ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

Department of Computer Science and Engineering

Master Thesis in Artificial Intelligence

Selectional Restriction Extraction for Frame-Based Knowledge Graph Augmentation

Relatore:
Chiar.ma Prof.ssa
Valentina Presutti

Presentata da:
Michele Luca Contalbo

Sessione

Anno Accademico 2021/2022

To Giuseppe, Lucia and Andrea for being my backbone

To Michele, Teresa, Innocente and Carmela for the affection given to me

Introduction

The Semantic Web is an ambitious project aimed at creating a global, machine-readable web of data, with the goal of enabling intelligent agents to access and reason over this data. Ontologies are a key component of the Semantic Web, as they provide a formal description of the concepts and relationships in a particular domain. However, ontology development is often a time-consuming and labor-intensive process that requires significant domain expertise.
Some of the knowledge graphs publicly available are mostly *fact-based*, meaning that their focus is to represent factual data without a precise definition of the **knowledge graph schema** nor a precise documentation. This approach makes reasoning over data more difficult due to the lack of ontological axioms. One of the most relevant knowledge graphs showing these disadvantages is **Wikidata**.
Exploiting the expressiveness of knowledge graphs together with a more logically sound ontological schema can be crucial to represent *consistent* knowledge and to infer new relations over the data. In other words, constraining the entities and predicates of knowledge graphs leads to **improved semantics**.
The same benefits can be found for restrictions over **linguistic resources**, which are knowledge graphs used to represent natural language. More specifically, it is possible to specify constraints on the arguments that can be associated with a given frame, based on their semantic roles (**selectional restrictions**). However, most of the linguistic resources define **very**

general restrictions because they must be able to represent different domains. Hence, the main research question tackled by this thesis is whether the use of **domain-specific** selectional restrictions are useful for **ontology augmentation**, **ontology definition** and **neuro-symbolic tasks** on knowledge graphs.
To this end, we developed a tool able to empirically **estimate the types of classes** covering certain roles in a frame, together with the estimated probabilities. The new restrictions have been expressed in **OWL-Star**, a less verbose variant of the **Ontology Web Language** (OWL). We show that the obtained selectional restrictions can be used to provide better explanations of the ontologies, i.e. they can improve and automatize ontology documentation.
We also developed an **OWL-Star to OWL mapping** and proved theoretical properties about the translation. Specifically, we show that the mapping is *information preserving*, meaning that it does not lead to ambiguities if certain conditions hold. We apply the OWL-Star to OWL mapping to obtain the OWL representation of the selectional restrictions. Subsequently, we add the resulting OWL ontology with selectional restrictions to **Framester**, an open lexical-semantic resource for the English language.
We demonstrate the benefits of augmenting the knowledge graphs with selectional restrictions for **neuro-symbolic tasks**. In other words, we show that using selectional restrictions for the *class membership* task leads to better ranking scores, meaning that the calculated embeddings are qualitatively better with respect to the ones obtained without selectional restrictions. The domain used for applying these tools is the **MusicBO dataset**, a knowledge graph dataset automatically created from textual descriptions. The dataset is part of the Polifonia Project at the University of Bologna and describes the musical history of Bologna. The pipeline described in this thesis is shown in Figure 1.
In summary, the contributions of this thesis are the following:

- creation of a domain-specific selectional extraction tool, analyzing how they can be used

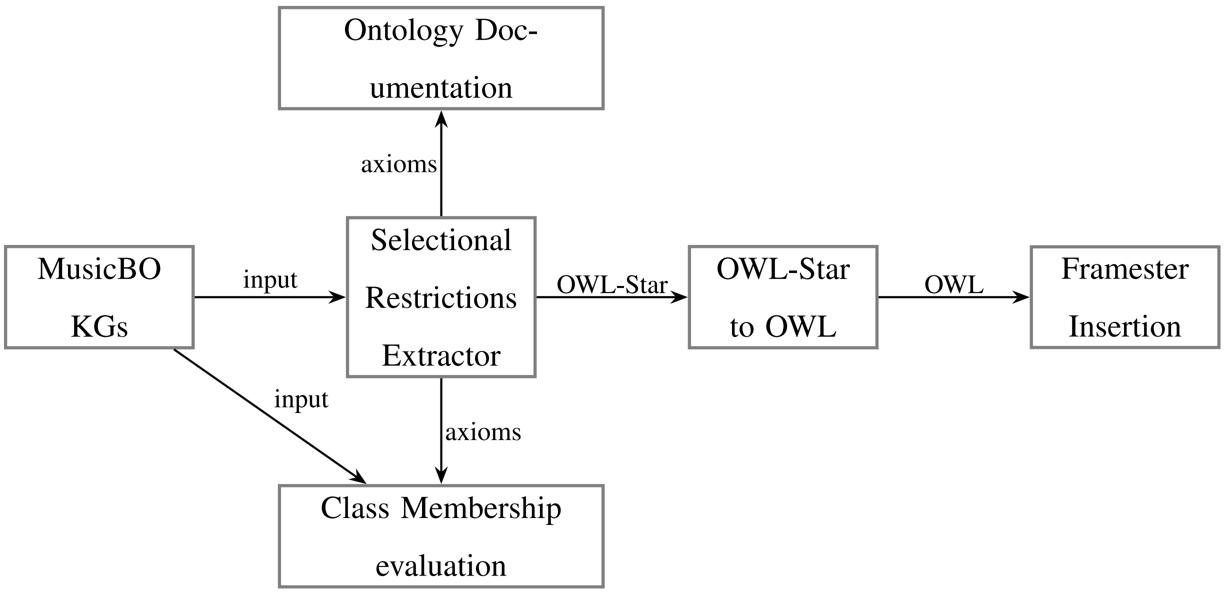


Figure 1: Graphical view of the pipeline describing the tools used

for ontology documentation[1];

- creation of an OWL-Star to OWL mapping together with proofs on its foundational aspects[2];
- insertion of domain-specific (MusicBO) selectional restrictions inside Framester;
- application of neural models for the class membership task over MusicBO augmented knowledge graphs.

The thesis is organized as follows. Chapter 1 introduces concepts and notions at the basis of the topics discussed. Chapter 2 shows the related works, highlighting the position of the thesis in the literature. Chapter 3 shows the implementation and methodology used for

[1]https://github.com/lucacontalbo/selrestr_maker
[2]https://github.com/lucacontalbo/OWLStar-to-OWL

extracting the selectional restrictions out of the MusicBO knowledge graphs. Chapter 4 introduces RDF* and OWL-Star, discussing about the mapping between OWL-Star to OWL and developing proofs regarding the translation. Chapter 5 discusses about the machine learning approaches regarding knowledge graphs, showing the implementation of the class membership task on selectional restriction augmented ontologies. In the end, Chapter 6 shows the obtained results.

Contents

List of Figures

List of Tables

Chapter 1

Background

The Semantic Web is an extension of the *World Wide Web* where data is more interconnected and machine-readable. It aims to enable computers to process and understand the meaning of information on the web, rather than just its raw format. RDF (Resource Description Framework) and OWL (Ontology Web Language) are two of the key technologies used in this topic. They provide a way to describe and model the relationships between entities and concepts in a structured and meaningful manner.
In this chapter, we will introduce the basics of RDF and OWL and explore how they can be used in combination with Linguistic Resources such as FrameNet, VerbNet, Propbank, and WordNet to provide a deeper understanding of language and meaning. We will also introduce Framester, a scientific tool that acts as a bridge between these resources and the semantic web. In addition, we will discuss the use of Design Patterns and Ontology Design Patterns in the Semantic Web. These patterns provide a reusable and standardized solution to common modeling problems and can be used to facilitate the creation of consistent and well-designed ontologies. Finally, we will examine Polifonia, a project that demonstrates the potential of semantic web technologies for the music domain.

1.1 RDF

The **Resource Description Framework** (**RDF**) is a data model used for the exchange of graph data. RDF is based on subject-predicate-objects triples: this simple but powerful data model allows to represent abstract relationships between entities, such as axioms, as well as more detailed information about instances of particular domains. Moreover, it is able to represent syntactical and semantical structure of text data by aligning entities to pre-defined **linguistic resources** (Section 1.2).
RDF has a crucial role in several scientific fields and purposes, like:

- adding machine-readable information to Web pages enabling them to be displayed in an enhanced format on search engines or to be processed automatically by third-party applications;
- enriching a dataset by linking it to third-party datasets, hence providing to the user (or machine) a broader knowledge base;
- provide data for ML based models;
- using the datasets currently published as Linked Data, for example building aggregations of data around specific topics;
- providing a standards-compliant way for exchanging data between databases.

Triples are constituted by:

- **Resource**: entities identified by IRIs.
- **Property**: resource used to describe relations about resources.
- **Property value**: can be either a literal or a resource.

Subjects and predicates are resources, while objects are property values. Hence, literals cannot be used as subjects of a triple.

Example 1.1.1. The sentence *Led Zeppelin have participated at the Live Aid* can be formalized as

```
<wd:Q2331, wd:Property:P1344, wd:Q193740>
```

where `wd` is the namespace for *wikidata.org/wiki/* and the strings at the end of the IRIs are identifiers of *Led Zeppelin*, *participant in*, *Live Aid* respectively.

Sets of RDF triples can be represented as **directed graphs**, which can be serialized in different syntaxes (e.g. Turtle, Manchester syntax etc.).
Formally,

Definition 1.1.1. Given a set of IRIs U, a set of blank nodes B and a set of literals L, a triple t is defined as

$$t = (s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$$

An RDF graph G is a set of triples

$$G \subseteq (U \cup B) \times U \times (U \cup B \cup L)$$

1.1.1 Semantics

RDF triples are great for defining relations between resources, but to define the meaning of each entity involved in a graph, the standard RDF vocabulary needs to be extended. With such extensions, RDF graphs may support more extensive entailments[48]. These extensions define **entailment regimes** which are valid under a particular RDF extension.
RDF is a base notation, used by a variety of extensions. Each extension must adhere to the

truth conditions of narrower vocabularies, meaning that they can be extended but not negated. Specifically,

Definition 1.1.2 (Simple interpretation)**.** Consider `R` and `L` to be the set of IRIs and literals respectively. A **simple interpretation** I consists of a quintuple `<IR,IP,IEXT,IS,IL>`, such that

- `IR` is a non-empty set of resources.
- `IP` is a set of properties.
- `IEXT` is a function $\mathtt{IP} \mapsto P(\mathtt{IR} \times \mathtt{IR})$. It identifies which properties are true for set of pairs of resources.
- `IS` is a function $\mathtt{R} \mapsto \mathtt{IR} \cup \mathtt{IP}$.
- `IL` is a partial function $\mathtt{L} \mapsto \mathtt{IR}$.

with $P(\Theta)$ being the power set function, i.e. the set of all subsets of set Θ.

Definition 1.1.3 (Semantic conditions for ground graphs)**.** The semantics of a ground graph are given by the following rules:

- $x \in \mathtt{L} \rightarrow \mathtt{I}(x) = \mathtt{IL}(x)$.
- $x \in \mathtt{R} \rightarrow \mathtt{I}(x) = \mathtt{IS}(x)$.
- if $t = $ `<s,p,o>` is a ground triple, then $\mathtt{I(p)} \in \mathtt{IP} \wedge \mathtt{<I(s),I(o)>} \in \mathtt{IEXT(I(p))} \rightarrow \mathtt{I}(t) = \mathtt{true}$ else $\mathtt{I}(t) = \mathtt{false}$
- if G is a ground RDF graph, then $\exists t \in G, \mathtt{I}(t) = \mathtt{false} \rightarrow \mathtt{I}(G) = \mathtt{false}$, else $\mathtt{I}(G) = \mathtt{true}$

All RDF extensions must be compliant to RDF semantic conditions and extend them by adding further semantic rules. Indeed, Definition 1.1.3 does not define any set constraints, for example regarding set disjointness or sub classing of literal entities. In RDF it is possible to define resources as subclasses of string labels, or it is possible to define instances of multiple disjoint classes.
In other words, RDF has no denotation for properties relevant for axiom triples: extensions such as OWL tackle this problem by extending the vocabulary by adding resources with fixed semantics.

1.1.2 RDF Schema

RDF Schema (RDFS)[49] is a semantic extension of RDF. It defines new resources with a fixed meaning, such as

- rdfs:Resource, the root class;
- rdf:type, used for meronymy (membership relations);
- rdfs:Class, used for representing sets;
- rdf:Property, used for representing relations between resources;
- rdfs:subClassOf, used for representing the subset relation;
- rdfs:domain, which indicates the subject class of a given property;
- rdfs:range, which indicates the object class of a given property;
- rdfs:label, used for adding human-readable labels to resources;
- rdfs:Literal, which is the class of literal values.

RDFS does not impose syntactic restrictions on RDF graphs, but it defines an entailment regime strictly dependant on the IRIs assumptions made before. Some of the entailments are shown below.

$$(p\ \texttt{rdfs:domain}\ c) \wedge (x \rightarrow^{p} y) \implies x\ \texttt{rdf:type}\ c$$
$$(p\ \texttt{rdfs:range}\ c) \wedge (x \rightarrow^{p} y) \implies y\ \texttt{rdf:type}\ c$$
$$x \rightarrow^{p} y \implies x\ \texttt{rdf:type rdfs:Resource}$$
$$x \rightarrow^{p} y \implies y\ \texttt{rdf:type rdfs:Resource}$$
$$(p_1\ \texttt{rdfs:subPropertyOf}\ p_2) \wedge (p_2\ \texttt{rdfs:subPropertyOf}\ p_3) \implies$$
$$p_1\ \texttt{rdfs:subPropertyOf}\ p_3$$
$$p\ \texttt{rdf:type rdf:Property} \implies p\ \texttt{rdfs:subPropertyOf}\ p$$
$$(p_1\ \texttt{rdfs:subPropertyOf}\ p_2) \wedge (x \rightarrow^{p_1} y) \implies x \rightarrow^{p_2} y$$
$$x\ \texttt{rdf:type rdfs:Class} \implies x\ \texttt{rdfs:subClassOf rdfs:Resource}$$
$$(x\ \texttt{rdfs:subClassOf}\ y) \wedge (z\ \texttt{rdf:type}\ x) \implies z\ \texttt{rdf:type}\ y$$

The IRI assumptions can help parsers make inferences on the provided knowledge graphs. The obtained triples can also help in the completion of knowledge graphs, expressing relations between entities in a more explicit way.
The usage of more complete knowledge graphs can aid machine learning algorithms in calculating graph embeddings (Section 5). Some approaches in the Semantic Web[52, 10] provide node labels to deep learning models for obtaining entity embeddings, leveraging the outcomes for particular AI tasks. Hence, the introduction of more user-friendly properties like `rdfs:label` does not only help in making the knowledge graph more explainable, but they can be used for obtaining semantic embeddings for the resources they relate to.
Data models based on Descriptive Logic[33] like RDFS introduce the **T-Box/A-Box dualism** for knowledge graph development. The two terms are used for referring to different types of

statements in knowledge bases: T-Box describes the terminology components which constrain the concepts of the represented knowledge (e.g. `rdfs:domain, rdfs:subClassOf`...) while A-Box statements are assertions associated with the T-Box model. Thus, A-Box is dependent from T-Box and must be compliant to its axioms. The combination of the two components constitutes an **ontology**, which is a formal model representing the main concepts and assertions of a domain of knowledge.

1.1.3 OWL

OWL[41], which stands for Web Ontology Language, is a language for expressing ontologies (formal models of concepts and their relationships) on the World Wide Web. OWL is designed to be used in conjunction with other web standards such as RDF (Resource Description Framework) and RDFS (RDF Schema), and it provides a way of formally defining concepts and relationships in a domain of interest.

There are three main types of OWL: OWL Lite, OWL DL, and OWL Full.

- **OWL Lite**: this is the simplest and most restricted form of OWL. It provides a limited set of constructs for expressing ontologies and is intended for use in applications where computational complexity is a concern. OWL Lite is suitable for modeling basic taxonomies with class hierarchy and simple relationships between classes.

- **OWL DL**: it is a more expressive version of OWL that is based on the description logic formalism. It provides a greater range of constructs for expressing ontologies and is intended for use in applications where the ability to represent complex relationships is required. OWL DL is suitable for modeling complex ontologies with rich class hierarchies, complex relationships between classes, and constraints on the use of individuals.

- **OWL Full**: it is the most expressive version of OWL and provides the full set of constructs for expressing ontologies. OWL Full is intended for use in applications where maximum expressiveness is required, but it is also the most computationally complex form of OWL and is not recommended for most applications.

Each of these OWL types provides a different level of expressiveness and computational complexity, and the choice of which type to use will depend on the requirements of the particular application. For example, in a domain where a simple class hierarchy is required, OWL Lite may be the most appropriate choice, whereas in a domain where complex relationships and constraints need to be represented, OWL DL may be a better choice. OWL Full, instead, is defined theoretically but not used very much, due to the difficulties of implementing reasoners that work with OWL Full. Indeed, OWL Full allows ontologies to augment the meaning of the pre-defined (RDF or OWL) vocabulary, which is the main cause of the implementation problems of a OWL Full reasoner.
With the introduction of *OWL Punning* in OWL 2, **OWL 2 DL** has slightly closed the gap with OWL Full. For the remainder of this paper, the term OWL will be used referring to OWL 2 DL.
OWL includes all RDFS primitives, like class hierarchies, property hierarchies, membership relations and restrictions on properties domain and range. At the same time, we can model knowledge bases by augmenting classes with set operations like union and intersection, hence being more specific when two resources are disjoint or equivalent; we can use universal and existential quantifiers for applying property restrictions; cardinality restrictions; inverse, symmetric, reflexive and transitive relations; reuse and linking of different ontologies.
OWL is based on the following basic notions:

- **Axioms**: basic statements that an OWL ontology expresses.

- **Entities**: elements used to refer to real world objects. 0-ary predicates are called *individuals*, e.g. the band *Led Zeppelin*, unary predicates are called *classes*, e.g. *Person*, binary predicates are *properties*, which can be
  - **Object properties**, relating 2 different objects, e.g. *hasFriend*.
  - **Datatype properties**, relating one object and a data value, e.g. *hasAge*.
  - **Annotation properties**, relating objects and ontologies to meta information, e.g. *versionInfo*.
- **Expressions**: combination of entities to form complex descriptions of basic ones called **constructors**, e.g. `Son` $\leftrightarrow$ `Person` $\cap$ `Male`

OWL constructors can be used to refer to new entities without defining them. In the previous example, we can avoid defining a new class `Son` and use `Person` $\cap$ `Male` in every expression in which it is needed. Constructors are created in the OWL language by using square brackets.
OWL enriches the the assumptions made by RDF and RDFS, thus introducing a more complex and complete entailment regime. It introduces new rules starting from Definition 1.1.3, imposing special syntactic restrictions over RDF graphs. This means that OWL reasoners are more powerful and can detect more inconsistencies inside knowledge bases. OWL also imposes syntactical restrictions, prohibiting, for example, to define resources as subclasses of literals. Table 1.1 shows examples of ontologies for some resources introduced by OWL.

OWL Punning

In naive set theory, sets can contain other sets as elements, leading to the possibility of self-contradicting sets (such as the set of all sets that do not contain themselves). Russell's type theory[12] aims to avoid these paradoxes by assigning a type to every mathematical object,

Case	Ontology example
Class disjointness	`[] rdf:type owl:AllDisjointClasses ;` `owl:members ( :Woman :Man )`
Property negation	`[] rdf:type owl:NegativePropertyAssertion ;` `owl:sourceIndividual :Bill ;` `owl:assertionProperty :hasWife ;` `owl:targetIndividual :Mary`
Intersection	`:Mother owl:equivalentClass [` `rdf:type owl:Class ;` `owl:intersectionOf ( :Woman :Parent ) ]`
Union	`:Parent owl:equivalentClass [` `rdf:type owl:Class ;` `owl:unionOf ( :Mother :Father )]`
Existential quantification	`:Parent owl:equivalentClass [` `rdf:type owl:Restriction ;` `owl:onProperty :hasChild ;` `owl:someValuesFrom :Person]`
Universal quantification	`:HappyPerson rdf:type owl:Class ;` `owl:equivalentClass [` `rdf:type owl:Restriction ;` `owl:onProperty :hasChild ;` `owl:allValuesFrom :Happy ]`

Table 1.1: Ontology examples

such as a natural number, a set, or a proposition, and restricting the ways in which objects of different types can be combined. According to Russell's type theory, each object is assigned a type that reflects its logical or mathematical category. For example, natural numbers have type 0, sets have type 1, and propositions have type 2. Objects of the same type can be combined in certain ways, but objects of different types cannot be combined. For example, it is not possible to form a set that contains both natural numbers and sets. In this way, Russell's Theory does not suffer from the **Russell Paradox**, which instead has proved the naive set theory to be inconsistent.
In OWL, the issue of entities that belong to multiple categories is addressed through the concept of **punning**, which allows an individual to be treated as both an instance of a class and as an individual. Often times, when representing particular domains, it might happen that certain concepts need to be described at two different levels, i.e. from an **intensional** (abstract, related to *classes*) or from an **extensional** (related to *individuals*) point of view.

Example 1.1.2. Consider the following sentence

Mark has a dog. Dogs are living beings

The term *dog* is used as an individual (*Mark's dog*) and as a class (*Dogs are living beings*). Hence, in this case, there is the need to describe *dogs* in a general way (intensional) and in a factual manner (extensional). In OWL we could describe this knowledge in the following way:

```
:Dog rdf:subClassOf owl:Class .
:MarksDog rdf:type :Dog .

:Dog rdf:type a:LivingBeing .
```

Note that `:Dog` is both a **class** and an **instance**.

Russell's type theory and OWL punning are related in that they both address the issue of representing entities that belong to multiple categories in a consistent and non-ambiguous manner. While Russell's type theory addresses this issue by assigning a type to each mathematical object, OWL Punning assigns a category (*class*, *instance*) to each intensional and extensional entity. Indeed, OWL Punning simply allows to use the same URI to identify the same entity, regardless of whether it is used as a class or an instance. However, from a logical standpoint, it is important to differentiate the two categories: with punning this task is performed automatically by the OWL reasoner. In other words, it is important to clearly distinguish different categories when dealing with mathematical and logical objects, otherwise it is possible to obtain inconsistencies.

1.2 Language Resources

A **language resource**[6] is a composition of linguistic material used for documenting and describing a language for its computational modeling, necessary for a plethora of natural language processing (NLP) systems. Each language resource may be specific for a certain part of linguistics, such as lexicon, but most of the resources used in NLP systems are based on large and structured sets of machine-readable texts.
Since the main use of these datasets are for machine learning models, the design of corpora must be carefully implemented, ensuring that they satisfy data properties for developing effective learning pipelines. Corpora must be **balanced**, thus equally representing each text category in the domain of interest, and a clear **sampling** procedure must be defined, indicating the number and length of each text sample. A careful data creation pipeline ensures representativeness of the full range of variability in a population.
Corpora annotating sentences at various linguistic levels, more specifically syntax and gram-

matical rules, are called **treebanks**. As the name suggests, the main structure used to represent such models are trees structures. Indeed, they are useful to describe the syntactic relations between words in a sentence, but also for parsing of non-natural languages like programming languages (through the use of particular structures called Abstract Syntax Trees[19]).
Each treebank, hence each language resource, is created by following certain linguistic assumptions, defining the guidelines of annotation of the syntactical structure (**annotation scheme**).
Most annotation schemes provide information about part-of-speech tags (POS) and lemmatized or stemmed versions of the words. The choice of these implementation details is mainly dependant on the linguistic theory adopted for the creation of the treebank.
One of the main groups of treebanks are the ones annotating **phrase structure**, like *Penn Treebank*.

Example 1.2.1 (Penn Treebank (taken from [60])). The sentence *"Jones followed him into the front room, closing the door behind him"* can be annotated in the Penn Treebank as

```
( (S (NP-SBJ-1 Jones)
    (VP followed
        (NP him)
        (PP-DIR into
            (NP the front room))
        ,
        (S-ADV (NP-SBJ *-1)
            (VP closing
                (NP the door)
                (PP behind
                    (NP him)))))
```

.))

using a skeletal parsing where constructs like `S`,`NP`,`VP`,`PP` annotate the syntactic tagset (strictly correlated to POS tags), and constructs with the `-` character are functional tags expressing grammatical mappings.

1.2.1 PropBank Treebank

PropBank (PB)[43] is a text corpus which builds on top of the Penn Treebank's annotation scheme by defining and assigning semantic roles in relation to each predicate. It builds on the assumption that each verb and its associated roles define the semantics of the sentences in which they appear, thus associating meaning to the syntactical structure given by the Penn Treebank.
Each predicate is associated with a **sense ID**, disambiguating the verb to one of its corresponding meanings.

Example 1.2.2. The predicate *allow* can be used as *let* in

"The reforms allow the Big Board to halt trading for one hour"

where in the following case

"Editorials in the Greenville newspaper allowed that Mrs. Yeargin was wrong"

it is used as an admission of truth.

The predicate *allow* in the previous example has different meanings and use cases, identified with their corresponding sense ID as `allow.01` and `allow.03` in PropBank. The combination of predicate and sense ID defines **framesets**, which also specify the associated roles.

The arguments are numbered for associating them to general semantic roles: `ARG0` up to `ARG4` are for *agent, patient, instrument or benefactive or attribute, starting point or benefactive or attribute and ending point* roles respectively. Instead, the `ARGM` argument stands for *modifier* and it is mainly used for expressing *manners*, *location* or *temporal* information.

Example 1.2.3. The following sentence is parsed using the PropBank annotation schema

$$[\textit{John}]_{\texttt{ARG0}}\ [\textit{ca}]_{\texttt{ARGM-MOD}}[\textit{n't}]_{\texttt{ARGM-NEG}}\ [\textit{keep up}]_{\texttt{keep.05}}\ [\textit{with the technological developments}]_{\texttt{ARG1}}$$

with `ARGM-MOD` indicating modal verbs, `ARGM-NEG` indicating negation and `keep.05` expressing to maintain one's position, with its corresponding roles `ARG0` and `ARG1` representing the mantainer of the position and to what it is relative.

Note that it is trivial to parse these annotations to tree structures, thus obtaining more graphical representations like in all treebanks.
PropBank has been lately used for **Abstract Meaning Representation** (AMR)[4], which is a semantic representation language using direct acyclic graphs (DAGs) to depict semantics of texts.

1.2.2 VerbNet

VerbNet (VN)[54] is the largest network of english verbs that links their syntactic and semantic patterns. It is based on Levin classes[34], but it extends its hierarchical structure by defining additional subclasses, together with the corresponding thematic roles and selectional restrictions. It is very similar to PropBank, but the latter is **more syntax based** (even though AMR does not follow this pattern), while VerbNet is more **independent from the text structure**, being more amenable for tasks like machine translation which are based on semantic features[63, 25]. It is argued that PropBank labels generalize well across unseen verbs, specially for `ARG2-ARG5` roles which tend to be less constraining. However, VerbNet is more informative by defining **selectional restrictions** which are specific semantic constraints on the roles associated to verbs.

Verb grouping is based on the same hypothesis followed for defining Levin classes: a verb's meaning influences its syntactic behaviour and viceversa, i.e. verbs appearing in similar contexts have similar meanings, allowing us to determine semantically coherent groupings of such predicates.

Each class is composed by the **members**, **roles** and **frames**. The members are the predicates belonging to a given class, the roles are the arguments shared across the verbs and the frames indicate all the possible sentence structures and semantics in which the class can be used, together with some examples.

Table 1.2 shows all the possible thematic roles with example of classes that uses them. Each one of them can be associated with one or more selectional restrictions, who follow a hierarchical structure shown in Figure 1.1.

In `CUT-21.1` class, the roles and restrictions are:

- `AGENT [+INT-CONTROL]`
- `PATIENT [+CONCRETE]`
- `INSTRUMENT [+CONCRETE]`
- `SOURCE`
- `RESULT`

Role	**Example**
Actor	used for some communication classes (e.g. `Meet-36.2`) when both arguments can be considered symmetrical (pseudo-agents)
Agent	generally a human or an animate subject.
Cause	used mostly by classes involving Psychological Verbs.
Destination	end point of the motion, or direction towards which the motion is directed.
Source	start point of the motion.
Location	underspecified destination, source, or place, in general introduced by a locative or path prepositional phrase
Instrument	used for objects (or forces) that come in contact with an object and cause some change in them.
Patient	used for participants that are undergoing a process or that have been affected in some way.
Predicate	used for classes with a predicative complement
Theme	used for participants in a location or undergoing a change of location

Table 1.2: Some of the thematic roles available in VerbNet

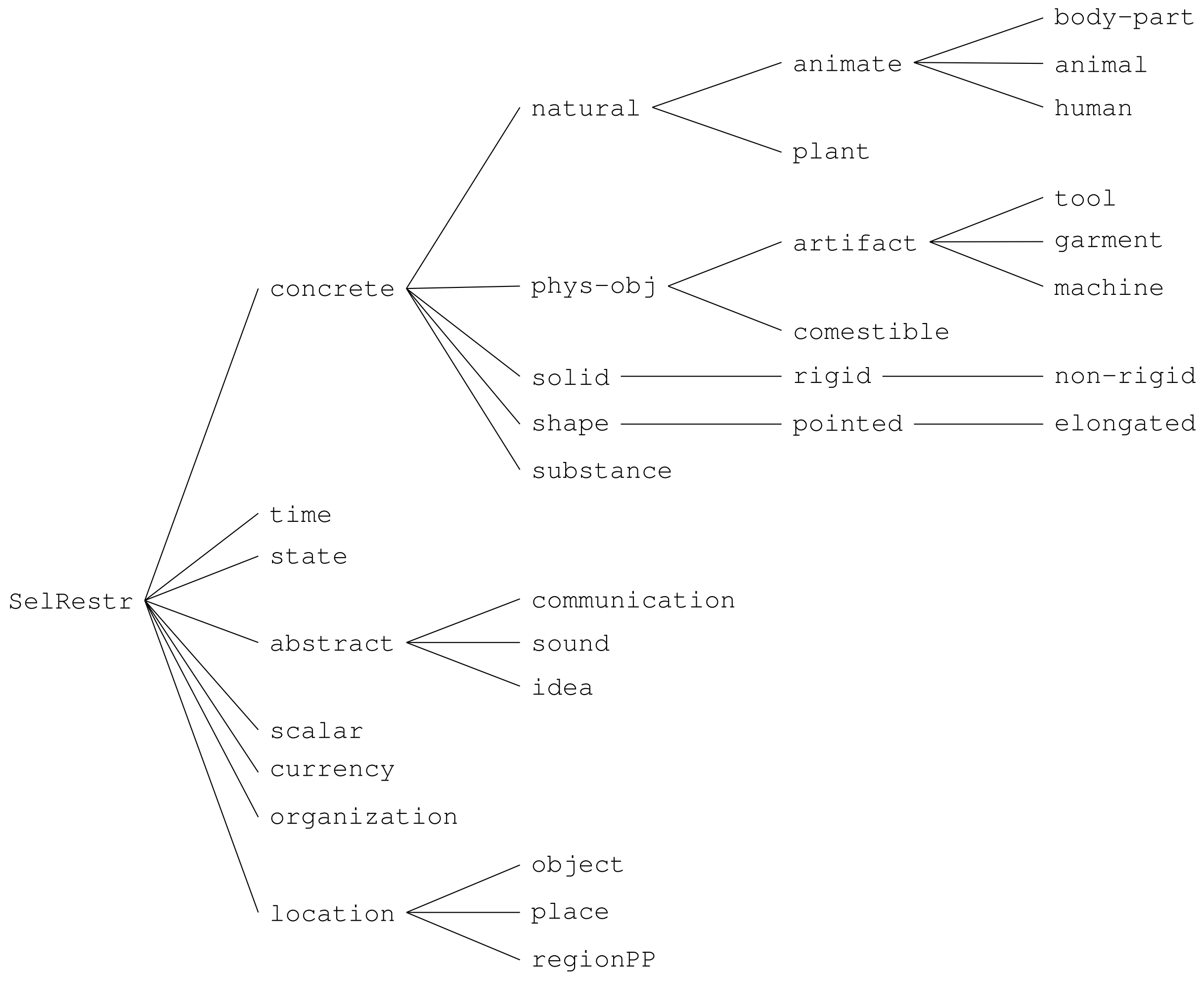


Figure 1.1: Selectional restrictions hierarchy

while some of its frames are shown in Table 1.3. The thematic roles are mapped on top of the POS syntax, while the semantics are specified using a Neo-Davidsonian variable E[28].

VerbNet classes are structured in a **hierarchical** manner. The class `CUT-21.1-1` inherits from `CUT-21.1` roles and frames but no members, since it does not have any. The following table shows the members of class `CUT-21.1-1`.

NP V **NP**	
Example	*Carol cut the bread*
Syntax	AGENT V PATIENT
Semantics	cause(AGENT, E) manner(during(E), Motion, AGENT) contact(during(E), ?INSTRUMENT, PATIENT) degradation_material_integrity(result(E), PATIENT)
NP V NP PP.instrument	
Example	*Carol cut the bread with a knife*
Syntax	AGENT V PATIENT {with} INSTRUMENT
Semantics	cause(AGENT, E) manner(during(E), Motion, AGENT) contact(during(E), ?INSTRUMENT, PATIENT) degradation_material_integrity(result(E), PATIENT) use(during(E), AGENT, INSTRUMENT)
NP V ADVP-Middle	
Example	*The bread cuts easily*
Syntax	PATIENT V ADV
Semantics	property(PATIENT,PROP), adv(PROP)

Table 1.3: Some frames for CUT.21-1 class

Members of CUT-21.1-1	
CHIP (FN 1,2; WN 1,2,5)	HEW (WN 2)
CHOP (FN 1; WN 1)	REAM (WN 2,3)
CLIP (WN 1,4)	RIP (FN 1,2; WN 1)
CUT (FN 1,2,3; WN 1,24,25,30,32)	SAW (WN 1)
HACK (WN 1,3)	SCARIFY (WN 1,2,3)

`WN` and `FN` next to class members indicate mappings with **WordNet synsets** and **FrameNet frames** respectively.
The roles are extended by allowing the `PATIENT` thematic role to have `body-part` as an additional semantic constraint and by adding a new frame.

1.2.3 FrameNet

FrameNet (FN)[3] is a treebank based on Fillmore's linguistic theory. The main assumption is that it is impossible to understand the meaning of a word without considering the context in which it appears and the relation with other textual entities.
The main linguistic construct are **frames**, organized in a hierarchical structure, which are representation of situations involving several arguments. A frame is composed of:

- frame elements (FE): same as semantic roles, but furtherly divided in **core and non-core FE**. The subgrouping is based on whether the depicted roles are essential to the meaning of the frame or not.

- lexical units (LU): lemmas with POS tags that evoke a particular frame. The relation between LUs and frames is many-to-many, meaning that multiple LUs can evoke a frame, but multiple frames can share a LU as well (due to word senses).

- frame-to-frame relations: e.g. inheritance, subframing, temporal, causative relations etc.

- frame definition.

- example sentences.

1.2.4 PB vs VN vs FN

PropBank, VerbNet and FrameNet are all very similar treebanks, representing situations based on verbs (or, rarely, nouns) and their roles. The reader may know wonder similar treebanks have been created and what leads latest applications to choose one corpus over the other.
The main difference is on **specificity**: PB is **more general**, while FN is the **most specific** treebank. Hence, PB is more suited to be used for the training procedure of **Semantic Role Labeling** systems (SRL), while FN and VN are more amenable for tasks such as **Machine Translation** (MT) and **Question Answering** (QA) (see Section 1.2.2).

Example 1.2.4. Given the following sentence *Mark ate the soup with a spoon*, PB, VN and FN would represent the text in the following ways respectively:

$$[\textit{Mark}]_{\texttt{ARG0}}\ [\textit{ate}]_{\texttt{PREDICATE}}\ [\textit{the soup}]_{\texttt{ARG1}}\ \textit{with}\ [\textit{a spoon}]_{\texttt{ARGM-MANNER}}$$
$$[\textit{Mark}]_{\texttt{AGENT}}\ [\textit{ate}]_{\texttt{PREDICATE}}\ [\textit{the soup}]_{\texttt{PATIENT}}\ \textit{with}\ [\textit{a spoon}]_{\texttt{INSTRUMENT}}$$
$$[\textit{Mark}]_{\texttt{INGESTOR}}\ [\textit{ate}]_{\texttt{PREDICATE}}\ [\textit{the soup}]_{\texttt{INGESTIBLE}}\ \textit{with}\ [\textit{a spoon}]_{\texttt{INSTRUMENT}}$$

It is easy to see the differences in generality between the aforementioned linguistic resources

1.2.5 Framester

The usefulness of linguistic resources is affected by their limited coverage and non-standard semantics. One way to reduce this problem is to apply linking strategies among lexical and factual resources to broaden FrameNet's coverage. This idea brought to the creation of **Framester**[22], a frame-based ontological resource acting as a hub between linguistic resources such as FrameNet, WordNet, VerbNet, BabelNet, DBpedia, Yago, DOLCE-Zero etc. (Figure 1.2)

Most Semantic Web projects are opportunistic, meaning that the formal semantics of Linguistic Linked Data is delegated to the user through the use of *references* that may be estabilished between linguistic resources and classes of an existing ontology. This approach is greedy and leads to confusion on the usage of certain resources. Framester tackles this problem by applying a rigorous formal treatment of **Fillmore's frame semantics**.

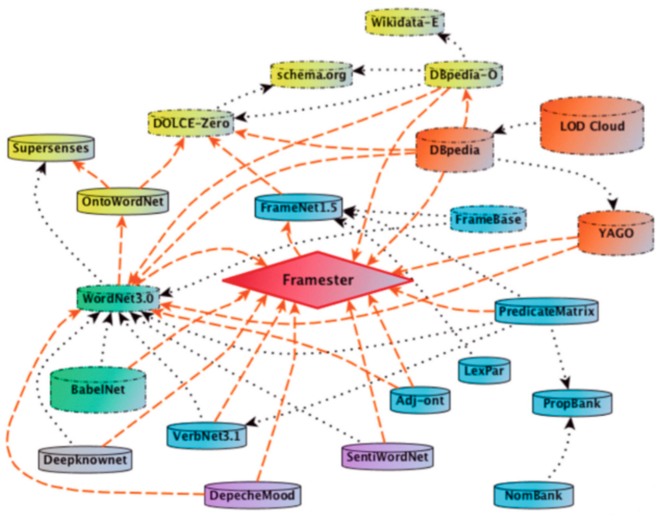


Figure 1.2: Framester Cloud. Red represents the main hub, blue the role-oriented lexical resources, orange the fact-oriented data, green the WordNet-like lexical resources, yellow the ontology schema, purple the datasets for Sentiment Analysis. Orange arrows indicate Framester's existing links (taken from [22])

Framester uses the **D&S** (Descriptions and Situations)[22] knowledge pattern. D&S allows to distinguish the reification of the intension of a predicate (a *description*) from the reification of the extensional denotation of a predicate (a *situation*). In other words, a description d defines a set of concepts $c_1, ..., c_n$ that classify entities $e_1, ..., e_m$ involved in a situation s. Thus, a description d deals with interpreting a set of abstract concepts (classes): their instances is what

constitutes the situation *s*. This decoupling allows a more effective representation of abstract concepts and, hence, more precise semantics.
Specifically, Framester uses **OWL Punning** (Section 1.1.3) to represent **D&S**: an entity can be used both as a *class* and as an *individual*, where the *class* is the **intension** (*description*) and the *individual* is the **extensional denotation** (*situation*). Each frame specifies a set of *lexical units* that can evoke them and a set of *roles* (also called *frame elements*).
We have specified that Framester uses multiple linguistic resources, combining their strengths into a singular web resource. The most tedious task to perform to link different linguistic resources is to align each frame and synsets together based on their semantics. The linking has been done by exploiting existing tools like SemLink[1], which connects PropBank, VerbNet and FrameNet by using a manually created mapping.

1.3 Design patterns

Desing patterns[20] are **typical solutions to recurrent design problems**. Software development is a very complex task and the programmer has the responsibility to maximise the reuse of objects by defining classes and their relations in a clear manner.
During the software development process, the programmer can stumble upon recurrent and well known problems: thus it is pointless to *"re-invent the wheel"*, just like it is pointless to re-implement an already existing library without adding new features. In these cases, the software development process can be sped up by using well known templates that have proved to lead to high quality projects. Using design patterns can improve the software creation pipeline in several ways:

[1]`https://github.com/cu-clear/semlink`

- by **speeding up the process**, since programmers do not have to define new templates for representing the problem.
- they are **abstract**, allowing them to be shared designers with different point of view.
- they are **not complex**, since their design is optimized for augment code readability and reusability.
- they are **not domain specific**, allowing them to be used in different use case scenarios.

Given these properties, it is trivial to evince that design patterns are **not based on a specific language**, but they describe general concepts for solving a task. The focus is not on the details of how to solve a problem by stating a set of actions, but more on the general approach to be employed.
Design patterns are often used in object-oriented paradigms, showing the relations and interactions between classes and objects. One example is shown in Figure 1.3, namely the Abstract Factory design pattern, which delegates object creation from client code to a Factory object. This improves code adaptation: if a new object type must be created, the programmer must only change the client code by calling another Factory class. Since these classes return a pointer to the same abstract type regardless of which Factory has been called, the client code does not see these changes, since it has to deal with the same object. Thus, changes to the client's code is reduced.

The major criticism directed at design patterns is mainly due to its applications: some problems are slightly different from the ones in theory, but programmers tend to use design patterns as they are, without being flexible enough to adapt the approaches to specific domains. Also, some of the templates are only used in few programming languages, since most of them already provide the needed amount of abstraction through functional language constructs[26]. Despite

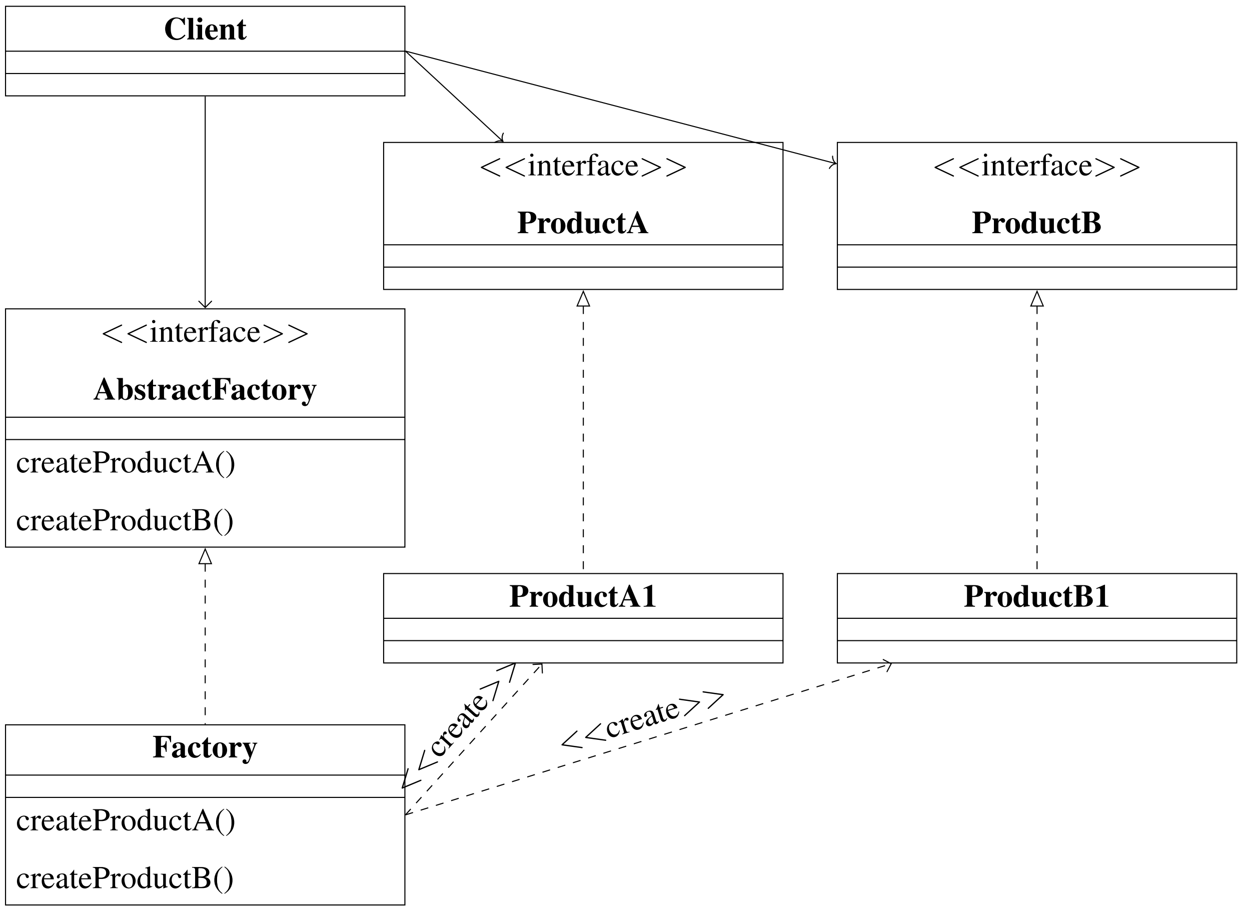


Figure 1.3: Abstract Factory

these drawbacks, pattern-based programming tends to have more advantages than liabilities, proving to be a **useful resource** for software development.

1.4 Ontology Design Patterns

Section 1.1.3 shows why it is important to extend the RDFS vocabulary with additional OWL axioms, allowing KGs to encode class relations in a similar way with respect to Description Logic. Reasoners based on OWL are able to detect more logical inconsistencies due to the ontology richer entailment regime. Different subjects define in a different way the notion of consistency.

Mathematical (and logical) truths **are different** from truths based on the experienceable world. In physics or chemistry the notion of truth is based on whether an experiment can be replicated or not. Logics, instead, is more abstract and it is not based on what can be experienced, but on axioms: every mathematical model of a specific theory must interpret mathematical primitives in such a way that each axiom of the theory is consistent, i.e. is true. Indeed, in literature we do not have a single mathematical theory, but **multiple ones**. Each of them can be more suited for a specific task than the others. For example, in the 19th century, researchers have tried to tweak the Euclidean axioms, giving rise to multiple non-Euclidean geometries. As done by Riemann, it is possible to challenge the *parallel postulate*[24] (i.e. that there exists one and only one line r passing through a point P and parallel to line l not passing through P) by assuming that r does not exist. By changing other axioms, it is possible to obtain a consistent theory, different from the euclidean one, which has made a huge impact on a variety of topics, like **Representation Theory** (crucial for Deep Learning).

Knowledge representation suffers from the same flexibility: since it is based on logics, it is possible to define multiple theories with consistent ontologies which are, however, uncorrelated with the domain of interest. Even though OWL provides logical language constructs to represent various scenarios, it does not give any guidelines on their use, making **arbitrary** the decision to encode RDF resources as classes, properties or instances. Considering these premises, it is technically possible to create ontologies with a wrong use of OWL axioms like `owl:sameAs`, obtaining KGs which are logically consistent, but not with regards to the domain of interest. Hence, logical consistency is not enough: in **ontology creation** it is crucial to be compliant to the **domain requirements**, thus leveraging the expressibility of logic-based formalisms to more concrete settings. The practice that is often employed to reduce this gap is the use of **Ontology Design Patterns** (ODPs)[21].

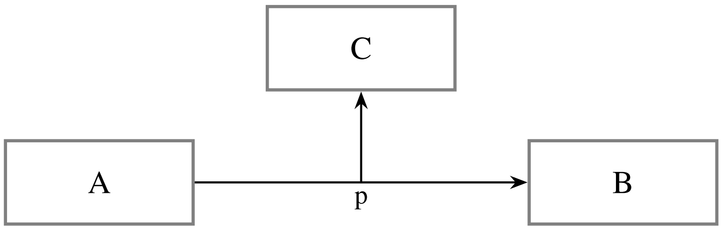
Ontology design patterns can be divided in several parts, based on the problem they strive to solve. The most important ones are **Logical ODPs** and **Content ODPs**.

1.4.1 Logical ODPs

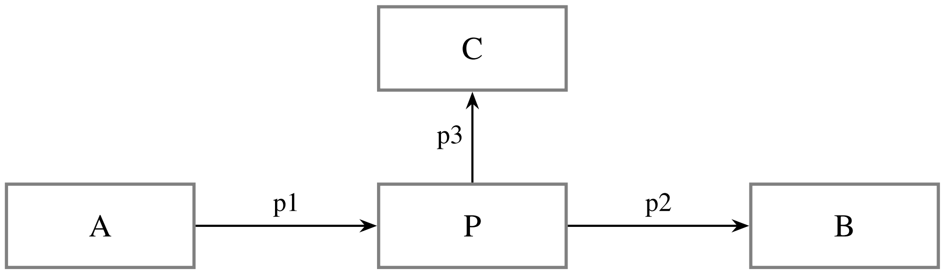
Logical ODPs solve problems of expressivity and it is based on OWL. Instead of focusing on the domain of the ontology, it defines general templates for solving recurring problems regarding axioms, hence focusing on the T-Box of our KG. One of the main logical ODP are **n-ary relations** [21, 15].

N-ary relations

In a triple $A \rightarrow^{p} B$, we might be interested in adding another entity to the relation. In a graphical way,



Obviously, this cannot be represented in RDF, since its basic structures are triples. One of the most applied workarounds in these cases is to **treat the property as a class**.



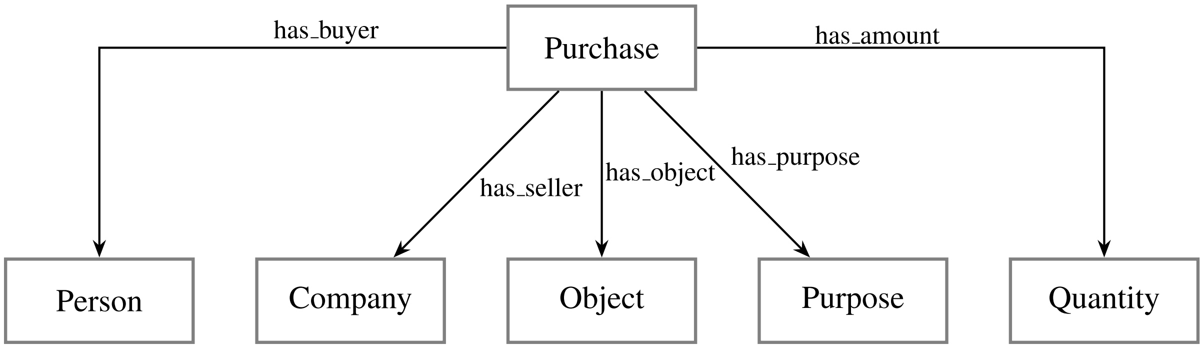
It is possible to model the ontology by directly defining relations $r1$ and $r2$ such that $A \rightarrow^{r1} B$ and $A \rightarrow^{r2} C$. But when classes B and C are semantically closely related, it is usually better to represent the triple as $A \rightarrow^{p1} P$, meaning that class A is in relation through property $p1$ with a **complex relation** P, which correlates semantically similar classes B and C.

Definition 1.4.1 (n-ary relation)**.** An **n-ary relation** r is a relation such that

$$\forall i, 1 \leq i \leq n. \quad (r, p, o_i) \in G \qquad \texttt{with} \quad n \geq 2$$

with n being the cardinality of the objects. The objects can be either classes or datatypes.

Example 1.4.1 (taken from [15])**.** Consider the following n-ary relation.



The diagram represent an n-ary relation with no participant (no subject connected to the n-ary class). Its corresponding Turtle code is the following

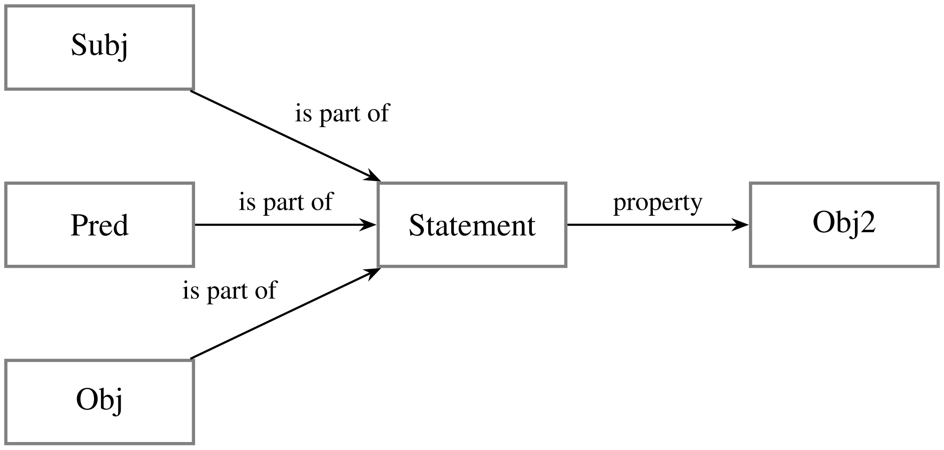
```
:Purchase
    a     owl:Class ;
    rdfs:subClassOf
          [ a     owl:Restriction ;
           owl:allValuesFrom :Purpose ;
           owl:onProperty :has_purpose
          ] ;
    rdfs:subClassOf
          [ a     owl:Restriction ;
           owl:cardinality 1 ;
           owl:onProperty :has_buyer
          ] ;
    rdfs:subClassOf
          [ a     owl:Restriction ;
           owl:onProperty :has_buyer ;
           owl:someValuesFrom :Person
          ] ;
    rdfs:subClassOf
          [ a     owl:Restriction ;
          owl:cardinality 1 ;
          owl:onProperty :has_seller
        ] ;
  rdfs:subClassOf
        [ a    owl:Restriction ;
          owl:onProperty :has_seller ;
          owl:someValuesFrom :Company
        ] ;
  rdfs:subClassOf
        [ a    owl:Restriction ;
          owl:onProperty :has_object ;
          owl:someValuesFrom :Object
        ] .
```

For each binary relations, existential, universal and cardinality constraints have been added to correctly define the axioms needed for representing inter-class relations. Usually, also inverse relations are added, but this has been omitted in the example.

Note that the synsets and frames defined by PropBank, VerbNet, FrameNet and WordNet can be all considered as ODPs, more specifically as n-ary relations, due to their multiple roles. This is not the only ODP that is used by these linguistic resources, but it is the most important one.

Reification

The term *reification* [21, 56] refers to the process of making a statement about a statement, or in other words, representing a statement as an entity in its own right. In an ontology, reification is achieved by representing statements as individuals and attaching properties to these individuals to describe the statement. Graphically, it can be represented in this way:



Example 1.4.2. Consider the following statement

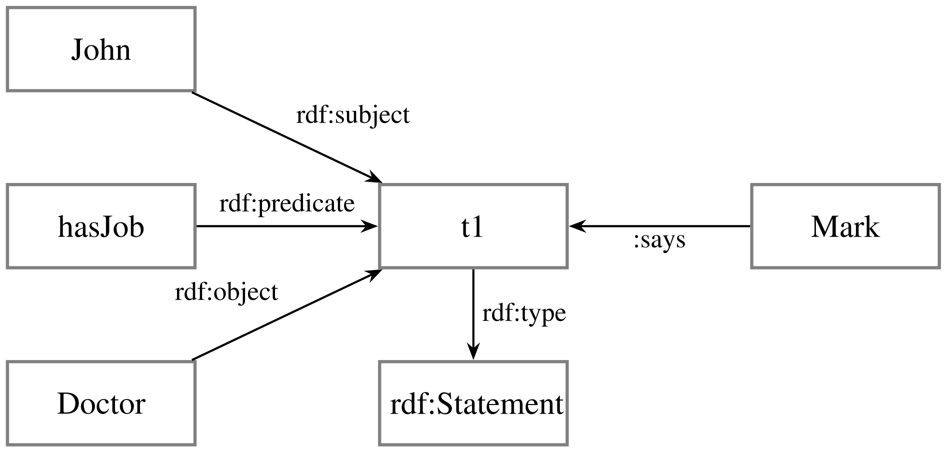
Mark said that John is a doctor

we can reify this statement by creating a new individual, say `t1` and attaching properties to it to represent the subject, predicate, and object of *John is a doctor*. Using RDF vocabulary we can represent this in the following way:

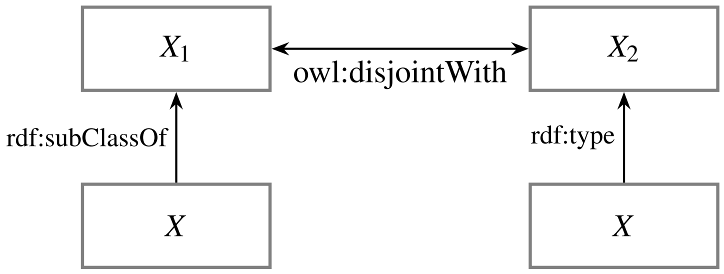
```
:t1 rdf:type rdf:Statement ;
   rdf:subject :john ;
   rdf:predicate :hasJob ;
   rdf:object :Doctor .

:mark :says :t1 .
```

and graphically represented as



The term reification is also used to represent the intensional reification of classes as values. As already stated in Section 1.1.3 and in Section 1.2.5, there might be cases in which concepts need to be represented as classes and instances simultaneously. Graphically:



Even though OWL Punning allows to use the same identifier for the two different uses, OWL reasoners differentiate between the two behaviours, thus following the representation illustrated above.

1.4.2 Content ODP

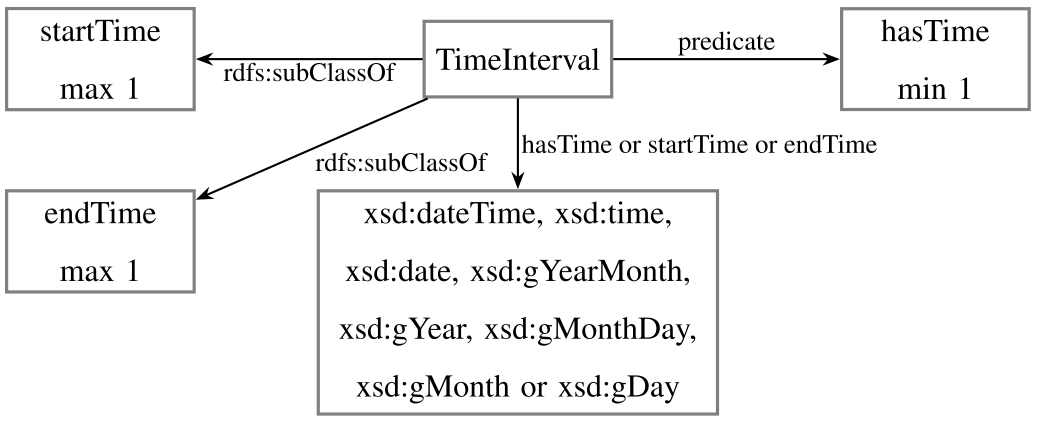
Logical ODPs are formal and crucial for defining ontologies, but they are not specialized for representing domain-dependent knowledge. For this task, **Content ODPs** [21, 46] are used in conjuction with Logical ODPs, introducing a set of design patterns that can be used to model different case scenarios.

The main difference between CODPs and LODPs is in the level of formality and the target audience. Content ODPs are designed to be accessible and understandable even by domain experts, while Logical ODPs are designed to be processed and utilized by automated reasoning systems.

When modeling a domain it is crucial to define precise *competency questions* (CQ): a well-defined ontology needs to be able to respond to questions about what is the topic represented, the reason why we represent it and whether we have the resources to maintain it. Hence, the competency questions affect the decision of which content design patterns need to be used for a specific domain. Here we will present only one Content ODP, the reader may check [21, 46] to have a broader overview of Content ODPs.

Time Interval ODP

The Time Interval ODP is a pattern used in modeling temporal information in an ontology. The main purpose of the pattern is to represent time intervals and the relationships between them. A time interval can be represented as a start time and an end time and can be used to represent a period such as a day, week, year or an undefined amount of time. The Time Interval Ontology Design Pattern can be used in various domains to represent temporal information. By using this pattern, it is possible to represent the relationships between time intervals and other entities in a structured and semantically rich way, enabling reasoning and analysis over the temporal information. This design pattern can be represented in the following way:



1.5 Polifonia

The Polifonia project [45] is a 3 million euro initiative funded by the EU Horizon 2020 Program, set to run from 2021 to 2024. Its goal is to reestablish the connections between music, people, places, and events from the 16th century to the present day and make the findings accessible to everyone via a global knowledge graph database on the web. This project aims to provide a new perspective on European musical heritage. The consortium behind the Polifonia project consists of a diverse group of researchers and music enthusiasts, including computer scientists, historians, linguists and musical heritage archivists. The project's mission is to bring about a significant change in the way musical heritage is preserved, managed, researched, interacted with, and promoted. The target audience for Polifonia includes memory institutions and providers of musical heritage resources, music domain experts, music cataloguers, music producers and artists, students and teachers, music lovers, cultural and creative industries. The project's approach is decentralized, interdisciplinary, open-source, and validation-driven. The project uses semantic web technologies and data science methods, as well as linguistic corpora and innovative human-machine interaction techniques to study and experience musical heritage. The project also includes reproducible live artistic installations. The expected outcomes of the Polifonia project include a deeper understanding of European cultural heritage, increased reuse of music data, a more competitive environment for companies, advanced research methodologies in the music field, increased cultural tourism and public engagement, and improved preservation and protection strategies for musical heritage. The Polifonia project utilizes a variety of cutting-edge technologies in order to facilitate access and discovery of European Musical Heritage. These technologies include:

- **Knowledge Graphs**: Created through the use of semantic web technologies, these graphs provide a comprehensive view of musical heritage data.
- **Data Analytics**: Advanced data science methods are utilized to extract information about music patterns and features of musical objects, enhancing the knowledge graphs.
- **Textual Corpora**: The project includes linguistic corpora covering multilingual and multidisciplinary texts from various time periods, providing information about musical heritage.
- **Structured Data Extraction**: Techniques are used to extract data about cultural, social, and historical aspects of musical heritage, including tangible and intangible assets and their spatial and temporal connections.
- **Interactive Human-Machine Experiences**: The project incorporates innovative ways for people to study and experience musical heritage.
- **Artistic Installations**: Reproducible live artistic installations provide a unique and engaging way to experience musical heritage.

Each of these technologies plays a role in creating a comprehensive, accessible, and creative database of European Musical Heritage. The project's interdisciplinary approach and use

of data analytics and knowledge graphs allow for the extraction and representation of valuable information about musical heritage.
The Polifonia project can be subdivided in several pilots:

- **Bells**: preserving historical bells cultural heritage by building a publicly available knowledge graph;
- **Organs**: building a knowledge graph about pipe organs;
- **Facets**: building a search engine for large collections of music documents;
- **Interlink**: connect entities and concepts hidden in digital music libraries and audiovisual archives;
- **Child**: music experience in childhood by building a knowledge graph of its historical experience;
- **Musicbo**: make Bologna's musical heritage available and accessible to the wide public;
- **Tunes**: influences between music tradition over centuries by interlinking melody collections of Dutch early popular music culture with other European collections;
- **Tonalities**: developing tools for the modal-tonal identification, exploration, and classification of monophonic and polyphonic notated music from the Renaissance to the twentieth century;
- **Access**: develop new ways to enhance participation and engagement in music for the general public, for those with hearing impairments and for those with physical disabilities;

- **Meetups**: support music historians and teachers by providing a Web tool that enables the exploration and visualisation of encounters between people in the musical world.

As stated in the introduction, the knowledge graphs used in this thesis belong to **Musicbo** (Section 3.2).

Chapter 2

Related Work

Selectional restrictions have been widely studied to obtain typed frameworks. Early approaches like Kozlowski et al. [32] have tried to use type constraints to aid in the generation of natural language sentences. This approach builds on top of previous developments in the use of selectional restrictions for Word Sense Disambiguation [51, 1] and shows the importance of type constraints to map *word senses* to their translation in another language, e.g. the verb *eat*, in german, can be translated to *essen* or *fressen* depending on the argument of the role `Agent`. Some research has been done to detect the misuse of natural language constructs in sentences. E. Chersoni et al. [11] have developed models to do anomaly detection of semantic roles, focusing their research to distinguish rare arguments from improbable ones. Improbable arguments are also the center of the research done by T.Warren et al. [62] who highlight the effect of selectional restrictions violations on eye movements in reading. A formal picture of the process of meaning combination with selectional restrictions is provided by N.Asher [2], showing how to integrate a rich system of types into semantic composition through lambda calculus. The formalisation is also capable of taking into account metaphores or fictional entities by defining types in an intensional manner instead of estentional.

To determine domain-dependent selectional restrictions means extracting emerging patterns from ontologies. Carriero et al. [9] have described a method for obtaining emerging patterns from Wikidata in the form of statistically frequent domain-property-range triplets. Indeed, [44, 8] have highlighted the difficulties in reusing the Wikidata ontology. Wikidata provides guidelines on its use but, most of the times, they are not enough to cover all the possible use cases.

Other approaches [13, 47, 18, 37] have also tried to extract patterns from ontologies by leveraging machine learning techniques and the graph structure of the ontology. Some of the approaches in literature suffer from high computational demand for very large graphs, leading the programs to crash for large amounts of triples.

Researchers have focused on improving graph embeddings by graph augmentation. D.D'Auria et al. [14] have showed how, in the clinical domain, linking concepts to biomedical ontologies can be beneficial for the link prediction task. B.Ding et al. [17] have shown how to leverage a richer entailment regime to improve knowledge graph embeddings. Another approach exploiting ontological reasoning is provided by N.Jain [30], which iteratively determines inconsistent predictions for the link prediction task and uses the negative samples to retrain the model.

For what concerns the OWL-Star to OWL mapping, as far as we know, no research has been done to provide this translation. Previous work [27] focuses on providing theoretical foundations of RDF* and RDF* to RDF mapping properties.

Chapter 3

Domain-dependent selectional restrictions

Selectional restrictions are useful to ensure syntactical and semantical compatibility over frames by limiting the possible types taken by argument roles.

Example 3.0.1. taken from [58]

Consider the following two sentences:

I break the window

The hammer breaks the window

The participants of the frame `break` have identical syntactical features, but they cover different roles. Following VerbNet's selectional restrictions, in the first sentence *I* is an *agent*, while *hammer* is an *instrument*. The difference concerns the semantic aspect of the sentence and, hence, it cannot be detected by syntactical trees. By constraining the roles, we give more context to the arguments, thus adding semantic specificity to the overall sentence.

These constraints are mainly used for **semantic role identification** and **class membership** tasks. Type estimation in RDF involves using reasoning techniques, such as rule-based inference or machine learning algorithms, to automatically determine the type of a resource based on its properties and relationships to other resources in the graph. This process can be used to validate the consistency of an RDF graph and to make inferences about resources and relationships in the graph that can be used to support various applications. In other words, selectional restrictions can aid rule based and machine learning based algorithms. This feature will be shown in Chapter 5.

When defining selectional restrictions, specificity is an issue: too general types tend to group together semantically different roles, while too specific ones may limit the flexibility of the model. This chapter addresses this issue and provides an implementation of domain-dependent selectional restrictions. Moreover, it provides a method to create frame-based ontology documentation.

3.1 Drawbacks of VN selectional restrictions

The more selectional restrictions are general, the more flexible the semantic parsing will be. This is the approach taken by VerbNet. This linguistic resource has been created to possibly model every scenario and every domain: the generality of the constraints ensure robustness, but they might also be useless in specific case scenarios.

As an example, let us say that we want to represent a domain specific ontology like the one represented by MusicBO [45] for musical knowledge. The hierarchy shown in Figure 1.1 does not add relevant knowledge to our representation, since it is not able to account for musical classes and properties.

Example 3.1.1. Consider the following sentence:

The orchestra played a sonata

In VerbNet, the roles and restrictions would be

$[\textit{The orchestra}]_{\texttt{AGENT[+organization]}}$ $[\textit{played}]_{\texttt{PREDICATE}}$ $[\textit{a sonata}]_{\texttt{THEME[+sound]}}$

VerbNet's restrictions have been designed to maximise the recall, i.e. to be sure that plausible entities do not get discarded. In some domains, however, the restrictions can be further specialized: if our domain focuses on classical music, we might reduce the `+sound` restriction to a newly created more specific typing like `classical_music`.
The advantages of this approach are threefold:

- better documentation of what the ontology describes;
- more specific embeddings for neuro-symbolic approaches;
- more constraining reasoning entailments.

3.2 About MUSICBO dataset

MusicBO Knowledge Graph stores information about the role of music in the city of Bologna from a historical and social perspective. It aims to satisfy the requirements of MusicBO pilot use case, namely conveying knowledge about music performances and encounters between musicians, composers, critics and historians in Bologna.
MusicBO knowledge graphs have been automatically extracted from the MusicBO pilot's corpus documents. Each initial textual document belongs to a different author: thus, the extracted knowledge graphs are not from the same source. This is leveraged in Chapter 5 and 6 to provide test results, following the intuition that validation and test set must be composed by documents belonging to authors not appearing in the train set, in order to obtain more meaningful results.
The text-to-KG process leverages **Abstract Meaning Representation** (AMR [4]) as an intermediate semantic representation. AMR graphs obtained through the semantic parsing of MusicBO textual corpus' sentences are then transformed into RDF/OWL knowledge graphs

based on FRED's [23] logic form by leveraging the AMR2Fred [36] tool. AMR implements neo-Davidsonian semantics; furthermore, graphs encoded following its formalism rely heavily on PropBank framesets [43]. Those features make AMR particularly suited to capture *"who is doing what to whom"* in a sentence. Equally, the RDF/OWL KGs obtained from the AMR2Fred tool are based on **neo-Davidsonian semantics** and are **event-centric**. Therefore, the knowledge graphs obtained through the AMR2Fred tool are appropriate to fulfil the MusicBO pilot's requirements. They inherently organise knowledge pivoting on framesets and frame arguments, making it easier to encode information about **events and situations** about the role of Bologna in European music history.

3.2.1 Text-to-KG process

The **text-to-KG** process leverages two modules: the **Polifonia Knowledge Extractor** pipeline and the **AMR2Fred** tool. Polifonia Knowledge Extractor pipeline uses Abstract Meaning Representation to parse sentences into semantic graphs. The two modules are orchestrated by the Machine Reading suite, which queries both components through the Text-to-AMR-to-FRED API and generates RDF *named graphs* from natural language text.
The Text2KG process for the automatic creation of the MusicBO KG can be broken down into its main steps as follows:

- **Input**: *.pdf* or *.docx* documents to apply the text-to-KG pipeline to;
- **Preprocessing**: due to the input format, pre-processing techniques like corrections, coreference resolution and sentence splitting have been applied;
- **Text-to-AMR**: the obtained sentences are submitted to a Text-to-AMR parser (**SPRING** [5]);

- **Filtering**: AMR parsers are not trained over historic musical documents, so their output may be inaccurate. To filter out wrong AMR graphs, a back-translation approach is applied to convert AMR graphs to their sentences using SPRING (the tool is invertible). Then, sentence embeddings with an high difference are discarded. In other words, considering e as the embedding function, f as the Text-to-AMR mapping, s as the sentence and `BLEURT` [55] as the similarity metric,

$$\texttt{BLEURT}(e(s), e(f^{-1}(f(s)))) < 0$$

  indicates which AMR graphs to discard. The function e can be considered as an large pre-trained language model (like **BERT** [16]);

- **AMR2Fred translation**: the graphs obtained in the previous steps are given to FRED [23] to produce OWL/RDF knowledge graphs, which are augmented in post-processing with Framester's semantic hub. Specifically, **Word Sense Disambiguation** (WSD) is applied to those nodes that do not have alignments to any linguistic resources. The alignment is implemented via `owl:equivalentClass` predicate:

```
<https://w3id.org/stlab/mr_data/MusicBO_120_1113_amr/History>
 <http://www.w3.org/2002/07/owl#equivalentClass>
  <https://w3id.org/framester/wn/wn30/instances/synset-history-noun-1>
```

The obtained knowledge graphs are **frame-based** and based on **Propbank**. The main difference from the standard Propbank representation (Section 1.2.1) is that the roles produced by the **AMR2Fred** are **localroles**, i.e. roles created by replacing `ARG0`, `ARG1` etc. with their labels parsed in **snake case**. This choice is due to the fact that Propbank frames are not informative enough and providing more meaningful role names can be beneficial for several tasks.

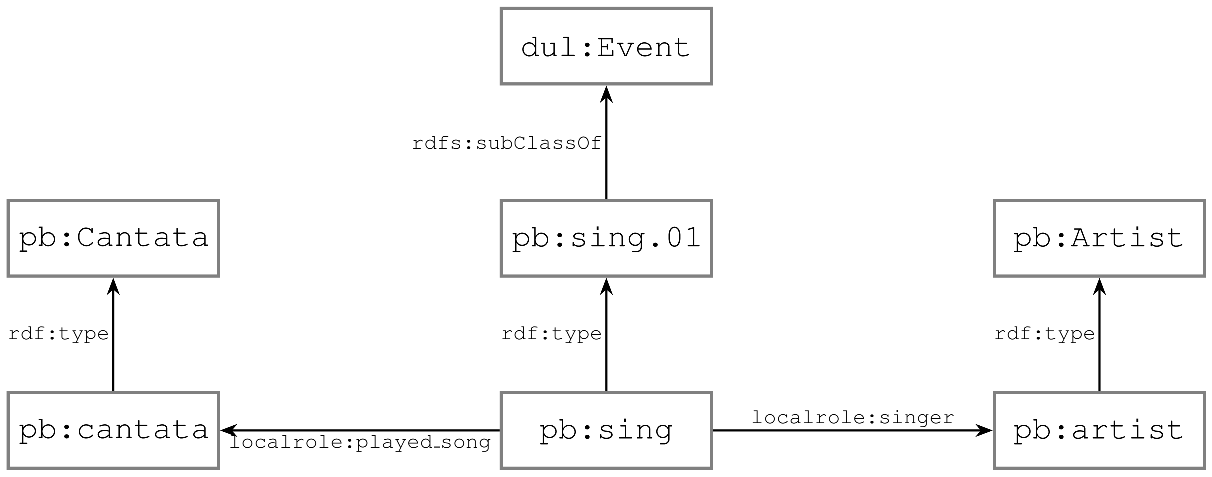


Figure 3.1: Example of a `sing-01` frame produced by FRED. Resources starting with an uppercase letter are *types*, lowecase are *instances* and subclasses of `dul:Event` are *frames*.

An example of a *frame* produced by FRED is shown in Figure 3.1. As previously stated, each *frame* produced by FRED is a subclass of `dul:Event` [7].

3.3 Association rule generation

The Knowledge Graphs extracted are fed to a selectional restriction extractor module which empirically estimates the types of the classes covering specific frame roles. Selectional restrictions force semantic constraints on arguments: a violation of these results in sentence anomalies. At the same time, restrictions can be helpful in understanding the contexts in which the argument is used, especially for domain-specific knowledge. The pipeline can be described as a multi-step approach:

1. **Roles identification**: we create a data structure in which we store each `<frame_type, role, object_type>` triple. For instance, considering the following triple

```
<https://w3id.org/framester/pb/data/style-01>
 <https://w3id.org/framester/pb/localrole/thing_being_styled>
  <https://w3id.org/stlab/mr_data/MusicBO_120_1113_amr/Music>
```

the `frame_type` (*`style-01`*) is taken by searching for subclasses of `dul:Event` [7]. The role (*`thing_being_styled`*) is obtained by searching through the local-roles of the frame. The `object_type` (*`Music`*) is extracted via `rdf:type` from the frame argument instance. Indeed, it must be noted that `<frame_type, role, object_type>` is not a real triple appearing in the knowledge graph, but the frame and frame argument instances are the resources linked by `role`;

2. **Type estimation**: based on the triples obtained in the previous step, we can empirically estimate the probability of having a certain class as the type of the argument of frame roles. This is done by simple probabilistic frequency: considering `frame_instance` and `object_instance` as the instances of `frame_type` and `object_type` respectively, the probability of having `object_type` given `frame_type,role` is calculated with

$$\frac{N(\texttt{frame\_instance,role,object\_instance})}{N(\texttt{frame\_instance,role})}$$

where N is a function which outputs the absolute frequency;

3. **Type generalization**: most of the obtained types for each `frame_type,role` are very similar. For example, considering `write-01` and `thing_written` as `frame_type, role` respectively, the output is composed by `Motet, Madrigal, Mazurka` and other classes that can be aggregated in order to obtain less specific types. Since the MusicBO knowledge graphs do not specify the entire class hierarchy, the aggregation must be applied by leveraging the WordNet alignments appearing inside the knowledge

graphs. Specifically, from the WordNet mapping of `object_type`, a SPARQL query is launched, visiting the class hierarchy through Framester. When a common generalization of multiple types is found, the probability of the new inferred type is simply estimated by adding the frequencies of the non-generalized classes.
The degree of generalization over the class hierarchies can be a fixed scalar value k which is manually set by the user of the tool. The default behaviour of the algorithm is, for every `frame_type` and `role`, to generalize the frame arguments to a unique class;

4. **Parsing to OWL-Star**: the obtained restrictions are parsed into an ontology language (Chapter 4) together with their probabilities.

The **Type Estimation** process can be considered as extracting **association rules** from the domain. Specifically, association rules highlight co-occurrence patterns inside the dataset by defining rules $A \Rightarrow B$, where A and B are called **itemsets**. The $\Rightarrow$ symbol does not imply logical implication (boolean) but implication with some *level of truth*.
The *level of truth* is strictly dependent from the following metrics:

- **Support count (σ)**: frequency of an itemset;
- **Support**: $\frac{\sigma(A \cup B)}{\sigma^*}$ where σ^* is the total number of association rules;
- **Confidence**: $\frac{\sigma(A \cup B)}{\sigma(A)}$.

In our case, an example of an association rule can be $\{$*style-01*, *thing_being_styled*$\} \Rightarrow \{$*Music*$\}$.
Other meaningful values that can be extracted regarding the selectional restrictions can be exploited by the **Kullback-Leibler divergence** (KL divergence), i.e. the difference of two probability distributions $D(P||Q) = \sum_x P(x)\ log \frac{P(x)}{Q(x)}$. The KL divergence can be exploited to perform a **local analysis** about how much a frame is informative regarding the semantic classes

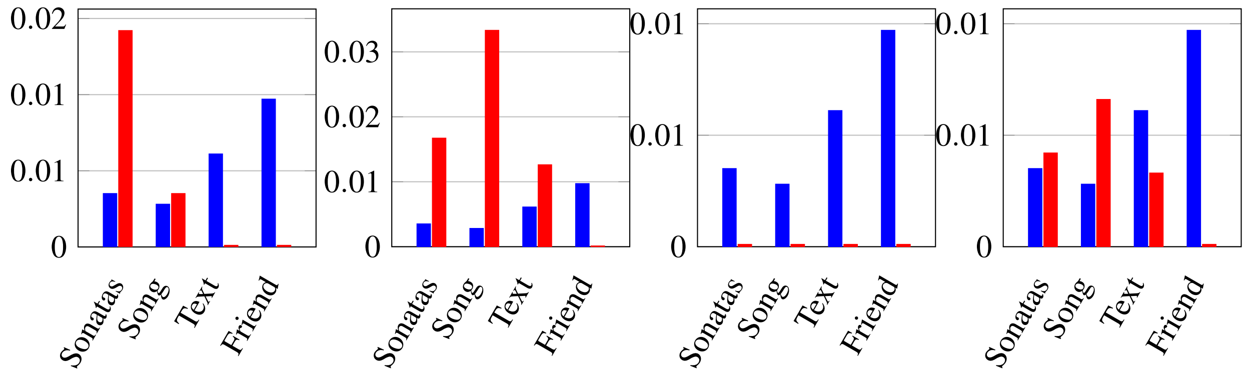
of their arguments. Considering f and c to be the *frame* and a possible *type* respectively, the metrics that can provide such insights are:

- **Selectional Preference**: $S(f) = D(P(c|f)||P(c))$, which expresses how much information the frame expresses over its argument types;
- **Selectional Association**: $A(f,c) = \frac{1}{S(f)} P(c|f)\ log \frac{P(c|f)}{P(c)}$, which expresses the contribution of the type c to the general preference of the frame f.

The analysis using these metrics over the MusicBO dataset can be seen in Chapter 6.

3.3.1 Ontology documentation through Selectional Restrictions

Ontology documentation provides information about the entities, relations, and properties in an ontology. The documentation should accurately represent the intended meaning of the knowledge represented in the ontology, and provide guidance on how the ontology should be used. The inclusion of selectional restrictions and competency questions in the documentation of an ontology can help to ensure that the knowledge representation is semantically consistent and free from errors, and accurately captures the intended meaning of the relationships between entities. Indeed, **competency questions** test the completeness of the knowledge representation by ensuring that the ontology can answer a set of representative questions about the domain. However, for some ontologies, documentation is very scarce and understanding what the ontology represents can become a difficult task. For this reason, defining a method to automatically extract competency questions can be beneficial in knowledge representation.
Theoretically, the best approach to obtain competency questions is to use generative language models that, from a triple, produce the relative questions. For this approach, we would need to define how to linearize the initial triple (converting the triple to a sentence) and, more importantly, we would need a huge amount of computational resources due to the generation of



(a) `compose-01` (b) `sing-01` (c) `record-01` (d) `write-01`

Figure 3.2: Bar plots showing the different posterior distributions given some frames for *Sonatas*, *Song*, *Text* and *Friend* types. Most informative results are achieved by more general frames like `write-01`.

questions for each possible fact appearing in the knowledge graph (**brute-force approach**). To reduce the complexity of the algorithm, we could use the definition given in Section 3.3 of **Selectional Preference**. Specifically, we could follow the intuition that, to understand the topic covered by a frame-based ontology, we could focus on **general purpose frames**. Indeed, while *specific frames* like `sing-01` tend to be used with a small set of *frame arguments*, *general frames* like `write-01` can be used in different contexts, thus being more informative if they occur with domain-relevant *types*. This intuition is shown in Figure 3.2.

Following this idea, we can devise a three stage approach:

- **Selectional Preference Filtering**: we filter out *specific frames* by keeping the *frames* with high selectional preference.

- **Linearization** [31]: the triples are linearized into a string by only considering the last part of the uri, i.e. the frame name, the predicate and class name respectively.

- **Question generation**: the **MixQG** [39] model is employed to produce questions from the linearization of the retrieved data points.

The question generation model (**MixQG**) is a pretrained text-to-text model based on **T5** capable of generating questions from natural language statements. This model has been designed to work with paragraphs, hence it would provide better results if we started from the original text instead of its knowledge graph representation. However, with a good linearization technique, the model is also capable of outputting reliable results starting from knowledge graphs, hence proving its worth in case scenarios in which there is no input text associated to the ontologies.
For the question generation, the model has been applied two times per statement based on the part of sentence to which the question referred, i.e. subject or object. The results can be seen in Chapter 6.

Chapter 4

Ontology generation and Framester augmentation

The restrictions and probabilities do not serve as mere data, but can be used to improve existing linguistic resources with domain-specific data. In Section 1.2.5 we discussed about **Framester**, which is an ontological web resource acting as a hub between multiple linguistic resources. Framester is **frame-based**, meaning that it follows Fillmore's linguistic theory: the main constructs are frames and every situation can be represented through them.
The same assumption is followed by FRED, in which frames are represented as subclasses of **dul:Event**. The selectional restriction extractor tool, applied to FRED-like KGs, hence focuses on Framester frames, i.e. the frequency data can be added into domain-specific frames inside Framester.
In order to do so, a crucial step is to describe the extracted selectional restrictions with their probabilities into a language for ontologies, which can subsequently be linked to linguistic resources.
In the last years, new languages for ontology representation have gained popularity, namely

RDF* and its OWL extension **OWL-Star**. These new languages provide new vocabulary to represent **statements about statements** and **probability**. Statements about statements can also be represented by RDF reification, but the knowledge graph becomes very prolix and difficult to interpret especially for graphical representations. Hence, in some cases it might be useful to be able to switch from one language to the other.
This chapter shows the implementation of an OWL-Star to OWL compiler together with an analysis of its mapping properties. Moreover, the chapter discusses about the integration of the obtained OWL ontologies into the Framester web resource.

4.1 RDF*

RDF* [50] is an extension of RDF which provides a compact alternative to **reification** (discusses in Section 1.4.1). Reification in RDF is usually verbose, hard to understand and hard to query. RDF-Star introduces **quoted triples** to reduce the verbosity, allowing to represent ”statement of statements” in a more compact manner.

Example 4.1.1. Consider the following sentence:

Mark said that the sun is shining

The sentence can be represented in RDF as

```
:Sun :action :shines .
:_triple rdf:type rdf:Statement ;
       rdf:subject :Sun ;
       rdf:predicate :action ;
       rdf:object :shines .
```

```
:Mark :say :_triple .
```

while in RDF* as

```
:Mark :say << :Sun :action :shines >> .
```

The triple between $<<>>$ characters is a quoted triple. It is different from standard triples used in RDF (**assertion triples**) since it is not factual that *"The sun is shining"*, but it is a claim made by *"Mark"*.
It may happen that a triple has both types. This is the case with scenarios in which we want to represent claims that are also factual.
The following definition extends the one given in Definition 1.1.1.

Definition 4.1.1. An **RDF* triple** is a 3-tuple defined recursively as:

- any RDF triple is an RDF* triple;
- if t and t' are RDF* triples, $s \in U \cup B$, $p \in U$, $o \in U \cup B \cup L$, then (t, p, o), (s, p, t) and (t, p, t') are RDF* triples.

4.2 OWL-Star

RDF* has the same pitfalls of RDF. It is not possible to define existential and universal restrictions with RDF*, thus it is not possible to check precisely the logical consistency of the ontology. **OWL-Star** [42] is a non-standard extended vocabulary that tackles this problem like OWL with RDF.
Since RDF* only extends RDF with additional triples to render reification in a more efficient way, it could be possible to introduce OWL axioms with the RDF* syntax. This approach, anyway, is verbose and can be simplified with the vocabularies introduced by OWL-Star

Example 4.2.1 (taken from [42])**.** Consider the sentence

Finger is a part of hand

The sentence could be rendered in OWL in the following way

```
:finger rdfs:subClassOf [
 a owl:Restriction ;
 owl:onProperty :part-of ;
 owl:someValuesFrom :hand
]
```

and in OWL-Star in the following way

```
<<:finger :part-of :hand>> owlstar:interpretation
   owlstar:AllSomeInterpretation .
```

In other words, OWL-Star provides a simplified and extended way to deal with existential and universal restrictions. It also provides a specific vocabulary for **Temporal, Contextual and Fuzzy Logic**:

- **Temporal and Contextual Logic**: through the predicate `owlstar:context` it is possible to correlate triples with temporal information, i.e. frame a possibly changing situation into a particular time.
- **Fuzzy Logic**: through the predicate `owlstar:probability` it is possible to correlate triples with probability information. The probabilities can be associated with existential and universal restrictions for more advanced cases.

Example 4.2.2 (Contextual Logic)**.** Consider the sentence

John is the CEO of the company

The sentence could be rendered in OWL-Star in the following way

```
<<:John :hasRole :CEO>> owlstar:context :t1 .
:tf1 rdf:type bfo:TemporalRegion .
```

where `bfo` is the namespace for the **Basic Formal Ontology**.

Example 4.2.3 (Fuzzy logic)**.** Considering we have previous knowledge about the probability of a certain triple, it is possible to indicate its probability in the following way:

```
<<:bob foaf:friendOf :alice>> os:probability 0.9 ^^ xsd:float .
```

OWL-Star *interpretations* (existential, universal, cardinality restrictions) can be mapped into OWL restrictions. Table 4.1 shows the mapping patterns.

4.3 Mapping OWL-Star to OWL

In this section, the work by [27] is extended for considering OWL-Star mappings.
Before describing the details of the mapping, a few definitions over RDF* triples are needed.

4.3.1 RDF* formal foundations

Definition 4.3.1 (Ψ_{rdf})**.** Consider R^* to denote the infinite set of RDF* triples and R to denote the infinite set of RDF triples. Then we define as Ψ_{rdf} the set difference $\Psi_{rdf} = R^* \setminus R$, namely the resources not available inside RDF.

In standard RDF reification, triples are given a specific IRI or blank node of type `rdf:Statement`, which is later used to specify the `rdf:Subject`, `rdf:Predicate` and `rdf:Object` of

OWL-Star	**OWL**
$<<?s\ ?p\ ?o>>$ `os:interpretation` `os:AllSomeInterpretation`	$?s$ `rdfs:subClassOf` [`rdf:type` `owl:Restriction` ; `owl:onProperty` $?p$; `owl:someValuesFrom` $?o$]
$<<?s\ ?p\ ?o>>$ `os:interpretation` `os:AllOnlyInterpretation`	$?s$ `rdfs:subClassOf` [`rdf:type` `owl:Restriction` ; `owl:onProperty` $?p$; `owl:allValuesFrom` $?o$]
$<<?s\ ?p\ ?o>>$ `os:interpretation` `os:AllNumberInterpretation`; `os:min` 5 ; `os:max` 5	$?s$ `rdfs:subClassOf` [`rdf:type` `owl:Restriction` ; `owl:onProperty` $?p$; `owl:cardinality` 5]
$<<?s\ ?p\ ?o>>$ `os:interpretation` `os:SomeSomeInterpretation`	[`rdf:type` $?s$] $?p$ [`rdf:type` $?o$]

Table 4.1: OWL-Star to OWL mappings

the reified triple. It is crucial to define such mappings for a precise OWL-Star to OWL translation.

Definition 4.3.2 (Triple-to-ID mapping)**.** A **triple-to-ID mapping** is a injective function *id*: $R^* \longrightarrow U \cup B$.

The *id* function could be defined in a incremental fashion. For instance, for an RDF* graph G, each triple $t_i \in G$. $t_i = (s_i, p_i, o_i)$ one possible implementation is the following

$$id(t_i) = \texttt{:\_i}$$

with i spanning the triples inside G.

This approach depends on graph G and on a predefined ordering between its triples. For simplicity, we adopt this approach: the *id* mapped triples are constrained with RDF reification, hence triples with the same *id* mapping belonging to different RDF* graphs do not share semantics due to local restrictions.

Another approach is to define the mapping by concatenating s,p,o. In this way, the *id* function would provide a one-to-one mapping between triples and output URIs independently of the graphs they belong to, but the results would be prolix and difficult to read.

Definition 4.3.3 (Υ function)**.** Let *id* be a triple-to-ID mapping. An $\Upsilon : R^* \longrightarrow \mathscr{P}(U \cup B \cup L)$ function can be defined such that

$$\Upsilon(t) = \{s, p, o\} \cup \{x \in \Upsilon(t') \mid t' \in \{s, o\} \cap R^*\}$$

In other words, $\Upsilon(t)$ is the set of RDF terms and RDF* triples mentioned in t.

The definition can be extended to graphs as follows:

$$\forall G.\ \Upsilon(G) = \bigcup_{t \in G} \Upsilon(t)$$

Definition 4.3.4 (Ω function)**.** The $\Omega : \mathscr{P}(R^*) \longrightarrow \mathscr{P}(R^*)$ function can be defined as

$$\Omega(G) = G \cup (\Upsilon(G) \cap R^*)$$

Ω indicates the set of all RDF* triples inside a graph G, including those triples that appear recursively inside other triples. The expression $\mathscr{P}(\mathscr{R}^*)$ indicates the set of all possible subsets of R^*, i.e. the set of all possible R^* graphs.

Definition 4.3.5 ($\texttt{REDUCE}^{id}$)**.** The ***id*-specific reduction** of an RDF* triple $t = (x_1, x_2, x_3)$ called $\texttt{REDUCE}^{id}(t)$ is a mapping $R^* \longrightarrow R$ such that it holds that, $\forall i \in \{1, 2, 3\}$,

$$\texttt{REDUCE}^{id}(x_i) = \begin{cases} id(x_i) & \text{if } x_i \in R^* \ \wedge \ i \in \{1,3\} \\ x_i & \text{otherwise} \end{cases} \tag{4.1}$$

Definition 4.3.5 defines a **reduction** operation over any possible triple, i.e. a mapping strategy able to disambiguate R^* triples into R. The additional condition $i \in \{1,3\}$ is due to the fact that predicates cannot be RDF* triples (Definition 4.1.1).
The function $\texttt{REDUCE}^{id}$ is crucial for the definition of the RDF*-to-RDF translator, but it is not sufficient to preserve the initial information. Indeed, applying $\texttt{REDUCE}^{id}$ does not deal with reification, which is the main update introduced by RDF*. In other words, we lose the information provided by **statements about statements**.
For this reason, another operator called $\texttt{REIF}^{id}$ has to be introduced.

Definition 4.3.6. The ***id*-specific standard RDF representation of RDF*** is an RDF*-to-RDF mapping $m : \mathscr{P}(R^*) \longrightarrow \mathscr{P}(R)$

$$m(G) = \{\texttt{REDUCE}^{id}(t) \mid t \in \Omega(G)\} \cup \bigcup_{t \in \Upsilon(G) \cap R^*} \texttt{REIF}^{id}(t)$$

where

$$\begin{aligned} \forall t.t \in R^*,\ \texttt{REIF}^{id}(t) = \{&(id(t)\ \texttt{rdf:type rdf:Statement}), \\ &(id(t)\ \texttt{rdf:subject}\ s_1), \\ &(id(t)\ \texttt{rdf:predicate}\ p), \\ &(id(t)\ \texttt{rdf:object}\ o_1)\} \end{aligned}$$

with $\texttt{REDUCE}^{id}(t) = (s,p,o)$, $s_1 = id(s)$ if $s \in R^*$ else $s_1 = s$, $o_1 = id(o)$ if $o \in R^*$ else $o_1 = o$.

4.3.2 OWL-Star formal foundations

With regards to the OWL-Star vocabulary, additional operators must be defined to deal with OWL-Star *interpretations*, Contextual and Fuzzy logic.
As done for RDF* triples, we must first formally define the set of OWL-Star triples.

Definition 4.3.7 (Ψ_{owl})**.** Consider O^* to denote the infinite set of OWL-Star triples and O to denote the infinite set of OWL triples. Then we define as Ψ_{owl} the set difference $\Psi_{owl} = O^* \setminus O$, namely the resources not available inside OWL.

Trivially, $R \subseteq O$ and $R^* \subseteq O^*$ since both O and O^* extend the vocabulary of R and R^* respectively. At the same time, $R \subseteq R^*$ and $O \subseteq O^*$ due to Definition 4.1.1.
To ease the readability of the next definitions, we define the function Γ over existential, universal and cardinality restrictions and the function `blank`.

Definition 4.3.8 (Γ functions)**.** We define the following set of functions $\Gamma_{\exists}, \Gamma_{\forall}, \Gamma_{cardinality} : (U \cup B, U \cup B \cup L) \rightarrow B$:

- $\Gamma_{\exists}(p,o) =$ `[rdf:type owl:Restriction; owl:onProperty` p`;`
  `owl:someValuesFrom` o`]`
- $\Gamma_{\forall}(p,o) =$ `[rdf:type owl:Restriction; owl:onProperty` p`;`
  `owl:allValuesFrom` o`]`
- $\Gamma_{cardinality}(p,o) =$ `[rdf:type owl:Restriction; owl:onProperty` p`;`
  `owl:cardinality` o`]`

Definition 4.3.9 (`blank` function)**.** Given a OWL class c, the function `blank` $: U \rightarrow B$ is defined as

$$\texttt{blank}(c) = [\texttt{rdf:type}\ c]$$

OWL-Star *interpretations* can be translated using Table 4.1. Hence, we define the following $\texttt{INTERPRET}^{id}$ function to parse OWL-Star existential, universal and cardinality restrictions.

Definition 4.3.10 ($\texttt{INTERPRET}^{id}$)**.** The *id***-specific interpretation** of a triple $t \in O^*$. $t = (s, p, o)$ called $\texttt{INTERPRET}^{id}(t)$ is a mapping $O^* \longrightarrow O$ such that, if $s \in O^*$ and $p = \texttt{os:interpretation}$ with $s = (x_1, x_2, x_3)$,

$$\texttt{INTERPRET}^{id}(t) = \begin{cases} (x_1, \texttt{rdfs:subClassOf}, \Gamma_{\exists}(x_2, x_3)) & \text{if } o = \exists \\ (x_1, \texttt{rdfs:subClassOf}, \Gamma_{\forall}(x_2, x_3)) & \text{if } o = \forall \\ (x_1, \texttt{rdfs:subClassOf}, \Gamma_{cardinality}(x_2, x_3)) & \text{if } o = C \\ (\texttt{blank}(x_1), x_2, \texttt{blank}(x_3)) & \text{if } o = S \\ t & \text{otherwise} \end{cases} \tag{4.2}$$

where $\forall$, $\exists$, C and S stand for `os:AllOnlyInterpretation`, `os:AllSomeInterpretation`, `os:AllNumberInterpretation` and `os:SomeSomeInterpretation`. If $t \in O^*$. $t = (s, p, o)$ and $s \notin O^* \vee p \neq \texttt{os:interpretation}$, then $\texttt{INTERPRET}^{id}(t) = t$.

Thus, the $\texttt{INTERPRET}^{id}$ operator is defined over all possible O^* triples: if the input triple does not show any of the patterns appearing in Table 4.1, then $\texttt{INTERPRET}^{id}$ does not apply any changes, i.e. it becomes the identity function. The same pattern is shown for all the functions introduced to deal with O^* triples (Definition 4.3.11, 4.3.12): this allows us to be more lenient on the function definitions, avoiding to use mappings that are not defined with specific inputs.

The temporal axioms can be translated using the **OWL-Time** ontology[1]. The ontology provides a vocabulary for expressing facts about topological (ordering) relations among instants

[1]`https://www.w3.org/TR/owl-time/`

and intervals, together with information about durations, and about temporal position including date-time information. In our case, we model contexts without using specific temporal information (like duration, start/end of interval etc.) but just by specifying a time entity with an undefined temporal position.

Definition 4.3.11 ($\texttt{TIME}^{id}$)**.** The ***id*-specific temporal reduction** of a triple $t \in O^*.\, t = (s,p,o)$ called $\texttt{TIME}^{id}(t)$ is a mapping $O^* \longrightarrow O$ such that

$$\texttt{TIME}^{id}(t) = \begin{cases} (s, \texttt{time:hasTime}, o) & \text{if } p = \texttt{os:context} \\ t & \text{otherwise} \end{cases} \tag{4.3}$$

The object o is defined as $(o, \texttt{rdf:type}, \texttt{time:TemporalEntity})$ if $p = \texttt{os:context}$.

Note that Definition 4.3.11 may be too confident about the type of the context. Indeed, in OWL-Star contexts are not only defined with temporal properties through `bfo:TemporalRegion`, but may generalize over different situations. OWL-Star does not specify the range of `os:context`, but here we map to the OWL-Time ontology on a statistical basis.
The probability predicate `os:probability`, instead, is simply translated by removing the prefix and creating a new DataType property for probabilistic relations. This is formally defined in Definition 4.3.12.

Definition 4.3.12 ($\texttt{PROB}^{id}$)**.** The *id*-specific probability reduction of a triple $t \in O^*.\, t = (s,p,o)$ called $\texttt{PROB}^{id}(t)$ is a mapping $O^* \longrightarrow O$ such that

$$\texttt{PROB}^{id}(t) = \begin{cases} (s, \texttt{:probability}, o) & \text{if } p = \texttt{os:probability} \\ t & \text{otherwise} \end{cases} \tag{4.4}$$

The predicate `:probability` is defined as

$$(\texttt{:probability}, \texttt{rdf:type}, \texttt{owl:DatatypeProperty})$$

if $p = \texttt{os:probability}$.

We can now extend Definition 4.3.6 to also consider the OWL-Star vocabulary.

Definition 4.3.13. Let $h = (\texttt{INTERPRET}^{id} \circ \texttt{TIME}^{id}) \circ \texttt{PROB}^{id}$, where $\circ$ is the function composition.
The *id***-specific standard OWL representation of OWL-Star** is an OWL-Star-to-RDF mapping $m : \mathscr{P}(O^*) \longrightarrow \mathscr{P}(O)$ that

$$m(G) = \{h(\texttt{REDUCE}^{id}(t)) \mid t \in \Omega(G)\} \cup \bigcup_{t \in \Upsilon(G) \cap O^*} \texttt{REIF}^{id}(h(t))$$

4.4 About OWL-Star-to-OWL mapping properties

The previous section introduced the formal foundations of the OWL-Star-to-OWL mapping. When translating an ontology from a language to another, we want to ensure that certain properties hold, i.e. that the translation does not alter the semantics of the original knowledge. One desiderable property for such mappings is to be *information preserving*, meaning that the mapping needs to be *invertible*, thus allowing to retrieve the initial data from the translated knowledge.

Definition 4.4.1 (Information preserving)**.** An OWL-Star-to-OWL mapping $m : O^* \longrightarrow O$ is **information preserving** for a graph G if $\exists m' : O \longrightarrow O^*.\ m'(m(G)) = G$.

Even though the knowledge representation with OWL and OWL-Star may be structurally different, this property ensures that m' is capable of reproducing the original ontology, i.e. $m(G)$ retains the variance of the original data.
Note that the function *id* may map triples $t \in \Omega(G)$ to resources that already appear inside graph G. In these cases, the mapping is said to be in *conflict* with G. This is an important

property to avoid when implementing the translator: when in *conflict* the mapping may inadvertedly output semantically different graphs, thus not preserving the original information of the input data.
We generalize the notion of *conflict* to all possible mappings.

Definition 4.4.2 (Conflict)**.** Consider a mapping $\eta : O^* \longrightarrow O^* \cup U \cup B$ and a graph $G^* \in \mathscr{P}(O^*)$. The mapping η is in **conflict** with G^* if $\exists t \in G^*.\ \eta(t) \in \Upsilon(G^*)$

Hence, *conflict* is also defined for functions that output triples (i.e. $\texttt{REIF}^{id}$, $\texttt{INTERPRET}^{id}$, $\texttt{TIME}^{id}$, $\texttt{PROB}^{id}$). This extention of the notion will be crucial for the next theorems, since it allows us to define *invertible* operations over certain types of graphs.
For proving that Definition 4.3.13 is *information preserving*, we need to prove a series of lemmas about the invertibility of $\texttt{INTERPRET}^{id}$,$\texttt{TIME}^{id}$ and $\texttt{PROB}^{id}$. Indeed, proving that a mapping is *information preserving* follows from proving the existence of the inverse mapping. Since the inverse of the composition of two functions f and g is the composition of the inverses of the functions (i.e. $(f \circ g)^{-1} = f^{-1} \circ g^{-1}$), we do not focus on h but on the operators that it applies.

Theorem 1. *A function f is invertible if and only if it is injective.*

Beware that a function is *injective* if and only if $\forall x_1, x_2 \in dom(f).\ x_1 \neq x_2 \rightarrow f(x_1) \neq f(x_2)$. From this we can proceed in proving the invertibility of h.

Lemma 2. *Let Σ be the set of OWL-Star graphs such that for every $G^* \in \Sigma$ it holds that $\texttt{PROB}^{id}$ is not in conflict with G^*. The function $\texttt{PROB}^{id}$ is invertible for Σ.*

Proof. Consider two triples $t_1, t_2 \in G^*.\ t_1 = (s_1, p_1, o_1) \wedge t_2 = (s_2, p_2, o_2)$ with $t_1 \neq t_2$. We can proceed by cases:

- if $(p_1 = p_2) \wedge (p_1 = \texttt{os:probability})$: considering $t'_1 = (s'_1, p'_1, o'_1)$, $t'_2 = (s'_2, p'_2, o'_2)$ to be $t'_1 = \texttt{PROB}^{id}(t_1)$ and $t'_2 = \texttt{PROB}^{id}(t_2)$, we know that $(p'_1 = p'_2) \wedge (p'_1 = \texttt{:probability})$. By $\texttt{PROB}^{id}$ we know that $\forall i.\ i \in \{1,2\} \rightarrow (s_i = s'_i) \wedge (o_i = o'_i)$: since, by hypothesis, we know that $t_1 \neq t_2$, it must be that $(s_1 \neq s_2) \vee (o_1 \neq o_2)$, hence $(s'_1 \neq s'_2) \vee (o'_1 \neq o'_2)$. Thus, $\texttt{PROB}^{id}(t_1) \neq \texttt{PROB}^{id}(t_2)$

- if $(p_1 = p_2) \wedge (p_1 \neq \texttt{os:probability})$: in this case $\texttt{PROB}^{id}$ is the identity function which is injective.

- if $(p_1 \neq p_2) \wedge (p_1 = \texttt{os:probability})$: considering $t'_1 = (s'_1, \texttt{:probability}, o'_1)$, $t'_2 = (s'_2, p'_2, o'_2)$ to be $t'_1 = \texttt{PROB}^{id}(t_1)$ and $t'_2 = \texttt{PROB}^{id}(t_2)$, we know that $p'_2 \neq \texttt{:probability}$ since t_1 and t_2 belong to G^* which is not in conflict with $\texttt{PROB}^{id}$. Hence, $p'_1 \neq p'_2$ and $\texttt{PROB}^{id}(t_1) \neq \texttt{PROB}^{id}(t_2)$.

- if $(p_1 \neq p_2) \wedge (p_2 = \texttt{os:probability})$: analogous to before

Since $\forall G^* \in \Sigma, \forall t_1, t_2.\ t_1 \neq t_2 \rightarrow \texttt{PROB}^{id}(t_1) \neq \texttt{PROB}^{id}(t_2)$, $\texttt{PROB}^{id}$ is *injective* and, for Theorem 1, *invertible*.

□

Lemma 3. *Let Σ be the set of OWL-Star graphs such that for every $G^* \in \Sigma$ it holds that $\texttt{TIME}^{id}$ is not in conflict with G^*. The function $\texttt{TIME}^{id}$ is invertible for Σ.*

Proof. Analogous to Lemma 2 □

Lemma 4. *Let id be an invertible triple-to-ID mapping and let Σ be the set of OWL-Star graphs such that for every $G^* \in \Sigma$ it holds that $\texttt{REIF}^{id}$ is not in conflict with G^*. The function $\texttt{REIF}^{id}$ is invertible for Σ.*

Proof. Analogous to Lemma 2 □

Before proving that $\texttt{INTERPRET}^{id}$ is invertible, we must first state the invertibility of $\Gamma_\exists$, $\Gamma_\forall$, $\Gamma_{cardinality}$ and `blank` since the operator depends on them.

Lemma 5. *The functions $\Gamma_\exists$, $\Gamma_\forall$, $\Gamma_{cardinality}$ and `blank` are invertible*

Proof. We will prove the invertibility of $\Gamma_\exists$ since the proofs on the other functions are analogous.
Consider two pairs of predicate and object (p_1, o_1), (p_2, o_2) such that $(p_1, o_1) \neq (p_2, o_2)$. By applying the $\Gamma_\exists$ function, we obtain `[rdf:type owl:Restriction; owl:onProperty` p_1`; owl:someValuesFrom` o_1`]` and `[rdf:type owl:Restriction; owl:onProperty` p_2`; owl:someValuesFrom` o_2`]`. Trivially, $\Gamma_\exists(p_1, o_1) \neq \Gamma_\exists(p_2, o_2)$ due to the hypothesis that $(p_1 \neq p_2) \vee (o_1 \neq o_2)$. Thus, by Theorem 1, the function is invertible.

□

Lemma 6. *Let Σ be the set of OWL-Star graphs such that for every $G^* \in \Sigma$ it holds that $\texttt{INTERPRET}^{id}$ is not in conflict with G^*. The function $\texttt{INTERPRET}^{id}$ is invertible for Σ.*

Proof. We will prove the invertibility of $\texttt{INTERPRET}^{id}$ on the sub-case in which the object is `os:AllSomeInterpretation` since the other proofs are analogous.
Consider two triples $t_1, t_2 \in G^*$. $t_1 = (s_1, p_1, o_1) \wedge t_2 = (s_2, p_2, o_2)$ and let $t'_1 = \texttt{INTERPRET}^{id}(t_1) = (s'_1, p'_1, o'_1)$, $t'_2 = \texttt{INTERPRET}^{id}(t_2) = (s'_2, p'_2, o'_2)$.
We can proceed by cases:

- if $\bigwedge_{i=1}^{2} ((s_i \notin O^*) \vee (p_i \neq \texttt{os:interpretation}))$: trivial, we obtain the identity function which is invertible.

- if $\bigoplus_{i=1}^{2} ((s_i \notin O^*) \vee (p_i \neq \texttt{os:interpretation}))$: consider that t_2 is the triple that does not change, i.e. $t_2 = t'_2$. Hence, by hypothesis that the mapping is not *conflictual* and $t_1 \neq t_2$, then $t'_1 \neq t'_2$, i.e. by Theorem 1 the function is invertible. The same proof can be done in case $t_1 = t'_1$.

- otherwise: consider $s_1 = (x_1, y_1, z_1)$, $s_2 = (x_2, y_2, z_2)$. Then:
  - if $(o_1 = o_2) \wedge (o_1 = \texttt{os:AllSomeInterpretation})$: we know that $o_1 = o_2 \wedge p_1 = p_2$ so, by hypothesis $t_1 \neq t_2$, we know that $s_1 \neq s_2$. Since $(x_1 \neq x_2) \vee (y_1 \neq y_2) \vee (z_1 \neq z_2)$, then $(s'_1 \neq s'_2) \vee (o'_1 \neq o'_2)$ because $x_1 = s'_1$, $x_2 = s'_2$ and by Lemma 5. Hence, $t'_1 \neq t'_2$ and by Theorem 1, the function is invertible.
  - if $(o_1 \neq o_2) \wedge (o_1 = \texttt{os:AllSomeInterpretation})$: we know that $t'_2 = t_2$. Hence, by hypothesis that the mapping is not *conflictual* and $t_1 \neq t_2$, then $t'_1 \neq t'_2$, i.e. by Theorem 1 the function is invertible.
  - if $(o_1 \neq o_2) \wedge (o_1 = \texttt{os:AllSomeInterpretation})$: proof analogous to the previous case.
  - if $(o_1 \neq \texttt{os:AllSomeInterpretation}) \wedge (o_2 \neq \texttt{os:AllSomeInterpretation})$: trivial, we obtain the identity function which is invertible.

$\square$

For Theorem 7 we will assume that every OWL-Star graph G is *redundancy complete*, meaning that $\forall t \in G, \forall t' \in t \rightarrow t' \in G$. We can now prove the existence of an *information preserving* mapping for a subset of OWL-Star graphs.

Theorem 7. *Let id be a triple-to-ID mapping and let Σ be the set of OWL-Star graphs such that for every $G^* \in \Sigma$ it holds that*

- *id, $\texttt{REIF}^{id}$, $\texttt{INTERPRET}^{id}$, $\texttt{TIME}^{id}$, $\texttt{PROB}^{id}$ are not in conflict with G^*.*
- *G^* is redundancy complete.*
- *G^* does not use the RDF reification vocabulary.*

The id-specific standard OWL representation of OWL-Star is information preserving for G^*.

Proof. Let us define a mapping m' as per Definition 4.4.1. For every OWL graph G, let $\Re(G)$ be the set of all 5-tuples (x,s,p,o,RS) with $x,s,p,o \in (U \cup B \cup L)$ and $RS \subseteq G$ such that

$$\begin{aligned} RS = \{ & (x \,\texttt{rdf:type rdf:Statement}), \\ & (x \,\texttt{rdf:subject}\, s), \\ & (x \,\texttt{rdf:predicate}\, p), \\ & (x \,\texttt{rdf:object}\, o)\} \end{aligned}$$

Let $N(G) = \{t \in G \mid t \notin RS \; \forall (x,s,p,o,RS) \in \Re(G)\}$, i.e. $N(G)$ is the set of triples belonging to G that do not use the RDF reification vocabulary, considering that G is the *id*-specific standard OWL representation of G^*. Due to the hypothesis that G^* is *redundancy complete*, $N(G)$ preserves the triples (s,p,o) and loses information about the reification, which is however retained by the *id* mapping.
Note that if $G^* \notin \Sigma$ and G is the *id*-specific standard OWL representation of G^* then it would be impossible for m' to differentiate whether a reified triple t must be translated using the RDF* syntax or RDF: with the hypothesis that $G^* \in \Sigma$ we avoid this problem.
Since *id* is *invertible* (Definition 4.3.2) and not in conflict with G^* by hypothesis, $\forall (x,s,p,o,RS) \in \Re(G)$ we have that x is the image of mapping *id* and it is *unique*. Hence, we may define a mapping $\texttt{DEREIF}^{\Re(G)}$ such that $dom(\texttt{DEREIF}^{\Re(G)}) = \{x \mid (x,s,p,o,RS) \in \Re(G)\}$ and $\texttt{DEREIF}^{\Re(G)}(x) = (s,p,o)$. As our last preliminary, we introduce a recursively-defined mapping $\texttt{REVERSE}^{\Re(G)}$ that maps every OWL triple $t = (x_1,x_2,x_3)$ to an OWL-Star triple $t^* = (x_1^*,x_2^*,x_3^*)$ such that

$$\forall i \in \{1,2,3\}.\; x_i^* = \begin{cases} \texttt{REVERSE}^{\Re(G)}(\texttt{DEREIF}^{\Re(G)}(x_i)) & \text{if } x_i \in dom(\texttt{DEREIF}^{\Re(G)}) \\ x_i & \text{otherwise} \end{cases} \tag{4.5}$$

We can now define m' as

$$m'(G) = \{t' \mid \forall t \in N(G).\ t' = \texttt{REVERSE}^{\Re(G)}(h'(t))\} \tag{4.6}$$

where h' is the inverse of h (Definition 4.3.13). We know h' exists since h is the composition of invertible functions by hypothesis.

The crux of this proof regards showing that m' is indeed the inverse of m. For this let us proceed by cases on G^* knowing that the graph is *redundancy complete*:

- if G^* is the empty graph then $m(G^*) = m(\{\}) = \{\}$ and $m'(\{\}) = \{\}$, so $m'(m(\{\})) = \{\}$.

- if $G^* = \{(x,y,z),(s,p,o)\}$ with $x = (s,p,o)$, then $m(G^*) = \{h(id(x),y,z), h(s,p,o)\} \cup \{\texttt{REIF}^{id}(h(s,p,o))\}$. Hence, $m'(m(G^*)) = \{(x,y,z),(s,p,o)\}$: we are mapping triples in $N(G)$, so $\texttt{REIF}^{id}((h(s,p,o))$ is not considered by m'. As a consequence, $m'(m(G^*)) = G^*$.

- if $G^* = \{(x,y,z),(s,p,o)\}$ with $z = (s,p,o)$: analogous to the previous case.

- if $G^* = \{(x,y,z),(s_1,p_1,o_1),(s_2,p_2,o_2)\}$ with $x = (s_1,p_1,o_1)$ and $z = (s_2,p_2,o_2)$, then $m(G^*) = \{h(id(x),y,id(z)), h(s_1,p_1,o_1), h(s_2,p_2,o_2)\} \cup \{\texttt{REIF}^{id}(h(s_1,p_1,o_1)),$ $\texttt{REIF}^{id}(h(s_2,p_2,o_2))\}$. So, $m'(m(G^*)) = \{(x,y,z),(s_1,p_1,o_1),(s_2,p_2,o_2)\}$ and $m'(m(G^*)) = G^*$.

- if $G^* = \{(x,y,z)\}$ then $m(G^*) = \{h(x,y,z)\}$. Thus, $m'(\{h(x,y,z)\}) = \{(x,y,z)\}$ and $m'(m(G^*)) = G^*$.

- Consider $G_1, G_2 \subseteq G^*$ such that $m'(m(G_1)) = G_1$ and $m'(m(G_2)) = G_2$. Then $m'(m(G_1 \cup G_2)) = G_1 \cup G_2$ because $m'(m(G_1 \cup G_2)) = m'(m(G_1) \cup m(G_2)) = m'(m(G_1)) \cup m'(m(G_2)) = G_1 \cup G_2$.

Hence m' is the inverse of m and m is **information preserving** for G^*.

□

Note that the possible *conflictuality* of *id* leads to different outcomes than $\texttt{REIF}^{id}$, $\texttt{INTERPRET}^{id}$, $\texttt{TIME}^{id}$ or $\texttt{PROB}^{id}$: the last ones, indeed, would cause the output OWL graph to have some repeated triples, which would lead to obtain a different OWL-Star graph than the original if the reverse mapping m' is applied. However, this would not cause any change to the semantics of the graph. This result is justified by the fact that the vocabulary introduced by OWL-Star is a syntactical transformation. For the *id* function, instead, its *non-conflictuality* is a crucial property for keeping the graph semantics, otherwise different graph nodes may be mistakenly grouped under the same resource.
As a result, while there might be a discrepancy on the possible consequences of *conflictuality* on the graph semantics, the *invertibility* of all the introduced mappings remains crucial.

4.5 Framester Augmentation

Considering Figure 1, we have created a tool for extracting selectional restrictions and a tool for switching from OWL-Star to OWL. After running these tools on the MusicBO dataset, we have obtained OWL ontologies that can be inserted inside **Framester**. Specifically, considering Figure 3.1, we can extract the `pb:sing-01` type constraints, create an OWL ontology and map everything to Framester by creating a domain-specific subframe for `pb:sing-01` (Figure 4.1). Remember that Framester is a hub between language resources: the `skos:closeMatch` predicate maps Framenet *frames* (which are the main backbone of Framester) to *frames* belonging to other resources (like Propbank).

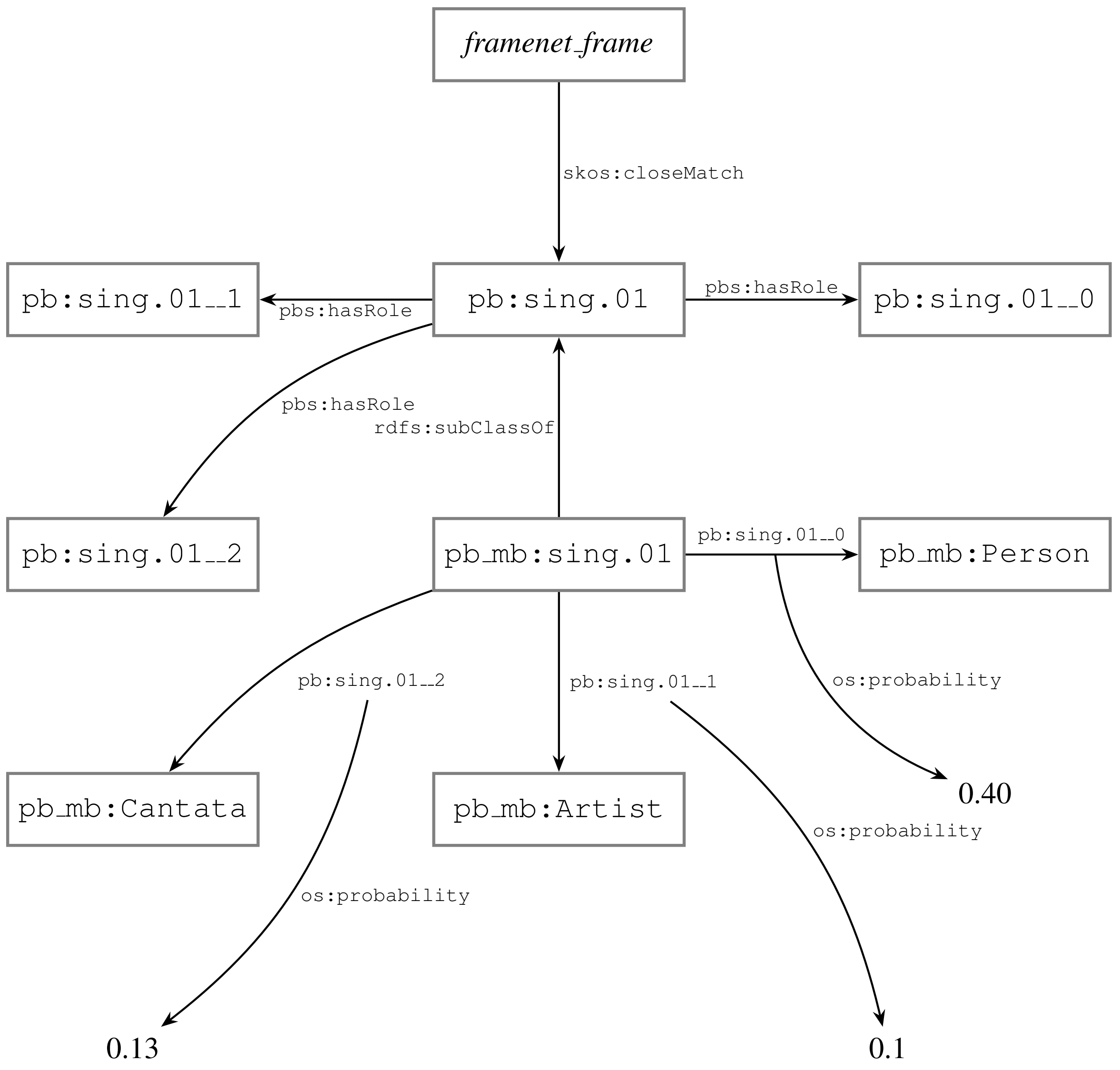


Figure 4.1: Augmented `sing-01` in Framester. The *framenet_frame* box is not a real resource, but is used to represent a set of Framenet *frames* since the mapping can be *many-to-one*. The prefix `pb` stands for *Propbank* and `pb_mb` for newly created MusicBO namespace

Chapter 5

Selectional restrictions for Neuro-Symbolic tasks

Up to this chapter we have described how domain-specific selectional restrictions can be extracted from a corpus of knowledge graphs, creating ontologies in OWL or OWL-Star describing the associated probabilities.
With the recent development of *neuro-symbolic* approaches like **Deep Learning**, the scientific community is trying to understand how neural networks (and machine learning in general) can benefit from the integration of symbolic approaches like knowledge graphs. While both knowledge graphs and relational databases are used to store information, the type of knowledge they can represent is semantically and structurally different. This gap is evident when building neural networks: indeed, most neuro-symbolic approaches rely on tabular data, while training and testing with data stored in triples seems to be an harder task.
Nevertheless, the combination of neural networks and knowledge graphs can be beneficial in several tasks by leveraging the advantages of *representation learning* together with more *factual* data.

This chapter will describe recent popular approaches for Machine Learning on knowledge graphs and how they can be used to integrate the extracted selectional restrictions for a task of *class membership*. Overall, we will show how these additional constraints over role types are beneficial for learning embeddings.

5.1 Machine Learning approaches for KG

In the latest years the scientific community has tried to exploit machine learning approaches for reasoning over KGs. One of the main difficulties is to map existing methods to new structured inputs. Indeed, KGs are structurally different from tabular, textual or image datasets.
The methods shown in literature can be divided in three categories [53]:

- **Tensor Decomposition Models**: these approaches use multi-dimensional matrices to represent knowledge graphs (bilinear or non bilinear models);

- **Geometric Models**: relies on encoding the *relation* as a geometric transformation between *head* and *tail* of the triple. In other words, when the embedding of *head* is summed to the embedding of *relation*, the output should be close to the embedding of *tail*. Thus, the model focuses on minimizing the distance between $E(head)+E(relation)$ and $E(tail)$ with E being the embedding model.
  Geometric models can be:

  - Pure Geometric Models: the basic model belonging to this category is *TransE*, which forces the embeddings to satisfy the vector sum described before. More complex models are *TransH* and *TransR*, different hyperplanes to represent relations and entities;

  - Geometric Models with additional embeddings: the embeddings calculated by *Pure Geometric Models* can be improved with other information sources like text descriptions or constraints. *STransE* is an approach that exploits two additional matrices W_r^h and W_r^t for each $< h,r,t >$ triple. These weights multiply $E(h)$ and $E(t)$ respectively before applying the vector sum, thus allowing to change the embeddings of an entity based on the triple considered. This reduces the disadvantages introduce by *Pure Geometric Models* with reference to *one-to-many*, *many-to-many* or *many-to-one* relations;
  - Roto-Translation models: uses rotation transformations instead of or in addition to translation. *RotatE* belongs to this category and represents the relation r as a rotation between h to t in the complex space;

- **Deep Learning Models**: exploit the recent developments of Deep Neural Networks to learn patterns from the knowledge graphs. By using *neuro-symbolic* models, long-distance nodes are able to influence each other, i.e. they resolve the limitations of distance-based models suffered by previously introduced models. These approaches can be divided in:
  - Graph Convolutional Networks: use convolutional layers to represent embeddings by considering long-distance entities (*ConvE*);
  - Capsule Neural Networks: use convolution-based networks with *capsules* and combine their outputs to form more stable representations (*CapsE*);
  - Recurrent Neural Networks: use RNNs to learn over arbitrary sequences of facts instead of sequences of triples with a fixed length (*RSN*).

Another type of approaches which has not been explicitly stated by [53] is the use of well-known *Natural Language Processing* (NLP) techniques to embed knowledge graph data. Sev-

eral approaches have tried to develop text-to-text models to translate KGs in coherent textual data, most of the times exploiting several linearization techniques to represent KGs in easily parsable syntaxes (e.g. **AMR** [4]). Once the textual data is extracted, standard NLP techniques can be used to obtain the desired outputs.
This two stage approach can have several pitfalls. Some of them are the following:

- the models used in the first stage can add noise and, thus, provide the subsequent step with unreliable data.
- structural information about entities is lost, specifically axioms, hence for reasoners it is more difficult to ascern consistency due to the change of representation.

Some approaches like **RDF2Vec** [52] and **OWL2Vec** [10] do not work by parsing KGs to textual data, but they use language models for graph embeddings by generating sequences of entities: the assumption is that entities that are linked together inside a graph should be spatially closer in the embedding space. In this way, it is possible to apply NLP techniques like **word2vec** [38] or **RNN**s to learn over KGs.

5.1.1 RDF2Vec

The first part of generating graph embeddings with **RDF2Vec** is the generation of sequences of entities. This can be done in several ways, but the approaches proposed by [52] are the following:

- **Random walks**: given a graph $G = (V, E)$ where V is the set of nodes and E is the set of edges, random walks can be generated with a *breadth-first* algorithm, i.e. starting from an input root node r we extract the neighbours of r. After this, the process is recursively repeated for some random sampled neighbours, while the other adjacent nodes are discarded. The recursion stops when the maximum sequence length is reached (hyperparameter of the model) or when the sampled node is terminal.

- **Weisfeiler-Lehman Subtree RDF Graph Kernels** [57]: this technique is usually applied for calculating the similarity of two graphs and, essentially, it is based on the concept of node relabeling given the node's structural context. The idea of applying this technique instead of **random walks** is that, hypothetically, it could be possible to group different nodes together if they are relabeled in the same way, hence being able to interchangeably swap grouped nodes when creating sequences.

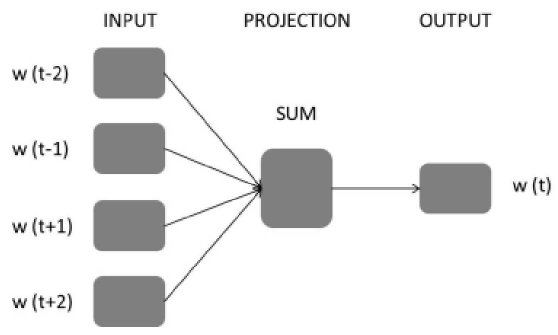
G.Vandewiele et al. [61] have proved that the use of the Weisfeiler-Lehman kernel on RDF graphs **does not provide more insight** on the KG compared to random walks. For this reason, in this paper, when evaluating the use of selectional restrictions on the *class membership* task we only use **random walks**.
After the sequences collection, a self-supervised training pipeline can be used to learn the entities embeddings. RDF2Vec uses **word2vec** which is a two-layer neural network model. There are two different **word2vec** models: the **Continuous Bag of Words** model (CBOW) or the **Skip-Gram** model.

CBOW

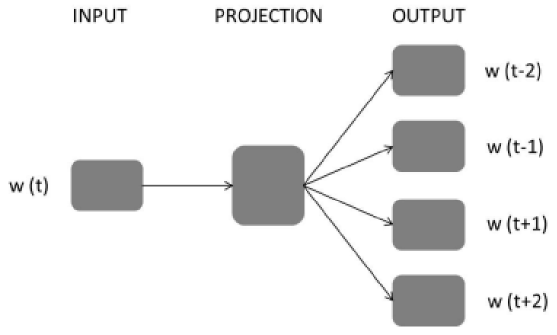
The CBOW model predicts the probability of a certain word given its context words within a given window. In particular, given a set of words $w_1, ..., w_N$ and a positive integer context window c, the objective of CBOW is to maximize

$$\frac{1}{N}\sum_{n=1}^{N} log\ p(w_n \mid w_{n-c}...w_{n+c})$$

where $w_{n-c}...w_{n+c}$ indicates all the words inside window c except w_n and p is the softmax function.



(a) CBOW



(b) Skip-Gram

Figure 5.1: CBOW and Skip-Gram architectured (taken from [52])

The CBOW architecture averages all context words and applies neural layers with softmax to get a score for each word. The architecture is shown in Figure 5.1a.

Skip-Gram

The Skip-Gram model is the inverse of CBOW, i.e. tries to predict the context words given a target word. More specifically, its objective is to maximize

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{-c\leq h\leq c,\ j\neq 0} log\ p(w_{n+j} \mid w_n)$$

The architecture is shown in Figure 5.1b.

5.1.2 OWL2Vec

OWL2Vec is an extension of RDF2Vec which is able to exploit OWL ontologies to generate three documents that capture different aspects of the semantics of the ontology:

- one document for the **graph structure and the logic constructors**;

- one document for the **lexical information**;
- one document for the their **combination**.

These documents are subsequently used for extracting the sequences needed for the learning procedure.
Prior to training, some mappings (also called *projections*) need to be applied to translate OWL ontologies to RDF triples. The mapping strategy used by OWL2Vec is the one proposed by [59, 10] which is based on **approximations**. Specifically, complex relations like *existential*, *universal* or *cardinality restrictions* are parsed into triples without using blank nodes: the idea is that blank nodes may cause noise and make training more difficult since the different entities are not directly linked between each other; at the same time, not the exact logical relations would be kept in the resulting RDF.

Example 5.1.1. Consider the following *existential restriction*

```
:Purchase rdfs:subClassOf
           [ a     owl:Restriction ;
             owl:onProperty :has_seller ;
             owl:someValuesFrom :Company
           ] .
```

The mapping strategy would parse the *existential restriction* to

```
:Purchase :has_seller :Company .
```

Moreover, the proposed mapping strategy creates inverse relations for *membership* and *subsumption predicates* (`rdf:type` and `rdfs:subClassOf`) to create bidirectional walks between two entities. Note that other projection strategies may be applied instead, like the transformation according to the **OWL to RDF Graph Mapping** [40] defined by W3C.

Overall, OWL2Vec applies the same approach adopted by RDF2Vec, but it also introduces additional techniques to map OWL ontologies and additional sequences produced by the *walk generator* algorithm.

Structure Document

The structure document aims to represent both structural and logical constructs by applying a *walk generator* algorithm over the knowledge graph. The ontology axioms are transformed into the OWL Manchester Syntax [29], i.e. sequences are written without including the namespaces for `owl`, `rdf` and `rdfs` relations. Note that the entities IRIs are left unchanged.

Example 5.1.2. Consider Example 5.1.1. The sequence obtained from the *projection* is

(`:Purchase`,`:has_seller`,`:Company`)

Note that it is possible to add another sequence introducing the existential restriction over the initial triple:

(`:Purchase`,`subClassOf`,`:has_seller`,`some`,`:Company`)

where the relations `subClassOf` and `some` follow the OWL Manchester Syntax.

Lexical Document

The lexical document is composed by the sequences extracted in the structure document with the entity IRIs replaced by their `rdfs:label` values. If the entity has no `rdfs:label` relation, then the last part of the IRI is treated as *label* and, assuming it follows the camel-case notation, each different word is added into the sequence.

Example 5.1.3. Consider the following sequence

```
(:COMPANY031,:hasRival,:COMPANY032,:locatedIn,:LOCATION001)
```

where

```
:COMPANY031 rdfs:label "Samsung"^^xsd:String .
:COMPANY032 rdfs:label "Apple"^^xsd:String .
:LOCATION001 rdfs:label "California"^^xsd:String .
```

and the predicates have no labels.
The obtained lexical sequence is

(*"Samsung"*,*"has"*,*"rival"*,*"Apple"*,*"located"*,*"in"*,*"California"*)

Combined Document

To preserve the correlation between IRI entities and lexical information, a combined document is created as a mix of structural and lexical sequences. Specifically, one possible strategy would be to create sequences from the structural document in which a few IRIs remain unchanged, while other entities are replaced with their corresponding labels.
This procedure would benefit the embeddings of the IRIs since they can use the semantics of their labels: this is useful when no label is available and the last part of the IRI is not informative by itself. On the other hand, word embeddings would also benefit from the structural semantics coming from the structural document. Indeed, while the lexical document is more similar to a natural language sentence, the structural document preserves the original triples and relations.

5.2 Preprocessing of MusicBO dataset for OWL2Vec

Showing whether *selectional restrictions* can be useful for *neuro-symbolic* tasks amounts to show empirically if, by using these constraints, the output of machine learning models qualitatively improve for specific tasks. As in most machine learning tasks, a preliminary step of *data preprocessing* must be applied to make the data compatible to the model's architecture. As stated in Chapter 3, this paper focuses on the MusicBO dataset, i.e. a dataset of automatically created KGs from textual description about Bologna's musical history. The dataset is not fully compliant with RDF2Vec and OWL2Vec for the following reasons:

- **.nq to .owl**: the input files are `.nq`, i.e. *N-Quads* files storing information about triples and an IRI labeling what graph in the dataset the triple belongs to (**named graph**). Hence, the files are composed by a set of quadruples. This is not required by OWL2Vec, hence the named graph IRIs need to be deleted to obtain a `.owl` file.

- **Class and Object Properties**: OWL2Vec needs to specifically state which resources are of types `owl:Class` or `owl:ObjectProperty` in order to apply its algorithm. Without these axioms, OWL2Vec would not be able to infer *class membership* and *subsumption* axioms and, thus, it would not be able to create sequences of resources for the *structural document*. Thus, for each frame, for each type of the instance covering frame roles and for each predicate we add *membership relations* to `owl:Class` and `owl:ObjectProperty` respectively.

- **MusicBO IRI reduction**: some of the IRIs contain meta information about the KG they belong to. For example, the IRI associated to the class *Music* in one of the MusicBO knowledge graphs is *https://w3id.org/stlab/mr_data/MusicBO_120_1113_amr/Music*, where the *MusicBO_120_1113_amr* part changes from KG to KG. Since OWL2Vec computes labels and IRI embeddings, without preprocessing it would output different vectors for semantically equivalent concepts due to the difference between the IRIs. To avoid this problem, every IRI mentioning *"MusicBO"* is reduced by deleting the suffix, i.e. for *Music* the final IRI would be *https://w3id.org/stlab/mr_data/Music*.

Apart from the mentioned necessary changes for OWL2Vec compatibility, the dataset can be augmented by:

- **Selectional Restriction Augmentation**: the probability axioms obtained from Chapter 3 can be used to augment the MusicBO KGs. Specifically, it is possible to add relations at the TBox level by connecting *frame* and *type* of the instance covering a frame role through a predicate *role*. While this procedure may not be logically sound since we are using the same *role* IRI for expressing a relation between instances and a relation between classes, it is assumed that it benefits OWL2Vec since we use the same embedding. This procedure is applied **only to the train set** as to not drastically change the KGs used for evaluation and testing.
- **Label Insertion**: every MusicBO KG has no `rdf:label` relation: this is crucial for the creation of the *lexical document* for OWL2Vec. Labels can be automatically added to each entity by extracting the last part of their IRIs, i.e. the string after the last '/' or '#'. This step is **optional**: beware that creating the *lexical document* can improve the sequence sampling, but it is not a necessary step for deploying the OWL2Vec model.

The obtained dataset needs to be split into **train**, **validation** and **test** set. Due to computational reasons, even though the MusicBO dataset is composed by 5000 KGs, we only use 500, 100 and 200 KGs for train, validation and test set.
Since the *class membership* task is a **classification task**, we need to find the targets, i.e. the classes that can be predicted by the OWL2Vec model given the instance and its context. The targets are obtained on the **train set** by extracting each type found for each frame. Due to this,

the evaluation and test set may contain types that do not appear in the train set and hence in the targets. In such cases, the relative instances are not considered for evaluation since it would be **impossible to obtain the correct prediction** since the right type is not within the targets.

5.3 OWL2Vec on domain-specific probabilistic axioms

The Selectional Restriction Augmentation inserts axioms regarding the *reification* of the constraint so to assign a probability for each type. In the *random walk* algorithm it is not possible to access the reified triple (unless it is the root of the walk) due to the fact that its relations are **not bidirectional**. Indeed, from the *frame* or the *type*, the *random walk* algorithm cannot obtain the reified triple since the predicates (`rdf:subject` and `rdf:object`) map the reification to their respective resources (*frame* and *type*) and not viceversa. In other words, the OWL2Vec algorithm cannot access the information about the probability of the *type* covering a *frame role*.

Moreover, obtaining in a single *walk* all the information about the reification is not guaranteed since it would need to access different triples in the same run, i.e. it would need to access the `rdf:subject`, `rdf:predicate`, `rdf:object` and `:probability` relations in order to correlate each part of the statement with the probability. It is safe to assume that, with an infinite number of *walks*, each part of the reified triple is considered and, thus, can learn its embedding with the probability value. Nevertheless, the training procedure would be more complex due to the additional IRIs to be learned and to the not straightforward correlation between the probability and its arguments.

Due to these issues we propose, before the OWL2Vec training phase, to implement a **rule-based** embedding phase to exploit the types extracted by the algorithm in Chapter 3. While some probabilistic patterns can be found through learning by the OWL2Vec model, we believe that the specification of background knowledge via **probabilistic rules** may aid the learning process. Note that there is no technical constraint regarding the dataset used for the generation of probabilistic patterns or the dataset used by OWL2Vec, meaning that they can also be different.

In specific, the proposed rule-based embeddings would be extracted by training a neural network with a margin loss based on the **confidence** of the specific association rule $A \Rightarrow B$ (Section 3.3), i.e. the negative logarithm of the confidence value will be the distance between the *frame* and its possible *argument type*, thus forcing the co-occurrence probability onto the feature space.

The general architecture of the model is shown in Figure 5.2.

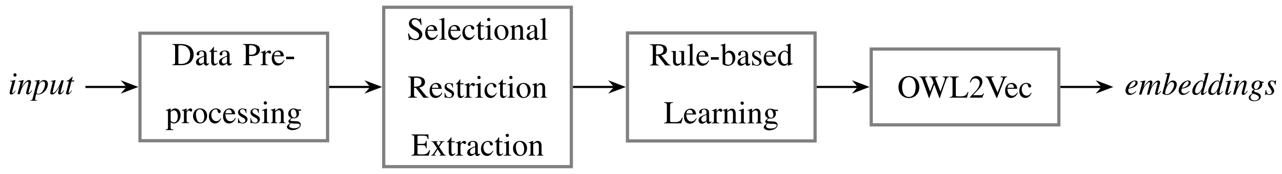


Figure 5.2: Model architecture

Chapter 6

Results

This chapter focuses on the evaluation methodologies applied to the selectional restrictions extraction and the *class membership* task together with their results. A brief section is also dedicated to the OWL-Star to OWL translation, providing examples showing successful and unsuccessful mappings.

6.1 Selectional Restriction Extraction analysis

A qualitative analysis of the obtained selectional restrictions is done with the main focus on how constraints can show and **explain** the input dataset. Table 6.2 shows the most frequent types with their frequency. Such simple analysis can already provide some insights on the topics discusses, since a very high number of types concern the **musical** and **artistic** domain, not considering general and cross-domain types like `NaturalPerson`, `Thing` etc.
The main assumption is that the argument types determine what topics are relevant for the domain.
The general types, together with their original types and frequency, are shown in Table 6.3.

Informative		**Not Informative**	
Frame	**Preference**	**Frame**	**Preference**
`possible-01`	7.109	`embarass-01`	1.922
`contrast-01`	7.031	`precipitate-01`	1.921
`write-01`	6.837	`encapsulate-01`	1.841
`recommend-01`	6.126	`embroider-01`	1.827
`new-01`	5.935	`testify-01`	1.802

Table 6.1: Global Selectional Preference

The generalization of the types can also be useful for providing explanations and the general concepts described inside the domain.

We also perform an analysis based on **selectional preferences**, i.e. the capability of a frame to provide information about its argument types. Table 6.1 shows the most and least informing frames of the MusicBO dataset. The obtained global results for selectional preferences can be useful to provide local interpretations: when we compare the *prior* with the *posterior* distribution assuming a domain-relevant frame like `compose`, `sing` or `record` we can see that types like *Sonatas* and *Song* tend to get higher probabilities. Interestingly, more general frames like `write`, which can be used to describe various different situations (i.e. have an higher selectional preference), in this case tend to foster the probabilities of domain-relevant types, thus showing that this type of analysis can be useful for ontology documentation. Specifically, considering `write-01`, it is possible to say that the domain is more focused on *writing a sonata* instead of *writing to a friend* or *writing a text*, thus showing the fact that the ontology covers the musical domain. This analysis also shows that frame-based knowledge documentation must focus on **informative frames** based on their **selectional preference values**. This follows the idea that drastic changes of distribution for general frames leads to a better analysis of the domain, instead of more specific frames that are always used with a small set of types.

Type	**Frequency**	**Type**	**Frequency**
`NaturalPerson`	2109	`Concerto`	101
`Thing`	641	`Book`	97
`Male`	359	`Publication`	93
`Neuter`	334	`Letter`	93
`Music`	258	`Work-of-art`	90
`Man`	164	`Orchestra`	84
`Opera`	154	`Artist`	83
`City`	147	`Musician`	77
`Female`	107	...	...

Table 6.2: Frequent types in MUSICBO corpus

Generalized Type	**Original Types**	**Frequency**
Abstraction	Interval, Position, Level, Property...	193
Attribute	Position, Level, Property, Accuracy...	122
Communication	Secret, Essay, Overture, Tale...	121
Person	Philosopher, Hack, Hero, Scholar...	106
Auditory_communication	Epigram, Song, Minuet, Quartet...	77
Cognition	Faith, Religion, Culture, Mind...	76
State	Unease, Position, Loneliness, Opportunity...	75
...	...	...

Table 6.3: Frequent generalized types in MUSICBO corpus

Linearization	Answer	Question
write thing written music	music	What is the subject of the thing written?
write thing written music	write	What has been done to the music?
write thing written duet	duet	What is the name of the song that is written in two parts?
write thing written duet	write	What has been done to the duet?
write thing written instrumental	instrumental	What type of music is the thing written?
write thing written instrumental	write	What has been done to the song?

Table 6.4: Competency questions about `write-01`

We apply the pipeline described in Section 3.3.1 to the most relevant frames based on Table 6.1. The results are shown Table 6.4. The questions can trivially be paired with their respective SPARQL query by applying a *selection* with the variables inside the answer column.

6.2 OWL-Star to OWL analysis

The properties of the OWL-Star to OWL mapping have been proven in Chapter 4 and then mapped into corresponding Framester subframes. Here we show examples of successful and unsuccessful applications of the OWL-Star to OWL mapping, together with the resulting subframe created to augment Framester.
Considering the following OWL-Star code,

```
pbdata:style-01 rdf:type owl:Class .
pbdata:Music rdf:type owl:Class .
pbdata:Concerto rdf:type owl:Class .
pbrole:thing_being_styled rdf:type owl:ObjectProperty .
```

```
<< << pbdata:style-01 pbrole:thing_being_styled pbdata:Music >>
   os:interpretation os:AllSomeInterpretation >> os:probability "33.33%" .
<< << pbdata:style-01 pbrole:thing_being_styled pbdata:Concerto >>
   os:interpretation os:AllSomeInterpretation >> os:probability "33.33%" .
```

it can be correctly mapped into OWL:

```
pbdata:style-01 rdf:type owl:Class .
pbdata:Music rdf:type owl:Class .
pbdata:Concerto rdf:type owl:Class .
pbrole:thing_being_styled rdf:type owl:ObjectProperty .

:_0 rdf:type rdf:Statement;
      rdf:subject pbdata:style-01;
      rdf:predicate pbrole:thing_being_styled;
      rdf:object pbdata:Music .

pbdata:style-01 rdfs:subClassOf [a owl:Restriction ;
             owl:onProperty pbrole:thing_being_styled ;
             owl:someValuesFrom pbdata:Music] .

:_0 :probability "33.33%" .

:_1 rdf:type rdf:Statement;
      rdf:subject pbdata:style-01;
      rdf:predicate pbrole:thing_being_styled;
      rdf:object pbdata:Concerto .

pbdata:style-01 rdfs:subClassOf [a owl:Restriction ;
             owl:onProperty pbrole:thing_being_styled ;
             owl:someValuesFrom pbdata:Concerto] .

:_1 :probability "33.33%" .
```

If the *id* mapping is conflictual, then the translation may give the same name to different reification triples, resulting in a semantically ambiguous ontology. In other words, in the previous code, if regardless of `pbdata:Music` and `pbdata:Concerto` the two triples get mapped to `:_0`, the output ontology would be ambiguous.
A similar case could happen if the OWL reification of a OWL-Star triple appears in the OWL-Star code: in that case , if *id* is conflictual, it could introduce new reified triple names that are already appearing in the OWL-Star code, thus obtaining the same ambiguity of the previous case. If the new reified triple has the same semantics of the initial OWL reified triple, then no ambiguity would be introduced but redundancy, since multiple triples would express the same fact. These results follow the theorems proved in Chapter 4.
The result of the translation is inserted inside Framester. An example of a created subframe can be seen in Figure 4.1 with the frame `sing-01`.

6.3 Class Membership analysis

The *class membership* analysis has been applied with three different strategies:

1. **Strategy 1**: train, validation and test set created **randomly** from the MusicBO dataset;
2. **Strategy 2**: train, validation and test set created based on the **source of the sentences** used to generate the graphs, i.e. the sets do not share authors;
3. **Strategy 3**: train and validation set created following the strategy of point 2, while the test set is created from a knowledge graph extracted from **Wikidata** regarding the

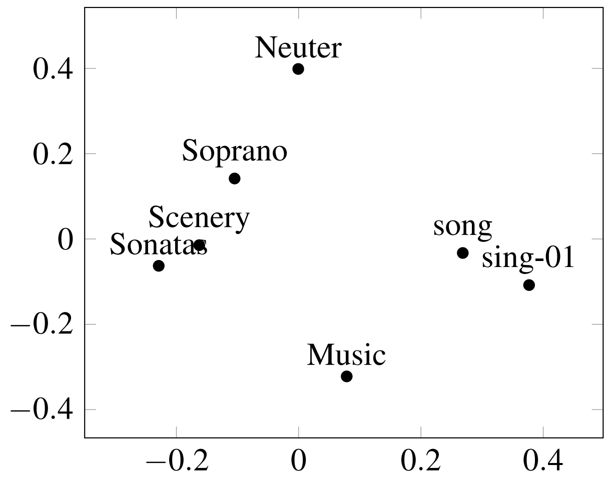


Figure 6.1: Graphical representation of the distance between resources after the rule-based learning pipeline. The selectional restrictions considered are {`sing-01,song`} $\Rightarrow_{0.25}$ {Music}, {`sing-01,song`} $\Rightarrow_{0.02}$ {Sonata}, {`sing-01,song`} $\Rightarrow_{0.02}$ {Scenery}, {`sing-01,song`} $\Rightarrow_{0.02}$ {Soprano}, {`sing-01,song`} $\Rightarrow_{0.02}$ {Neuter}. Resources with capital letters are *types*, lowercase are *roles* and with the `sense-id` are *frames*. The embedding dimensions have been reduced with PCA [35]

musical domain.

Our main concern which led us to follow three different strategies is that the obtained metrics would be too unreliable due to the similarity of the train and test sets. At the same time, the dataset is not a **gold standard** since it has not been manually created and its quality is not assured. Thus, the MusicBO dataset is a **silver standard** and precautionary measures need to be adopted to avoid obtaining very optimistic and biased results.

Each selectional restriction (i.e. each rule with its probability) has been subjected to the **rule-based learning** pipeline explained in Section 5.3. Each resource inside the rules has been mapped to an embedding based on the selectional restriction probability. An example is shown

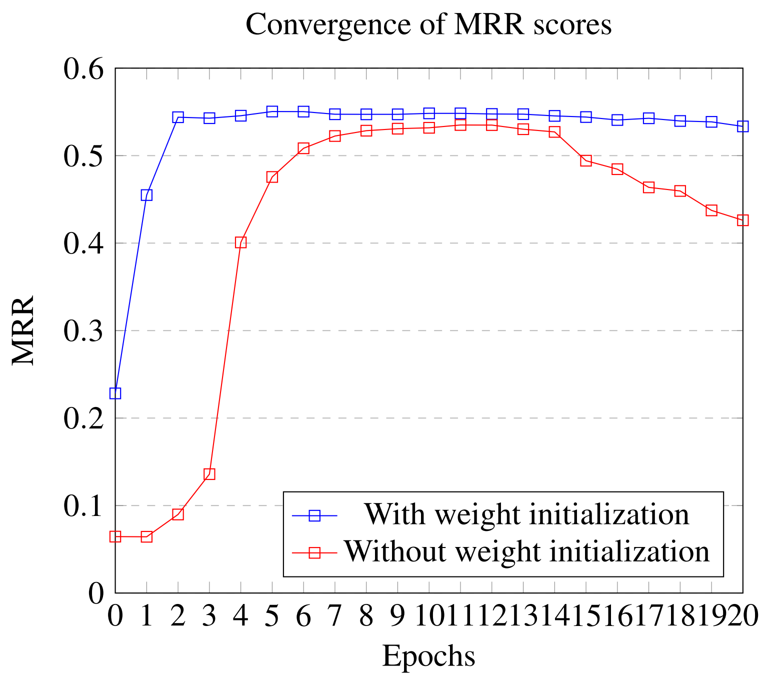
Model Name	**MRR**	**NDCG@1**	**NDCG@2**	**NDCG@5**
Strategy 1 w/o rules	0.5461	0.4427	0.4839	0.5612
Strategy 1 with rules	0.5584	0.4577	0.5025	0.5824
Strategy 2 w/o rules	0.5350	0.4377	0.4772	0.5495
Strategy 2 with rules	0.5504	0.4490	0.4957	0.5751
Strategy 3 w/o rules	0.3651	0.2542	0.3186	0.3874
Strategy 3 with rules	0.3686	0.2601	0.3247	0.3928

Table 6.5: Ranking results

in Figure 6.1: since `Music` has an higher probability to occur with `sing-01` with respect to the other *types*, it is closer to the *frame*. Beware that the graphical representation shown in Figure 6.1 may not be precise due to the dimensionality reduction. Additionally, a lot of types are involved in several *frames* (specially `Music`, as shown in Figure 6.2) so their distances may not be equal to their real confidence with `sing-01`, but may be altered by other typing constraints. Each obtained embedding is subsequently used by the OWL2Vec model. We use ranking metrics to test the model like **MRR** and **NDCG@k** with **k** restraining the metric calculation over the **top k** results. The results are shown in Table 6.5.

The comparison between the usage of the initial rule-based learning highlights the benefits of incorporating previous knowledge into our embeddings. It can be noted that, with an increase of variance between train and test set (i.e. from strategy 1 to 3), the scores tend to decrease, showing that the model performs better on **similar test data**. Overall, the scores obtained from strategy 3 seem to be **more reliable** and provide a greater insight on the model true performance.

The other advantage of using rule-based learning embeddings for **weight initialization** is the improved convergence speed to reach the local minima (Figure 6.3). The main intuition is that, while usually random weights are more robust and avoid the vanishing and exploding gradients, previous knowledge can be applied to start the weights from a better position, thus allowing the model to quickly reach an optimal result. Indeed, this procedure can be thought as a form of transfer learning, even though we are using the same dataset.



Conclusions

In this thesis we have provided several applications of selectional restrictions, highlighting how they can be used in different tasks and case scenarios. We have implemented a selectional restrictions extractor tool to obtain the relevant type patterns for the *frame arguments* of the MusicBO domain and we have provided a program implementing the mapping between OWL-Star to OWL. We have shown that the mapping is *information preserving* if certain conditions hold (Chapter 4). Additionally, we have augmented the Framester hub by defining domain-specific subframes.

We have tested the use of selectional restrictions for neuro-symbolic tasks and shown that, while there is a slight improvement over the model trained without the type constraints, the major gain is achieved with regards to the convergence speed of the model. Moreover, we have seen how selectional restrictions can be used to understand what is the domain represented by an ontology by providing useful statistics on general purpose frames.

Future work can focus on providing a well-defined framework for ontology documentation based on selectional restrictions. The tool could aid ontology engineers in the task of *ontology alignment*, i.e. linking resources between different ontologies. In general, it could be a significant asset to provide insights on knowledge graphs. Another significant development would be to integrate the selectional restrictions or the *class membership* predictions into *Protege*. The plugin could be useful for ontology engineers by suggesting which class to assign to untyped instances. Moreover, new reasoners compliant with the additional type constraints could be developed to provide a better entailment regime for domain-specific ontologies.

Bibliography

[1] Kenneth Antley. "McCawley's Theory of Selectional Restriction". In: *Foundations of Language* 11 (1974), pp. 257–272.

[2] Nicholas Asher. "Selectional restrictions, types and categories". In: *Journal of Applied Logic* 12 (2014), pp. 75–87. DOI: 10.1016/j.jal.2013.08.002.

[3] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. "The Berkeley FrameNet Project". In: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*. 1998, pp. 86–90. DOI: 10.3115/980845.980860.

[4] Laura Banarescu et al. "Abstract Meaning Representation for Sembanking". In: *LAW@ACL*. 2013.

[5] Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. "One SPRING to Rule Them Both: Symmetric AMR Semantic Parsing and Generation without a Complex Pipeline". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35 (2021), pp. 12564–12573. DOI: 10.1609/aaai.v35i14.17489.

[6] Steven Bird and Gary Simons. "Extending Dublin Core Metadata to Support the Description and Discovery of Language Resources". In: *CoRR* cs.CL/0308022 (2003). DOI: 10.48550/ARXIV.CS/0308022.

[7] Stefano Borgo et al. “DOLCE: A Descriptive Ontology for Linguistic and Cognitive Engineering1”. In: *Appl. Ontol.* 17 (2022), pp. 45–69. DOI: 10.3233/AO-210259.

[8] Freddy Brasileiro et al. “Applying a Multi-Level Modeling Theory to Assess Taxonomic Hierarchies in Wikidata”. In: *Proceedings of the 25th International Conference Companion on World Wide Web*. 2016, pp. 975–980. DOI: 10.1145/2872518.2891117.

[9] Valentina Carriero, Paul Groth, and Valentina Presutti. “Towards improving Wikidata reuse with emerging patterns”. In: *CEUR Workshop Proceedings*. 2022.

[10] Jiaoyan Chen et al. “OWL2Vec*: Embedding of OWL Ontologies”. In: *CoRR* abs/2009.14654 (2020). DOI: 10.48550/arXiv.2009.14654.

[11] Emmanuele Chersoni et al. “Modeling Violations of Selectional Restrictions with Distributional Semantics”. In: *Proceedings of the Workshop on Linguistic Complexity and Natural Language Processing*. 2018, pp. 20–29.

[12] Alonzo Church. “Russellian Simple Type Theory”. In: *Proceedings and Addresses of the American Philosophical Association* 47 (1973), pp. 21–33. DOI: 10.2307/3129899.

[13] Andrea Cimmino, Alba Fernandez-Izquierdo, and Raul Garcia-Castro. “Astrea: Automatic Generation of SHACL Shapes from Ontologies”. In: *The Semantic Web*. 2020, pp. 497–513. ISBN: 9783030494612.

[14] Daniela D’Auria et al. “Improving graph embeddings via entity linking: A case study on Italian clinical notes”. In: *Intelligent Systems with Applications* 17 (2023), p. 200161. DOI: 10.1016/j.iswa.2022.200161.

[15] *Defining N-ary Relations on the Semantic Web*. https://www.w3.org/TR/swbp-n-aryRelations/. [Online; accessed 18-February-2023]. 2006.

[16] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018).

[17] Boyang Ding et al. "Improving Knowledge Graph Embedding Using Simple Constraints". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Vol. 1. 2018, pp. 110–121. DOI: 10.18653/v1/P18-1011.

[18] Martin Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. 1996, pp. 226–231.

[19] Daniel Fernandez-Alvarez, Jose Emilio Labra-Gayo, and Daniel Gayo-Avello. "Automatic extraction of shapes using sheXer". In: *Knowledge-Based Systems* 238 (2022), p. 107975. DOI: 10.1016/j.knosys.2021.107975.

[20] G. Fischer, J. Lusiardi, and J. Wolff von Gudenberg. "Abstract Syntax Trees - and their Role in Model Driven Software Development". In: *International Conference on Software Engineering Advances (ICSEA 2007)*. 2007, pp. 38–38. DOI: 10.1109/ICSEA.2007.12.

[21] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN: 0201633612.

[22] Aldo Gangemi. "Ontology Design Patterns for Semantic Web Content". In: *The Semantic Web – ISWC 2005*. 2005, pp. 262–276. ISBN: 978-3-540-32082-1.

[23] Aldo Gangemi et al. "Framester: A Wide Coverage Linguistic Linked Data Hub". In: 2016, pp. 239–254. DOI: 10.1007/978-3-319-49004-5_16.

[24] Aldo Gangemi et al. "Semantic Web Machine Reading with FRED". In: *Semantic Web* 8 (2017), pp. 873–893.

[25] Jian Ge, Luis Guijarro, and Pedro Solorzano. *Riemannian Rigidity of the Parallel Postulate in Total Curvature*. 2016. DOI: `10.48550/ARXIV.1611.07761`.

[26] Ana-Maria Giuglea and Alessandro Moschitti. "Semantic Role Labeling via FrameNet, VerbNet and PropBank". In: *ACL 2006*. 2006, pp. 17–21. DOI: `10.3115/1220175.1220292`.

[27] Paul Graham. *The Revenge of the Nerds*. `http://www.paulgraham.com/icad.html`. [Online; accessed 18-February-2023]. 2002.

[28] Olaf Hartig. "Foundations of RDF and SPARQL (An Alternative Approach to Statement-Level Metadata in RDF)". In: *11th Alberto Mendelzon International Workshop on Foundation of Databases and the Web (AMW)*. 2017.

[29] Petr Homola. "Neo-Davidsonian Semantics in Lexicalized Grammars". In: *International Workshop/Conference on Parsing Technologies*. 2013.

[30] Matthew Horridge et al. "The Manchester OWL syntax". In: *Proc. of the 2006 OWL Experiences and Directions Workshop (OWL-ED2006)*. 2006.

[31] Nitisha Jain et al. "Improving Knowledge Graph Embeddings with Ontological Reasoning". In: *The Semantic Web - ISWC 2021*. 2021, pp. 410–426. DOI: `10.1007/978-3-030-88361-4_24`.

[32] Neema Kotonya et al. "Graph Reasoning with Context-Aware Linearization for Interpretable Fact Extraction and Verification". In: *CoRR* (2021). DOI: `10.48550/arXiv.2109.12349`.

[33] Raymond Kozlowski, Kathleen Mccoy, and K. Vijay-Shanker. "Selectional restrictions in natural language sentence generation". In: (2002).

[34] Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. "A Description Logic Primer". In: *CoRR* abs/1201.4089 (2012). DOI: `10.48550/arXiv.1201.4089`.

[35] Beth Levin. *English Verb Classes and Alternations: A Preliminary Investigation*. The University of Chicago Press, 1993. ISBN: 9780226475332.

[36] Andrzej Mackiewicz and Waldemar Ratajczak. “Principal components analysis (PCA)”. In: *Computers and Geosciences* (1993), pp. 303–342. DOI: 10.1016/0098-3004(93)90090-R.

[37] Antonello Meloni, Diego Reforgiato Recupero, and Aldo Gangemi. “AMR2FRED, A Tool for Translating Abstract Meaning Representation to Motif-Based Linguistic Knowledge Graphs”. In: *The Semantic Web: ESWC 2017 Satellite Events*. 2017, pp. 43–47. ISBN: 978-3-319-70407-4.

[38] Nandana Mihindukulasooriya et al. “RDF Shape Induction Using Knowledge Base Profiling”. In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. 2018, pp. 1952–1959. DOI: 10.1145/3167132.3167341.

[39] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *Proceedings of Workshop at ICLR* 2013 (2013). DOI: 10.48550/arXiv.1301.3781.

[40] Lidiya Murakhovs’ka et al. “MixQG: Neural Question Generation with Mixed Answer Types”. In: *CoRR* (2021). DOI: 10.48550/arXiv.2110.08175.

[41] *OWL 2 Web Ontology Language Mapping to RDF Graphs (Second Edition)*. https://www.w3.org/TR/owl2-mapping-to-rdf/. [Online; accessed 18-February-2023]. 2012.

[42] *OWL Web Ontology Language*. https://www.w3.org/TR/owl-guide/. [Online; accessed 18-February-2023]. 2004.

[43] *OWL-Star*. https://github.com/linkml/owlstar. [Online; accessed 18-February-2023]. 2022.

[44] Martha Palmer, Daniel Gildea, and Paul Kingsbury. "The Proposition Bank: An Annotated Corpus of Semantic Roles". In: *Computational Linguistics Journal* (2005), pp. 71–106. DOI: `10.1162/0891201053630264`.

[45] Alessandro Piscopo and Elena Simperl. "Who Models the World? Collaborative Ontology Creation and User Roles in Wikidata". In: *Proc. ACM Hum.-Comput. Interact.* 2 (2018). DOI: `10.1145/3274410`.

[46] *Polifonia*. `https://polifonia-project.eu/`. [Online; accessed 18-February-2023]. 2021.

[47] Valentina Presutti et al. "EXtreme Design with Content Ontology Design Patterns". In: *Proceedings of the 2009 International Conference on Ontology Patterns - Volume 516*. 2009, pp. 83–97.

[48] Kashif Rabbani, Matteo Lissandrini, and Katja Hose. "SHACL and ShEx in the Wild: A Community Survey on Validating Shapes Generation and Adoption". In: *Companion Proceedings of the Web Conference 2022*. 2022, pp. 260–263. DOI: `10.1145/3487553.3524253`.

[49] *RDF 1.1 Semantics*. `https://www.w3.org/TR/rdf11-mt/`. [Online; accessed 18-February-2023]. 2014.

[50] *RDF Schema 1.1*. `https://www.w3.org/TR/rdf-schema/`. [Online; accessed 18-February-2023]. 2014.

[51] *RDF-star Working Group Charter*. `https://www.w3.org/2022/08/rdf-star-wg-charter/`. [Online; accessed 18-February-2023]. 2022.

[52] Philip Resnik. "Selectional Preference and Sense Disambiguation". In: (2002).

[53] Petar Ristoski et al. "RDF2Vec: RDF graph embeddings and their applications". In: *Semantic Web* 10 (2018), pp. 1–32. DOI: `10.3233/SW-180317`.

[54] Andrea Rossi et al. “Knowledge Graph Embedding for Link Prediction: A Comparative Analysis”. In: *ACM Trans. Knowl. Discov. Data* 15 (2021). DOI: 10.1145/3424672.

[55] Karin Schuler, Anna Korhonen, and Susan Brown. “VerbNet overview, extensions, mappings and applications”. In: *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*. 2009, pp. 13–14. DOI: 10.3115/1620950.1620957.

[56] Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. “BLEURT: Learning Robust Metrics for Text Generation”. In: *CoRR* abs/2004.04696 (2020).

[57] Paula Severi, Jose Fiadeiro, and David Ekserdjian. “Guiding Reification in OWL through Aggregation.” In: *Proceedings of the 23rd International Workshop on Description Logics*. 2010. URL: %5Curl%7Bhttp://ceur-ws.org/Vol-573/paper%5C_19.pdf%7D.

[58] Nino Shervashidze et al. “Weisfeiler-Lehman Graph Kernels”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2539–2561.

[59] Lei Shi and Rada Mihalcea. “Putting Pieces Together: Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing”. In: *Computational Linguistics and Intelligent Text Processing*. 2005, pp. 100–111. ISBN: 978-3-540-30586-6.

[60] Ahmet Soylu et al. “OptiqueVQS: a Visual Query System over Ontologies for Industry”. In: *Semantic Web* 9 (2017). DOI: 10.3233/SW-180293.

[61] Ann Taylor, Mitchell Marcus, and Beatrice Santorini. *The Penn Treebank: An overview*. 2003. DOI: 10.1007/978-94-010-0201-1_1.

[62] Gilles Vandewiele et al. “Walk Extraction Strategies for Node Embeddings with RDF2Vec in Knowledge Graphs”. In: *CoRR* abs/2009.04404 (2020). DOI: 10.48550/arXiv.2009.04404.

[63] Tessa Warren and Kerry McConnell. “Investigating effects of selectional restriction violations and plausibility violation severity on eye-movements in reading”. In: *Psychon Bull Rev* (2007), pp. 770–775. DOI: 10.3758/bf03196835.

[64] Beñat Zapirain, Eneko Agirre, and Lluís Màrquez. “A Preliminary Study on the Robustness and Generalization of Role Sets for Semantic Role Labeling”. In: *Computational Linguistics and Intelligent Text Processing*. 2008, pp. 219–230. DOI: 10.1007/978-3-540-78135-6_19.

Ringraziamenti

Ringrazio Lucia, Giuseppe e Andrea per avermi sostenuto durante questo percorso. I vostri contributi, anche in maniera implicita, mi hanno permesso il raggiungimento di questo traguardo.
Ringrazio Teresa, Michele, Carmela e Innocente per l'esempio che mostrano (hanno mostrato) di perseveranza e affetto.
Ringrazio Sara per l'arduo e sottovalutato compito di sopportarmi e supportarmi.
Ringrazio i miei amici, senza i quali sarei rimasto fin troppo tempo davanti ad uno schermo digitale rovinandomi la vista. I miei occhi vi ringraziano.
Infine, ci tengo a ringraziare la prof.ssa Presutti e il suo team di ricerca per il supporto datomi durante la stesura di questa tesi.