# Learning signatures for ontology relations

Luis Galárraga

February 28, 2016

## 1 Introduction

Schema learning is the process of inducing a schema definition from a collection of data. In the context of ontologies and knowledge bases (KBs), the schema of a KB, often referred as the T-Box or *terminological component*, consists of a declaration of concepts that are relevant to the ontology. For instance, an ontology designed to represent information about locations, would require to define formalisms for countries and cities. Such declarations consists of statements, e.g., *Country is a class of resources* or *Countries trade with other countries*. The languages RDFs[1] and OWL[2], provide a framework to express these statements in a formal manner:

1. `rdf:type(Country, rdfs:Class)`
2. `rdfs:domain(dealsWith, Country)`
3. `rdfs:range(dealsWith, Country)`
4. `rdf:type(dealsWith, owl:SymmetricProperty)`

The first statement declares the concept of countries, while the second and third statements convey that `dealsWith` is a relationship between instances of the class Country. The last statement declares the symmetry of the `dealsWith` relation.

The T-Box statements are conceived to govern the structure of the *assertion component* of the KB, usually abbreviated as the A-Box. The A-box contains the facts that describe the instances (resources in RDF) represented in the KB as well as their relationships with other instances. Typical statements of the A-box are: *France trades with Germany*, *France and Germany are countries*. In the RDF[3] formalism, these assertions can be expressed as the triples:

1. `rdf:type(France, Country)`
2. `rdf:type(Germany, Country)`
3. `dealsWith(France, Germany)`

[1] `https://www.w3.org/TR/rdf-schema/`
[2] `https://www.w3.org/TR/2012/REC-owl2-primer-20121211/`
[3] `https://www.w3.org/RDF/`

The problem of schema learning consists of inducing the T-box of a KB, given the statements in its A-box. In this particular report, I focus on the problem of learning relation signatures, namely `rdfs:domain` and `rdfs:range` statements, given a type hierarchy (T-box) and facts from the A-box. For ontologies with a complex class hierarchies, this problem is not trivial. To see this, consider our example KB in addition to the following T-box and A-box statements:

- `rdfs:subClassOf(Country, AdministrativeDivision)`
- `rdfs:subClassOf(UNCountry, Country)`
- `rdf:type(France, UNCountry)`
- `rdf:type(Île de France, AdminDivision)`
- `rdf:type(Germany, UNCountry)`
- `rdf:type(Kosovo, Country)`
- `dealsWith(Kosovo, Germany)`

`UNCountry` stands for country recognized by the United Nations. Imagine that we want to induce a schema for the relation `dealsWith`. In this simple type hierarchy, the choices are `AdministrativeDivision`, `Country` and `UNCountry`. I claim that `AdministrativeDivision` is too general because there are not occurrences of the relation with instances that exclusively belong to this class. In contrast, the class `UNCountry` is too specific because we would not be able to explain the triple `dealsWith(Kosovo, Germany)`. In reality, the list of choices can be much larger, e.g., in YAGO, some relations occur with hundreds of classes when the whole type deductive closure is considered. The next section elaborates on how to address the problem.

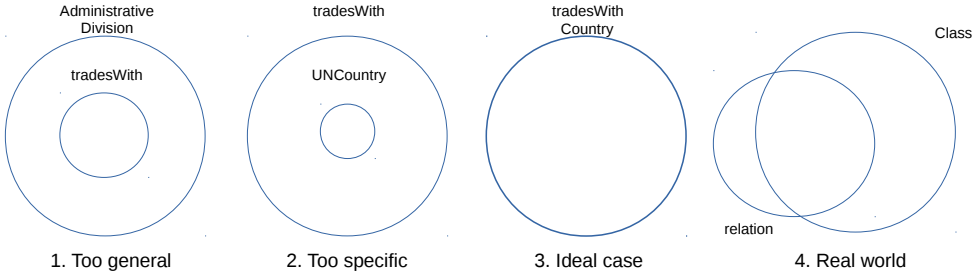
## 2 Relation signature learning

This work was motivated from my experience working with YAGO2s and Wikidata. While YAGO defines domains and ranges for relations, some of them are too general or fuzzy. Examples using the notation `r: domain` $\rightarrow$ `range` are:

- `livesIn: Person` $\rightarrow$ `Location`
- `dealsWith: Location` $\rightarrow$ `Location`
- `created: LegalActor` $\rightarrow$ `Thing`

This inaccuracy can be a hazzle for KB maintainers and users. For instance, there is no restriction at the schema level to accept the facts `livesIn(Barack Obama, NorthAmerica)`, `livesIn(Barack Obama, USA)`. Even though both facts are correct, data analytics on this relationship should take care of not counting these two values as two different places, but as granularity variants of the same fact. By learning the signatures for relations in YAGO, we can identify how the relations are actually being used in the data and aid the YAGO providers in improving the ontology schema.

In Wikidata, this work was motivated by the fact that I could not find explicit relation signatures for relations.




## 2.1 The approach

My simple approach for signature learning assumes we know the class hierarchy of the KB and that the instance information is complete. It is important to remark that the latter assumption may not hold. In [1], the authors show that the 43% of the entities in YAGO2s and 64% of the entities in DBpedia 3.7 have at least one type. Nevertheless, I use the class hierarchy to compute the type deductive closure for entities.

The rationale of the approach can be explained by Figure 2.1 and is equally applicable for learning both domains and ranges. I resort to the example introduced in Section 1 where we want to learn the domain of the relation `dealsWith`. In the first figure, the class seems too general as domain for `dealsWith` because most of the instances do not have a value for the relation (this uses a Closed World Assumption). Conversely, in the second figure, the relation occurs for all the values in the class but also in many others outside the class, thus the class is too specific. The third scenario illustrates the ideal case where the relation is perfectly aligned with the class as all countries have trade partners. The last scenario is what generally happens in reality due to multiple factors such incompleteness or semantic particularities in the relation. Notice that the ideal scenario depends on the relation. One would expect all countries to have trade partners, however we cannot extrapolate this behavior to other relations such as marriage or prizes. In such cases, the ideal scenario looks like the first figure, e.g., the class Person is a superset of those who are married.

Using the example from the last figure as general reference, we can say that a good candidate for the domain of the relation is the class that maximizes the ratio between the intersection of the sets and the sets themselves. In this way we maximize both (a) the evidence that the instances of the class have values for the relation (case 1 in Fig. 2.1) and (b) the exclusivity of the relation w.r.t the class (case 2 in Fig. 2.1). For a relation $r$ and a class $C$, this notion is captured by the jaccard coefficient:

$$jaccard(r, C) = \frac{\#x : \exists y : r(x, y) \wedge type(x, C)}{\#x : \exists y : r(x, y) \vee type(x, C)}$$

The class $C$ that maximizes the $jaccard$ function is said to be the best candidate for the domain of the relation. The same principle applies for learning the range of the relation:

| Relation | Signature | Domain | Range |
|---|---|---|---|
| dealsWith | Location → Location | Economy | Economy |
| livesIn | Person → Location | Scientist | AdminDistrict |
| isPoliticianOf | Person → USState | Governor | USState |
| playsFor | Person → Organization | FootballPlayer | EnglishTeam |
| hasWonPrize | GeoActor → Award | Person | Award |
| worksAt | Person → Organization | Scientist | University |
| created | LegalActor → Thing | LivingPeople | Album |

Table 1: Signatures learnt for some relations in YAGO2s

$$jaccard(r, C') = \frac{\#y : \exists x : r(x, y) \wedge type(y, C)}{\#y : \exists x : r(x, y) \vee type(y, C)}$$

It is important to remark that this simple approach learns the domains and ranges of relations separately. It does not guarantee that the instances of $C$ co-occur mostly with instances of $C'$ in the relation. Our jaccard coefficient could, however, be extended to analyze complete sub-relations inside a general relation. The joint version of the jaccard coefficient has the following formula:

$$joint\text{-}jaccard(r, C, C') = \frac{\#(x, y) : r(x, y) \wedge type(x, C) \wedge type(y, C')}{\#(x, y) : r(x, y) \vee type(x, C) \vee type(y, C')}$$

As for the simple Jaccard, the pair $C, C'$ with the highest score becomes the best candidate as signature for the relation. Nonetheless, some preliminary results in Wikidata suggest that the *joint-jaccard* metric delivers poor performance in terms of recall, i.e., the signatures with the best score normally covered a relatively small size of the relation. Still, the *joint-jaccard* could used to learn sub-properties of a relation if we consider the top-k best candidates. A different variant that normalizes the intersection to the size of the relation could also identify the most representative sub-relations of a given relation:

$$joint\text{-}jaccard'(r, C, C') = \frac{\#(x, y) : r(x, y) \wedge type(x, C) \wedge type(y, C')}{\#(x, y) : r(x, y)}$$

# 3 Experiments

I carried out some experiments in YAGO2s [2] and a dump of Wikidata [4] from December 2015. The results for some relations are shown in Table 1 and Table 2.

The cases in Table 1 correspond to the YAGO relations whose declared signatures are too general, even though the relation is only being used with more specific classes. For some relations, e.g., `livesIn`, `playsFor`, the proposed signature seems to specific, i.e., it is not feasible to directly change the ontology. In those cases, however, it makes sense to look at the ranking of classes proposed by the score. For instance, the class `Person` is the second best candidate class for the domain of the relation `created`. Moreover, the second best candidate

[4] `http://tools.wmflabs.org/wikidata-exports/rdf/index.php?content=dump_download.php&dump=20151228`

| Relation | Signature |
|---|---|
| citizenship | Human → Country |
| birthPlace | Human → City |
| residence | Human → City |
| govermentHead | Human → SpanishMunicipality |
| programmingLanguage | Application → ProgrammingLanguage |
| countryOfOrigin | Film → Country |
| employer | Human → University |

Table 2: Signatures learnt for some relations in Wikidata

for the range, corresponds to the class `Book`. This may suggest the existence of latent subrelations within the relation `created`.

Since I did not have any reference signatures for Wikidata, Table 2 shows the signatures for some common relations.

# References


[1] Aldo Gangemi, Andrea Giovanni Nuzzolese, Valentina Presutti, Francesco Draicchio, Alberto Musetti, and Paolo Ciancarini. Automatic typing of dbpedia entities. In *Proceedings of the 11th International Conference on The Semantic Web - Volume Part I*, ISWC'12, pages 65–81, Berlin, Heidelberg, 2012. Springer-Verlag.

[2] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence Journal*, 2013.