Unsupervised Embedding Enhancements of Knowledge Graphs using Textual Associations

Neil Veira[1*], **Brian Keng**[2], **Kanchana Padmanabhan**[2] and **Andreas Veneris**[1]
[1]Department of Electrical and Computer Engineering, University of Toronto
[2]Data Science, Rubikloud Technologies Inc.
nveira@eecg.toronto.edu, brian.keng@rubikloud.com, kanchana.padmanabhan@rubikloud.com, veneris@eecg.toronto.edu,

Abstract

Knowledge graph embeddings are instrumental for representing and learning from multi-relational data, with recent embedding models showing high effectiveness for inferring new facts from existing databases. However, such precisely structured data is usually limited in quantity and in scope. Therefore, to fully optimize the embeddings it is important to also consider more widely available sources of information such as text. This paper describes an unsupervised approach to incorporate textual information by augmenting entity embeddings with embeddings of associated words. The approach does not modify the optimization objective for the knowledge graph embedding, which allows it to be integrated with existing embedding models. Two distinct forms of textual data are considered, with different embedding enhancements proposed for each case. In the first case, each entity has an associated text document that describes it. In the second case, a text document is not available, and instead entities occur as words or phrases in an unstructured corpus of text fragments. Experiments show that both methods can offer improvement on the link prediction task when applied to many different knowledge graph embedding models.

1 Introduction

Multi-relational data plays an increasingly crucial role in many areas of artificial intelligence, including social network analysis, information retrieval, and question answering. In this context, a multi-relational knowledge base is a collection of facts, which are represented as triplets containing a pair of entities (subject and object) and a relationship (predicate) that they share. Several popular projects have demonstrated the power of representing relational knowledge in a graph structure, such as YAGO [Suchanek *et al.*, 2007], DBpedia [Auer *et al.*, 2007], and Freebase [Bollacker *et al.*, 2008]. These knowledge graphs contain millions of nodes and edges corresponding to entities and relations, and can be used to reason and infer new facts about the world.

Many approaches to relational learning have been studied, with those based on knowledge graph embeddings demonstrating particular success in both performance and scalability. These approaches learn continuous latent feature representations (embeddings) of the entities and relations, which can be used to compute a score for any potential fact. New facts can then be predicted based on relative scores.

Recent work on embedding techniques has led to significant improvements in prediction accuracy [Bordes *et al.*, 2011; Bordes *et al.*, 2013; Lin *et al.*, 2015; Nickel *et al.*, 2011; Yang *et al.*, 2014; Nickel *et al.*, 2016b; Socher *et al.*, 2013; Nickel *et al.*, 2016a; Wang *et al.*, 2017]. However, such methods rely exclusively on known facts from the knowledge graph to learn an embedding. As a result, their predictive ability is fundamentally limited by the information stored explicitly or implicitly in the existing database, which is often far from complete.

A promising avenue for improvement is to incorporate additional information from text documents such as Wikipedia or news articles. Such documents are widely available and often contain mentions of entities and relationships which can be indicative of new facts. Relation extraction aims to make such facts explicit in order to build a knowledge graph or extend an existing one [Mintz *et al.*, 2009; Riedel *et al.*, 2010; Surdeanu *et al.*, 2012; Lao *et al.*, 2012; Riedel *et al.*, 2013; Weston *et al.*, 2013]. These techniques typically only consider textual relations that are explicitly mentioned in conjunction with a pair of entities. Yet text documents often contain more information than just mentions of (entity, relation, entity) triplets. For instance, they may also contain entity attributes which could indirectly indicate various forms of semantic similarity between entities.

Due to the wide variety of data formats that can appear in natural language, such information is often best captured by embedding representations of objects in the text. The information can be incorporated into the knowledge graph by enhancing its embedding with textual embeddings rather than by adding discrete triplets. One class of methods accomplishes this by optimizing an objective function which combines entity and text objects to learn joint embeddings [Wang *et al.*, 2014; Toutanova *et al.*, 2015]. Another approach proposed by [Socher *et al.*, 2013] uses a simple average of word vectors to model the corresponding entity embedding. These methods show improved performance over pure knowledge

[*]Work done as an intern at Rubikloud

graph embeddings, however, they do not handle all the different cases of associated text to entity nor utilize the text fully in each situation.

In this work we consider two different forms of text data that may be associated with entities in a knowledge graph and propose methods which can augment existing knowledge graph embedding models with each type of data. We first consider the case in which a text description or document is associated with each entity. For instance, descriptions of many real-world entities and words can be readily obtained from Wikipedia or a dictionary. We propose a model for this scenario which uses a relation-specific weighted mean of word vectors to represent a particular entity.

In the second case, the entities in the knowledge graph do not have an associated document but rather are mentioned in a corpus at various points. For this case we propose a different model which first obtains representations of entities by training word embeddings [Mikolov *et al.*, 2013] on the corpus. The resulting word vectors provide semantic descriptions of the entities based on co-occurrences of entity names with other words in the text. These vectors are then combined with entity embeddings to enhance the representation.

Both methods retain the same optimization objective as traditional knowledge graph embedding techniques. This allows the methods to leverage the ability of these techniques to infer new facts from existing ones, while simultaneously supplementing them with the rich entity information contained in the text. It also allows our enhancements to be seamlessly integrated with any graph embedding model. As such, we empirically show performance improvement on knowledge completion tasks with six different models. This improvement is obtained in a fully unsupervised manner by simply applying textual data to the embedding procedure. Because our approach does not require any structure or labels within the text, such data can be collected and applied at a large scale with minimal effort.

2 Background and Related Work

Knowledge graphs express knowledge as a collection of facts, where each fact is a triplet consisting of a *subject*, a *predicate*, and an *object*. Subjects and objects are represented as nodes, collectively referred to as *entities*, and predicates are represented as labeled directed edges, referred to as *relations*. Thus a fact (e_s, r_p, e_o) is represented by an edge of type r_p from node e_s to e_o.

The task of predicting new facts is usually best accomplished when formulated as a ranking problem. Formally, a scoring function $f : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \mathbb{R}$ is defined, where $\mathcal{E}$ denotes the set of entities and $\mathcal{R}$ denotes the set of relations. Given a subject e_s and relation r_p, the score of correct objects e_o should be greater than that of the incorrect ones: $f(e_s, r_p, e_o) > f(e_s, r_p, e'_o)$ for all $e'_o \neq e_o$. f is typically defined in terms of continuous latent feature representations (embeddings) of the entities and relations, which can be learned via gradient descent on a ranking loss:

$$\mathcal{L} = \sum_{i=1}^{n} \max\left(0, \gamma - f(e_{s_i}, r_{p_i}, e_{o_i}) + f(e_{s'_i}, r_{p_i}, e_{o'_i})\right) \tag{1}$$

Model	Scoring function $f(e_s, r_p, e_o)$
SE	$-\left\|\mathbf{e}_s\mathbf{R}_p^{(1)} - \mathbf{e}_o\mathbf{R}_p^{(2)}\right\|_1$
TRANSE	$-\left\|\mathbf{e}_s + \mathbf{r}_p - \mathbf{e}_o\right\|_1$
TRANSR	$-\left\|\mathbf{e}_s\mathbf{R}_p + \mathbf{r}_p - \mathbf{e}_o\mathbf{R}_p\right\|_1$
RESCAL	$\mathbf{e}_s\mathbf{R}_p\mathbf{e}_o$
DISTMULT	$(\mathbf{e}_s \odot \mathbf{r}_p)\mathbf{e_o}^T$
HOLE	$\mathbf{r}_p^T(\mathbf{e}_s \star \mathbf{e}_o)$

Table 1: Summary of popular triplet scoring functions. $\odot$ denotes element-wise multiplication, and $\star$ denotes circular convolution.

where i denotes the index of a triplet in the training set, $(e_{s'_i}, r_{p_i}, e_{o'_i})$ is a negative (untrue) sample corresponding to $(e_{s_i}, r_{p_i}, e_{o_i})$, and γ denotes the margin hyperparameter.

Many different knowledge graph embedding models have been proposed, differing primarily in the definition of the scoring function. In this work we consider six of the most significant and fundamental ones, namely SE [Bordes *et al.*, 2011], TRANSE [Bordes *et al.*, 2013], TRANSR [Lin *et al.*, 2015], RESCAL [Nickel *et al.*, 2011], DISTMULT [Yang *et al.*, 2014], and HOLE [Nickel *et al.*, 2016b]. The scoring functions used by these models are summarized in Table 1. These models were selected for their simplicity, cost–effectiveness, and proven performance through extensive evaluation in previously published work. This makes them ideal for evaluating the effect of our enhancement methodology.

Throughout this paper we use the following notational conventions. Objects such as entities, relations, and words are denoted by lowercase letters (such as e_i, r_j, w_k), while their corresponding vector representations (*i.e.* embeddings) are denoted by the same letters in boldface font (e.g. $\mathbf{e}_i, \mathbf{r}_j, \mathbf{w}_k$). Tensors of rank two or more are denoted by uppercase letters (e.g. $\mathbf{E}$). A single subscript i denotes the i^{th} slice of the tensor (e.g. $\mathbf{E}_i$), while a double subscript ij denotes the j^{th} element of the i^{th} slice (e.g. E_{ij}).

2.1 Combining Knowledge Graphs and Text

A large body of work exists that combines information from both knowledge graphs and textual sources. *Relation extraction* aims to extend an existing knowledge graph by identifying new triplets mentioned in an associated text document. Most approaches to relation extraction require distant supervision for training purposes, where each existing fact in the database is tagged with its mentions in the text [Surdeanu *et al.*, 2012; Lao *et al.*, 2012; Riedel *et al.*, 2013; Weston *et al.*, 2013]. Heuristic techniques are typically used to automatically align the existing knowledge graph and the text in this way. However, this can lead to highly noisy results because the co-occurrence of two entities in a sentence does not necessarily imply that the sentence is stating a relation between them. Therefore, methods have also been proposed that reduce the amount of labeled data required [Mintz *et al.*, 2009; Riedel *et al.*, 2010].

The use of text embeddings in relation extraction has also been explored. [Gardner *et al.*, 2013] learns continuous latent feature representations of verbs parsed from text in order to reduce the number of relation types, while [Gardner *et al.*, 2014] uses them to define semantic similarity between parsed verbs for guiding random walks. This work is largely complementary to ours; while it performs knowledge completion by adding new explicit facts to a knowledge graph, our approach aims to improve the graph's predictive ability by enhancing its embedding.

Along this line, [Toutanova *et al.*, 2015] incorporates textual information by adding new triplets and new relation types extracted from text during training. In this way, embeddings of textual relations are learned along with embeddings of the original knowledge graph. [Wang *et al.*, 2014] jointly learns embeddings of words and entities by simultaneously maximizing the likelihoods of triplets occurring in the graph and word–word and word–entity pairs co-occurring in the text. [Socher *et al.*, 2013] proposes a different approach to joint embeddings based on the observation that entity names are often composed of multiple words, and entities whose names have common words are more likely to be related and should therefore have more similar representations. This is achieved by defining each entity vector to be the mean vector of all words in the entity's name. However, this method will assign equal significance to all words regardless of the particular entity or relation.

3 Embedding Enhancement Methodology

In this section we discuss new approaches to incorporate textual data into knowledge graph embeddings. This additional information allows the training procedure to learn entity representations that simultaneously reflect facts from the knowledge base and associated text. We consider two distinct scenarios depending on the form of the available textual data. In the first scenario, each entity has a document associated with it which describes or defines the entity; for example, the Wikipedia entry for *Europe*. Such data may be obtained from numerous sources including an encyclopedia or a dictionary. In the second scenario, we consider an unstructured corpus that is not directly linked to any entity, but contains mentions of entities at arbitrary locations. For example, a news article which mentions *Europe* may be a part of this corpus. No assumptions are made about the organizational structure of this text, so it can generally be a collection of sentences gathered from multiple documents.

The key distinction between these two forms of data is that in the former, entities are the underlying topics of all words in a document, whereas in the latter, entities are merely objects mentioned in documents containing a mixture of unknown topics. As a result, the first case contains the additional information that each word pertains to a known entity in some way. For the second case, we only assume associations exist between words that occur in the same context. This distinction is illustrated in Figure 1.

3.1 Embedding Model for Entity Descriptions

In this section we present a model for the first scenario in Figure 1, where textual data is available as entity descriptions. Our approach builds upon the WordVectors model of [Socher *et al.*, 2013], which defines entity vectors as the mean of word vectors in the entity names. First, we observe that the idea can also be applied to entity descriptions so as to force entity embeddings to share common textual features such as attribute or relationship words. This results in more similar vectors for more semantically similar entities. We then improve upon this model by adding new parameters to control how strongly each word contributes to the composition of an entity for a given relation.

We begin with a formalization of the WordVectors model which we have adapted to the case of entity descriptions. Let $\text{text}(e_i) = w_{i,1}, w_{i,2}, \ldots$ be the sequence of words associated with entity e_i. Let $\mathbf{W}$ denote an $n_w \times d$ matrix of word vectors, where n_w is the number of words in the vocabulary and d is the embedding dimensionality. Let $\mathbf{A}_i$ denote an n_w-dimensional vector such that A_{ik} is number of times word w_k appears in $\text{text}(e_i)$. Then the embedding vector for e_i can be expressed as

$$\mathbf{e}_i = \frac{\mathbf{A}_i \mathbf{W}}{\|\mathbf{A}_i\|_1} \tag{2}$$

A limitation of Eq. 2 is that all words in a description are treated equally; usually it is the case that certain words are much more relevant for predicting a relation than others. For instance, the words *instrument*, *drum* and *career* are likely much more indicative of the *musical-group-membership* relation than other words such as *actress* or *saturday*. Therefore, an entity should be represented by the words *instrument, drum* and *career* when predicting the *musical-group-membership* relation.

On the other hand, when predicting a different relation such as *education*, words such as *canadian, curriculum* or *ivy* are likely to be much more relevant, and so the entity should be represented more strongly by these words. In this way, the vectors of entities containing the word *ivy* will be much more similar for the *education* relation than for *musical-group-membership*. Such a model could therefore predict that two entities share the former relation but not the latter.

This behaviour can be achieved by introducing an $n_r \times n_w$ matrix $\mathbf{B}$ such that B_{jk} represents the significance of word w_k in predicting relation r_j. We can then define the representation of entity e_i under relation r_j as

$$\mathbf{e}_i^{(r_j)} = \frac{(\mathbf{A}_i \odot \mathbf{B}_j)\mathbf{W}}{\|\mathbf{A}_i \odot \mathbf{B}_j\|_1} \tag{3}$$

where $\odot$ denotes element-wise multiplication. In this way the overall weight of word w_k to the entity vector is a combination of the frequency of w_k in $\text{text}(e_i)$ (*i.e.* A_{ik}) and the relevance of w_k for r_i (*i.e.* B_{jk}).

However, the significance of each word for predicting each relation is generally not known. We therefore initialize $\mathbf{B}$ with $B_{ij} = 1$ for all i, j and learn these parameters via gradient descent. As demonstrated in section 4, this procedure is able to automatically learn associations of words to different relations without any supervision.

We can incorporate the textual information into any knowledge graph embedding model by using Eq. 3 in place of $\mathbf{e}_i$ in Table 1. For instance, the augmented TRANSE model is $f(e_s, r_p, e_o) = -\left\|\mathbf{e}_s^{(r_p)} - \mathbf{r}_p + \mathbf{e}_o^{(r_p)}\right\|_1$. We refer to this method as weighted word vectors (WWV).

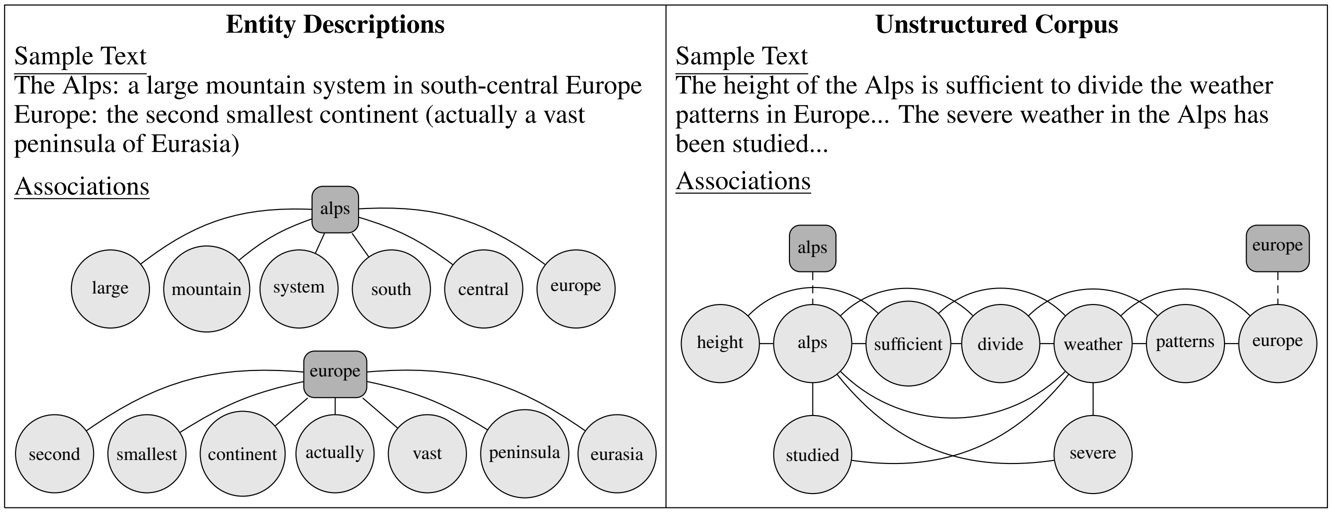
Entity Descriptions
Sample Text
The Alps: a large mountain system in south-central Europe
Europe: the second smallest continent (actually a vast peninsula of Eurasia)
Associations
alps
large
mountain
system
south
central
europe
europe
second
smallest
continent
actually
vast
peninsula
eurasia
Unstructured Corpus
Sample Text
The height of the Alps is sufficient to divide the weather patterns in Europe... The severe weather in the Alps has been studied...
Associations
alps
europe
height
alps
sufficient
divide
weather
patterns
europe
studied
severe

Figure 1: Example of text structures and associations between words and entities after preprocessing. For entity descriptions we create an association between the entity and each word in its description. Entities are shown in boxes while words are shown in circles. For the unstructured corpus associations are assumed by the word2vec algorithm (with a context window of width 2 for this example).

3.2 A Parameter–Efficient Weighting Scheme

A potential shortcoming of the WWV model as described in the previous section is that the number of parameters in the matrix $\mathbf{B}$ is $n_r \times n_w$, which may be prohibitively large for some datasets. This can be improved by allowing B_{ij} to be derived from a smaller number of parameters instead of defining each as an independent parameter. Towards this end, we introduce an $n_r \times d$ matrix $\mathbf{P}$ and define the weight for relation r_i and word w_j as follows:

$$B_{ij} = \frac{\exp\left(\mathbf{P}_i \mathbf{W}_j^T\right)}{\sum_{k=1}^{n_w} \exp\left(\mathbf{P}_i \mathbf{W}_k^T\right)} \tag{4}$$

The intuition behind Eq. 4 is that $\mathbf{P}_i$ is a representation of relation r_i in *word feature space* — the same feature space as the word vector $\mathbf{W}_j$. Because $\mathbf{P}_i$ and $\mathbf{W}_j$ use the same features, $\mathbf{P}_i \mathbf{W}_j^T$ is a measure of the similarity between r_i and w_j, which serves as the weight between them. For example, we might expect the vector $\mathbf{P}_i$ for relation $r_i =$ *musical-group-membership* to be similar to the vector $\mathbf{W}_j$ for $w_j = instrument$ because the concepts of musical groups and instruments are semantically related.

While it is conceptually appealing to define the weight B_{ij} using a softmax function, in practice the normalization factor is not needed because all weights are normalized again in Eq. 3. Therefore, we can express the representation for entity e_i under relation r_j as:

$$\mathbf{e}_i^{(r_j)} = \frac{\sum\limits_{w_k \in \text{text}(e_i)} A_{ij} \exp(\mathbf{P}_j \mathbf{W}_k^T) \mathbf{W}_k}{\sum\limits_{w_k \in \text{text}(e_i)} A_{ij} \exp(\mathbf{P}_j \mathbf{W}_k^T)} \tag{5}$$

where we have expressed the vector–matrix multiplication of Eq. 3 in expanded form to more clearly show the weighted averaging of words. The number of trainable parameters is thus reduced from $n_r \times n_w$ to $n_r \times d$. We refer to the model in Eq. 5 as parameter–efficient weighted word vectors (PE-WWV). Despite having many fewer parameters, Section 4 demonstrates that PE-WWV performs comparably to WWV in prediction accuracy.

3.3 Training Procedure

A challenge arises in training the WWV and PE-WWV models due to the fact that both the word embedding parameters $\mathbf{W}$ and the weights $\mathbf{B}$ must be learned simultaneously with no supervision. That is, the optimizer must discover the most important words for each relation without being given any understanding or characterization of the relations or words. Due to initial randomness, the optimizer may over-emphasize irrelevant words in the early stages of training and then never find a good solution.

We find that this issue can be greatly alleviated by holding the word weights ($\mathbf{B}$ and $\mathbf{P}$) constant for the first 50 training epochs. This allows the optimizer to first learn semantically meaningful word representations without disruption from varying word weights. Then, for the remainder of the training period, we optimize all parameters and are able to discover the most relevant words.

3.4 Embedding Model for Unstructured Corpus

In this section we consider the second scenario in Figure 1. To capture information from unstructured data we train the word2vec model [Mikolov *et al.*, 2013] on the given corpus to learn embedding vectors of words. Word2vec is trained to assign similar vectors to words that commonly appear in the same context, which makes it well suited for learning entity vectors. For example, the sentence fragment *Brian Jones and fellow guitarist Keith Richards developed a unique...* clearly states a relation between *Brian Jones* and *Keith Richards*. Because the entities *Brian Jones* and *Keith Richards* appear as words within the same context, the word2vec vectors of these entities will be more similar.

The word2vec vectors can also capture features which appear as attributes rather than objects of the sentence. In the example above, the sentence also indicates an association between *Brian Jones* and *guitar*. This can be a strong hint for predicting other types of relations for *Brian Jones*, such as instruments played or musical group membership. When given this training sentence, word2vec learns to encode this information implicitly in the vector for *Brian Jones*. This makes it possible to incorporate the information into the entity embeddings by augmenting them with the word2vec feature vectors.

With this intuition the overall model works as follows. Let $\mathbf{w}_i$ denote the word2vec vector for the name of entity e_i, and let $\mathbf{e}_i$ denote the entity vector. We define an augmented vector for entity e_i as:

$$\hat{\mathbf{e}}_i = \mathbf{e}_i + \mathbf{w}_i\mathbf{M} \tag{6}$$

Thus, each latent feature in $\hat{\mathbf{e}}_i$ contains a contribution from the original entity vector and from the word2vec vector. As with Eqs. 3 and 5, Eq. 6 can be applied to any knowledge graph embedding model by replacing $\mathbf{e}_i$ with $\hat{\mathbf{e}}_i$ in Table 1.

Because word2vec learns a different set of latent features than the knowledge graph embedding, we use the matrix $\mathbf{M}$ to map vectors from word2vec feature space to entity feature space. Note that unlike the relation–specific transformations that operate on entities in the SE, TRANSR, and RESCAL models (*i.e.* $\mathbf{R}$, $\mathbf{R}^{(1)}$, and $\mathbf{R}^{(2)}$), $\mathbf{M}$ is a global matrix that is common to all relation types. Thus, the vector $\mathbf{wM}$ contains features that are useful for predicting triplets but learned from text. We refer to Eq. 6 as the FeatureSum model.

The FeatureSum model is trained in three phases. First, word2vec is trained on the corpus to obtain the $\mathbf{w}_i$ vectors. The next two phases optimize for the ranking loss objective (Eq. 1). Initially, $\mathbf{M}$ is set to zero and held constant while the entity and relation parameters $\mathbf{E}$ and $\mathbf{R}$ are optimized for 100 epochs. Finally, all parameters including $\mathbf{M}$ and $\mathbf{w}_i$ are trained together for the remainder of the training period. However, in some cases it was found that training the $\mathbf{w}_i$ vectors during the third phases offers no benefit over holding them constant after initialization with word2vec. This is discussed in further detail in Section 4.

4 Experimental Results

In this section we evaluate the proposed embedding enhancement methods on standard subsets of Freebase [Bollacker *et al.*, 2008] and Wordnet [Miller, 1995]. We apply the methods to each of the scoring functions in Table 1, demonstrating their ability to augment existing embedding models. We first quantitatively compare the WWV and PE-WWV models on the link prediction task against alternative methods for incorporating entity descriptions, and then we qualitatively examine the WWV model to better understand its performance. We next compare the FeatureSum model against alternative methods using an unstructured text corpus. For reproducibility all code and data has been made publicly available[1].

4.1 Data Sets

The Wordnet knowledge graph [Miller, 1995] is a database of English words and lexical relationships between them. Its purpose is to serve as a thesaurus, with different types of relations to indicate synonymy, similarity, and hypernymy/hyponymy between words. We use a subset of Wordnet introduced by [Bordes *et al.*, 2013] named WN18, consisting of 18 relations, 40,943 entities, and 156,442 facts.

Freebase is a large general-purpose knowledge graph containing a wide variety of facts about the world, including people, places, and culture. In our experiments we use FB15k, a subset of Freebase by [Bordes *et al.*, 2013] consisting of 1,345 relations, 14,951 entities, and 592,213 facts.

We perform multiple experiments utilizing different sources of textual data. As an unstructured text corpus we use the Google News data set and word2vec vectors pretrained on it[2]. For entity descriptions in Wordnet we use definitions provided with the data set. For Freebase entity descriptions we use the summary section of the Wikipedia article associated with each entity. We remove stopwords in all cases.

Unfortunately, some of the names and Wikipedia articles for Freebase entities were not available. In order to accurately measure the effectiveness of our approach we remove all entities without descriptions as well as all triplets involving them. The resulting number of entities, relations, and triplets is given in Table 4. This table also gives the number of words in the vocabularies and the breakdown of triplets into training, validation, and test sets.

4.2 Implementation Details

All experiments are performed by optimizing Eq. 1 using the AdaGrad algorithm [Duchi *et al.*, 2011] , with separate experiments for each of the scoring functions given in Table 1. We evaluate how prediction performance differs only by redefining the entity vectors $\mathbf{e}_s$ and $\mathbf{e}_o$. For the WV, WWV, PE-WWV, and FeatureSum models each instance of the entity vector $\mathbf{e}_i$ is replaced with the augmented vector defined in Eqs. 2, 3, 5, and 6, respectively. The word2vec vectors $\mathbf{w}_i$ in Eq. 6 are obtained from the words in the name of each entity e_i. When entity names contain multiple words we take the mean of all word vectors in the name. Any words for which no word2vec vector is known (due to absence in the word2vec training corpus) are ignored.

We observe empirically that after initializing $\mathbf{w}_i$ in Eq. 6 using word2vec, allowing these vectors to be optimized can improve performance in some cases, while in other cases they are better held constant. After measuring the performance of each option on a set of validation triplets, we chose to hold all $\mathbf{w}_i$ constant for SE and TRANSR, while for other scoring functions they are optimized.

We measure the link prediction performance of each embedding model using the metrics of mean rank and hits@10. For each test triplet (e_s, r_p, e_o), we compute the score $f(e_s, r_p, e_o)$ as well as the score of corrupted triplets $f(e_s, r_p, e'_o)$ for all $e'_o \in \mathcal{E}$. We then measure the rank of the correct triplet among corrupted ones. Similarly, we measure the rank of $f(e_s, r_p, e_o)$ among corrupted triplets $f(e'_s, r_p, e_o)$ for all $e'_s \in \mathcal{E}$, which serves as another data point. Mean rank (MR) is defined as the mean of all such

[1]github.com/rubikloud/kg-text-embeddings

[2]code.google.com/archive/p/word2vec

Model	SE		TransE		TransR		Rescal		DistMult		HolE	
	MR	H@10	MR	H@10	MR	H@10	MR	H@10	MR	H@10	MR	H@10
Wordnet (WN18)												
Base	1292.00	**0.373**	885.23	**0.554**	720.91	**0.452**	829.31	**0.605**	715.90	**0.690**	742.15	**0.678**
WV-desc	245.60	0.300	149.99	0.461	212.77	0.334	161.78	0.270	173.52	0.206	212.03	0.179
WWV	246.41	0.300	134.77	0.463	209.79	0.324	159.30	0.320	159.78	0.238	186.92	0.235
PE-WWV	**221.49**	0.302	**130.79**	0.483	**178.60**	0.327	**146.45**	0.337	**102.94**	0.387	**118.08**	0.360
Freebase (FB15k)												
Base	844.42	0.149	188.74	0.389	320.07	0.247	219.84	0.286	251.93	**0.390**	193.45	0.295
WV-desc	**169.94**	**0.285**	142.67	0.374	**150.74**	0.332	174.05	0.277	243.43	0.191	224.21	0.211
WWV	178.34	0.277	137.46	0.393	159.05	0.328	172.67	0.295	239.39	0.228	198.88	0.268
PE-WWV	174.25	0.276	**134.72**	**0.409**	153.98	**0.335**	**167.67**	**0.305**	**155.30**	0.316	**154.31**	**0.444**

Table 2: Mean rank (MR) and hits at 10 (H@10) for link prediction using entity descriptions on WN18 and FB15k

Relation	**Top words**
/people/person/place_of_birth	canadian, hungarian, italian, scottish, israeli, swedish, arizona, lovely, english, minneapolis
/organization/organization_member/member_of./-organization/organization_membership/organization	country, economic, un, college, countries, city, nations, american, development, eu
/music/performance_role/regular_performances./-music/group_membership/role	instrument, used, strings, tuned, pitch, organ, viola, bell, career, drum
/award/award_category/winners./award/award_honor/-ceremony	grammy, oscars, tony, born, staples, press, screen, primetime, british, given
/tv/tv_program/regular_cast./tv/regular_tv_appearance/-actor	glee, anatomy, wire, elsewhere, charmed, saturday, hbo, downton, brother, deadwood
/education/educational_institution/students_graduates./-education/education/student	canadian, applications, german, tufts, scottish, ontario, curriculum, ivy, symphony, australian

Table 3: Top weighted words for Freebase relations learned by WWV with TransE

Data Set	# Ent. (n_e)	# Rel. (n_r)	# Train triplets	# Valid triplets	# Test triplets	Vocab. size
WN18	40943	18	146442	5000	5000	32666
FB15k	12479	817	287065	29849	35121	36015

Table 4: Dataset characteristics

ranks, while hits@10 is defined as the fraction of ranks that are less than or equal to 10.

Training in all experiments is performed using 200 epochs and a batch size of 1024. The learning rate was selected based on validation performance, resulting in a learning rate of 0.01 for the baseline and FeatureSum experiments and 0.1 for WV, WWV, and PE-WWV. We use an embedding dimensionality of $d = 100$ and a margin of $\gamma = 1.0$ for the ranking loss.

4.3 WWV and PE-WWV Results

In this section we evaluate both variants of the weighted word vectors model — WWV and PE-WWV. We compare against the alternative method for incorporating textual data formatted as entity descriptions, namely, the WordVectors model but applied to entity descriptions rather than names. We refer to this model as WV-desc. We also consider the baseline method, named Base, in which entity vectors are simply initialized randomly and optimized with no supplementary text.

The mean rank and hits@10 metrics are given in Table 2 for both datasets. We expect that WWV should perform strictly better than WV-desc considering that WWV is a generalization of WV and can be reduced to the latter by simply setting $B_{jk} = 1$ in Eq. 2 for every j, k. Indeed, WWV outperforms WV-desc in both mean rank and hits@10 in most cases.

Somewhat surprisingly, the PE-WWV model performs at least as well as WWV and in many cases even better. One might expect that PE-WWV should perform worse because its representational capacity is no greater than that of WWV. WWV can be made equivalent to PE-WWV by setting $B_{jk} = \exp(\mathbf{P}_j \mathbf{W}_k^T)$, meaning it is theoretically able to perform at least as well. Upon closer investigation we find that PE-WWV tends to learn relatively stronger weights for the top words than WWV, which in turn allows it to create greater variability between entity representations for different relations. Thus it appears that WWV is limited by the optimization algorithm rather than its theoretical properties.

When compared to the baseline, the mean rank improves in most cases, while the hits@10 shows differing results between Wordnet and Freebase. For Wordnet, the baseline performs best on hits@10 compared to any text augmentation method, which suggests that the text in this data set may not be very indicative of the associated entities. For Freebase, however, the descriptions provide significant benefit, improving both mean rank and hits@10 in most cases.

To better understand this behaviour, we examined the Wordnet test triplets that were ranked significantly better by the baseline than WWV-desc. We observe that in many of these cases, related entities are described by completely dissimilar text. For example, one such triplet is (*kilobyte*, *has_part*, *word*), for which the Wordnet definitions of the subject and object are "unit information equal bytes" and "word string bits stored computer memory large computers use words bits long" (stopwords omitted). Because these def-

Model	SE		TransE		TransR		Rescal		DistMult		HolE	
	MR	H@10	MR	H@10	MR	H@10	MR	H@10	MR	H@10	MR	H@10
Wordnet (WN18)												
Base	1292.00	0.373	885.23	0.554	720.91	0.452	829.31	**0.605**	715.90	0.690	742.15	**0.678**
WV-names	990.55	0.267	593.96	0.391	772.71	0.284	501.92	0.420	396.71	0.444	407.30	0.456
WV-names-init	642.59	0.251	453.86	0.375	587.03	0.291	439.99	0.323	**191.81**	0.459	**248.92**	0.545
FeatureSum	**488.74**	**0.637**	**285.34**	**0.613**	**322.46**	**0.519**	**376.63**	0.488	393.96	**0.745**	487.34	0.658
Freebase (FB15k)												
Base	844.42	0.149	188.74	0.389	320.07	0.247	219.84	0.286	251.93	0.390	193.45	0.295
WV-names	348.54	0.208	175.91	0.366	261.21	0.272	216.47	**0.293**	185.59	0.327	214.66	0.249
WV-names-init	345.30	0.195	201.06	0.346	327.42	0.225	315.97	0.263	158.21	**0.443**	191.38	**0.415**
FeatureSum	**211.06**	**0.326**	**120.65**	**0.501**	**183.78**	**0.298**	**187.67**	0.270	**144.67**	0.366	**172.06**	0.307

Table 5: Mean rank (MR) and hits at 10 (H@10) for link prediction using unstructured text documents on WN18 and FB15k

initions contain no common words, it may appear to the WV-desc model that they are unrelated. In contrast, the Wikipedia summaries of *kilobyte* and *word* contain common keywords such as *unit*, *digital*, and *memory*, which may explain why WV-desc performs much better on Freebase.

To further test this hypothesis, we compute the average number of common words between the subject and object descriptions in each triplet. In Wordnet, triplets that are ranked significantly better by Base than by WV-desc contain an average of 0.67 common words, whereas triplets that are ranked significantly better by WV-desc contain an average of 0.89 common words. In comparison, triplets in Freebase contain an average of 20.0 common words. Thus, it appears that the word vectors methods require more detailed entity descriptions than are available in Wordnet.

Qualitative Results

To better understand how the WWV model works we examine which words are assigned the greatest weight for each relation after training. Table 3 lists the top 10 words — extracted from the strongest weights in $\mathbf{B}_i$ — for several relations r_i in Freebase, trained using TransE.

We observe that many of the top words are semantically similar to the relation. For example, the *place_of_birth* relation tends to emphasize words that are nationalities, whereas the relation for membership in a musical group emphasizes words regarding musical instruments such as *instrument, strings*, and *drum*. This demonstrates that the model functions as our intuition suggests and represents entities by the words that best indicate the relation in question. Note that the associations between words and relations in this table are learned in a fully unsupervised manner given only the text and training triplets.

4.4 FeatureSum Results

In this section we compare the FeatureSum model against other methods for incorporating information from an unstructured text corpus. Each method differs in the way the entity vectors are defined. The WV-names model applies the Word-Vectors technique of [Socher *et al.*, 2013] (Eq. 2) where each entity is associated with its name's constituent words. This model also does not use any supplementary textual data, but can improve over the baseline. The WV-names-init model is similar to WV-names but with each $\mathbf{w}_i$ initialized with the word2vec vector for word w_i. This model does incorporate the textual data via the training of the word2vec vectors and is therefore a key reference point for the FeatureSum model.

The results are given in Table 5 for both data sets. Note that these results are not directly comparable to Table 2 because the two sets of experiments use different textual data. Comparisons should be made across rows within the same table. In doing so, the mean ranks on Wordnet generally indicate that applying WV to the entity names alone already gives a significant improvement, while initialization with word2vec vectors improves the results further, as suggested by [Socher *et al.*, 2013]. For hits@10 the results are mixed, with the WV-names methods showing benefit in some cases but losses in others. More interestingly, in most cases all of these models are outperformed by the proposed FeatureSum method.

5 Conclusion

This paper discusses two novel methods methods to augment entity embeddings in a knowledge graph with information from textual data. The first method represents entity vectors as a direct function of the words associated with each entity, and is applicable whenever textual data is available in the form of entity descriptions. The second method trains the word2vec algorithm on the text documents and adds the features it learns for entity names to the original entity feature vector. Empirical results show that if the textual data is of sufficiently high quality, then both methods can improve link prediction accuracy on many different embedding models when compared against embeddings with no text and alternative methods for incorporating text.

References

[Auer *et al.*, 2007] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[Bollacker *et al.*, 2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

[Bordes *et al.*, 2011] Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. Learning structured embeddings of knowledge bases. In *AAAI*, volume 6, page 6, 2011.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[Gardner *et al.*, 2013] Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 833–838, 2013.

[Gardner *et al.*, 2014] Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 397–406, 2014.

[Lao *et al.*, 2012] Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1017–1026. Association for Computational Linguistics, 2012.

[Lin *et al.*, 2015] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187, 2015.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[Miller, 1995] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[Mintz *et al.*, 2009] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

[Nickel *et al.*, 2011] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816, 2011.

[Nickel *et al.*, 2016a] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[Nickel *et al.*, 2016b] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. Holographic embeddings of knowledge graphs. In *AAAI*, volume 2, pages 3–2, 2016.

[Riedel *et al.*, 2010] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer, 2010.

[Riedel *et al.*, 2013] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, 2013.

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

[Suchanek *et al.*, 2007] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[Surdeanu *et al.*, 2012] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465. Association for Computational Linguistics, 2012.

[Toutanova *et al.*, 2015] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, 2015.

[Wang *et al.*, 2014] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1591–1601, 2014.

[Wang *et al.*, 2017] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[Weston *et al.*, 2013] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. Connecting language and knowledge bases with embedding models for relation extraction. *arXiv preprint arXiv:1307.7973*, 2013.

[Yang *et al.*, 2014] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.