


RESEARCH

Open Access



# Relation prediction in knowledge graph by Multi-Label Deep Neural Network

Yohei Onuki<sup>1\*</sup> , Tsuyoshi Murata<sup>1</sup>, Shun Nukui<sup>1</sup>, Seiya Inagi<sup>2</sup>, Xule Qiu<sup>2</sup>, Masao Watanabe<sup>2</sup> and Hiroshi Okamoto<sup>2</sup>

\*Correspondence:

[oonuki@net.c.titech.ac.jp](mailto:oonuki@net.c.titech.ac.jp)

<sup>1</sup>Department of Computer Science,  
School of Computing, Tokyo  
Institute of Technology, W8-59  
2-12-1 Ookayama, 152-8552  
Meguro, Tokyo, Japan

Full list of author information is  
available at the end of the article

## Abstract

Knowledge graph will be useful for the intelligent system. As the relationship prediction on the knowledge graph becomes accurate, construction of a knowledge graph and detection of erroneous information included in a knowledge graph can be performed more conveniently. The goal of our research is to predict a relation (predicate) of two given Knowledge Graph (KG) entities (subject and object). Link prediction between entities is important for developing large-scale ontologies and for KG completion. TransE and TransR have been proposed as the methods for such a prediction. However, TransE and TransR embed both entities and relations in the same (or different) semantic space(s). In this research we propose a simple architecture model with emphasis on relation prediction by using a Multi-Label Deep Neural Network (DNN), and developed KGML. KGML embeds entities only; given subject and object are embedded and concatenated to predict probability distribution of predicates. Since the output of KGML is the probability distribution in  $[0, 1]$ , output can be classified as positive and negative by using the threshold of 0.5. Since the output of the existing method TransE is the score in  $[0, \infty)$ , the threshold value must be calculated each time. Experimental results showed that predictions by KGML are more accurate than those by TransE and TransR. KGML is more accurate than DKRL which uses both KG triples and entity descriptions for learning. KGML is more accurate than PTransE in and its learning speed is faster than PTransE. The code of KGML is available at <https://github.com/yo0826jp/KGML>.

**Keywords:** Knowledge graph, DNN, Embedding, TransE, Link prediction

## Introduction

Ontology learning is one of the important topics for developing the Semantic Web. In general, there are many entity pairs where the relations between them are unknown (Kavalec and Svatek 2003; Weichselbraun et al. 2010). If we can predict such relations accurately, we can augment a given ontology. Since many Semantic Web data (such as Google's Knowledge Graph) are already available, techniques for predicting relations between entities are important for developing large-scale ontologies.

The goal of our research is to predict relations between two given entities in Knowledge Graph (KG) accurately. KG is a kind of semantic network, and each triple in KG is composed of three entities (a subject, a predicate, and an object). A subject and an object are entities, and a predicate is the relation between the entities. Suppose (Tokyo, is-capital-of, Japan) is an example of such a triple. We would like to predict "is-capital-of"

when “Tokyo” and “Japan” are given. For this purpose, we propose a method for predicting a predicate from a subject and an object by using a Deep Neural Network (DNN), and developed KGML.

### Outline

Freebase, Wordnet and Wikidata are used as the datasets of our experiments. The following experiments are performed:

“**Comparison with previous methods**” section comparison with previous methods

“**Learning speed**” section Learning speed

“**Hyperparameter  $\alpha$  and prediction accuracy**” section Hyperparameter  $\alpha$  and prediction accuracy

“**Failure analysis**” section Failure analysis

“**Embedding dimension and prediction accuracy**” section embedding dimension and prediction accuracy

“**Embedding dimension and computational time**” section embedding dimension and computational time

As the results of our experiments, KGML is more accurate than TransE (Bordes et al. 2013) and TransR (Lin et al. 2015) for predicting a predicate from given subject and object. Although KGML learns from KG triples only, its prediction accuracy is better than that of DKRL (Xie et al. 2016) which uses both KG triples and entity descriptions for learning. KGML’s prediction accuracy is better than that of PTransE (Lin et al. 2015) in its accuracy and its learning speed is faster than PTransE. We also propose a method for finding appropriate embedding dimensions in KGML. KGML is an improved version of our previous method RDFDNN (Yohei et al. 2017).

### Previous methods for predicting relations

#### TransE

TransE (Bordes et al. 2013) embeds both entities and relations in the same vector space. Based on vector operations of entities and relations, TransE predicts  $t$  from given  $h$  and  $l$ , and predicts  $l$  from  $h$  and  $t$ . It generates the vector space that satisfies the following equation:

$$d(h + l, t) = 0, \quad (1)$$

where  $d$  is the function of Euclidean distance between two given vectors.

TransE obtains vector representation of entities and relations by minimizing the following objective function  $L$  by gradient descent:

$$L = \sum_{(h,l,t) \in S} \sum_{(h',l,t') \in S'} \max(\gamma + d(h + l, t) - d(h' + l, t'), 0), \quad (2)$$

where  $\gamma$  is the margin for training,  $S$  is the set of triples in the dataset,  $S'_{(h,l,t)}$  is the set of random sampled triples,  $E$  is the set of entities, and  $S'_{(h,l,t)}$  is defined as follows:

$$S'_{(h,l,t)} = \{(h', l, t) | h' \in E\} \cap \{(h, l, t') | t' \in E\}. \quad (3)$$

$S'_{(h,l,t)}$  contains a small number of positive triple, but many are negative triple because Knowledge graph is a sparse data set.

**TransR**

TransR (Lin et al. 2015) is the extension of TransE, and it has an ability to learn 1-to-N relations, which is not possible for TransE. 1-to-N relation means that there are more than one  $t$ s for a given pair of  $h$  and  $l$ , such as (John, likes, pizza) and (John, likes, hamburger). In the case of TransE, learning 1-to-N relations is not possible because there is only one vector that satisfies  $d(h + l, t) = 0$ . In the above example, both pizza and hamburger are represented as the same vector, which is the problem of TransE.

In the case of TransR,  $h$  and  $t$  are mapped by means of the transformation matrix  $M_l$  which is unique to  $l$  and then vector operation is performed in order to avoid the above problem. TransR generates a vector space that satisfies the following equation:

$$d(hM_l + l, tM_l) = 0, \tag{4}$$

where  $M_l$  is the transformation matrix corresponding to relation  $l$ . TransR accepts vector representations of the entities obtained by TransE as its initial values, and it minimizes the following objective function  $L$  by gradient descent in order to obtain vectors and matrices corresponding each relation:

$$L = \sum_{(h,l,t) \in S} \sum_{(h',l',t') \in S'} \max(\gamma + d(h_l + l, t_l) - d(h'M_l + l, t'M_l), 0), \tag{5}$$

where  $\gamma$  is the margin for training,  $S$  is the set of triples in the dataset,  $S'$  is the set of random sampled triples,  $h_l = hM_l$  and  $t_l = tM_l$ .  $S'$  contains a small number of positive triple, but many are negative triple because Knowledge graph is a sparse data set.

**DKRL**

Description-Embodied Knowledge Representation Learning (DKRL) (Xie et al. 2016) is a method for learning the embedding of KG taking advantages of entity descriptions. Since it learns embedding from both KG triples and external entity descriptions, prediction of novel entities with descriptions only (zero-shot scenario) is possible. DKRL employs two encoders (continuous bag-of-words (CBOW) model and deep convolutional neural model) to represent semantics of entity descriptions. Input to DKRL is the representation of keywords in the explanation of entities learned by word2vec, and its output is the representation of entities. Representations of entities and relations obtained by TransE are used as the initial values of DKRL. DKRL performs learning by minimizing the following  $L$  by stochastic gradient descent:

$$L = \sum_{(h,l,t) \in S} \sum_{(h',l',t') \in S'} \max(\gamma + d(h + l, t) - d(h' + l', t'), 0), \tag{6}$$

where  $\gamma$  is the margin,  $S$  is the set of triples in given dataset,  $S'$  is the set of random sampled triples,  $E$  is the set of entities in the dataset, and  $R$  is the set of relations in the dataset, respectively.  $S'$  is defined as follows:

$$S' = \{(h', l, t) | h' \in E\} \cup \{(h, l, t') | t' \in E\} \cup \{(h, l', t) | l' \in R\}. \tag{7}$$

$S'$  contains a small number of positive triple, but many are negative triple because Knowledge graph is a sparse data set.

### PTransE

Path-based TransE (PTransE) (Lin et al. 2015) regards relation paths as translations between entities for representation learning. PTransE obtains vector representation of entities and relations by minimizing the following objective function  $L$  by gradient descent:

$$L = E(h, l, t) + E(h, P(h, t), t), \quad (8)$$

where  $E(h, l, t)$  is the same as TransE's objective function and  $P(h, t)$  is a set of paths between  $h$  and  $t$ .  $E(h, P(h, t), t)$  is the inference correlations between relations with multiple step relation path triples, which is defined as follows:

$$E(h, P(h, t), t) = \frac{1}{Z} \sum_{p \in P(h, t)} R(p|h, t) E(h, p, t), \quad (9)$$

where  $R(p|h, t)$  indicates the reliability of the relation path  $p$  given the entity pair  $(h, t)$  and  $Z = \sum_{p \in P(h, t)} R(p|h, t)$  is a normalization factor.

### Other approaches

Our attempt for obtaining low dimensional embeddings is related to DeepWalk (Perozzi et al. 2014) and LINE (Tang et al. 2015). However, these approaches focus on simple networks composed of only one type of relation, while our target is KG triples composed of several types of relations.

Ristoski et al. propose RDF2Vec (Ristoski and Paulheim 2016), which embeds KG graph based on graph walk and Weisfeiler-Lehman Subtree KG Graph Kernel. The goal of RDF2Vec is to obtain general-purpose low dimensional embeddings for arbitrary data mining algorithms, and experimental results show their superiority over other basic methods (Naive Bayes, k-Nearest Neighbors, C4.5, SVM, Linear Regression, and M5Rules) for the tasks of classification and regression. On the other hand, the goal of our paper is to predict entity relations accurately.

Wang et al. propose a method for incorporating inference rules into embedding models for accurate predictions (Wang et al. 2015). Physical and logical rules are employed in order to impose constraints on candidate facts. Wang et al. also propose a method for ranking relations between entities (Wang et al. 2016). Relations highly correlated to each other are detected first, and then multi-task learning is performed in order to couple the predictions of the relations. Guo et al. propose rule-enhanced relation learning (Guo et al. 2016). The method uses rules to refine the results obtained by previous embedding methods such as TransE. Xiao et al. propose a generative model for knowledge graph embeddings (Xiao et al. 2016). The model focuses on multiple meanings of a relation, and it has abilities of detecting latent semantics for a relation. Xie et al. propose a method called Type-embodied Knowledge Representation Learning (TKRL) (Xie et al. 2016) to take advantages of hierarchical entity types in KG. Although these approaches are rather different from ours, our work on relation prediction can be integrated to these approaches for the goal of KG completion.

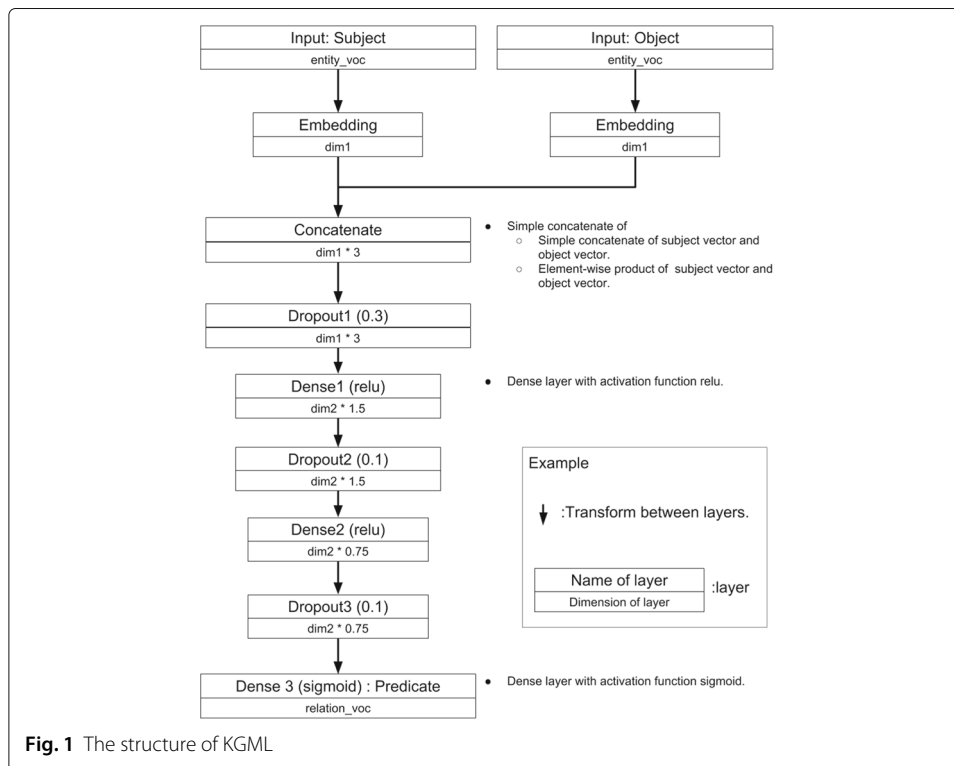
Although KG is a type of multiplex network, there is also a method of link prediction with a small multiplex network compared to KG. Xu et al. propose MTNE (Xu et al. 2017), MTNE improves the embedding using common parts between layers after enforcing the entity for each layer. De Bacco et al. propose MULTITENSOR (De Bacco

et al. 2017) a method for link prediction of multiplex networks by tensor decomposition. Matsuno et al. propose a method MELL (Ryuta and Tsuyoshi 2018) that introduces a layer vector and expresses the features of layers to link prediction by embedding. These methods are designed with an emphasis on relation prediction in a small network compared to KG. For example, 7153 triples are present in the dataset that these methods use in experiments, with the largest number of triples. On the other hand, 151,442 triples exist even if the triple number is the smallest among the KGs targeted in this study. For this reason, it is difficult to predict using these methods in a large-scale network such as KG from the question of learning time. On the other hand, if we use the method proposed this time, we can also predict the relationship in a small network targeted by these methods. Therefore, in order to examine how effectively the proposed method works in datasets other than KG, it can be compared with KGML.

**Proposed method:KGML**

Since the output of the existing method is the score in  $[0, \infty)$ , the threshold value must be calculated each time. We propose a more practical relationship prediction method by creating a model with a fixed threshold. Multi-label DNN and multi-class DNN are the candidates of the models that can use a fixed threshold. In relation prediction, there can be multiple relations between two entities. For example, (“Tokyo”, “is-capital-of”, “Japan”) and (“Tokyo”, “is-city-of”, “Japan”). Multi-label DNN has one or more positive outputs, while multi-class DNN has only one positive output. Therefore, we employ multi-label DNN for relation prediction.

The structure of KGML is shown in Fig. 1. When  $h$  and  $t$  of a KG triple  $(h, l, t)$  are given as inputs, KGML will output  $l$ . Rectangles are layers of DNN, and arrows are the



transformation between layers. Its inputs are one-hot codes of  $h$  and  $t$ , and its output is the probability distribution of the n-hot codes of  $l$ . One-hot code is a sequence of bits for representing entities. For example, the following vector of length  $n$  (whose  $i$ -th bit is 1 and others are 0) is the representation of  $i$ -th entity among  $n$  entities.

$$\left( 0 \ 0 \ \dots \ 0 \ 1 \ 0 \ \dots \ 0 \right) \tag{10}$$

$entity\_voc$  and  $relation\_voc$  are the numbers of entities and relations, respectively.  $dim1$  is the embedding dimensions of entity.  $dim2$  is a hyper parameter to determine dimension of layer  $Dense2$  and  $Dense3$ . Embedding is the transformation from KG entities to their vector representations. The length of the transformed vector is called embedding dimension.

As shown in Fig. 1, there are eight hidden layers in KGML: *Embedding*, *Dropout1*, *Dense1*, *Dropout2*, *Dense2*, *Dropout3* and *Dense3*. *Dense1* and *Dense2* are the dense layers for activation by Relu function, and *Dense3* is the layers for activation by sigmoid function. *Concatenate* is the composition of the embeddings of  $h$  and  $t$ . We employ simple concatenation of two embedding vectors and element-wise multiplication of two embedding vectors for the concat together. *Embedding* is the layer for transforming an entity to its embedding. The two *Embedding* layers share weights and biases. L2 normalization is used for the layer of *Embedding*. *Dropout1*, *Dropout2* and *Dropout3* are dropout layers to suppress overfitting.

The architecture of KGML is determined based on experiments. KGML has 3 dense layers. When it was reduced to 2, the expressive power is lacking, and the prediction accuracy decreased. When the number is increased to 4, the prediction accuracy is hardly improved, and the learning time is greatly extended because the model is complicated. The dimensions of each layer were determined based on experiments. The dropout rate of the dropout layer was determined based on experiments. The prediction accuracy was higher if we made one higher and make the others smaller than equalizing the dropout rates of the dropout layer.

Since the output of KGML is the probability distribution of one-hot representation of relation  $l$ , the following two objective functions are used as the objective function for learning KGML:

$$L_{bce}(h, t, l) = - \sum_{i \in relation\_voc} l_i \log P(h, t)_i - \sum_{i \in relation\_voc} (1 - l_i) \log(1 - P(h, t)_i), \tag{11}$$

$$L_{dwbce}(h, t, l) = - w_{pos} \sum_{i \in relation\_voc} l_i \log P(h, t)_i - w_{neg} \sum_{i \in relation\_voc} (1 - l_i) \log(1 - P(h, t)_i), \tag{12}$$

where

$$w_{pos} = \frac{\sum_{i \in relation\_voc} l_i}{relation\_voc} \tag{13}$$

and

$$w_{neg} = 1 - w_{pos}. \tag{14}$$

$S$  is the set of triples in training data of knowledge graph,  $relation\_voc$  is the set of relations in knowledge graph,  $P(h, t)$  is the output of KGML when  $h$  and  $t$  are given as its inputs,  $l$  is the binary representation of training data and  $i$  is the integer index that satisfies  $0 \leq i < relation\_voc$ .  $L_{bce}$  is binary cross entropy. Binary cross entropy is the loss function used in multi-label classification.  $L_{dwbce}$  is dynamic weighted binary cross entropy. We add two weights  $w_{pos}$  and  $w_{neg}$  into binary cross entropy for adjust imbalance between the number of positive samples and the number of negative samples.  $w_{pos}$  and  $w_{neg}$  are calculated for each training triple. In training with  $L_{dwbce}$ , we expect that the recall improvement takes precedence over the improvement in precision. KGML use the following loss function as the objective function:

$$L_{bce}(h, t, l) = \alpha L_{dwbce}(h, t, l) + (1 - \alpha)L_{bce}(h, t, l), \quad (15)$$

where  $\alpha$  is a hyper parameter in  $[0, 1]$ .  $\alpha$  is a balancing parameter between  $L_{dwbce}$  and  $L_{bce}$ . We expected that decreasing  $\alpha$  increases precision and decreases recall.  $L_{dwbce}$  is a loss function that emphasizes recall. Since the knowledge graph is a sparse data set, the value of  $w_{pos}$  becomes large and the value of  $w_{neg}$  becomes small. When using the loss function of  $L_{dwbce}$ , prioritizing the output of the positive example to 1 over giving the output of the negative example closer to 0 is given priority. As the optimizer of the above objective function, Adam (Kingma and Ba 2015) is used.

Existing methods such as TransE, TransR, DKRL and PTransE use negative sampling for learning. Negative sampling uses one randomly generated negative sample for each positive sample of training data. Therefore, the existing method has the same number of positive and negative samples. We are not able to introduce the  $\alpha$  method proposed in KGML into the existing method because there is no imbalance between positive and negative samples in existing methods. In KGML an imbalance occurs between the positive and negative examples because all triples  $(h, l', t)$  are used as a negative examples for each positive sample  $(h, l, t)$  in train data.  $l'$  is relation that satisfies  $l \neq l'$  and triple  $(h, l', t)$  is not in train data.

KGML embeds entities only; subjects and objects are embedded and concatenated to predict probability distribution of predicates. If we can embed both entities and relations accurately, the embedding can be used for entity prediction and relation prediction. Since entity embedding is enough for predicting relations, KGML focuses on entity embedding. Although KGML cannot predict an object from a predicate and a subject, it has abilities of predicting a predicate accurately from a subject and an object.

## Evaluation

### Dataset

In our experiments, we have used the FB15k, WN18, WD40k and WD40k\_nl, which are the samples of the following four datasets. Details of FB15k, WN18, WD40k and WD40knl are shown in Table 1. FB15k and WN18 are the same as the ones used in the experiments of previous research (Bordes et al. 2013; Lin et al. 2015). In addition to FB15k and WN18, we created WD40k and WD40k\_nl from wikidata (Vrandečić and Kröttsch 2014). In Table 1, density is defined by the following equation:

$$density = \frac{\#training\ triples + \#validation\ triples + \#testing\ triple}{relation\_voc \times entity\_voc \times (entity\_voc - 1)} \quad (16)$$



**Table 1** Details of FB15k, WN18, WD40k and WD40k\_nl

	FB15k	WN18	WD40k	WD40k_nl
Original data	Freebase	Wordnet	Wikidata	wikidata
Number of entities (entity_voc)	14,951	40,943	40,000	40,000
Number of relations (relation_voc)	1,345	18	568	568
Number of triples for training	483,142	141,442	193,043	193,043
Number of triples for validation	50,000	5,000	19,461	19,461
Number of triples for testing	59,071	5,000	19,370	13,456
Density	$1.980 \times 10^{-6}$	$5.019 \times 10^{-6}$	$2.551 \times 10^{-7}$	$2.551 \times 10^{-7}$
% Test Linked	80.9	94.0	30.5	0.0

Original datasets of FB15k, WN18, WD40k and WD40k\_nl are as follows:

**Freebase (Bollacker et al. 2008) (FB15k)**

A large collaborative online knowledge base.

**Wordnet (Miller 1995) (WN18)**

A large lexical database of English.

**Wikidata (Vrandečić and Krötzsch 2014) (WD40k and WD40k\_nl)**

A large collaborative online knowledge base. Relations of wikidata are abstracted and organized based on Freebase's relations.

FB15k, WN18 and WD40k contain triplets in the training set that are simply the inverse of triples contained in the test set (Dettmers et al. 2018; Toutanova et al. 2015); a hypothetical example would be the training set containing (Japan, has-capital, Tokyo) and the test set containing (Tokyo, is-capital-of, Japan). It called test set leakage. In addition, since there is a knowledge graph that includes (Japan, has-capital, Tokyo) but does not include (Tokyo, is-capital-of, Japan). In this reason, experiments with data set with test set leakage has significance. The causes of test set leakage are triples  $(h, l, t)$  that satisfy the following.  $(h, l, t)$  is included in the test set, and  $(h, l', t)$  or  $(t, l', h)$  is included in the training set. However,  $l'$  means a relation different from  $l$ . As the last row (% Test Linked) of the Table 1 shows, such triples occupy 80.9% of the test set of FB15k (Toutanova and Chen 2015). It is less compared to 30.5% in WD40k. For this reason, in this study, we made the data set WD40k\_nl from WD40k. WD40k\_nl is the WD40k's test set minus triples that can cause testset leakage. WD40k\_nl doesn't include test set leakage.

**Criteria for Evaluation**

We have implemented KGML using keras and TensorFlow. Keras (<https://keras.io>) is a Python-based library executable on TensorFlow and Theano. Training of DNN by keras is done using CuDNN library on GPU. We use Python, keras and TensorFlow for the implementation. The CPU used in our experiments is 28 x Intel Xeon E5-2680 V4 Processor and 4 x Tesla P100 for NVLink-Optimized Servers. We used a super computer TSUBAME3.0 in Tokyo Institute of Technology.

We evaluated the results by  $Hits@k$ , Precision and Recall. We used  $Hits@k$  for comparison with existing methods. For  $Hits@k$ , after 50 epochs of training,  $h$  and  $t$  of a triple in the test data are given to KGML as input, and its output  $l$  is evaluated by  $Hits@k$ , whose value is one if the correct answer is included in top-k plausible outputs, and is zero otherwise. All test data are used and results are averaged for evaluation. As with experiments in TransE, we used two settings, Raw and Filterd (Filt). In Raw setting, evaluation is done



by normal  $Hits@k$ . In Filt setting, if the model answered wrong relation  $l_w$  for given entities  $h$  and  $t$ , and the triplet  $(h, l_w, t)$  is in training triples set, the answer  $l_w$  is ignored and next answered relation for given entities  $h$  and  $t$  is evaluated.

In the link prediction in the knowledge graph, the positive example is extremely small compared with the negative examples. For this reason, metrics like AUC and accuracy had a problem that the score was too big. For example, the AUC scores of PTransE and KGML exceeded 0.99 for both datasets. We used metrics that evaluate with emphasis on small positive examples such as  $hits@1$  that focus on only the top one.

### Setting for comparison

For the comparison of accuracy with previous methods, we set the parameters as  $dim1 = 100, dim2 = 100$  for FB15k, WN18, WD40k and WD40k\_nl and  $\alpha = (0.0, 0.5, 1.0)$  for FB15k, WN18, WD40k and WD40k\_nl. These parameters are empirically determined through experiments using training set and validation set. For failure analysis, we set the parameter as  $dim1 = 100, dim2 = 100, \alpha = 1.0$  for WD40k. For the experiments of embedding dimension and accuracy, parameters  $dim1$  and  $dim2$  are set as each of 2, 4, 6, 8 and 10 for FB15k. For the experiments of relations between embedded dimension and computational time,  $dim1$  is set to 60, 120, 180, and 240, and  $dim2$  is set to 20, 40, 60, and 80 for FB15k. We trained KGML by 50 epochs. Learning rate  $lr$  for optimizer adam gradually decreased.  $l$  decreased for every 10 epochs as follows: 0.001, 0.00033, 0.0001, 0.000033, 0.00001. The learning rate,  $dim1$  and  $dim2$  are empirically determined by experiments using validation sets.

We have compared KGML with TransE, TransR and PTransE implemented by previous research (Lin et al. 2015). For TransE, the learning rate is set to 0.01,  $\gamma$  is set to 1, and embedding dimensions are set to 50 for FB15k, and 100 for WN18 and WD40k, respectively. For TransR, the learning rate is set to 0.001,  $\gamma$  is set to 1, and embedding dimensions are set to 50 for FB15k, and 100 for WN18 and WD40k, respectively. For PTransE, the learning rate is set to 0.01,  $\gamma$  is set to 1, and embedding dimensions are set to 100 for FB15k, WD40k and WD40k\_nl. Path size of PTransE is 2-step and setting is ADD setting.

When  $h$  and  $t$  are given, TransE computes  $d(t-h, l)$  for all possible relations and selects the relation  $l$  of its minimum value as its prediction. This is because the vector space satisfying  $d(t-h, l) = 0$  is generated in TransE, so the relation  $l$  that takes the minimum value of  $d(t-h, l)$  for given  $h$  and  $t$  is expected to constitute a valid triple  $(h, l, t)$  rather than other relations.

When  $h$  and  $t$  are given, TransR computes  $d(tM_l - hM_l, l)$  for all possible relations and select the relation  $l$  of its minimum value as its prediction. This is because the vector space satisfying  $d(tM_l - hM_l, l) = 0$  is generated in TransR, so the relation  $l$  that takes the minimum value of  $d(tM_l - hM_l, l)$  for given  $h$  and  $t$  is expected to constitute valid triple  $(h, l, t)$  rather than other relations.

### Comparison with previous methods

For the comparison of accuracy with previous methods, we set the parameters as  $dim1 = 100, dim2 = 100$  for FB15k, WN18 and WD40k and  $\alpha = (0.0, 0.5, 1.0)$  for FB15k, WN18 and WD40k.

The codes and FB15k dataset available at <https://github.com/xrb92/DKRL> are used for the comparison with DKRL. We use the same parameters as in (Xie et al. 2016) for our

experiments, and evaluated by *Hits@k*. Experiments of DKRL are performed only for FB15k dataset because entity descriptions are not available for WN18 dataset and WD40k dataset.

Table 2 is the results of comparison with the FB15k, WN18, WD40k and WD40k\_nl datasets, respectively. As shown in these figures, KGML is more accurate than TransE and TransR in all datasets. There is a large difference between the prediction accuracy of KGML, TransE and TransR at the results of FB15k and WD40k data. In the following discussion, we will discuss the results of FB15k and WD40k data. The reason for the large difference between the prediction accuracy of KGML, TransE and TransR is that the numbers of relations in FB15k (1345) and WD40k (568) are much more than that in WN18 (18). Prediction of relations is harder when the number of relations is much more.

Comparing the prediction accuracy of PTransE and KGML in WD40k and WD40k\_nl shows that there is almost no difference. Prediction accuracy is improved in all row settings. From following results, the influence of test set leakage is extremely small for PTransE and KGML.

Column FB15k in Table 2 shows the results of KGML, TransE, TransR, DKRL and PTransE. Although DKRL uses both KG triples and entity descriptions for learning, KGML is more accurate than DKRL. We can claim that KGML is widely applicable for accurate prediction of entity relations even when entity descriptions are not available. KGML is slightly better than PTransE with predictive accuracy, and KGML is superior in the learning speed of the model. We will discuss about this in “[Learning speed](#)” section. Column WD40k in Table 2 shows the results of KGML, TransE, TransR and PTransE. Some relations in FB15k are unified in WD40k. For example, there are “members of the sports team” and “members of the cabinet” separately in FB15k, but these are unified as “member of” in WD40k. In WD40k, the descriptive text of each entity is so short that it is impossible to perform experiment with DKRL. Compared with FB15k, TransE and TransR have significantly lower prediction accuracy in WD40k. This is because WD40k is a sparse network compared with FB15k, so that TransE and TransR cannot do enough learning. For PTransE and KGML, the accuracy is improved because WD40k has a smaller number of relation and PTransE and FB15k can learn efficiently from less data. In KGML, the pattern of entity can be found by sharing the embedding layer of head entity and tail entity. *Hits@10* scores of PTransE at all dataset and *hits@10* scores of KGML at all dataset both exceeded 0.99. For this reason, we did not consider PTransE and KGML at *hits@10*.

**Table 2** Hits@1

Method	hits@1							
	FB15k		WN18		WD40k		WD40k_nl	
	raw	filt	raw	filt	raw	filt	raw	filt
TransE	0.651	0.843	0.586	0.586	0.201	0.205	-	-
TransR	0.421	0.502	0.713	0.714	0.0951	0.0981	-	-
PTransE	0.695	0.936	-	-	0.858	0.890	0.863	0.893
DKRL	0.685	0.872	-	-	-	-	-	-
KGML ( $\alpha = 1.0$ )	<u>0.725</u>	0.943	<u>0.975</u>	<u>0.980</u>	0.873	0.907	0.907	0.908
KGML ( $\alpha = 0.5$ )	<u>0.725</u>	0.933	0.972	0.977	0.877	0.911	0.910	0.911
KGML ( $\alpha = 0.0$ )	0.667	<u>0.963</u>	<u>0.975</u>	0.979	<u>0.879</u>	<u>0.914</u>	<u>0.914</u>	<u>0.916</u>

Underlining in the table means the best accuracy in each method

Consider the  $hits@1$  score of KGML in each dataset. The score at WN18 is high compared with the score at FB15k and the score at WD40k.  $relation\_voc$  of WN18 is considerably smaller than  $relation\_voc$  of FB15k and  $relation\_voc$  of WD40k. This means that the smaller the value of  $relation\_voc$ , the larger the  $hits@1$  score. The value of  $relation\_voc$  of WD40k is smaller than the value of  $relation\_voc$  of FB15k, but the score of  $hits@1$  is about the same. This is because WD40k is sparser in density than FB15k. Training is more difficult in sparse data sets.  $Hits@1$  score gets lower if the dataset is sparse.  $Hits@1$  of KGML depends on the value of  $\alpha$ . We will discuss about this in “Hyperparameter  $\alpha$  and prediction accuracy” section.

### Learning speed

#### Learning speed

In “Comparison with previous methods” section, we perform learning of 50 epochs for KGML and 1000 epochs for PTransE. It takes 27 s for one epoch in KGML and 6.5 s for one epoch in PTransE. KGML takes 1343 s for training and PTransE takes 6509 s for training. This means that KGML’s learning speed is faster than PTransE.

#### Convergence speed of learning

We compared the convergence speed of learning of KGML and PTransE. In this experiment, we perform learning of 50 epochs for KGML and 1000 epochs for PTransE in order to compare the convergence speed with the same length of time. We introduce convergence degree  $C$  which increases as learning converges.  $C$  is expected to be within the range  $[0, 1]$  but sometimes  $C > 1$  because loss of objective function in learning process becomes smaller than loss of objective function in the last epoch. We defined the convergence degree  $C$  in epoch  $e$  as follows:

$$C(e) = \frac{Loss(e_{last})}{Loss(e)}, \quad (17)$$

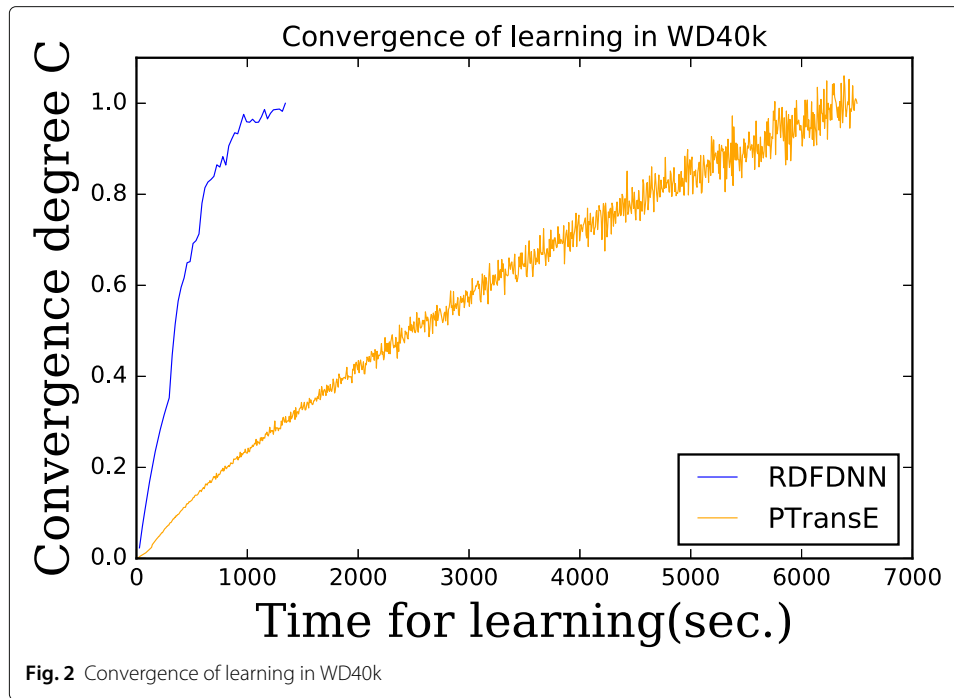
where  $Loss(e)$  is loss of objective function in epoch  $e$  and  $Loss(e_{last})$  is loss of objective function in the last epoch. Convergence degree  $C$  increases as learning converges because the value of loss of objective function  $Loss(e)$  decreases as learning converges.

Figure 2 shows the learning convergence of KGML and PTransE in WD40k. The X-axis is time for learning, and the Y-axis is convergence degree  $C$ . The larger the value in the Y-axis is, the more learning converges. Figure 2 shows that learning of KGML converges more quickly than learning of PTransE. This is because KGML does not use paths for learning, so it does not need much computation.

### Further experiment of KGML

In “Hyperparameter  $\alpha$  and prediction accuracy” section, we use  $dim1 = 100$ ,  $dim2 = 100$  and the used datasets are FB15k, WN18 and WD40k. In “Failure analysis” section, we use  $dim1 = 100$ ,  $dim2 = 100$ ,  $\alpha = 1.0$  and the used dataset is WD40k. In “Embedding dimension and prediction accuracy” section and “Embedding dimension and computational time” section, we use  $\alpha = 1.0$ , and the used dataset is WD40k.

We evaluate results by  $Recall_{test}$  and  $Precision$ . In FB15k, WN18 and WD40k, since the number of training data is larger than the number test data, there is a problem that over-fitting can not be detected when evaluating by  $Recall$ . We evaluate the result with  $Recall_{test}$  instead of  $Recall$ :



$$Recall_{test} = \frac{TP_{test}}{TP_{test} + FN_{test}}, \tag{18}$$

where  $TP_{test}$  is number of true positive in test data and  $FN_{test}$  is number of false negative in test data. Threshold for the output is 0.5.

We can not evaluate previous methods with  $Recall_{test}$  and  $Precision$  because previous methods has not fixed threshold. In principle the median score of the score distribution the other methods generate can be used to set a cutoff for define the threshold of existing methods. The WD40k source used in this study, wikidata, contains approximately 18,000 relationships. In order to perform cutoff using the median score of the score distribution, it is necessary to perform all 18,000 calculations. We think there are practical problems in making predictions for all 18,000 relationships. Since comparison with existing methods has already been made at hits @ 1, this section evaluated KGML only.

**Hyperparameter  $\alpha$  and prediction accuracy**

Table 3 shows change in  $Recall_{test}$  and  $Precision$  when hyperparameter  $\alpha$  is changed. We change the hyper parameter  $\alpha$  as (1.0, 0.5, 0.0) and we observe the change of  $Recall_{test}$  and  $Precision$ . In WN18,  $Recall_{test}$  and  $Precision$  does not change even if  $\alpha$  is changed. This is because WN18 has a small number of relations and relation prediction is not so difficult. In FB15k and WD40k, decreasing  $\alpha$  increases  $Precision$  and decreases  $Recall_{test}$ . As shown

**Table 3**  $Recall_{test}$  and  $Precision$

Method	FB15k		WN18		WD40k	
	$Recall_{test}$	$Precision$	$Recall_{test}$	$Precision$	$Recall_{test}$	$Precision$
KGML( $\alpha = 1.0$ )	0.981	0.848	<u>0.974</u>	<u>0.978</u>	<u>0.913</u>	0.897
KGML( $\alpha = 0.5$ )	<u>0.986</u>	0.809	0.973	0.976	0.899	0.908
KGML( $\alpha = 0.0$ )	0.829	<u>0.964</u>	<u>0.974</u>	<u>0.978</u>	0.880	<u>0.912</u>

Underlining in the table means the best accuracy in each method

in Eq 15, we expected that decreasing  $\alpha$  increases precision and decreases recall. The result of case study is as we expected.

In Table 2, the smaller  $\alpha$ , the higher  $hits@1$ . It is because the *Precision* is more important than  $Recall_{test}$  for high  $hits@1$  score. For example, there is only one positive data in test data and it is predicted as 0.49. In  $Recall_{test}$ , 0.49 is negative because  $0.49 < 0.5$  and score is 0. In  $hits@1$ , if there is no output larger than 0.49, score is 1 because  $hits@1$  is ranking evaluation.

### Failure analysis

KGML's false positive predictions can be classified to the following four categories.

- A: deceived by majority cases
- B: structurally similar
- C: complete failure

For this failure analysis, 100 triples of KGML failures are randomly sampled. Then the triples are manually evaluated and classified into the above three categories in order to obtain the results in Table 4.

As shown in Table 4, the most frequent failure is type C, complete failure. As an example of type C, KGML's prediction of relation between "Norway" and "Eurocontrol" is "place of death", while its correct answer is "member of". This kind of misprediction can be removed easily, because "place of death" can not be relation between nation and organization if we use filtering by meta knowledge.

The second most frequent failure is type A, deceived by majority cases. One of the examples is the prediction of the relation between "Yvelines (location)" and "Bailly (location)". The correct answer should be "contains administrative territorial entity", but the prediction by KGML was "shares border with". "shares border with" is the most frequent one for the relation between tow locations.

The least frequent failure is type B, structually similar. Structually similar means that the abstract representation of relation A is the same as the abstract representation of relation B. This failure means that KGML recognizes structural similarity between relations. As an example of this type, KGML predicts the relation between "Generalissimo (word)" and "Brockhaus and Efron Encyclopedic Dictionary" as "residence", while its correct answer is "described by source". "residence" and "described by source" can be paraphrased as "in". From the above failure analysis, we can say that even when KGML failed, more than half of its failed prediction are complete failure but easy to remove. Other failed prediction are valid in some sense (in types A and B).

### Embedding dimension and prediction accuracy

For the dataset WD40k, we set parameters  $dim1$  and  $dim2$  as each of 4, 8, 12, 16 and 20, and observed the  $Recall_{test}$  and *Precision* of KGML.

**Table 4** Types and the numbers of failed predictions

Type	Number of failures
A	25
B	9
C	<u>66</u>

Underlining in the table means the best accuracy in each method

Figures 3 and 4 are the results when  $dim1$  and  $dim2$  are set to each of 4, 8, 12, 16 and 20. In Fig. 3, The X-axis is  $dim1$ , and the Y-axis is  $Recall_{test}$ . In Fig. 4, The X-axis is  $dim1$ , and the Y-axis is  $Precision$ . Results of the same  $dim2$  with five different values of  $dim1$  are drawn in a line. As shown in Figs. 3 and 4,  $dim2$  is more important for the accuracy of KGML than  $dim1$ .

**Embedding dimension and computational time**

For the dataset WD40k, we set parameters  $dim1$  and  $dim2$  as each of 4, 8, 12, 16 and 20, and observed the the computational time of KGML.

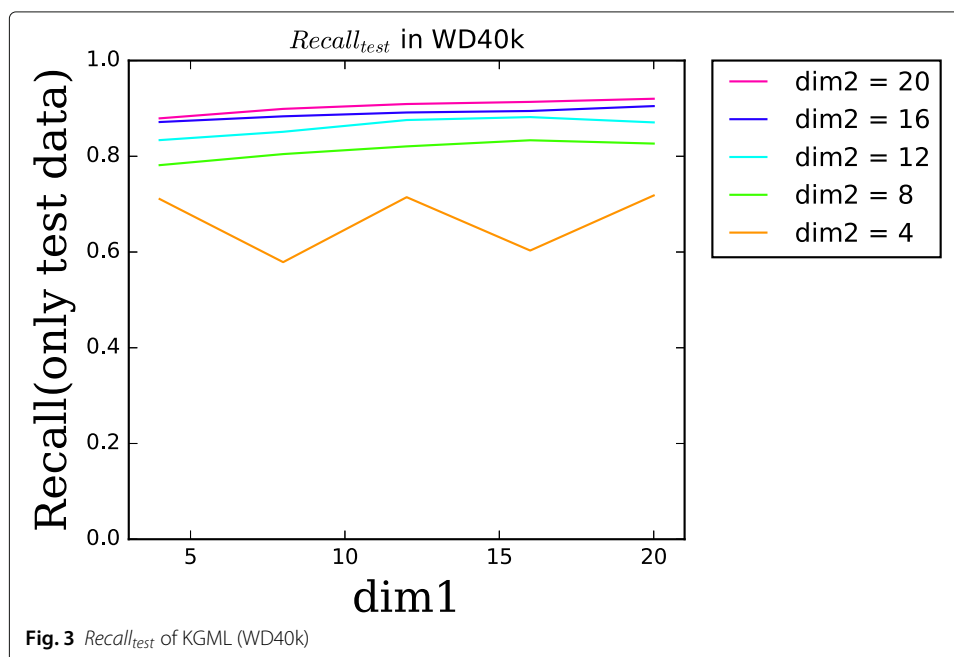
Figure 5 shows the computational times of KGML for different embedding dimensions. The X-axis is  $dim1$ , and the Y-axis is the computational time per one epoch (seconds). As shown in the figure, the value of  $dim2$  is not relevant to the computational time of KGML. Only the value of  $dim1$  is relevant to the computational time. As shown in Table 1,  $entity\_voc \gg relation\_voc$  in WD40k, which is the reason that  $dim1$  is relevant to the computational time of KGML because  $dim1$  is the embedding dimensions of entity.

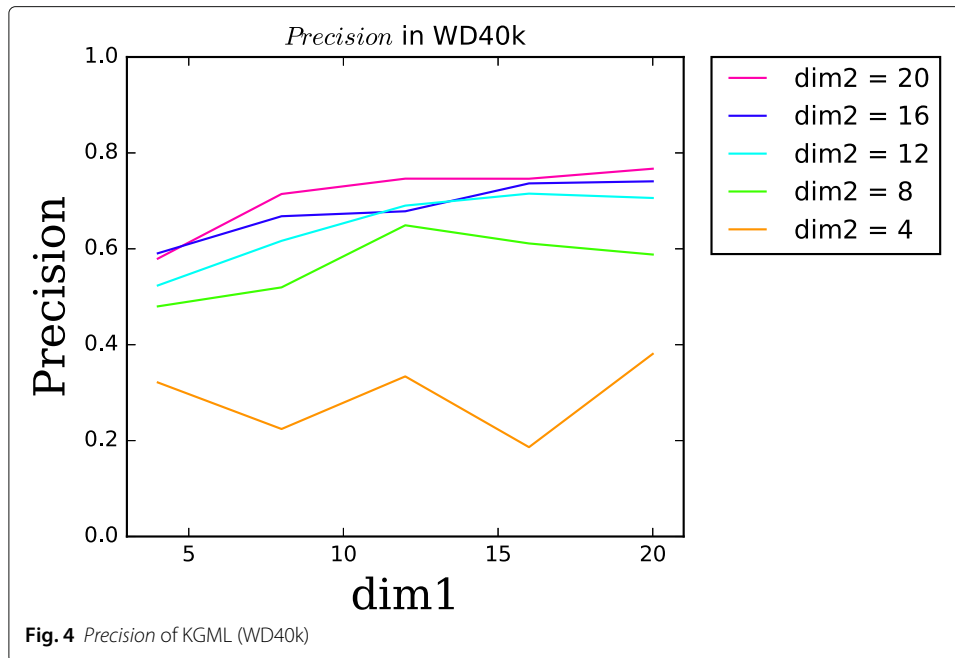
**Discussion**

As shown in “Comparison with previous methods” section, KGML is more accurate than previous methods for the prediction of relations between two given entities. The difference of accuracy with FB15k and WD40k is bigger than that with WN18, so we can conclude that prediction is harder when there are more possible relations.

As shown in “Hyperparameter  $\alpha$  and prediction accuracy” section, decreasing  $\alpha$  increases  $Precision$  and decreases  $Recall_{test}$ . In KGML, although there is a trade-off relationship between  $Precision$  and  $Recall_{test}$ , it can be controlled by hyper parameter  $\alpha$ .

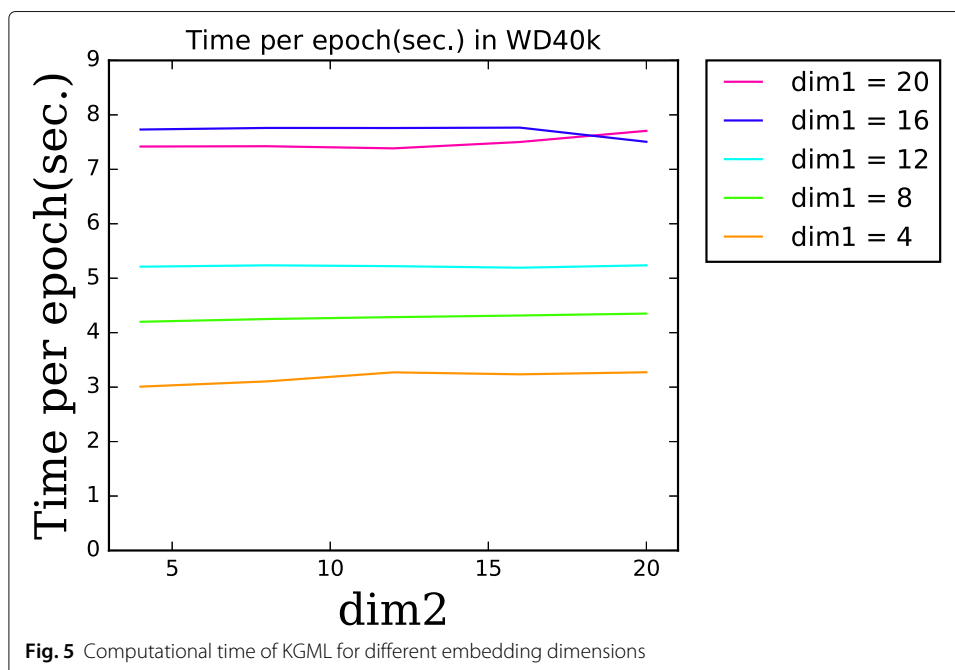
As shown in “Embedding dimension and prediction accuracy” section,  $dim2$  is more important for the accuracy of KGML than  $dim1$ . As shown in “Embedding dimension and computational time” section, decreasing  $dim1$  shortens the learning time. From the above,





in order to achieve high prediction accuracy with short training time, hyperparameters *dim1* and *dim2* should be searched as follows:

1. Set the parameter *dim1* as halves of those in previous methods as initial setting.
2. Set the parameter *dim2* = *dim1*
3. Until prediction accuracy is saturated, set them as twice of previous values, keeping *dim1* = *dim2*.





4. While prediction accuracy does not decrease, fix *dim2* and set *dim1* as the half of previous value.

The reason for setting parameters as half or twice of previous values is that the range of possible embedding dimensions is fairly wide, so repeated bipartitioning will be desirable for finding better parameters with less trials.

In this research, the parameters *dim1* and *dim2* of KGML were determined as follows. According to step 4 of the method of determining *dim1* and *dim2*, when *dim1* was reduced from 100 to 50, the prediction accuracy was greatly reduced. For this reason, *dim1* has never been halved, and then *dim1* and *dim2* are equal.

## Conclusion

In this paper, we propose KGML for predicting relations of KG from two given entities. KGML is more accurate compared with TransE and TransR. KGML is more accurate to DKRL which uses both KG triples and entity descriptions for learning. KGML is more accurate to PTransE in and its learning speed is faster than PTransE. The characteristics of KGML are as follows.

- Decreasing  $\alpha$  increases *Precision* and decreases *Recall<sub>test</sub>*
- Bigger *dim2* for better accuracy
- Smaller *dim1* for less computational time

The followings are left for our future work: prediction of *t* from given *h* and *l* is not easy for KGML, while it is possible for TransE and TransR.

There are attempts for generating semantic topic networks (Osborne and Motta 2015) and for evaluating computational semantic analysis systems (SIGLEX 2017). Relation prediction by KGML is the first step to contribute to the community of the Semantic Web.

## Acknowledgements

This work was supported by Tokyo Tech - Fuji Xerox Cooperative Research (Project Code KY260195), JSPS Grant-in-Aid for Scientific Research(B) (Grant Number 17H01785) and JST CREST (Grant Number JPMJCR1687).

## Funding

This work was supported by Tokyo Tech - Fuji Xerox Cooperative Research (Project Code KY260195), JSPS Grant-in-Aid for Scientific Research(B) (Grant Number 17H01785) and JST CREST (Grant Number JPMJCR1687).

## Availability of data and materials

FB15k and WN18 are the same as the ones used in the experiments of previous research (Bordes et al. 2013; Lin et al. 2015). FB15k and WN18 are available at <https://github.com/thunlp/KB2E>. We created WD40k from wikidata (Vrandečić and Kröttsch 2014). WD40k is available at <https://github.com/yo0826jp/KGML>.

## Authors' contributions

Nukui invented an idea of our method. Onuki developed our method and interpreted the result of experimental result, and was a major contributor in writing the manuscript. All authors read and approved the final manuscript.

## Competing interests

The authors declare that they have no competing interests.

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

## Author details

<sup>1</sup>Department of Computer Science, School of Computing, Tokyo Institute of Technology, W8-59 2-12-1 Ookayama, 152-8552 Meguro, Tokyo, Japan. <sup>2</sup>Research & Technology Group, Fuji Xerox Co., Ltd., 6-1 Minatomirai, Nishi-ku, 220-8668 Yokohama, Kanagawa, Japan.

Received: 4 November 2018 Accepted: 9 April 2019

Published online: 02 May 2019

## References

- Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J (2008) Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, Vancouver. pp 1247–1250
- Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O (2013) Translating Embeddings for Modeling Multi-relational Data. Part Adv Neural Inf Process Syst 26(NIPS'13):2787–2795
- De Bacco C, Power EA, Larremore DB, Moore C (2017) Community detection, link prediction, and layer interdependence in multilayer networks. *Phys Rev E* 95:042317
- Dettmers T, Minervini P, Stenetorp P, Riedel S (2018) Convolutional 2d knowledge graph embeddings. In: Thirty-Second AAAI Conference on Artificial Intelligence. Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, Louisiana
- Google Freebase Data Dumps. <https://developers.google.com/freebase/data>
- Google Google Knowledge Graph Search API - Google Developers. <https://developers.google.com/knowledge-graph/>
- Guo S, Ding B, Wang Q, Wang L, Wang B (2016) Knowledge Base Completion via Rule-Enhanced Relational Learning. In: China Conference on Knowledge Graph and Semantic Computing (CCKS 2016). Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence Buenos Aires, Argentina. pp 219–227
- Hinton GE, Osindero S, Teh Y-W (2006) A fast learning algorithm for deep belief nets. *Neural Comput* 18(7):1527–1554
- Kavalec M, Svatek V (2003) A Study on Automated Relation Labelling in Ontology Learning. In: Ontology Learning from Text: Methods, Evaluation and Applications. IOS Press, Nieuwe Hemweg
- Kingma D, Ba J (2015) Adam: A Method for Stochastic Optimization. In: Proceedings of the 3rd International Conference for Learning Representations (ICLR'15). San Diego
- Lassila O, Swick RR Resource Description Framework(RDF) Model and Syntax Specification. <https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>
- Lin Y, Liu Z, Luan H, Sun M, Rao S, Liu S (2015) Modeling relation paths for representation learning of knowledge bases 1506:00379. arXiv preprint arXiv. <https://arxiv.org/abs/1506.00379>
- Lin Y, Liu Z, Sun M, Liu Y, Zhu X (2015) Learning Entity and Relation Embeddings for Knowledge Graph Completion. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15). pp 2181–2187
- Miller GA (1995) WordNet: A Lexical Database for English. *Commun ACM* 38(11):39–41
- Osborne F, Motta E (2015) Klink-2: Integrating Multiple Web Sources to Generate Semantic Topic Networks. In: Proceedings of International Semantic Web Conference (ISWC 2015). Springer, Cham Vol. 2015. pp 408–424
- Perozzi B, Al-Rfou R, Skiena S (2014) DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, New York. pp 701–710
- Ristoski P, Paulheim H (2016) RDF2Vec: RDF Graph Embeddings for Data Mining. In: Proceedings of International Semantic Web Conference, Bethlehem Vol. 2016. pp 498–514
- Ryuta M, Tsuyoshi M (2018) MELL: Effective Embedding Method for Multiplex Networks. In: WWW'18 Companion Proceedings of the The Web Conference. Proceedings of the Web Conference, Lyon. pp 1261–1268
- SIGLEX (2017) International Workshop on Semantic Evaluation (SemEval-2017). <http://alt.qcri.org/semeval2017/>
- Tang J, Meng Q, Wang M, Zhang M, Yan J, Mei Q (2015) LINE: Large-scale Information Network Embedding. In: Proceedings of the 24th International Conference on World Wide Web, Florence. pp 1067–1077
- Toutanova K, Chen D (2015) Observed Versus Latent Features for Knowledge Base and Text Inference. In: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, Beijing. pp 57–66
- Toutanova K, Chen D, Pantel P, Poon H, Choudhury P, Gamon M (2015) Representing text for joint embedding of text and knowledge bases. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon. pp 1499–1509
- Vrandečić D, Krötzsch M (2014) Wikidata: a free collaborative knowledgebase. *Commun ACM* 57(10):78–85
- Wang Q, Liu J, Luo Y, Wang B, Lin C-Y (2016) Knowledge Base Completion via Coupled Path Ranking. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin. pp 1308–1318
- Wang Q, Wang B, Guo L (2015) Knowledge Base Completion Using Embeddings and Rules. In: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15). pp 1859–1865
- Weichselbraun A, Wohlgenannt G, Scharl A (2010) Refining non-taxonomic relation labels with external structured data to support ontology learning. *Data Knowl Eng* 69(8):763–778
- Xiao H, Huang M, Zhu X (2016) TransG: A Generative Model for Knowledge Graph Embedding. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin. pp 2316–2325
- Xie R, Liu Z, Jia J, Luan H, Sun M (2016) Representation Learning of Knowledge Graphs with Entity Descriptions. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix. pp 2659–2665
- Xie R, Liu Z, Sun M (2016) Representation Learning of Knowledge Graphs with Hierarchical Types. In: Proceedings of the 25th International Joint Conference on Artificial Intelligence, New York. pp 2965–2971
- Xu L, Wei X, Cao J, Yu PS (2017) Multitask (2017) network embedding. In: Proceedings of the 2017 IEEE International Conference on Data Science and Advanced Analytics, Tokyo. pp 571–580
- Yohei O, Tsuyoshi M, Shun N, Seiya I, Xule Q, Masao W, Hiroshi O (2017) Predicting relations of embedded RDF entities by Deep Neural Network. In: Proceedings of the International semantic web conference 2017 Posters and Demonstrations Track, Vienna