KADE: Aligning Knowledge Base and Document Embedding Models using Regularized Multi-Task Learning*

Matthias Baumgartner[1]✉, Wen Zhang[2,3]✉, Bibek Paudel[1]✉,
Daniele Dell'Aglio[1], Huajun Chen[2,3], and Abraham Bernstein[1]

[1] Department of Informatics, University of Zurich, Zurich, Switzerland
{baumgartner|bpaudel|dellaglio|bernstein}@ifi.uzh.ch
[2] College of Computer Science and Technology, Zhejiang University, Hangzhou, China
{wenzhang2015|huajunsir}@zju.edu.cn
[3] Alibaba-Zhejiang University Joint Institute of Frontier Technologies, Hangzhou, China.

Abstract. Knowledge Bases (KBs) and textual documents contain rich and complementary information about real-world objects, as well as relations among them. While text documents describe entities in freeform, KBs organizes such information in a structured way. This makes these two information representation forms hard to compare and integrate, limiting the possibility to use them jointly to improve predictive and analytical tasks. In this article, we study this problem, and we propose KADE, a solution based on a regularized multi-task learning of KB and document embeddings. KADE can potentially incorporate any KB and document embedding learning method. Our experiments on multiple datasets and methods show that KADE effectively aligns document and entitie embeddings, while maintaining the characteristics of the embedding models.

1 Introduction

In recent years, the open data and open knowledge movements gain more and more popularity, deeply changing the Web, with an exponential growth of open and accessible information. Wikipedia is the most successful example of this trend: it is among the top-5 most accessed Web sites and offers more than 40 million articles. Based on Wikipedia, several Knowledge Bases (KBs) have been created, such as FreeBase and DBpedia. Wikidata is a sibling project of Wikipedia which focuses on the construction of a collaboratively edited KB.

It is therefore natural to ask, how precise and complete information can be retrieved from such open repositories. One of the main challenges arising from this question is data integration, where knowledge is usually distributed and complementary, and needs to be combined to get a holistic and common view of the domain. We can envision two common cases where this challenge is relevant:

* M. Baumgartner, W. Zhang, and B. Paudel contributed equally to this work.

when users need open knowledge from different repositories, and when users need to combine open and private knowledge.

Key to the success of data integration is the *alignment* process, i.e. the combination of *descriptions* that refer to the same real-world object. This is because those descriptions come from data sources that are heterogeneous not only in content, but also in structure (different aspects of an object can be modelled in diverse ways) and format, e.g. relational database, text, sound and images. In this article, we describe the problem of *KB entity-document alignment*. Different from previous studies, we assume that the same real-world object is described as a KB entity and a text document. Note that the goal is not to align an entity with its surface forms, but rather with a complete document. We move a step towards the solution by using existing embedding models for KBs and documents.

A first problem we face in our research is how to enable comparison and contrast of entities and documents. We identify *embedding models* as a possible solution. These models represent each entity in a KB, or each document in a text corpus, by an embedding, a real-valued vector. Embeddings are represented in vector spaces which preserve some properties, such as similarity. Embeddings gained popularity in a number of tasks, such as finding similar entities and predicting new links in KBs, or comparing documents in a corpus.

So far, there are no algorithms to create embeddings starting form descriptions in different formats. Moreover, embeddings generated by different methods are not comparable out of the box. In this study, we ask if *is it possible to represent embeddings from two different models in the same vector space, which (i) brings close embeddings describing the same real world object, and (ii) preserves the main characteristics of the two starting models ?*

Our main contribution is KADE, a regularized multi-task learning approach for representing embeddings generated by different models in a common vector space. KADE is a generic framework and it can potentially work with any pair of KB and document embedding models. To the best of our knowledge, our study is the first to present a generic embedding model to deal with this problem. Our experiments show that KADE integrates heterogeneous embeddings based on what they describe (intuitively, embeddings describing the same objects are close), while preserving the semantics of the embedding models it integrates.

2 Related Work

Document Embedding. Paragraph Vector [4], also known as Doc2vec, is a popular document embedding model, based on Word2vec [6]. It has two variants: Distributed Memory and Skip Gram. They represent each document by an embedding such that the learned embeddings of similar documents are close in the embedding space. Document embeddings can also be learned with neural models like CNN and RNN [3,12], topic models, or matrix factorization. Furthermore, pre-trained embeddings from multiple models can be combined in an ensemble using dimensionally reduction techniques [20,10].

KB Embedding. Translation-based embedding models like TransE [1], TransR [5], and TransH [17] have been shown to successfully representing KB entities. Apart from entities, these models also represent each relation by an embedding. They aim to improve the link prediction task and treat each triple as a relation-specific translation. Several other embedding models have since been introduced [15,8]. RDF2Vec [9] uses the same principle as Word2vec in the context of a KB and represents each entity by an embedding. Other type of KB embedding models include neural, tensor and matrix based methods [7].

Combining Text and KB. Several approaches improve KB embeddings by incorporating textual information. [14] exploits lexical dependency paths extracted from entities mentioned in documents. [21] and [16] jointly learn embeddings of words and entities, by using an alignment loss function based on the co-occurrence of entity and words in its descriptions, or Wikipedia anchors, respectively. A similar approach has been used for named entity disambiguation [19]. DKRL [18] treats text descriptions as head and tail entities, and ignores the linguistic similarity while learning KB embeddings.

Further, multiple pre-trained embeddings can be combined into one multimodal space. [13] concatenates embeddings of aligned images, words, and KB entities, then fuses them via dimensionality reduction.

Regularized Multi-Task Learning (MTL). Regularized MTL [2] exploits relatedness between multiple tasks to simultaneously learn their models. It enforces task relatedness by penalizing deviations of individual tasks using regularization.

In contrast to these methods, our goal is to align documents with entities, while at the same time retaining the properties of both document and KB embeddings. Our model is flexible since it does not require a predefined alignment loss function, and learns by means of regularization through task-specific representations of documents and KB. It does not depend on the availability of further linguistic resources like a dependency parser, careful extraction of features like anchor text, or defining a separate alignment loss function. Our solution aims at preserving linguistic and structural properties encoded in the document and KB embeddings respectively, while also representing them in a common space.

3 Preliminaries

Knowledge Bases and Documents. A Knowledge Base $\mathcal{K}$ contains triples $k_j = \{(h, r, t) \mid h, t \in \mathcal{E}; r \in \mathcal{R}\}$, where $\mathcal{E}$ and $\mathcal{R}$ are the sets of entities and relations, respectively. (h, r, t) indicates that the head entity h and tail entity t are related by a relation r. Entities in KBs, such as Freebase and DBPedia, describe real-world objects like places, people, or books.

Text is the most popular way to describe real-world objects. It is usually organised in documents, which delimit the description to a specific object or concept, such as a country or a person. Formally, a document d in a corpus $\mathcal{D}$ is represented as a sequence $\langle w_1, \ldots, w_i, \ldots, w_{n_d} \rangle$, where $|d| = n_d$, and w_i denotes a word in the corpus drawn from a word vocabulary $\mathcal{W}$.

For example, Wikipedia contains a textual document about Mike Tomlin, the head coach of Pittsburgh Steelers, at `https://en.wikipedia.org/wiki/Mike_Tomlin` (denoted d_{mt}); FreeBase contains a graph-based description of Mike Tomlin, here identified by `m.0c5f_j`[4], e.g. the triple (`m.0c5f_j`, `current_team_head_coached`, `m.05tfm`) states that *Mike Tomlin* (the head entity) is the *head coach* (the relation) of *Pittsburgh Steelers* (the tail entity, `m.05tfm`).

We name such different forms of information as *descriptions* of real-world objects, e.g. d_{mt} and $m.0c5f\_j$ are two different descriptions of Mike Tomlin.

Embeddings. Embeddings are dense vectors in a continuous vector space which could be regarded as another representation of descriptions[5]. Embeddings gained popularity in recent years, due to their successful applications [6,3,12].

In this study, we consider two families of embedding models: the *translation-based models* for Knowledge Bases and the *paragraph vector models* for documents. The models of the former family represent KB entities as points in the continuous vector space, and relations as translations from head entities to tail entities. Representative models of the former family are TransE [1], TransH [17] and TransR [5]. TransE defines the score function for a triple (h, r, t) as: $S(h, r, t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$. TransH and TransR extend TransE to overcome the limited capability of TransE to encode 1-N, N-1 and N-N relations. They define the score function for a triple as: $S(h, r, t) = \|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\|$, in which $\mathbf{h}_\perp$ and $\mathbf{t}_\perp$ are projected head and tail entities. TransH projects entities to relation specific hyperplanes with $\mathbf{w}_r$ as normal vectors, thus $\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r$ and $\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r$. TransR projects entities from the entity space to the relation space via relation specific matrices $\mathbf{M}_r$, thus $\mathbf{h}_\perp = \mathbf{M}_r\mathbf{h}$ and $\mathbf{t}_\perp = \mathbf{M}_r\mathbf{t}$. Let (h, r, t) be a triple in $\mathcal{K}$, and (h', r', t') be a negative sample, i.e., (h', r', t') is not in $\mathcal{K}$. The margin-based loss function l_{KM} for (h, r, t) is defined as:

$$l_{KM}((h, r, t), (h', r, t')) = \max(0, (\gamma + S(\mathbf{h}, \mathbf{r}, \mathbf{t}) - S(\mathbf{h}', \mathbf{r}', \mathbf{t}'))), \tag{1}$$

where γ is a margin. The objective of translation-based models is to minimize the margin-based loss function for all triples, i.e., $\mathcal{L}_{KM}(\mathcal{K}) = \sum_{(h,r,t)\in\mathcal{K}} l((h, r, t), (h', r, t'))$, where (h', r', t') is a negative example sampled for (h, r, t) during training.

The paragraph vector models represent documents in a continuous vector space. Examples of models of this family are PV-DM (Distributed Memory) and PV-SG (Skip Gram or DBOW) [4]. Both models learn embeddings for variable-length documents, by training to predict words (in PV-DM) or word-contexts (in PV-SG) in the document. For every document d, the objective of PV-DM is to maximize the average log probability of a target word w_t appearing in a given context c_t, conditioned not only on the context words but also on the document:

$$l_{DM}(d) = \frac{1}{n_d} \sum_{t=1}^{n_d} log\ p(\mathbf{w}_t|\mathbf{d}, \mathbf{c}_t), \tag{2}$$

[4] We omit prefixes for the sake of readability.

[5] Througout this paper, we denote vectors in lowercase bold letters and matrices in uppercase bold letters.

where $p(\mathbf{w}_t|\mathbf{d}, \mathbf{c}_t) = \sigma(\mathbf{b} + \mathbf{U}g(\mathbf{d}, \mathbf{c}_t))$, and $\sigma(\cdot)$ is the logistic function, $\mathbf{U}$ and $\mathbf{b}$ are weight and bias parameters, and g is constructed by concatenating (or averaging) its parameters. The PV-SG objective is to maximize l_{DM} defined as:

$$l_{DM}(d) = \frac{1}{n_c} \sum_{t=1}^{n_c} log\ p(\mathbf{c}_t|\mathbf{d}), \tag{3}$$

where $p(\mathbf{c}_t|\mathbf{d}) = \sigma(\mathbf{b} + \mathbf{U}g(\mathbf{c}_t))$. In this way, both PV-DM and PV-SG capture similarities between documents by using both context and document embeddings to maximize the probability of context or target words. As a result, the embeddings of similar documents are close in the embedding space.

4 Aligning Embedding Models

From Section 3, we see that the descriptions of Mike Tomlin in FreeBase and Wikipedia bring complementary information. There is an intrinsic value in considering these two descriptions together, since they offer a more complete view of the person. Embedding models offer a space where descriptions can be contrasted and compared, and it is therefore natural to ask ourselves if we can integrate multiple descriptions by exploiting those models. However, embedding models represent descriptions in different vector spaces. In other words, while two embeddings generated by the same model are comparable, two embeddings generated by different models are not.

We want to study if it is possible to bridge together different embedding models, while preserving the characteristics of the individual models and enabling new operations. This operation, that we name alignment, should take into account two characteristics, namely relatedness and similarity, explained below.

Relatedness. Descriptions of the same real-world object are related. Let $\mathcal{T}$ be a set of descriptions; the document and entity sets ($\mathcal{D}$ and $\mathcal{E}$) are two disjoint subsets of $\mathcal{T}$. We introduce the notion of *relatedness*, to indicate that two descriptions refers to the same real-world object, as a function $rel : \mathcal{T} \mapsto \mathcal{T}$. *rel* is a symmetric relation, i.e. if $t_i \in \mathcal{T}$ relates to $t_j \in \mathcal{T}$, then the vice versa holds.

Similarity. In addition to relatedness, we intuitively introduce the notion of *similarity*. Let's consider the following example: Pittsburgh Steelers and San Francisco 49ers are two football teams. They are two real-world objects, and in the context of a set of real-world objects, we can define a notion of similarity between them. For example, if we consider all the sport teams in the world, the two teams are similar, since they share several features—they are football teams, they play in the same nation and in the same leagues. However, in the context of NFL, the two teams are less similar—they play on different coasts and have different achievements. From this example, we observe that the notion of similarity can be relation- or context-specific.

Our proposed model is agnostic to the notions of similarity adopted by individual KB and document embedding models. We want our model to be robust to

the choice of individual embedding models. In this way, we only loosely interpret the model-specific loss functions and and choice of similarity measure.

Alignment. The problem we investigate in our research is of *aligning* the embeddings of related descriptions. It is worth stressing, that alignment captures both, the notion of relatedness and similarity. The goal is to obtain a space where the embeddings of related descriptions are close (i.e. relatedness), while preserving the semantics of their original embeddings (i.e., similarity).

Assumptions. In this study, we make the following assumptions. First, relatedness is defined as a function that relates each entity of $\mathcal{E}$ to a document in $\mathcal{D}$, and vice versa. Formally, we denote this relation with rel_{ed} and it holds: (i) rel_{ed} is injective, (ii) $\forall e \in \mathcal{E}$, $rel_{ed}(e) \in \mathcal{D}$ and (iii) $\forall d \in \mathcal{D}$, $rel_{ed}(d) \in \mathcal{E}$. Based on rel_{ed}, we define the *entity-document relatedness set* as $Q = \{(e, d) \mid rel_{ed}(e) = d \wedge e \in \mathcal{E} \wedge d \in \mathcal{D}\}$, e.g. $(m.0c5f\_j, d_{mt}) \in Q$.

The second assumption is based on the fact that, in real scenarios, it often happens that only some relations between document and entities are known. We therefore assume that the algorithm can access a relatedness set $Q' \subset Q$.

In the following, we abuse the notation and we use d and e to indicate embeddings when it is clear from the context. When not, we use $v(t)$ to indicate the embedding of the description (either entity or document) $t \in \mathcal{T}$.

5 A Regularized Multi-Task Learning Method for Aligning Embedding Models

In this section we present KADE, a framework to align **K**nowledge base **a**nd **D**ocument **E**mbedding models. KADE can be separated into three parts, as shown in Fig. 1: (i) a Knowledge Base embedding model, on the left, where vectors are denoted by cirlces, (ii) a document embedding model, on the right, where vectors are denoted by squares, and (iii) a regularizer process, in the center.

The construction of the Knowledge Base (or document) embedding model is represented by the arrows with dashed lines, which represent the moving direction for entity (or document) embeddings according to the underlying model.

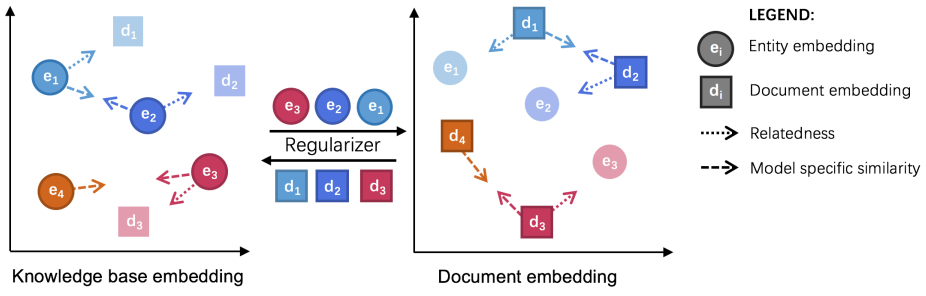


Fig. 1: Illustration of the intuition behind KADE.

The colors and index numbers in the two models indicate that the vectors are describing the same real-world object, i.e., $(e_i, d_i) \in Q, i \in [1, 4]$. Moreover, (e_1, d_1), (e_2, d_2) and (e_3, d_3) are known related entity-document pairs, i.e., $(e_i, d_i) \in Q', i \in [1, 3]$, while the relatedness information of (e_4, d_4) is not known, i.e., $(e_4, d_4) \in Q$ and $(e_4, d_4) \notin Q'$.

The regularization process builds an embedding space which represents both, the document and the KB entity vectors. There are two regularizers: each of them applies to the training of the document and Knowledge Base embedding, forcing the related document and the entity vectors to be close in the vector space. The regularizer process is shown through the arrows with dotted lines, which represent the moving direction influenced by the related entity-document pairs. This is done by exploiting the information from Q', i.e., the regularizer process cannot use the (e_4, d_4) pair, since it is not in Q'.

Regularizer for the Knowledge Base embedding model. We define KADE's objective function $\mathcal{L}^K$ for a Knowledge Base embedding model as follows:

$$\begin{aligned}\mathcal{L}^K(\mathcal{D}, \mathcal{K}) = \sum_{\substack{(h,r,t)\in\mathcal{K} \\ (h',r,t')\notin\mathcal{K}}} \Big(& l_{KM}((h,r,t),(h',r,t')) + \\ & \lambda_k \Big(\|v(h) - v(rel_{ed}(h))\| + \|v(t) - v(rel_{ed}(t))\| + \\ & \|v(h') - v(rel_{ed}(h'))\| + \|v(t') - v(rel_{ed}(t'))\| \Big) \Big),\end{aligned} \tag{4}$$

where l_{KM} is defined in (1), $v(rel_{ed}(e))$ is the embedding corresponding to the document describing e (from the relatedness set), and λ_k is the regularizer parameter for the KB embedding model. If the related entity-document pair is missing in Q', then the regularization term for that entity is zero.

Regularizer for the document embedding model. Similarly, we define the KADE's objective function for a document embedding model as follows:

$$\mathcal{L}^D(\mathcal{D}, \mathcal{K}) = \sum_{d\in\mathcal{D}} \Big(l_{DM}(d) + \lambda_d \|v(d) - v(rel_{ed}(d))\| \Big) \tag{5}$$

where l_{DM} is the loss function as defined in either (2) or (3), $v(rel_{ed}(d))$ is the embedding for the entity related to d, and λ_d is the regularizer parameter for document embedding model. As above, if the related entity-document pair is not in Q', then the regularization term for the document is set to zero.

Learning Procedure. The learning procedure of KADE is described in Algorithm 1. To learn the complete model, the algorithm trains the document and the KB embedding models alternately. This is done by using batch stochastic gradient descent, the common training procedure for these kinds of models. Specifically, one iteration of the learning keeps the KB model fixed and updates the document model. Within this step, the document model is updated for a given number of iterations. After that, the KB model is updated by keeping the

Algorithm 1: Iterative learning of embeddings in KADE

Input: A Knowledge Base $\mathcal{K}$ and document corpus $\mathcal{D}$, a KB embedding model KM and its loss function $\mathcal{L}_{KM}$, a document embedding modem DM and its loss function $\mathcal{L}_{DM}$, regularizer parameters λ_K and λ_D, relatedness relation rel_{ed} between entities in $\mathcal{K}$ and documents in $\mathcal{D}$, embedding dimension k, number of iterations num_iters, loss threshold $\epsilon < 1.0$, iterations per model $iter\_model$.

Result: Document embeddings $D \in \mathbb{R}^{|\mathcal{D}| \times k}$, Entity embeddings $E \in \mathbb{R}^{|\mathcal{E}| \times k}$.

Initialize E and D along with other model variables according to KM and DM; set $iters = 0$, $iters\_model = 0$, $loss\_change = km\_loss\_change = dm\_loss\_change = 1$.
while $iters \leq num\_iters$ *AND* $loss\_change > \epsilon$ **do**
  for $i_{km} = 1$ *;* $i_{km} \leq iters\_model$ *;* $i_{km} = i_{km} + 1$ **do**
    $\mathcal{L}_{KM} \leftarrow$ Calculated according to (1);
    $\mathcal{L}^K(\mathcal{K}, \mathcal{D}) \leftarrow$ Calculated according to (4) ;
    Update E and other KM variables using the gradients of $\mathcal{L}^K(\mathcal{K}, \mathcal{D})$;
  end
  for $i_{dm} = 1$ *;* $i_{dm} \leq iters\_model$ *;* $i_{dm} = i_{dm} + 1$ **do**
    $\mathcal{L}_{DM} \leftarrow$ Calculated according to (2);
    $\mathcal{L}^D(\mathcal{K}, \mathcal{D}) \leftarrow$ Calculated according to (5);
    Update D and other DM variables using the gradients of $\mathcal{L}^D(\mathcal{K}, \mathcal{D})$;
  end
  if $iters > 0$ **then**
    $km\_loss\_change \leftarrow |prev\_km\_loss - \mathcal{L}^K(\mathcal{K}, \mathcal{D})|$;
    $dm\_loss\_change \leftarrow |prev\_dm\_loss - \mathcal{L}^D(\mathcal{K}, \mathcal{D})|$;
  end
  $prev\_km\_loss \leftarrow \mathcal{L}^K(\mathcal{K}, \mathcal{D})$;
  $prev\_dm\_loss \leftarrow \mathcal{L}^D(\mathcal{K}, \mathcal{D})$;
  $loss\_change = max(km\_loss\_change, dm\_loss\_change)$;
  $iters \leftarrow iters + 1$;
end

document model fixed. In a similar way, this step lasts for a given number of iterations. Then the next iterations of KADE proceeds in a similar fashion, until convergence or a fixed number of steps.

6 Experimental Evaluation

Hypotheses. Our experiments aim at verifying three hypotheses. The model learned by KADE retains the characteristics of the document (HP1) and KB (HP2) embedding models, i.e. it does not break the semantics of the document and KB models. Additionally, KADE represents documents and KB entities in the same embedding space such that related documents and entities are close to each other in that space (HP3).

Datasets. We consider three open datasets, which are widely used to benchmark KB embeddings. Their properties are summarized in Table 1. Each of them consists of a Knowledge Base and a document corpus. Related documents are known for all entities, and vice-versa. **FB15k** [1] is derived from the most popular 15,000 entities in Freebase. The text

	FB15k	FB40k	DBP50
Entitites	14,904	34,609	24,624
Relations	1,341	1,292	351
Total Triples	530,663	322,717	34,609
Train triples	472,860	258,175	32,388
Unique words	35,649	50,805	158,921

Table 1: Dataset sizes

corpus is constructed from the corresponding Wikipedia abstracts. **FB40k**[6] is an alternative to FB15k, containing about 34,000 Freebase entities. It is sparser than FB15k: it includes more entities but fewer relations and triples. The text corpus is derived in the same fashion as for FB15k. **DBP50** is provided by [11]. It is extracted from DBpedia, it has fewer triples than the other datasets, but its documents are longer and its vocabulary is larger. We apply standard preprocessing steps like tokenization, normalization and stopword removal on the documents of the three datasets. The KB triples are split into training and test sets, in a way that each entity and relation in the test set is also present in the training set. The relatedness set Q contains pairs of Freebase/DBpedia entities and documents describing the same real-world object. When not specified, experiments use Q as known relatedness set. Otherwise, we describe how we built $Q' \subset Q$.

Methods. As explained in Section 3, for documents, we use two *Paragraph Vector* models: distributed memory (PV-DM) and skip-gram (PV-SG). For KBs, we consider TransE, TransH, and TransR. The method configuration is indicated in parenthesis, e.g. KADE(TransR,PV-SG). We use KADE(TransE, PV-DM) as our reference configuration, denoted by KADE_{ref} . When KB or document models are trained on their own, we refer to them as *Independent* models.

Parameters. The hyperparameters for TransE and TransR are embedding dimension k, learning rate α, margin γ, and a norm. TransH has an additional weight parameter C. KADE further introduces the regularizers λ_k and λ_d. We did a preparatory parameter search to find reasonable values for embedding dimension k and regularizers λ_k, λ_d. The search was limited to KADE_{ref} on FB15k and resulted in $k = 100$, $\lambda_k = 0.01$, and $\lambda_d = 0.01$. For the remaining parameters, we adopted the values reported by TransE authors [1]: $\alpha = 0.01, \gamma = 1, \text{norm} = \text{L1}$. We used these values for TransH and TransR as well. We adopted $C = 1$ for TransH from [16].

The parameters for the document model are the embedding dimension k, learning rate α, window size w, and number of negative samples n_{neg}. After a preparatory parameter search, we adopted $\alpha = 15, w = 5, neg = 35$.

Experiments are iterated 9600 times[7], in batches of 5000 samples. KADE switches between models every ten training batches ($iters\_model$ in Algorithm 1).

Implementation. We use our own implementation of TransE, TransH, TransR, since some methods do not provide a reference implementation. To assess our implementations, we ran the link prediction experiments of [1,17,5] with parameters values described above. Results are summarized in Table 2. For each model, the performance of KADE is reasonably close to the originally reported values, although they do not match exactly. These differences are due to parameter settings and implementation details: We used the same parameters for all three

[6] Available at `https://github.com/thunlp/KB2E`

[7] This matches the number of training runs over the whole dataset reported in [1].

		TransE		TransH		TransR	
		ours	Bordes [1]	ours	Wang [17]	ours	Lin [5]
HITS@10	raw	0.469	0.349	0.459	0.425	0.443	0.438
	filtered	0.712	0.471	0.692	0.585	0.656	0.655
Mean rank	raw	225.410	243.000	234.494	211.000	229.433	226.000
	filtered	86.304	125.000	97.527	84.000	92.689	78.000

Table 2: Performance of our implementations and published results on KB link prediction. Higher HITS@10 values are better; lower mean rank values are better.

methods, while [5,17] optimized such parameters, and we found that slightly different sampling strategies, as implemented in [5], can further improve the result. Ours, as well as other implementations of TransE [8], normalize the embedding vectors during loss computation. This has a positive effect on performance.

It is worth noting that these slight differences in performance do not affect the study of our hypotheses, which is to align document and KB embeddings while retaining the semantics of the original models.

6.1 HP1: KADE Retains the Document Embedding Model

HP1 states that KADE retains the quality of document embedding models. We study HP1 by using the document embeddings as features for binary classifiers[9]. We report the results of KADE_{ref} for FB15k; we had similar results for FB40k.

We first build the category set by retrieving categories from Freebase for each entity in FB15k. Since each entity is related to a textual document through Q, we assign categories to the documents. As a result, we retrieved 4,069 categories, with an average of 46.2 documents per category, a maximum of 14,887 documents[10], and a minimum of one. To have enough positive and negative training examples for each category, we remove categories that belonged to fewer than 10% or more than 50% documents, resulting in 27 categories, with an average of 2,400 documents per category (max: 4,483, min: 1,502). For each category C, we randomly select documents not belonging to C, so that we obtain equal number of documents in positive and negative classes. Next, we randomly assign 70% documents from each class to the training set and the remaining 30% to the testing set. Finally, we train a logistic regression classifier for each category and test the classification accuracy on the testing set. As a result, we train two classifiers for each category (54 classifiers in total): one related to KADE and the other one related to the independent document embedding model PV-DM.

We repeated this procedure five times and report the average result (the standard deviations are small and omitted) in Fig. 2. KADE significantly improves the classification accuracy on all categories. The average classification accuracy from KADE_{ref} is 0.92, while the one from independent training is 0.80. On many

[8] `https://github.com/thunlp/KB2E`

[9] We preferred binary over multi-class classification because it is simpler to explain.

[10] For the class `http://rdf.freebase.com/ns/common.topic`.

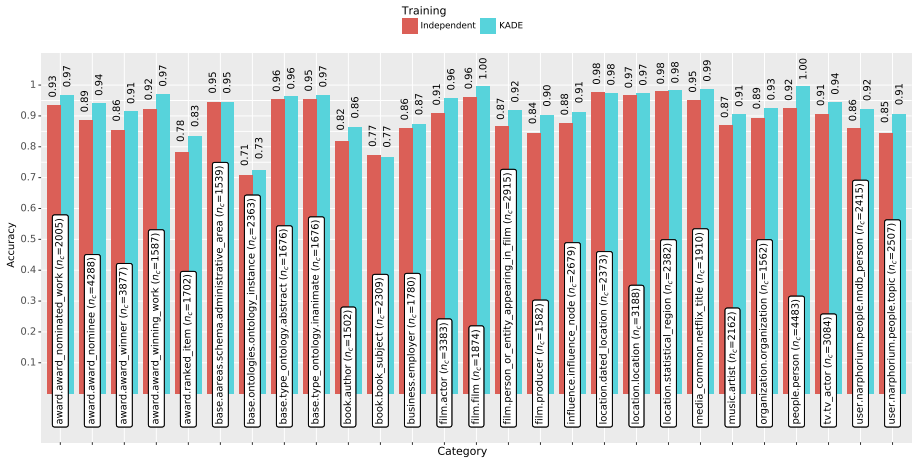


Fig. 2: Accuracy of binary document classification using KADE_{ref} and independently trained document model. The name of Freebase categories and number of documents in each category is listed along the vertical bars.

categories, embeddings from KADE achieve an accuracy above 0.95, while the maximum accuracy of embeddings from independent training is 0.89.

The classification accuracy of the document model increases when KADE uses PV-SG rather than PV-DM. The accuracy in the independent case lifts to 88%, with a maximum of 98%. It sill holds that KADE improves over the independent models, even though in many cases the values rougly match.

The results we obtained in this experiment suggest that document embeddings learned by KADE are not worse than the one learned by independent document embedding models. On the contrary, the document embeddings learned by KADE perform better in document classification.

6.2 HP2: KADE Retains the KB Embedding Model

The second hypothesis relates to the ability of KADE to maintain the semantics of the KB embedding model used in the alignment process. We study this hypothesis by performing the link prediction experiment proposed in [1].

Similar to previous studies, for every triple in the test set, we construct corrupted triples by replacing the head (or tail) entity with every other entity in the Knowledge Base. While testing link prediction, we rank all true and corrupted triples according to their scores and get the rank of current test triple.

We report the mean rank (MR) of the test triples and the ratio of test triples in the 10 highest ranked triples (HIT@10). As noted by [1], some corrupted triples be present in the KB. To cope with this, we report *filtered* results, where corrupted triples that are present in the training set are removed.

			FB15k		FB40k		DBP50	
			KADE	indep.	KADE	indep.	KADE	indep.
TransE	HITS@10	raw	**0.470**	0.469	**0.590**	0.583	**0.440**	0.382
		filtered	**0.715**	0.712	**0.754**	0.746	**0.469**	0.400
	Mean rank	raw	**221.257**	225.410	**835.899**	962.244	**1178.849**	2451.215
		filtered	**82.053**	86.304	**471.996**	598.209	**1130.392**	2403.093
TransH	HITS@10	raw	0.456	**0.459**	**0.579**	0.571	**0.434**	0.386
		filtered	0.689	**0.692**	**0.740**	0.731	**0.463**	0.405
	Mean rank	raw	**233.073**	234.494	**857.130**	1007.940	**1174.965**	2511.550
		filtered	**96.009**	97.527	**496.215**	649.459	**1126.766**	2463.208
TransR	HITS@10	raw	0.414	**0.443**	0.554	**0.556**	**0.376**	0.374
		filtered	0.651	**0.656**	0.705	**0.711**	**0.395**	0.392
	Mean rank	raw	242.700	**229.433**	**928.089**	929.848	**2414.541**	2430.336
		filtered	98.620	**92.689**	**561.778**	563.693	**2366.314**	2382.091

Table 3: HIT@10 and MR of KADE and independent training. HITS@10 reports the fraction of true triples in the top 10 predicted triples (higher is better). MR indicates the position of the original head (or tail) in the ranking (lower is better).

Table 3 compares the link prediction performance of KADE_{ref} and independent training over the datasets. KADE_{ref} slightly but consistently outperforms independent training. While there is always an improvement, it is more pronounced on sparser datasets. We conduct the same experiment with TransH and TransR with PV-DM as document model. All other parameters and experimental protocols remain identical. In a few cases KADE embeddings do not outperform the independent embeddings. However, the difference is small ($< 7\%$), compared to the differences introduced by the method or dataset.

Also in this case, experiments suggest that HP2 holds, and KADE retains the semantics of the KB embedding models.

6.3 HP3: KADE Aligns KB and Document Embeddings

The first two hypotheses are meant to assess if KADE retains the semantics of the embedding models. Our last hypothesis takes a different perspective, and it aims at verifying if the resulting model is effectively aligning entity and document embeddings. We study this hypothesis in two experiments.

Progression of KADE training. The first experiment tests if KADE brings document and KB entity embeddings into the same embedding space. For this, we randomly select 100 related entity-document pairs from Q' and call this set Q_R. At different stages of training (i.e. after different numbers of iterations), we take both document and entity embeddings from Q_R. Then, we retrieve the most similar entity embeddings learned by KADE using cosine similarity. In this way, we get two ranked lists of 100 entities, corresponding to a pair $q_R \in Q_R$: one retrieved using the document embedding of q_R as query, and another using the entity embedding of q_R as query. We compare the changes in set overlap of these two ranked lists, and rank correlation as the training of KADE proceeds.

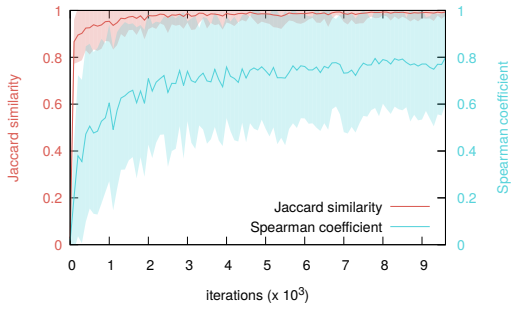


Fig. 3: The process of embedding documents and KB entities in the same space, with KADE_{ref} on FB40k. Shaded areas indicate the standard deviations.

Fig. 3 shows the results of this experiment for KADE_{ref} on the FB40k dataset. In the early stages of training, set overlap (measured using Jaccard similarity) as well as rank correlation (measured by the Spearman coefficient) are very low, meaning that the two ranked lists are very different. Both measures show significant improvements as training progresses, showing that KADE is able to align the entities of documents and entities in the same space. We repeated this experiment with the other datasets and we observed a similar behaviour. The same holds when retrieving the document embeddings instead of entity embeddings for query pairs in Q_R.

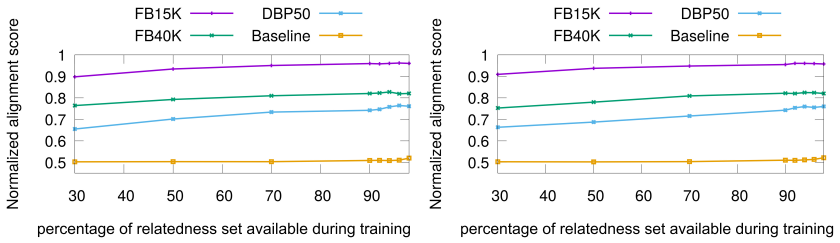
Alignment generalization by KADE. This experiment investigates to what extent KADE *generalizes*: can KADE align entity-document pairs that are not available during training, i.e., which are not in Q'? We first introduce the *alignment score* as an evaluation measure, afterwards we present the results.

Given a query document $d \in \mathcal{D}$, the alignment model orders all entities $e \in \mathcal{E}$ with respect to their similarity to d. Let $r_d(e) \in [1, |\mathcal{E}|]$ denote the *rank* of entity e, retrieved for query document d. We define the rank r_e of a document d in an analogous way. A perfect alignment model exhibits $r_d(e) = |\mathcal{E}|$ iff $(e, d) \in Q$, i.e., the related entity is ranked highest out of all possible choices. We further define the normalized version of the ranking measure as $r_d^N(e) = (r_d(e) - 1)/(|\mathcal{E}| - 1) \in [0, 1]$. We define $r_e^N(d)$ analogously.

For a pool of test documents $\mathcal{D}_t \subset \mathcal{D}$ and test entities $\mathcal{E}_t \subset \mathcal{E}$, we average the rankings for all document and entity queries (with respect to their related counterparts) to get the alignment score:

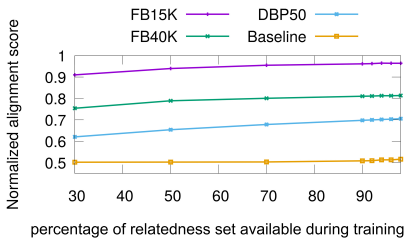
$$AS := 1/2 \left(\frac{1}{|\mathcal{E}_t|} \sum_{e \in \mathcal{E}_t} r_e^N[rel_{ed}(e)] + \frac{1}{|\mathcal{D}_t|} \sum_{d \in \mathcal{D}_t} r_d^N[rel_{ed}(d)] \right)$$

Next, we train KADE with varying sizes of $Q'$and examine to what extent embeddings of entity-document pairs in $Q \setminus Q'$ are aligned by KADE. Note that embeddings are still computed for all documents and entities, however, the document-entity pairs in $Q \setminus Q'$ are not regularized by KADE.

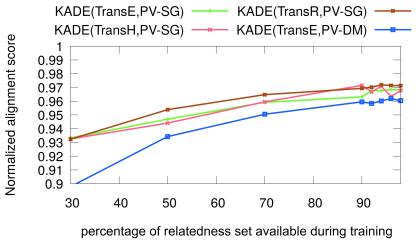


(a) KADE(TransE, PV-DM)

(b) KADE(TransH, PV-DM)



(c) KADE(TransR, PV-DM)



(d) KADE(PV-SG) on FB15k

Fig. 4: KADE aligns embeddings of related documents and entities. The x-axis show the percentage of known related pairs ($|Q'|/|Q|$). We measure the alignment score on a fixed set of 2% of Q. In the y-axis, 1 implies that for every query entity (document), the related document (entity) was retrieved as the first result. 0.5 indicates the random baseline.

Fig. 4a show what happens when Q' varies from 30% to 98% of Q. To ensure comparability between the different training data sets, we required the same 2% of Q to be omitted from all cases, and used them to calculate the alignment score. This results in test sets sizes of 299 (FB15k), 693 (FB40k), and 493 (DBP50).

We compare KADE against a baseline model built with independently constructed embeddings for documents and entities to show the impact of regularized multi-task learning. In this baseline model, the alignment is achieved by projecting document embeddings onto entity embeddings, and vice-versa, i.e.:

$$\forall e \in \mathcal{E} : e = \sigma\left(rel_{ed}(e)\mathbf{P}_1 + \mathbf{b}_1\right) \forall d \in \mathcal{D} : d = \sigma\left(rel_{ed}(d)\mathbf{P}_2 + \mathbf{b}_2\right)$$

The sigmoid function σ allows the model to account for nonlinearity. The projection matrices $\mathbf{P}_{\{1,2\}}$ and biases $\mathbf{b}_{\{1,2\}}$ are estimated from Q'. The model is evaluated on FB15k, with the same test set as used for KADE.

Fig. 4a shows that KADE's performance improves when the size of Q' increases. This effect reflects that embeddings of test pairs are constrained by their neighboring documents and entities.

Further, the performance differs across datasets. FB15k shows the best performance. This can be explained by the fact that we optimized the model parameters based on this dataset. We further explain this result considering the dataset

density: higher interconnection allows more coherent embedding construction in the document and entity models.

KADE consistently outperforms the baseline. Although the baseline enhances with more training data, it only narrowly improves over random guessing ($AS = 0.5$). This indicates that independent embedding construction leads to incompatible spaces and highlights the impact of KADE's regularized multi-task learning.

The same experiment is repeated for the other KB or document models. While the method is exchanged, the experimental setup and all parameters are maintained. For KB models (Fig. 4b and 4c), we observe that the influence of the dataset is much higher than the one of the KB model. This reflects the results from Table 3. For document models, Fig. 4d shows the effect of using PV-SG instead of PV-DM. As baseline, KADE_{ref} is plotted. Consistent with results from Section 6.1, PV-SG outperforms PV-DM, independent of the KB model. The effect is more pronounced if the size of Q' is small.

The experiments in this section assess HP3: KADE aligns the embeddings from different models in a common vector space, built according to the relatedness among the descriptions.

7 Conclusion and Future Work

In this paper, we introduced KADE, a flexible regularized multi-task learning method that represents embedding from heterogeneous models in a common embedding space. We show that KADE can work with different KB or document embedding methods. Our experiments showed that the KADE regularization process does not break the semantics of the underlying document and Knowledge Base embedding models, and in some cases it may even improve their performance.

Looking at the assumptions we took, we think that a promising and important direction of this research is to cope with changes in the set of objects that are considered. In other words, how can KADE cope with description of new objects that are added to the document corpus or the KB? KADE makes use of document embeddings that are pre-trained on a very large corpus. It follows, that the word embeddings which the model learns are general enough for new and unseen documents. This means that KADE might use the already computed document embeddings to find embeddings for new documents. As suggested in [4], it is possible to learn the embeddings for new documents by keeping the network weights and other embeddings constant, and updating the gradients of the document embedding for a few iterations. Similarly, translation-based KB embedding models cannot deal with new entities out of the box. However, we believe it may be possible to learn embeddings for unseen KB entities by using approaches similar to the above one.

Acknowledgements We would like to thank the SNF Sino Swiss Science and Technology Cooperation Programme program under contract RiC 01-032014, NSFC 61473260/61673338, and the Swiss Re Institute, in particular Axel Mönkeberg, for discussions and financial support.

References

1. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating Embeddings for Modeling Multi-relational Data. In: NIPS. pp. 2787–2795 (2013)
2. Evgeniou, T., Pontil, M.: Regularized multi–task learning. In: KDD. pp. 109–117 (2004)
3. Kim, Y.: Convolutional Neural Networks for Sentence Classification. In: EMNLP. pp. 1746–1751. ACL (2014)
4. Le, Q.V., Mikolov, T.: Distributed Representations of Sentences and Documents. In: ICML. vol. 32, pp. 1188–1196. JMLR.org (2014)
5. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI. vol. 15, pp. 2181–2187 (2015)
6. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In: NIPS (2013)
7. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A Review of Relational Machine Learning for Knowledge Graphs. Proceedings of the IEEE 104(1), 11–33 (2016)
8. Nickel, M., Rosasco, L., Poggio, T.A.: Holographic Embeddings of Knowledge Graphs. In: AAAI. pp. 1955–1961. AAAI Press (2016)
9. Ristoski, P., Paulheim, H.: RDF2vec: RDF Graph Embeddings for Data Mining. In: ISWC (1). Lecture Notes in Computer Science, vol. 9981, pp. 498–514 (2016)
10. Saul, L.K., Roweis, S.T.: Think globally, fit locally: unsupervised learning of low dimensional manifolds. JMLR 4(Jun), 119–155 (2003)
11. Shi, B., Weninger, T.: Open-World Knowledge Graph Completion. CoRR abs/1711.03438 (2017)
12. Tang, D., Qin, B., Liu, T.: Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. In: EMNLP. pp. 1422–1432. ACL (2015)
13. Thoma, S., Rettinger, A., Both, F.: Towards holistic concept representations: Embedding relational knowledge, visual attributes, and distributional word semantics. In: International Semantic Web Conference. pp. 694–710. Springer (2017)
14. Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., Gamon, M.: Representing Text for Joint Embedding of Text and Knowledge Bases. In: EMNLP. pp. 1499–1509. ACL (2015)
15. Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., Bouchard, G.: Complex Embeddings for Simple Link Prediction. In: ICML. JMLR Workshop and Conference Proceedings, vol. 48, pp. 2071–2080. JMLR.org (2016)
16. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge Graph and Text Jointly Embedding. In: EMNLP. pp. 1591–1601. ACL (2014)
17. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI. pp. 1112–1119. AAAI Press (2014)
18. Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M.: Representation Learning of Knowledge Graphs with Entity Descriptions. In: AAAI. pp. 2659–2665. AAAI Press (2016)
19. Yamada, I., Shindo, H., Takeda, H., Takefuji, Y.: Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation. In: CoNLL. pp. 250–259. ACL (2016)
20. Yin, W., Schütze, H.: Learning word meta-embeddings. In: ACL (1). ACL (2016)
21. Zhong, H., Zhang, J., Wang, Z., Wan, H., Chen, Z.: Aligning Knowledge and Text Embeddings by Entity Descriptions. In: EMNLP. pp. 267–272. ACL (2015)