Fuzzy Constraints for Knowledge Graph Embeddings

Michael Weyns [a,*], Pieter Bonte [b], Filip De Turck [b], and Femke Ongenae [b]
[a] *IDLab, Ghent University, Belgium*
E-mail: michael.weyns@ugent.be
[b] *IDLab, Ghent University, Belgium*
E-mails: pieter.bonte@ugent.be, filip.deturck@ugent.be, femke.ongenae@ugent.be

Abstract. Knowledge graph embeddings can be trained to infer which missing facts are likely to be true. For this, false training examples need to be derived from the available set of positive facts, so that the embedding models can learn to recognise the boundary between fact and fiction. Various negative sampling strategies have been proposed to tackle this issue, some of which have tried to make use of axiomatic knowledge claims to minimise the number of nonsensical negative samples being generated. By putting constraints on the construction of each candidate sample, these techniques have tried to maximise the number of true negatives outputted by the procedure. Unfortunately, such strategies rely exclusively on binary interpretations of constraint-based reasoning and have so far also failed to incorporate literal-valued entities into the negative sampling procedure. To alleviate these shortcomings, we propose a negative sampling strategy based on a combination of fuzzy set theory and strict axiomatic semantics, which allow for the incorporation of literal-awareness when determining domain or range membership values. When evaluated on benchmark datasets AIFB and MUTAG, we found that these improvements offered significant performance gains across multiple metrics with respect to state of the art negative sampling techniques.

Keywords: Knowledge graph embeddings, Negative sampling, Type constraints, Fuzzy sets

1. Introduction

In recent years, knowledge graphs (KGs) haven increasingly proven themselves to provide an easy and scalable way of representing and disclosing data extracted from heterogeneous sources [1]. By employing a graphical overview, KGs are able to enrich raw data instances with schematic context information, thus transforming diverse sources into a unified repository of knowledge. The largest and most popular of such repositories is the Semantic Web (SW), a large conglomerate of countless subdomains, each with its own specialised schematic conventions. In contrast with older expert systems, which were unfailingly curated by a central authority, much of the Semantic Web is subject to distributed maintenance and automatic generation. As a result, many of today's graphs are to a large degree incomplete and exhibit high levels of sparsity [2].

*Corresponding author. E-mail: michael.weyns@ugent.be.

Two complementary graph completion approaches can be applied to reduce sparsity. By relying on rule-based deductive entailment, inference engines can be used to infer new facts directly [3]. More flexible and generic are the more recently developed KG embedding techniques, which are able to make inductive predictions about the existential likelihood of specific facts [4]. While rule-based completion relies heavily on the semantic richness of an elaborate schema, statistical embedding approaches typically depend only on the graphical topology to learn the representations of individual entities and relations. Also, while rule-based approaches can be used out-of-the-box, like most statistical techniques, embedding models have to be trained beforehand.

During training, embedding models are primed to distinguish false facts from true ones. Directly relevant to this is the way the SW treats the veracity of any given statement about the world. Specifically, the SW adheres to an open world assumption (OWA), which holds that any uncertainty with respect to the truth of a fact does not imply its falsity. Per this assumption, except where logical contradiction is concerned or negation is explicitly invoked, conclusively negative facts do not exist—and in any case are very uncommon. To supply the training routine with plausible counterfactuals, a negative sampling procedure is required.

While deductive and inductive approaches to KG completion are complementary, they are not often synergised. Indeed, deductive inference can be used to enrich the KG's semantic super-structure (i.e., the schema) and thus expand what we know about the facts in the graph. In this paper, we will be making more explicit use of the semantic enrichment granted to us by rule-based approaches *inside* the negative sampling procedure used by inductive approaches. Many negative sampling strategies have tried to extend the standard approach involving the (random) corruption of positive examples [5]. While certain approaches do exist where axiomatic statements inside the KG are exploited to minimise the number of useless negatives, these approaches are constrained to sharp interpretations of inclusion and exclusion [6, 7]. More specifically, samples are either accepted or rejected on the basis of binary semantic selection criteria, allowing no room for nuance. Furthermore, we will incorporate literal-valued entities into the negative sampling process, which, to our knowledge, has not been attempted before, even though many KGs also include different kinds of literal information [8]. While increasingly such information is incorporated into the modelling schemes of various inductive approaches, it has not yet been integrated into the sampling procedure, thus neglecting what literals have to add when selecting appropriate counterfactuals.

Specifically, the novel contributions of this paper are:

- We propose a negative sampling strategy based on *fuzzy* constraints. This strategy will allow us to exploit schematic knowledge in a non-binary fashion by leveraging semantic nuance to select for more appropriate counterfactuals.
- We offer two different variants of this approach, according to the different interpretations one can attribute to semantic constraints. The *standard-fuzzy* approach assumes an exclusively *closed-world* interpretation, while the *hybrid-fuzzy* approach combines this with an *open-world* interpretation, thus effecting a tradeoff. Because both interpretations have merit in their own right, combining them will allow us to maximally utilise the benefits of fuzzy constraints.
- We enhance the negative sampling approach with literal awareness strategies to better capture literal-based semantics.
- We evaluate our proposed enhancements on two benchmark datasets previously used to evaluate KG embedding techniques [9]. These come supplied with elaborate schemas and plenty of literal-valued information, making them ideal candidates for the purpose of evaluation.

The rest of this paper is structured as follows. In section 2 we present the existing research on negative sampling strategies and in particular we review the most prominent efforts to incorporate schematic information into the sampling process. In section 3, we then define the link prediction problem and provide all of the terminology relevant to our particular methodology. Following this, section 4 contains everything pertaining to the negative sampling strategy based on fuzzy constraints, outlining the entire algorithmic procedure behind the sampling method. Section 5 outlines the evaluation setup and lists all the results for various test settings. Section 6 draws conclusions from the results and makes an overall assessment of the merits of the research. Finally, section 7 concludes with a reflection on what has been accomplished and what will be addressed in future work.

2. Related Work

The most basic form of negative sampling takes a hard-line stance on the closed world assumption (CWA): All triples not observed to be true, are considered false [5]. Because every KG is incomplete to some degree, such an assumption is usually incorrect and therefore ineffective. Better alternatives are to randomly perturb existing triples (by replacing either the head or the tail with another entity) or to assume a *locally* closed world in which any valid triple entails the set of *all* possible triples with the same subject and relationship as the original but with different objects [10]. For the latter option, all triples that cannot be entailed in such a fashion are presumed merely *unknown* (i.e., potentially valid) instead of explicitly false. We notice that this setup always yields true negatives for functional relationships (i.e., relationships mapping subjects onto exactly one object). Both of these alternatives are preferable to the basic CWA because they only generate negative triples that are more likely to be actually false.

Various extensions have been proposed to improve on the baseline perturbation scheme. The first of these was suggested by Bordes et al. when they introduced their TransE embedding model: using so-called *filtered* negative samples [11]. Filtered negative samples are subjected to perturbation as per usual, but are then made to undergo an additional step of being checked against the valid triples in the train set. Should the perturbed triple appear in this set, a new perturbation is generated so as to avoid populating the negative sample set with triples that are actually valid. An early addition to this simple scheme was introduced by Wang et al. for TransH and is sometimes called the Bernoulli trick [12]. The Bernoulli trick involves trying to reduce false negative triples by using different probabilities for the head and tail when performing a perturbation. This discrepancy is based on the mapping property of the relationship (i.e., one-to-one, many-to-one, one-to-many, and many-to-many). In fact, the Bernoulli trick enhances the baseline perturbation scheme with a mapping-sensitive approach already present in the locally closed world assumption (LCWA) mentioned previously. The difference here is that this sensitivity is extended to more complex kinds of relationships than only functional ones. Specifically, for more functional, many-to-one relationships, we would corrupt the tail entity with a higher probability, while for more inversely functional, one-to-many relationships, we would do the opposite.

More recent improvements include the use of fake triples, positive unlabelled learning, adaptive negative sampling, and distributional negative sampling, each of which is discussed below.

Introducing fake triples involves reversing existing relationships to add additional (implicit) facts to the training set in those cases where nodes have either no incoming or no outgoing edges [13]. In other words, when an entity acts either only as a subject or only as an object in all relationships, its connectivity can be enhanced by supplying additional facts that are the inverse of the existing ones. While this might be heuristically beneficial for certain datasets, the assumption strictly holds only for relationships that are in fact specified to be reversible (e.g., instances of *owl:SymmetricProperty*) or are implicitly so. Notably, fake triples improve the sampling process by adding more positive samples to the training set, not more negatives.

Positive unlabelled learning employs a two-stage logistic regression filter to iteratively refine the pool of negative samples [14]. This method assumes a locally closed world to construct two types of unknown triples for each known one—$(*, r_k, e_j)$ and $(e_i, r_k, *)$ for (e_i, r_k, e_j), where * refers to any possible substitute entity for e_i and e_j respectively. For each of these types a given number of samples is generated by way of random sampling. These samples are then combined into an initial unlabelled set, which together with the set of positive facts is fed into the logistic regression filter. Overall, an iterative scheme based on positive and unlabelled learning (PU-learning) is used to generate the final set of negative samples. This process filters out those unlabelled samples of which the regression model is sufficiently certain they should actually belong to the positive set. After a number of iterations, the remaining negative samples are considered reliable enough to be used for training. In short, the approach refines the basic perturbation scheme by removing artificial samples that are too similar to valid ones. By proceeding in this fashion, the technique fails to consider whether the final samples will actually make sense or not.

Adaptive negative sampling makes use of a method like K-Means to adaptively (i.e., iteratively) group entities into clusters based on their current embeddings [15]. Given these clusters, it is able to generate negative triples by selecting the perturbed entity (either the subject or the object, depending on which part of the triple has been corrupted) from the neighbours of the entity that is being replaced. This process is repeated every few epochs, as embeddings are updated over the course of the training procedure.

Finally, distributional negative sampling relies on the stochastic observation that two entities tend to have similar types if they share most of their relations [16]. This property makes them *distributionally similar*. Whereas adaptive negative sampling constructs neighbourhoods as a way of mimicking domains and ranges pertaining to relationships, distributional negative sampling exploits entity embeddings that were constructed by a different knowledge graph embedding model as a baseline repository from which to select a candidate replacement entity for each triple that needs to be corrupted. The replacement entity will be the entity with the greatest similarity to the original entity that also does not result in a known valid triple.

In practice, distributional sampling is very similar to adaptive sampling, the primary difference being the use of external embeddings and forgoing the use of an explicit clustering technique. Because of this similarity, we can subsume both under the more general category of *nearest neighbourhood sampling* techniques, the shared characteristic of which is that they make use of (e.g., pre-trained or adaptively trained) embeddings to select neighbouring entities as replacements [5]. This category of sampling techniques has an unfortunate drawback. While sampling neighbours will often lead to sensical triples, there is no contingency in place to mitigate the possibility that these will themselves be valid facts.

None of the techniques discussed so far have tried to incorporate knowledge that might be derived from the KG's schema. While certainly most of these techniques have the benefit of being relatively lightweight in that they require only the positive sample set to function, schematic knowledge makes explicit many dataset properties that otherwise remain latent. Type information and other axiomatic expressions, such as the domains and ranges belonging to certain relationships, can be exploited to offer additional *a priori* knowledge that can help with generating more appropriate negative samples. In this vein, a few approaches have tried to enhance negative sampling specifically by exploiting such information [5].

While TRESCAL explicitly tries to make use of type constraints, it only explores their applicability to the RESCAL model, so that its overall adoptability remains limited [6]. On the other hand, the work by Toutanova et al. does not consult schema information directly, but instead defines entity types as pairs of sets [17]. The first set in such a pair contains all the relationships for which the given entity has served as a subject, while the second contains those relationships for which the entity has served as an object. This is similar to the locally closed world approach proposed by Krompaß et al. [7]. In this same work, the general approach introduced by TRESCAL is extended to translation-based approaches.

Following the example of TRESCAL, Krompaß et al. make use of type constraints by taking into account *domain* and *range* expressions for various relationships [7]. When generating negative samples using perturbation, it then becomes possible to check whether a new candidate entity actually fits the role suggested by the schema. The candidate's type must correspond to the type restriction expressed by the relationship's domain (or range, depending on which entity we are perturbing). A typed locally closed world approach was also suggested by these same authors in order to cope with situations where no explicit schematic knowledge was available. In this case, as explained in the previous paragraph, artificially constructed domains and range are used to perform the constraint checking when perturbing a triple's head or tail entity.

Overall, the baseline negative sampling strategy making use of Bernoulli-enhanced perturbations has been improved on in various ways. To be concise, these improvements can be subdivided into two major groups: *data-driven* negative sampling and *schema-enhanced* negative sampling. In the first group we find sampling strategies based on PU-learning and nearest neighbourhood sampling, while in the second group we find type-constrained sampling strategies.

Techniques in the second group will try to leverage type information in some way to improve the quality of the generated samples. Whether this type information is inferred stochastically, or is derived directly from the available schema information, the goal is always to use this information to constrain the number of relevant replacement candidates. Nearest neighbourhood sampling strategies do this *positively* by defining alternative *suggestions* whenever an entity is perturbed, while Krompaß et al.'s approach does this *negatively* by filtering the basic replacement set based on relation-specific constraints. In other words, both of these alternatives will construct modified candidate sets, but they do so by making use of complementary kinds of knowledge. We should also note that these existing approaches do not try to accommodate literal values as part of the sampling strategy.

The intuition behind our own approach is synergistic and departs from the schema-aware outline provided by Krompaß et al., augmented with the notion shared by nearest neighbour sampling approaches that there is a degree to which some replacements are more suitable than others [7]. The result of this synthesis is called *fuzzy negative sampling* or *negative sampling based on fuzzy constraints*, which allows us to combine strict (binary) interpretations of constraints with a notion of membership that reflects the degree to which a candidate is able to serve as a suitable replacement. This kind of fuzzy sampling has multiple benefits over a traditional schema-aware approach. First, it leverages the schema to a much greater degree, reflecting the vagueness of semantic appropriateness more accurately. Second, it is able to easily incorporate literal-awareness. Finally, because its logic is based on a synthesis between data-driven and schema-aware approaches, fuzzy negative sampling can be extended to combine multiple negative sampling strategies (data-driven or schema-aware) into a single framework.

3. Problem Description

Because the term "schema" is in itself quite vague, we will introduce a few definitions before moving on. Taking graphs in the Resource Description Framework (RDF) as our template, we will refer to a KG as a tuple $(\mathcal{E}, \mathcal{R})$, where $\mathcal{E} = \{e_1, \dots e_{N_e}\}$ refers to the set of all distinct entities (subjects or objects, depending on the entity's role within a given relationship) in the graph and $\mathcal{R} = \{r_1, \dots r_{N_r}\}$ refers to the set of all dyadic relationships between these entities [10, 18]. Every relationship $r_i \in \mathcal{R}$ is a binary relationship between entities. Therefore, the knowledge graph might be considered a subset of the collection of all possible triples: $(e_i, r_k, e_j) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. However, this basic description, in which all entities are treated equally, forgoes the conceptual differences between different *kinds* of entities expressed in most KGs adhering to RDF, RDF Schema (RDFS), and the Web Ontology Language (OWL). If we distinguish classes $\mathcal{C}$ (entities of type *rdfs:Class* or *owl:Class*, which represent categories of entities) from other entities and also note that relationships can figure as subjects (e.g., when they are defined in a given schema), the definition of a KG can be redefined as a subset of $(e_i, r_k, e_j) \in (\mathcal{E} \cup \mathcal{R} \cup \mathcal{C}) \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{R} \cup \mathcal{C})$. To illustrate this, a few examples of valid triples might be:

- (*ex:Jenny*, *ex:hasBrother*, *ex:Mark*)
- (*ex:Jenny*, *rdf:type*, *ex:Sister*)
- (*ex:hasBrother*, *rdf:type rdf:Property*),
- (*ex:Sister*, *rdf:type*, *rdfs:Class*)

In line with the possibility of identifying entities of various types, the schema allows us to distinguish between valued and non-valued entities. In RDF, each term is either an IRI, a blank node or a literal. The first two categories we will refer to as non-valued entities, while the latter is considered a valued entity. A literal is always associated with a lexical form or value and a data type identifier. If we further distinguish literals $\mathcal{L}$ from other entities, the definition of a KG can be reformulated as a subset of $(e_i, r_k, e_j) \in (\mathcal{E} \cup \mathcal{R} \cup \mathcal{C}) \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{R} \cup \mathcal{C} \cup \mathcal{L})$. An additional example adhering to this extended definition might be (*ex:Jenny*, *ex:age*, *35^^*xsd:int), where 35 is a literal value and xsd:int identifies an integer data type.

Each *possible* triple $x_{ikj} = (e_i, r_k, e_j)$ is associated with a random variable $y_{ikj} \in \{0, 1\}$, for which $y_{ikj} = 1$, if x_{ikj} exists and 0, otherwise. We want to estimate $P(Y)$ with $y_{ikj} \in Y$, so that $Y \subseteq \{0, 1\}^{N_e \times N_r \times N_e}$ (where N_e is the total number of assertional entities and N_r the total number of assertional relations), given a set of observed triples T and a parameter set Θ, i.e., $P(Y|T, \Theta)$ [10]. Here, T is composed of triples where $y_{ikj} = 1$ (i.e., T^+) and false triples fabricated by negative sampling (i.e., T^-). Importantly, any triple (e_i, r_k, e_j) for which either $e_i \in \mathcal{C}$ or $e_j \in \mathcal{C}$, will not be included in the set of positive facts used to train the model, nor will such a triple be used for the purpose of evaluation. Triples belonging to the TBox or domain ontology of the KG will be employed strictly as supplementary knowledge for augmentation. Any fact pertaining to relationships between concepts (terminology, axioms), as opposed to instantiated entities (assertions), will therefore not be considered part of the training set. Essentially,

$$\forall i, k, j, (e_i, r_k, e_j) \in O \implies e_i \notin \mathcal{C}, e_j \notin \mathcal{C} \quad (1)$$

$$\forall (e_i, r_k, e_j) \in O, (e_i, r_k, e_j) \in O^+ \iff y_{ikj} = 1 \quad (2)$$

$$\forall (e_i, r_k, e_j) \in O, (e_i, r_k, e_j) \in O^- \implies y_{ikj} = 0 \quad (3)$$

What we have called negative sampling pertains to the construction of T^-. The set T^- of negative assertions contains statements about the world that are to be considered false. As mentioned in the introduction, we do not usually have such knowledge explicitly—hence the need for negative sampling. When it comes to the status of truth and falsehood, the OWA and the CWA come into play. To reiterate, the Semantic Web assumes an open world view of knowledge. The OWL language guide specifically says this, clarifying that "[new] information can be contradictory, but facts and entailments can only be added, never deleted [19]." OWA here means that facts can be true irrespective of whether they are known to be true. When the truth-value of a given fact is unknown, it cannot be explicitly deemed false, as would happen in a CWA. This corresponds to the notion that knowledge is decentralised, and that a given representation of the facts is always potentially incomplete. Hence the OWA is entirely commensurate with OWL's ontological commitment to positive monotonicity.

The OWA as spelled out above has significant implications for what the schema is able to express. Indeed, OWL is only able to express definitional axioms. Such axioms are used to infer additional schematic information. For instance, suppose we know that a given relationship *rdf:type* is associated with an *rdf:range* axiom that relates it to *rdfs:Class*. When we then encounter a specific use of the relation *rdf:type*, e.g., (*ex:Jenny rdf:type ex:Person*), then we are able to infer, based on the RDFS rules pertaining to domain axioms, that (*ex:Person rdf:type rdfs:Class*) [20]. It is crucial to recognise that OWL is not able to express constraints so much as it is able to posit axiomatic claims. This is completely in line with the OWA, which puts axioms in service of the entailment of additional facts. Axioms express to us something we did not already know. They do not actually restrict what we can express.

So how are we supposed to derive constraints from axioms? To derive integrity constraints from logical axioms, an artificially closed world interpretation must be imposed. To accomplish this, one can first make use of the axiomatic interpretation to expand the original ontology. Deductive reasoning should be considered complementary to statistical relational learning (SRL) for link prediction. Making use of the modelling ontologies[1][2][3], one can compute the deductive closure of any given domain ontology. Once the ontology has been expanded according to the OWA's internal logic, one can impose a restrictive interpretation on each logical axiom within the context of a negative sampling scheme. Indeed, given that it may be assumed that each entity's type declarations have been expanded beforehand according to what is already presupposed to be terminologically valid—according to the KG's TBox or domain ontology—whenever one encounters a triple where the participating entities' types are not axiomatically consistent, one can meaningfully say this triple must be invalid.

At this point we must clarify that while constraints impose a closed world view with respect to OWL's usual axiomatic interpretation, constraints themselves can also be interpreted in two different ways. On the one hand, one can make use of an open world interpretation, where T^- contains only invalid triples (i.e., triples that do not satisfy the constraints), while on the other, a closed world interpretation can be imposed, where T^- contains only valid triples (i.e., triples that do satisfy the constraints). In the prior case, we know that none of the negative examples will ever appear in the test set. Under this interpretation, every negative example is truly false; no possible facts are excluded except when they are nonsensical. It is no coincidence that this interpretation aligns best with the SW's OWA, where falseness is impossible except where nonsense is concerned. In the latter case, we are in fact eliminating useless examples from T^-, on the assumption that nonsensical counterfactuals introduce needless model complexity because they only account for noise. Such facts are not useful for deriving a decision boundary between what exists and what does not. The problem with this interpretation is that it permits T^- to be populated by false negatives. In fact, both interpretations have merit, and it is necessary to decide how they might trade off. In what follows, we will suggest an interpretation of T^- that blends these interpretations in a way that combines the best of both worlds.

[1] http://www.w3.org/1999/02/22-rdf-syntax-ns(rdf)
[2] http://www.w3.org/2000/01/rdf-schema (rdfs)
[3] http://www.w3.org/2002/07/owl (owl)

4. Methodology

In this section, we first specify what we mean by *constraint-based* negative sampling. We then provide a brief overview of the concepts pertaining to fuzzy sets that are also relevant to our framework. We also go over the entire strategy in detail, working out how the various forms of constraints are constructed and validated, and also making sure to integrate the strategy into the overall negative sampling procedure. Finally, we go over the enhancements made to both the embedding procedure and the negative sampling strategy to better exploit statements about literal-valued entities in the link prediction task.

Figure 1 shows an overview of the entire methodology that we propose. In this overview, we distinguish between three kinds of data: *axioms*, *types*, and *train-val-test* triples. The axioms and the types are extracted from an ontology (TBox), while the train-val-test triples are extracted from a corresponding dataset (ABox). Using these three data types, we train a *KG embedding model* using *constraint-based negative sampling*. After the model is trained, a *test ranking procedure* is used to evaluate it.

4.1. Constraint-Based Negative Sampling

As detailed in section 3, constraints must be thought of in the context of an artificially closed world view that offers a restrictive interpretation of axiomatic claims. This restrictive interpretation can itself be treated as either an OWA or a CWA within the negative sampling scheme, i.e., respective to the actions (accept or reject) that must be taken regarding negatives in violation of constraints.

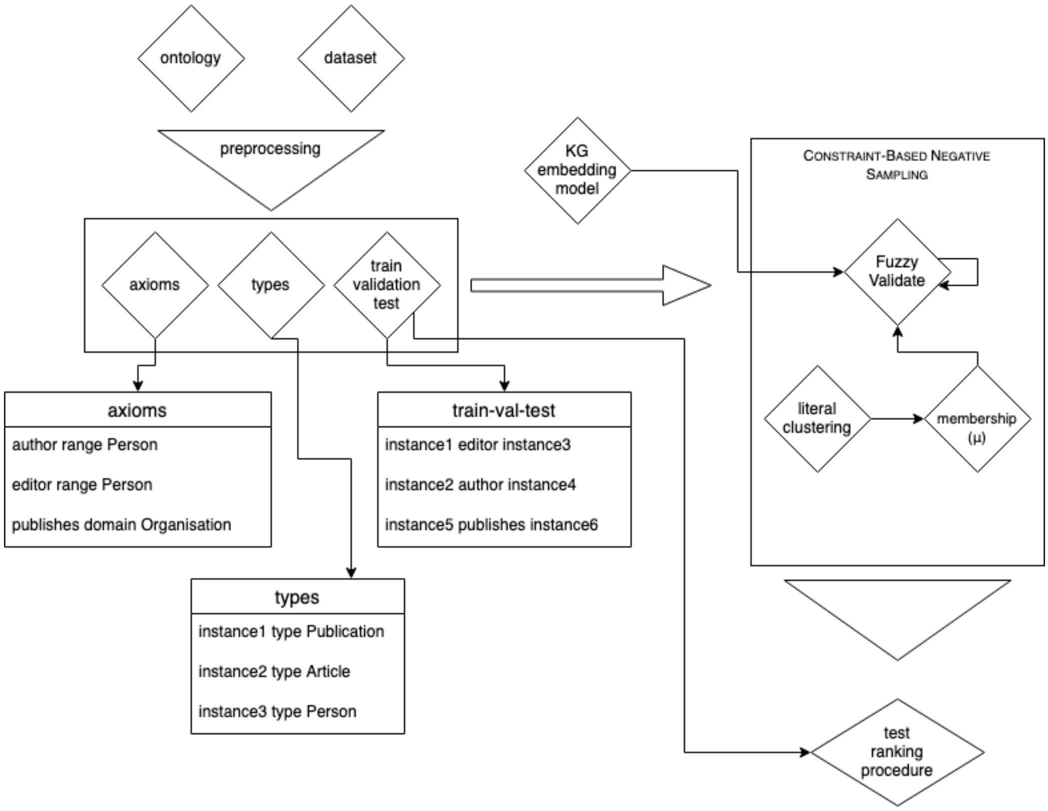

Figure 1. Overview: Fuzzy Negative Sampling

4.1.1. *Integrity Constraints*

Following previous work [21], we define two sorts of constraints: RDFS constraints, which are context-free (i.e., subject and object can be independently validated), and OWL constraints, which are conditional or nested. The RDFS domain and range *constraints* can easily be derived from their open world formulations as follows [22]:

- *rdfs:domain* is an instance of *rdf:Property* that is used to state that any resource that has a given property *must be* an instance of one or more classes.
- *rdfs:range* is an instance of *rdf:Property* that is used to state that the values of a property *must be* instances of one or more classes.

Formally, one can define the domain and range *axioms* as follows:

$$\forall k \in \mathcal{K}, \forall c \in \mathcal{C}, \forall (r_k, \text{rdfs:domain}, c) \in \text{TBox} \implies \forall i, j \in \mathcal{I}, (e_i, r_k, e_j) \implies (e_i, \text{rdf:type}, c) \quad (4)$$

$$\forall k \in \mathcal{K}, \forall c \in \mathcal{C}, \forall (r_k, \text{rdfs:range}, c) \in \text{TBox} \implies \forall i, j \in \mathcal{I}, (e_i, r_k, e_j) \implies (e_j, \text{rdf:type}, c) \quad (5)$$

where $\mathcal{K} = \{1 \ldots N_r\}, \mathcal{I} = \{1 \ldots N_e\}$. The corresponding *integrity constraints* can be derived as follows:

$$\forall k \in \mathcal{K}, \forall c \in \mathcal{C}, \forall (r_k, \text{rdfs:domain}, c) \in \text{TBox} \implies \forall i, j \in \mathcal{I}, (e_i, \text{rdf:type}, c) \implies (e_i, r_k, e_j) \text{ is valid} \quad (6)$$

$$\forall k \in \mathcal{K}, \forall c \in \mathcal{C}, \forall (r_k, \text{rdfs:range}, c) \in \text{TBox} \implies \forall i, j \in \mathcal{I}, (e_j, \text{rdf:type}, c) \implies (e_i, r_k, e_j) \text{ is valid} \quad (7)$$

To illustrate this, suppose we have a relationship *ex:age* with (*ex:age*, *rdfs:domain*, *ex:Person*) and (*ex:age*, *rdfs:range*, *xsd:int*). Given these definitions, a constraint-based interpretation would decide that (*ex:Jenny*, *ex:age*, *35*^^xsd:int) was *valid*, while either (*"Do you know a girl named Jenny?"*^^xsd:string, *ex:age*, *35*^^xsd:int) and (*ex:Jenny*, *ex:age*, *ex:Mark*) would be considered *invalid*.

Drawing inspiration from the Semantic Web's Shapes Constraint Language (SHACL), we note that "[property restrictions] can only be [defined] within the context of an *owl:Restriction*... [where the] *owl:onProperty* element indicates the restricted property [23]." Conditional *constraints* can thus be derived in the following manner:

- The *owl:allValuesFrom* restriction requires that for every instance of the class that has instances of the specified property, the values of the property *must all be* members of the class indicated by the *owl:allValuesFrom* clause [19].
- The *owl:someValuesFrom* restriction describes a class of all individuals for which at least one value of the property concerned *must be* an instance of the class description or a data value in the data range [24].

To clarify, *owl:allValuesFrom* and *owl:someValuesFrom* are *local* to their containing class definitions, meaning that their application is contingent on the subject type. For these restrictions, the *axioms* can formally be defined as follows:

$$\begin{aligned}
&\forall k \in \mathcal{K}, \forall c, c' \in \mathcal{C}, \forall (b(c, r_k), \text{owl:onProperty}, r_k) \in \text{TBox} \;\; \& \\
&\forall (b(c, r_k), \text{owl:allValuesFrom}, c') \in \text{TBox} \\
&\quad \implies \forall i, j \in \mathcal{I}, (e_i, \text{rdf:type}, c) \implies (e_i, r_k, e_j) \\
&\quad \implies (e_j, \text{rdf:type}, c') \qquad (8) \\
&\forall k \in \mathcal{K}, \forall c, c' \in \mathcal{C}, \forall (b(c, r_k), \text{owl:onProperty}, r_k) \in \text{TBox} \;\; \& \\
&\forall (b(c, r_k), \text{owl:someValuesFrom}, c') \in \text{TBox} \\
&\quad \implies \forall i \in \mathcal{I}, \exists j \in \mathcal{I}, (e_i, \text{rdf:type}, c) \\
&\quad \implies (e_i, r_k, e_j) \;\; \& \;\; (e_j, \text{rdf:type}, c') \qquad (9)
\end{aligned}$$

where $b(c, r_k)$ projects a restricted class $c \in \mathcal{C}$ onto the blank node representing its restriction for relation r_k. The corresponding *integrity constraints* are:

$$\begin{aligned}
&\forall k \in \mathcal{K}, \forall c, c' \in \mathcal{C}, \forall (b(c, r_k), \text{owl:onProperty}, r_k) \in \text{TBox} \;\; \& \\
&\forall (b(c, r_k), \text{owl:allValuesFrom}, c') \in \text{TBox} \\
&\quad \implies \forall i, j \in \mathcal{I}, (e_i, \text{rdf:type}, c) \implies (e_j, \text{rdf:type}, c') \\
&\quad \implies (e_i, r_k, e_j) \text{ is valid} \qquad (10) \\
&\forall k \in \mathcal{K}, \forall c, c' \in \mathcal{C}, \forall (b(c, r_k), \text{owl:onProperty}, r_k) \in \text{TBox} \;\; \& \\
&\forall (b(c, r_k), \text{owl:someValuesFrom}, c') \in \text{TBox} \\
&\quad \implies \forall i \in \mathcal{I}, \exists j \in \mathcal{I}, (e_i, r_k, e_j) \;\; \& \;\; (e_j, \text{rdf:type}, c') \\
&\quad \implies (e_i, \text{rdf:type}, c) \implies (e_i, r_k, e_j) \text{ is valid} \qquad (11)
\end{aligned}$$

To illustrate this, suppose we have a relationship *ex:age*, with (*b(ex:Person, ex:age)*, *owl:onProperty*, *ex:age*) and (*b(ex:Person, ex:age)*, *owl:allValuesFrom*, *xsd:int*). Given these definitions, a constraint-based interpretation would decide that (*ex:Jenny*, *ex:age*, *35*^^xsd:int) was *valid*. In this case, while (*"Do you know a girl named Jenny?"*^^xsd:string, *ex:age*, *35*^^xsd:int) would also be valid, (*ex:Jenny*, *ex:age*, *ex:Mark*) would be considered *invalid*.

4.1.2. Validation Based on Integrity Constraints

Now that we have derived context-free and conditional integrity constraints from RDFS and OWL axioms, we can define two variants of constraint-based negative sampling based on a validation procedure that works on a per-triple basis. This will serve as a formalisation of what was previously introduced in [21].

The *VALIDATE* procedure (cfr. algorithm 1) checks a single fact of knowledge (in practice an RDF triple) against all relevant constraints. To do this, it first gathers the types associated with both the subject and object in the triple, and gets the domain and range constraints associated with the predicate. Note that the *GET CONSTRAINTS* sub-procedure mentioned on line 4 also returns an indication of the *kind* of constraints we are able to find. The kind refers to either RDFS constraints or OWL constraints. Once we have acquired this information, we can proceed with the actual validation. RDFS domain and range constraints are verified in sequence. For each, we check whether all the types associated with the constraint also appear in the type collection associated with the subject and object respectively. This means both of these constraints are bound to a conjunctive interpretation. As we have already mentioned before, OWL constraints are beholden to a nested interpretation. If we know that any of the domain types corresponds with a subject type, then we can proceed to verify the range in the same fashion as before. Note that the *owl:someValuesFrom* constraints are not being validated because they require a global view of the training samples and cannot be evaluated straightforwardly on a per-triple basis.

Based on the *VALIDATE* procedure, we can now formulate two variants of the constraint-based negative sampling procedure. Algorithm 2 shows that for each batch in the original training set, we consider each separate triple and, by relying on the aforementioned

```
1: procedure VALIDATE(subject, predicate, object)
2:     subject_types ← get_types(subject)
3:     object_types ← get_types(object)
4:     domain, range, kind ← GET CONSTRAINTS(predicate)
5:     if kind = RDFS then
6:         if not all([t in subject_types for t in domain]) then
7:             return False
8:         end if
9:         if not all([t in object_types for t in range]) then
10:            return False
11:        end if
12:        return True
13:    else                                        ▷ in case of owl:allValuesFrom
14:        if all([t in subject_types for t in domain]) then
15:            if any([t in object_types for t in range]) then
16:                return True
17:            end if
18:            return False
19:        end if
20:    end if
21:    return True
22: end procedure
```

algorithm 1: Validate a single RDF triple

Bernoulli sampling, corrupt either the subject or object entity. Considering the case of subject corruption, we select a random subject from the total set of entities as a candidate substitute. If the new triple is validated correctly according to the *VALIDATE* procedure outlined earlier, and the resultant triple does not already belong to the original train set, then we accept the new triple as part of the negative sample set. The only difference between the two variants concerns the acceptance criterion. Namely, for the CWA procedure, we accept the corrupted triple when it can be validated, while for the OWA we do the opposite, and only accept the candidate substitute when the resulting triple actually violates the constraints. At this point, algorithm 2 can facilitate both interpretations of T^{-} mentioned at the end of section 3.

4.2. Fuzzy Sets

So far, we have illustrated how schematic knowledge can be used to refine which triples we accept or reject during negative sampling. This kind of constraint-based approach imposes a strict, all-or-nothing interpretation on the validation procedure. Candidates are either valid or they are not. Construing things this way ignores the fact that while two candidates can both be valid in a strict sense, one may still be more appropriate as a replacement than the other. We can better capture this behaviour using fuzzy sets to model domains and ranges.

We must first give a brief overview of the relevant concepts belonging to the domain of fuzzy set theory and how they might be applied to give us an alternative, more generic interpretation of constraints. A fuzzy set generalises the concept of a regular set by allowing for non-binary membership functions. Formally, a fuzzy set F defined over a universe U subsuming all possible elements, can be defined according to the membership function $\mu_F(x) : x \in U \rightarrow [0, 1]$, which expresses the degree to which any element x that is a part of the universe U is said to belong to the fuzzy set F. Here, we agree that if $\mu_F(x) = 0$, x does not belong to F *at all*, and if $\mu_F(x) = 1$, x belongs to F *completely*. If $\mu_F(x) =]0, 1[$, then we say that x belongs to F only *to a degree*.

Fuzzy sets admit various interpretations. A given set's membership function can be understood to express either a degree of *correspondence* or a degree of *uncertainty*. While the first option is said to offer a *conjunctive* interpretation, the last option adheres to a *disjunctive* interpretation. Correspondence refers simply to the degree to which a given element approximates the fuzzy concept represented by the set. As such, the

```
procedure CWA CONSTRAINT-BASED NEGATIVE SAMPLING(train_set, all_entities, neg_ratio)
    new_train_set ← ∅
    for batch in train_set do
        neg_batch ← ∅
        for (s, p, o) in batch do
            for i ← 0, neg_ratio do
                pr ← RANDOM()
                corrupted_triple ← (s, p, o)
                if pr > BERNOULLI(p) then                          ▷ Use Bernoulli sampling
                    do
                        s′ ← select(all_entities)
                        valid ← VALIDATE(s′, p, o)
                    while not valid or (s′, p, o) in train_set      ▷ valid for OWA
                    corrupted_triple ← (s′, p, o)
                else
                    do
                        o′ ← select(all_entities)
                        valid ← VALIDATE(s, p, o′)
                    while not valid or (s, p, o′) in train_set      ▷ valid for OWA
                    corrupted_triple ← (s, p, o′)
                end if
                APPEND(neg_batch, corrupted_triple)
            end for
        end for
        new_batch ← CONCAT(batch, neg_batch)
        APPEND(new_train_set, new_batch)
    end for
    return new_train_set
end procedure
```

algorithm 2: CWA Constraint-Based Negative Sampling

conjunctive sense of the concept modelled by the set is wholly determined by the collective of elements subsumed by it and their corresponding membership values. Uncertainty on the other hand arises within the context of possibility theory, to determine the possibility that a certain parameter is to assume a certain value.

When modelling fuzzy constraints, we want to give a fuzzy interpretation to what it means to belong either to the domain or range of a given relationship. The interpretation we choose for this purpose follows the conjunctive sense of membership, as this most closely resembles the way domains and ranges are determined semantically by the elements composing them.

We now define $S = \{e_i | (e_i, r_k, e_j) \in T\}$ and $O = \{e_j | (e_i, r_k, e_j) \in T\}$. For the purposes of fuzzy constraint evaluation, we might choose to model every predicate as a fuzzy relation $R : S \rightarrow O \subset S \times O$, where every element $(s, o) \in S \times O$ is associated with a certain membership score. In other words, $\mu_R((s, o)) \in [0, 1]$.

```
procedure FUZZY VALIDATE 1(s, p, o, position)
    pr ← RANDOM(0, 1)     ▷ Draw floating point number from Uniform(0, 1)
    if position = head and μ_S^p(s) ⩾ pr then
        return True
    end if
    if position ≠ head and μ_O^p(o) ⩾ pr then
        return True
    end if
    return False
end procedure
```

algorithm 3: Fuzzy Validate 1

We can derive this membership function from the respective memberships μ_S^R and μ_O^R by means of an aggregation operator. Because algorithm 2 specifies that we only need to validate perturbations, what this boils down to in practice is having to calculate the member-

```
procedure FUZZY VALIDATE 2(s, p, o, position)
    λ_S ← AVG(μ_S^p))          ▷ Calculate average membership degree for subjects
    λ_O ← AVG(μ_O^p))          ▷ Calculate average membership degree for objects
    if position = head and μ_S^p(s) ⩾ λ_S then
        return True
    end if
    if position ≠ head and μ_O^p(o) ⩾ λ_O then
        return True
    end if
    return False
end procedure
```

algorithm 4: Fuzzy Validate 2

ships for individual subjects or objects serving as potential replacements. This allows us to use $\mu_S^R(s) \in [0,1]$ and $\mu_O^R(o) \in [0,1]$ directly. Algorithms 3 and 4 demonstrate how we can use these membership functions in two alternative approaches to fuzzy validation. These algorithms can be considered the fuzzy alternatives to algorithm 1.

In algorithm 3, we generate a random number $pr \in [0,1]$ that we use to determine whether or not we accept the candidate by comparing it with the membership score. This means we will accept the candidate with a probability equal to this corresponding membership score. An alternative approach is presented in algorithm 4, where instead of generating a random number, we use the λ score as a cutoff value to decide whether or not we will accept the candidate triple. Option two is based on a standard way of turning a fuzzy set into a *crisp* set based on the lambda-cut method. The lambda-cut set of a given fuzzy set F is defined as $F_\lambda = \{x|\mu_F(x) \geqslant \lambda\}$ with $0 \leqslant \lambda \leqslant 1$, which indeed corresponds with the procedure followed in algorithm 4. While this approach is a good candidate for what we are trying to achieve, algorithm 3 has a few benefits that outclass what is offered by the other option. Namely, the lambda-cut set used in algorithm 4 imposes a very binary selection criterion that does not take into account the membership values to a very significant degree. All values below the average are thrown away, and all values above this average are selected with equal probability. Conversely, the alternative approach presented in algorithm 3 allows us to select all candidates modulated exactly to their degree of correspondence, thus introducing a level of diversity that the other approach clearly lacks. One obvious problem with the probability-based approach is that all candidates for a given predicate might correspond with very low membership values, so that it becomes very difficult to find suitable replacements. This is easily solved by normalising the memberships against the maximum across all possible candidates. In closing, we note how algorithm 3 establishes a natural continuum between the CWA and OWA interpretations of constraint-based negative sampling, allowing very unlikely candidates that will probably produce nonsensical negative examples to be selected at a reduced rate. Depending on whether we wish to favour the CWA or the OWA approach, we can invert the selection criterion to favour either high or low memberships.

4.3. Negative Sampling Based on Fuzzy Constraints

Now that we have defined a constraint validation procedure based on the concepts of fuzzy set theory, we still need to determine how μ_S^R and μ_O^R are to be specified exactly. Recent work by Chen et al. [25] has attempted to tackle the problem of knowledge base correction by leveraging various complementary approaches, such as lexical matching, KG embeddings, and semantic constraint matching. While the link prediction problem tackled in our own work belongs to a different species of quality improvement pertaining to *completion*, the aforementioned work is specifically concerned with the *correction* of facts with object or literal entities that need to be replaced.

Their solution involves a pipeline of steps used to identify appropriate candidate substitutes for a given erroneous statement. First, a batch of candidate entities is selected by evaluating semantic relatedness with the aforementioned lexical matching techniques. Next, a subgraph is extracted from the overall KG to represent the semantic context of the candidates. Based on this subgraph a link prediction model is trained to predict the likelihood that each candidate assertions is existentially valid. The remaining assertions are finally checked against a number of property range and cardinality constraints. Within the context of our fuzzy negative sampling scheme, we suggest integrating the constraints used in the final steps of this approach with the constraint checking procedures introduced in algorithm 1 in order to generate useful definitions for μ_S^R and μ_O^R [25]. The final procedure for μ_S^R is listed as algorithm 5; the procedure for μ_O^R runs along very similar lines [25].

4.3.1. Fuzzy Membership

To calculate the domain membership score μ_S^R of a given entity e for a given predicate p, we must calculate both the cardinality score and the constraint score relating the entity to the domain of the predicate. Succinctly put, the cardinality of a given predicate is represented as a probability distribution $car_p(k) \in [0, 1]$, which maps a number of subjects onto the fraction of objects related to that number of subjects via the given predicate. For instance, if a given predicate *hasChildren* associates parents with children, and nine children have two parents, while only one child has one parent, then $car_p(k = 1) = \frac{1}{10}$ and $car_p(k = 2) = \frac{9}{10}$. Given this measure of soft cardinality, we can compute a cardinality score, indicating the degree to which we can be confident the property is either an inverse functional property (functional property for μ_O^R) or a non-functional property. However, to do this we must deviate from the procedure as it was originally set up. To compute n (cfr. line 5) exactly we must count the number of subjects associated with any given predicate-object pair. As a result, each score $\mu_S^R(e)$ is predicated on a specific object e'. Computationally, this is undesirable as it denies us the possibility to calculate the membership score of a given subject based on the characteristics of that subject alone. While it would be possible to avoid the cardinality score altogether, a much better option is to use a summary statistic as an approximation. If we define $n_{p,e'} = |subjects_{p,e'} \cup \{e\}|$, where $subjects_{p,e'} = \{s | (s, p, e') \in train\_set\}$ for a given subject e, predicate p, and object e', we calculate n for e as the average over all $n_{p,e'}$.

Functionally, this allows us to compute membership scores in the context of evaluating candidate triples wherein either the subject or the object has been corrupted (as in algorithms 3 and 4). In case the predicate is not (inversely) functional ($n \neq 1$), we can use the exceeding rate to progressively degrade the cardinality score. After all, the exceeding rate r expresses how much we are allowed to be in violation of the verified maximal subjective association. (SN_{max} is the maximum number of subjects associated with a given object for the given relation).

Having computed the cardinality score, the next step in the process involves computing the constraint scores. On line 21, $SD(c)$ is defined as a function mapping each class to its corresponding *supporting degree*, which expresses the degree to which the class is supported by the given relationship domain. We calculate this by getting the ratio between the number of subjects belonging to the given class and the total number of subjects associated with the relationship. Next, $SC(p)$ allows us to map each predicate (relationship) onto the classes associated with its subjects. Within this set of classes we can distinguish between generic classes and specific classes, and indeed on the basis of this distinction we will define two separate constraint scores (con_{sp} and con_{ge}). First we identify RDF top-level classes as <http://www.w3.org/2000/01/rdf-schema#Resource> and <http://www.w3.org/2002/07/owl#Thing>. All classes participating as objects in a <http://www.w3.org/2000/01/rdf-schema#subClassOf> relationship, we refer to as *generic*, while all other classes (besides the top-level ones) will be called *specific*. This means that specific classes express the most fine-grained type information we have about a given entity, while generic classes express hierarchical abstractions. Besides the top-level classes we also ignore *subClassOf* relationships that express either identity (e.g., (*ex:Person*, *rdfs:subClassOf*, *ex:Person*)) or nullity (e.g., (*owl:Thing*, *rdfs:subClassOf*, *owl:Nothing*)). The constraint scores are then computed (cfr. lines 23 and 24) simply by taking the product of each class' supporting degree, over the set of all relevant subject classes (including those belonging to e). The final membership score is a weighted sum of the cardinality score and the two constraint scores, where more weight is given to constraints than cardinality and specific constraints are valued more than generic ones.

4.3.2. Standard and Hybrid Alternatives

At this point, we should note that because algorithm 5 computes a *replacement* score, it becomes very likely that the negative samples generated during the execution of algorithm 2 will in fact be *valid* test triples. These valid test triples belong to the set of false negatives that are inevitably generated during the sampling procedure. The key to a good negative sampling strategy is to generate negative examples that will allow any given embedding model to effectively distinguish truth from falsity, or in other words, to generate false statements that make sense. However, it also means that we must avoid generating negatives that are actually true statements, a danger that increases precisely when our artificial samples become more sensible. Two alternatives exist to solve this issue. On the one hand, we can use an OWA interpretation of algorithm 2. Under this interpretation, the highly appropriate triples positively validated by algorithm 3 are rejected in lines 13 and 19 of algorithm 2. In other words, the triples most likely to

procedure μ_S^R(p, e, train_set)
  $car \leftarrow 0.0$ ▷ Cardinality score
  $subjects_p \leftarrow \{s|(s,p,o) \in train\_set\}$ ▷ Subjects associated with predicate p
  $objects_p \leftarrow \{o|(s,p,o) \in train\_set\}$ ▷ Objects associated with predicate p
  $n \leftarrow \frac{\sum_{o \in objects_p} |\{s|(s,p,o) \in train\_set\} \cup \{e\}|}{|objects_p|}$
  $SN(o) \leftarrow |\{s|(s,p,o) \in train\_set\}|$ ▷ #subjects associated with a given object o, for p
  $SN_{max} \leftarrow \mathrm{MAX}(\{SN(o)|o \in objects_p\})$ ▷ Max SN, across all objects associated with p
  $car_p(k) \leftarrow \frac{|\{o \in objects_p | SN(o)=k\}|}{|objects_p|}$, for $k \in [1, SN_{max}]$ ▷ Cardinality for given SN
  $car_p(k > a) \leftarrow \sum_{i=a+1}^{SN_{max}} car_p(k = i)$ ▷ Cumulative cardinality
  $r \leftarrow \frac{n - SN_{max}}{SN_{max}}$ ▷ Exceeding rate
  if $SN_{max} == 0$ or $r \geqslant 0.8$ **then**
    if $n == 1$ **then** ▷ Inverse functional predicate
      $car \leftarrow car_p(k = 1)$
    else ▷ Non-functional predicate
      $car \leftarrow car_p(k > 1)$
      if $r > 0$ **then**
        $car \leftarrow car \cdot (1 - r)$
      end if
    end if
  end if
  $SD(c) \leftarrow \frac{|\{s | s \in subjects_p, c \in C(s)\}|}{|subjects_p|}$ ▷ Supporting degrees for given class c
  $SC(p) \leftarrow \{c | s \in subjects_p, c \in C(s)\}$ ▷ Subject classes associated with p
  $con_{sp} \leftarrow 1 - \prod_{SC_{sp}(p) \cap C(e)} (1 - SD_{sp}(c))$ ▷ For specific subject classes
  $con_{ge} \leftarrow 1 - \prod_{SC_{ge}(p) \cap C(e)} (1 - SD_{ge}(c))$ ▷ For generic subject classes
  return $0.2 \cdot car + 0.8 \cdot (0.2 \cdot con_{ge} + 0.8 \cdot con_{sp})$
end procedure

algorithm 5: membership function μ_S^R

correspond to successfully *corrected* facts are rejected most often as negative samples.

A more sophisticated approach involves a two-step procedure that combines the advantages of both strict and fuzzy semantics. This means using strict semantics as a first pass, to reject negative samples that are blatantly nonsensical. If the triples pass this test, then their appropriateness can be gauged further by resorting to an OWA interpretation of algorithm 3 (by inverting the $\geqslant$ sign on lines 4 and 9), where a triple with a high score is rejected more often than one with a low score. This allows only schematically correct, but probably inappropriate candidates to be selected as valid perturbations inside the negative sampling scheme.

So, in the first case, we use a purely OWA interpretation of fuzzy negative sampling, while in the second case, we use a CWA interpretation of strict, constraint-based negative sampling, combined with an OWA interpretation of fuzzy triple validation. We will use the first case as our *standard-fuzzy* approach, and the second case we will refer to as our *hybrid-fuzzy* approach.

4.4. Literal-Enhanced KG Embeddings & Negative Sampling

Now that we have discussed all of the principal components in the negative sampling procedure, we can move on to incorporate literal-valued entities into this procedure. To accomplish this, we follow a two-pronged approach: On the one hand, we will make use of enhancements to the embedding technique itself inspired by the work of Kristiadi et al. on LiteralE [26]. On the other hand, we will enhance the membership score presented in subsection 4.3 to take literal values into account based on a clustering mechanism. This way, we can evaluate the effect of integrating literals into the negative sampling procedure in two different ways.

```
1:  procedure LITERAL CLUSTERING(p, train_set)
2:      literals ← {o|(s, p, o) ∈ train_set, o is literal}
3:      if LENGTH(literals) > 0 then
4:          embeddings ← GET EMBEDDINGS(literals)
5:          centroid ← MEAN(embeddings)
6:          radius ← MAX({COSINE DISTANCE(centroid, e)|e ∈ embeddings})
7:          cluster_p ← (centroid, radius)
8:          return cluster_p
9:      end if
10: end procedure
```

algorithm 6: literal clustering

4.4.1. *Literal Clustering*

We start with the second part of our approach. Algorithm 6 demonstrates how we are able to generate a literal cluster per predicate p. Normally, one might expect a clustering procedure to rely on an unsupervised technique such as k-nearest neighbours. However, in our case, we already know which elements belong to each cluster and how many clusters there are, since we can group literals values according to the predicates that feature them as objects. What we need to do is calculate the characteristics of each literal cluster so that we can check whether previously unseen literals are likely to belong to a given cluster or not. On line 2, we start by identifying all of the literal objects belonging to a given predicate (which is then called a data property). For all of these literals we then retrieve their embedded representations (line 4) and calculate the centroid of the cluster by taking the mean across these embeddings. The way these embeddings are constructed will be explained further on. Each embedding here is a vector of size d, so that the centroid is also a vector of d, where a given dimension i is calculated as $\frac{\sum_{j=1}^{n} v_j[i]}{n}$ for n literal embeddings $v_j, j = 1 \dots n$. The radius then is defined as the greatest distance between the centroid and a given embedding inside the cluster. Using only the centroid and the radius we can then determine the likelihood that a given literal belongs to the cluster. We do this by calculating an additional literal score as follows:

$$score_{lit} = \max(0.0, \frac{1}{a} \cdot (a - (\frac{\cos_{dist}(centroid, v_{lit})}{radius}))) \tag{12}$$

This score is calculated by taking the ratio between the cosine distance $cos_{dist}(v, v') = 1 - cos_{sim}(v, v') = 1 - (\frac{\sum_{i=1}^{d} v_i v'_i}{\sqrt{\sum_{i=1}^{d} v_i^2}\sqrt{\sum_{i=1}^{d} v_i'^2}})$ between the cluster centroid and the embedding of the literal in question, and the radius of the cluster. The larger the numerator, the larger the ratio becomes, and the smaller the score becomes. When the numerator exceeds the denominator by more than $a - 1.0$ (i.e., if the ratio exceeds the cutoff value $a > 1.0$), we set the score to zero. We divide by a to ensure that the score does not exceed 1.0. Of course, this is not the only possible formulation of the literal score. An alternative definition, which obviates the need for a cutoff value, could be the following:

$$score_{lit} = e^{\frac{-\cos_{dist}(centroid, v_{lit})}{radius}} \tag{13}$$

By incorporating the literal score, we calculate the final score as follows:

$$\begin{aligned} score_{final} = \\ 0.2 \cdot car + 0.8 \cdot (\quad & 0.5 \cdot (0.2 \cdot con_{ge} + 0.8 \cdot con_{sp}) \\ & +0.5 \cdot score_{lit}) \end{aligned} \tag{14}$$

Here, we follow the weighting scheme suggested by Chen et al. when balancing out the influence of con_{ge} and con_{sp} [25]. We extend this same distribution of weights when taking the average of the cardinality and constraint scores. The literal score and the constraint score are themselves also combined using an unweighted average.

To get the literal embeddings required for algorithm 6, we rely on domain-specific embedding techniques. Specifically for our purposes, we distinguish between numerical and textual literals. To embed numbers we simply use a single fully connected layer with an input dimension of 1 and an output dimension of d. The resulting embeddings are normalised per feature dimension. To embed textual data, we make use of Doc2Vec with vector size d and window size 5 [27]. Each textual literal is treated as a tagged document inside the Doc2Vec model. The model is trained for 20 epochs, and at the start of each new epoch the corpus of documents is reshuffled. To identify which entities qualify as numbers or text, we rely on the schema. http://www.w3.org/2001/XMLSchema defines a number of data types for RDF graphs. We can say, provisionally, that a literal is a number if it has at least one of the following data types: *decimal, integer, double, float, short, int, long*. A literal is said to express textual information if it has data type *string*.

4.4.2. Enhancing KG Embeddings with Literals

Now that we know how literal embeddings are created, we can specify how we want to make use of LiteralE. LiteralE is an enhancement technique, aimed at improving any sort of direct encoding technique with supplementary literal information. Concretely, LiteralE uses a gating mechanism—specifically a gated recurrent unit (GRU)—to learn whether incorporating literal information is useful or not. For a single type of literal information, given an input vector x the activation for this unit might be formulated as follows:

$$h^{\langle t\rangle} = u \odot h^{\langle t-1\rangle} + (1-u) \odot \tilde{h}^{\langle t\rangle} \tag{15}$$

Here, $h^{\langle t\rangle}$ is the hidden unit at time step t, u is the *update gate*, $\tilde{h}^{\langle t\rangle}$ is the new hidden state at time step t, and $\odot$ denotes pointwise multiplication. The update gate u acts as a memory cell and corresponds with the following expression:

$$u = \sigma(W_h h^{\langle t-1\rangle} + W_{\tilde{h}} x + b) \tag{16}$$

Here, $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid activation function, W_h and $W_{\tilde{h}}$ are learnable weight matrices for respectively the hidden state and the new hidden state, and b contains the bias weights. The update gate allows us to learn how much information from the new hidden state $\tilde{h}^{\langle t\rangle}$ will be used to update the current hidden state [28]. The original gating mechanism also includes a *reset gate* so as to potentially ignore the previous hidden state and replace it with the input. However, because in the case of LiteralE the gating mechanism is not used in successive time steps, but only to enhance a given embedding with literal information whenever the embedding is accessed, this becomes unnecessary.

In the context of KG embeddings, we refer to the result of an embedding operation, $emb(e)$, where $e \in \mathcal{E}$, as the embedded vector v_e. LiteralE operates by enhancing v_e with literal information. This information is provided by a literal vector, denoted as l_e. Essentially, LiteralE employs a flexible mapping function $\mathrm{g} : \mathbb{R}^{\mathrm{d}} \times \mathbb{R}^{\mathrm{N_{dr}}} \rightarrow \mathbb{R}^{\mathrm{d}}$ (with d the original embedding dimension) to combine any entity embedding with a literal vector, thereby producing a literal-enhanced propositionalisation of that respective entity. The literal vector has a dimensionality N_{dr} corresponding to the number of relations in the dataset for which literal objects have been found. Each dimensional entry inside this literal vector is simply filled in with the literal embedding v_l corresponding to the relationship, unless the relationship is not defined for the given subject, in which case the entry is set to zero. More succinctly put, for each entity e, a literal vector l_e is defined of dimensionality N_{dr}. Each entry inside l_e corresponds to a *data* relationship, meaning that the dimension can only represent a relationship for which literal values have been found in the dataset. For instance, say we have an entity called Jenny whose age is 35 and whose weight (in kilograms) is 65, the associated literal vector corresponds with [fc(35), fc(65)], with $\mathrm{fc} : \mathbb{R} \rightarrow \mathbb{R}^{\mathrm{d}}$.

To translate equation 16 into a usable mapping function, we notice that the original embedding v_e can be equated to the previous hidden state $h^{\langle t-1\rangle}$ and that the new hidden state $\tilde{h}^{\langle t\rangle}$ can be written as $\tanh(W_h[v_e, l_e])$, with l_e the literal vector as input x and W_h a weight matrix. Thus we get the following expression for the literal enhanced embedding of e:

$$\begin{aligned} v_e^{lit} = g(v_e, l_e) &= \sigma(W_h v_e + W_{\tilde{h}} l_e + b) \\ &\odot v_e + (1 - \sigma(W_h v_e + W_{\tilde{h}} l_e + b)) \\ &\odot \tanh(W_h[v_e, l_e]) \end{aligned} \tag{17}$$

As part of this paper, we will improve on this scheme in the following way. Using the gating mechanism mentioned above, it becomes possible to use literal embeddings to enhance literals' pre-existing relational embedding similar to how LiteralE already uses per-entity literal vectors to enhance those entities' relational embeddings. In other words, if we consider that embedding techniques normally treat literals as regular entities, we can enhance each literal's regular embedded representation with the information stored in the corresponding literal embedding.

To do this, we apply the gating mechanism to literal entities, but instead of enhancing their embeddings with a literal vector, we use the literal embedding directly. As the gating mechanism allows us to learn whether or not to ignore this information encoded by the embedding, it is ideally suited to our purpose. The question now becomes why adding this information would be useful. After all, for every triple with a literal object, LiteralE already incorporates literal information into the embedding of its subject. When calculating the score of that triple, the literal information would already be available, rendering this addition more or less redundant. Apart from scenarios where one might wish to have access to the literal embeddings directly, another benefit of this addition pertains to the corrupted triples generated via negative sampling.

Indeed, picture the following scenario, where we have an observed triple (*ex:Jenny*, *ex:age*, *35*^^xsd:int). Let us assume that Jenny has an associated literal vector [0, 35, 170], for data relationships *ex:marriedFor, ex:age, ex:height*, and that the literal 35 has [0, 0, 0] as its literal vector. In other words, Jenny has been married for 0 years, is 35 years old, and is 170 cm tall. The value 35 is associated with the value 0 for each of these properties because it does not, as a literal, participate in any data relationships. When scoring triple (*ex:Jenny*, *ex:age*, *35*^^xsd:int) the literal information is incorporated through the literal enhancement of v_{Jenny}. However, when we artificially generate a negative (i.e., invalid) triple (*ex:Mark*, *ex:age*, *35*^^xsd:int), where Mark has an associated literal vector [23, 50, 185], we lose the literal value of 35, which will consequently be ignored by the score function. Enhancing v_{35} with the literal embedding of value 35, namely fc(35), instead of the empty literal vector [0, 0, 0], allows us to preserve this information when scoring the triple.

Following the formulations of Nickel et al. embedding techniques each define a different scoring function $f(x_{ikj};\Theta)$ to estimate the degree of certainty that a given triple exists (so that $x_{ikj}=1$) given the parameter set Θ [10]. Based on the enhancement scheme presented in this subsection, we can now define the scoring functions of the embedding techniques that will be used during evaluation.

$$TransE : f(h,r,t) = ||g(v_h,l_h) + v_r - g(v_t,l_t)|| \tag{18}$$

$$DistMult : f(h,r,t) = ||g(v_h,l_h) * v_r * g(v_t,l_t)|| \tag{19}$$

Note that when either h or t is a literal entity, the literal vectors l_h and l_t should be replaced with the literal embeddings v^l_h and v^l_t.

5. Evaluation Setup

In the previous section, we gave a detailed overview of the entire negative sampling strategy based on fuzzy constraints, and we introduced a mechanism for integrating literal information with the negative sampling procedure. In this section, we will describe the evaluation setup used to verify the methodology's performance. First we will provide an overview of the datasets used to attain the results. Then we will finish by going over the evaluation procedure itself as well as the different settings that will be evaluated. All the code and data used to perform the evaluation described in this section was made available on GitHub[4].

5.1. Benchmark datasets

To test the negative sampling procedure we will rely on two different benchmark datasets containing RDF triples, each accompanied by an elaborate schema: AIFB[5] and MUTAG[6] [9]. The original purpose of these datasets was to facilitate benchmarking of specific relational prediction tasks [29]. For AIFB, the learning objective involves predicting the research group affiliation for various researchers in the dataset. The AIFB dataset as a whole is essentially a linked data description of the Institute of Applied Informatics and Formal Description Methods, its staff members and their group affiliations, as well as their publications. For the MUTAG dataset, the learning objective involves predicting whether certain complex molecules are mutagenic (which may or may not contribute to carcinogenesis) or not. Overall, these datasets were used to test prediction techniques that solve a *part* of the overall link prediction problem. Accordingly, AIFB partitions a number of researchers into fixed training, validation, and test sets, and then trains a predictor to determine which of four research groups a certain researcher is affiliated with. MUTAG does something similar for molecules.

[4]https://github.com/IBCNServices/FuzzyConstraints

[5]http://data.dws.informatik.uni-mannheim.de/rmlod/LOD_ML_Datasets/data/datasets/RDF_Datasets/AIFB/

[6]http://data.dws.informatik.uni-mannheim.de/rmlod/LOD_ML_Datasets/data/datasets/RDF_Datasets/MUTAG/

In the AIFB sub-problem described above, we want to estimate $P(A)$ with $a_{ikj} \in A$, so that $A \subseteq \{0,1\}^{N_{re} \times 1 \times 4}$, where N_{re} is the total number of researchers and the other two dimensions refer to the *affiliation* predicate and the four research groups, given a set of observed triples T and a parameter set Θ, i.e., $P(A|T,\Theta)$. Similarly, in the MUTAG sub-problem, we want to estimate $P(M)$ with $m_{ikj} \in M$, so that $M \subseteq \{0,1\}^{N_{cm} \times 1 \times 2}$, where N_{cm} is the total number of complex molecules and the other two dimensions refer to the *isMutagenic* predicate and the two associated possibilities. To evaluate link prediction *in general*, we can extend each sub-problem to the entire corresponding dataset. In other words, in both cases we want to estimate $P(Y)$ with $y_{ikj} \in Y$, so that $Y \subseteq \{0,1\}^{N_e \times N_r \times N_e}$, as specified earlier in section 3. This final formulation of the problem is the one we will use to evaluate our own methods with.

5.2. Dataset Preparation

To use these datasets in the proposed manner, some additional preparation is required. Each dataset comes supplied with a file containing all the triples in the dataset in some format. It is insufficient simply to load these triples into RDFLib[7] and define a random training-validation-test split. We must take care to curate each RDF dataset in order to make it more suitable as an evaluation benchmark. Recent studies have highlighted some problems with previous evaluation benchmarks that we will try to avoid as much as possible by taking a few precautions [30, 31].

We want to make sure that the AIFB and MUTAG datasets adhere to certain *fairness* standards and avoid test *leakage* [31]. To engender fairness, we must ensure that entity degrees are taken into consideration when constructing the test set. When performing uniformly distributed random sampling, entities with high degrees will appear very often in both training and test sets, and since these entities also significantly boost performance for relationships mentioning them, not taking such characteristics into account skews the perception we might have of a given model's performance. Another way of taking this into account, without overly impacting the sampling strategy simply involves using a popularity agnostic evaluation metric, such as the recently proposed *strat-hits@k* and *strat-mrr* [32]. To avoid test leakage, we must take care to eliminate inverse relationships from the dataset, so that semantically identical triples are not split between training and test sets. In AIFB for example, the relationship *employs* is the inverse of the relationship *affiliation*, so that if we have a triple (id2041instance, affiliation, id1instance), we are also likely to have a triple (id1instance, member, id2041instance) stating the exact same fact. It is perfectly possible for the first triple to appear in the training set, while the latter appears in the test set, as they are considered separate facts. We want to avoid this as it is equivalent to having to predict facts we have already encountered during training as part of the testing procedure. Such inverse facts also do not strictly add much to the training procedure. Relationships for which the inverse is defined, can be used to entail inverse facts in a deterministic fashion. These facts do not need to be trained for, which means they can just be discarded.

To ensure this, we employ the following procedure. For every property associated with an inverse property declaration (http://www.w3.org/2002/07/owl#inverseOf), we maintain only the property that is most popular in the dataset. Its less popular inverse is discarded. Besides inverse relationships defined inside the ontology, we also want to remove duplicate and reverse duplicates. For this we refer to the work of Akrami et al [30]. We use the exact same metrics as they did to identify these kinds of relations, with θ_1 and θ_2 both set to 0.75. For each pair containing a relation and a (reverse) duplicate relation, we again remove the relation associated with the smallest number of triples in the dataset.

We refer to table 1 for a statistical overview of both datasets. Here, *axioms* refers to the number of RDFS or OWL declarations we find in the ontology; *literals* refers to the number of statements pertaining to numerical or textual literals across the entire dataset; *triples* means the total number of individual triples (including those pertaining to literals) comprising the dataset; and *types*, *rel in*, *rel out*, *rel-vals in*, and *rel-vals out* refer to the average, minimum, and maximum number of types per entity, subjects per relationship, objects per relationship, subjects per data relationship, and objects per data relationship.

5.3. Preprocessing

Beyond these preparatory steps, a number of additional things need to be done before the datasets can be used to evaluate our methodology. Namely, we need to separate type information and literal information, so that we can have direct access to them. Type information

[7] https://rdflib.readthedocs.io/en/stable/

Dataset	axioms		literals		triples	types			rel in			rel out			rel-vals in			rel-vals out		
Name	RDFS	OWL	num	text	total	avg.	min.	max.	avg.	min.	max.	avg.	min.	max.	avg.	min.	max.	avg.	min.	max.
AIFB	0	152	0	7009	24504	16	2	86	331	4	1302	260	4	1302	305	9	1041	196	1	1014
MUTAG	86	0	7520	0	51520	4	1	7	1324	5	9317	2021	5	9317	7520	7520	7520	1051	1051	1051

Table 1

Dataset Statistics

we can isolate by looking for all triples containing relationship http://www.w3.org/1999/02/22-rdf-syntax-ns#type. These triples are removed from the dataset and are not included inside either the training or the test set; they are used solely as supplementary information.

Now that we have training data, validation data, test data, and type information, as well as a dedicated ontology, we can proceed with the final preparations. First we add the default modelling ontologies to the dedicated ontology and, as mentioned in section 3, compute the resulting ontology's deductive closure. For this we make use of the OWL-RL[8] tool with combined RDFS and OWL semantics. After expanding the combined ontology, we add this ontology to the graph containing all of the isolated type information, and expand the type store in identical fashion. The resulting type declarations are again removed from the ontology and stored in a separate type graph. Finally, we also add the expanded ontology to the training graph and again perform the same expansion. Any type declarations resulting from this, we also add to our type information set. In terms of literal information, we make sure that the literals in both the training and test sets are also associated with all the correct type declarations. For every literal we explicitly add the XMLSchema data type, http://www.w3.org/2000/01/rdf-schema#Resource, http://www.w3.org/2002/07/owl#Thing, and http://www.w3.org/2000/01/rdf-schema#Literal to our repository of type information.

Specifically for the *standard-fuzzy* and *hybrid-fuzzy* sampling approaches, we also construct the literal clusters according to algorithm 6 and we gather all the constraint information we can find within each respective dataset. For the latter, this involves looking up all *owl:onProperty* restrictions defining an *owl:allValuesFrom* declaration as well as all *rdfs:domain* and *rdfs:range* declarations. As explained earlier, conditional OWL constraints of this kind do not explicitly say anything about the type of a triple's subject entity. They merely condition the types of the object on those of the subject. Implicitly, however, the conditional subject type is also the type actually expected by the relationship, in that all valid triples figuring the relationship in question will most likely also feature head entities with the conditional subject type. For this reason, we have opted to convert all OWL constraints to corresponding RDFS domain and range constraints. We can do this straightforwardly by setting the domain to the restricted subject type and the range to the object of the *owl:allValuesFrom* relationship. Following the example given in subsection 4.1, the new domain of *ex:age* becomes *ex:Person*, while the new range becomes *xsd:int*.

5.4. Procedure

The testing procedure follows the same methodology as the one sketched out in the original work by Bordes et al. [33]. This procedure requires the trained model to rank triples. For a given triple (e_s, r, e_o), we corrupt either the head (subject) or the tail (object), as many times as there are unique entities inside the entire dataset. For N_e entities, we get $N_e - 1$ corrupted triples, since we must exclude the original triple. We then also filter out any triples that we know already belong to the training set. Finally, we reinsert the valid triple. This whole set of triples is given to the embedding model, which predicts a score for each triple, thus allowing us to rank them from most to least likely to be true. We repeat this procedure for each triple in the test set, subjecting each to both head and tail corruption.

Different from the normal testing procedure, and, to our knowledge, all other KG embedding efforts, we allow our models (i.e., *standard-fuzzy* and *hybrid-fuzzy*) to use the strict schematic constraints defined in subsection 4.1 to potentially filter out many nonsensical candidates. The way we do this is simply by verifying for each batch of corrupted triples (and also the correct triple) which of these triples violate the schema based on the rules established in algorithm 1. In case a triple is considered valid, or no constraints were found for the respective relationship, the triple is retained for scoring by the model; otherwise the triple is ignored during scoring. After the model has attributed scores to all remaining triples, the triples that were deemed in

[8]https://owl-rl.readthedocs.io/en/latest/owlrl.html

violation of the schema are simply discarded. Taking into account the remarks made in subsection 5.2, the metrics relevant for evaluating ranking performance are given by equations 20 to 25.

$$\textit{strat-hits@}k(e_s, r, e_o) = \frac{w_o|\{e_o \in \text{top-k}(e_s, r, *)\}| + w_s|\{e_s \in \text{top-k}(*, r, e_o)\}|}{w_o + w_s} \tag{20}$$

$$\textit{strat-hits@}k = \frac{\sum_{r\in\mathcal{R}} w_r \sum_{(e_s,e_o)\in\mathcal{E}(r)} \textit{strat-hits@}k(e_s, r, e_o)}{\sum_{r\in\mathcal{R}} w_r} \tag{21}$$

$$\textit{strat-mrr}(e_s, r, e_o) = \frac{1}{w_s + w_o}\left(\frac{w_o}{\textit{rank}(e_o)\text{in}(e_s, r_k, *)} + \frac{w_s}{\textit{rank}(e_s)\text{in}(*, r_k, e_o)}\right) \tag{22}$$

$$\textit{strat-mrr} = \frac{\sum_{r\in\mathcal{R}} w_r \sum_{(e_s,e_o)\in\mathcal{E}(r)} \textit{strat-mrr}(e_s, r, e_o)}{\sum_{r\in\mathcal{R}} w_r} \tag{23}$$

$$\textit{strat-mr}(e_s, r, e_o) = \frac{1}{w_s + w_o}(w_o \textit{rank}(e_o)\text{in}(e_s, r_k, *) + w_s \textit{rank}(e_s)\text{in}(*, r_k, e_o)) \tag{24}$$

$$\textit{strat-mr} = \frac{\sum_{r\in\mathcal{R}} w_r \sum_{(e_s,e_o)\in\mathcal{E}(r)} \textit{strat-mr}(e_s, r, e_o)}{\sum_{r\in\mathcal{R}} w_r} \tag{25}$$

In the above, w_s, w_o, and w_r refer to popularity weights attached to subjects, objects, and relationships [32]. Here, $w_s = \frac{1}{N(s)^\beta}$, $w_o = \frac{1}{N(o)^\beta}$, and $w_r = \frac{1}{N(r)^\alpha}$, with $N(x)$ the frequency of x in the entire dataset. For our purposes, we choose two different sets of metrics based on the values of α and β. By choosing $\alpha = \beta = 0$, we are calculating the macro-averaged version of each metric. This version differs from the commonly used, mirco-averaged version where $\alpha = -1$ so that relations are weighted according to their frequency in the test set. Macro-averages discount this frequency, so that everything is weighed equally. We will be reporting both the standard mirco-averaged metrics as well as the more balanced macro-averaged ones.

Beyond the metrics used to evaluate our methodology, for the various embedding techniques we use a fixed set of hyperparameters. These include *batch size* $= 128$, *embedding size* $= 100$, *epochs* $=$ 100, *learning rate* $= 0.001$, *numerical embedding size* $=$ 100, *textual embedding size* $= 100$.

Finally, information on the various settings that will be evaluated can be found in table 2. We use these settings to gauge the impact of incorporating literals and of the magnitude of the negative ratio. Here, *literal enhancement* refers to the *embedding* enhancements of subsection 4.4 being applied or not. The *negative ratio* concerns the number of negative examples per positive example also referred to by algorithm 2. Testing the impact of the negative ratio is especially important in our case, given that our improvements mainly pertain to the negative sampling strategy itself.

Nr.	literal enhancement	neg. ratio
1	no	1
2	no	5
3	yes	1
4	yes	5

Table 2: Evaluation Settings for All Approaches

To evaluate the proposed enhancements properly, they must be contrasted with relevant reference approaches. When we refer to the *Bernoulli* approach, we mean the basic negative sampling procedure without any of the modifications presented throughout this paper. This basic approach involves the same steps as in algorithm 2, but without any of the logic pertaining to constraint validation. This means that the Bernoulli trick is used to evaluate whether we wish to corrupt the head or tail, and that whichever candidate is generated is always accepted without question.

Apart from the *Bernoulli* approach, we will compare our enhancements with three other negative sampling strategies covering the major categories in the state of the art: a *nearest neighbourhood* approach, a *typed LCWA* approach, and a *typed CWA* approach. The first of these is conceived as a generic example of the (data-driven) nearest neighbourhood sampling, which involves using K-Means with $\frac{|\mathcal{E}|}{k}$ centroids (where k = 25) to cluster the entity embeddings in the dataset every 5 epochs. Based on these clusters, the negative sampling procedure accepts a perturbation only if the replacement entity belongs to the same cluster as the original entity.

The (schema-enhanced) *typed LCWA* and *typed CWA* approaches were conceived along the lines of Krompaß et al.'s work on typed embeddings [7]. To reiterate, the LCWA approach creates artificial types based on the entity sets associated with each predicate, while the CWA approach makes use of strict schematic constraints to filter out nonsensical triples. Finally, the *standard-fuzzy* and *hybrid-fuzzy* approaches, introduced by this paper, were already described in subsection 4.3. For these, equation 12 with $a = 1.5$ was used to calculate the literal score.

6. Results & Discussion

Tables 3 to 14 collectively contain the results for the six different negative sampling approaches on the two benchmark datasets introduced earlier. Each approach was evaluated with two different embedding models, TransE and DistMult, for the four different model settings described in table 2, on ten different metrics (i.e., $MR_{micro/macro}$, $MRR_{micro/macro}$, $hits@1_{micro/macro}$, $hits@5_{micro/macro}$, $hits@10_{micro/macro}$). Additionally, in tables 15 and 16 we have gathered together all the results (for the various sampling techniques) for the basic model setting (i.e., setting 1 in table 2). This will help us compare the various techniques based on only the most rudimentary differences in how we sample, regardless of the impact of the negative ratio and the literal enhancements made to the embedding models. For tables 3 to 14, we have emboldened the best scores per dataset per model. The best scores overall, across these twelve tables, have also been underlined. For tables 15 and 16, however, we have only emboldened the best scores per dataset (independent of the embedding model).

Based on these results, we can first make a few general observations. First, it seems literal enhanced embeddings are usually able to improve overall model performance. However, the degree to which this is the case depends on the actual sampling technique being used. For instance, for the baseline *Bernoulli* approach, using literal enhanced embeddings usually outperforms using regular embeddings on the *MR* metrics. This means that introducing a literal enhancement usually pushes the correct triple higher in the overall ranking when averaged across different rankings. However, in this case, performance actually degrades on most of the other metrics. This suggests that while on average the model is able to rank the correct triple higher, its success is levelled out across different "queries".

An example might help to illustrate this. Suppose we have two embedding models, emb_a and emb_b, each associated with different rankings for two different test triples. Emb_a ranks both $triple_1$ and $triple_2$ at position 50. This means that the mean rank across these queries or rankings is $\frac{50+50}{2} = 50$ and the mean reciprocal rank is $\frac{\frac{1}{50}+\frac{1}{50}}{2} = \frac{1}{50}$. Emb_b, on the other hand, ranks $triple_1$ at position 1, but ranks $triple_2$ at position 99. The mean rank here is $\frac{1+99}{2} = 50$, but the mean reciprocal rank is $\frac{\frac{1}{1}+\frac{1}{99}}{2} = \frac{50}{99} > \frac{1}{2} > \frac{1}{50}$. Similarly, for emb_a, $hits@1 = hits@5 = hits@10 = 0$, while for emb_b, $hits@1 = hits@5 = hits@10 = 0.5$. An implicit tradeoff is apparently being introduced here between accuracy and precision, where the literal enhancements will promote higher precision at the cost of lower accuracies.

For other approaches besides the *Bernoulli* approach, this trend does not seem to hold. For e.g., the nearest neighbourhood approach (on AIFB) and the *typed CWA* approach (on AIFB and MUTAG), the literal enhanced models do appear to yield the best overall results. In terms of the effects of the negative ratio, it is straightforwardly the case that using more negative samples per positive sample usually increases performance across the board. For DistMult we observe that these trends are often only visible for the macro-averaged metrics, since the overall performance is usually drastically degraded for settings 3 and 4 on the micro-averaged counterparts.

If now we make an overall comparison, we see that the best two baseline models are the *nearest neighbourhood* and, surprisingly, the *Bernoulli* approach, with the *typed LCWA* approach coming in last. When we compare the *nearest neighbourhood* approach to the *fuzzy* sampling techniques, we observe that on AIFB they behave very similarly. The *standard-fuzzy* approach has the lowest *MR* scores overall (448.796 MR_{micro} and 651.225 MR_{macro}) and also shows very competitive macro-averaged results (0.133, 0.070, 0.204, 0.258

on *MRR* and *hits*@1/5/10). On the micro-averaged metrics, it falls slightly behind *Bernoulli* and *nearest neighbourhood* sampling (0.144, 0.075, 0.219, 0.289 on *MRR* and *hits*@1/5/10 versus 0.163, 0.082, 0.255, 0.334 for nearest neighbourhood and 0.151, 0.066, 0.247, 0.326 for Bernoulli).

If we compare the *standard-fuzzy* and *hybrid-fuzzy* approaches, we can see that on AIFB the hybrid approach performs significantly better than the standard approach on all metrics besides $MR_{micro/macro}$, coming much closer to the performance of the *Bernoulli* and *nearest neighbourhood* approaches. Specifically, the hybrid approach achieves 0.150, 0.070, 0.240, 0.326 on micro *MRR* and *hits*@1/5/10, and 0.131, 0.066, 0.200, 0.272 on macro *MRR* and *hits*@1/5/10, while the standard approach achieves 0.144, 0.075, 0.219, 0.289 on micro *MRR* and *hits*@1/5/10, and 0.133, 0.070, 0.204, 0.258 on macro *MRR* and *hits*@1/5/10. It appears that the hybrid approach is able to find a better trade-off between a good mean rank score and getting a larger number of high rankings across different "queries".

On MUTAG, the *hybrid-fuzzy* approach actually evinces some of the best overall scores among all negative sampling approaches (883.710, 0.091, 0.060, 0.121, 0.183 on micro *MR*/*MRR* and *hits*@1/5/10, and 484.001, 0.235, 0.158, 0.305, 0.325 on macro *MR*/*MRR* and *hits*@1/5/10) with the *standard-fuzzy* approach coming in second. Specifically, Looking at individual settings here, we also note that using the embedding enhancements outlined in subsection 4.4 adds very little to the embedding procedure or often even degrades the performance gains made by using fuzzy sampling. Note that settings 3 and 4, where the literal enhancements are applied, do not determine whether or not literal clustering is used. Literal clusters are always used by default in the fuzzy sampling scheme whenever a literal is encountered as a substitution candidate. With respect to the use of expensive literal embedding enhancements in the vein of LiteralE, the poor results might suggest that simply adding literal awareness to the sampling scheme might be more than sufficient to incorporate literal information effectively, as adding further enhancements only degrades performance and simultaneously increases computation costs. Further investigation will be required to arrive at more definitive conclusions.

Finally, we can confirm some of these observations by looking at tables 15 and 16. Without any enhancements (so, using the settings that are computationally the least demanding), the fuzzy sampling approaches achieve the largest number of high scores on AIFB (7 out of 10), with *Bernoulli* and *nearest neighbourhood* sampling following closely behind. On MUTAG, the fuzzy sampling approaches evince the highest scores on all metrics. These tables also allow us to see clearly the holistic differences between the micro-averages and the macro-averages. For AIFB, the micro-averaged scores are better than the macro-averages. This means that all of the approaches suffer to some degree from popularity bias. However, we can see that the absolute gap between these different kinds of scores is smaller for the fuzzy sampling techniques than for the state of the art techniques. For MUTAG, the same fuzzy sampling techniques actually do not suffer from this trend at all. Here, the macro-averages are even better than the micro-averages.

In closing, when we take a look again at tables 3 to 14, we find that, across all settings, the fuzzy sampling approaches show significant performance increases with respect to the state of the art. These increases are indicated between parentheses next to the respective metric scores and are calculated with reference to the best baseline score (i.e., they reflect the degree of improvement with respect to that score).

7. Conclusion

In this paper we investigated how fuzzy constraints could be used to improve negative sampling for KG embeddings. We also looked at how these constraints could be made use of further to integrate literal awareness into the sampling strategy directly, and leverage the additional information literals are able to convey about the facts in the KG.

To evaluate the effectiveness of these improvements, we compared the fuzzy negative sampling strategy to a number of baseline techniques across a few different settings. As part of this evaluation, we wanted to gauge how our literal-aware sampling strategy would compare to literal-enhanced embeddings. To this end, we proposed using an extension of an existing enhancement technique called LiteralE to enrich existing embeddings with literal information directly.

Based on thoroughgoing experimentation on two benchmark datasets, we found that the proposed strategies offered significant benefits to the state of the art across multiple dimensions. When we consider all the various model settings, we find that the *standard-fuzzy* approach offers competitive results on the AIFB dataset (with performance increases of up to 17.15% with respect to the state of the art), especially on the more unbiased, macro-averaged metrics, with the *hybrid-fuzzy* approach coming closely behind and even offering superior results on a number of metrics (notably, on $hits@5_{micro/macro}$ and $hits@10_{micro/macro}$). On the MUTAG dataset, the *hybrid-fuzzy* approach offers the overall best performance, achieving state of the art results across the board (with performance increases of up to 55.49%).

For both of the proposed techniques, we found that the effects of using literal-enhanced embeddings were by and large negative on the MUTAG dataset, with sometimes mild improvements on the AIFB dataset, suggesting the possible redundancy of these enhancements when using literal-aware sampling. Finally, when looking at the overall comparison of all sampling strategies for the baseline, unenhanced model setting, we found we were able to confirm these findings, with the fuzzy sampling strategies outperforming the state of the art on most metrics, and still offering competitive results whenever the state of the art proved superior.

With regard to future work, we will explore alternative definitions of the fuzzy membership functions, and investigate whether hybrid combinations can be formulated by combining fuzzy types with other kinds of fuzzy membership. Also, we would be interested in extending this study to other datasets and embedding models, and performing an in-depth study on the specific effects of various sampling strategies on the modelling capacities and biases of the various embedding techniques, all to get a better insight in the detailed mechanics of these improvements and the ways they affect the embeddings themselves.

Acknowledgments: Michael Weyns (1SD8821N) and Pieter Bonte (1266521N) are funded respectively by a strategic base research grant and a postdoctoral fellowship, both awarded by the Fund for Scientific Research Flanders (FWO).

Declarations

Reproducibility and code availability The code and data used to perform the evaluations described in this paper are provided on GitHub[9].

[9]https://github.com/IBCNServices/FuzzyConstraints

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE	1	714.229	0.139	0.059	0.226	0.307	1034.031	0.114	0.041	0.193	0.254
TransE	2	673.794	**0.151**	**0.066**	**0.247**	**0.326**	951.795	**0.128**	**0.053**	**0.214**	**0.275**
TransE	3	536.921	0.130	0.051	0.216	0.294	773.928	0.107	0.038	0.183	0.240
TransE	4	**536.584**	0.147	0.064	0.242	0.321	**786.063**	0.120	0.047	0.202	0.268
DistMult	1	871.217	0.059	0.023	0.089	0.131	1163.396	0.080	0.038	0.125	0.163
DistMult	2	690.831	**0.086**	**0.049**	**0.114**	**0.163**	1044.289	**0.111**	**0.064**	**0.153**	**0.200**
DistMult	3	605.405	0.055	0.021	0.086	0.125	**777.020**	0.048	0.021	0.072	0.103
DistMult	4	**596.341**	0.057	0.024	0.084	0.122	786.708	0.047	0.024	0.064	0.091

Table 3

Results for Bernoulli Approach, on AIFB

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE	1	3289.852	0.034	0.003	0.061	0.085	1685.353	0.123	0.016	0.250	0.272
TransE	2	**2098.615**	**0.060**	0.004	**0.113**	**0.153**	1219.451	0.151	0.041	**0.286**	**0.312**
TransE	3	3229.697	0.029	0.011	0.043	0.061	1292.940	0.137	0.046	0.254	0.272
TransE	4	2268.330	0.057	**0.026**	0.082	0.113	**1087.399**	**0.153**	**0.059**	0.279	0.301
DistMult	1	2663.255	0.030	0.006	0.038	0.075	3677.966	0.010	0.003	0.012	0.022
DistMult	2	**1643.673**	**0.091**	**0.049**	**0.118**	**0.177**	2495.676	**0.039**	**0.023**	**0.050**	**0.068**
DistMult	3	5120.864	0.015	0.006	0.018	0.027	2064.475	0.014	0.003	0.011	0.019
DistMult	4	5106.482	0.015	0.006	0.017	0.027	**2047.163**	0.013	0.003	0.009	0.017

Table 4

Results for Bernoulli Approach, on MUTAG

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE	1	666.345	0.127	0.043	0.216	0.303	931.913	0.104	0.026	0.183	0.250
TransE	2	655.707	0.152	0.067	0.251	0.329	939.605	0.127	0.051	**0.213**	0.271
TransE	3	**540.829**	0.145	0.069	0.228	0.302	833.498	0.111	0.039	0.185	0.254
TransE	4	547.434	**0.163**	**0.082**	**0.255**	**0.334**	**829.273**	**0.129**	**0.058**	0.202	**0.277**
DistMult	1	840.600	0.059	0.022	0.080	0.123	1247.770	0.083	0.042	0.114	0.155
DistMult	2	817.357	**0.086**	**0.050**	**0.116**	**0.156**	1274.207	**0.103**	**0.063**	**0.142**	**0.187**
DistMult	3	685.780	0.055	0.023	0.083	0.121	896.317	0.051	0.029	0.067	0.094
DistMult	4	**672.086**	0.064	0.034	0.090	0.127	**874.615**	0.058	0.037	0.068	0.099

Table 5

Results for Nearest Neighbourhood Approach, on AIFB

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE	1	3259.917	0.034	0.004	0.058	0.085	1646.920	0.119	0.006	0.256	0.282
TransE	2	**2072.952**	0.059	0.003	**0.113**	**0.148**	1225.465	0.136	0.019	0.274	**0.308**
TransE	3	3266.627	0.030	0.011	0.042	0.061	1308.994	0.147	0.047	0.264	0.283
TransE	4	2291.557	**0.062**	**0.029**	0.093	0.121	**1112.710**	**0.169**	**0.078**	**0.283**	0.301
DistMult	1	2332.254	0.035	0.008	0.045	0.085	3201.040	0.012	0.003	0.015	0.026
DistMult	2	**2247.156**	**0.074**	**0.035**	**0.097**	**0.152**	3444.341	**0.030**	**0.016**	**0.039**	**0.056**
DistMult	3	4466.209	0.013	0.004	0.015	0.023	1934.535	0.012	0.002	0.014	0.020
DistMult	4	4330.455	0.013	0.004	0.015	0.024	**1883.101**	0.013	0.002	0.008	0.016

Table 6

Results for Nearest Neighbourhood Approach, on MUTAG

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE	1	776.730	0.115	0.021	0.228	0.308	1052.312	0.095	0.015	0.192	0.248
TransE	2	788.667	0.115	0.022	0.227	**0.311**	1122.298	**0.101**	0.019	**0.195**	**0.259**
TransE	3	**565.281**	0.117	0.029	0.222	0.303	**840.851**	0.098	0.019	0.190	0.244
TransE	4	613.661	**0.120**	**0.031**	**0.230**	**0.311**	871.253	0.098	**0.021**	0.188	0.256
DistMult	1	1199.925	0.058	0.022	0.084	0.128	1465.793	0.080	0.041	0.116	0.152
DistMult	2	1306.442	**0.079**	**0.046**	**0.104**	**0.149**	1592.076	**0.100**	**0.061**	**0.139**	**0.184**
DistMult	3	**663.892**	0.064	0.033	0.086	0.121	**828.179**	0.059	0.037	0.073	0.096
DistMult	4	675.880	0.060	0.026	0.088	0.123	861.042	0.056	0.032	0.077	0.101

Table 7

Results for Typed CWA Approach, on AIFB

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE	1	3494.057	0.027	0.000	0.052	0.083	4255.097	0.090	0.000	0.208	0.231
TransE	2	**3171.545**	0.034	0.000	0.070	0.112	3772.957	0.086	0.000	0.194	0.226
TransE	3	3308.213	0.039	0.007	0.069	0.097	3495.577	0.103	**0.020**	0.193	0.233
TransE	4	3214.532	**0.053**	**0.013**	**0.092**	**0.122**	**2951.702**	**0.112**	0.015	**0.230**	**0.256**
DistMult	1	4177.682	0.026	0.005	0.034	0.068	7250.304	0.008	0.002	0.010	0.018
DistMult	2	**3006.311**	**0.037**	**0.008**	**0.048**	**0.096**	5798.607	0.012	**0.003**	**0.015**	**0.027**
DistMult	3	9299.525	0.013	0.005	0.014	0.022	5423.980	**0.021**	**0.003**	0.008	0.022
DistMult	4	7447.237	0.012	0.004	0.014	0.021	**4724.062**	0.018	0.002	0.011	0.015

Table 8

Results for Typed CWA Approach, on MUTAG

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE	1	**1296.705**	0.066	0.000	**0.137**	**0.203**	1752.820	0.055	0.000	0.116	0.161
TransE	2	1373.099	0.063	0.004	0.121	0.185	1773.761	0.058	0.002	0.116	0.165
TransE	3	1297.948	0.064	0.001	0.126	0.185	**1624.404**	0.061	0.000	**0.125**	**0.173**
TransE	4	1311.705	**0.067**	**0.010**	0.115	0.172	1657.084	**0.064**	**0.007**	0.115	0.157
DistMult	1	**1546.302**	0.064	0.031	0.086	0.131	1876.403	0.073	0.037	0.101	0.146
DistMult	2	1577.273	**0.081**	**0.046**	**0.109**	**0.151**	1916.083	**0.091**	**0.058**	**0.118**	**0.162**
DistMult	3	1836.415	0.050	0.028	0.070	0.092	1417.835	0.050	0.030	0.064	0.081
DistMult	4	1867.831	0.049	0.025	0.071	0.097	**1378.280**	0.045	0.026	0.060	0.080

Table 9

Results for Typed LCWA Approach, on AIFB

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE	1	3299.826	0.027	0.000	0.051	0.084	3702.734	0.090	0.001	0.195	0.249
TransE	2	**2764.486**	0.037	0.000	0.079	0.120	2642.171	0.099	0.001	0.216	**0.263**
TransE	3	3145.006	0.042	0.009	0.072	0.101	2585.694	0.102	0.012	0.213	0.242
TransE	4	2937.287	**0.056**	**0.012**	**0.100**	**0.129**	**1898.697**	**0.116**	**0.016**	**0.234**	0.252
DistMult	1	3172.978	0.029	0.005	0.036	0.075	6355.749	0.009	0.002	0.011	0.021
DistMult	2	**2995.409**	**0.035**	**0.007**	**0.047**	**0.090**	6692.294	0.011	**0.003**	**0.015**	0.026
DistMult	3	9811.598	0.012	0.005	0.014	0.021	5238.079	**0.024**	**0.003**	0.008	0.017
DistMult	4	7852.114	0.012	0.005	0.014	0.022	**5065.931**	0.022	**0.003**	0.008	**0.027**

Table 10

Results for Typed LCWA Approach, on MUTAG

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE	1	536.062	0.126	0.063	0.187	0.260	731.084	0.108	0.052	0.162	0.226
TransE	2	530.526	**0.144**	**0.075**	**0.219**	**0.289**	764.727	**0.133** ($\uparrow$ 3.10%)	**0.070** ($\uparrow$ 17.14%)	**0.204**	**0.258**
TransE	3	466.615	0.092	0.041	0.143	0.196	658.166	0.085	0.034	0.137	0.185
TransE	4	**448.796** ($\downarrow$ 16.36%)	0.123	0.065	0.182	0.242	**651.225** ($\downarrow$ 17.15%)	0.102	0.051	0.156	0.203
DistMult	1	**609.935**	0.047	0.017	0.073	0.105	847.484	0.078	0.040	0.113	0.149
DistMult	2	642.154	**0.079**	**0.046**	**0.104**	**0.155**	972.478	**0.106**	**0.067**	**0.147**	**0.197**
DistMult	3	639.942	0.065	0.034	0.091	0.124	**723.591**	0.071	0.044	0.089	0.124
DistMult	4	642.246	0.063	0.035	0.086	0.116	740.654	0.070	0.044	0.084	0.114

Table 11

Results for Standard-Fuzzy Approach, on AIFB

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE	1	2465.671	0.027	0.014	0.031	0.043	961.293	**0.221**	**0.172** ($\uparrow$ 120.51%)	0.260	0.272
TransE	2	**1864.924**	**0.052**	**0.029**	**0.070**	**0.089**	**762.093**	0.217	0.149	**0.282**	**0.292**
TransE	3	2974.773	0.017	0.006	0.022	0.030	1162.411	0.168	0.083	0.258	0.262
TransE	4	2628.473	0.028	0.012	0.037	0.052	1046.113	0.175	0.090	0.260	0.277
DistMult	1	**1055.342**	**0.063**	**0.024**	**0.082**	**0.134**	**566.024**	0.215	0.145	**0.275**	**0.292**
DistMult	2	1419.238	0.053	0.022	0.069	0.114	714.104	**0.218**	**0.158**	0.270	0.285
DistMult	3	3216.896	0.025	0.012	0.032	0.041	1228.801	0.186	0.106	0.258	0.272
DistMult	4	3274.288	0.025	0.012	0.032	0.043	1245.030	0.188	0.111	0.258	0.266

Table 12

Results for Standard-Fuzzy Approach, on MUTAG

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE	1	600.297	0.135	0.054	0.227	0.304	866.344	0.121	0.045	0.203	0.270
TransE	2	588.758	**0.150**	**0.070**	**0.240**	**0.326**	874.003	0.131	**0.066**	0.200	**0.272**
TransE	3	539.604	0.122	0.051	0.196	0.268	781.302	0.115	0.047	0.184	0.251
TransE	4	**519.637**	0.143	0.067	0.226	0.301	**738.456**	**0.132**	0.064	**0.207**	0.267
DistMult	1	840.441	0.070	0.039	0.089	0.135	1091.965	0.089	0.050	0.121	0.173
DistMult	2	708.071	**0.082**	**0.045**	**0.113**	**0.161**	895.879	**0.105**	**0.063**	**0.145**	**0.194**
DistMult	3	**600.457**	0.072	0.039	0.101	0.135	**734.636**	0.067	0.038	0.090	0.121
DistMult	4	604.426	0.070	0.036	0.101	0.140	752.526	0.067	0.039	0.088	0.127

Table 13

Results for Hybrid-Fuzzy Approach, on AIFB

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE	1	1958.911	0.046	0.025	0.061	0.083	746.547	0.215	0.142	0.278	0.289
TransE	2	**1286.337**	**0.091** (−)	**0.060** ($\uparrow$ 22.45%)	**0.116**	**0.142**	**521.227**	**0.235** ($\uparrow$ 39.05%)	**0.158**	**0.305** ($\uparrow$ 6.64%)	**0.318**
TransE	3	2397.949	0.033	0.014	0.043	0.062	916.092	0.190	0.108	0.271	0.290
TransE	4	1760.630	0.068	0.036	0.095	0.122	713.662	0.206	0.116	0.295	0.312
DistMult	1	1240.806	0.050	0.017	0.065	0.111	647.001	0.210	0.137	0.273	0.287
DistMult	2	**883.710** ($\downarrow$ 42.85%)	**0.090**	**0.042**	**0.121** ($\uparrow$ 2.54%)	**0.183** ($\uparrow$ 3.90%)	**484.001** ($\downarrow$ 55.49%)	**0.226**	**0.145**	**0.298**	**0.325** ($\uparrow$ 4.05%)
DistMult	3	3224.497	0.025	0.012	0.032	0.041	1227.711	0.185	0.102	0.264	0.273
DistMult	4	3156.200	0.025	0.012	0.033	0.041	1195.782	0.199	0.125	0.265	0.273

Table 14

Results for Hybrid-Fuzzy Approach, on MUTAG

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE-Bernoulli	1	714.229	**0.139**	0.059	0.226	0.307	1034.031	0.114	0.041	0.193	0.254
TransE-NN	1	666.345	0.127	0.043	0.216	0.303	931.913	0.104	0.026	0.183	0.250
TransE-T-CWA	1	776.730	0.115	0.021	**0.228**	**0.308**	1052.312	0.095	0.015	0.192	0.248
TransE-T-LCWA	1	1296.705	0.066	0.000	0.137	0.203	1752.820	0.055	0.000	0.116	0.161
TransE-S-Fuzzy	1	536.062	0.126	**0.063**	0.187	0.260	731.084	0.108	**0.052**	0.162	0.226
TransE-H-Fuzzy	1	**600.297**	0.135	0.054	0.227	0.304	866.344	**0.121**	0.045	**0.203**	**0.270**
DistMult-Bernoulli	1	871.217	0.059	0.023	0.089	0.131	1163.396	0.080	0.038	0.125	0.163
DistMult-NN	1	840.600	0.059	0.022	0.080	0.123	1247.770	0.083	0.042	0.114	0.155
DistMult-T-CWA	1	1199.925	0.058	0.022	0.084	0.128	1465.793	0.080	0.041	0.116	0.152
DistMult-T-LCWA	1	1546.302	0.064	0.031	0.086	0.131	1876.403	0.073	0.037	0.101	0.146
DistMult-S-Fuzzy	1	609.935	0.047	0.017	0.073	0.105	**847.484**	0.078	0.040	0.113	0.149
DistMult-H-Fuzzy	1	840.441	0.070	0.039	0.089	0.135	1091.965	0.089	0.050	0.121	0.173

Table 15

Overall comparison for setting 1, on AIFB

Embedding	Setting nr.	MR_{micro}	MRR_{micro}	$hits@1_{micro}$	$hits@5_{micro}$	$hits@10_{micro}$	MR_{macro}	MRR_{macro}	$hits@1_{macro}$	$hits@5_{macro}$	$hits@10_{macro}$
TransE-Bernoulli	1	3289.852	0.034	0.003	0.061	0.085	1685.353	0.123	0.016	0.250	0.272
TransE-NN	1	3259.917	0.034	0.004	0.058	0.085	1646.920	0.119	0.006	0.256	0.282
TransE-T-CWA	1	3494.057	0.027	0.000	0.052	0.083	4255.097	0.090	0.000	0.208	0.231
TransE-T-LCWA	1	3299.826	0.027	0.000	0.051	0.084	3702.734	0.090	0.001	0.195	0.249
TransE-S-Fuzzy	1	2465.671	0.027	0.014	0.031	0.043	961.293	**0.221**	**0.172**	0.260	0.272
TransE-H-Fuzzy	1	1958.911	0.046	**0.025**	0.061	0.083	746.547	0.215	0.142	**0.278**	0.289
DistMult-Bernoulli	1	2663.255	0.030	0.006	0.038	0.075	3677.966	0.010	0.003	0.012	0.022
DistMult-NN	1	2332.254	0.035	0.008	0.045	0.085	3201.040	0.012	0.003	0.015	0.026
DistMult-T-CWA	1	4177.682	0.026	0.005	0.034	0.068	7250.304	0.008	0.002	0.010	0.018
DistMult-T-LCWA	1	3172.978	0.029	0.005	0.036	0.075	6355.749	0.009	0.002	0.011	0.021
DistMult-S-Fuzzy	1	**1055.342**	**0.063**	0.024	**0.082**	**0.134**	**566.024**	0.215	0.145	0.275	**0.292**
DistMult-H-Fuzzy	1	1240.806	0.050	0.017	0.065	0.111	647.001	0.210	0.137	0.273	0.287

Table 16

Overall comparison for setting 1, on MUTAG

References

[1] P. Cudré-Mauroux, Leveraging knowledge graphs for big data integration: the xi pipeline, *Semantic Web* **11**(1) (2020), 13–17.

[2] H.-G. Yoon, H.-J. Song, S.-B. Park and S.-Y. Park, A translation-based knowledge graph embedding preserving logical property of relations, in: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 907–916.

[3] S. Mehla and S. Jain, Rule languages for the semantic web, in: *Emerging Technologies in Data Mining and Information Security*, Springer, 2019, pp. 825–834.

[4] Q. Wang, Z. Mao, B. Wang and L. Guo, Knowledge graph embedding: A survey of approaches and applications, *IEEE Transactions on Knowledge and Data Engineering* **29**(12) (2017), 2724–2743.

[5] B. Kotnis and V. Nastase, Analysis of the impact of negative sampling on link prediction in knowledge graphs, *arXiv preprint arXiv:1708.06816* (2017).

[6] K.-W. Chang, W.-t. Yih, B. Yang and C. Meek, Typed tensor decomposition of knowledge bases for relation extraction, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1568–1579.

[7] D. Krompaß, S. Baier and V. Tresp, Type-constrained representation learning in knowledge graphs, in: *International semantic web conference*, Springer, 2015, pp. 640–655.

[8] G.A. Gesese, R. Biswas, M. Alam and H. Sack, A survey on knowledge graph embeddings with literals: Which model links better literal-ly?, *Semantic Web* **12**(4) (2021), 617–647.

[9] P. Ristoski, G.K.D. De Vries and H. Paulheim, A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web, in: *International Semantic Web Conference*, Springer, 2016, pp. 186–194.

[10] M. Nickel, K. Murphy, V. Tresp and E. Gabrilovich, A review of relational machine learning for knowledge graphs, *Proceedings of the IEEE* **104**(1) (2015), 11–33.

[11] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston and O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[12] Z. Wang, J. Zhang, J. Feng and Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: *Twenty-Eighth AAAI conference on artificial intelligence*, 2014.

[13] Z. Yan, R. Peng, Y. Wang and W. Li, Enhance knowledge graph embedding via fake triples, in: *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, pp. 1–7.

[14] J. Niu, Z. Sun and W. Zhang, Enhancing Knowledge Graph Completion with Positive Unlabeled Learning, in: *2018 24th International Conference on Pattern Recognition (ICPR)*, IEEE, 2018, pp. 296–301.

[15] S. Qin, G. Rao, C. Bin, L. Chang, T. Gu and W. Xuan, Knowledge Graph Embedding Based on Adaptive Negative Sampling, in: *International Conference of Pioneering Computer Scientists, Engineers and Educators*, Springer, 2019, pp. 551–563.

[16] S. Dash and A. Gliozzo, Distributional Negative Sampling for Knowledge Base Completion, *arXiv preprint arXiv:1908.06178* (2019).

[17] K. Toutanova and D. Chen, Observed versus latent features for knowledge base and text inference, in: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, 2015, pp. 57–66.

[18] M. Nickel and V. Tresp, Tensor factorization for multi-relational learning, in: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2013, pp. 617–621.

[19] D. McGuinness and C. Welty, OWL Web Ontology Language Guide, W3C Recommendation, W3C, 2004, http://www.w3.org/TR/2004/REC-owl-guide-20040210/.

[20] P. Hayes and P. Patel-Schneider, RDF 1.1 Semantics, W3C Recommendation, W3C, 2014, https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/.

[21] M. Weyns, P. Bonte, B. Steenwinckel, F. De Turck and F. Ongenae, Conditional Constraints for Knowledge Graph Embeddings, in: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs*, CEUR-WS, 2020.

[22] R. Guha and D. Brickley, RDF Schema 1.1, W3C Recommendation, W3C, 2014, http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[23] H. Knublauch and D. Kontokostas, Shapes Constraint Language (SHACL), W3C Recommendation, W3C, 2017, https://www.w3.org/TR/2017/REC-shacl-20170720/.

[24] G. Schreiber and M. Dean, OWL Web Ontology Language Reference, W3C Recommendation, W3C, 2004, http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

[25] J. Chen, X. Chen, I. Horrocks, E. B. Myklebust and E. Jimenez-Ruiz, Correcting knowledge base assertions, in: *Proceedings of The Web Conference 2020*, 2020, pp. 1537–1547.

[26] A. Kristiadi, M.A. Khan, D. Lukovnikov, J. Lehmann and A. Fischer, Incorporating literals into knowledge graph embeddings, in: *International Semantic Web Conference*, Springer, 2019, pp. 347–363.

[27] Q. Le and T. Mikolov, Distributed representations of sentences and documents, in: *International conference on machine learning*, PMLR, 2014, pp. 1188–1196.

[28] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, *arXiv preprint arXiv:1406.1078* (2014).

[29] P. Ristoski and H. Paulheim, Rdf2vec: Rdf graph embeddings for data mining, in: *International Semantic Web Conference*, Springer, 2016, pp. 498–514.

[30] F. Akrami, M.S. Saeef, Q. Zhang, W. Hu and C. Li, Realistic re-evaluation of knowledge graph completion methods: An experimental study, in: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1995–2010.

[31] A. Rossi and A. Matinata, Knowledge Graph Embeddings: Are Relation-Learning Models Learning Relations?, in: *EDBT/ICDT Workshops*, 2020.

[32] A. Mohamed, S. Parambath, Z. Kaoudi and A. Aboulnaga, Popularity Agnostic Evaluation of Knowledge Graph Embeddings, in: *Conference on Uncertainty in Artificial Intelligence*, PMLR, 2020, pp. 1059–1068.

[33] A. Bordes, J. Weston, R. Collobert and Y. Bengio, Learning structured embeddings of knowledge bases, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 25, 2011.

[34] J. Hendler, O. Lassila and T. Berners-Lee, The semantic web, *Scientific American* **284**(5) (2001), 34–43.

[35] K. Kersting, L. De Raedt and S. Kramer, Interpreting Bayesian logic programs, in: *Proceedings of the AAAI-2000 workshop on learning statistical models from relational data*, 2000, pp. 29–35.

[36] J. Cussens, Loglinear models for first-order probabilistic reasoning, in: *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., 1999, pp. 126–133.

[37] S. Muggleton, Learning stochastic logic programs, *Electron. Trans. Artif. Intell.* **4**(B) (2000), 141–153.

[38] H. Khosravi and B. Bina, A survey on statistical relational learning, in: *Canadian Conference on Artificial Intelligence*, Springer, 2010, pp. 256–268.

[39] B. Ding, Q. Wang, B. Wang and L. Guo, Improving knowledge graph embedding using simple constraints, *arXiv preprint arXiv:1805.02408* (2018).

[40] Y. Wu, J. Pan, P. Lu, K. Lin and Z. Yu, Knowledge Graph Embedding Translation Based on Constraints,“, *Journal of Information Hiding and Multimedia Signal Processing, Ubiquitous International* (2017), 5.

[41] G. Diaz, A. Fokoue and M. Sadoghi, EmbedS: Scalable, Ontology-aware Graph Embeddings, 2018. doi:10.5441/002/edbt.2018.40.

[42] A. Fokoue, I. Horrocks, B.C. Grau, Z. Wu and B. Motik, OWL 2 Web Ontology Language Profiles (Second Edition), W3C Recommendation, W3C, 2012, http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/.