MDE: MULTIPLE DISTANCE EMBEDDINGS FOR LINK PREDICTION IN KNOWLEDGE GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Over the past decade, knowledge graphs became popular for capturing structured domain knowledge. Relational learning models enable the prediction of missing links inside knowledge graphs. More specifically, latent distance approaches model the relationships among entities via a distance between latent representations. Translating embedding models (e.g., TransE) are among the most popular latent distance approaches which use one distance function to learn multiple relation patterns. However, they are mostly inefficient in capturing symmetric relations since the representation vector norm for all the symmetric relations becomes equal to zero. They also lose information when learning relations with reflexive patterns since they become symmetric and transitive. We propose the Multiple Distance Embedding model (MDE) that addresses these limitations and a framework which enables collaborative combinations of latent distance-based terms (MDE). Our solution is based on two principles: 1) using limit-based loss instead of margin ranking loss and 2) by learning independent embedding vectors for each of terms we can collectively train and predict using contradicting distance terms. We further demonstrate that MDE allows modeling relations with (anti)symmetry, inversion, and composition patterns. We propose MDE as a neural network model which allows us to map non-linear relations between the embedding vectors and the expected output of the score function. Our empirical results show that MDE outperforms the state-of-the-art embedding models on several benchmark datasets.

1 INTRODUCTION

While machine learning methods conventionally model functions given sample inputs and outputs, a subset of Statistical Relational Learning (SRL) (De Raedt, 2008; Nickel et al., 2015) approaches specifically aim to model "things" (entities) and relations between them. These methods usually model human knowledge which is structured in the form of multi-relational Knowledge Graphs (KG). KGs allow semantically rich queries and are used in search engines, natural language processing (NLP) and dialog systems. However, they usually miss many of the true relations (West et al., 2014), therefore, the prediction of missing links/relations in KGs is a crucial challenge for SRL approaches.

A KG usually consists of a set of facts. A fact is a triple (head, relation, tail) where heads and tails are called entities. Among the SRL models, distance-based KG embeddings are popular because of their simplicity, their low number of parameters, and their efficiency on large scale datasets. Specifically, their simplicity allows integrating them into many models. Previous studies have integrated them with logical rule embeddings (Guo et al., 2016), have adopted them to encode temporal information (Jiang et al., 2016) and have applied them to find equivalent entities between multi-language datasets (Muhao et al., 2017).

Soon after the introduction of the first multi-relational distance-based method TransE (Bordes et al., 2013) it was acknowledged that it is inefficient in learning of symmetric relations, since the norm of the representation vector for all the symmetric relations in the KG becomes close to zero. This means the model cannot distinguish well between different symmetric relations in a KG.

To extend this model many variations are studied afterwards, e.g., TransH (Wang et al., 2014b), TransR (Lin et al., 2015b), TransD (Ji et al., 2015), and STransE (Dat et al., 2016). Even though they solved the issue of symmetric relations, they introduced a new problem: these models were no longer efficient in learning the inversion and composition relation patterns that originally TransE could handle. Besides, as noted in (Kazemi & Poole, 2018; Sun et al., 2019), within the family of distance-based embeddings, usually reflexive relations are forced to become symmetric and transitive. In this study, we take advantage of independent vector representations of vectors that enable us to view the same relations from different aspects and put forward a translation-based model that addresses these limitations and allows the learning of all three relation patterns.

In addition, we address the issue of the limit-based loss function in finding an optimal limit and suggest an updating limit loss function to be used complementary to the current limit-based loss function which has fixed limits.

Moreover, we frame our model into a neural network structure that allows it to learn non-linear patterns between embedding vectors and the expected output which substantially improves the generalization power of the model in link prediction tasks.

The model performs well in the empirical evaluations, improving upon the state-of-the-art results in link prediction benchmarks. Since our approach involves several elements that model the relations between entities as the geometric distance of vectors from different views, we dubbed it **m**ultiple-**d**istance **e**mbeddings (MDE).

2 Background and Notation

Given the set of all entities $\mathcal{E}$ and the set of all relations $\mathcal{R}$, we formally define a fact as a triple of the form $(\mathbf{h}, \mathbf{r}, \mathbf{t})$ in which $\mathbf{h}$ is the head and $\mathbf{t}$ is the tail, $\mathbf{h}, \mathbf{t} \in \mathcal{E}$ and $\mathbf{r} \in \mathcal{R}$ is a relation. A knowledge graph $\mathcal{KG}$ is a subset of all true facts $\mathcal{KG} \subset \zeta$ and is represented by a set of triples. An embedding is a mapping from an entity or a relation to their latent representation. A latent representation is usually a (set of) vector(s), a matrix or a tensor of numbers. A relational learning model is made of an embedding function and a prediction function that given a triple $(\mathbf{h}, \mathbf{r}, \mathbf{t})$ it determines if $(\mathbf{h}, \mathbf{r}, \mathbf{t}) \in \zeta$. We represent the embedding representation of an entity $\mathbf{h}$ with a lowercase letter h if it is a vector and with an uppercase letter H if it is a matrix. The ability to encode different patterns in the relations can show the generalization power of a model:

Definition 1. A relation r is symmetric (antisymmetric) if $\forall x, y$

$$r(x, y) \Rightarrow r(y, x) \;\; (\; r(x, y) \Rightarrow \neg r(y, x) \;).$$

A clause with such a structure has a symmetry (antisymmetry) pattern.

Definition 2. A relation r_1 is inverse to relation r_2 if $\forall x, y$

$$r_2(x, y) \Rightarrow r_1(y, x).$$

A clause with such a form has an inversion pattern.

Definition 3. A relation r_1 is composed of relation r_2 and relation r_3 if $\forall x, y, z$

$$r_2(x, y) \wedge r_3(y, z) \Rightarrow r_1(x, z)$$

A clause with such a form has a composition pattern.

3 Related Work

Tensor Factorization and Multiplicative Models define the score of triples via pairwise multiplication of embeddings. DistMult (Yang et al., 2015) simply multiplies the embedding vectors of a triple element by element $\langle h, r, t \rangle$ as the score function. Since multiplication of real numbers is symmetric, DistMult can not distinguish displacement of head relation and tail entities and therefore, it can not model anti-symmetric relations.

ComplEx (Trouillon et al., 2016) solves the issue of DistMult by the idea that the complex conjugate of the tail makes it non-symmetric. By introducing complex-valued embeddings instead of real-valued embeddings to DistMult, the score of a triple in ComplEx is $Re(h^\top diag(r)\bar{t})$ with $\bar{t}$ the

conjugate of t and $Re(.)$ is the real part of a complex value. ComplEx is not efficient in encoding composition rules (Sun et al., 2019). In RESCAL (Nickel et al., 2011) instead of a vector, a matrix represents the relation r, and performs outer products of h and t vectors to this matrix so that its score function becomes $h^\top Rt$. A simplified version of RESCAL is HolE (Nickel et al., 2016) that defines a vector for r and performs circular correlation of h and t has been found equivalent (Hayashi & Shimbo, 2017) to ComplEx.

Another tensor factorization model is Canonical Polyadic (CP) (Hitchcock, 1927). In CP decomposition, each entity e is represented by two vectors h_e, $t_e \in \mathbb{R}^d$, and each relation r has a single embedding vector $v_r \in \mathbb{R}^d$. MDE is similarly based on the idea of independent vector embeddings. A study (Trouillon et al., 2017) suggests that in CP, the independence of vectors causes the poor performance of CP in KG completion, however, we show that the independent vectors can strengthen a model if they are combined complementarily.

SimplE (Kazemi & Poole, 2018) analogous to CP, trains on two sets of subject and object entity vectors. SimplE's score function, $\frac{1}{2}\langle h_{e_i}, r, t_{e_j}\rangle + \frac{1}{2}\langle h_{e_j}, r^{-1}, t_{e_j}\rangle$, is the average of two terms. The first term is similar to DistMult. However, its combination with the second term and using a second set of entity vectors allows SimplE to avoid the symmetric issue of DistMult. SimplE allows learning of symmetry, anti-symmetry and inversion patterns. However, it is unable to efficiently encode composition rules, since it does not model a bijection mapping from h to t through relation r.

In **Latent Distance Approaches** the score function is the distance between embedding vectors of entities and relations. In the view of social network analysis, (Hoff et al., 2002) originally proposed distance of entities $-d(h,t)$ as the score function for modeling uni-relational graphs where $d(.,.)$ means any arbitrary distance, such as Euclidean distance. SE (Bordes et al., 2011) generalizes the distance for multi-relational data by incorporating a pair of relation matrices into it. TransE (Bordes et al., 2013) represents relation and entities of a triple by a vector that has this relation

$$S_1 = \| \ h + r - t \ \|_p \tag{1}$$

where $\| \ . \ \|_p$ is the p-norm. To better distinguish entities with complex relations, TransH (Wang et al., 2014a) projects the vector of head and tail to a relation-specific hyperplane. Similarly, TransR follows the idea with relation-specific spaces and extends the distance function to $\| \ M_r h + r - M_r t \ \|_p$. RotatE (Sun et al., 2019) combines translation and rotation and defines the distance of a t from tail h which is rotated the amount r as the score function of a triple $-d(h \circ r, t)$ where $\circ$ is Hadamard product.

Neural Network Methods train a neural network to learn the interaction of the **h**, **r** and **t**. ER-MLP (Dong et al., 2014) is a two layer feedforward neural network considering h, r and t vectors in the input. NTN (Socher et al., 2013) is neural tensor network that concatenates head h and tail t vectors and feeds them to the first layer that has r as weight. In another layer, it combines h and t with a tensor R that represents **r** and finally, for each relation, it defines an output layer r to represent relation embeddings. In SME (Bordes et al., 2014) relation r is once combined with the head h to get $g_u(h,r)$, and similarly it is combined with the tail t to get $g_v(t,r)$. SME defines a score function by the dot product of this two functions in the hidden layer. In the linear SME, $g(e,r)$ is equal to $M_u^1 e + M_u^2 r + b_u$, and in the bilinear version, it is $M_u^1 e \circ M_u^2 r + b_u$. Here, M refers to weight matrix and b is a bias vector.

4 MDE: Multiple Distance Embeddings

The score function of MDE involves multiple terms. We first explain the intuition behind each term and then explicate a framework that we suggest to efficiently utilize them such that we benefit from their strengths and avoid their weaknesses.

Inverse Relation Learning: Inverse relations can be a strong indicator in knowledge graphs. For example, if $IsParentOf(m,c)$ represents that a person m is a parent of another person c, then this could imply $IsChildOf(c,m)$ assuming that this represents the person c being the child of m. This indication is also valid in cases when this only holds in one direction, e.g. for the relations $IsMotherOf$ and $IsChildOf$. In such a case, even though the actual inverse $IsParentOf$ may not even exist in the KG, we can still benefit from inverse relation learning. To learn the inverse of the relations, we define a score function S_2 :

$$S_2 = \| \ t + r - h \ \|_p \tag{2}$$

Symmetric Relations Learning: It is possible to easily check that the formulation $\parallel h + r - t \parallel$ allows[1] learning of anti-symmetric pattern but when learning symmetric relations, $\parallel r \parallel$ tends toward zero which limits the ability of the model in separating entities specially if symmetric relations are frequent in the KG. For learning symmetric relations, we suggest the term S_3 as a score function. It learns such relations more efficiently despite it is limited in the learning of antisymmetric relations.

$$S_3 = \parallel h + t - r \parallel_p \tag{3}$$

Lemma 1. S_1 allows modeling antisymmetry, inversion and composition patterns and S_2 allows modeling symmetry patterns. *(See proof in Appendix A)*

Relieving Limitations on Learning of Reflexive Relations: A previous study (Kazemi & Poole, 2018) highlighted the common limitations of TransE, FTransE, STransE, TransH and TransR for learning reflexive relations where these translation-based models force the reflexive relations to become symmetric and transitive. To relieve these limitations, we define S_4 as a score function which is similar to the score of RotatE i.e., $\parallel h \circ r - t \parallel_p$ but with the Hadamard operation on the tail. In contrast to RotatE which represents entities as complex vectors, S_4 only holds in the real space:

$$S_4 = \parallel h - r \circ t \parallel_p \tag{4}$$

Lemma 2. The following restrictions of translation based embeddings approaches do not apply to the S_4 score function. R1: if a relation r is reflexive, on $\Delta \in \mathcal{E}$, r it will be also symmetric on Δ. R2: if r is reflexive on $\Delta \in \mathcal{E}$, r it will be also be transitive on Δ. *(See proof in Appendix B)*

Model Definition: To incorporate different views to the relations between entities, we define these settings for the model:

1. Using limit-based loss instead of margin ranking loss.
2. Each aggregated term in the score represents a different view of entities and relations with an independent set of embedding vectors.
3. In contrast to ensemble approaches that incorporate models by training independently and testing them together, MDE is based on multi-objective optimization (Marler & Arora, 2004) that jointly minimizes the objective functions.

However, when aggregating different terms in the score function, the summation of opposite vectors can cause the norm of these vectors to diminish during the optimization. For example if S_1 and S_3 are added together, the minimization would lead to relation(r) vectors with zero norm value. To address this issue, we represent the same entities with independent variables in different distance functions.

Based on CP, MDE considers four vectors $e_i, e_j, e_k, e_l, \in \mathbb{R}^d$ as the embedding vector of each entity **e** , and four vectors $r_i, r_j, r_k, r_l \in \mathbb{R}^d$ for each relation **r**.

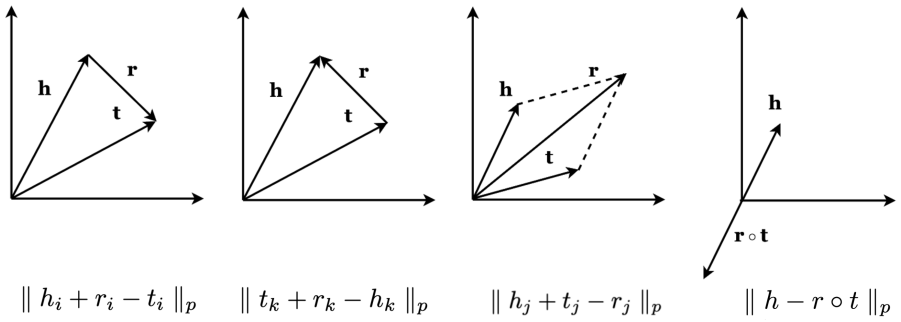
The score function of MDE for a triple (**h**, **r**, **t**) is defined as weighted sum of listed score functions:

$$f_{MDE} = w_1 S_1^i \ + \ w_2 S_2^j \ + \ w_3 S_3^k \ + \ w_4 S_4^l \ - \psi \tag{5}$$

where $\psi, w_1, w_2, w_3, w_4 \in \mathbb{R}$ are constant values. In the following, we show using ψ and limit-based loss, the combination of the terms in equation 5 is efficient, such that if one of the terms recognises if a sample is true F_{MDE} would also recognize it.

Limit-based Loss: Because margin ranking loss minimizes the sum of error from directly comparing the score of negative to positive samples, when applying it to translation embeddings, it is possible that the score of a correct triplet is not small enough to hold the relation of the score function (Zhou et al., 2017). To enforce the scores of positive triples become lower than those of negative ones, (Zhou et al., 2017) defines limited-based loss which minimizes the objective function such that

[1]We used the term "it allows" to imply that the encoding of such patterns do not inhibit the learning of relations having a particular pattern. Meanwhile in the literature SimplE uses "it can encode" and RotatE uses "the model infers".



$\| h_i + r_i - t_i \|_p$ $\| t_k + r_k - h_k \|_p$ $\| h_j + t_j - r_j \|_p$ $\| h - r \circ t \|_p$

Figure 1: Geometric illustration of the translation terms considered in MDE

the score for all the positive samples become less than a fixed limit. Sun et al. (2018) extends the limit-based loss so that the score of the negative samples become greater than a fixed limit. We train our model with the same loss function which is:

$$loss = \beta_1 \sum_{\tau \in \mathbb{T}^+} [f(\tau) - \gamma_1]_+ + \beta_2 \sum_{\tau' \in \mathbb{T}^-} [\gamma_2 - f(\tau')]_+ \tag{6}$$

where $[.]_+ = \max(., 0)$, $\gamma_1, \gamma_2 \in \mathbb{R}^+$. $\mathbb{T}^+, \mathbb{T}^-$ are the set of positive and negative samples and $\beta_1, \beta_2 > 0$ are constants denoting the importance of the positive and negative samples. This version of limit-based loss minimizes the aggregated error such that the score for the positive samples become less than γ_1 and the score for negative samples become greater than γ_2. To find the optimal limits for the limit-based loss, we suggest updating the limits during the training. *(See the explanation in Appendix D).*

Lemma 3. There exist ψ and $\gamma_1, \gamma_2 \geq 0$ ($\gamma_1 \geq \gamma_2$), such that only if one of the terms in f_{MDE} estimates a fact as true, f_{MDE} also predicts it as a true fact. Consequently, the same also holds for the capability of MDE to allow learning of different relation patterns. *(See proof in Appendix C)*

It is notable that without the introduction of ψ and the limits γ_1, γ_2 from the limit-based loss, Lemma 3 does not hold and framing the model with this settings makes the efficient combination of the terms in f_{MDE} possible.

In contrast to SimplE that ties the relation vectors of two terms in the score together, MDE does not directly relate them to take advantage of the independent relation and entity vectors in combining opposite terms.

The learning of the symmetric relations is previously studied (e.g. in (Yang et al., 2014; Sun et al., 2019)) and (Lin et al., 2015a) studied the training over the inverse of relations, however providing a way to gather all these benefits in one model is a novelty of MDE. Besides, complementary modeling of different vector-based views of a knowledge graph is a novel contribution.

4.1 MDE$_{NN}$: MDE AS A NEURAL NETWORK

The score of MDE is already aggregating a multiplication of vectors to weights. We take advantage of this setting to model MDE as a layer of a neural network that allows learning the embedding vectors and multiplied weights jointly during the optimization. To create such a neural network we multiply ψ by a weight w_5 and we feed the MDE score to an activation function. We call this extension of MDE as MDE$_{NN}$:

$$f_{MDE_{NN}} = \sigma(w_1 S_1^i + w_2 S_2^j + w_3 S_3^k + w_4 S_4^l + w_5 \psi) \tag{7}$$

where σ is logistic sigmoid function and $w_1, w_2, \ldots, w_5$ are elements of the latent vector w that are estimated during the training of the model. This framing of MDE reduces the number of hyperparameters. The major advantage of MDE$_{NN}$ in comparison to the current distance-based

models is that the logistic sigmoid activation function allows the non-linear mappings between the embedding vectors and the expected output for positive and the negative samples.

4.2 Time Complexity and Parameter Growth

Considering the ever growth of KGs and the expansion of the web, it is crucial that the time and memory complexity of a relational mode be minimal. Despite the limitations in expressivity, TransE is one of the popular models on large datasets due to its scalability. With $O(d)$ time complexity (of one mini-batch), where d is the size of embedding vectors, it is more efficient than RESCAL, NTN, and the neural network models. Similar to TransE, the time complexity of MDE is $O(d)$. Due to the additive construction of MDE, the inclusion of more distance terms keeps the time complexity linear in the size of vector embeddings.

5 Experiments

Datasets: We experimented on four standard datasets: WN18 and FB15k are extracted by (Bordes et al., 2013) from Wordnet (Miller, 1995) Freebase (Bollacker et al., 2008). We used the same train/valid/test sets as in (Bordes et al., 2013). WN18 contains 40,943 entities, 18 relations and 141,442 train triples. FB15k contains 14,951 entities, 1,345 relations and 483,142 train triples. In order to test the expressiveness ability rather than relational pattern learning power of models, FB15k-237 (Toutanova & Chen, 2015) and WN18RR (Dettmers et al., 2018) exclude the triples with inverse relations from FB15k and WN18 which reduced the size of their training data to 56% and 61% respectively.

Baselines: We compare MDE with several state-of-the-art relational learning approaches. Our baselines include TransE, RESCAL, DistMult, NTN, ER-MLP, ComplEx and SimplE. We report the results of TransE, DistMult, and ComplEx from (Trouillon et al., 2016) and the results of TransR and NTN from (Nguyen, 2017), and ER-MLP from (Nickel et al., 2016). The results on the inverse relation excluded datasets are from (Sun et al., 2019), Table 13 for TransE and RotatE and the rest are from (Dettmers et al., 2018)[2].

Evaluation Settings: We evaluate the link prediction performance by ranking the score of each test triple against its versions with replaced head, and once for tail. Then we compute the hit at N (Hit@N), mean rank (MR) and mean reciprocal rank (MRR) of these rankings. We report the evaluations in the filtered setting.

Implementation: We implemented MDE in PyTorch[3]. Following (Bordes et al., 2011), we generated one negative example per positive example for all the datasets. We used Adadelta (Zeiler, 2012) as the optimizer and fine-tuned the hyperparameters on the validation dataset. The ranges of the hyperparameters are set as follows: embedding dimension 25, 50, 100, 200, batch size 100, 150, and iterations 50, 100, 1000, 1500, 2500, 3600. We set the initial learning rate on all datasets to 10. For MDE, the best embedding size and γ_1 and γ_2 and β_1 and β_2 values on WN18 were 50 and 1.9, 1.9, 2 and 1 respectively and for FB15k were 200, 10, 13, 1, 1. The best found embedding size and γ_1 and γ_2 and β_1 and β_2 values on FB15k-237 were 100, 9, 9, 1 and 1 respectively and for WN18RR were 50, 2, 2, 5 and 1.

We selected the coefficient of terms in equation 5, by grid search in the range 0.1 to 1.0 and testing those combinations of the coefficients where they create a convex combination. Found values are w_1 = 0.16, w_2 = 0.33, w_3 = 0.16, w_4=0.33. We also tested for the best value for ψ between $\{0.1, 0.2, \ldots, 1.5\}$. We use ψ = 1.2 for all the experiments.

For MDE$_{NN}$, we use the same γ_1, γ_2, β_1 and β_2 values except for WN18 that the γ_1 and γ_2 are 4. We use the embedding size 50 for WN18RR, 200 for WN18, 200 for FB15k-237 and 200 for FB15k. We use ψ = 2 for all the MDE$_{NN}$ experiments. To regulate the loss function and to avoid over-fitting, we estimate the score function for two sets of independent vectors and we take their average in the prediction. Another advantage of this operation is the reduction of required training iterations. As a result, MDE reaches to the 99 percent of its ranking performance in 100 iterations, and MDE$_{NN}$ reaches its best performance in the benchmarks in just 50 iterations.

[2]Scores of ConvE on FB15k is from `https://github.com/TimDettmers/ConvE/issues/26`

[3]`https://pytorch.org`

Model	WN18 MR	WN18 MRR	WN18 Hit@10	FB15k MR	FB15k MRR	FB15k Hit@10
TransE	–	0.454	0.934	–	0.380	0.641
RESCAL	–	0.890	0.928	–	0.354	0.587
DistMult	–	0.822	0.936	–	0.654	0.824
SimplE	–	0.942	0.947	–	0.727	0.838
NTN	–	0.53	0.661	–	0.25	0.414
ER-MLP	–	0.712	0.863	–	0.288	0.501
ConvE	504	0.942	0.955	51	0.657	0.831
ComplEx	–	0.941	0.947	–	0.692	0.84
RotatE	309	**0.949**	0.959	40	**0.797**	0.884
MDE	118	0.871	0.956	49	0.652	0.857
MDE$_{NN}$	**3**	0.916	**0.980**	**2**	0.56	**0.976**

Table 1: Results on WN18 and FB15k. Best results are in bold.

Model	WN18RR MR	WN18RR MRR	WN18RR Hit@10	FB15k-237 MR	FB15k-237 MRR	FB15k-237 Hit@10
DistMult	5110	0.43	0.49	254	0.241	0.419
ComplEx	5261	0.44	0.51	339	0.247	0.428
ConvE	5277	0.46	0.48	246	0.316	0.491
RotatE	3340	0.476	0.571	177	0.338	0.533
MDE	3121	0.455	0.536	189	0.288	0.484
MDE$_{NN}$	**5**	**0.662**	**0.962**	**2**	**0.500**	**0.999**

Table 2: Results on WN18RR and FB15k-237. Best results are in bold.

5.1 Entity Prediction Results

Table 1 summarizes our results on FB15k and WN18 showing MDE$_{NN}$ outperforms all the state-of-the-art models in MR and Hit@10 tests and Table 2 shows the result of our experiment on FB15k-237 and WN18RR, where the improvement is much more significant.

Due to the existence of hard limits in the limit-based loss, the mean rank in both MDE and MDE$_{NN}$ is much lower than other methods.

The comparison of MDE to other state-of-the-art models, regardless of the MDE$_{NN}$, shows the competitive performance of MDE. It is observable that while MDE generates only one negative sample per positive sample and is using vector sizes between 50 to 200, it challenges RotatE which employs relatively large embedding dimensions (from 125 up to 1000) and high number of negative samples (up to 1024).

We observe that the application of sigmoid in MDE$_{NN}$ improves it significantly in all the benchmarks. Particularly, in the more challenging tests over WN18RR and FB15k-237, the improvement is more significant. For example, we can see that the construction of the neural network from the model increased its Hit@10 result on FB15k-237 from 0.484 to 0.999.

From analyzing the MRR scores, we can see that RotatE must be totally off in few cases whereas the MDE$_{NN}$ model almost never seems to be far off, but frequently fails to put the correct entity on top.

To our knowledge, MDE$_{NN}$ outperforms all the current embedding models in the MR and Hit@10 measures and specially performs better than all the existing models in all the measures on WN18RR and FB15k-237 benchmarks.

5.2 Ablation Study

To better understand the role of each term in the score function of MDE, we embark two ablation experiments. First, we train MDE using one of the terms alone, and observe the link prediction performance of each term in the filtered setting. In the second experiment, we remove one of the terms at a time and test the effect of the removal of that term on the model after 100 iterations.

Table 3 summarizes the results of the first experiment on WN18RR and FB15k-237. We can see that S_4 outperforms the other terms while S_1 and S_3 performs very similar on these two datasets. Between the four terms, S_2 performs the worst since most of the relations in the test datasets follow an antisymmetric pattern and S_2 is not efficient in modeling them.

Table 4 shows the results of the second experiment. The evaluations on WN18RR and WN18 show that removal of S_4 has the most negative effect on the performance of MDE. The removal of S_1 that was one of the good performing terms in the last experiment has the least effect. Nevertheless, S_1 improves the MRR in the MDE. Also, when we remove S_2, the MRR and Hit@10 are negatively influenced, indicating that there exist cases that S_2 performs better than the other terms, although, in the individual tests, it performed the worst between all the terms.

	WN18RR			**FB15k-237**		
Individual Term	MR	MRR	Hit@10	MR	MRR	Hit@10
S_1	3137	0.184	0.447	187	0.260	0.454
S_2	8063	0.283	0.376	439	0.204	0.342
S_3	3153	0.183	0.449	**186**	0.258	0.455
S_4	**2245**	**0.323**	**0.467**	220	**0.273**	**0.462**

Table 3: Results of each individual term in MDE on WN18RR and FB15k-237. Best results are in bold.

	WN18RR			**WIN18**		
Removed Term	MR	MRR	Hit@10	MR	MRR	Hit@10
S_1	3983	0.417	**0.501**	**113**	0.838	**0.946**
S_2	**3727**	0.358	0.490	131	0.823	0.943
S_3	3960	0.427	0.499	161	0.850	0.943
S_4	3921	0.366	0.478	163	0.705	0.929
$None$	3985	**0.428**	**0.501**	151	**0.844**	**0.946**

Table 4: Results of MDE after 100 iterations when removing one of the terms. Best results are in bold.

6 CONCLUSION

In this study, we showed how MDE relieves the expressiveness restrictions of the distance-based embedding models and proposed a general method to override these limitations for the older models. Beside MDE and RotatE, most of the existing KG embedding approaches are unable to allow modeling of all the three relation patterns. We framed MDE into a Neural Network structure and validated our contributions via both theoretical proofs and empirical results.

We demonstrated that with multiple views to translation embeddings and using independent vectors (that previously were suggested to cause poor performance (Trouillon et al., 2017; Kazemi & Poole, 2018)) a model can outperform the existing state-of-the-art models for link prediction. Our experimental results confirm the competitive performance of MDE and particularly MDE$_{NN}$ that achieves state-of-the-art MR and Hit@10 performance on all the benchmark datasets.

REFERENCES

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

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

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

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

Quoc Nguyen Dat, Sirts Kairit, Qu Lizhen, and Johnson Mark. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 460–466, 2016.

Luc De Raedt. *Logical and relational learning*. Springer Science & Business Media, 2008.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 601–610. ACM, 2014.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 192–202, 2016.

Katsuhiko Hayashi and Masashi Shimbo. On the equivalence of holographic and complex embeddings for link prediction. *arXiv preprint arXiv:1702.05563*, 2017.

Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

Peter D Hoff, Adrian E Raftery, and Mark S Handcock. Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098, 2002.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pp. 687–696, 2015.

Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. Encoding temporal information for time-aware link prediction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2350–2354, 2016.

Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*, pp. 4284–4295, 2018.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379*, 2015a.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015b.

R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

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

Chen Muhao, Tian Yingtao, Yang Mohan, and Zaniolo Carlo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *In Proceedings of IJCAI*, pp. 1511–1517, 2017.

Dat Quoc Nguyen. An overview of embedding models of entities and relationships for knowledge base completion. *arXiv preprint arXiv:1703.08098*, 2017.

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

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *Thirtieth Aaai conference on artificial intelligence*, 2016.

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

Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*, pp. 4396–4402, 2018.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HkgEQnRqYQ.

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

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pp. 2071–2080, 2016.

Théo Trouillon, Christopher R Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. Knowledge graph completion via complex tensor factorization. *The Journal of Machine Learning Research*, 18(1):4735–4772, 2017.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*, 2014a.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*, 2014b.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pp. 515–526. ACM, 2014.

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

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Xiaofei Zhou, Qiannan Zhu, Ping Liu, and Li Guo. Learning knowledge embeddings by combining limit-based scoring loss. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 1009–1018. ACM, 2017.

APPENDIX

A PROOF OF LEMMA 1.

Let r_1, r_2, r_3 be relation vector representations and e_i, e_j, e_k are entity representations. A relation r_1 between (e_i, e_k) exists when a triple (e_i, r_1, e_k) exists and we show it by $r_1(e_i, e_k)$. Formally, we have the following results:

Antisymmetric Pattern. If $r_1(e_i, e_j)$ and $r_1(e_j, e_i)$ hold, in equation 1 for S_1, then:

$$e_i + r_1 = e_j \quad \wedge \quad e_j + r_1 \neq e_i \quad \Rightarrow \quad e_i + 2r_1 \neq e_i \qquad \square$$

Therefore S_1 allows encoding of relations with antisymmetric patterns.

Symmetric Pattern. If $r_1(e_i, e_j)$ and $r_1(e_j, e_i)$ hold, for S_2 we have:

$$e_i + e_j - r_1 = 0 \quad \wedge \quad e_j + e_i - r_1 = 0 \quad \Rightarrow \quad e_j + e_i = r_1$$

$\square$

Therefore S_2 allows encoding relations with symmetric patterns. For S_1 we have:

Inversion Pattern. If $r_1(e_i, e_j)$ and $r_2(e_j, e_i)$ hold, from Equation 1 we have:

$$e_i + r_1 = e_j \quad \wedge \quad e_j + r_2 = e_i \quad \Rightarrow \quad r_1 = -r_2 \qquad \square$$

Therefore S_1 allows encoding relations with inversion patterns.

Composition Pattern. If $r_1(e_i, e_k)$, $r_2(e_i, e_j)$ and, $r_3(e_j, e_k)$ hold, from equation 1 we have:

$$e_i + r_1 = e_k \quad \wedge \quad e_i + r_2 = e_j \quad \wedge \quad e_j + r_3 = e_k \quad \Rightarrow \quad r_2 + r_3 = r_1 \qquad \square$$

Therefore S_1 allows encoding relations with composition patterns.

B PROOF OF LEMMA 2.

Proof. R1: For such reflexive r_1, if $r_1(e_i, e_i)$ then $r_l(e_j, e_j)$. In this equation we have:

$$e_i = r_1 e_i \wedge e_j = r_1 e_j \Rightarrow r_1 = U \not\Rightarrow e_i = r_1 e_j$$

where U is unit tensor.

R2: For such reflexive r_1, if $r_1(e_i, e_j)$ and $r_l(e_j, e_k)$ then $r_1(e_j, e_i)$ and $r_l(e_k, e_j)$. In the above equation we have:

$$e_i = r_1 e_j \wedge e_j = r_1 e_k \Rightarrow e_i = r_1 r_1 e_j e_k \wedge r_i = U \Rightarrow e_i = e_j e_k \not\Rightarrow e_i + e_k = r_l \qquad \square$$

C PROOF OF LEMMA 3.

We show there is boundries for $\gamma_1, \gamma_2, w_1, w_2, w_3, w_4$, such that learning a fact by one of the terms in f_{MDE} is enough to classify a fact correctly.

Proof. We show the boundaries for three aggregated terms in the the distance function, it is easily possible to extend it to four and more terms. It is enough to show that there is at least one set of boundaries for the positive and negative samples that follows the constraints. The case to prove is when three of the distance functions classify a fact negative N and the one distance function e.g. s_2 classify it as positive P, and the case that s_1 and s_3 classify a fact as positive and s_2 classify it as negative. We set $w_1 = w_3 = 1/4$ and $w_2 = 1/2$ and assume that Sum is the value estimated by the score function of MDE, we have:

$$a > \frac{N}{2} \geq \frac{\gamma_2}{2} \wedge \frac{\gamma_1}{2} > \frac{P}{2} \geq 0 \Rightarrow a + \frac{\gamma_1}{2} > Sum + \psi \geq \frac{\gamma_2}{2} \tag{8}$$

There exist $a = 2$ and $\gamma_1 = \gamma_2 = 2$ and $\psi = 1$ that satisfy $\gamma_1 > Sum \geq 0$ and the inequality 8. $\square$

Algorithm 1 Guided Limit Loss

Initilize: $\delta, \gamma_1 = \gamma_2 \in \mathbb{R}^+$, $\psi \in \mathbb{R}$, $i = 0$, $\xi \in \mathbb{R}^+$, $threshold \in \mathbb{R}^+$.
Inside training iterations...
if Using $loss_{guided}$ instead of $loss_{limit-based}$ **then**
    $loss^+ = \beta_1 \sum_{\tau \in \mathbb{T}^+} [f(\tau) - (\gamma_1 - \delta)]_+$
    $loss^- = \beta_2 \sum_{\tau' \in \mathbb{T}^-} [(\gamma_2 - \delta') - f(\tau')]_+$
    $loss = loss^+ + loss^-$
    if $loss^+ = 0$ & $\gamma_1 \geq \xi$ **then**
        $\delta = \delta + \xi$
        if $loss^- > threshold$ & $\gamma_2 \geq \xi$ **then** $\delta' = \delta' + \xi$

if Using $loss_{limit-based}$ **then**
    $loss$ = the result from equation 6

It can be easily checked that without introduction of ψ, there is no value of Sum that can satisfy both $\gamma_1 > Sum \geq 0$ and the inequality 8 and we calculated the value of ψ based on the values of γ_1, γ_2 and a. In case that future studies discover new interesting distances, this Lemma shows how to basically integrate them into MDE.

D SEARCHING FOR THE LIMITS IN THE LIMIT-BASED LOSS

While the limit-based loss resolves the issue of margin ranking loss with distance based embeddings, it does not provide a way to find the optimal limits. Therefore the mechanism to find limits for each dataset and hyper-parameter is the try and error. To address this issue, we suggest updating the limits in the limit-based loss function during the training iterations. We denote the moving-limit loss by $loss_{guide}$.

$$loss_{guide} = \lim_{\delta, \delta' \to \gamma_1} \beta_1 \sum_{\tau \in \mathbb{T}^+} [f(\tau) - (\gamma_1 - \delta)]_+ + \beta_2 \sum_{\tau' \in \mathbb{T}^-} [(\gamma_2 - \delta') - f(\tau')]_+ \tag{9}$$

where the initial value of δ_0, δ'_0 is 0. In this formulation, we increase the δ_0, δ'_0 toward γ_1 and γ_2 during the training iterations such that the error for positive samples minimizes as much as possible. We test on the validation set after each 50 epoch and take those limits that give the best value during the tests. The details of the search for limits is explained in Algorithm 1. After observing the most promising values for limits in the preset number of iterations, we stop the search and perform the training while having the δ values fixed(fixed limit-base loss) to allow the adaptive learning to reach loss values smaller than the $threshold$.

We based this approach on the idea of adaptive learning rate (Zeiler, 2012), where the Adadelta optimizer adapts the learning rate after each iteration, therefore in the $loss_{guided}$ we can update the limits without stopping the training iterations. In our experiments, the variables in the Algorithm 1, are as follows. $threshold = 0.05$, $\xi = 0.1$.