Entity Prediction in Knowledge Graphs with Joint Embeddings

Matthias Baumgartner
University of Zurich
baumgartner@ifi.uzh.ch

Daniele Dell'Aglio
University of Aalborg
University of Zurich
dade@cs.aau.dk

Abraham Bernstein
University of Zurich
bernstein@ifi.uzh.ch

Abstract

Knowledge Graphs (KGs) have become increasingly popular in the recent years. However, as knowledge constantly grows and changes, it is inevitable to extend existing KGs with entities that emerged or became relevant to the scope of the KG after its creation. Research on updating KGs typically relies on extracting named entities and relations from text. However, these approaches cannot infer entities or relations that were not explicitly stated. Alternatively, embedding models exploit implicit structural regularities to predict missing relations, but cannot predict missing entities. In this article, we introduce a novel method to enrich a KG with new entities given their textual description. Our method leverages joint embedding models, hence does not require entities or relations to be named explicitly. We show that our approach can identify new concepts in a document corpus and transfer them into the KG, and we find that the performance of our method improves substantially when extended with techniques from association rule mining, text mining, and active learning.

1 Introduction

Knowledge graphs (KGs) have gained popularity as a versatile, general-purpose, and domain-independent model to represent information and are the major backbone for many applications on the web (Noy et al., 2019). KGs express knowledge as collections of head-relation-tail statements, named *triples*, e.g. $(:Cheney, :vice\text{-}of, :Bush)$ expresses that Cheney is the vice president of Bush. Since KGs are mostly built through automatic processes (Carlson et al., 2010; Dong et al., 2014) they are often incomplete, e.g. a KG may contain the fact that Cheney was the vice-president of Bush, but not that Cheney is a US citizen. In addition, KGs evolve and require maintenance: they grow and change as the knowledge they describe expands and adapts to the real world.

The problem of deriving missing portions of knowledge is known as *KG completion*. So far, the problem has been tackled by *link prediction*, i.e. finding relationships between previously known entities in the graph. In this paper, we focus on the problem of *adding and integrating new entities into the KG*—a task we call *entity prediction*. This is different from link prediction, where entities are ex-ante partially described in the KG. In entity prediction, *we discover the existence of an entity from an external source and the KG neither contains the entity nor any information about how it relates to the other entities in the KG*.

As external source, we target document corpora, which describe the entities and the relations between them. For example, a document corpus like Wikipedia contains a description of Joe Biden and his relations with Obama (vice president) and Cheney (successor). This lead to our core research question: *given a KG $\mathcal{G}$ and a document corpus $\mathcal{D}$, how can we complete $\mathcal{G}$ with entities (textually) described in $\mathcal{D}$ but not yet contained in $\mathcal{G}$?*

As a solution, we represent the KG and document corpus in a common metric space and exploit this space in conjunction with user feedback and graph features to derive statements describing the new entities. Specifically, we leverage *joint embedding models* for creating a numerical space to represent the KG and the background source. Whereas KG embedding models give good performance on the link prediction task (Cai et al., 2018a), joint embedding models combine a KG with a document corpus to draw conclusions in terms of similarity between documents and KG entities. Our experiments determine that joint embedding models are well-founded methods to propose new entities to a KG. We also discuss how the prediction performance can be improved by integrating user feedback and explicit graph features.

The next section discusses related literature and introduces relevant notations. Section 4 outlines

Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15), pages 22–31

our solution based on joint embedding models, user feedback, and graph features. Section 5 describes the experimental setup and evaluates our hypotheses. Finally, Section 6 presents overall conclusions and outlines future work.

2 Related work

Like in our scenario, ontology population and ontology enrichment[1] extract information from documents (Petasis et al., 2011). Ontology population adds instances to an existing ontology, whereas the structure of the ontology remains unchanged. In contrast to our problem it does not need to learn relations between instances and assumes an ontology to guide the information extraction process (Buitelaar et al., 2006; Etzioni et al., 2004; Petasis et al., 2013). Ontology enrichment inserts new concepts or relations into an ontology. It differs from our setting in that it extends the schema of an ontology, using its concepts, instances, and schema constraints, while we solely rely on relationships between entities (Faure et al., 1998; Cimiano and Völker, 2005; Hahn and Marko, 2002).

KG enrichment aims at completing a given KG with new statements, or identifying erroneous ones (Paulheim, 2017), by predicting entity types (Nickel and Tresp, 2013; Socher et al., 2013) or links (Oren et al., 2007; Socher et al., 2013). In contrast, our goal is to complete a KG by adding new entities and statements related to them. (Paulheim, 2017) states that no such approach was known until 2015 and to the best of our knowledge, this has not changed meanwhile.

Automatic Knowledge Base Construction (AKBC) methods such as NELL or OpenIE approach a similar problem by means of text processing. They extract named entities and relations from a document, then arrange them as a KG (Verga and McCallum, 2016; Mitchell et al., 2018; Martínez-Rodríguez et al., 2018). Similarly, Entity Linking extracts named entities from text, then disambiguates and links them with a background database (Hoffart et al., 2014). These approaches assume that all entities and all relations are explicitly mentioned in the text under their canonical name. In contrast, we consider the scenario where entities are not stated in the text but described implicitly.

[1] In this study we do not distinguish between KG and ontology. We opt for ontology when it is used to refer to a known problem in literature, i.e. ontology population.

3 Background

This section presents the key concepts and notation used throughout the paper.

Knowledge graphs. We define a *knowledge graph* as $\mathcal{G} := (\mathcal{V}, \mathcal{R}, \mathcal{E})$, with $\mathcal{V}$ a set of vertices (entities), $\mathcal{R}$ a set of relations, and $\mathcal{E}$ a set of directed edges, also known as statements. A statement is a triple $(h, r, t) \in \mathcal{E}$, with $h, t \in \mathcal{V}$ the head/tail entities, and $r \in \mathcal{R}$ the relation. For example, the sentence "Joe Biden is the vice president of Barack Obama" is represented by the triple $(:Biden,\ :vice\text{-}of,\ :Obama)$. Let $\mathcal{U}$ be the the universe of the entities which can be described. $\mathcal{V}$ identifies the entities described in $\mathcal{G}$, and $\mathcal{V} \subset \mathcal{U}$, i.e., $\mathcal{G}$ does not include all possible entities that may be described, which is the case in real KGs.

KG embedding. Embedding models create a numeric, low-dimensional representation of a KG by learning a latent vector (embedding) for each KG entity and relation. Ideally, the distance between entity embeddings resembles the relatedness of the KG entities, e.g. the embeddings of $:Biden$ and $:Obama$ are close. Embedding models exploit structural regularities in the KG by defining an optimization problem in terms of a loss function that incorporates the graph's structure, as well as embeddings of a given size.

(Bordes et al., 2013) introduced the TransE embedding model, based on the idea that a relation r is a translation from the source entity h to the target entity t in the embedding space, i.e.:

$$\mathcal{L} \sim \sum_{(h,r,t)\in\mathcal{E}} \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_2 \qquad (1)$$

We denote embeddings in bold font and elements of $\mathcal{V} \cup \mathcal{R}$ in italic, e.g. $\mathbf{h}$ and $\mathbf{r}$ are the embeddings of $h \in \mathcal{V}$ and $r \in \mathcal{R}$, respectively.

While there exists a range of embedding models (Wang et al., 2017; Cai et al., 2018b), we focus on TransE due to its popularity and conceptual clarity.

Joint embedding models. Not only do embedding models exist for KGs but also for text (Mikolov et al., 2013; Le and Mikolov, 2014). Joint embedding models combine two embedding models for different modalities, allowing to compare embeddings between them while maintaining the characteristics of the individual models. These models make two principal assumptions. First, each document d from a cor-

pus $\mathcal{D}$ is a textual description of a single entity $e \in \mathcal{U}$, e.g. the Wikipedia document d_B (https://en.wikipedia.org/wiki/Joe_Biden) describes the entity e_B ($:Biden$). The description may be implicit, i.e. not actually mention the entity name, and can mention other entities. Second, the two modalities are linked to each other via known correspondences. We define the correspondences as a bijective function $m : \mathcal{D} \to \mathcal{U}$, and its inverse $m' : \mathcal{U} \to \mathcal{D}$, e.g. $m[d_B] = e_B$ and $m'[e_B] = d_B$.

Two joint embedding models are *KADE* and *StarSpace*. Both take a KG $\mathcal{G}$, a document corpus $\mathcal{D}$, and known correspondences m as input. They then create embeddings for each document, entity, and relation in a shared embedding space such that embeddings of a document and a corresponding entity are, i.e. $\mathbf{d} \sim \mathbf{e}$ if $m[d] = e$. *KADE* (Baumgartner et al., 2018) joins the TransE KG embedding model and the par2vec document embedding model (Le and Mikolov, 2014) by adding a regularizer term (weighted by λ) to both models, then training them in an alternating fashion. The regularization forces embeddings of corresponding documents and entities to be close to each other:

$$\mathcal{L}^{KADE}_{\text{Docs}} \sim \mathcal{L}_{\text{Docs}} + \lambda_d \sum_{d \in \mathcal{D}} \|\mathbf{d} - \mathbf{m}[d]\|$$

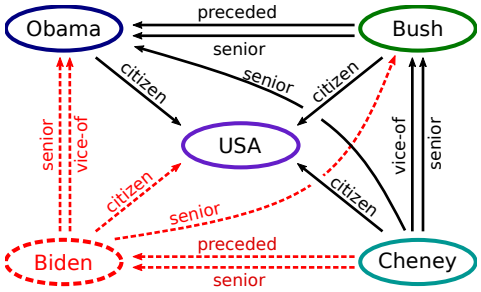
$$\mathcal{L}^{KADE}_{\text{KG}} \sim \mathcal{L}_{\text{KG}} + \lambda_g \sum_{e \in \mathcal{V}} \left\|\mathbf{e} - \mathbf{m}'[e]\right\|$$

StarSpace (Wu et al., 2018) models entities, relations, and words as atomic features, and defines objects as aggregates over these features. A document embedding thus becomes the sum of its words' embeddings. It then learns feature embeddings by minimizing the distance (inner product or cosine) between related objects:

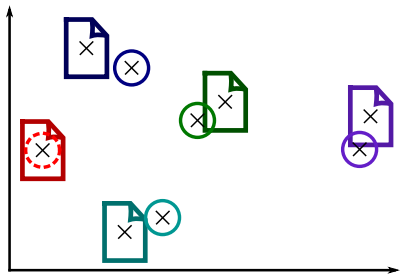
$$\mathcal{L}^{SS} \sim \sum_{(h,r,t) \in \mathcal{E}} dst(\mathbf{h}+\mathbf{r}, \mathbf{t}) + \sum_{e \in \mathcal{V}} dst(\mathbf{e}, \sum_{w \in m'[e]} \mathbf{w})$$

4 Approach

As inputs, our approach receives a KG $\mathcal{G}$, a document corpus $\mathcal{D}$, and correspondences m between the two modalities. While every entity has at least one corresponding document, we assume a number of *surplus documents* in $\mathcal{D}$ that are not associated with any entity in $\mathcal{G}$. A surplus document can either describe a novel entity we want to add to $\mathcal{G}$ or its association to an existing entity in $\mathcal{G}$ is unknown. The problem of entity prediction can then be divided into two subproblems:



(a) The goal of entity prediction is to add the entity $:Biden$ and all red edges to the KG.



(b) The entity embedding is inferred from the document embedding in a joint embedding space.

	$:Obama$	$:USA$	$:Bush$	$:Cheney$
$:citizen$	0.8	0.4	0.75	0.78
$:senior$	0.5	0.9	0.6	0.7
$:preceded$	0.6	0.98	0.7	0.65
$:vice\text{-}of$	0.1	0.8	0.3	0.5

(c) Triple plausibility is estimated via the KG embedding loss on joint embeddings (lower is better).

Figure 1: An example of entity prediction via a joint embedding model.

1. Identify whether a surplus document describes a novel entity. E.g. a document that describes the entity $:Biden$ which is not part of the KG in Figure 1a.

2. Add an entity e^* to $\mathcal{G}$ and propose edges between e^* and the graph's current entities $\mathcal{V}$. E.g. in Figure 1a, we would ideally add $:Biden$ and all red colored edges.

We address both problems in the next sections by describing how our method adds one entity to the KG. For multiple entities, we repeat the procedure for each one independently, and leave an approach that updates the KG and its embeddings incrementally as future work to study.

4.1 Novelty detection

We first discuss how to distinguish surplus documents that describe novel entities from those that have an unknown association to an entity in $\mathcal{G}$. We approach this task as a binary classification prob-

lem: a surplus document is either an alternative description of an entity in $\mathcal{G}$ or a novel entity. Our intuition is that documents that describe the same entity are more similar to each other than to the remaining documents in the corpus. Since joint embeddings preserve the characteristics of the the document embedding, we measure the document similarity via the embedding distance. Hence, we train the joint embedding model, then compute the distances between surplus document's embedding to the other document embeddings. We compute the mean, variance, minimum, maximum, percentiles, span, entropy, and skew of these distances, concatenate them with the surplus document's embedding, and use the resulting feature vector as input to a binary classifier.

4.2 Triple reconstruction

Next, we discuss how to derive new triples that have e^* as head or tail entity. This task is challenging because the number of possible triples is $2|\mathcal{V}||\mathcal{R}|$, which is orders of magnitude larger than the average number of triples an entity typically takes part in[2]. In the remainder of this section we describe the three components we use to tackle this challenge. First, we measure a potential triple's plausibility in a joint embedding space. Second, we propose a method to find likely relations with the help of user feedback. Third, we incorporate explicit features of the graph's structure into the previous methods. For the sake of brevity, we only discuss the case where e^* is in the head position.

Triple loss. We first look into the joint embedding space to retrieve the most plausible triples. Joint embedding models strive to produce the same embedding for a corresponding document and entity. Therefore, they suggest that the entity e^* is located at the same position in the embedding space as its corresponding document d, i.e. $\mathbf{e}^* := \mathbf{d}$. This is exemplified in Figure 1b: it shows the embeddings of all entities and documents, with corresponding items close to each other. Since $:Biden$ is missing from the graph, its embedding is proposed to be at the position of the document describing it.

KG embedding models define a triple loss $\mathcal{L}(h, r, t)$ that expresses the plausibility of a triple: *KADE* uses TransE's triple loss $\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_2$, *StarSpace* defines it as $dst(\mathbf{h} + \mathbf{r}, \mathbf{t})$. We compute the triple loss for every possible triple and collect them in a *loss matrix* $\mathbf{S}^{e^*} : \mathbb{R}^{|\mathcal{R}|\times|\mathcal{V}|}$, where each cell is defined as:

$$\mathbf{S}^{e^*}_{r,e} = \mathcal{L}(e^*, r, e) \tag{2}$$

Figure 1c presents an example triple loss matrix. For the sake of readability, we omit indices of $\mathbf{S}$ if possible, e.g. we use $\mathbf{S}^{e^*}_r$ to indicate the row $\mathbf{S}^{e^*}_{r,\cdot}$.

For the triple reconstruction, we are mostly interested in the ranking of losses — the triple with the lowest value in $\mathbf{S}$ is the most plausible one, irrespective of the actual value. We therefore rank triples in $\mathbf{S}$ in ascending order, i.e. assign the lowest rank to the triple that the embedding model determines to be the most plausible. Without further information, it is optimal to select the N lowest ranked triples in $\mathbf{S}^{e^*}$, which we denote as the *TopN* method.

User feedback. We refine the triple reconstruction from joint embedding models by incorporating additional information from a user's feedback. The main challenge of the triple reconstruction is that the number of true triples to restore is much lower than the number of possible triples. To circumvent this issue, we split the triple reconstruction into two subtasks: First, we identify relations $r \in \mathcal{R}$ present at e^*, then we identify the tail entities given the previously found relations. We propose to involve the user in the first subtask, then to solve the second one autonomously. This is because there are typically fewer relations than vertices in a KG, thus the user has to take fewer decisions while their feedback's impact is maximized.

We formalize this idea in the *UF* procedure in Algorithm 1. We employ a logistic classifier to distinguish relations that should be present at e^* from those that should not. The inputs to the classifier are the triple loss statistics of one relation r, i.e. the mean, median, variance, minimum, maximum, quantiles, entropy, and skew of $\mathbf{S}^{e^*}_r$. Its output is the likelihood of e^* having any triple with relation r. Out of the M most likely relations we then ask a user to select a correct one. For the chosen relation r we add the triples with the lowest ranks in $\mathbf{S}^{e^*}_r$. Since e^* can have multiple triples with the same relation to different entities, we pick the N_r lowest ranked ones, whereas N_r is the average number of r-triples (i.e. triples with relation r) at entities in $\mathcal{G}$. In addition, we discard triples that have a rank in $\mathbf{S}^{e^*}$ larger than a threshold θ. The process repeats until the user judges that no relation is valid.

One issue of *UF* is that the algorithm terminates without proposing any triple if the initial set of

[2] In *FB15k-237* the average entity only occurs in about 21 out of 302'455 possible triples.

Algorithm 1: *UF*: Iterative triple reconstruction with user feedback. The || symbol denotes list concatenation.

Input: Knowledge graph $\mathcal{G}$, proposed entity e^* and its loss matrix $\mathbf{S}^{e^*}$
Result: List of proposed triples $[(e^*, r, t)]$

```
/* Build relation features                          */
features ← [] ;
for r ∈ R do
    features(r) ← [avg(S_r^{e*}), var(S_r^{e*}), ...] ;
end
/* Predict initial candidate relations              */
candidates ← M highest scoring relations according to clf(features(r)) ;
feedback ← let the user select one relation from candidates ;
/* Select triples and iterate                       */
triples = [] ;
while feedback is valid do
    r ← feedback ;
    N_r ← mean number of r-triples on vertices having at least one r-triple ;
    s ← ⌈N_r⌉ lowest ranked triples in S_r^{e*} ;
    triples ← triples || items of s whose rank in S^{e*} is lower than θ;
    candidates ← M highest scoring relations according to clf(features(r)) ;
    feedback ← let the user select one true relation from candidates ;
end
return triples ;
```

relations suggested to the user lacks a valid one. To prevent this problem, we introduce *UF-s*, which keeps generating initial candidate relations (line 5 in Algorithm 1) until the user selects one of them, then continues in the same way as *UF*.

Graph features We further improve the triple reconstruction performance by exploiting the graph structure. In the following, we define two features and integrate them into the *UF* method.

The first feature focuses on improving the selection of a relation in *UF*. Once a user has selected a relation, we use this new evidence to improve the estimate of other relations' likelihoods. For this, we use the *confidence* measure (CO) from Association Rule Learning (Agrawal et al., 1993), which expresses how certain we are about an entity having a relation r_i if we know that it has r_j:

$$\text{conf}(r_j \Rightarrow r_i) = p(r_i|r_j) = \frac{|\{h|(h, r_j, \cdot) \in \mathcal{E}\} \cap \{h|(h, r_i, \cdot) \in \mathcal{E}\}|}{|\{h|(h, r_j, \cdot) \in \mathcal{E}\}|}$$

We integrate the confidence into *UF* by multiplying it with the respective likelihood predicted by the *clf* classifier. Note that this notion of confidence assumes that both r_i and r_j have the same direction, e.g. e^* in the head position. To incorporate the case where their direction differs (i.e. one is inbound, the other outbound to e^*), we modify Algorithm 1 to alternate between reconstructing triples with e^* in the head and tail position.

The second feature helps with finding the tail entities under a given relation. With no other information than the graph, it is reasonable to add an edge from e^* to the entity that is most frequently used with the given relation. This measure is especially informative if the relation occurs at few entities. To express these ideas, we use the BM25 weighting scheme (BM), popular in information retrieval (Robertson and Zaragoza, 2009). It assigns a large weight to an edge if the entity is likely to have the relation (term frequency) and if having that relation is also informative (document frequency). We integrate this feature into the *UF* method by dividing each value in $\mathbf{S}^{e^*}$ by its BM25 score.

Both of these features can be calculated in a single pass over the graph, i.e. they have complexity $\mathcal{O}(|\mathcal{E}|)$. Training an embedding model requires multiple iterations over the graph, making their complexity $\mathcal{O}(k|\mathcal{E}|)$ with k typically in the thousands. Therefore, calculating the graph statistics does not impact the scalability of the method.

5 Results

In this section, we first describe the experimental setup, then discuss the novelty detection, and finally show the triple reconstruction results.

5.1 Setup

We evaluate our methods on *FB15k-237* and *DBP50*, two popular KGs for KG embedding model benchmarking (Toutanova and Chen, 2015; Shi and Weninger, 2018). For entities from either KG, we select a random section of their respective Wikipedia article as corresponding document. To ensure that our methods do not learn from explicitly mentioned entity names, we replace all mentions of any of the entity label's words with 'entity'. For example, if the label is 'Joe Biden', we replace any occurrence of 'Joe' and 'Biden' with 'entity'. We then apply tokenization, normalization, and stopword removal on the documents. Finally, we remove entities from the graphs that cannot be associated with a unique document. Table 1 reports the resulting dataset sizes.

To test our methods, we randomly sample $k =$

	FB15k-237	DBP50
Entitites	14'375	24'005
Relations	236	351
Total Triples	300'423	33'486
Train triples	263'907	31'336
Unique words	90'824	66'745

Table 1: Dataset sizes

100 entities from each KG and remove them from the respective graph, leaving k surplus documents in both datasets. We then sample another k of the remaining entities in both KGs and add a second document to each of them, again randomly selected from their Wikipedia article and preprocessed in the same way as before. We omit these documents from the known correspondences m. This produces a total of $2 \cdot k$ surplus documents: For half of them no entity exists in the KG, the other half of them have existing yet unknown entities in the graph.

We then train *StarSpace* and *KADE* on the KGs and corpora. As our goal is not to get the best performance out of the embedding models, we use common parameters for these models: for *KADE* we set $\lambda_d = \lambda_g = 0.01$, for *StarSpace* we use the inner product. In both cases, we use embedding vectors of size 100 and we train for 1000 epochs.

5.2 Novelty detection results

We first discuss the results of novelty detection with joint embedding models. We hypothesize that joint embedding models have a higher accuracy in distinguishing novel from unassociated documents than other unsupervised document models.

Experiment 1: Novelty classification. To test the novelty detection, we train a boosting decision stumps classifier on the $2 \cdot k$ surplus documents, whereas half of them describe a novel entity, half of them describe an existing one. We compare *KADE* document embeddings to a bag-of-word document representation, and evaluate the classifier in a 10-fold cross-validation setting.

Table 2 shows the two classifiers' performances in terms of accuracy (overall correct classification), type-I (mistaken as novel) and type-II errors (mistaken as unassociated), precision, and recall. The bag-of-words model achieves near-random accuracy, while *KADE* embeddings achieve a substantially higher performance. The advantage of *KADE* is mostly in its lower type-I error rate, which prevents redundancy, i.e. that existing entities are being added a second time to the KG.

	FB15k-237		*DBP50*	
	KADE	BoW	*KADE*	BoW
Accuracy	**0.640**	0.505	**0.585**	0.515
Type-I error	**0.190**	0.285	**0.205**	0.275
Type-II error	**0.170**	0.210	**0.210**	**0.210**
Precision	**0.626**	0.501	**0.578**	0.511
Recall	**0.650**	0.590	0.573	**0.581**

Table 2: Performance of classifying documents as describing a novel or existing entity. Higher accuracy, precision, and recall are better, while lower type-I and type-II errors are better.

5.3 Triple reconstruction results

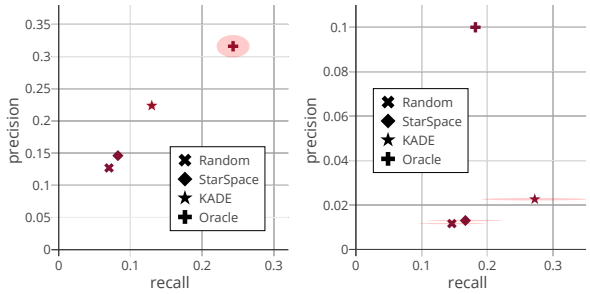
In the following, we compare the different triple reconstruction methods and their variations. First, we discuss the triple reconstruction considering only the embedding model's triple loss (*TopN*). Second, we investigate the impact of the separation into relation and triple prediction with user feedback (*UF*, *UF-s*). Third, we compare different combinations of graph features (BM, CO, or both) on their effect on Algorithm 1. Last, we discuss how much effort the different procedures inflict on the user.

We evaluate our methods on the binary classification metrics precision and recall. The precision indicates the portion of correct triples out of all proposed triples. The recall measures the portion of correctly proposed triples out of all correct triples. We apply our methods on the k novel entities individually and report the averaged metrics.

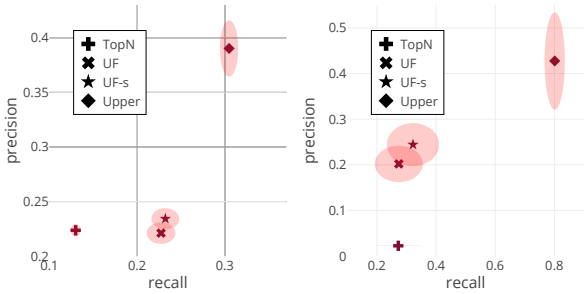
Experiment 2: Joint embedding model. In this experiment, we study how joint embedding models perform in the triple reconstruction task. Specifically, we compare the two joint embedding methods *StarSpace* and *KADE* in the *TopN* setting, and investigate how well the document embedding serves as embedding of the novel entity, as proposed by these models. To test the latter, we train a TransE model on the KG (without the k omitted entities) and derive the embedding of a novel entity e^* according to TransE's loss function, i.e.

$$\arg\min_{\mathbf{e}^*} \sum_{(e^*,r,t)} \|\mathbf{e}^* + \mathbf{r} - \mathbf{t}\| + \sum_{(h,r,e^*)} \|\mathbf{h} + \mathbf{r} - \mathbf{e}^*\|$$

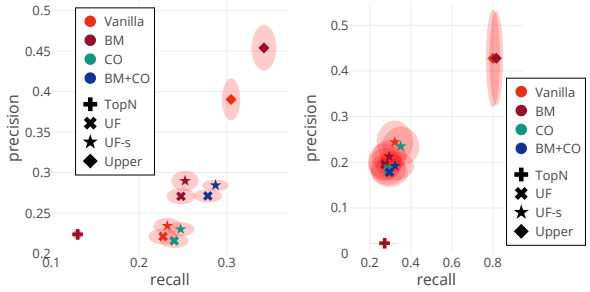
by using triples from the original KG. We denote this as *Oracle*, as it computes the optimal entity embedding from ground-truth data. As lower baseline, we use random embeddings (*Random*). For each entity embedding (from a baseline or joint embedding space), we select triples as specified by *TopN*. We set $N = 10$ which gave the best performance in our experiments. We hypothesize that the triple



(a) Entity restoration with joint embeddings.



(b) Impact of user feedback.



(c) Impact of user feedback and graph features combined. Marker shapes indicate the user feedback method, their color the graph features.

	Vanilla	BM	CO	BM+CO
FB15k-237 (upper baseline 103.658)				
UF	26.798	27.237	20.943	20.937
UF-s	29.002	29.329	23.799	23.800
DBP50 (upper baseline 80.562)				
UF	30.740	30.738	32.677	32.677
UF-s	64.037	64.037	66.429	66.429

(d) Number of user decisions per method and graph feature (lower is better).

Figure 2: Experimental results for *FB15k-237* (left plots) and *DBP50* (right plots). The red ellipses show the variance. Note that axes have different scales.

reconstruction performs substantially better with joint embeddings than the *Random* baseline.

Figure 2a shows the precision and recall of *TopN* with embeddings from the different models. It shows that *KADE* performs substantially better than *StarSpace* in both metrics and datasets, meaning that the more constrained document model used by *KADE* is advantageous in the entity prediction task. As expected, the performance of *StarSpace* and *KADE* lies between the two baselines, however compared to the *Oracle* baseline their performance is unsatisfactory, motivating further improvements.

Experiment 3: User feedback. Next, we hypothesise that user feedback increases precision and recall in the triple reconstruction task. For these experiments, we use *KADE* embeddings, $\theta = 500$, and $M = 10$. Since a user study would exceed the scope of this experiment, we provide user feedback by automatically selecting one random correct relation out of the suggested ones. Out of the k entities omitted from the KG, one is selected as e^*. The other omitted entities are used to train the classifier *clf* by constructing features from their triple loss matrices as stated in Section 4.2 and using their triples from the ground-truth KG as training targets. We repeat this procedure for all k entities, and report the averaged evaluation metrics.

Figure 2b contrasts the *UF* and *UF-s* methods with *TopN* and an upper bound *Upper*. The *Upper* baseline knows all true relations of e^*, then picks the lowest ranked N triples for each of them, according to $\mathbf{S}^{e^*}$. Varying N shifts the trade-off between precision and recall, whereas we use the N that maximizes the F1 score (i.e. the harmonic mean of precision and recall). Note that like *Oracle*, this baseline is not practically viable as it uses ground-truth information, but rather indicates the maximum performance that could be achieved given the triple loss of the joint embeddings.

Figure 2b shows that that the user feedback has a positive impact on at least one of the two evaluation metrics, and that *UF-s* improves over *UF* with an average increase of 13.43% in precision and 9.83% in recall. The latter is expected, since *UF-s* has a guaranteed initial relation. In *FB15k-237*, the user feedback improves recall by 76.95%, implying that the classifier learns relations present at e^*. While more relations are considered, the ratio of correctly selected triples does not improve with *UF*, meaning that the ranking of triples within one relation is about as accurate as the rankings in the full triple loss matrix. In *DBP50*, the precision increases much more than the recall. In contrast to *FB15k-237*, this dataset is sparser (about 2.5 triples per

vertex), which makes it harder to find relations. However, once a relation is known, the triple scores from the joint embedding model are reliable, i.e. the true triples have low ranks and are thus selected.

Experiment 4: Graph statistics. As the last part of our triple reconstruction method we evaluate the combination of user feedback and graph features. We first hypothesize that combining *UF* and *UF-s* with either BM or CO increases their precision and recall. Our second hypothesis is that the combination of BM and CO yields an improvement over having only one of them. For this experiment, we use the same setup as in the previous one and apply the BM25 parameters $b = 0.75$ and $k_1 = 2.0$. Figure 2c shows the use of the different graph features in *UF*, *UF-s*, and the *Upper* baseline, and includes *TopN* as lower baseline. Vanilla indicates no graph features were in effect. Note that the *Upper* baseline is not affected by the CO feature as it does not select relations iteratively.

On *FB15k-237*, the BM feature improves the precision by 23.05%, while the CO feature increases the recall by 6.10% (at the cost of slightly lowering the precision). The combination of both features ($BM + CO$) amplifies these effects, with an improvement of 23.11% in recall and 22.02% in precision. These observations relate to how the features affect the different parts of Algorithm 1: CO helps in finding more relations, hence increases the recall; BM increases the number of retrieved true triples of a given relation, hence increases the precision. These effects are greater on *UF-s* than on *UF* because the user provides at least one relation, which in turn allows the graph features to become more effective. On *DBP50*, the graph features have no significant effect in our methods nor the *Upper* baseline; instead, the variance is larger than the difference between the methods. We attribute this to the sparsity of the dataset, since it provides too few samples to estimate frequencies accurately and small differences in the triple selection have a huge impact on the precision and recall.

Experiment 5: User involvement. To evaluate the user workload, we measure how many relations a user has to judge during the triple reconstruction task, assuming that the user reports the first valid relation they find in each iteration.

Figure 2d shows the number of judgements of *UF*, *UF-s*, their variations, and an upper baseline. The upper baseline expresses the case where at most one relation is valid in each iteration. The lower baseline is $2M = 20$ since the user has to review all presented relations at least once.

It is apparent that *UF-s* involves more judgements than *UF*, which comes from two factors: A higher effort to find an initial relation, and more subsequent iterations. We further observe that the number of judgements is substantially lower than the upper baseline in *FB15k-237*, meaning that it finds multiple valid relations per iteration. On the other hand, it is more difficult to find a valid relation in *DBP50*, hence the user workload is higher, especially in *UF-s*. Graph features generally show a positive impact in *FB15k-237* and a marginal negative effect in *DBP50*, in particular the CO feature as it affects which relations are shown to the user.

6 Conclusion and Future Work

In this paper, we studied the problem of integrating new entities into a KG given their textual description. We exploited joint embeddings to identify entity candidates, and combined information from the joint embedding model with user feedback and graph features to improve the triple reconstruction. Our method solely relies on structural patterns in the data and does not need explicit mentions of entities or relations in the text. Our experiments suggest that joint embeddings are viable methods for entity prediction, and confirm that user feedback and graph features have a substantial impact on the triple reconstruction. In particular, experiments indicate that user feedback, features on relations (CO), and features on entities (BM) treat different aspects of the problem, making their combination more successful than using only one of them.

Comparing the results with the upper baselines shows that there is room for improvement. A possible way to fill this gap is to integrate explicit information into the process, e.g. considering the schema or the semantics of the relations or entities. Another approach is to study the incremental addition of new entities and triples: We restore entities and triples independently of each other, however, the restoration provides new information that can be exploited subsequently. Finally, our method could be extended in a straight-forward manner to other external data sources such as images, or to predict novel relations instead of entities.

Acknowledgements. We thank the Swiss National Science Foundation for their partial support under contract number #407550_167177 and Ralph Bobrik from Swiss Re for his helpful insights.

References

Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. 1993. Mining association rules between sets of items in large databases. In *SIGMOD Conference*.

Matthias Baumgartner, Wen Zhang, Bibek Paudel, Daniele Dell'Aglio, Huajun Chen, and Abraham Bernstein. 2018. Aligning knowledge base and document embedding models using regularized multi-task learning. In *International Semantic Web Conference (1)*, volume 11136 of *Lecture Notes in Computer Science*, pages 21–37. Springer.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.

Paul Buitelaar, Philipp Cimiano, Stefania Racioppa, and Melanie Siegel. 2006. Ontology-based information extraction with SOBA. In *LREC*, pages 2321–2324. European Language Resources Association (ELRA).

HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. 2018a. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Trans. Knowl. Data Eng.*, 30(9):1616–1637.

HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. 2018b. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.*, 30(9):1616–1637.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr, and Tom M. Mitchell. 2010. Toward an Architecture for Never-Ending Language Learning. In *AAAI*. AAAI Press.

Philipp Cimiano and Johanna Völker. 2005. Text2onto. In *NLDB*, volume 3513 of *Lecture Notes in Computer Science*, pages 227–238. Springer.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *KDD*, pages 601–610. ACM.

Oren Etzioni, Michael J. Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In *WWW*, pages 100–110. ACM.

David Faure, Claire Nédellec, and Céline Rouveirol. 1998. Acquisition of semantic knowledge using machine learning methods: The system" asium". In *Universite Paris Sud*. Citeseer.

Udo Hahn and Kornl G Marko. 2002. Ontology and lexicon evolution by text understanding. In *Proceedings of the ECAI 2002 Workshop on Machine Learning and Natural Language Processing for Ontology Engineering (OLT 2002), Lyon, France*.

Johannes Hoffart, Yasemin Altun, and Gerhard Weikum. 2014. Discovering emerging entities with ambiguous names. In *WWW*, pages 385–396. ACM.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1188–1196. JMLR.org.

José-Lázaro Martínez-Rodríguez, Ivan López-Arévalo, and Ana B. Rios-Alvarado. 2018. Openie-based approach for knowledge graph construction from text. *Expert Syst. Appl.*, 113:339–355.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR (Workshop)*.

Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha P. Talukdar, Bo Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matt Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil A. Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2018. Never-ending learning. *Commun. ACM*, 61(5):103–115.

Maximilian Nickel and Volker Tresp. 2013. Tensor factorization for multi-relational learning. In *ECML/PKDD (3)*, volume 8190 of *Lecture Notes in Computer Science*, pages 617–621. Springer.

Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. 2019. Industry-scale knowledge graphs: Lessons and challenges. *Commun. ACM*, 62(8):36–43.

Eyal Oren, Sebastian Gerke, and Stefan Decker. 2007. Simple algorithms for predicate suggestions using similarity and co-occurrence. In *ESWC*, volume 4519 of *Lecture Notes in Computer Science*, pages 160–174. Springer.

Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508.

Georgios Petasis, Vangelis Karkaletsis, Georgios Paliouras, Anastasia Krithara, and Elias Zavitsanos. 2011. Ontology population and enrichment: State of the art. In *Knowledge-Driven Multimedia Information Extraction and Ontology Evolution*, volume 6050 of *Lecture Notes in Computer Science*, pages 134–166. Springer.

Georgios Petasis, Ralf Möller, and Vangelis Karkaletsis. 2013. BOEMIE: reasoning-based information extraction. In *NLPAR@LPNMR*, volume 1044 of *CEUR Workshop Proceedings*, pages 60–75. CEUR-WS.org.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Baoxu Shi and Tim Weninger. 2018. Open-world knowledge graph completion. In *AAAI*, pages 1957–1964. AAAI Press.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China. Association for Computational Linguistics.

Patrick Verga and Andrew McCallum. 2016. Row-less universal schema. In *AKBC@NAACL-HLT*, pages 63–68. The Association for Computer Linguistics.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.*, 29(12):2724–2743.

Ledell Yu Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. 2018. Starspace: Embed all the things! In *AAAI*, pages 5569–5577. AAAI Press.