Evaluation of Characteristics of Knowledge Base Completion Models

Kavita Chopra

Abstract

As the backbone of the semantic web, knowledge graphs are emerging as important sources for machine readable applications. As a consequence, the utility of those applications will depend on the quality and completeness of knowledge bases. Large and especially non-curated knowledge bases are prone to containing false facts as well as missing facts which is why the completion and refinement of knowledge bases is increasingly getting in the focus of industry and research. Given an existing knowledge base the task of predicting new facts (Link Prediction) as well as detecting false facts is known as Knowledge Base Completion (KBC). KBC and related inference and reasoning tasks on knowledge graphs are successfully addressed by feature models for relational data which embed entities and relations in vector spaces of low latent dimensions. With the advantage of a common framework we implemented and evaluated low-complex state of the art models and based on our extensive empirical results on two benchmark datasets drew conclusions for best practices and configurations. Apart from Link Prediction we employed a classification task to verify the soundness and usefulness of the learned embeddings.

1 Introduction

1.1 Motivation

As the main pillars of the semantic web, knowledge bases are emerging as important structured data sources for web applications that seek to utilize the potential of the semantic web which is best expressed by the machine readability of semantically structured and interlinked knowledge artifacts across the entire web. Applications relying on knowledge graphs include search engines, such as Google, Yahoo!, and Microsoft Bing, as well as AI-based Question-Answering systems, for instance IBM Watson [1] or Wolfram Alpha[1]. Since the quality and usefulness of web applications will depend on the quality and size of the knowledge graphs they rest on, it should be no surprise that not just the initial construction of knowledge graphs but also their continuous maintenance and refinement [14] are increasingly getting in the focus of both industry and research. Many works have shown that incompleteness and inconsistency of facts in knowledge bases are wide spread which limits the usefulness of applications that rely on them [9], [17]. Regardless of whether knowledge bases are constructed automatically from structured or unstructured web content or through manual collaborative efforts, they are found to be prone to having a considerable extent of missing facts which could be mandatory for applications based on them. For example, in Freebase [2] for more than 70% of all the people covered the *place of birth* is missing [5], [17]. Furthermore, the quality of knowledge bases can suffer from inconsistencies through false facts. In the Linked Data Cloud (LOD), knowledge graphs or categories thereof are interlinked, forming very large knowledge graphs, of which the Google Knowledge Graph is a prominent example. This is how shortcomings of one knowledge graph are transferred to other knowledge graphs, possibly leading to contradictions and further inconsistencies. Interestingly, it has been successfully proven that missing facts can often be inferred from existing facts in the knowledge base by leveraging automated methods.

[1] https://www.wolframalpha.com/

In research, automated approaches to repair inconsistencies and adding missing facts based on existing facts in the knowledge graph is termed as Knowledge Base Completion (KBC) [10], [7], [9], [15], [17]. This task is also known as Link Prediction [10], that is to answer, whether a given fact can be inferred from the graph or in the case of a false fact, whether it belongs to the graph.

1.2 KBC through Relational Latent Feature Learning

Reasoning and inference tasks on knowledge graphs were formerly approached through reasoners based on Horn rules [9] but recent approaches rely on Representation Learning methods to embed entities and relations in vector spaces of low latent dimensions. Representation Learning, also known as feature learning, aims to learn the structure of the instance space through vector embeddings thereby allowing mathematical calculations on the instances whose results can be interpreted in a semantic way. The instances in a knowledge base that we aim to embed are the entities and relations that are used as the vocabulary to express facts as triples. For KBC tasks, the learned embeddings of triples are transformed to scores using scoring functions which are formulated such that they lead to higher scores for true triples, and lower scores for false triples. Literature has proposed a number of different score-based models to embed knowledge graphs in vector spaces of low latent dimensions (e.g. [3], [11], [12], [18], [8]). We developed a general framework under which we subsumed a number of score-based KBC models and based on extensive empirical experiments, we proposed new training configurations outperforming results from literature as well as tested new variants of the existing models to improve the state of the art. The advantage of a common framework is clearly the fact that results will be consistent across all evaluated models and implementational differences will not influence the results. To increase the validity and reliability of our results - to the extent it was possible with regard to the compatibility of our framework and the implementations by various other authors active in this field - we replicated their results based on their optimal configurations before we presented our improvements.

Outline In section 2, the framework for score-based KBC-models will be formalized to effectively convey the general idea behind many of the KBC models in literature and to facilitate the subsequent introduction of state of the art models. Section 3 will present a thorough discussion of the evaluation performance of the trained models based on Link Prediction and a classification task. Finally, section 4 will conclude highlighting the main insights derived from this evaluative work.

2 A Framework for Score-Based Relational Latent Feature Models

Knowledge graphs as the major input of the embedding models contain a collection of facts which are represented as triples where a single triple relates two entities through a specific relation. Therefore, knowledge graphs are known to contain relational data which is the target of relational latent feature models that learn embeddings of low dimensions for both entities and relations. The low latent embedding dimension should should still encode the most relevant features such that appropriate representations of the original space become feasible. The resulting learned embeddings can then be exploited for a number of reasoning and inference

tasks, such as Link Prediction, Entity Resolution and Link-Based Clustering (for a thorough review see [10]).
Before we present the models implemented and evaluated under the implemented framework, we introduce the notation required to formalize the framework followed by the framework's main components:

Notation In the semantic web, a triple (s, r, o) expresses a fact or a sentence from the knowledge graph by relating an entity, named subject, s $\in E$ (set of all entities) to an entity, named object, o $\in E$, via a relation r $\in R$ (set of all relations) as a labeled, directed edge between the two entity nodes. With $|E| = N_E$ and $|R| = N_R$, N_E and N_R denote the number of entities and relations, respectively. By N_T we refer to the number of triples existing in the knowledge base which is why they are also known as positive triples. As to the training set, by ${T_r}^+$ we denote the set of positive triples, whereas by ${T_r}^-$ we denote the negative set, which is based on the positive set ${T_r}^+$ with regard to a common relation r but with either subject or object corrupted through substitution by a random entity. We generally denote the corrupted triple by (s', r, o') because as long as the side to be corrupted is random, and subject and object are not replaced simultaneously, we do not care which side was corrupted. With $e_s, e_o \in Ent_{Emb}$ we express the embedding of the subject and object entity, respectively, while $r \in Rel_{Emb}$ reflects the relation embedding. The embedding dimension is denoted by d. The score- and margin-based learning of entities and relations rests on the idea that a fixed margin, $\gamma \in \mathbb{R}_{>0}$ will be enforced between the range of scores of false triples and the range of scores of true triples, with regard to the common r, respectively.

Input and Output of Framework The input of the KBC models is a knowledge base of triples in the form (s, r, o) and the hyperparameters, both general and model-specific, including the margin γ, and the learning rate LR. The output of the KBC models are the learned vector or matrix embeddings for all entities and relations.

Training Data In Machine Learning two types of learning problems are distinguished: supervised and unsupervised settings. In supervised learning, training data consists of pairs (x, y) where x is the representation of an object from the instance space, while y is the associated label. In unsupervised learning only x is given, and the task is to train the model such that each instance is classified based on its intrinsic features, e.g. distance to other instances. The pairwise learning procedure applied in score-based relational latent feature models can be understood as unsupervised since negative triples are not externally provided but are created during training. The training data consists of a set of triples present in the knowledge base which we call the positive set. In each training iteration, we create a corresponding negative set, where based on each positive triple denoted by (s, r, o), a corrupted triple (s', r, o') is created by exclusively replacing either subject or object with a random entity.

Scoring Function The scoring function is a function : $Ent_{Emb} \times Rel_{Emb} \times Ent_{Emb} \rightarrow \mathbb{R}$ which expresses the confidence in the truthness of a triple (s, r, o) and maps an embedding of a true triple (s, r, o) to a higher score than the embedding of a false triple (s', r, o').

Loss Function The loss function computes the sum of the actual margins between the scores of all positive triples represented by the learnable model parameters of (s, r, o) and their negative counterpart triples (s', r, o') where either s or o from the true triple were replaced by a random entity. In order to maximize the difference between the score of a positive and negative

triple up till a certain γ, a pairwise optimization procedure is conducted by minimizing the following loss function:

$$\sum_{(e_s,r,e_o)\in T_r^+} \sum_{(e_{s'},r,e_{o'})\in T_r^-} [\gamma - f(e_s,r,e_o) + f(e_{s'},r,e_{o'})]_+$$

$$= \sum_{(e_s,r,e_o)\in T_r^+} \sum_{(e_{s'},r,e_{o'})\in T_r^-} max\{\gamma - f(e_s,r,e_o) + f(e_{s'},r,e_{o'}), 0\}$$

Optimization Procedure The minimization is accomplished through an optimization procedure, such as the stochastic Gradient Descent which updates the model parameters not with respect to the entire training set at once, but based on a smaller batch, e.g. with a size of 100 triples.

Closed World vs. Open World Assumption The pairwise learning approach requires that we need both positive and negative triples. While positive triples correspond to the facts present in the knowledge base, a definition of negative triples used in the training process is not as straight-forward. According to the *Closed World Assumption*, any fact not found in the knowledge base is false, and therefore a valid negative fact. Under an *Open World Assumption*, non-encountered facts still have a probability to be true. On account of the corruption procedure many KBC models follow a *Local Closed World Assumption* (for more see [10]).

2.1 State of the Art Models for KBC

After having elaborated the general framework under which score-based models for KBC are trained, we can now introduce two state of the art models identified in literature, namely TransE and Bilinear, by focusing on the respective model-specific scoring functions as well as the underlying idea behind the models. Since many works revolve around improvements and extensions of these two models some notable variants of the two models will be presented and a new proposal for a variant of the Bilinear model, namely the Bilinear-Decomposed, will be made.

2.1.1 TransE Model

TransE can be regarded as a linear latent distance model [3] because it derives the score of a triple based on some distance measure applied on representations of entities and relations both of which are embedded as vectors in the same space of a low latent dimension d. Hence, we have $Ent_{Emb}, Rel_{Emb} \subseteq \mathbb{R}^d$ and the scoring function is given by $f(e_s,r,e_o) = -dist_{L^p}(e_s + r, e_o) = -||e_s + r - e_o||_{L^p}$. As distance measures, [3] considered the L^1 or L^2 norm. The model's name is motivated by the idea that relationships can be modeled as translations or simple vector additions in the embedding space, where for a valid triple (s, r, o) we enforce $e_s + r \approx e_o$.

2.1.2 Bilinear Model

In the Bilinear model, the idea is to model entities as vectors, that is $e_s, e_o \in \mathbb{R}^d$ and relations as relation-specific square matrices $M_r \in \mathbb{R}^{d\times d}$. A relationship between two entities can then be expressed as a vector-matrix multiplication $f(e_s,r,e_o) = e_s \times M_r \times e_o^T$ which was proposed in [13], one of the early works where this form of operation was applied on triples but restricted to binary relations. In [11], the bilinear approach for modeling relational data is

introduced as a tensor factorization problem. The advantage of the formulation of the vector-matrix-multiplication as a tensor factorization is the special optimization approach proposed in [10] which the authors have named as RESCAL. The bilinear RESCAL model relies on an Alternate Least-Square-method (ALS) where the parameters are learned in an alternating fashion. Although the bilinear approach solved by means of RESCAL does not fit in the framework of score- and margin-based models for KBC, we chose to refer to it since many works on KBC include the performance of RESCAL on Link Prediction as a benchmark to the performance of their own proposals.

The major difference between the score- and margin-based bilinear model and the approach used by RESCAL is related to the assumption about negative examples, that is, the triples not contained in the knowledge base but required for the pairwise training procedure [10]. The tensor-based RESCAL model attempts to consider every possible combination between two different entities s, o and a relation r and models none-existing facts to have a score of zero. The BILINEAR model subsumed under our framework does not assign any probabilities in advance but approximates these in a way, that the margin between the negative and positive scores reaches at most γ. Hence, RESCAL learns embeddings by assuming a Closed World Assumption while the BILINEAR model under our framework models embeddings based on the Local Closed World Assumption (see 2).

2.1.3 Variants of Bilinear

Bilinear-Diagonal The authors of [18] proposed to restrict the relation matrix M_r in the bilinear model to a diagonal matrix D_r imposing zeros in off-diagonal entries, such that the relation matrix has d number of parameters. This restriction causes BILINEAR-DIAGONAL to have the same parameter complexity as TRANSE. BILINEAR-DIAGONAL can also be seen as a variant of TRANSE with multiplicative interactions between entities and relations [18], [7].

Bilinear-Decomposed Interestingly, in the empirical results of [18] BILINEAR-DIAGONAL outperformed the classical BILINEAR model as well as the TRANSE model which encouraged us to further experiment with bilinear variants based on lower number of parameters. On this account, we proposed the BILINEAR-DECOMPOSED model in which the relation matrix M_r is decomposed into a product of two matrices A_r, $B_r \in \mathbb{R}^{d \times a}$ of two latent dimensions: d and a. For $a < \frac{d}{2}$ we would achieve a lower number of parameters than in classical BILINEAR. The decomposed formulation would provide us a way to control one hidden dimension of the relation embeddings instead of being bound to $d \times d$ parameters.

Model	**scoring function**	**#Parameters**	**#Operations**
TransE	$-\|\|e_s + r - e_o\|\|_{L^p}$	$O(N_E d + N_R d)$	$O(N_T)$
Bilinear	$s \times M_r \times o^T$	$O(N_E d + N_R d^2)$	$O((d^2 + d)N_T)$
Bilinear-Diag	$e_s \times diag(r) \times e_o^T$	$O(N_E d + N_R d)$	$O((2d)N_T)$
Bilinear-Decomp	$e_s \times A_r \times B_r \times e_o^T$	$O(N_E d + N_R da)$	$O((d^2 + d + da)N_T)$

Table 1: Space and runtime complexity of score-based-models for KBC

2.2 Complexity of Models

The complexity of the KBC models formulated under the score-based framework is determined by the parameters occurring in the scoring function. As to the runtime complexity, it is desirable

if the model scales at most linearly in the data size, that is in the number of triples N_T [10]. Table 1 shows the parameter and runtime complexities of the introduced models. TransE and Bilinear-Diagonal have the lowest number of parameters. Bilinear scales quadratic in the latent embedding dimension d and linear in N_T. The Bilinear-Decomposed model has a slightly worse runtime than the classical Bilinear model but depending on a has less parameters than the classical Bilinear. When evaluating KBC models with respect to their complexity, it is reasonable to take into consideration the magnitude of the most relevant parameters. Generally, knowledge graphs contain millions of entity nodes, while the number of relations is only in the thousands. On this account, the complexity will be dominated by $N_E \times d$.

2.3 Related Works

While we subsumed state of the art models under a simple score- and margin-based framework, in some works the same or related models are implemented as neural tensor networks (e.g. [18], [15], [5]), of which the Knowledge Vault approach [5], currently a research project at Google, is one prominent example for the largest application of score-based KBC models, where every fact contained in the Knowledge Vault is complemented with a score expressing the certainty at which the fact is true. Other approaches for KBC models are based on compositional methods, where the training data does not come in batches of triples but sequences of entities and relations resulting from random traversals along the knowledge graphs. A promising way of processing sequences as inputs in the context of KBC, is through Recurrent Neural Networks (RNNs), as shown by [4] and [9].

3 Discussion

The objective of this work is an extensive evaluation of state of the art models for Knowledge Base Completion (KBC) based on a specifically designed and implemented score- and margin-based framework for latent feature learning of relational data. We realized the optimization algorithms of the framework using the Machine Learning library TensorFlow and with Python on an Intel Core i3-4160 CPU with 3.60 Ghz. The code for the implemented framework including the scripts for evaluation tasks and visualizations is available under https://github.com/kavchop/KGC.

Datasets Knowledge bases have rapidly grown in size counting up to billions of facts involving millions of entities and thousands of relations. Training data of this calibre requires models that are easily scalable to large and growing knowledge bases through manageable complexities. While research on KBC models has attempted to consider these success criteria when proposing new models or improved variants, empirical experiments had been limited to smaller subsets of existing knowledge bases. [3] released subsets of Freebase (referred to as FBK15) and WordNet which included splits for training, validation and test data (see table 2). Those subsets have since become references for the evaluation results of many other works on the domain of KBC, e.g. [3], [18], [8]. While implementing the framework, we took into account considerations that would make training efficient, accurate and especially scalable to large datasets. However, to effectively measure the performance of our implementation as well as the contributions made by our proposed variants we decided to use the benchmark datasets provided by [3].

Dataset	Training triples	Validation triples	Test triples	Total triples	Entities	Relations
FB15k	483,142	50,000	59,071	592,213	14,951	1,345
WordNet	141,442	5,000	5,000	151,442	40,943	18

Table 2: Statistics on number of triples, entities and relations for datasets used

3.1 Evaluation Results on the Link Prediction Task

Nearly all of the works on KBC in literature evaluate the learned embeddings on the task of Link Prediction. The idea of this task is to compute the score of the test triple as well as the scores of every possible and valid corruption of that test triple in order to be able to report the rank of the score of the true triple against the ascendingly ordered scores of all corruptions. In detail, for every test triple (s, r, o) s is fixed while for all entities $\in E$ as possible objects the triple scores are computed (and analogously triple scores are computed for all entities as possible subjects while o is fixed). If the ranking is repeated over all test triples in the test set, the mean rank as the Mean Average Precision (MAP) can be reported. Ideally, MAP should be minimal, however the metric is not normalized and has to be assessed relative to the total number of corruptions for a single test triple which is equal to the number of entities. If we consider a filtered setting where we exclude corrupted triples turning out to be true by being present in the knowledge base, the number of corruptions for a single evaluated test triple will be lower affecting the maximum rank against which the filtered MAP will be reported. The Mean Reciprocal Rank (MRR) is another rank metric which has the advantage that it is normalized between 0 and 1. It is computed according to the formula $MRR = \sum_{i=1}^{|E|} \frac{1}{rank_i}$. What is reported in almost every Link Prediction evaluation across different works on KBC is the percentage metric hits@ten which corresponds to the proportion of test triples whose ranks were found to be in the top 10. Not all works on KBC report these three measures for Link Prediction, which is why it becomes difficult to compare the models, even though in many cases they were trained on the same datasets. With these considerations, in our work we made sure to report all metrics for every model trained. Due to the strong evaluation focus of this work we chose to report the results for every embedding dimension a model was trained with (which consistently were $d = 20$, $d = 50$, $d = 100$), as contrary to many other works which only published the optimal configurations. All results we report are filtered results.

Rank Measure	d=20	d=50	d=100	d=50 [3]*	replic. of [3]*
MRR	0.31	0.47	0.55	n/a	0.28
MAP	89	54	55	125	131
hits@10	52.40	77.05	78.93	47.10	48.01

Table 3: Evaluation results of TransE model on FB15K

3.1.1 TransE Results

The original TransE paper [3] trained the embeddings with Stochastic Gradient Descent using mini batches of 100 triples and a constant learning rate of $LR = 0.01$. Their optimal configuration was obtained for $d = 50$, L^1 as the distance measure in the scoring function, and $\gamma = 1$. Their results for a filtered MAP of 125, and hits@10 of 47.1% were confirmed by our close results

Rank Measure	d=20	d=50	d=100	d=20 [3]*	replic. of [3]*
MRR	0.37	0.40	0.38	n/a	0.37
MAP	389	476	537	251	394
hits@10	81.43	87.73	87.89	89.2	83.76

Table 4: Evaluation results of TransE model on WordNet

of MAP = 131 and hits@10 = 48.01% based on a replication with the same configurations using our framework, see table 3. Since [3] did not report the MRR, the respective row indicating the results from the original paper are marked as 'n/a'. We found, that TransE could substantially be improved, by optimizing the loss function based on the AdaGrad optimizer [6] which supports adaptive learning rates depending on the frequency of features encountered during training. The initial learning rate was set to 0.1. For all of the three embedding dimensions we obtained significantly better performances. Even for $d = 20$, our results based on AdaGrad are significantly better than the results for the higher dimension of $d = 50$ based on a constant learning rate. Furthermore, AdaGrad significantly accelerated convergence of the embeddings which is highly favourable with respect to large knowledge graphs. For WordNet, [3] found their optimal results for a configuration of $d = 20$, L^1 as the distance measure, and a margin $\gamma = 2$. Again the learning rate was held constant at 0.01 in their setting. Our results confirmed that for WordNet TransE's performance could indeed be improved if we trained with a margin of 2 (see table 4). However, we were not able to replicate the published results by [3] on the WordNet database for their optimal dimension $d = 20$. Instead of their MAP of 251, we obtained 394 as the best MAP despite several training repetitions, and our hits@10 of 83.76% also remained below their optimal value of 89.2%. However, our proposal to use adaptive learning rates through AdaGrad was again fruitful on experiments with the WordNet dataset, since for all dimension the hits@ten consistently improved when resorting to AdaGrad. What we noticed with our experiments with WordNet is that for all three dimensions and even across the different models we used (see tables 7, 9) the reported MAP was larger than the MAPs obtained on Freebase, independent of the model. As elaborated earlier, the MAP-measure is not normalized and has to be examined relative to the worst case rank. With 40,943 entities, the WordNet database has far more instances than FB15K counting up to 14,951 entities. Hence, for Freebase the maximum rank relative to which the MAP values are reported is lower than in WordNet which explains why the values for the MAPs on Freebase were consistently better (i.e. lower) than the values on WordNet. It is therefore recommended to consider normalized measures alongside the MAP.

Rank Measure	d=20	d=50	d=100	d=50 [18]*	Dropout on d=50	Dropout on d=100
MRR	0.32	0.36	0.21	0.31	0.34	0.28
MAP	101	95.5	169	n/a	96	164
hits@10	49.79	54.71	37.70	51.82	55.82	43.50

Table 5: Evaluation results of Bilinear model on FB15K & results based on regularization through dropout

Rank Measure	eval. on train. set d=50	eval. on train. set d=100
MRR	0.43	0.42
MAP	22	24
hits@10	65.09	63.70

Table 6: Evaluation results of BILINEAR model on training data in FB15K

Rank Measure	d=20	d=50	d=100	d=50 [18]*	eval. on train. set d=100	eval. on train. set d=100
MRR	0.34	0.40	0.66	0.31	0.90	0.95
MAP	550	629	649	n/a	2	3
hits@10	60.28	83.36	72.65	51.90	98.07	97.84

Table 7: Evaluation results of BILINEAR model on WordNet

3.1.2 Bilinear Results

Table 5 shows an overview of the Link Prediction results for the BILINEAR model with regard to the embedding dimensions. For none of the considered dimensions the BILINEAR model outperformed the TRANSE model. In contrast to the results of TRANSE, BILINEAR does not improve with increasing embedding dimensions. The BILINEAR model has far more parameters than TRANSE which raised the presumption that overfitting could have been the case in the BILINEAR model. Overfitting refers to the observation that a model performs well on training data, but worse on unseen test data. To prove our presumption, we evaluated the model on a large portion (60,000 triples) of randomly drawn training triples. The evaluation performance on the training triples was indeed significantly better than the results based on the test triples (table 5. Effects of overfitting were also reflected by the evaluation results on WordNet (table 7). The hits@ten dropped from 83.36% to 72.65% when increasing the d from 50 to 100.

Countermeasures for Overfitting in Bilinear To address the condition of overfitting observed in BILINEAR for large embedding dimensions, we need to examine the factors leading to overfitting. Firstly, overfitting occurs, when the expressivity of the model, reflected in the number of parameters is high compared to the provided training data. If this is the case, the model succeeds in memorizing the training data, but fails to generalize to unseen test data. However, it should be noted that we cannot simply solve the problem of overfitting by increasing the training data size. The aim of vector space models is to learn the structure of the graph by enforcing a latent dimension on the embeddings of entities and relations to learn the most relevant features of entities and relations. If the latent complexity of the training data is low making it feasible to describe the graph through low number of features, then a lower latent dimensionality could be sufficient to embed the graph. A higher dimension would learn less relevant differences amongst entities and relations, leading to poor generalizations when the model encounters unseen test triples. Furthermore, the training sample, and for measurement reasons also the validation and test sample have to be representative of the underlying population in order to ensure good generalizations of the model. Apart from the elaborated considerations to avoid overfitting, literature on Machine Learning generally proposes regularization which

adds a penalty term to the loss function where the former slows down convergence of the loss function in an effort to improve validation and evaluation performance on unseen triples. The penalty term is usually a norm of the model parameters. However, the dominating practice in training KBC models across the board has been to normalize the entity embeddings during training. Another approach is to employ a method called *dropout* on those model parameters which appear prone to overfitting. The idea behind this method is to randomly set a specified proportion (determined by the dropout-rate) of a model parameter to zero, e.g. in the case of a matrix parameter, randomly chosen matrix entries are set to zero before it is updated in the next training step. Due to the successful empirically backed experience with dropout, it has become a popular practice employed on the layers of deep neural networks [16] which are known for their high parameterization requiring large amounts of training data. We proposed to apply dropout on the relation matrix of BILINEAR when training with larger dimensions where overfitting was empirically demonstrated (see tables 6 and 7). The results we obtained significantly improved the model performance, where for $d = 100$ on Freebase, the previous hits@10 of 37.7% (MRR = 0.21) increased to 43.50% (MRR = 0.28) after applying dropout (see table 5). However, we would also like to stress that dropout remarkably slowed down the learning process. For the reported results, we applied it at every mini-batch iteration step, and noticed that applying it less frequently, e.g. after every epoch made the results even worse rather than improving them.

Rank Measure	**d=20**	**d=50**	**d=100**	**[18]*: d=100**
MRR	0.32	0.40	0.48	0.35
MAP	102	80	70	n/a
hits@ten	49.79	66.16	76.05	57.70

Table 8: Evaluation results of BILINEAR-DIAGONAL model on FB15K

Rank Measure	**d=20**	**d=50**	**d=100**	**[18]*: d=100**
MRR	0.37	0.66	0.72	0.36
hits@10	504	462	498	n/a
hits@ten	77.97	94.21	94.6	57.7

Table 9: Evaluation results of BILINEAR-DIAGONAL model on WordNet

3.1.3 Comparison between Bilinear-Diagonal and TransE

In literature it was proposed to restrict the relation matrix in BILINEAR to a diagonal relation which led to better results according to [18]. This BILINEAR-DIAGONAL variant was implemented and tested with our framework and the finding that BILINEAR-DIAGONAL proved to be better than the classical bilinear approach, was confirmed. The collated results in table 8 demonstrate that with BILINEAR-DIAGONAL on FB15k we achieved significantly better results with our framework (hits@10=76.05% vs. hits@10 = 57.70% [18]), even though both results were based on a margin of 1 as well as on adaptive learning rates through AdaGrad. There were, however other differences between the settings used in our work and [18], such as the mini-batches of 10 (instead of 100) which we found as remarkably slowing down the training process as well as the consideration of two negative triples for every positive triple processed during the pairwise and score-based training. Since unlike [18] we reported the results with regard to all the three embedding dimensions, we could obtain a good overview of how the model performs

when complexity increases. Interestingly, by restricting BILINEAR such that the relation matrix is only allowed to be a diagonal matrix, we arrive at the same number of parameters for both TRANSE and BILINEAR-DIAGONAL. It is therefore fairly reasonably to compare both models using the relevant tables 3 and 8, as well as 4 and 9 for Freebase and WordNet, respectively. In TRANSE, we observed that with increasing d, the results improved when we used AdaGrad [6]. In classical BILINEAR, we saw that an increased d failed to improve the results, an observation which we traced back to the empirically demonstrated condition of overfitting. However, for BILINEAR-DIAGONAL we see the same trend as in TRANSE, that is, the model is improving with increasing d (see tables 8 and 9), even though the improvements are degressive. While TRANSE outperforms BILINEAR-DIAGONAL on Freebase [3], BILINEAR-DIAGONAL succeeds in outperforming TRANSE on WordNet [9]. Both can therefore be regarded as equally powerful state of the art models for KBC with low parameter and runtime complexities (see 2.2).

3.1.4 Bilinear-Decomposed

In general, TRANSE clearly outperforms BILINEAR even though TRANSE has lower number of parameters than the classical BILINEAR. It was however interesting to find in literature as well as confirm with our own implementation that BILINEAR-DIAGONAL performed nearly as good as TRANSE and that both contain the same number of learnable parameters. Our presumption that BILINEAR was generally prone to overfitting was significantly backed by our empirical finding (table 5) where we evaluated the learned model on the training data and then employed dropout as a means to reduce the condition of overfitting on the model performance. These indications as well as the promising results of BILINEAR-DIAGONAL motivated us to hold on to the BILINEAR model as an expressive model that should have room for improvement if the parameterization could be controlled. For this reason, we were encouraged to implement and test our proposed BILINEAR-DECOMPOSED. Table 10 shows the results for $d = 20$ with regard to different values of a. The results reflected our presumption that until the point where overfitting would set in, we could obtain better model performance on the Link Prediction task when using more parameters. Therefore we see that the evaluation results within the same dimension improve with increasing a. However, for none of the dimensions, BILINEAR-DECOMPOSED was capable of outperforming the TRANSE, BILINEAR-DIAGONAL or even the classical BILINEAR, which is why in parts the table is incomplete for experiments on WordNet.

	a=1			a=5			a=10		
Dataset	**MAP**	**hits@10 (%)**	**MRR**	**MAP**	**hits@10 (%)**	**MRR**	**MAP**	**hits@10 (%)**	**MRR**
FB15k	336	33.43	0.22	128	44.96	0.29	116	46.37	0.30
WordNet	358	31.9	0.20						

Table 10: Evaluation Results on both Dataset with regard to different hidden dimensions for a and $d = 20$

3.2 Evaluation Results on the Classification Task

Apart from Link Prediction, which is the typical task in literature KBC models are evaluated on, we used the learned embeddings for a classification task that was inspired by the procedure presented in NTN [15]. Our approach differed in the way that we resorted to a ROC-analysis which dynamically finds the optimal threshold for separating negative scores from positive

Model	d=20	d=50	d=100
TransE Learned	0.985	0.987	0.983
TransE Initial	0.525	0.532	0.541
Diagonal Learned	0.998	0.997	0.995
Diagonal Initial	0.516	0.551	0.453
Bilinear Learned	0.948	0.992	0.985
Bilinear Initial	0.493	0.481	0.534

Table 11: Classification results (mean-AUC from relation-specific classification sets over all relations) based on Freebase embeddings

Model	d=20	d=50	d=100
TransE Learned	0.955	0.984	0.976
TransE Initial	0.516	0.486	0.520
Diagonal Learned	0.980	0.991	0.986
Diagonal Initial	0.455	0.537	0.501
Bilinear Learned	0.948	0.974	0.967
Bilinear Initial	0.480	0.531	0.496

Table 12: Classification results (mean-AUC from relation-specific classification sets over all relations) based on WordNet embeddings

scores with regard to triples sharing the same relation r. In our binary classification setting the task is to distinguish, by means of the resulting scores based on the learned embeddings, between a sample of positive triples and a sample of negative triples. Negative triples are again created through corruption based on positive triples (s, r, o), that is, the substitution of either s or o by a random entity, however, only if that random entity is ever seen in a combination with relation r. For instance, if the positive fact is (WashigtonDC, capital, US), a valid corruption is (WashigtonDC, capital, US), but not (ElvisPresley, capital, US), or even (Cologne, capital, U.S) because 'Cologne' has never been in a triple with a combination of 'capital'. To plot the ROC-curve, we compute the False-Positive-rate (FP-rate) and the True-Positive-rate (TP-rate) for a number of different thresholds on the scale of both negative and positive scores based on a classification set with common r. Each threshold δ_i, i.e. each point on the ROC-curve represents a linear classifier which tells negative and positive triples apart with respect to threshold δ_i. An optimal classifier will correspond to a point on the ROC-curve which has the largest TP-rate and the lowest FP-rate. What is typically reported to assess the quality of a classifier by means of a ROC-analysis, is the Area under the ROC-curve (AUC). In case of perfect classification with 100% TP-rate and 0% FP-rate, the AUC will result to 1. A random classification will result in a straight line through the origin, leading to an AUC of 0.5. We conducted a ROC-analysis on classification sets based on each and every relation encountered in the database. To express the quality of the overall model considering all of its learned entities and relations, we averaged the AUCs obtained for all relations and reported these. Tables 11 and 12 clearly show the almost ideal performance of all the models with regard to the test sets of both datasets. Furthermore, we see that the performance between the different dimensions does not significantly differ, which in case of model selection would induce us to consider the lowest dimensionality as well as the model with the lowest number of parameters. For further verification, we computed the mean-AUC over the classification sets with regard to r based on the initial embeddings as well, and as expected obtained values close to a mean-AUC of 0.5.

4 Conclusion

4.1 Summary and Future Work

The objective of this work was to extensively evaluate state of the art models on KBC. For this purpose we implemented a framework for score- and margin-based latent feature models for relational data under which we trained the TransE, Bilinear, Bilinear-Diagonal, and the Bilinear-Decomposed model, with the latter being the model that we proposed. The framework accepts any knowledge base as input where facts are represented in the triple format (s, r, o) corresponding to relational data and learns the embeddings based on a pairwise learning approach, which we refer to as the score- and margin-based approach. We trained our models on a subset of Freebase as well as on WordNet, both benchmark datasets [3] used in many different works on KBC, and evaluated the learned embeddings on two tasks, namely Link Prediction and a classification task. For the trained models, only Link Prediction was tested in literature, while classification tasks were not. With our framework we were able to significantly improve the results reported in original literature for the respective models. In a nutshell, we found that simple low-complex models, such as TransE or the Bilinear-Diagonal model lead to the highest performance within all configurations and dimensions. With the growing semantic web, the size of knowledge bases is also rising rapidly, making scalability to large and growing knowledge bases an important success criteria for the practicality of KBC models. Scalability is determined by the complexity of the models. Apart from scalability being negatively affected by high complexities, we empirically saw that high complexities, reflected by the number of parameters either through higher dimensions or more complex models have a negative impact on the learning capability of a model by increasing the chances of overfitting, where the model fails to generalize well on unseen triples but performs well on training data. Due to these considerations, not only runtime will benefit from lower dimensions and low-complex models but also the quality of learning. Besides, we observed that optimizers with adaptive learning rates such as AdaGrad [6] not only performed better but also accelerated the convergence process, which is again favourable with regard to large and growing knowledge bases. The quality of learning was furthermore convincingly verified with a classification task aimed at separating positive triples from negative triples sharing a common relation r.

As a limitation we would like to remark that that our framework does not recognize or treat literals in a different way, e.g. in (CityX, hasPopulation, '300,000') the numerical literal as the object of the triple is not differentiated as such. An appropriate processing of literals in knowledge graphs could therefore be one possible avenue for future research in KBC.

References

[1] Joanna Biega, Erdal Kuzey, and Fabian M. Suchanek. Inside yago2s: a transparent information extraction architecture. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume*, pages 325–328, 2013.

[2] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250, 2008.

[3] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013.*

Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States., pages 2787–2795, 2013.

[4] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. *CoRR*, abs/1607.01426, 2016.

[5] 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 *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 601–610, 2014.

[6] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[7] Kelvin Guu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. *CoRR*, abs/1506.01094, 2015.

[8] 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 of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 687–696, 2015.

[9] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. Compositional vector space models for knowledge base completion. *CoRR*, abs/1504.06662, 2015.

[10] 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, 2016.

[11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 809–816, 2011.

[12] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing YAGO: scalable machine learning for linked data. In *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, pages 271–280, 2012.

[13] Alberto Paccanaro and Geoffrey E. Hinton. Learning distributed representations of concepts using linear relational embedding. *IEEE Trans. Knowl. Data Eng.*, 13(2):232–244, 2001.

[14] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[15] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 926–934, 2013.

[16] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[17] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, pages 515–526, 2014.

[18] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575, 2014.