOn/contributer` property path in the source to the `related` property in the target. Note this example requires no value invention and yet is typically not considered in the literature. In addition, Kensho considers path correspondences where not every resource on the target path exists in the source, and hence value invention is required. (See Section 2.4 for more detail.)

An important innovation in Kensho is to consider the different ways in which each correspondence can be covered in a mapping. Our associations can capture multiple interpretations of the same correspondence even in a single renaming (see Example 2.7). This functionality is important especially in cases where value invention is needed, since correct mapping requires that all desired resources and their relationships must be in a single query so that the newly created blank node IRIs can group resources properly. However, by allowing multiple interpretations of a correspondence, a renaming might need to map multiple attributes or concepts of the same type. For instance, in Figure 6 our association needs to express a rule that describes an employee, her current employer address, and the address of places where she has worked in the past. In such cases, we need to make sure that the addresses are mapped to the correct organizations. Kensho uses c2a validity to ensure concepts and attributes are collectively mapped properly. The main goal of c2a validity is to require the mappings to respect the internal structure of the source KB and interconnections between concepts and attributes, so that the attribute values of individuals are preserved.

Definition 2.13. (c2a Validity) Given two correspondences $C_1 : s \rightsquigarrow_{c2c} t$ and $C_2 : a_s \rightsquigarrow_{a2a} b_t$, and a skeleton $\langle S, T\rangle$, a renaming $\Re$ is c2a valid for C_1 and C_2 if: $\forall n \in S, m \in T$, such that $concept(n) = s$, $concept(m) = t$, if $\Re(var(m)) = var(n)$, then $\Re(var(m,b)) = var(n,a)$. ■

Note that a data engineer might choose a renaming that does not transfer the attribute value(s) at all, but if the value is mapped, then that value should be associated with the same individual which is mapped from the source. Although current KMGTs do not consider this validity, traditional mapping generation tools (MGTs) produce mappings which are c2a valid. We have defined additional notions of validity [31] for scenarios where multiple attributes of a concept are mapped, but their concept is not.

2.4 Mapping Generation

For each skeleton renaming ($\langle S, T\rangle, \mathcal{C}, \Re$), we can create a mapping. We use the semantic associations S and T to create source and target query patterns. Then we use $\Re$ to create what is effectively an inclusion dependency from the source query pattern to the target query pattern. We first describe how an **association query pattern** is defined based on a semantic association. Given a semantic association $\mathcal{A}(u_0)$, to create an association query pattern, we start from the root node n_0 (where $concept(n_0)$ is u_0). We use $var(n_0)$, v_0, to create a SPARQL pattern that expresses the type of instances of node n_0. That is, `?v0 a u0`. For each of the root's children, n_i, we create a fact that represents $l(u_0, concept(n_i))$, where l is the label of the edge which connects n_0 to n_i. That is, `?v0 l ?vi`, where v_i is $var(n_i)$ unless l is label `a`, in that case, the variable of the parent, v_0, will be used instead for representing that node through out the whole process of query generation. We repeat the process recursively for each of n_0's children. If l is anything other than label `a`, we will nest what follows for that child in the `OPTIONAL` clause. Also, for each node, $n_j, j \geq 0$, we will express each of $concept(n_j)$'s attributes q, using statements

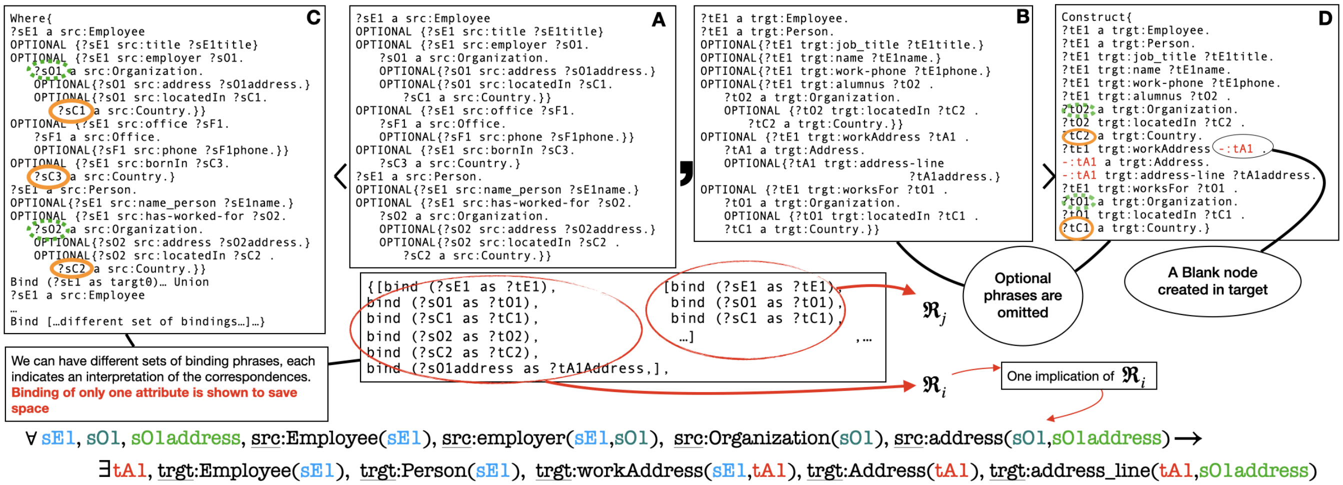

Figure 6: Query generation process.

like ?v$_j$ q ?attr$_j$, where attr$_j$ is $var(n_j, q)$, and v_j is $var(n_j)$. Each of these attribute statements will be nested within an OPTIONAL clause.

Example 2.14. Figure 6 Box A and B, are association query patterns created from the semantic association trees of $\mathcal{A}$(src:Employee) and $\mathcal{A}$(trgt:Employee) of Figure 3 respectively. Note that in both patterns, the Employee variable is used for representing the Person - and the clause that represents the fact that an employee is a person is not nested within an OPTIONAL keyword. The reason is that the semantics of inheritance implies that if B is a subclass of A, then every individual of type B is also of type A. ■

Given source and target association query patterns generated from a skeleton $\langle S, T\rangle$, we now consider how skeleton renamings can be used over association query patterns to create mappings. We create a SPARQL *construct query* for each skeleton. Assuming that there are N possible renamings for skeleton $\langle S, T\rangle$, (that is, $|\bar{\Re}_{S,T}| = N$), *for each* renaming $\Re \in \bar{\Re}_{S,T}$, an association query pattern of S will be expanded with a set of binding clauses created from $\Re$ to create an $S_{\Re}$ graph pattern. More specifically, for each target variable v_i which is mapped to a source variable $\Re(v_i)$, the binding clause "bind $\Re(v_i)$ as v_i" will be created and added to the triple pattern of $S_{\Re}$. The final Where clause of the construct query will be the *union* of all graph patterns (see Figure 6 Box C for an example.) Note that a data engineer can choose to only use a subset of the renamings in $\bar{\Re}_{S,T}$. In particular, in practice, we expect to only be working with valid renamings.

In order to create the construct clause of our mapping query, it is important to note that some variables of the target's semantic association may be mapped to ϵ. In order to materialize the target, sometimes we will have to fill in the values for the undetermined variables. For instance, the renaming $\Re_2$ of Example 2.10, shown in Figure 4, represents a translation which transfers the attribute values of $sO1_{address}$ to $tA1_{address\_line}$. However, $tA1$ maps to ϵ. For this example, new blank nodes need to be created for $tA1$ if we want our construct query to be able to transfer the values in the source $sO1_{address}$ attribute to the target $tA1_{address\_line}$ attribute. These blank nodes correspond to existential variables that are common in data exchange [23]. In the same example, the renaming $\Re_2$ maps some of the other target variables such as $tO2$ to ϵ. In this case, no value invention is need to maintain the structure of the translated data, we can simply not map any data to this node. For target attribute values, generally value invention is not required.

To summarize, Kensho will create blank nodes for a variable in the target construct query if that variable is representing a concept which is not an aligned concept or if it represents an aligned concept which is mapped to an ϵ *and* if these variables (whether they are aligned or not) are in a path that leads to mapped concepts or mapped attribute variables. To generate the Construct clause of the query, the association query pattern of T will be used (without the OPTIONAL keywords). In addition, as described above, the variable names in this pattern will be converted to blank nodes, if necessary. Elsewhere we have provided additional details about how these blank nodes are created [31].

3. MAPPING RANKING

For a given skeleton, the set of possible renamings, even valid renamings, each of which creates a different mapping, might be large due to the diverse ways in which knowledge can be represented in KBs. Thus, we introduce a set of heuristics that help rank renamings of each skeleton. A data engineer can then browse through the ranked set of mappings (or through a set of examples created using mapping queries) and choose the most desirable set.

Many cues in the KB can be used to rank our mappings. For instance, Maponto [7], which produces mapping rules in settings that involve relational models and KBs (see Section 5 for details) suggests that reach axioms such as functionality or cardinality can be used to select better mappings. However, these reach axioms are usually not available [16]. In addition, instances of source and target KBs can be used to select mappings which exchange more facts that are already present in the target [57]. However, one of the most important applications of this work is to populate a target, and thus it is not reasonable to assume that target is already populated with a large amount of the source's data. Furthermore, in the presence of set of positive or negative facts, rule mining approaches (s.a. [28, 48, 52]) can help identify important paths (or rules) that best fit a set of given examples. However, we cannot always assume that such examples exists and creating a good set of examples is itself an interesting research challenge. In this work, we present three complementary heuristics that do not require the information mentioned above, but that can be used in concert with other information when available. Our first two heuristics are structural, giving higher ranks to mappings that are more consistent with the structure of the KBs, the third is based on the coverage of the mappings, giving higher ranks to renamings that cover more target elements.

First, we rank mappings based on the degree to which source and target **paths** are collectively mapped. Recall that an association path does not go through any other aligned concept. If a mapping maps two target concepts between which there is an association path, to two source concepts that have no association path between them, we rank this mapping lower than a mapping that uses two source concepts which are connected with an association path.

Definition 3.1. (Path Priority) Assume $s_0 \rightsquigarrow_{c2c} t_0$. For any association path p with root s_0, the cost of adding p to s_0 with respect to the *target* KB, $Cost(s_0, p)$, is 1 if there exists no association path between t_0 and t_1, where $s_1 \rightsquigarrow_{c2c} t_1$, and s_1 is the tail of p, and is 0 otherwise. The cost of adding a set of ordered association paths, $\{p_1, ...p_k\}$, to s_0, where $k > 2$, and s_0 is the root of p_1, s_i is the tail of p_i and the root of p_{i+1}, with respect to the *target* KB, $Cost(s_0, p_k)$, is:

$$Cost(s_0, p_1) + \sum_{i=1}^{k-1} Cost(s_i, p_{i+1})$$

For node n in $\mathcal{A}(u_0)$, where $concept(n)$ is an aligned concept, a set of ordered association paths, P, can be defined such that P includes association paths that are used to connect n_0 to n, in that order, where n_0 is the root of the association tree. The Path score of variable v of node n is the cost of adding P to u_0, and is $score(v, \mathcal{A}(u_0))$. For Skeleton $\langle S, T\rangle$, The Path cost of $\Re$ is:

$$Path_{cost}(\Re, \langle S, T\rangle) = \sum_{v} score(v, S)$$

where v ranges over all the variables of nodes of aligned concepts in S. We say $\Re_i$ has higher Path priority than $\Re_j$ if it has a lower cost. ■

Example 3.2. In Figure 3, the $score(\texttt{sC3}, \mathcal{A}(\texttt{src:Employee}))$ is 1, since there is no association path between `trgt:Employee` and `trgt:Country` in the **target** KB, while $score(\texttt{sC1}, \mathcal{A}(\texttt{src:Employee}))$ or $score(\texttt{sC2}, \mathcal{A}(\texttt{src:Employee}))$ are 0. Note that sometimes it is tempting to pick the shortest path between two concepts as the best property path that describes the relationship between them (e.g., `src:born_in` here), however this example shows that the shortest path is not always best. ■

Next, we consider semantic associations that include multiple subpaths containing identical concepts (like the two `Organization, Country` paths in our running example). These paths will have identical Path scores. Mappings that only use a single one of these multiple paths (rather than mixing concepts in two or more such identical paths) are generally better reflections of the domain. To define **Consistency Priority**, we note that the variables in a semantic association (either S or T) are organized in a tree starting at the variable for the root. We say v_j is reachable from v_i, if v_j is a descendant of v_i in this tree.

Definition 3.3. (Consistency Priority) For skeleton $\langle S, T\rangle$, set the Consistency cost of the renaming $\Re$, $Consistency_{cost}(\Re, \langle S, T\rangle)$, to zero. For each pair of target variables v_i and v_j which both are bounded by some source variables in $\Re$:

- if v_j is reachable from v_i (in T), and $\Re(v_j)$ is **not** reachable from $\Re(v_i)$ (in S), then increase $Consistency_{cost}(\Re, \langle S, T\rangle)$ by 1;
- if v_j is **not** reachable from v_i (in T) and $\Re(v_j)$ **is** reachable from $\Re(v_i)$ (in S), then increase $Consistency_{cost}(\Re, \langle S, T\rangle)$ by 1;

We say $\Re$ has higher priority if it has lower cost. ■

In addition to the two structural heuristics, we also use **Coverage Priority** which prioritizes renamings that contain translations for (or cover) more target elements.

Definition 3.4. (Coverage Priority) For a skeleton $\langle S, T\rangle$, Coverage cost of $\Re$, $Coverage_{cost}(\Re, \langle S, T\rangle)$, is the number of variable in T that are mapped to ϵ. We say $\Re$ has higher priority if it has lower cost. ■

4. EVALUATION

We begin by discussing current benchmarks for knowledge exchange and comparing Kensho with other KGMTs in handling the scenarios in these benchmarks. Then, in Section 4.2, we showcase some of the most important real-world applications of KB translation and show the effectiveness of Kensho in these contexts. In Section 4.3, we use 50 synthesized settings to stress test the performance of Kensho. Note that we use a version of Kensho which creates renamings in which each source variable is assigned to at most one target variable. This strategy significantly reduces the number of possible renamings produced, at a cost of not being able to produce some valid renamings in rare cases. We have also done a small case study which compared the results of data engineers manually writing mapping rules vs. selecting, using data examples, mappings generated by Kensho.

4.1 Benchmark Evaluation

STBenchmark [4] introduced the use of (micro) scenarios for comparing data exchange systems that use a structured or semi-structured model. This idea was generalized by the meta-data generator iBench [12] that permits the efficient creation of benchmarks with large and complex schemas and data exchange scenarios. DTSBenchmark [61] provides a set of scenarios where the source and target are both KBs. These scenarios were later refined in LODIB (linked open data integration benchmark) [66] which is mainly designed to benchmark the expressive power of mapping languages. Both Mosto and Kensho can automatically generate the desirable mapping rules for all the scenarios proposed in DTSBenchmark. In addition, it is reported that queries generated by Mosto can support the expression of all fifteen LODIB scenarios except for three (the ones which need conditional clauses or aggregation queries) [66]. Queries generated by Kensho have the same expressive power. Kensho, like Mosto, cannot automatically learn the relationship between values (e.g., when a value must be transformed using function `usDollarsToEuros`), and thus it cannot automatically generate the mapping rules for the LODIB scenarios that require these types of transformations.

Additional scenarios have recently been identified that need to be supported by KMGTs [30]. Two of these scenarios are inspired by value invention in relational data exchange.

A third involves handling source KBs that are incomplete. A fourth involves mapping creation even when the set of given correspondences are incomplete (a very common case in practice). Finally, a fifth scenario involves mappings that use cyclic property paths. Kensho can handle all five scenarios while the previous approaches (Mosto and Qin et al.) cannot. It is important to note that since the associations that existing KMGTs create are a subset of what Kensho creates, and since their mapping language is not more expressive than Kensho's, there would be no scenario in which these KMGTs can create a mapping which Kensho cannot.

4.2 Knowledge Translation Usecases

We now investigate the effectiveness of mapping generation and ranking strategies implemented in Kensho using several scenarios that showcase some of the important applications of KB translation. Note that it is important to evaluate Kensho on real scenarios because all mappings which are created by our algorithm are valid interpretations of the set of correspondences, however, depending on the context, some mappings may be more desirable than others. In all settings presented, an expert identified the desired (gold) mapping rule in the selected skeleton, henceforth called $\Re_{gold}$. For each scenario (described below), in Table 1, column baseline reports the number of mappings which are baseline valid, column c2a reports the number of mappings which are baseline and *c2a* valid, column r2r reports the number of mappings which are baseline and *r2r* valid , and column Kensho shows number of mappings that adhere to all of our validity constraints. Table 1 also shows the effect of our ranking methods in facilitating the selection of the best set of mappings. For the different ranking strategies, we report on the rank of $\Re_{gold}$ in the Kensho rankings, and since our rankings can have ties, we report on the number of other mappings ranked better than $\Re_{gold}$, then the number of mappings ranked equal to $\Re_{gold}$ (cols. 9-12). For example, 2-1-0 means $\Re_{gold}$ was ranked second, with only one other mapping ranked ahead of $\Re_{gold}$ and with no ties. The last column of Table 1 reports the similarity between the two KBs (informally, the number of resources that model the same real world entities).

4.2.1 Populating a KB using Open Data

We believe the most important application of Kensho is populating an existing domain-specific ontology using other currently available structured data sources where the data source does not have to be a KB as long as it can be automatically converted to one. Expertise finding is one area for which having a domain-specific KB can be very beneficial [37, 53]. For this reason, there are carefully designed ontologies in the literature that model expertise for competency management. In this setting, for our target KB, we use one of these ontologies [26] and augment it with two more concepts, `Employee` and `Organization`, to be able to model the skills of employees in various companies. Our goal in this scenario (Row 1 of Table 1) is to generate mapping rules that can be used to populate this KB using open data published by the US Patent and Trademark Office (USPTO). To use the USPTO corpus as our source KB, we started with a subset of the USPTO's patent XML corpus [2] and automatically created a linked data corpus from it using Xcurator [77] and further enriched it using Vizcurator [29]. The original set of correspondences was provided by a domain expert and then was verified by the crowd using the protocol proposed by Sarasua et al. [67].

Our two KBs mainly model different domains (one models skills in an organization and the other models patent applications), so the similarity between these two KB is low with only 2.6% of the concepts and properties modeling the same information (last column of Table 1). Additionally, there are two correspondences between concepts and only one between attributes (col. 4). Mosto and Qin et al. [58] cannot find the gold mapping which connects the associated concepts and attributes through properties with no correspondences. Kensho produces two possible mappings including the gold mapping.

One side benefit of our approach is that the results can be used to enrich the alignment. For instance, in this scenario $\Re_{gold}$ suggests that a Rel2Rel correspondence may exists between the source `Patent - Inventor` path and the target `hasSkill - Skills` path.

4.2.2 Migrating Data between KBs

This example (Row 2 in Table 1) is from Mosto [63], the source is a portion of DBpedia version 3.2 and the target is the similar portion of DBpedia version 3.6. In order to generate the mapping rules in this scenario, Mosto requires the correspondences and also a user-provided axioms called `mosto:strongRange`. Note that Kensho does not require users to specify axioms such as these to guide the algorithm. In Mosto, if two source concepts (s_1 and s_2) have correspondences to two target concepts (t_1 and t_2) then these concepts will only be included in the same mapping if both the source and the target concepts are in a subclass relation, or if they are the domain and range of an *aligned* property, or if a data engineer has manually added a `mosto:strongRange` or `mosto:strongDomain` relation between them. In this scenario, without the annotation, Mosto would exchange the `Actor` and `AcademyAward` data in DBpedia, but not the relationship between these two concepts (representing about 850 facts indicating who won which award).

In this scenario, Kensho generates four mappings. If we do not use our r2r strategy, then we generate seven mappings. (including some that are inconsistent with the Rel2Rel correspondences). The four mappings that Kensho produces include $\Re_{gold}$ and another three that are less complete (containing some existential target variables). Our **Coverage** ranking strategy correctly ranks the gold mapping before these more incomplete mappings.

4.2.3 Enriching the Result of Ontology Alignment

In this scenario (Row 3), we highlight the fact that Kensho can enrich the rules produced by the current state-of-the-art alignment tools. Note that while alignment tools generate correspondences, Kensho enriches these correspondences and produces *executable queries* by interpreting the correspondences collectively. Previous to this work, KMGTs assumed that the set of property correspondences are complete. Thus, if there is no correspondence to or from an object property, the KMGT did not automatically consider any path that included that property. One of the main advantages of Kensho is that it considers such properties in mapping generation. To show the benefit of this approach, in this experiment, we have used two KBs from the OAEI (Ontology Alignment Evaluation Initiative) campaign and the correspondences between them which were produced by AML (Agreement Maker Light) – one of the best performers

Table 1: **Mapping Generation and Ranking Performance. Generation time is not mentioned if it is < 1 sec.**

Sec	Source	Target	Corr	Number of mapping generated				Effect of Ranking $\Re_{gold}$'s rank–# $\Re$s ranked better–# $\Re$s ranked equla				Similarity
				baseline	r2r	c2a	Kensho	Coverage	Path	Consistency	Path + Consistency	
4.2.1	#classes(C): 64 #Attributes(A):67 #objectProp(P):69	C: 10 A:2 P:11	#c2c: 2 #atr2atr: 1 #rel2rel: 0	2	2	2	2	1-0-0	2-1-0	1-0-1	2-1-0	Low (2.6%)
4.2.2	C: 5 A:2 P:2	C: 6 A:3 P:3	#c2c: 5 #atr2atr: 2 #rel2rel: 2	7	4	7	4	1-0-0	1-0-3	1-0-3	1-0-3	High (91%)
4.2.3	C: 38 A: 23 P: 13	C: 49 A:11 P:17	#c2c: 4 #atr2atr: 1 #rel2rel: 0	26	26	26	26	2-6-11	1-0-25	3-15-3	3-15-3	Low (10%)
4.2.4	C: 6 A: 12 P: 5	C: 5 A: 14 P: 6	#c2c: 4 #atr2atr: 7 #rel2rel: 1	407×10^5 creation time: $\sim$ 2.5 hr	65700 creation time: $\sim$ 10 min	4015	657	1-0-447	1-0-20	1-0-15	1-0-3	High (81%)

in the 2018 matching challenge [25]. The KBs we have used, *confTool* and *Sigkdd*, are from the OntoFarm dataset [78] of the *Conference* track of the OAEI.

In this scenario, Kensho produces 26 mappings (including the gold mapping). These two KBs are not very similar, so our structural rankings are not helpful. For example, our **Path** ranking ranked all 26 mappings equally. Our **Consistency** ranking did better with only eighteen mappings ranked equal or higher than the gold mapping. Our results suggest some interesting possible correspondences between two KBs which are not provided in the results of AML, for instance, inverse property correspondences such as:

`sigkdd:submit`$^\wedge \rightsquigarrow_{r2r}$ `conftool:writtenBy`

In this scenario, the number of given property correspondences is very low. As a result, existing KMGTs do not create the gold mapping.

4.2.4 Translating KBs with Cycles

In this scenario (Row 4), we show that unlike existing KMGTs, Kensho can handle cycles, a situation that occurs in many existing KBs such as those created from social networks. Cycles define associations between resources of the same type such that multiple resources in the source can be mapped into multiple resources in the target, a situation resulting in a large number of possible mappings. This scenario showcases that our mapping generation and ranking strategies are especially useful in cases where the number of possible interpretations becomes very large due to the existence of cyclic paths.

This setting (depicted in Figure 2) was created from the example proposed by Qin et al. [58] which was developed using Carnegie Mellon's *Person & Employee Ontology* and the University of Maryland's *People Ontology*. To add complexity, we incorporated a cycle by adding an additional object property, `src:related`, which has `src:Employee` as its domain and range. This caused a large number of mappings to be created. However, our mapping generation validity strategies reduce the number of mappings to a few hundred (which is still unmanageable for a human), but the **Consistency** ranking ranks the gold mapping first, in a tie with only 15 other mappings (a much more reasonable task for an engineer to understand). Together, our Path and Consistency ranking reduces the ties to only three.

4.3 Performance Evaluation

Mapping generation tools are usually evaluated using metadata generators [4, 12]. They allow the data engineer to systematically vary specific parameters that influence the difficulty of relational mapping creation or data exchange. In a similar vein, MostoBM [62] identifies three

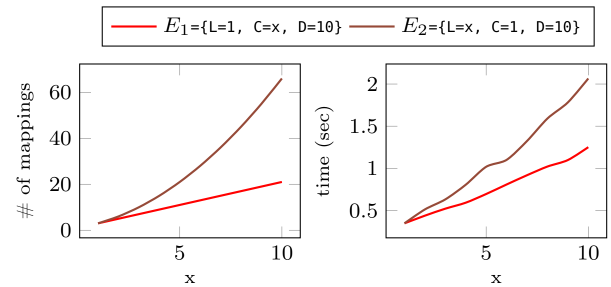

Figure 7: Increasing Breadth(C) E_1 & Depth(L) E_2.

schema-level parameters (namely, L **or depth** of the class relationships, C **or breadth** of the class relationships, and D, the number of attributes) that can affect the complexity of the task of KB mapping generation. To investigate the effect of breadth, we followed the MostoBM approach and generated ten settings for one of the MostoBM exchange scenarios - called *sink properties*. More specifically, we fixed $D = 10$, $L = 1$, and vary the value of C between one and ten. We call this group of settings that vary C, E_1. For the first setting of E_1, `{L=1, C=1, D=10}`, the source contains two concepts, $\mathtt{A}_0$ and its child $\mathtt{A}_1$, and $\mathtt{A}_0$ is the domain of ten attributes $\{\mathtt{d}_1,\ldots,\mathtt{d}_{10}\}$. The target has the same structure, except that the domain of $\mathtt{d}_1$ is $\mathtt{A}_1$. Resources with the same labels correspond to each other (e.g., `src:A`$_0 \rightsquigarrow_{c2c}$ `trgt:A`$_0$). For the next setting in E_1, `{L=1, C=2, D=10}`, $\mathtt{A}_0$ has one more child, $\mathtt{A}_2$, in both the source and target and the domain of $\mathtt{d}_2$ in the *target* is $\mathtt{A}_2$. We repeated the same procedure for investigating the effect of depth and created a group of ten depth settings, E_2 by keeping C constant and varying L. The difference here is that the new concept for each consecutive setting will be nested inside the most specific type (as opposed to being added to the root). For instance, when $L = 2$, $\mathtt{A}_2$ is a child of $\mathtt{A}_1$ and $\mathtt{A}_1$ is a child of $\mathtt{A}_0$. In this section, we report on the number of *all* mappings created (as opposed to the number of mappings created for a specific skeleton as was done in the previous section). Note that changing D while fixing L and C will not change the number of mappings created and so we did not include such settings in our evaluation.

Figure 7 shows the number of mappings generated by Kensho in each of the settings. Note that Mosto does not create all possible mappings and thus in our evaluation we could not compare it with Kensho. As the breadth (C) of the KB increases, the number of mappings generated also increases linearly and Kensho's performance also scales linearly, remaining under a second for a setting with breadth of 8.

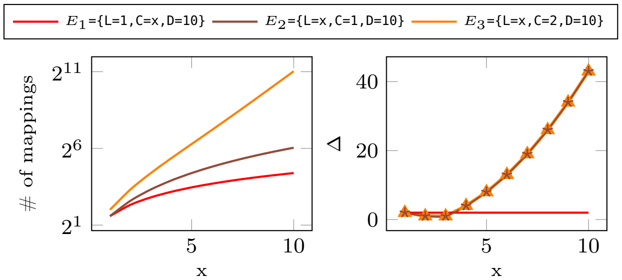

Figure 8: Increasing Breadth(C) E_1 & Depth(L) E_2 and E_3. Δ = #mappings - #correspondences

In contrast, increasing the depth (L) has an exponential effect on the number of mappings generated. This is expected since increasing the depth actually adds a layer of nesting. Kensho's performance passes one second at a depth of five which corresponds to the generation of 21 mappings.

Settings in E_1 and E_2 include relatively small KBs. To push Kensho further, we created a set of 10 settings, E_3, for the same scenario using parameters `{L=x, C=2, D=10}`, where $1 \leq$ `x` ≤ 10. The taxonomy created using these parameters contains 2^n concepts at depth `n` from the root. For instance, when `L = 10`, both the source and target contain 2^{10} most specific types (in total each contains more than 2000 concepts - more than three times greater than the number of concepts in DBpedia). Figure 8 left, shows the number of mappings created in E_3 vs. E_1 and E_2, note the scale change on the Y-axis. When `L = 10`, Kensho creates more than 2000 mappings and the creation time is almost three hours. One factor that affects the number of mappings is the number of correspondences. In E_3, the number of correspondences grows as the depth L increases. To explain this effect, we have also plotted Δ as the difference between the number of mapping rules generated for each setting and the number of correspondences. The right plot in Figure 8 shows the result. Obviously, the difference Δ is constant for the E_1 (breadth) settings (all have $L = 2$). However it is important to note that for E_2 and E_3, Δ is exactly the same for each value of the depth L. This is because the number of possible mappings becomes greater than number of correspondences if there are various ways for interpreting correspondences. As we have seen, correspondences can be interpreted in various ways only if the resources can be associated with each other in multiple ways. In this scenario, multiple interpretations are the result of various ways that one can interpret Concept2Concept correspondences in conjunction with Attr2Attr correspondences. Thus increasing only the number of concepts and Concept2Concept correspondences (as we are doing here) will not result in multiple interpretations of the correspondences.

Nonetheless, Kensho is sensitive to the number of possible interpretations of a correspondence, so we created a final set of 20 settings (E_4) to understand how far we can push Kensho on this dimension. We have used the setting represented in Figure 2, our running example which already includes multiple interpretations, as our least complex setting (Setting 1). To create the rest of the settings, we injected elements into the source, such that in each setting, the number of possible assignments for each target variable is increased by one. For instance, Figure 6 shows that there are three source variables of type `country` that can be assigned to variable `tC1`. The next setting contains four possibilities for this variable, and so on. Figure 9 represents the result of our various mapping strategies. The number of mappings generated grows exponentially; however, this figure demonstrates that Kensho is able to reduce this search space by only generating valid mappings. In our largest setting, the source KB contains nearly 120 object properties and 5 concepts leading to over 500K interpretations (we have kept the target fixed in this experiment). Only one fifth of these properties corresponds to a property in the target. Note that real world KBs usually contain many fewer object properties among concepts. Kensho created 4579 possible mappings for this setting. For the biggest skeleton in this setting, Kensho ranked 42 mappings as rank one. This highlights one of the weaknesses of Kensho. As the number of *property paths* among corresponding concepts or attributes grows, the number of mappings generated by Kensho will grow exponentially and this makes the process of selecting the best set of mappings overwhelming. However, note that in the previous experiment we have shown that Kensho is not as sensitive to the increasing number of concepts, and most large real world KBs such as medical KBs tend not to have a large number of properties for all concepts as we do in this case (which included an average of 24 properties for every concept in this experiment).

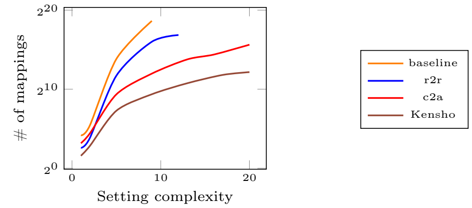

Figure 9: Effect of increasing # of interpretations.

In summary, we performed experiments on synthetic scenarios (inspired by existing KB exchange benchmarks [62]) to show how the performance of our mapping generation algorithm is affected by increasing complexity of the scenarios. Our results show that Kensho scales very well as the KB size increases, with the largest bottleneck being the number of possible interpretations of a correspondence.

4.4 Case Study

In this study, we make use of a scenario from OntoMerge [19]. It includes their Yale bibliography ontology as our source, and their CMU bibliography ontology as our target. We also use their manually curated mapping rules as our gold standard. We manually identified the set of correspondences between these two ontologies (from the gold standard mapping rules). These ontologies are very simple, the source contains only 6 classes and no object properties, the target contains 9 classes and 3 object properties. In Phase I of our experiment, we asked two collaborators, *c1* and *c2* (both experienced in SPARQL) to create mapping rules by manually writing SPARQL queries. We gave each collaborator 20 minutes to familiarize themselves with the ontologies and one hour to write the mapping rules. The queries created by *c1* were able to transfer 78% of the facts while queries created by *c2* transferred 61% of the facts. In Phase II, we used a ranked list of examples created by Kensho and gave *c1* and *c2* each 30 minutes to go through them and choose desirable examples. Note that examples were created based on query solutions obtained by running queries using query pattern of each renaming. Both *c1* and *c2* identified all (and only) the desirable examples. In Phase III we gave them 20 minutes to update the queries they generated in Phase I, based on what they learned from the examples. Collaborator *c2* was able to write all the queries correctly while *c1* was able to make changes such that 92% of the facts translated correctly. It is possible that *c1* and *c2* were able to more easily identify examples from Kensho's ranked list because in Phase I they had each tried to generate their own mapping rules. To account for this, we asked a third collaborator *c3* (also with SPARQL experience) to first select desirable examples from Kensho's automatically generated list. Similar to *c1* and *c2*, *c3* was given 30 minutes and was also able to pick the desirable (and only desirable) examples. We then gave *c3* one hour to write queries that can transfer data from the source to the target. The resulting queries were able to transfer only 78% of the facts.

In summary, all participants were able to correctly identify the desirable examples (including the correct optional combinations) which highlights the fact that it is more intuitive for humans to select from examples than to write queries manually.

5. RELATED WORK

Among mapping generation tools only Maponto [7], which maps between a single relational table to a KB, traverses all paths of the target ontology to find semantic associations among concepts (as Kensho does for both source and target). Even to map a single table, Maponto relies heavily on the existence of enriched ontological constraints (such as cardinality) to narrow down the search over all paths. These constraints are rarely present in real KBs.

The first step of any data sharing task is **alignment**, which is the task of finding a set of suitable correspondences between the source and the target. The second step involves interpreting a set of candidate correspondences *collectively* to solve a specific data sharing task (e.g., data exchange) [20]. The output of most alignment tools (Step 1), are simple correspondences, each specifying that a resource in the source (or multiple resources like a path) has some set-theoretic relationship to a resource (resources) in the target. We take this output (from Step 1) as our input and find queries that collectively interpret correspondences. Ontology and schema structure have been used extensively in the first step [18; 21; 39; 40; 44; 45; 47; 51; 54, et al.]. In contrast to these approaches, we assume the correspondences are given as *input* and use the semantics and structure of the KB to create (often complex) data exchange queries that can be used to correctly translate a full source instance into a full target instance without losing or changing the semantics of the data and the way values are connected. Note that although many heuristics for ranking correspondences are proposed [20], they are not applicable here since they are mostly designed to express the degree of similarity of two resources and cannot rank the complex queries discovered by Kensho. Nevertheless, the aggregation methods used to combine various integration rankings [27, 59] could potentially provide insight on aggregating the rankings of the Kensho mappings.

Similar to our approach, rule mining approaches [28, 48, 52]) traverse a knowledge graph to associate a set of resources, but they do this guided by a set of positive and/or negative examples. We aim to find rules among corresponding resources of two different KBs, so we find associations in the source, associations in the target, and then we create our rules by combining these associations using the constraints imposed by the correspondences. On the other hand, the discovery phase of most rule mining algorithms try to fit a set of given examples. It would be interesting to investigate how rule mining approaches can help in enriching the set of constraints that we use in order to further refine our mapping rules. It is also interesting to see how these rules can help in ranking the mapping rules by helping to find *more important* paths which fit certain sets of examples.

6. CONCLUSION

We introduced Kensho, a tool that translates knowledge between two KBs by generating mapping rules between them. Kensho is the first KB mapping tool that is effective even when there are missing object property correspondences. Kensho is also the first to take advantage of correspondences between object or data paths, if they are available, though our approach also allows these to be incomplete. Kensho improves upon existing methods by producing mappings that perform value invention in a principled way without assuming a complete source KB.

We are currently extending Kensho in the spirit of the *integration by example paradigm* [76]. A large body of traditional data exchange literature is dedicated to identifying or exploiting examples that can shown to data engineers and incorporating their feedback [3, 5, 15, 17, 33, 41, 57, 76]. Similarly, we are working on approaches for incorporating user feedback to improve upon our mapping rules.

When source or target instances are available, Kensho can simplify the semantic associations by running queries which are built using the graph pattern representation of the semantic association. We can remove the parts of the query in which the variables are not being bound. In this mode, `OPTIONAL` keywords can be removed if adding them to the query does not exchange additional facts. Note that all of our experiments are performed without this feature. Although using the above technique can help, the queries generated by our tool might still contain a considerable number of `OPTIONAL` clauses. Thus, we believe our approach can benefit from research on how to optimize the execution of SPARQL `OPTIONAL` queries [75].

As part of our ongoing work, we are exploring ideas on dealing with scale using slicing and we plan to investigate whether sophisticated methods such as modularization [32, 34, 35, 43, 72] or partitioning [36, 69, 71] may help in dealing with knowledge translation between large KBs. We will also consider how the process of mapping generation can be refined in the presence of other constraints such as functionality or cardinality [24]. Ontology based data access initiatives (OBDA) [74] facilitate the exchange and integration of data between a relational source and a target KB. It is interesting to see how our approach can be adopted in such settings.

7. ACKNOWLEDGMENTS

This research was funded in part by an NSERC Strategic Partnership Grant.

8. REFERENCES

[1] The linked open data cloud. https://lod-cloud.net/, Accessed on 03.24.2020.

[2] Uspto bulk downloads: Patent grant full text. https://tny.sh/USPTO, Accessed on 03.24.2020.

[3] B. Alexe, L. Chiticariu, R. J. Miller, and W.-C. Tan. Muse: Mapping understanding and design by example. In *ICDE*, pages 10–19, 2008.

[4] B. Alexe, W.-C. Tan, and Y. Velegrakis. STBenchmark: towards a benchmark for mapping systems. *PVLDB*, 1(1):230–244, 2008.

[5] B. Alexe, B. ten Cate, P. G. Kolaitis, and W.-C. Tan. Designing and refining schema mappings via data examples. In *SIGMOD*, pages 133–144, 2011.

[6] F. Alkhateeb, J.-F. Baget, and J. Euzenat. Extending sparql with regular expression patterns (for querying rdf). *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(2):57–73, 2009.

[7] Y. An, A. Borgida, and J. Mylopoulos. Discovering the semantics of relational tables through mappings. In *Journal on Data Semantics VII*, pages 1–32. Springer, 2006.

[8] M. Arenas, E. Botoeva, and D. Calvanese. Knowledge base exchange. In *DL*, volume 4, 2011.

[9] M. Arenas, E. Botoeva, D. Calvanese, and V. Ryzhikov. Knowledge base exchange: The case of OWL 2 QL. *Artificial Intelligence*, 238:11–62, 2016.

[10] M. Arenas, E. Botoeva, D. Calvanese, V. Ryzhikov, and E. Sherkhonov. Exchanging description logic knowledge bases. In *KR*, 2012.

[11] M. Arenas, J. Pérez, and J. Reutter. Data exchange beyond complete data. *JACM*, 60(4):28, 2013.

[12] P. C. Arocena, B. Glavic, R. Ciucanu, and R. J. Miller. The iBench integration metadata generator. *PVLDB*, 9(3):108–119, 2015.

[13] C. Bizer and A. Schultz. The R2R framework: Publishing and discovering mappings on the web. In *COLD*, pages 97–108, 2010.

[14] A. Bonifati, G. Mecca, P. Papotti, and Y. Velegrakis. Discovery and correctness of schema mapping transformations. In *Schema matching and mapping*, pages 111–147. 2011.

[15] A. Bonifati, G. Mecca, A. Pappalardo, S. Raunich, and G. Summa. Schema mapping verification: The spicy way. In *EDBT*, pages 85–96, 2008.

[16] L. Bühmann and J. Lehmann. Universal OWL axiom enrichment for large knowledge bases. In *EKAW*, pages 57–71, 2012.

[17] B. T. Cate, P. G. Kolaitis, K. Qian, and W.-C. Tan. Approximation algorithms for schema-mapping discovery from data examples. *TODS*, 42(2):1–41, 2017.

[18] I. F. Cruz, F. P. Antonelli, and C. Stroe. Agreementmaker: Efficient matching for large real-world schemas and ontologies. *PVLDB*, 2(2):1586–1589, 2009.

[19] D. Dou, D. McDermott, and P. Qi. Ontology translation on the semantic web. In *Journal on data semantics II*, pages 35–57. 2005.

[20] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, 2nd edition, 2013.

[21] J. Euzenat and P. Valtchev. Similarity-based ontology alignment in OWL-Lite. In *ECAI*, pages 323–327, 2004.

[22] R. Fagin, L. Haas, M. Hernández, R. Miller, L. Popa, and Y. Velegrakis. Clio: Schema mapping creation and data exchange. *Conceptual modeling: foundations and applications*, pages 198–236, 2009.

[23] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[24] W. Fan and P. Lu. Dependencies for graphs. *TODS*, 44(2):5:1–5:40, 2019.

[25] D. Faria, C. Pesquita, B. S. Balasubramani, T. Tervo, D. Carriço, R. Garrilha, F. M. Couto, and I. F. Cruz. Results of AML participation in OAEI 2018. In *OM*, pages 125–131, 2018.

[26] M. Fazel-Zarandi and M. S. Fox. An ontology for skill and competency management. In *FOIS*, pages 89–102, 2012.

[27] A. Gal, H. Roitman, and R. Shraga. Heterogeneous data integration by learning to rerank schema matches. In *ICDM*, pages 959–964, 2018.

[28] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE. *VLDB J.*, 24(6):707–730, 2015.

[29] B. Ghadiri Bashardoost, C. Christodoulakis, S. Hassas Yeganeh, O. Hassanzadeh, R. J. Miller, and K. Lyons. Vizcurator: A visual tool for curating open data. In *WWW Companion*, pages 195–198, 2015.

[30] B. Ghadiri Bashardoost, R. J. Miller, and K. Lyons. Towards a benchmark for knowledge base exchange. In *DI2KG*, 2019.

[31] B. Ghadiri Bashardoost, R. J. Miller, K. Lyons, and F. Nargesian. Knowledge translation: Extended technical report. Technical report, University of Toronto, 2019. Available at https://tny.sh/KBTranslation.

[32] A. Ghazvinian, N. F. Noy, and M. A. Musen. From mappings to modules: Using mappings to identify domain-specific modules in large ontologies. In *K-CAP*, pages 33–40, 2011.

[33] G. Gottlob and P. Senellart. Schema mapping discovery from data instances. *J. ACM*, 57(2):6:1–6:37, Feb. 2010.

[34] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: Extracting modules from ontologies. In *WWW*, pages 717–726, 2007.

[35] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: Theory and practice. *JAIR*, 31:273–318, 2008.

[36] B. C. Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Automatic partitioning of OWL ontologies using E-Connections. *Description Logics*, 147, 2005.

[37] M. T. Hansen and N. Nohria. How to build collaborative advantage. *MIT Sloan Management Review*, 46(1):22, 2004.

[38] M. A. Hernández, P. Papotti, and W.-C. Tan. Data exchange with data-metadata translations. *PVLDB*, 1(1):260–273, 2008.

[39] W. Hu, N. Jian, Y. Qu, and Y. Wang. Gmo: A graph matching for ontologies. In *Integrating Ontologies*, pages 41–48, 2005.

[40] W. Hu, Y. Qu, and G. Cheng. Matching large ontologies: A divide-and-conquer approach. *Data & Knowledge Engineering*, 67(1):140–160, 2008.
[41] A. Kimmig, A. Memory, R. J. Miller, and L. Getoor. A collective, probabilistic approach to schema mapping using diverse noisy evidence. *TKDE*, 2018.
[42] N. Kolbe, S. Kubler, J. Robert, Y. Le Traon, and A. Zaslavsky. Linked vocabulary recommendation tools for internet of things: a survey. *CSUR*, 51(6):1–31, 2019.
[43] B. Konev, C. Lutz, D. Walther, and F. Wolter. Model-theoretic inseparability and modularity of description logic ontologies. *Artificial Intelligence*, 203:66–103, 2013.
[44] J. Li, J. Tang, Y. Li, and Q. Luo. Rimom: A dynamic multistrategy ontology alignment framework. *TKDE*, 21(8):1218–1232, 2009.
[45] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *VLDB*, volume 1, pages 49–58, 2001.
[46] B. Marnette, G. Mecca, P. Papotti, S. Raunich, D. Santoro, et al. ++ spicy: an open-source tool for second-generation schema mapping and data exchange. *PVLDB*, 19(12):1438–1441, 2011.
[47] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pages 117–128, 2002.
[48] C. Meng, R. Cheng, S. Maniu, P. Senellart, and W. Zhang. Discovering meta-paths in large heterogeneous information networks. In *WWW*, pages 754–764, 2015.
[49] R. J. Miller. Using schematically heterogeneous structures. In *SIGMOD*, pages 189–200, 1998.
[50] R. J. Miller, L. M. Haas, and M. A. Hernández. Schema mapping as query discovery. In *VLDB*, pages 77–88, 2000.
[51] N. F. Noy and M. A. Musen. Anchor-prompt: Using non-local context for semantic matching. In *Proceedings of the IJCAI workshop on ontologies and information sharing*, pages 63–70, 2001.
[52] S. Ortona, V. V. Meduri, and P. Papotti. Robust discovery of positive and negative rules in knowledge bases. In *ICDE*, pages 1168–1179, 2018.
[53] G. Paquette. An ontology and a software framework for competency modeling and management. *Educational Technology & Society*, 10(3):1–21, 2007.
[54] C. Pinkel, C. Binnig, E. Kharlamov, and P. Haase. Incmap: Pay as you go matching of relational schemata to OWL ontologies. In *OM*, pages 37–48, 2013.
[55] A. Polleres, F. Scharffe, and R. Schindlauer. SPARQL++ for mapping between RDF vocabularies. In *OTM*, pages 878–896, 2007.
[56] L. Popa, Y. Velegrakis, M. A. Hernández, R. J. Miller, and R. Fagin. Translating web data. In *VLDB*, pages 598–609, 2002.
[57] L. Qian, M. J. Cafarella, and H. V. Jagadish. Sample-driven schema mapping. In *SIGMOD*, pages 73–84. ACM, 2012.
[58] H. Qin, D. Dou, and P. LePendu. Discovering executable semantic mappings between ontologies. *OTM*, pages 832–849, 2007.
[59] A. Radwan, L. Popa, I. R. Stanoi, and A. A. Younis. Top-k generation of integrated schemas based on directed and weighted correspondences. In *SIGMOD*, pages 641–654, 2009.
[60] C. R. Rivero, I. Hernández, D. Ruiz, and R. Corchuelo. Generating SPARQL executable mappings to integrate ontologies. In *ER*, pages 118–131, 2011.
[61] C. R. Rivero, I. Hernández, D. Ruiz, and R. Corchuelo. On benchmarking data translation systems for semantic-web ontologies. In *CIKM*, pages 1613–1618, 2011.
[62] C. R. Rivero, I. Hernandez, D. Ruiz, and R. Corchuelo. Benchmarking data exchange among semantic-web ontologies. *TKDE*, 25(9):1997–2009, 2012.
[63] C. R. Rivero, I. Hernández, D. Ruiz, and R. Corchuelo. Exchanging data amongst linked data applications. *Knowledge and Information Systems*, 37(3):693–729, 2013.
[64] C. R. Rivero, I. Hernández, D. Ruiz, and R. Corchuelo. MostoDEx: A tool to exchange rdf data using exchange samples. *Journal of Systems and Software*, 100:67–79, 2015.
[65] C. R. Rivero, I. Hernández, D. Ruiz, and R. Corchuelo. Mapping RDF knowledge bases using exchange samples. *Knowledge-Based Systems*, 93:47–66, 2016.
[66] C. R. Rivero, A. Schultz, C. Bizer, and D. Ruiz Cortés. Benchmarking the performance of linked data translation systems. In *LDOW*, 2012.
[67] C. Sarasua, E. Simperl, and N. F. Noy. Crowdmap: Crowdsourcing ontology alignment with microtasks. In *ISWC*, pages 525–541, 2012.
[68] A. Schultz, A. Matteini, R. Isele, C. Bizer, and C. Becker. Ldif-linked data integration framework. In *COLD*, pages 125–130, 2011.
[69] J. Seidenberg and A. Rector. Web ontology segmentation: Analysis, classification and use. In *WWW*, pages 13–22, 2006.
[70] SPARQL. SPARQL 1.1 Query Language: W3C Recommendation 21 March 2013. https://www.w3.org/TR/sparql11-query/, 2013.
[71] H. Stuckenschmidt and M. Klein. Structure-based partitioning of large concept hierarchies. In *ISWC*, pages 289–303, 2004.
[72] H. Stuckenschmidt, C. Parent, and S. Spaccapietra. *Modular ontologies: concepts, theories and techniques for knowledge modularization*, volume 5445. Springer, 2009.
[73] G. Stumme and A. Maedche. Fca-merge: Bottom-up merging of ontologies. In *IJCAI*, pages 225–230, 2001.
[74] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyaschev. Ontology-based data access: A survey. In *IJCAI*, pages 5511–5519, 2018.
[75] G. Xiao, R. Kontchakov, B. Cogrel, D. Calvanese, and E. Botoeva. Efficient Handling of SPARQL optional for OBDA. In *ISWC*, pages 354–373, 2018.
[76] L. L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. Data-driven understanding and refinement of schema

mappings. In *SIGMOD*, pages 485–496, 2001.

[77] S. H. Yeganeh, O. Hassanzadeh, and R. J. Miller. Linking semistructured data on the web. 2011.

[78] O. Zamazal and V. Svátek. The ten-year ontofarm and its fertilization within the onto-sphere. *Journal of Web Semantics*, 43:46–53, 2017.