International Journal of Software Engineering
and Knowledge Engineering
(2025) 1–49

DOI: 10.1142/S0218194025500421

World Scientific
www.worldscientific.com

Fuzzy Constraints for Knowledge Graph Embeddings

Michael Weyns *

IDLab, Ghent University-imec,
Technologiepark-Zwijnaarde 126
B-9052 Gent, Belgium
michael.weyns@ugent.be

Pieter Bonte

Department of Computer Science,
KU Leuven Campus Kulak, Etienne Sabbelaan 53
8500 Kortrijk, Belgium
pieter.bonte@kuleuven.be

Filip De Turck † and Femke Ongenae ‡

IDLab, Ghent University-imec, Technologiepark-Zwijnaarde 126
B-9052 Gent, Belgium
†*filip.deturck@ugent.be*
‡*femke.ongenae@ugent.be*

Received 13 June 2024
Revised 30 June 2025
Accepted 1 August 2025
Published

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 recognize 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 minimize the number of nonsensical negative samples being generated. By putting constraints on the construction of each candidate sample, these techniques have tried to maximize the number of true negatives outputted by the procedure. However, 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, suggesting that fuzzy semantics and literal-awareness can help to improve the quality of generated negative samples. On AIFB, our fuzzy negative sampling

* Corresponding author.

approach outperforms baselines on four metrics, with performance gains up to 17.14%. On MUTAG, our fuzzy negative sampling approach outperforms baselines on eight metrics, with performance gains up to 55.49%.

Keywords: Knowledge graph embeddings; negative sampling; type constraints; fuzzy sets.

1. Introduction

In recent years, knowledge graphs (KGs) have increasingly proven themselves to provide an easy and scalable way of representing and disclosing data extracted from heterogeneous sources [1]. 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 specialized schematic conventions. In contrast with older expert systems, which were usually curated by a central authority, much of the SW 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].

Two complementary graph completion approaches can be applied to reduce sparsity. On the one hand, inference engines relying on rule-based deductive entailment can be used to infer new facts directly [3]. On the other hand, the more recently developed KG embedding techniques can be used to make inductive predictions about the existential likelihood of specific facts [4]. While rule-based completion relies heavily on formal semantics captured by 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 true facts from false 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 complementary in principle, deductive and inductive approaches to KG completion are rarely synergized. 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. By enriching the schema via deductive KG completion, we can leverage type information to better determine whether candidate negative samples make sense or not. Many negative sampling strategies have already tried to extend the standard approach involving the (random) corruption of positive examples [5]. While some of these approaches also

exploit axiomatic statements inside the KG to minimize the number of meaningless negatives, they are usually constrained to sharp interpretations of inclusion and exclusion [6–10]. More specifically, samples are either accepted or rejected on the basis of binary selection criteria, allowing little room for semantic nuance. By contrast, we will make use of fuzzy semantics to facilitate membership-based sample generation. This allows us to accept or reject negative samples based on fuzzy criteria, incorporating a range of factors pertaining to samples' schematic properties. As a result, we avoid reductive binary selection while also leveraging the schema to determine whether individual samples are more or less appropriate. 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 [11]. To be sure, literal information has already been incorporated into the modeling schemes of various inductive approaches. However, it has yet to be integrated into the sampling procedure, meaning that such information is by and large neglected when selecting appropriate counterfactuals.

Specifically, the novel contributions of this paper are as follows:

- 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 trade-off. Because both *closed-world* and *open-world* interpretations have merit in their own right, combining them will allow us to maximally utilize 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 [12]. 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 Sec. 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 Sec. 3, we then define the link prediction problem and provide all of the terminology relevant to our particular methodology. Following this, Sec. 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, Sec. 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]. Since every KG is incomplete to some degree, such an assumption is usually incorrect and therefore ineffective. Better alternatives involve randomly perturbing existing triples (by replacing either the head or the tail with another entity) or assuming 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 [13]. 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 note 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 the TransE embedding model: using so-called *filtered* negative samples [14]. We subject filtered negative samples to perturbation as per usual, but then we additionally check them against the valid triples in the train set. In case the perturbed triple appears in this set, we generate a new perturbation 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 [15]. The Bernoulli trick is used to reduce false negative triples by resorting to 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.

2.1. *Improved data-driven negative sampling*

More recent improvements include the use of fake triples, positive unlabeled learning, adaptive negative sampling, distributional negative sampling, self-adversarial sampling, mixing for harder negative samples, structure-aware negative sampling, textually enhanced negative sampling and simple sampling [16–24]. Amongst these

various improvements, there are numerous similarities. For example, distributional sampling is very similar to adaptive sampling, the primary difference being the use of external embeddings, as opposed to its own embeddings, in the former to generate more suitable replacement candidates. Due to this similarity, we can subsume both under the more general category of *nearest neighborhood sampling* techniques. The shared characteristic of the latter is the usage of some selection criterion (e.g. the similarity between pre-trained or adaptively trained embeddings) to select neighboring entities as replacements [5]. Other variants of the nearest-neighborhood paradigm are simple sampling, textually enhanced negative sampling and structure-aware negative sampling. While sampling neighbors to generate perturbation candidates will often (though not necessarily) lead to more sensical triples, there is no contingency in place to mitigate the possibility that these will be false negatives.

Apart from nearest neighborhood sampling, 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 [16]. 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. Next, positive unlabeled learning employs a two-stage logistic regression filter to iteratively refine the pool of negative samples [17]. 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. Similar shortcomings may be attributed to self-adversarial sampling [20], which makes use of a sampling distribution derived from statistical thermodynamics to score negative samples differently via the loss function. While mixing for harder samples [21] is explicitly geared toward mitigating the problem of generating false negatives—by leveraging the same thermodynamic weighting scheme as in self-adversarial sampling to dynamically select for more robust (i.e. "harder") negatives—it ultimately suffers from the same lack of formal validation.

2.2. *Schema-enhanced negative sampling*

None of the techniques discussed so far have tried to incorporate knowledge that might be derived from the KG's schema. Most of these techniques have the benefit of being relatively lightweight in that they often require only the positive sample set to function. However, they ignore schematic knowledge that would be able to explicitise 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 scope of application 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 [25]. 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]. Notably, in the latter work, the general approach introduced by TRESCAL is extended to translation-based approaches, as discussed below.

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.

López-Rodríguez *et al.* and Bernardi *et al.* offer examples of more recent work on schema-enhanced negative sampling in the same vein as the approach suggested by Krompaß *et al.* [7–9]. Specifically, López-Rodríguez *et al.* suggest a negative sampling algorithm along the lines of the type-constrained sampling of Krompaß *et al.*, the main difference being that the prior seem to also exploit functional and inverse relations as well as equivalent and disjoint classes, in addition to the domain and range axioms already leveraged by the latter. Similarly, the synthetic negatives generation proposed by Bernardi *et al.* is almost entirely analogous to the work of Krompaß *et al.*, except for the valuable addition of allowing a portion of the negative samples to come from the total set of randomly perturbed triples (and not only from the set of corrupted triples conforming to the prevailing domain and range axioms). Finally, the work of Jain *et al.* presents an iterative negative sampling approach called ReasonKGE, which dynamically samples corrupted triples by relying on model predictions and ontology-based consistency checking [10]. As such, the negative sample set produced by ReasonKGE will contain only incorrect triples, which would have led to consistency issues in the overall KG. Based on the schema-enhanced negative sampling approaches reviewed so far, we observe that each of these relies on binary selection criteria, as emphasized previously in Sec. 1.

Besides trying to leverage constraints to enrich the sampling procedure, it is also possible to incorporate type information directly into the embedding procedure itself, usually by modifying the embedding loss functions [26–30]. Since approaches of this kind typically sample negative triples in a way that is uninformed by schema

information, they are categorically different from the work presented here and as such will not be considered further.

2.3. *Summary and fuzzy negative sampling*

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

Techniques in the second group try to leverage type information in order to improve the quality of the generated samples. Whether this type information is inferred stochastically (as in the work of Toutanova *et al.*), or is derived directly from the available schema information (as in the work of Krompaß *et al.*), the goal is to use this information to constrain the number of relevant replacement candidates. Much the same is true for nearest neighborhood sampling strategies. However, while nearest neighborhood sampling and other data-driven approaches do this *positively* by defining alternative *suggestions* whenever an entity is perturbed, 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 builds on the schema-aware outline provided by Krompaß *et al.*, augmented with the notion shared by data-driven, nearest neighborhood 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 easily able to accommodate 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 and Background

Since 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 $(\mathcal{E}, \mathcal{P})$, where $\mathcal{E} = \{e_1, \ldots, e_{N_e}\}$ 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 $\mathcal{P} = \{p_1, \ldots, p_{N_p}\}$ refers to the set of all dyadic relationships (predicates) between these entities [13, 31]. Every relationship $p_i \in \mathcal{P}$ is a binary relationship between entities. Therefore, the KG might be considered a subset of the collection of all possible triples: $(e_i, p_k, e_j) \in \mathcal{E} \times \mathcal{P} \times \mathcal{E}$. 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 or objects (e.g. when they are defined in a given schema), the definition of a KG can be redefined as a subset of $(e_i, p_k, e_j) \in (\mathcal{E}^* \cup \mathcal{P} \cup \mathcal{C}) \times \mathcal{P} \times (\mathcal{E}^* \cup \mathcal{P} \cup \mathcal{C})$, with $\mathcal{E}^* = \mathcal{E} \backslash \mathcal{C}$. 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 $(e_i, p_k, e_j) \in (\mathcal{E}^* \cup \mathcal{P} \cup \mathcal{C}) \times \mathcal{P} \times (\mathcal{E}^* \cup \mathcal{P} \cup \mathcal{C} \cup \mathcal{L})$, with $\mathcal{E}^* = \mathcal{E} \backslash (\mathcal{C} \cup \mathcal{L})$. 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.

3.1. *Knowledge graph embeddings and negative sampling*

Now that we have defined what a KG is, we can turn to KG embedding models. A KG embedding model is a latent-feature model used for KG completion. In this context, each *possible* triple $x_{ikj} = (e_i, p_k, e_j)$ in the KG is associated with a binary random variable $y_{ikj} \in \{0, 1\}$, for which $y_{ikj} = 1$, if x_{ikj} exists and 0, otherwise. The purpose is to estimate $\mathrm{P}(Y)$ (the probability that each possible triple exists) with $y_{ikj} \in Y$ (so that $Y \in \{0, 1\}^{N_e \times N_p \times N_e}$, where N_e is the total number of assertional entities and N_p the total number of assertional relations), given a set of observed triples T and a parameter set Θ, i.e. $\mathrm{P}(Y|T, \Theta)$ [13].

To estimate $\mathrm{P}(Y)$, each embedding model defines a scoring function $f(x_{ikj}; \Theta)$. The two models we will be evaluating in this paper, TransE and DistMult, are

associated with the following scoring functions (where $v_e \in \mathbb{R}^d$ is the embedding of e and d is the embedding dimensionality) [14, 32]:

$$\text{TransE} : f(e_i, p_k, e_j) = \|v_{e_i} + v_{p_k} - v_{e_j}\|, \tag{1}$$

$$\text{DistMult} : f(e_i, p_k, e_j) = \|v_{e_i} \cdot v_{p_k} \cdot v_{e_j}\|. \tag{2}$$

Note that TransE and DistMult are additive and multiplicative variants of the same scoring scheme, both of which attempt to capture relationships between entities as basic linear transformations from one entity to the other.

When estimating $\mathrm{P}(Y|T, \Theta)$, 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, p_k, e_j) for which $e_i \in \mathcal{C} \cup \mathcal{P}$ or $e_j \in \mathcal{C} \cup \mathcal{P}$, 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,

$$\forall i, k, j, (e_i, p_k, e_j) \in T \;\Rightarrow\; e_i \notin \mathcal{C} \cup \mathcal{P}, \quad e_j \notin \mathcal{C} \cup \mathcal{P}, \tag{3}$$

$$\forall (e_i, p_k, e_j) \in T, (e_i, p_k, e_j) \in T^+ \iff y_{ikj} = 1, \tag{4}$$

$$\forall (e_i, p_k, e_j) \in T, (e_i, p_k, e_j) \in T^- \;\Rightarrow\; y_{ikj} = 0. \tag{5}$$

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 ready to hand—hence the need for negative sampling.

3.2. *Fuzzy sets*

Here, we will give a brief overview of the relevant concepts belonging to the domain of fuzzy set theory [33]. This will later be used to devise an alternative, more generic interpretation of constraints. A fuzzy set generalizes 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 $\mu_F(x) : x \in U \to [0, 1]$, 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. We posit 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 $\mu_F(x) \in]0, 1[$, then we say that x belongs to F only *to a degree*.

Fuzzy sets admit of various interpretations. A given set's membership function can be understood to express either a degree of *preference* (and, closely related, *similarity* or *correspondence*) or a degree of *uncertainty* [34]. While the first option is said to offer a *conjunctive* interpretation, the last option adheres to a *disjunctive* interpretation. Preference refers to the degree to which a given element is "true" or

fully realized within the fuzzy set. This interpretation is often chosen in the context of constraint modeling. 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.

4. Methodology

In this section, we first specify what we mean by *constraint-based* negative sampling (refer to Sec. 4.1), working out how the various forms of constraints are constructed and validated. Based on the concepts defined in Sec. 3.2, we explain how the strict interpretation of constraint-based negative sampling can be enhanced with the concept of fuzzy sets to induce a fuzzy interpretation of schematic constraints and constraint validation (refer to Sec. 4.2). We then explicitly define a way to calculate the fuzzy membership of potential replacement candidates and integrate the new fuzzy sampling strategy into the overall negative sampling procedure (refer to Sec. 4.3). 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 (refer to Sec. 4.4).

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). (Note that we provide more details on how this happens later on in Sec. 5.3.) 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*

When it comes to establishing the status of truth and falsehood within the context of KGs, the OWA and the CWA come into play. To reiterate, the SW 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. [35]" 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 decentralized, 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 encounter a specific use of the relation *rdf:type*,

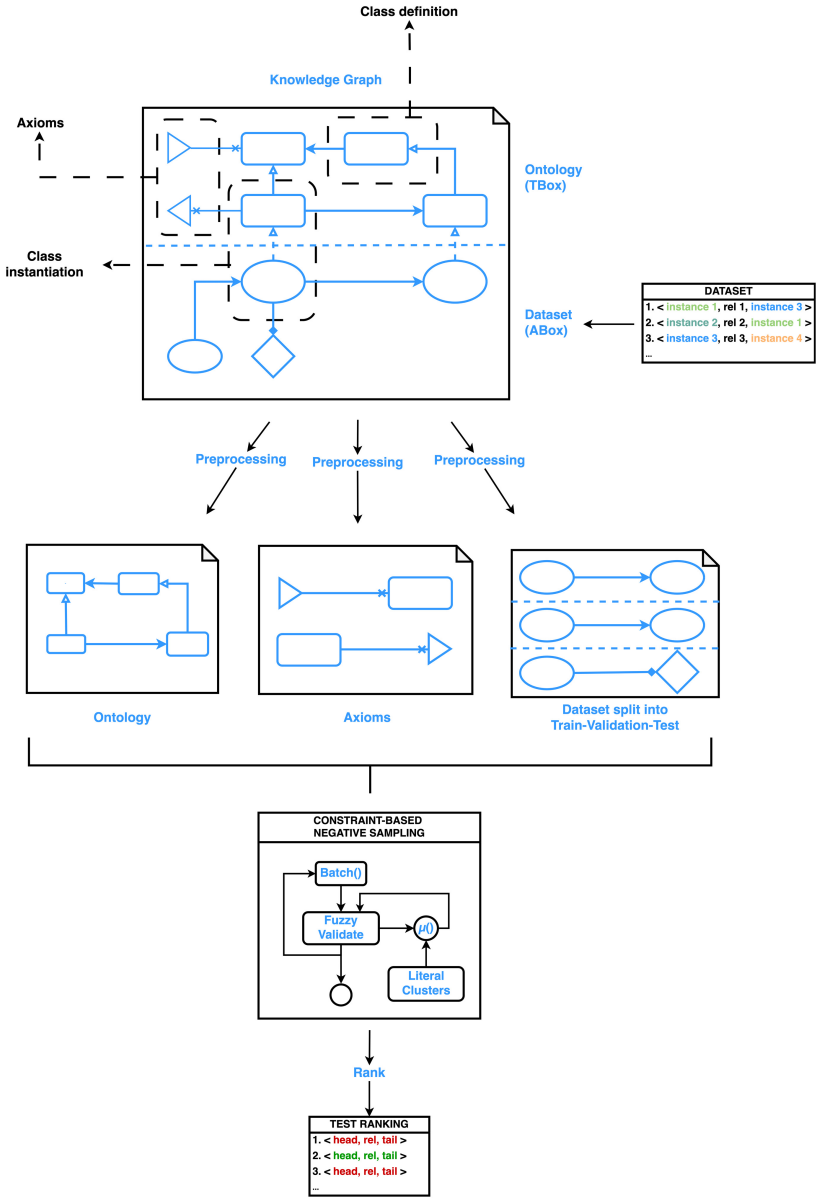

Fig. 1. Schematic overview of the fuzzy negative sampling methodology.

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*) [36]. It is crucial to recognize that OWL is not able to express constraints so much as it is able to posit axiomatic claims. This is in line with the OWA, which puts axioms in service of the entailment of additional facts.

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 RDF modeling ontologies,[a,b,c] one can compute the deductive closure of any given domain ontology. Once the ontology has been expanded according to the logic of the OWA, 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 as such impose a closed-world view with respect to OWL's usual axiomatic interpretation, they 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 positive 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. Conversely, in case we choose to interpret T^- in a closed-world manner, 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. Later on, we will suggest an interpretation of T^- that blends these worldviews in a way that combines the best of both.

In short, 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.

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

4.1.1. *Integrity constraints*

Following previous work, 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 [37]. The RDFS domain and range *constraints* can easily be derived from their open world formulations as follows [38]:

- *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:

$$\forall k \in \mathcal{K}, \forall c \in \mathcal{C}, \forall (p_k, \text{rdfs:domain}, c) \in \text{TBox} \Rightarrow \\ \forall i, j \in \mathcal{I}, (e_i, p_k, e_j) \Rightarrow (e_i, \text{rdf:type}, c), \tag{6}$$

$$\forall k \in \mathcal{K}, \forall c \in \mathcal{C}, \forall (p_k, \text{rdfs:range}, c) \in \text{TBox} \Rightarrow \\ \forall i, j \in \mathcal{I}, (e_i, p_k, e_j) \Rightarrow (e_j, \text{rdf:type}, c), \tag{7}$$

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

$$\forall k \in \mathcal{K}, \forall c \in \mathcal{C}, \forall (p_k, \text{rdfs:domain}, c) \in \text{TBox} \Rightarrow \\ \forall i, j \in \mathcal{I}, (e_i, \text{rdf:type}, c) \Rightarrow (e_i, p_k, e_j) \text{ is valid}, \tag{8}$$

$$\forall k \in \mathcal{K}, \forall c \in \mathcal{C}, \forall (p_k, \text{rdfs:range}, c) \in \text{TBox} \Rightarrow \\ \forall i, j \in \mathcal{I}, (e_j, \text{rdf:type}, c) \Rightarrow (e_i, p_k, e_j) \text{ is valid}. \tag{9}$$

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 SW'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. [39]" 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 [35].
- 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 [40].

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

$$\begin{aligned}&\forall k \in \mathcal{K}, \forall c, c' \in \mathcal{C}, \forall (b(c, p_k), \text{owl:onProperty}, p_k) \in \text{TBox} \;\&\\ &\quad \forall (b(c, p_k), \text{owl:allValuesFrom}, c') \in \text{TBox}\\ &\quad\Rightarrow\; \forall i, j \in \mathcal{I}, (e_i, \text{rdf:type}, c) \;\Rightarrow\; (e_i, p_k, e_j)\\ &\quad\Rightarrow\; (e_j, \text{rdf:type}, c'),\end{aligned} \tag{10}$$

$$\begin{aligned}&\forall k \in \mathcal{K}, \forall c, c' \in \mathcal{C}, \forall (b(c, p_k), \text{owl:onProperty}, p_k) \in \text{TBox} \;\&\\ &\quad \forall (b(c, p_k), \text{owl:someValuesFrom}, c') \in \text{TBox}\\ &\quad\Rightarrow\; \forall i \in \mathcal{I}, \exists j \in \mathcal{I}, (e_i, \text{rdf:type}, c)\\ &\quad\Rightarrow\; (e_i, p_k, e_j) \;\&\; (e_j, \text{rdf:type}, c').\end{aligned} \tag{11}$$

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

$$\begin{aligned}&\forall k \in \mathcal{K}, \forall c, c' \in \mathcal{C}, \forall (b(c, p_k), \text{owl:onProperty}, r_k) \in \text{TBox} \;\&\\ &\quad \forall (b(c, p_k), \text{owl:allValuesFrom}, c') \in \text{TBox}\\ &\quad\Rightarrow\; \forall i, j \in \mathcal{I}, (e_i, \text{rdf:type}, c) \;\Rightarrow\; (e_j, \text{rdf:type}, c')\\ &\quad\Rightarrow\; (e_i, p_k, e_j)\text{is valid},\end{aligned} \tag{12}$$

$$\begin{aligned}&\forall k \in \mathcal{K}, \forall c, c' \in \mathcal{C}, \forall (b(c, p_k), \text{owl:onProperty}, p_k) \in \text{TBox} \;\&\\ &\quad \forall (b(c, p_k), \text{owl:someValuesFrom}, c') \in \text{TBox}\\ &\quad\Rightarrow\; \forall i \in \mathcal{I}, \exists j \in \mathcal{I}, (e_i, p_k, e_j) \;\&\; (e_j, \text{rdf:type}, c')\\ &\quad\Rightarrow\; (e_i, \text{rdf:type}, c) \;\Rightarrow\; (e_i, p_k, e_j) \text{ is valid.}\end{aligned} \tag{13}$$

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.

The *VALIDATE* procedure (see 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 (see lines 2 and 3), 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

Algorithm 1. Validate a single RDF triple

```
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 | subject_types |> 0
7:             and not all([t in subject_types for t in domain]) then
8:              return False
9:          end if
10:         if | object_types |> 0
11:            and not all([t in object_types for t in range]) then
12:             return False
13:         end if
14:         return True
15:     else                                   ▷ in case of owl:allValuesFrom
16:         if | subject_types |> 0
17:            and any([t in subject_types for t in domain]) then
18:             if | object_types |> 0
19:               and all([t in object_types for t in range]) then
20:                 return True
21:             end if
22:             return False
23:         end if
24:     end if
25:     return True
26: end procedure
```

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 (see lines 7 and 11). 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 that 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 (see line 17), then we can proceed to verify the range in the same fashion as before (see line 19). Note that every check in Algorithm 1. also takes into account whether any type information can be found for the subject and object associated with the triple being evaluated (see lines 6, 10, 16 and 18). In case no information can be found, the element in question is considered valid

by default. Also 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: a CWA version and an OWA version. Algorithm 2. shows that for each batch in the original training set (see line 3), we consider each separate triple (see line 5) and, by relying on the aforementioned

Algorithm 2. CWA Constraint-Based Negative Sampling

```
1: procedure CWA Constraint-Based Negative Sampling(train_set, all_entities, neg_ratio)
2:     new_train_set ← ∅
3:     for batch in train_set do
4:         neg_batch ← ∅
5:         for (s, p, o) in batch do
6:             for i ← 0...neg_ratio do
7:                 pr ← RANDOM(0, 1)          ▷ Draw floating point number from Uniform(0, 1)
8:                 corrupted_triple ← (s, p, o)
9:                 if pr > BERNOULLI(p) then          ▷ Use Bernoulli sampling
10:                    do
11:                        s′ ← select(all_entities)
12:                        valid ← VALIDATE(s′, p, o)
13:                    while not valid or (s′, p, o) in train_set          ▷ valid for OWA
14:                    corrupted_triple ← (s′, p, o)
15:                else
16:                    do
17:                        o′ ← select(all_entities)
18:                        valid ← VALIDATE(s, p, o′)
19:                    while not valid or (s, p, o′) in train_set          ▷ valid for OWA
20:                    corrupted_triple ← (s, p, o′)
21:                end if
22:                APPEND(neg_batch, corrupted_triple)
23:            end for
24:        end for
25:        new_batch ← CONCAT(batch, neg_batch)
26:        APPEND(new_train_set, new_batch)
27:    end for
28:    return new_train_set
29: end procedure
```

Bernoulli sampling, corrupt either the subject (see line 11) or object (see line 17) 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 (see line 12), 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 (shown in Algorithm 2.), we accept the corrupted triple when it can be validated, while for the OWA procedure we do the opposite, and only accept the candidate substitute when the resulting triple actually violates the constraints. This way, constraint-based negative sampling can facilitate both interpretations of T^- mentioned at the beginning of Sec. 4.1.

4.2. *Fuzzy constraints*

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 behavior using fuzzy sets to model domains and ranges.

When modeling 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 discussed earlier (refer to Sec. 3.2), as this interpretation most closely resembles the way domains and ranges are determined semantically by the elements composing them.

We now define $S = \{e_i | (e_i, p_k, e_j) \in T\}$ and $O = \{e_j | (e_i, p_k, e_j) \in T\}$. For the purposes of fuzzy constraint evaluation, we might choose to model every predicate as a fuzzy relation $R : S \to O$, where $R \subseteq S \times O$ and where every element $(s, o) \in S \times O$ is associated with a certain membership score. In other words, $\mu_R((s, o)) \in [0, 1]$. 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 memberships 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. Algorithm 3. demonstrates how we can use these membership functions to enable fuzzy validation. This algorithm implements a fuzzy alternative to Algorithm 1. Accordingly, Algorithm 3. can be integrated into Algorithm 2. by replacing the calls to the *VALIDATE* procedure (see lines 12 and 18) with calls to the *FUZZY VALIDATE* procedure, taking care to add the correct *position* variable for each call (*position* should be equal to *head* on line 12 and equal to *tail* on line 18).

Algorithm 3. Fuzzy validate a single RDF triple

```
1: procedure FUZZY VALIDATE(s, p, o, position)
2:     pr ← RANDOM(0, 1)      ▷ Draw floating point number from Uniform(0, 1)
3:     if position = head and μ_S^p(s) ≥ pr then
4:         return True
5:     end if
6:     if position ≠ head and μ_O^p(o) ≥ pr then
7:         return True
8:     end if
9:     return False
10: end procedure
```

The standard way of turning a fuzzy set into a *crisp* set is based on the lambda-cut method. The lambda-cut set of a given fuzzy set F is defined as $F_\lambda = \{x|\mu_F(x) \geq \lambda\}$ with $0 \leq \lambda \leq 1$. A validation approach based on the lambda-cut set involves using the λ score as a cutoff value to decide whether or not we will accept the candidate triple. Algorithm 3. has a few benefits that outclass what is offered by this standard option. In Algorithm 3., we generate a random number $pr \in [0, 1]$ (see line 2) 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, whether it is a subject or an object, with a probability equal to this corresponding membership score (see lines 3 and 6). By contrast, using the λ score as a cutoff value imposes a 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. Meanwhile, the approach presented in Algorithm 3. allows us to select all candidates modulated exactly to their degree of preference, 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 difficult to find suitable replacements. This is solved by normalizing the memberships against the maximum across all possible candidates. In closing, we note how Algorithm 3. establishes a 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 favor the CWA or the OWA approach, we can invert the selection criterion to favor 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. Previous work by Chen *et al.* [41] 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. Notably, the link prediction problem tackled in our own work belongs to a species of quality improvement pertaining to *completion*, while the aforementioned work is specifically concerned with the *correction* of facts with object or literal entities that need to be replaced.

Chen *et al.*'s solution involves a multi-step pipeline 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 assertion 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 [41]. The final procedure for μ_S^R is listed as Algorithm 4.; the procedure for μ_O^R runs along very similar lines. The new definitions for μ_S^R and μ_O^R obtained via this integration are ideally suited for determining appropriate negative samples, seeing as we are calculating replacement values for corrupted subjects and objects to estimate their appropriateness given the prevailing constraints.

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 (see 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 $n_{p,e'} = |subjects_{p,e'} \cup \{e\}|$,

Algorithm 4. Membership function μ_S^R

1: **procedure** μ_S^R(p, e, train_set)
2: $car \leftarrow 0.0$ ▷ Cardinality score
3: $subjects_p \leftarrow \{s \mid (s,p,o) \in train\_set\}$ ▷ Subjects associated with predicate p
4: $objects_p \leftarrow \{o \mid (s,p,o) \in train\_set\}$ ▷ Objects associated with predicate p
5: $n \leftarrow \frac{\sum_{o \in objects_p} |\{s|(s,p,o) \in train\_set\} \cup \{e\}|}{|objects_p|}$
6: $SN(o) \leftarrow |\{s \mid (s,p,o) \in train\_set\}|$ ▷ #subjects associated with a given object o, for p
7: $SN_{max} \leftarrow \text{MAX}(\{SN(o) \mid o \in objects_p\})$ ▷ Max SN, across all objects associated with p
8: $car_p(k) \leftarrow \frac{|\{o \in objects_p | SN(o)=k\}|}{|objects_p|}$, for $k \in [1, SN_{max}]$ ▷ Cardinality for given SN
9: $car_p(k > a) \leftarrow \sum_{i=a+1}^{SN_{max}} car_p(k=i)$ ▷ Cumulative cardinality
10: $r \leftarrow \frac{n - SN_{max}}{SN_{max}}$ ▷ Exceeding rate, defined for $SN_{max} > 0$
11: **if** $SN_{max} == 0$ or $r \geq 0.8$ **then**
12: $car \leftarrow 0$
13: **else**
14: **if** $n == 1$ **then** ▷ Inverse functional predicate
15: $car \leftarrow car_p(k=1)$
16: **else** ▷ Non-functional predicate
17: $car \leftarrow car_p(k>1)$
18: **if** $r > 0$ **then**
19: $car \leftarrow car \cdot (1-r)$
20: **end if**
21: **end if**
22: **end if**
23: $SD(c) \leftarrow \frac{|\{s|s \in subjects_p, c \in C(s)\}|}{|subjects_p|}$ ▷ Supporting degrees for given class c
24: $SC(p) \leftarrow \{c \mid s \in subjects_p, c \in C(s)\}$ ▷ Subject classes associated with p
25: $con_{sp} \leftarrow 1 - \prod_{SC_{sp}(p) \cap C(e)} (1 - SD_{sp}(c))$ ▷ For specific subject classes
26: $con_{ge} \leftarrow 1 - \prod_{SC_{ge}(p) \cap C(e)} (1 - SD_{ge}(c))$ ▷ For generic subject classes
27: **return** $0.2 \cdot car + 0.8 \cdot (0.2 \cdot con_{ge} + 0.8 \cdot con_{sp})$
28: **end procedure**

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 Algorithm 3). In case the predicate is not (inversely) functional ($n \neq 1$), we can use the exceeding rate to progressively degrade the cardinality score (see lines 17 to 20). 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 23, $SD(c)$ 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, on line 24, $SC(p)$ 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 (see lines 25 and 26) 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 (see line 27) 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 [41].

4.3.2. *Standard and hybrid alternatives*

At this point, we should note that because Algorithm 4. computes a *replacement* score, it is likely that the negative samples generated during the execution of Algorithm 2. will in fact be *valid* test triples. Such valid triples would 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 alternative strategies 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 correspond to successfully *corrected* facts are rejected most often as negative

samples. The problem with this approach is that it also implies that we will reject many of the most sensible facts, supplying predominately facts that are nonsensical and hence of little value to the sampling procedure. 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 $\geq$ sign on lines 3 and 6), 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 and 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 will contrast two different approaches: On the one hand, we will enhance the membership score presented in Sec. 4.3 to take literal values into account based on a clustering mechanism. On the other hand, we will make use of embedding enhancements inspired by the work of Kristiadi *et al.* on LiteralE [42]. This way, we can evaluate the effect of integrating literals into the negative sampling procedure in two different, complementary ways.

4.4.1. *Literal clustering*

We start with the first proposition. Algorithm 5. 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 neighbors. However, in our case, we already know which elements belong to each cluster and how many clusters there are, since we can group literal 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 or 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. (We explain later on how individual literal embeddings are constructed.) Each embedding is a vector of size d, so that the centroid is also a vector of d, where a given dimension i is calculated as

Algorithm 5. literal clustering

```
procedure LITERAL CLUSTERING(p, train_set)
    literals ← {o | (s, p, o) ∈ train_set, o is literal}
    if LENGTH(literals) > 0 then
        embeddings ← GET EMBEDDINGS(literals)
        centroid ← MEAN(embeddings)
        radius ← MAX({COSINE DISTANCE(centroid, e) | e ∈ embeddings})
        cluster_p ← (centroid, radius)
        return cluster_p
    end if
end procedure
```

$\frac{\sum_{j=1}^{n} v_j[i]}{n}$ for n literal embeddings $v_j, j = 1, \ldots, n$. The radius is then defined as the greatest (cosine) 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, using a hinge function, as follows:

$$score_{lit} = \max\left(0.0, \frac{1}{a} \cdot \left(a - \left(\frac{\cos_{dist}(centroid, v_{lit})}{radius}\right)\right)\right). \tag{14}$$

This score is calculated by taking the ratio between the cosine distance $\cos_{dist}(v, v') = 1 - \cos_{sim}(v, v') = 1 - \left(\frac{\sum_{i=1}^{d} v_i v'_i}{\sqrt{\sum_{i=1}^{d} v_i^2}\sqrt{\sum_{i=1}^{d} {v'}_i^2}}\right)$ (which has been demonstrated to be an effective measure of similarity or dissimilarity when comparing high-dimensional, directional data [43]) 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 cut-off 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. Other membership functions, such as a form of exponential decay, could also have been chosen to represent literal cluster membership. However, the linear decay modeled by the hinge function corresponds well with our intuition regarding the interpretation of individual literal memberships.

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

$$score_{final} = 0.2 \cdot car + 0.8 \cdot (0.5 \cdot (0.2 \cdot con_{ge} + 0.8 \cdot con_{sp}) + 0.5 \cdot score_{lit}). \tag{15}$$

Here, we follow the weighting scheme suggested by Chen *et al.* when balancing out the influence of con_{ge} and con_{sp} [41]. 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, as we do not know on *a priori* basis how much they should ideally trade off against one another.

To get the literal embeddings required for Algorithm 5, 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 initialized with weights drawn from a uniform distribution, and with an input dimension of 1 and an output dimension of d [44]. The resulting embeddings are normalized per feature dimension. To embed textual data, we make use of Doc2Vec with vector size d and otherwise default parameters [45]. 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 also 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 can be formulated as follows:

$$h^{\langle t \rangle} = u \odot h^{\langle t-1 \rangle} + (1 - u) \odot \tilde{h}^{\langle t \rangle}. \tag{16}$$

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:

$$u = \sigma(W_h h^{\langle t-1 \rangle} + W_{\tilde{h}} x + b). \tag{17}$$

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 [46]. 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.

Recall that, 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 $\mathrm{g} : \mathrm{IR}^{\mathrm{d}} \times \mathrm{IR}^{N_{dr}} \rightarrow \mathrm{IR}^{d}$ (with d the original embedding dimension) to combine any entity embedding with a literal vector, thereby producing a literal-enhanced propositionalization 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 filled 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 of dimensionality N_{dr} is defined. 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 $\mathrm{f}c : \mathrm{IR} \rightarrow \mathrm{IR}^{d}$.

To translate Eq. (17) into a usable mapping function, we note 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:

$$\begin{aligned} v_e^{lit} = g(v_e, l_e) = \sigma(W_h v_e + W_{\tilde{h}} l_e + b) \\ \odot v_e + (1 - \sigma(W_h v_e + W_{\tilde{h}} l_e + b)) \\ \odot \tanh(W_h[v_e, l_e]). \end{aligned} \tag{18}$$

To further adapt this scheme to the context of negative sampling, we suggest the following improvement. 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 the relational embeddings of regular, non-literal entities. 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.

If we recall Sec. 3.1, 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 [13]. Based on the enhancement scheme presented in this subsection, we can now refine the scoring functions of the embedding techniques that will be used during evaluation.

$$\text{TransE} : f(s, p, o) = \|g(v_s, l_s) + v_p - g(v_o, l_o)\|, \tag{19}$$

$$\text{DistMult} : f(s, p, o) = \|g(v_s, l_s) * v_p * g(v_o, l_o)\|. \tag{20}$$

Note that when either s or o is a literal entity, the literal vectors l_s and l_o should be replaced with the literal embeddings v_s^l and v_o^l.

5. Evaluation Setup

In the previous section, we provided a detailed overview of the entire negative sampling strategy based on fuzzy constraints, and we introduced a mechanism for integrating literal information into 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.[d]

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

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[e] and MUTAG[f] [12]. The original purpose of these datasets was to facilitate benchmarking of specific relational prediction tasks [47]. 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 (see Fig. 2 for a visual depiction of a fragment of the dataset). For the MUTAG dataset, the learning objective involves predicting whether certain complex molecules are mutagenic or not (see Fig. 3 for a visual depiction of a fragment of the dataset). Overall, these datasets were used to test prediction techniques aimed at solving 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 adopts a similar procedure, but for molecules.

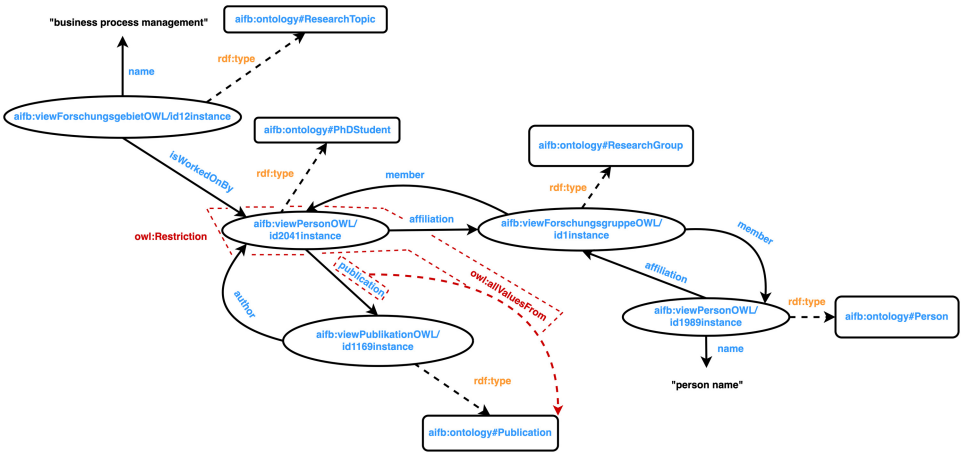

Fig. 2. (Color online) Fragment of the AIFB dataset. Here, the dense, directed, labeled arrows signify relationships between different entities, e.g. *affiliation*, or properties of entities, e.g. *name*. The dotted black arrows signify class or type instantiations, such that e.g. *aifb:viewPersonOWL/id1989instance* is an instance of an abstract *aifb:ontology#Person*. The dotted red arrow signifies an *owl:allValuesFrom* restriction conditioned on the PhD student *aifb:viewPersonOWL/id2041instance* and defined over the relationship *publication*.

[e] `http://data.dws.informatik.uni-mannheim.de/rmlod/LOD_ML\_Datasets/data/datasets/RDF_Datasets/AIFB/`.
[f] `http://data.dws.informatik.uni-mannheim.de/rmlod/LOD_ML_Datasets/data/datasets/RDF_Datasets/MUTAG/`.

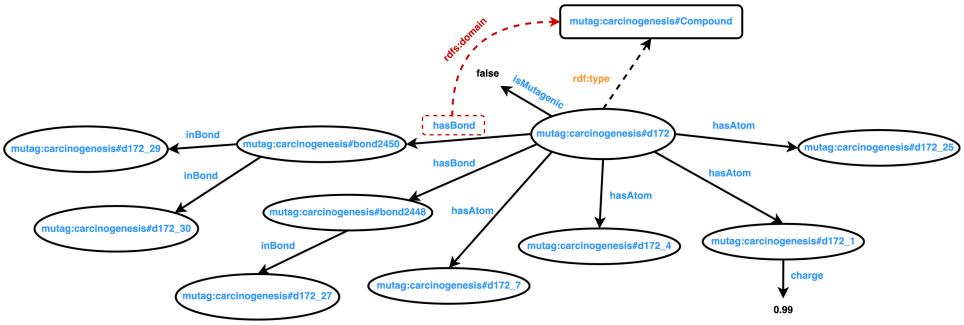

Fig. 3. (Color online) Fragment of the MUTAG dataset. Here, the dense, directed, labeled arrows signify relationships between different entities, e.g. *hasBond*, or properties of entities, e.g. *charge*. The dotted arrows signify class or type instantiations, such that e.g. *mutag:carcinogenesis#d172* is an instance of an abstract *mutag:carcinogenesis#Compound*. The dotted red arrow signifies an *rdfs:domain* axiom indicating that the relationship *hasBond* only admits subjects of type *mutag:carcinogenesis#Compound*.

In the AIFB sub-problem described above, we want to estimate $\mathrm{P}(A)$ with $a_{ikj} \in A$, so that $A \in \{0,1\}^{N_{re} \times 1 \times 4}$, 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. $\mathrm{P}(A|T,\Theta)$. Similarly, in the MUTAG sub-problem, we want to estimate $\mathrm{P}(M)$ with $m_{ikj} \in M$, so that $M \in \{0,1\}^{N_{cm} \times 1 \times 2}$, 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 $\mathrm{P}(Y)$ with $y_{ikj} \in Y$, so that $Y \in \{0,1\}^{N_e \times N_p \times N_e}$, as specified earlier in Sec. 3. This final formulation of the problem is the one we will use to evaluate our own methods.

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[g] 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 [48, 49].

We want to make sure that the AIFB and MUTAG datasets adhere to certain *fairness* standards and avoid test *leakage* [49]. To engender fairness, we must ensure

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

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* [50]. 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, employs, 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 duplicates and reverse duplicates. For this we refer to the work of Akrami *et al.* [48]. 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 an overview of the basic statistics for both datasets, after the aforementioned preparations have been undertaken:

- *Axioms* refers to the number of RDFS domain and range axioms or OWL restrictions (as defined in Sec. 4.1.1) we find in the ontology.
- *Literals* refers to the number of triples including numerical or textual literals across the entire dataset (i.e. not including axioms, prefix definitions, class definitions, etc.).
- *Triples* means the total number of individual triples (including those pertaining to literals) comprising the dataset (i.e. not including axioms, prefix definitions, class definitions, etc.).
- *Types*, *rel in*, *rel out*, *rel-vals in* and *rel-vals out* refer to the average, minimum and maximum number of unique types per entity (excluding blank nodes), incoming relationships per entity (i.e. the number of unique subject-relationship pairs with a

Table 1. Dataset statistics.

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	4	2	7	6.7	1	267	6.6	1	946	1.5	1	193	3	1	9
MUTAG	86	0	7520	0	51520	4	1	7	1.8	1	5	4.2	2	464	7.1	1	155	1	1	1

given entity as their object), outgoing relationships per entity (i.e. the number of unique relationship-object pairs with a given entity as their subject), incoming data-relationships per entity (i.e. same as *rel in*, but with only data-relationships), and outgoing data-relationships per entity (i.e. same as *rel out*, but with only data-relationships).

5.3. *Preprocessing*

Beyond these preparatory steps, a number of additional things needs 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. We can isolate type information 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 modeling ontologies to the dedicated ontology and, as mentioned in Sec. 4.1, compute the resulting ontology's deductive closure. For this, we make use of the OWL-RL[h] 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 XML Schema 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 5. 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

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

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 Sec. 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.* [51]. This procedure requires the trained model to rank triples. For a given triple (s, p, 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 entire 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 Sec. 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 violation of the schema are simply discarded. Taking into account the remarks made in Sec. 5.2, the metrics relevant for evaluating ranking performance are given by Eqs. (21)–(26).

$$\text{strat-hits@}k(s,p,o) = \frac{w_o|\{o \in \text{top} - \text{k}(s,p,*)\}| + w_s|\{s \in \text{top-k}(*,p,o)\}|}{w_o + w_s}, \tag{21}$$

$$\text{strat-hits@}k = \frac{\sum_{p\in\mathcal{P}} w_p \sum_{(s,o)\in\mathcal{E}(p)} \text{strat-hits@}k(s,p,o)}{\sum_{p\in\mathcal{P}} w_p}, \tag{22}$$

$$\text{strat-mrr}(s,p,o) = \frac{1}{s+o}\left(\frac{w_o}{\text{rank}(o)\text{in}(s,p,*)} + \frac{w_s}{\text{rank}(s)\text{in}(*,p,o)}\right), \tag{23}$$

$$\text{strat-mrr} = \frac{\sum_{p\in\mathcal{P}} w_p \sum_{(s,o)\in\mathcal{E}(p)} \text{strat-mrr}(s,p,o)}{\sum_{p\in\mathcal{P}} w_p}, \tag{24}$$

$$\text{strat-mr}(s,p,o) = \frac{1}{w_s + w_o}(w_o\text{rank}(o)\text{in}(s,p,*) + w_s\text{rank}(s)\text{in}(*,p,o)), \tag{25}$$

$$\text{strat-mr} = \frac{\sum_{p\in\mathcal{R}} w_p \sum_{(s,o)\in\mathcal{E}(p)} \text{strat-mr}(s,p,o)}{\sum_{p\in\mathcal{P}} w_p}. \tag{26}$$

In the above, w_s, w_o, and w_p refer to popularity weights attached to subjects, objects and relationships [50]. Here, $w_s = \frac{1}{N(s)^\beta}$, $w_o = \frac{1}{N(o)^\beta}$ and $w_p = \frac{1}{N(p)^\alpha}$, 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, micro-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 micro-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, optimizer = Adam, numerical embedding size = 100, textual embedding size = 100.

Finally, information about 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, LiteralE* refers to the adapted version of LiteralE that was introduced in Sec. 4.4 being applied or not. The *negative ratio* concerns the number of negative examples per positive example also referred to in 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.

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 follows the same steps as 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, except when it violates the limitations set by the filtered setting also mentioned in Sec. 2.

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 neighborhood* 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 neighborhood* 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.

Table 2. Evaluation settings for all approaches.

Nr.	LiteralE*	Neg. ratio
1	no	1
2	no	5
3	yes	1
4	yes	5

In other words, clustering is performed periodically to take into account updates to the individual embeddings during training. Based on the evolving clusters, the negative sampling procedure accepts a perturbation only if the replacement entity belongs to the same cluster as the original entity. After all, shared cluster membership suggests the entities in question are neighbors in terms of their embedded representations, potentially indicating semantic proximity. Like the *Bernoulli* approach discussed previously, the implementation of *nearest neighborhood* sampling follows the same steps as Algorithm 2. However, the main difference compared to the logic displayed in that algorithm concerns the *VALIDATE* procedure, which does not, in the case of *nearest neighborhood* sampling, include constraint validation, but rather, involves checking cluster memberships.

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. To be more specific, just like *nearest neighborhood* sampling, both *LCWA* and *CWA* sampling follow the same steps as shown in Algorithm 2. In fact, *CWA* sampling is just what Algorithm 2 outlines, with *VALIDATE* following the steps outlined in Algorithm 1. For *LCWA* sampling, the main difference from Algorithm 2 is in how the types are constructed. The implementation of the *VALIDATE* procedure for *LCWA* sampling follows the exact same steps as in Algorithm 1, but the types retrieved on lines 1 and 2 are not extracted from a predefined schema. Instead, entity types are assigned based on which entity sets they participate in. For a given triple, the subject entity is assigned the artificial type associated with the set of subjects belonging to the predicate of that triple. By assigning artificial types to each entity in this manner, it is then possible to perform constraint validation in the same way as for *CWA* sampling.

Finally, the *standard-fuzzy* and *hybrid-fuzzy* approaches, introduced by this paper, were already described in Sec. 4.3. For these, Eq. (14) with $a = 1.5$ was used to calculate the literal score, codifying the interpretation that distances exceeding the radius (when the ratio in Eq. (14) has a value greater than 1.0) correspond with literals that are "poorly" represented (as opposed to "decently" or "well-represented", in a fuzzy sense) by the cluster. As a final note, we would like to emphasize that every one of the approaches mentioned here incorporates the Bernoulli trick as well as the filtered setting.

6. Results and Discussion

Tables 3–14 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 and also on the dataset being evaluated. 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 leveled 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, $hits@1 = hits@5 = hits@10 = 0.5$. Regarding the phenomenon discussed above, an implicit trade-off is apparently introduced between accuracy and precision, where the literal enhancements will promote higher precision at the cost of lower accuracies. Or, in other words, occasionally correct rankings are sacrificed more often for greater consistency across rankings.

For other approaches besides the *Bernoulli* approach, this trend does not seem to hold as strongly. For e.g. the nearest neighborhood approach (on AIFB) and the *typed CWA* approach (on AIFB and MUTAG), the literal-enhanced models do appear to yield the best results overall (see especially the results for the TransE embedding model). In terms of the effects of the negative ratio, using more negative samples per positive sample usually increases performance across the board (except in the case of literal-enhanced DistMult, where the impact is largely inconclusive). For DistMult we observe that aforementioned 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 neighborhood* and, surprisingly, the *Bernoulli* approach, with the

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	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

Note: The best scores per dataset, per model are highlighted in bold.

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	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

Note: The best scores per dataset, per model are highlighted in bold.

Table 5. Results for nearest neighborhood 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	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

Note: The best scores per dataset, per model are highlighted in bold.

Table 6. Results for nearest neighborhood 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	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

Note: The best scores per dataset, per model are highlighted in bold.

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	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

Note: The best scores per dataset, per model are highlighted in bold.

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	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

Note: The best scores per dataset, per model are highlighted in bold.

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	**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

Note: The best scores per dataset, per model are highlighted in bold.

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	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**

Note: The best scores per dataset, per model are highlighted in bold.

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	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** (↑3.10%)	**0.070** (↑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** (↓16.36%)	0.123	0.065	0.182	0.242	**651.225** (↓15.80%)	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

Note: The best scores per dataset, per model are highlighted in bold.

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	2465.671	0.027	0.014	0.031	0.043	961.293	**0.221**	**0.172** (↑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

Note: The best scores per dataset, per model are highlighted in bold.

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	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

Note: The best scores per dataset, per model are highlighted in bold.

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	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** (↑22.45ϱ)	**0.116**	**0.142**	**521.227**	**0.235** (↑39.05%)	**0.158**	**0.305** (↑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** (↓42.85%)	**0.090**	**0.042**	**0.121** (↑2.54ϱ)	**0.183** (↑3.90%)	**484.001** (↓55.49%)	**0.226**	**0.145**	**0.298**	**0.325** (↑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

Note: The best scores per dataset, per model are highlighted in bold.

typed LCWA approach coming in last. When we compare the *nearest neighborhood* approach to the *fuzzy* sampling techniques, we observe that on AIFB they behave 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). However, also on AIFB, the best *fuzzy* sampling scores for the micro-averaged metrics fall behind those of *Bernoulli* and *nearest neighborhood* sampling (0.144, 0.075, 0.219, 0.289 on micro *MRR* and *hits*@1/5/10 for *standard-fuzzy* versus 0.163, 0.082, 0.255, 0.334 for *nearest neighborhood* 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 similarly to the standard approach on most metrics besides $MR_{micro/macro}$ (where the standard approach clearly outperforms the hybrid approach), also more closely approximating the performance of the *Bernoulli* and *nearest neighborhood* approaches. Specifically, on AIFB, the best scores achieved by the hybrid approach are 0.150, 0.070, 0.240, 0.326 on micro *MRR* and *hits*@1/5/10 and 0.132, 0.066, 0.207, 0.272 on macro *MRR* and *hits*@1/5/10, while those for the standard approach are 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', i.e. between precision and accuracy. On MUTAG, the *hybrid-fuzzy* approach 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, 0.305, 0.325 on macro *MR*/*MRR* and *hits*@5/10).

Looking more closely at individual settings, we can also note that using the embedding enhancements outlined in Sec. 4.4 often degrades performance on the MUTAG dataset, while usually leading to gains in performance on AIFB (taking into account, however, the precision-accuracy trade-off discussed earlier). While visible across all sampling approaches, the trend is clearest for fuzzy sampling, where, on the MUTAG dataset, embedding enhancements consistently lead to lower scores for all metrics. Note that for fuzzy sampling, 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 while simultaneously increasing computation costs. Also, with respect to MUTAG in particular, we should note that the 7520 literal triples contained in the dataset all make use of the same `http://dl-learner.org/carcinogenesis#charge` relationship, minimizing the effectiveness of the literal vector l_e introduced in Sec. 4.4.2, as its dimensionality, N_{dr}, will be equal to one. This is very different from the characteristics of AIFB, where multiple kinds of data relationships

Table 15. Overall comparison for setting 1, on AIFB.

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

Note: The best scores per dataset are highlighted in bold.

Table 16. Overall comparison for setting 1, on MUTAG.

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

Note: The best scores per dataset are highlighted in bold.

are present, leading to a higher dimensionality for the literal vectors and thus potentially more effective literal enhancements. Further investigation will be required to arrive at more definitive conclusions.

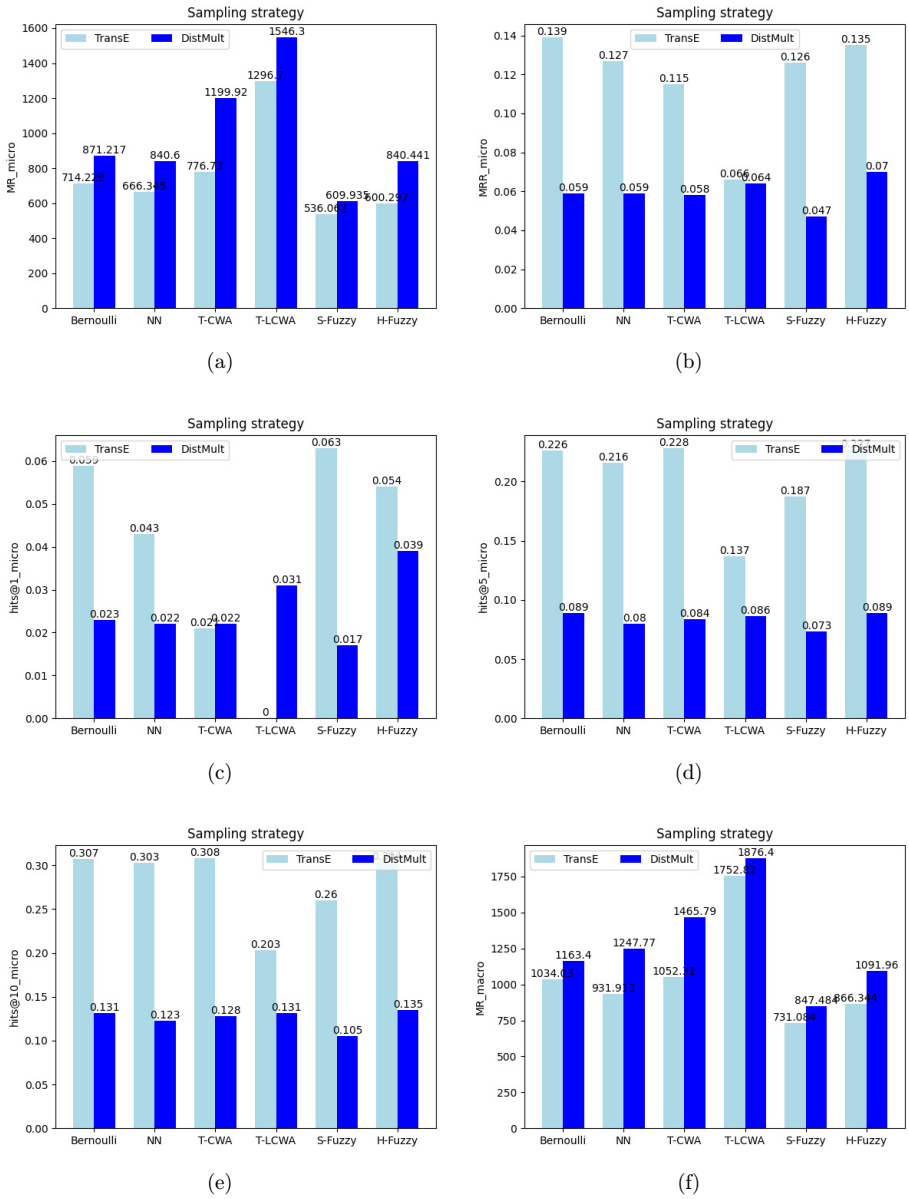

Fig. 4. Overall comparison for setting 1, on AIFB, visualized as separate charts. Each chart demonstrates the performance of the various sampling strategies on a specific metric, where the performance for each sampling strategy is shown for both of the tested embedding techniques, i.e. TransE and DistMult.

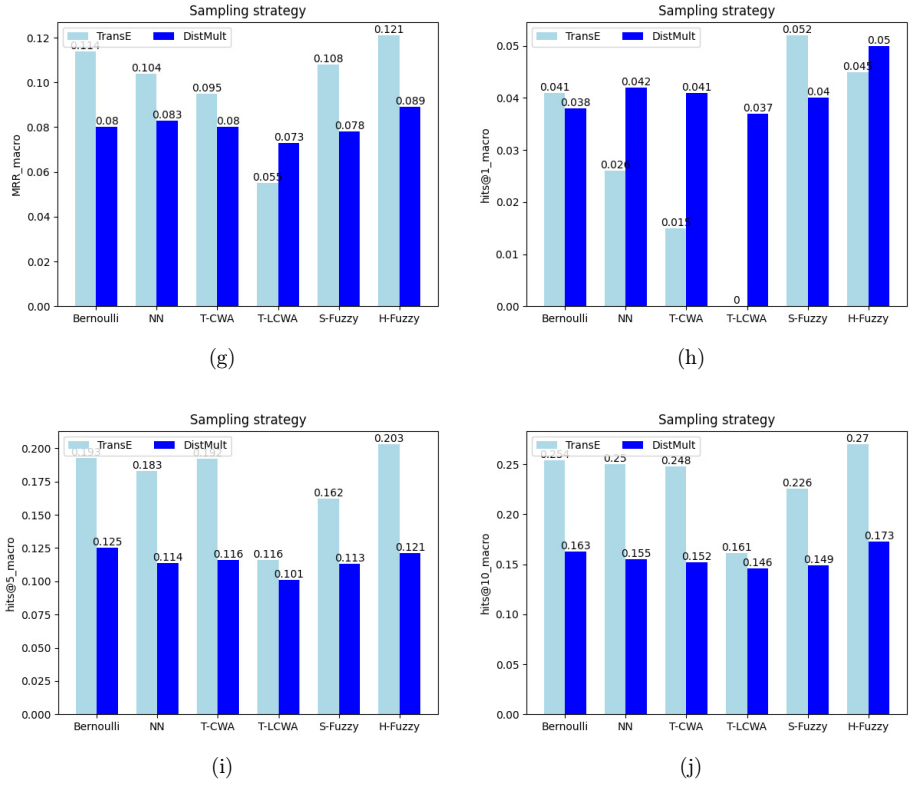

Fig. 4. (*Continued*)

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 neighborhood* 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 indicates 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 baseline techniques. For MUTAG, the same fuzzy sampling techniques actually do not suffer from this trend at all. Here, the macro-averages are better than the micro-averages. Practically speaking, with respect to the results presented in Tables 15 and 16, we can say that the fuzzy approaches will rank correct triples higher on average (i.e. they will evince lower micro and macro *MR*) and will more frequently perform accurate rankings (i.e. they will evince comparable or higher *hits*@1/5/10 scores) than the baseline approaches. Additionally, as stated

earlier, the gap between micro-averages and macro-averages is smaller for the fuzzy approaches, suggesting less exposure to popularity bias. Overall, then, based on these results, fuzzy sampling appears to lead to more reliable link prediction performance when compared to the other sampling strategies. For a graphical representation of the same observations, please refer to Figs. 4 and 5, where the results displayed in

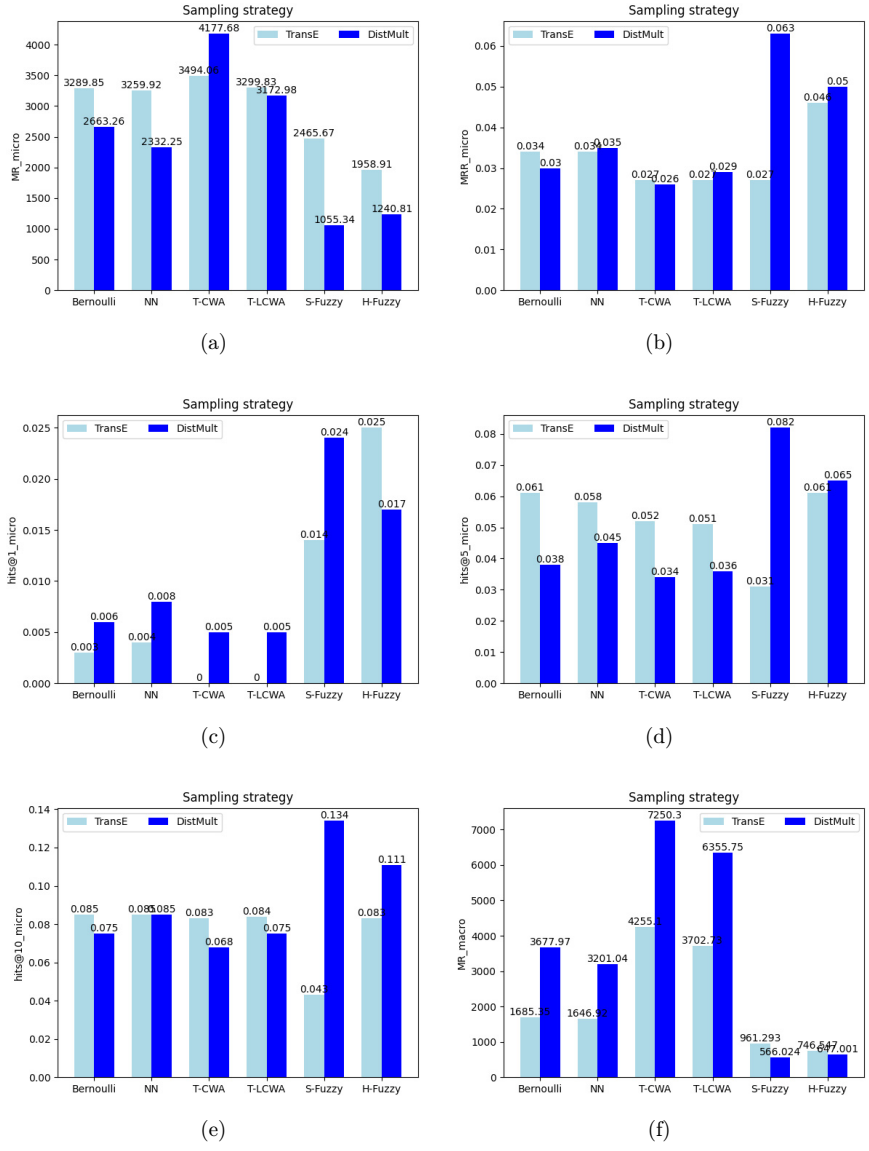

Fig. 5. Overall comparison for setting 1, on MUTAG, visualized as separate charts. Each chart demonstrates the performance of the various sampling strategies on a specific metric, where the performance for each sampling strategy is shown for both of the tested embedding techniques, i.e. TransE and DistMult.

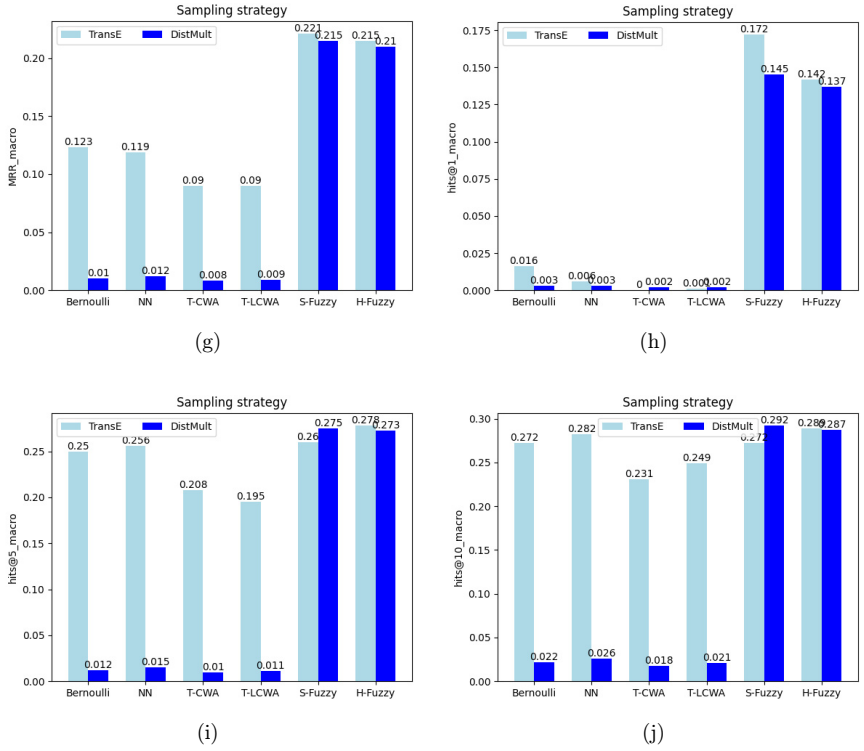

Fig. 5. (*Continued*)

Tables 15 and 16 have been laid out in separate graphs for the various metrics being evaluated (with sampling strategies displayed on the x-axis and metric values displayed on the y-axis).

In closing, when we take a look again at Tables 3 through 14, we find that, across all settings, the fuzzy sampling approaches show significant performance increases with respect to the baselines. 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). Specifically, for AIFB, the *standard-fuzzy* approach achieves a performance improvement of 16.36% for MR_{micro} with respect to the *Bernoulli* approach (which is the best baseline for this metric). The same *standard-fuzzy* approach also evinces performance increases of 15.80%, 3.10% and 17.14% on MR_{macro}, MRR_{macro} and $hits@1_{macro}$ (also for AIFB), with respect to the *Bernoulli* and *nearest neighborhood* approaches. For MUTAG, the *hybrid-fuzzy* approach evinces performance improvements of 42.85%, 22.45%, 2.54%, 3.90%, 55.49%, 39.05%, 6.64% and 4.05% on MR_{micro} and $hits@1_{micro}$, $hits@5_{micro}$, $hits@10_{micro}$, MR_{macro}, MRR_{macro}, $hits@5_{macro}$ and, $hits@10_{macro}$, with respect to the *Bernoulli* and *nearest neighborhood* approaches.

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 thorough 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 17.14% with respect to baselines), 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 $hits@10_{micro/macro}$). Notably, for AIFB, the *Bernoulli* and *nearest neighborhood* approaches do outperform the fuzzy sampling approaches on a number of metrics (specifically, on MRR_{micro}, $hits@1_{micro}$, $hits@5_{micro}$, $hits@10_{micro}$, $hits@5_{macro}$ and $hits@10_{macro}$). 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 baselines on most metrics, and still offering competitive results when baselines 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. More specifically, we could investigate ways to devise a generic framework to dynamically integrate different data-driven and schema-driven sampling strategies using fuzzy aggregations. This way, the benefits of combining data-driven and schema-driven approaches could be explored in far more detail, across a greater range of sampling strategies. 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 modeling capacities and biases of the various embedding techniques. This way, we aim to get a greater insight into the mechanics of these improvements and the ways they affect the embeddings themselves.

Acknowledgments

Funding 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).

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

ORCID

Michael Weyns https://orcid.org/0000-0002-6157-5997
Pieter Bonte https://orcid.org/0000-0002-8931-8343
Filip De Turck https://orcid.org/0000-0003-4824-1199
Femke Ongenae https://orcid.org/0000-0003-2529-5477

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 *Proc. 2016 Conf. North American Chapter 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, Singapore, 2019), pp. 825–834.
4. Q. Wang, Z. Mao, B. Wang and L. Guo, Knowledge graph embedding: A survey of approaches and applications, *IEEE Trans. Knowl. Data Eng.* **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:1708.06816.
6. K.-W. Chang, W.-T. Yih, B. Yang and C. Meek, Typed tensor decomposition of knowledge bases for relation extraction, in *Proc. 2014 Conf. Empirical Methods in Natural Language Processing*, 2014, pp. 1568–1579.
7. D. Krompaß, S. Baier and V. Tresp, Type-constrained representation learning in knowledge graphs, in *Proc. Int. Semantic Web Conf.*, 2015, pp. 640–655.
8. A. Bernardi and L. Costabello, Domain and range aware synthetic negatives generation for knowledge graph embedding models, arXiv:2411.14858.
9. Y. A. López-Rodríguez, O. G. Toledano-López, Y. Hidalgo-Delgado, H. González Diéz and R. Segundo-Guerrero, Good negative sampling for triple classification, in *Proc. Int. Workshop Artificial Intelligence and Pattern Recognition*, 2023, pp. 323–334.
10. N. Jain, T.-K. Tran, M. H. Gad-Elrab and D. Stepanova, Improving knowledge graph embeddings with ontological reasoning, in *Proc. Int. Semantic Web Conf.*, 2021, pp. 410–426.

11. 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.
12. 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 *Proc. Int. Semantic Web Conf.*, 2016, pp. 186–194.
13. M. Nickel, K. Murphy, V. Tresp and E. Gabrilovich, A review of relational machine learning for knowledge graphs, *Proc. IEEE* **104**(1) (2015) 11–33.
14. 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.
15. Z. Wang, J. Zhang, J. Feng and Z. Chen, Knowledge graph embedding by translating on hyperplanes, in *Proc. 28th AAAI Conf. Artificial Intelligence*, 2014, pp. 1112–1119.
16. Z. Yan, R. Peng, Y. Wang and W. Li, Enhance knowledge graph embedding via fake triples, in *Proc. 2019 Int. Joint Conf. Neural Networks*, 2019, pp. 1–7.
17. J. Niu, Z. Sun and W. Zhang, Enhancing knowledge graph completion with positive unlabeled learning, in *Proc. 24th Int. Conf. Pattern Recognition*, 2018, pp. 296–301.
18. S. Qin, G. Rao, C. Bin, L. Chang, T. Gu and W. Xuan, Knowledge graph embedding based on adaptive negative sampling, in *Proc. Int. Conf. Pioneering Computer Scientists, Engineers and Educators*, 2019, pp. 551–563.
19. S. Dash and A. Gliozzo, Distributional negative sampling for knowledge base completion, arXiv:1908.06178.
20. T. Le, N. Le and B. Le, Knowledge graph embedding by relational rotation and complex convolution for link prediction, *Expert Syst. Appl.* **214** (2023) 119122.
21. F. Che and J. Tao, M2ixkg: Mixing for harder negative samples in knowledge graph, *Neural Networks* **1777** (2024) 106358.
22. K. Ahrabian, A. Feizi, Y. Salehi, W. L. Hamilton and A. J. Bose, Structure aware negative sampling in knowledge graphs, arXiv:2009.11355.
23. M. R. A. H. Rony, M. M. Alam, S. Ali, J. Lehmann and S. Vahdati, Lemon: Language model for negative sampling of knowledge graph embeddings, arXiv:2203.04703.
24. M. K. Islam, S. Aridhi and M. Smaïl-Tabbone, Simple negative sampling for link prediction in knowledge graphs, in *Proc. 10th Int. Conf. Complex Networks and Their Applications*, 2022, pp. 549–562.
25. K. Toutanova and D. Chen, Observed versus latent features for knowledge base and text inference, in *Proc. 3rd Workshop Continuous Vector Space Models and their Compositionality*, 2015, pp. 57–66.
26. N. Hubert, P. Monnin, A. Brun and D. Monticolo, Treat different negatives differently: Enriching loss functions with domain and range constraints for link prediction, in *Proc. European Semantic Web Conf.*, 2024, pp. 22–40.
27. C. d'Amato, N. F. Quatraro and N. Fanizzi, Injecting background knowledge into embedding models for predictive tasks on knowledge graphs, in *Proc. 18th Int. Conf. Semantic Web*, 2021, pp. 441–457.
28. S. Guo, Q. Wang, B. Wang, L. Wang and L. Guo, Semantically smooth knowledge graph embedding, in *Proc. 53rd Annual Meeting Association for Computational Linguistics and 7th Int. Joint Conf. Natural Language Processing*, 2015, pp. 84–94.
29. Z. Cui, P. Kapanipathi, K. Talamadupula, T. Gao and Q. Ji, Type-augmented relation prediction in knowledge graphs, in *Proc. AAAI Conf. Artificial Intelligence*, 2021, pp. 7151–7159.
30. Z. Li, X. Liu, X. Wang, P. Liu and Y. Shen, Transo: a knowledge-driven representation learning method with ontology information constraints, *World Wide Web* **26**(1) (2023) 297–319.

31. M. Nickel and V. Tresp, Tensor factorization for multi-relational learning, in *Proc. Joint European Conf. Machine Learning and Knowledge Discovery in Databases*, 2013, pp. 617–621.
32. B. Yang, W.-t. Yih, X. He, J. Gao and L. Deng, Embedding entities and relations for learning and inference in knowledge bases, arXiv:1412.6575.
33. L. A. Zadeh, Fuzzy sets, *Inf. Control* **8**(3) (1965) 338–353.
34. D. Dubois and H. Prade, The three semantics of fuzzy sets, *Fuzzy Sets Syst.* **90**(2) (1997) 141–150.
35. D. McGuinness and C. Welty, OWL web ontology language guide, W3C recommendation, W3C (2004), http://www.w3.org/TR/2004/REC-owl-guide-20040210/.
36. P. Hayes and P. Patel-Schneider, RDF 1.1 semantics, W3C recommendation, W3C (2014), https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/.
37. M. Weyns, P. Bonte, B. Steenwinckel, F. De Turck and F. Ongenae, Conditional constraints for knowledge graph embeddings, in *Proc. Workshop Deep Learning for Knowledge Graphs*, 2020.
38. R. Guha and D. Brickley, RDF schema 1.1, W3C recommendation, W3C (2014), http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.
39. H. Knublauch and D. Kontokostas, Shapes constraint language (SHACL), W3C recommendation, W3C (2017), https://www.w3.org/TR/2017/REC-shacl-20170720/.
40. G. Schreiber and M. Dean, OWL web ontology language reference, W3C recommendation, W3C (2004), http://www.w3.org/TR/2004/REC-owl-ref-20040210/.
41. J. Chen, X. Chen, I. Horrocks, E. B. Myklebust and E. Jimenez-Ruiz, Correcting knowledge base assertions, in *Proc. Web Conf.*, 2020, pp. 1537–1547.
42. A. Kristiadi, M. A. Khan, D. Lukovnikov, J. Lehmann and A. Fischer, Incorporating literals into knowledge graph embeddings, in *Proc. Int. Semantic Web Conf.*, 2019, pp. 347–363.
43. A. Banerjee, I. S. Dhillon, J. Ghosh, S. Sra and G. Ridgeway, Clustering on the unit hypersphere using von mises-fisher distributions, *J. Mach. Learn. Res.* **6**(9) (2005) 1345–1382.
44. P. Pezeshkpour, L. Chen and S. Singh, Embedding multimodal relational data for knowledge base completion, arXiv:1809.01341.
45. Q. Le and T. Mikolov, Distributed representations of sentences and documents, in *Proc. Int. Conf. Machine Learning*, 2014, pp. 1188–1196.
46. 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:1406.1078.
47. P. Ristoski and H. Paulheim, Rdf2vec: Rdf graph embeddings for data mining, in *Proc. Int. Semantic Web Conf.*, 2016, pp. 498–514.
48. F. Akrami, M. S. Saeef, Q. Zhang, W. Hu and C. Li, Realistic re-evaluation of knowledge graph completion methods: An experimental study, in *Proc. 2020 ACM SIGMOD Int. Conf. Management of Data*, 2020, pp. 1995–2010.
49. A. Rossi and A. Matinata, Knowledge graph embeddings: Are relation-learning models learning relations? in *EDBT/ICDT Workshops*, 2020.
50. A. Mohamed, S. Parambath, Z. Kaoudi and A. Aboulnaga, Popularity agnostic evaluation of knowledge graph embeddings, in *Proc. Conf. Uncertainty in Artificial Intelligence*, 2020, pp. 1059–1068.
51. A. Bordes, J. Weston, R. Collobert and Y. Bengio, Learning structured embeddings of knowledge bases, in *Proc. Twenty-Fifth AAAI Conf. Artificial Intelligence*, 2011, 301–306.