Learning Structured Embeddings of Knowledge Bases

Antoine Bordes[*]
Heudiasyc UMR CNRS 6599
Université de Technologie
Compiègne, France

Jason Weston
Google
111 8th Avenue
New York. NY, USA

Ronan Collobert
IDIAP
Rue Marconi 19
Martigny, Switzerland

Yoshua Bengio
Département IRO
Université de Montréal
Montréal. QC, Canada

Abstract

Many Knowledge Bases (KBs) are now readily available and encompass colossal quantities of information thanks to either a long-term funding effort (e.g. WordNet, OpenCyc) or a collaborative process (e.g. Freebase, DBpedia). However, each of them is based on a different rigorous symbolic framework which makes it hard to use their data in other systems. It is unfortunate because such rich structured knowledge might lead to a huge leap forward in many other areas of AI like natural language processing (word-sense disambiguation, natural language understanding, ...), vision (scene classification, image semantic annotation, ...) or collaborative filtering. In this paper, we present a learning process based on an innovative neural network architecture designed to embed any of these symbolic representations into a more flexible continuous vector space in which the original knowledge is kept and enhanced. These learnt embeddings would allow data from any KB to be easily used in recent machine learning methods for prediction and information retrieval. We illustrate our method on WordNet and Freebase and also present a way to adapt it to knowledge extraction from raw text.

Introduction

A fundamental challenge for AI has always been to be able to gather, organize and make intelligent use of the colossal amounts of information generated daily (Davis, Shrobe, & Szolovits 1993). Recent developments in this area concern the building of large web-based Knowledge Bases (KBs), special kinds of relational database especially designed for knowledge management, collection, and retrieval. Thanks to long-term funding efforts or collaborative processes, promising progress has been accomplished and several KBs, which encompass a huge amount of data regarding general and specific knowledge, are now readily available on-line: OpenCyc, WordNet, Freebase, DBpedia, etc.[1]

These KBs have been conceived for differing purposes, ranging from approaching human-like reasoning, producing an intuitively usable dictionary and thesaurus or proposing a global on-line information resource for *semantic web* applications. However, their highly-structured and organized data could also be useful in many other AI areas like in Natural Language Processing (NLP) for word-sense disambiguation or natural language understanding, in computer vision for scene classification or image semantic annotation, or in collaborative filtering. Even if WordNet is widely used in NLP (e.g. in (Ng & Cardie 2002; Snow *et al.* 2007)), this remains a small contribution compared to what could be achieved with such gigantic knowledge quantities. This could be explained by the fact that it is usually hard to take advantage of KBs data in other systems. Indeed, their underlying symbolic frameworks, whilst being very efficient for their original purposes, are not flexible enough to be fruitfully exported, especially to statistical learning approaches.

In this paper, we study an original way of leveraging the structured data encompassed by KBs into statistical learning systems. Our work is based on a model that learns to represent elements of any KB into a relatively low (e.g. 50) dimensional embedding vector space. The embeddings are established by a neural network whose particular architecture allows to integrate the original data structure within the learnt representations. More precisely, considering that a KB is defined by a set of entities and a set of relations between them, our model learns one embedding for each entity (i.e. one low dimensional vector) and one operator for each relation (i.e. a matrix). Furthermore, we show that using Kernel Density Estimation in the low dimensional embedding space allows one to estimate the probability density within that space, so that the likelihood of a relation between entities can be quantified, even when that tentative fact was not previously in the KB. Low dimensional spaces are appropriate for achieving good results with a density estimator because such an estimator can misbehave in high dimensions if there exists no smooth low-dimensional manifold capturing the distribution (Bengio, Larochelle, & Vincent 2006).

As we detail in the following, this new framework is appealing for several reasons: it is flexible (simple to adapt to many KBs), compact (only a low-dimension vector per entity and a low-dimension matrix per relation type to store) and also exhibits generalization ability (the ability to infer new relations from the existing ones in the KB). Moreover, such representations potentially allow integration of KBs within systems of the recent machine learning trend of Deep Learning (see (Bengio 2009) for a review), as, for instance, those concerning NLP (Collobert & Weston 2008).

[*]Work done while A. B. was visiting Université de Montréal.

[1]Respect. available from `cyc.com/opencyc`, `wordnet.princeton.edu`, `freebase.com` and `dbpedia.org`.

Table 1: **Statistics** of datasets used in this paper.

Dataset	Rel. types	Entities	Train ex	Test ex
WordNet	11	55,166	164,467	4,000
Freebase	13	81,061	356,517	4,000

This paper is organized as follows. We first define our framework and discuss some related work. Then we introduce our model as well as its training protocol and display an empirical illustration of its potential on data from WordNet and Freebase. Finally, we present how this method can be extended to automatically extract knowledge from text.

Framework

Definitions This work considers Knowledge Bases as graph models. This means that the data structure of KBs is not necessarily hierarchical, and is just defined by a set of nodes and a set of links. To each individual node of the graph corresponds an element of the database, which we term an *entity*, and each link defines a *relation* between entities. Relations are directed and there are typically several different kinds of relations. In the remainder of the paper, a relation is denoted by a triplet (e^l, r, e^r), where e^l is the *left* entity, e^r the *right* one and r the *type* of relation between them.

Even if some KBs – like Freebase– are directly organized in a graph model, the majority are based on an ontology. We are aware that transforming such complex hierarchical structures into a somewhat flat graph structure may cause a loss of information and could forbid the use of any reasoning and deduction protocol that could be applied on the original ontology. However, our major goal is to propose a way to take advantage of the content of KBs to be exploited in other systems, and not to replace their original framework. Our work can actually be seen as complementary.

Datasets We chose to illustrate our work on two KBs: WordNet and Freebase. WordNet is directly formulated as a graph. In this case we considered all the entities that were connected with the relation types given in Table 2, although we did remove some entities for which we have too little information (see below). Freebase is also directly formalized as a graph. However, for simplicity of the experiments we did not consider the whole of the KB (several millions of entities). Instead, we restricted ourselves to entities of the Freebase type *deceased people*, and considered the subgraph defined by all relations involving at least one entity of this type. For each dataset, we only kept triplets regarding entities involved in at least 3 relations and relation types appearing at least 5,000 times. We also created 2 corresponding test sets by randomly picking 4,000 triplets and withholding them at training time. These triplets contain entities appearing in other relations of the training set.

The final statistics of both datasets are given in Table 1 and the corresponding relation types in Table 2. Examples of relations appearing in the Freebase set are (*_marylin_monroe*, *_profession*, *_actress*), (*_pablo_picasso*, *_place_of_birth*, *_màlaga*) or (*_john_f_kennedy*, *_religion*, *_catholicism*), and in the WordNet set: (*_door_1*, *_has_part*, *_lock_2*), (*_brain_1*, *_type_of*, *_neural_structure_1*) or (*_auto_1*, *_has_instance*, *_s_u_v_1*). As WordNet is composed of words with different meanings, here we term its entities as the concatenation of the word and an number indicating which sense it refers to i.e. *_auto_1* is the entity encoding the first meaning of the word "auto".

Table 2: **Relation types** of KBs used in this paper.

Freebase	WordNet
_place_lived	*_synset_domain_topic*
_place_of_birth	*_domain_region*
_place_of_death	*_domain_topic*
_profession	*_has_part*
_spouse	*_part_of*
_parents	*_type_of*
_children	*_has_instance*
_religion	*_subordinate_instance_of*
_ethnicity	*_similar_to*
_gender	*_member_holonym*
_cause_of_death	*_member_meronym*
_nationality	
_education_institution	

Related Work

Sutskever, Salakhutdinov, & Tenenbaum proposed the closest existing work that we know of. They propose to train a factorized representation of relations in a nonparametric Bayesian clustering framework. Their method is forced to have multiple embeddings per entity which our experiments indicate might be bad for generalization. Indeed in this paper is written: "The disadvantage of using two vectors for each object is that the model cannot as easily capture the position-independent properties of the object, especially in the sparse regime.". *Linear Relational Embeddings* (Paccanaro & Hinton 2001) are also close to this work but with a different loss function and architecture and have only been applied to small arithmetic and "family relation" tasks.

Several works have been targeting to use KBs to improve performance. For example, the approach of Singh & Gordon aims at jointly modeling a relational database and an item-users rating matrix to improve collaborative filtering. In (Bückner *et al.* 2002), knowledge is encoded via semantic networks to improve signal processing. Most work of this kind regards natural language processing (NLP), either to improve word-sense disambiguation (McRoy 1992) coreference resolution (Ng & Cardie 2002) (using WordNet) or grammar induction (Naseem *et al.* 2010) (using a small base of linguistic knowledge). The present paper proposes a new tool to ease the use of KB data in other methods.

The embedding idea has been used in NLP via the framework of language models (Bengio *et al.* 2003; Bengio 2008) where an embedding per word is learnt. Collobert & Weston showed that such evolved representations help to improve performance on standard NLP tasks. A similar idea has been successfully applied by Weston, Bengio, & Usunier for matching queries and images, both mapped to a common semantic space, also leading to meaningful data representations and state-of-the-art results. Recently, Bordes *et al.*. adapted this model to a (very) small custom KB for language understanding. All these works demonstrate that encoding data in distributed embeddings induce gains in performance.

Embedding Knowledge Bases

Structured Embeddings

The main idea behind our structural embedding of KBs is the following.

- Entities can be modeled in a d-dimensional vector space, termed the "embedding space". The i^{th} entity is assigned a vector $E_i \in \mathbb{R}^d$.
- Within that embedding space, for any given relation type, there is a specific similarity measure that captures that relation between entities. For example, the *_part_of* relation would use one measure of similarity, whereas *_similar_to* would use another. Note that these similarities are not generally symmetric, as e.g. *_part_of* is not a symmetric relation. We model this by assigning for the k^{th} given relation a pair $R_k = (R_k^{lhs}, R_k^{rhs})$, where R_j^{lhs} and R_j^{rhs} are both $d \times d$ matrices. The similarity function for a given entity is thus defined as:

$$S_k(E_i, E_j) = ||R_k^{lhs} E_i - R_k^{rhs} E_j||_p$$

  using the p-norm. In this work we chose $p = 1$ due to the simplicity of the gradient learning in that case. That is, we transform the entity embedding vectors E_i and E_j by the corresponding left and right hand relation matrices for the relation R_k and then similarity is measured according to the 1-norm distance in the transformed embedding space.

Neural network architecture The above description can in fact be modeled as a kind of neural network, in particular it can be seen as a generalization of a siamese network (Bromley *et al.* 1993; Hadsell, Chopra, & LeCun 2006) which conventionally takes a pair of inputs and tries to learn a similarity measure. In our case, however, we are learning several similarity measures (one per relation type), and the similarity measures are potentially non-symmetric, which is not normally the case in siamese nets.

Let us now more formally define our problem. We are given a training set x of m triplets:

$$x_1 = (e_1^l, r_1, e_1^r), \ldots, x_m = (e_m^l, r_m, e_m^r),$$

where $(e_i^l, r_i, e_i^r) \in \mathcal{X} = \{1, \ldots, D_e\} \times \{1, \ldots, D_r\} \times \{1, \ldots, D_e\}$ and D_e is the number of possible entities in the database and D_r is the number of possible relations in the database. That is, the triplets index the dictionary of entities and relation types.

To find a *useful* embedding we must define a training objective that learns relationships. Our training objective is thus defined as follows: if one of the the left- or right-hand side entities of a given triplet were missing, then we would like our model to be able to predict the correct entity. For example, this would allow us to answer questions like "what is part of a car?" or "where was Audrey Hepburn born?". That is, we wish to learn a real-valued function $f(e_i^l, r_i, e_i^r)$ (which we will also equivalently write as $f(x_i)$) such that for any training triplet x_i:

$$f(e_i^l, r_i, e_i^r) < f(e_j^l, r_i, e_i^r), \ \ \forall j : (e_j^l, r_i, e_i^r) \notin x \qquad (1)$$

and

$$f(e_i^l, r_i, e_i^r) < f(e_i^l, r_i, e_j^r), \ \ \forall j : (e_i^l, r_i, e_j^r) \notin x. \qquad (2)$$

The function f is trained to rank the training samples below all other triplets in terms of 1-norm distance. It is parametrized by the following neural network:

$$f(e_i^l, r_i, e_i^r) = ||R_{r_i}^{lhs} Ev(e_i^l) - R_{r_i}^{rhs} Ev(e_i^r)||_1. \qquad (3)$$

R^{lhs} and R^{rhs} are both $d \times d \times D_r$ tensors, where e.g. R_i^{lhs} means to select the i^{th} component along the third dimension of R^{lhs}, resulting in a $d \times d$ matrix. E is a $d \times D_e$ matrix containing the embeddings of the D_e entities and the function $v(n) : \{1, \ldots, D_e\} \to \mathbb{R}^{D_e}$ maps the entity dictionary index n into a sparse vector of dimension D_e consisting of all zeros and a one in the n^{th} dimension. Hence, Equation (3) means: (1) select the $(e_i^l)^{th}$ and $(e_i^r)^{th}$ columns of E, (2) transform them by the $d \times d$ left- and right-hand side matrices of relation r_i, and (3) measure the 1-norm distance between these relation-transformed entity embeddings.

Note that the matrix E which contains the representations of the entities is learnt via a *multi-tasking* process because a single embedding matrix is used for all relations. The embedding of an entity contains factorized information coming from all the relations in which the entity is involved. So, for each entity, the model is forced to learn how it interacts with other entities with respect to all the types of relation.

This also causes our distributed representation to be rather cheap in memory and to have the ability of potentially scaling to large KBs. Indeed, dimensions of E are $d \times D_e$ and R^{lhs} and R^{rhs} are of dimension $d \times d \times D_r$ where d is usually small, e.g. 50 dimensional.

Training

To train the parameters R^{lhs}, R^{rhs} and E of our model we use stochastic gradient descent (SGD) (Robbins & Monro 1951) to optimize the constraints (1) and (2). That is, we iterate the following procedure:

1. Select a positive training triplet x_i at random.
2. Select at random either constraint type (1) or (2). If we chose the first constraint we select an entity $e^{neg} \in \{1, \ldots, D_e\}$ at random and construct a negative training triplet $x^{neg} = (e^{neg}, r_i, e_i^r)$ otherwise we construct $x^{neg} = (e_i^l, r_i, e^{neg})$ instead.
3. If $f(x_i) > f(x^{neg}) - 1$ then make a gradient step to minimize $\max(0, 1 - f(x^{neg}) + f(x_i))$.
4. Enforce the constraints that each column $||E_i|| = 1, \forall i$.

The above procedure is iterated for a given fixed number of iterations. The constant 1 used in step (3) is the margin as is commonly used in many margin-based models such as SVMs (Boser, Guyon, & Vapnik 1992). The gradient step requires a learning rate of λ. The normalization in step (4) helps remove scaling freedoms from our model (where, for example, E can be made smaller while R^{lhs} and R^{rhs} can be made larger and still give the same output).

Probability Landscape Estimation

The approach described above allows one to learn a distributed representation for any kind of KB data. However it has the weakness that it somewhat dilutes some crucial

Table 3: **Ranking**. Predicted ranks on WordNet (55,166 candidates) and Freebase (81,061 candidates).

		WordNet		Freebase
		rank e^l	rank e^r	rank e^r
COUNTS	*Train*	662.7	804.1	541.8
	Test	6202.3	5894.2	804.9
EMB	*Train*	16.2	23.3	–
	Test	3414.7	3380.8	–
EMB$_{MT}$	*Train*	13.6	20.9	2.9
	Test	97.3	223.0	317.2
EMB$_{MT}$+KDE	*Train*	**11.8**	**19.9**	**1.6**
	Test	**87.8**	**192.5**	**314.5**

Table 4: **Top 10**. Rate of predictions ranked in the top 10 elements on WordNet (55,166 cand.) and Freebase (81,061 cand.).

		WordNet		Freebase
		rank e^l	rank e^r	rank e^r
COUNTS	*Train*	5.0%	5.0%	0.4%
	Test	0.3%	1.3%	1.7%
EMB	*Train*	76.4%	75.7%	–
	Test	4.0%	4.1%	–
EMB$_{MT}$	*Train*	83.9%	82.0%	95.8%
	Test	**71.7%**	**76.7%**	14.0%
EMB$_{MT}$+KDE	*Train*	**88.1%**	**85.8%**	**99.2%**
	Test	64.2%	68.3%	**17.0%**

information which is given by the training triplets from the KB. Indeed, a key property of the symbolic framework of the original KBs is that one is certain that all existing relations are true facts. When we transfer this data in our embedding space we lose that guarantee because any triple, whether it exists in the original KB or not, is associated a distance value f given by Equation (3). Even if the training process is expected to decrease the f value on training points, we would like to emphasize their degree of certainty. Hence, after training the structured embeddings, we propose to estimate the probability density at any point of the defined embedding space using Kernel Density Estimation (KDE). Because KDE bases its estimation on the training points, this guarantees that they get a high probability density, which is exactly what we are looking for.

Our kernel density estimation uses the following Gaussian kernel which is defined for any pair of triplets (x_i, x_j) and is based on the learnt embeddings:

$$K(x_i,x_j)= \frac{1}{2\pi\sigma} \exp\left(\frac{-1}{2\sigma^2}\left(||R^{lhs}_{r_i} Ev(e^l_i) - R^{lhs}_{r_j} Ev(e^l_j)||^2_2 + ||R^{rhs}_{r_i} Ev(e^r_i) - R^{rhs}_{r_j} Ev(e^r_j)||^2_2\right)\right)$$

The kernel density estimator f_{kde} is then defined as

$$f_{kde}(x_i) = \frac{1}{|S(x_i)|} \sum_{x_j \in S(x_i)} K(x_i, x_j) \qquad (4)$$

with $S(x_i) = \{(e^l_j, r_j, e^r_j) \in x : r_j = r_i \wedge (e^l_j = e^l_i \vee e^r_j = e^r_i)\}$. Performing the density estimation using all the training triplets would be very costly (especially on KBs with hundreds of thousand or millions of entities); for that reason, for a given x', we compute $f_{kde}(x')$ only using those training triplets that share the same relation type and a similar entity (left or right).

Because the function f_{kde} can estimate the density for any triplet, it can also be used to perform predictions. Hence, the following formula can predict e^r corresponding to (r,e^l):

$$\hat{e}^r = \text{argmax}_{e \in D_e} f_{kde}((e^l, r, e)) \qquad (5)$$

Predictions of e^l given (r,e^r) or r given (e^l,e^r) can be carried out in a similar manner. In other words, Equation (5) can answer the questions "what is part of a car?" (using WordNet data and r =*_has_part* and e^l =*_auto_1*) and "where was Audrey Hepburn born?" (using Freebase data and r =*_place_of_birth* and e^l =*_audrey_hepburn*). Note that the same mechanism can be used to find multiple answers to the question, each associated with a score, and these scores can be converted into probabilities (by normalizing the scores by the sum of the scores over all possible symbolic answers). Predictions could also be done using Equation (3), but using f_{kde} instead of f, we hope that answers regarding facts asserted in the training set will be more certainly correct. For relations that are not in the training set, but our system believes may be correct, with f_{kde} we will obtain a degree of confidence in that assertion.

Empirical Evaluation

We now illustrate the properties of the structured embeddings via their application on WordNet and Freebase.

Ranking

We first assess the quality of our representations using the following ranking task: for all training and testing triplets (e^l,r,e^r), (1) we remove e^l, (2) we compute densities $f_{kde}((e, r, e^r))$ for all $e \in D_e$, (3) we sort values by decreasing order, and (4) we record the rank of the correct entity e^l. An identical process is repeated for predicting e^r. This setting somewhat corresponds to question answering.

We compare our method, denoted EMB$_{MT}$+KDE, with 3 counterparts which rank triplets with different procedures:

- EMB$_{MT}$ uses the same embeddings as EMB$_{MT}$+KDE but performs ranking with the 1-norm of Equation (3) (and sorts in decreasing order).
- EMB also ranks with Equation (3) but its embeddings have been learnt without multi-tasking i.e. there is a different matrix E for each type of relation. This is much more costly in memory and did not scale on Freebase.
- COUNTS does not perform any learning but only counts the number of times pairs (e^l, r) and (r,e^r) $\forall$ e^l, r and e^r appear in the training set. Triplets are then ranked according to the score composed by the sum of their 2 corresponding pairs.

Although we did not compare to a direct factorization method like (Sutskever, Salakhutdinov, & Tenenbaum 2009) our results comparing EMB to EMB$_{MT}$ suggest having a single embedding per entity performs better than having separate embeddings for left and right entities.

Table 5: **Generalization**. Lists of e^r (top) and e^l (bottom) predicted using EMB_{MT}+KDE after training on WordNet. We removed from the lists all elements from the training set (usually top-ranked): the predictions below are generalized by the system. Lists are displayed by decreasing triplet probability density order.

e^l	_everest_1	_brain_1
r	_part_of	_has_part
e^r	*_north_vietnam_1*	*_subthalamic_nucleus_1*
	_hindu_kush_1	*_cladode_1*
	_karakoram_1	*_subthalamus_1*
	_federal_2	*_fluid_ounce_1*
	_burma_1	*_sympathetic_nervous_system_1*
e^l	*_judgement_3*	*_thing_13*
	_delayed_action_1	*_transfer_5*
	_experience_5	*_situation_1*
	_bawl_out_1	*_illness_1*
	_carry_over_1	*_cognition_1*
r	_type_of	_has_instance
e^r	_deciding_1	_language_1

Tables 3 and 4 present the results on WordNet and Freebase (data statistics are given in Table 1) but use different metrics: Table 3 gives the mean rank over all examples while Table 4 provides the proportion of correct answers within the top 10 of the list. For training error, we only ranked on 4,000 random training triplets to save time. We do not report the ranking results concerning the prediction of e^l on Freebase because they are not meaningful. Indeed, they end up trying to answer questions like "who is of gender male?" or "who is American?" for which they are many correct answers. All methods based on embeddings share the same hyperparameters: $d = 50$,[2] $\lambda = 0.01$ and have been trained for 1.5×10^9 updates (which takes ≈ 3 days). For KDE, $\sigma^2 = 0.5$.

Results of Tables 3 and 4 show that EMB_{MT} and EMB_{MT}+KDE perform best. As expected KDE helps on training examples. The rank is almost perfect on Freebase and the values are misleading on WordNet. Indeed, the correct answer is not ranked on top all the time because some other training triplets happen to be as correct as the considered test example (i.e. an "auto" does not have a single part): if training examples are removed from the lists, the ranks on WordNet become 1.1 for e^l and 1.3 for e^r. Hence, KDE achieves its goal since EMB_{MT}+KDE replicates (almost) perfectly the training KB. However, it hurts a bit generalization for the top-10 on WordNet, because there is slightly too much emphasis on training data. Indeed we chose a small Gaussian width to ensure to encode it well but this is slightly detrimental for generalization.

Generalization

Tables 3 and 4 show that COUNTS can record some information about train examples but can not generalize. To some extent, EMB exhibits the same behavior since it can almost replicate the train, but is bad on test triplets. Since both EMB_{MT} and EMB_{MT}+KDE perform much better on test examples, we deduce that generalization can only be possible via multi-tasking. This allows to encode information coming from different relations in the embeddings of entities, which can then be exploited by relation operators. This is a kind of analogy process which seems to be more efficient on WordNet than on Freebase because the same entities appear in more different types of relations.

Table 5 illustrates this a bit more by displaying top ranked entities for 4 WordNet relations. Since we removed any training examples from these lists, these are analogies performed by the system. The chosen entities are not always exactly correct but do make sense most of the time.

Entity Embeddings

The matrix E is a crucial part of the structured representation because it factorizes information from all relations in which the entity appears. Table 6 shows some nearest neighboring entities within the embedding space defined by EMB_{MT}+KDE for WordNet and Freebase. Distances are measured directly on the embeddings using the 1-norm.

Entities which are close in that space, exhibit some similarities but, interestingly, these are quite complex. For instance, if *_lawn_tennis_1* is close to other sports, the list of *_artist_1* is rather heterogeneous with a kind of artist (*_singer_1*), professions interacting with artists (*_critic_1*, *_prospector_1*), a role (*_part_7*) and the condition of being an artist. This also happens with *_audrey_hepburn* who is associated with other persons sharing different facts with her (place of birth, profession,...). This table also illustrates that our method can learn a good representation for proper nouns, something which is usually quite hard for language models such as the one used in (Collobert & Weston 2008).

The two columns *_field_1* and *_field_2* finally exhibit two representations for homonyms, something which could be interesting for an application to word-sense disambiguation.

Knowledge Extraction from Raw Text

This section presents how this work can be adapted for knowledge extraction from raw text. Exciting progress has been made recently in this area (Snow *et al.* 2007; Carlson *et al.* 2010). These methods are able to define entities and relations between them from plain text. We believe that our structured embeddings could be of some interest for this task because, as for conventional KBs, they would provide distributed representations and probability density estimation. Furthermore, our training process scales well and, since it is based on SGD, is online. It could thus be conducted together with an incremental process such as the NELL project (Carlson *et al.* 2010).

To illustrate that ability, we conducted our own (relatively simple) knowledge extraction with the following protocol. First, using the software SENNA[3], we performed Semantic Role Labeling, (i.e. for each proposition, label each semantic arguments associated with a verb with its grammatical role) on 40,000 Wikipedia articles and gathered all annotated sentences. Second, we simplified this data by keeping only phrases labeled with semantic roles following the

[2] Performance is sensitive to d. Even if we did not perform an exhaustive search, we also tried $d = 100$ which works as well as $d = 50$ (but is more costly) and $d = 20$ which is slightly worse.

[3] Freely available from `ml.nec-labs.com/senna/`.

Table 6: **Embeddings**. Closest entities from those of the top row according to the L1 distance of their embeddings.

_lawn_tennis_1	_artist_1	_field_1	_field_2	_pablo_picasso	_audrey_hepburn	_painter	_stanford_university
_badminton_1	*_critic_1*	*_yard_9*	*_universal_set_1*	*_lin_liang*	*_wil_van_gogh*	*_artist*	*_univ._of_california*
_squash_4	*_part_7*	*_picnic_area_1*	*_diagonal_3*	*_zhou_fang*	*_signe_hasso*	*_printmaker*	*_city_univ._of_new_york*
_baseball_1	*_singer_1*	*_center_stage_1*	*_analysis_situs_1*	*_wu_guanzhong*	*_joyce_grenfell*	*_visual_artist*	*_stanford_law_school*
_cricket_2	*_prospector_1*	*_range_11*	*_positive_10*	*_paul_cezanne*	*_greta_garbo*	*_struct._engineer*	*_virginia_union_univ.*
_hockey_2	*_condition_3*	*_eden_1*	*_oblique_3*	*_yves_klein*	*_ingrid_bergman*	*_producer*	*_cornell_university*
WordNet data				**Freebase data**			

Table 7: **Knowledge extraction**. Examples of lists of e^r predicted with the embeddings learnt out of raw text for e^l *="people"*. Lists are displayed by decreasing triplet probability density order.

e^l	people				
r	build	destroy	won	suffer	control
e^r	*livelihoods*	*icons*	*emmy*	*sores*	*rocket*
	homes	*virtue*	*award*	*agitation*	*stores*
	altars	*donkeys*	*everything*	*treatise*	*emotions*
	houses	*cowboy*	*standings*	*eczema*	*spending*
	ramps	*chimpanzees*	*pounds*	*copd*	*fertility*

scheme *subject-verb-direct object*, by removing adjectives, adverbs and pronouns from the *subject* and *direct object* noun phrases and by stemming the *verb*. Finally, we created a data set with all the "cleaned" *subject-verb-direct object* triplets coming from one of the 100 most frequents verb forms. This left us with 154,438 triplets, with 100 relation types (verbs) and 23,936 entities (nouns).

We then learned embeddings using the same training scheme as described before, which easily scaled to this task. Table 7 presents an illustration of the kind of result we obtained: a vector-based representation and a probability density have been associated with these simple facts.

Conclusion

This paper has introduced a new process to automatically learn structured distributed embeddings of Knowledge Bases. These new representations are compact and can be efficiently trained on KBs with hundred of thousands of entities and hundreds of relations. Furthermore, using KDE in the embedding space allows to estimate the probability density of any relation between entities. Experiments on two large KBs have shown that our distributed encoding made of entity vectors and relation operators preserves the knowledge of the original data, and presents the interesting ability of generalizing to new reasonable relations. Finally, we showed how we can adapt our approach on raw text for knowledge extraction.

This leaves open promising directions for future work including (i) multi-tasking from multiple KBs and raw text to learn a combined knowledge set in a single embedding space and (ii) using this learnt knowledge for other AI tasks.

References

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *JMLR* 3:1137–1155.

Bengio, Y.; Larochelle, H.; and Vincent, P. 2006. Non-local manifold parzen windows. In *Adv. in Neur. Inf. Proc. Syst. 18*.

Bengio, Y. 2008. Neural net language models. *Scholarpedia* 3(1):3881.

Bengio, Y. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning* 2(1):1–127.

Bordes, A.; Usunier, N.; Collobert, R.; and Weston, J. 2010. Towards understanding situated natural language. In *Proc. of the 13th Intern. Conf. on Artif. Intel. and Stat.*, volume 9, 65–72.

Boser, B.; Guyon, I.; and Vapnik, V. 1992. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory* 144–152.

Bromley, J.; Guyon, I.; LeCun, Y.; Sackinger, E.; and Shah, R. 1993. Signature verification using a siamese time delay neural network. In *Adv. in Neur. Inform. Process. Syst. 6*.

Bückner, J.; Pahl, M.; Stahlhut, O.; and Liedtke, C. 2002. A knowledge-based system for context dependent evaluation of remote sensing data. In *Pattern Recognition*, volume 2449. 58–65.

Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B.; Jr., E. R. H.; and Mitchell, T. M. 2010. Toward an architecture for never-ending language learning. In *Proc. of the 24th Conf. on Artif. Intel.*

Collobert, R., and Weston, J. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proc. of the 25th Inter. Conf. on Mach. Learn.*

Davis, R.; Shrobe, H.; and Szolovits, P. 1993. What is a knowledge representation? *AI Magazine* 14(1):17–33.

Hadsell, R.; Chopra, S.; and LeCun, Y. 2006. Dimensionality reduction by learning an invariant mapping. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'06)*.

McRoy, S. W. 1992. Using multiple knowledge sources for word sense discrimination. *Computatinal Linguistics* 18:1–30.

Naseem, T.; Chen, H.; Barzilay, R.; and Johnson, M. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proc. of EMNLP '10*, 1234–1244.

Ng, V., and Cardie, C. 2002. Improving machine learning approaches to coreference resolution. In *Proc. of ACL'02*, 104–111.

Paccanaro, A., and Hinton, G. 2001. Learning distributed representations of concepts using linear relational embedding. *IEEE Trans. on Knowl. and Data Eng.* 13:232–244.

Robbins, H., and Monro, S. 1951. A stochastic approximation method. *Annals of Mathematical Statistics* 22:400–407.

Singh, A. P., and Gordon, G. J. 2008. Relational learning via collective matrix factorization. In *Proc. of SIGKDD'08*, 650–658.

Snow, R.; Prakash, S.; Jurafsky, D.; and Ng, A. Y. 2007. Learning to merge word senses. In *Proceedings of EMNLP'07*, 1005–1014.

Sutskever, I.; Salakhutdinov, R.; and Tenenbaum, J. 2009. Modelling relational data using bayesian clustered tensor factorization. In *Adv. in Neur. Inf. Proc. Syst. 22*.

Weston, J.; Bengio, S.; and Usunier, N. 2010. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine Learning* 81:21–35.