Hello my name is Jihwan Bae, and in today’s session I will be presenting you on the topic of Translating Embeddings for Modeling Multi-relational Data. (Next) Here is the table of contents. Firstly, I will go over some of the necessary preliminaries and background knowledge for understanding TransE. Then, I will explain the model with accordance to the paper. Consequently, I will present my own implementation. And finally end with conclusion.

(Next) To start off, let’s review the some of the essential concepts by dissecting the title. (Next) First keyword is translating. Translating is a Type of transformation that takes each point in a figure and slides it the same distance in the same direction. (Next) So, in this case the initial point shifted by 3 and 2 in x and y direction respectively. (Next) Nextly, we will examine the embedding. Suppose that we have these 4 words. We could represent individual word as an one hot vector. But, the computational cost increases exponentially with increasing word. (Next) Instead, we could we embed each word onto 2 dimensional space composed of Royalty and Femineity axis. This way, we could gain representation without increasing the number of dimension. So, for instance, King plus Woman will give Queen. This is quite intuitive as king with feminine feature is queen. (Next) Finally, let’s examine the multi-relational data. (Next) On the left is the classical graph structure. Graphs are composed of nodes and edges, typically with only one type of node and edge. In contrast, Multi-relational data refers to directed graphs whose nodes correspond to entities and edges of the form head, label, and tail each of which indicates that there exists a relationship of label between the entities head and tail. Here the graph head is Tom Holland and the tail is English the relation between head and tail is nationality as tom Holland is an English actor. Likewise, Willem Dafoe is American.

(Next)Having these concepts in mind, let’s move on to the explanation the actual model. TransE focuses on modeling multi-relational data from Knowledge bases with the goal of providing an efficient tool to complete them by automatically adding new facts, without requiring extra knowledge. A knowledge graph is a collection of nodes and edges where each node represents a real-world entity, and each edge represents a relation between entities.

(Next) This is the whole algorithm, but for now let’s focus on the loss function. I will go over the algorithm in detail when I present my own implementation of the paper. (Next) Given a training set S of triplets (h, l, t) composed of two entities h, t ∈ E (the set of entities) and a relationship l ∈ L (the set of relationships), our model learns vector embeddings (value in Rk) of the entities and the relationships. We want that h + l ≈ t when (h, l, t) holds. In other words, t should be a nearest neighbor of h + l), while h + l should be far away from t for the set of corrupted triplets. The corrupted triplets are composed of training triplets with either the head or tail replaced by a random entity but not both at the same time). Let’s derive the loss function step by step. Firstly, we want to minimize the dissimilarity measure for the correct triplet relationship. Next, we also want to maximize the dissimilarity measures for the corrupted triplet, which is equivalent to minimizing the negative distance of the corrupted triplet. Finally, we add the margin hyperparameter in order to prevent the situation of the distance between the corrupted set and the correct set being exactly equal to each other. Let me explain in detail what this means in the next slide.

(Next) This is the situation that may occur when we do not introduce the margin term. The distance between the correct head and tail is exactly equal to the distance between corrupted head and the tail. But this is not what we want. We want the distance between the correct ones to be closer than the corrupted pairs. So we implement the gamma term to prevent such case. The optimization is carried out by stochastic gradient descent (in minibatch mode), over the possible h, l and t.

(Next) We will now move on to examine some of the related works. Structured embedding is a very closely related work. The basic idea is the same, but the difference is that instead of adding the relation vector l, SE embeds the head and tail entities in a way that their embeddings are close to each other in some subspace that depends on the relationship for the correct set. Two different projection matrices for the head and for the tail is used in order to account for the possible asymmetry of relationship.

(Next) SE is actually more expressive than the transE as linear operators in dimension k + 1 reproduce affine transformations in a subspace of dimension k. In other words, while transE works on k dimension, structured embedding works one dimension higher in k+1 dimension.

(Next) This can be explained by affine transformation. Let’s assume that k is equal to 2. And we want to perform translation of x y pair to x+1, y.

(Next) However this is impossible. If x y is (0,0), regardless of what the transformation matrix is, we cannot translate the x, y as the result will always be equal to zero, zero as shown on the right.

(Next) Instead, what we could do is increase one dimension. We increased one dimension by adding z-axis and achieved 2-d translation from (x,y) to (x+1, y). The 3x3 matrix actually performs shear in the 3d, but if we fix the z dimension to constant, in this case Z equal to 1, that ‘slice’ of the layer in performing translation in the 2d. To recap, SE, with L2 as the identity matrix and L1 taken so as to reproduce a translation is then equivalent to TransE.

(Next) Another related approach is the Neural Tensor Model. A special case of this model corresponds to learning scores s(h, l, t) as shown on the screen. Score is lower for the corrupted triplets. Considering our norm constraints (∥ h ∥2 =∥ t ∥2 = 1) because the embedding vectors are normalized each iteration, and the ranking criterion that ∥ l ∥2 does not play any role in comparing corrupted triplets, transE corresponds to the score formula of the neural tensor model. when L is the identity matrix, and l = l1 = −l2. As in the case of SE, we can see that TransE is a simpler model with much fewer parameters.

(Next) Despite the lower expressiveness of our model, we still reach better performance that other models including SE. It is likely due to the fact that transE model is a more direct way to represent the true properties of the relationship, and optimization is difficult in embedding models. For SE, greater expressiveness seems to be more synonymous to underfitting than to better performance.

Simpler model simplify the training and prevent underfitting, which seems to compensate for a lower expressiveness in this case.

Let’s examine the experiment more closely. For the data set, wordnet and freebase were used. Wordnet knowledge base is designed to produce an intuitively usable dictionary and thesaurus, and support automatic text analysis. Freebase Freebase is a huge and growing KB of general facts; For the testing, redundant data due to reversing the relationship were. We wanted to test for both small and large data set, the smaller one is named as FB15k on the section and the larger one with 25k relationships and more than 17 millions training triplets is referred as FB1M.

For the evaluation, the ranking procedure was used. For each test triplet, the head is removed and replaced by each of the entities of the dictionary in turn. Dissimi- larities (or energies) of those corrupted triplets are first computed by the models and then sorted by ascending order; the rank of the correct entity is finally stored. This whole procedure is repeated while removing the tail instead of the head. We report the mean of those predicted ranks and the hits@10, i.e. the proportion of correct entities ranked in the top 10.

These metrics are indicative but can be flawed when some corrupted triplets end up being valid ones, from the training set for instance. In this case, those may be ranked above the test triplet, but this should not be counted as an error because both triplets are true. To avoid such a misleading behavior, we propose to remove from the list of corrupted triplets all the triplets that appear either in the training, validation or test set (except the test triplet of interest). This ensures that all corrupted triplets do not belong to the data set. In the following, we report mean ranks and hits@10 according to both settings: the original (possibly flawed) one is termed raw, while we refer to the newer as filtered (or filt.). We only provide raw results for experiments on FB1M.

The first method is Unstructured, a version of TransE which considers the data as mono-relational and sets all translations to 0 (it was already used as baseline in [2]). We also compare with RESCAL, the collective matrix factorization model presented in [11, 12], and the energy-based models SE [3], SME(linear)/SME(bilinear) [2] and LFM [6]. RESCAL is trained via an alternating least-square method, whereas the others are trained by stochastic gradient descent, like TransE. Table 1 compares the theoretical number of parameters of the baselines to our model, and gives the order of magnitude on FB15k. While SME(linear), SME(bilinear), LFM and TransE have about the same number of parameters as Unstructured for low dimensional embeddings, the other algorithms SE and RESCAL, which learn at least one k × k matrix for each relationship rapidly need to learn many parameters. RESCAL needs about 87 times more parameters on FB15k because it requires a much larger embedding space than other models to achieve good performance. We did not experiment on FB1M with RESCAL, SME(bilinear) and LFM for scalability reasons in terms of numbers of parameters or training duration.

ables 3 displays the results on all data sets for all compared methods. As ex- pected, the filtered setting provides lower mean ranks and higher hits@10, which we believe are a clearer evaluation of the performance of the methods in link prediction. However, generally the trends between raw and filtered are the same.

Our method, TransE, outperforms all counterparts on all metrics, usually with a wide margin, and reaches some promising absolute performance scores such as 89% of hits@10 on WN (over more than 40k entities) and 34% on FB1M (over 1M entities). All differences between TransE and the best runner-up methods are important.

We believe that the good performance of TransE is due to an appropriate design of the model according to the data, but also to its relative simplicity. This means that it can be optimized efficiently with stochastic gradient. We showed in Section 3 that SE is more expressive than our proposal. However, its complexity may make it quite hard to learn, resulting in worse performance. On FB15k, SE achieves a mean rank of 165 and hits@10 of 35.5% on a subset of 50k triplets of the training set, whereas TransE reaches 127 and 42.7%, indicating that TransE is indeed less subject to underfitting and that this could explain its better performances. SME(bilinear) and LFM suffer from the same training issue: we never managed to train them well enough so that they could exploit their full capabilities. The poor results of LFM might also be explained by our evaluation setting, based on ranking entities, whereas LFM was originally proposed to predict relationships. RESCAL can achieve quite good hits@10 on FB15k but yields poor mean ranks, especially on WN, even when we used large latent dimensions (2, 000 on Wordnet).

The impact of the translation term is huge. When one compares performance of TransE and Un- structured (i.e. TransE without translation), mean ranks of Unstructured appear to be rather good (best runner-up on WN), but hits@10 are very poor. Unstructured simply clusters all entities co- occurring together, independent of the relationships involved, and hence can only make guesses of which entities are related. On FB1M, the mean ranks of TransE and Unstructured are almost similar, but TransE places 10 times more predictions in the top 10.

Detailed results Table 4 classifies the results (in hits@10) on FB15k depending on several cate- gories of the relationships and on the argument to predict for several of the methods. We categorized the relationships according to the cardinalities of their head and tail arguments into four classes: 1-TO-1, 1-TO-MANY, MANY-TO-1, MANY-TO-MANY. A given relationship is 1-TO-1 if a head can appear with at most one tail, 1-TO-MANY if a head can appear with many tails, MANY-TO-1 if many heads can appear with the same tail, or MANY-TO-MANY if multiple heads can appear with multiple tails. We classified the relationships into these four classes by computing, for each relation- ship l, the averaged number of heads h (respect. tails t) appearing in the FB15k data set, given a pair (l, t) (respect. a pair (h, l)). If this average number was below 1.5 then the argument was labeled as 1 and MANY otherwise. For example, a relationship having an average of 1.2 head per tail and of 3.2 tails per head was classified as 1-to-Many. We obtained that FB15k has 26.2% of 1-TO-1 relationships, 22.7% of 1-TO-MANY, 28.3% of MANY-TO-1, and 22.8% of MANY-TO-MANY.

These detailed results in Table 4 allow for a precise evaluation and understanding of the behavior of the methods. First, it appears that, as one would expect, it is easier to predict entities on the “side 1” of triplets (i.e., predicting head in 1-TO-MANY and tail in MANY-TO-1), that is when multiple entities point to it. These are the well-posed cases. SME(bilinear) proves to be very accurate in such cases because they are those with the most training examples. Unstructured performs well on 1-TO-1 relationships: this shows that arguments of such relationships must share common hidden types that Unstructured is able to somewhat uncover by clustering entities linked together in the embedding space. But this strategy fails for any other category of relationship. Adding the translation term (i.e. upgrading Unstructured into TransE) brings the ability to move in the embeddings space, from one entity cluster to another by following relationships. This is particularly spectacular for the well-posed cases.

Illustration Table 5 gives examples of link prediction results of TransE on the FB15k test set (predicting *tail*). This illustrates the capabilities of our model. Given a head and a label, the top predicted tails (and the true one) are depicted. The examples come from the FB15k test set. Even if the good answer is not always top-ranked, the predictions reflect common-sense.

Using FB15k, we wanted to test how well methods could generalize to new facts by checking how fast they were learning new relationships. To that end, we randomly selected 40 relationships and split the data into two sets: a set (named *FB15k-40rel*) containing all triplets with these 40 rela- tionships and another set (*FB15k-rest*) containing the rest. We made sure that both sets contained all entities. *FB15k-rest* has then been split into a training set of 353,788 triplets and a validation set of 53,266, and *FB15k-40rel* into a training set of 40,000 triplets (1,000 for each relationship) and a test set of 45,159. Using these data sets, we conducted the following experiment: (1) models were trained and selected using *FB15k-rest* training and validation sets, (2) they were subsequently trained on the training set *FB15k-40rel* but only to learn the parameters related to the fresh 40 rela- tionships, (3) they were evaluated in link prediction on the test set of *FB15k-40rel* (containing only relationships unseen during phase (1)). We repeated this procedure while using 0, 10, 100 and 1000 examples of each relationship in phase (2).

Results for Unstructured, SE, SME(linear), SME(bilinear) and TransE are presented in Figure 1. The performance of Unstructured is the best when no example of the unknown relationship is provided, because it does not use this information to predict. But, of course, this performance does not improve while providing labeled examples. TransE is the fastest method to learn: with only 10 examples of a new relationship, the hits@10 is already 18% and it improves monotonically with the number of provided samples. We believe the simplicity of the TransE model makes it able to generalize well, without having to modify any of the already trained embeddings.

We proposed a new approach to learn embeddings of KBs, focusing on the minimal parametrization of the model to primarily represent hierarchical relationships. We showed that it works very well compared to competing methods on two different knowledge bases, and is also a highly scalable model, whereby we applied it to a very large-scale chunk of Freebase data. Although it remains unclear to us if all relationship types can be modeled adequately by our approach, by breaking down the evaluation into categories (*1-to-1*, *1-to-Many*, . . . ) it appears to be performing well compared to other approaches across all settings.

Future work could analyze this model further, and also concentrates on exploiting it in more tasks, in particular, applications such as learning word representations inspired by [8]. Combining KBs with text as in [2] is another important direction where our approach could prove useful. Hence, we recently fruitfully inserted TransE into a framework for relation extraction from text [16].