Revisiting Ensembles for Knowledge Graph Embeddings

Master Thesis

presented by
Kuan Min Chen
Matriculation Number 1583702

submitted to the
Data and Web Science Group
Prof. Dr. Rainer Gemulla
Ph.D. candidate Daniel Ruffinelli
University of Mannheim

September 2020

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A Knowledge Graphs (KGs)[7] represents knowledge as a graph, which is often expressed as multi-relational data. A fact in a KG is stored in the form of triple (*head*, *relation*, *tail*) (also (*h*, *r*, *t*)), where *head* and *tail* are entities and *relation* represents the relation between the two entities, e.g., (*Paris*, *capital of*, *France*). WordNet[14], Freebase[4], Yago[21], and DBpedia[12] are examples for large-scale KGs.

Despite of the vast volumn, most KGs suffer from incompleteness, which has inspired research in the knowledge base completion (KBC) task aiming at predicting unseen relations *r* between two existing entities:(*head*, ?, *tail*), predicting the tail entity given the head entity or predicting the head entity given the tail entity in the query relation: (*head*,*relation*, ?). With conceptual clearity and excessive scalablity, the embedding based approaches, also called *knowledge graph embeddings*(KGEs) [5, 25], have attained promising outcomes in such tasks. Models applied are referred as KGE models that produce scores for querying task for further evaluation.

In other area of our work, ensemble learning is to utilize a series of models to learn, and then aggregate the result with a particular approach. With the presumably better generalization ability, the performances are expected to be better. This has been the case in a number of machine learning competitions, where the winning solutions used ensemble methods[22]. However, despite the fact that ensemble learning for improving single model performance has been an important direction in the current research of machine learning, and have so far shown efficacy in many areas, there are not many trials in KGEs.

Independently, there has been a lot of progress made in training individual KGE models. Ruffinelli et al.[19] report on the results of an extensive experimental study with popular model architectures and training strategies across a wide range

of hyper-parameter settings, with the finding that when trained appropriately, the relative performance differences between various model architectures often shrinks and sometimes even reverses when compared to prior results, which raises our interests. We should revisit ensembles with KGEs to see if they are still better than single KGE models.

Research Question. Previous studies reported improvements with ensembles over single KGE models. Furthermore, a recent work[19] pointed out that single KGE models were generally not trained well, so the ensembles applied in precedent works presumably used weak baselines. As it turns out, we want to know whether those ensembles still outperform well trained baselines.

In the present work, not only we investigate ensembles with approaches applied in the precedent studies, where we utilize the best configurations for models from the discovery[19] to facilitate our study, but also go further exploring the possibility of having better performance by applying other types of ensembles. We use LibKGE[1], a PyTorch-based library, which is the framework developed for [19] to run our experiments. Furthermore, we extend the functionality of the library to support different training and jointly learning approaches in our study.

The rest of this article is organized as follows. Chapter 2 reviews related work involving ensembles in KGEs. Chapter 3 presents technical definitions about KGs and , KGEs, ensembles, stacking and joint learning. Chapter 4 explains the experiments on different types of Stacking and the trials on fine-tuning and joint learning.

Chapter 2

Related Work

The early work from Kadlec et al.[10] run experiments on ensemble aside from the main focus on how an appropriately tuned baseline can affect the model performance on data set Freebase(FB15K). The ensemble they tried is simply taking the average of the score matrices of a set of DistMult, a KGE model, without identifying how many models used. The results show slightly improvements comparing to that of single DistMult.

Schlichtkrull et al.[20] attempted to experiment on ensemble while introducing R-GCN, a new KGE model in their work, focusing on link prediction task. While comparing how R-GCN out perform DistMult, baed on the nature of model R-GCN and DistMult relating to the perfomance on factorizations, which are complementary, they try to combine the strengths of both into a single model $R{+}GCN^{+}$. They applied weighted sum approach on the ensemble, where the weights are manually assigned to be fixed without stating the reason. They applied the same ensemble to experiments on all data set. It turns out that $R{-}GCN^{+}$ shows similar performance to that of the second best on Freebase(FB15K-237), but better on Freebase(FB15K) and WordNet(WN18), where the second best is R-GCN in all data set.

Krompaß et al.[11] tested the potential of combining various KGE models to drive prediction quality, i.e., the potential of ensemble. They extended the experiments on different data set, i.e., DBpedia and YAGO, with different approaches from earlier works to aggregate the scores from KGE models. First, they trained a logistic regression with subsample of the training data for each models, aiming to map the scores into values with range [0,1]. Afterwards, they average the transformed scores from each models. Their ensembles consist of many KGE models with same weighted scores contributing to the ensembles. Also, they experimented on ensemble at relation-type level. Area under precision-recall curve (AUPRC) is applied to evaluate the performance. Besides Freebase, they also extended the ex-

periments on DBpedia and YAGO. The settings lead to large improvements of the best single predictor of up to 11%.

Juric et al.[9] also focus on testing the power of ensemble, which consist of 9 KGE models. Different from works mentioned earlier assembling the scores from models to train the aggregate function once, they split the data set into subset by relation, and build ensembles that are trained on each different relations. The sub ensembles all together form one ensemble, and the sub ensemble trained with the corresponding relation will be applied during evalutaion. With the goal of solving a commonly known issue in KGs, the incomplete coverage, they built a medical data set assembling from multiple sources for experimenting. Here only one logistic regression that is used to aggregate the scores from models. With the same evaluation as previous work on AUPRC, the ensemble at relation level performs 8.92% better comparing to the best single model.

Wang et al.[26] tried out the basic ensemble and the ensemble at relation level, the same approach from Juric et al.[9], with the main focus of their work on bilinear embedding model study. Furthermore, the KGE models that are puzzled into ensemble are pretrained along with hyperparameter search. The fine work results in a better performed ensemble than the best single model on Freebase(FB15K) and WordNet(WNRR).

Meilicke et al.[13] presented a fine-grained evaluation with insights into the strengths and shortcomings of KGE models. Aside from the focus of their work, they experiment ensembles at relation level. They result in better performance than the best single model on Freebase(FB15K) and Freebase(FB15K-237). Also, interestingly, RuleN slightly outperform the ensemble at relation level on WordNet(WN18).

Chapter 3

Preliminaries

In the following, we introduce related knowledge and define the notations for better understanding in the later experiments.

3.1 Knowledge Graphs

A Knowledge Graphs (KG) represents knowledge as a graph, which is often expressed as multi-relational data. A fact in a KG is stored in the form of triple (*head*, *relation*, *tail*) (also (h, r, t)), where *head* and *tail* are entities and *relation* represents the relation between the two entities, e.g., (*Tim Cook*, *CEO of*, *Apple*). To define, a Knowledge Graph $K \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a collection of triples $\{(h,r,t)\}$, where $\mathcal{E}$ and $\mathcal{R}$ are the entity set and relation set, and $h, r \in \mathcal{E}$ and $r \in \mathcal{R}$. Note that $\{(h,r,t)\}$ can also be presented in RDF-based notion as $(subject, predicate, object)((s, p, o)$ for short). We refer to triples with the 2 types of notations interchangeably in the following work.

Also, the recognized issue of incomplete coverage, where most KGs suffer from, has inspired research in the knowledge base completion task aiming at predicting unseen relations r between two existing entities:(h, ?, t) or predicting the tail entity given the head entity in the query relation: (h,r, ?). With conceptual clearity and excessive scalablity, the embedding based approaches, also called *knowledge graph embeddings*, have attained promising outcomes in such tasks.

3.2 Knowledge Graph Embeddings

Knowledge graph embedding (KGE) is a function transforming an entity or a relation to an embedding, a n-dimensional vector space representations, which can be a vector or a tensor. With the embeddings, the transformation is performed without

leveraging the rich information from the relation structure, and often presented as *score function*, $f_r(h, t)$ computing the plausibility of a triple (h, r, t) in the embedding space.

For example, RESCAL[16], a latent semantics-based KGE method, has its score function $\mathbf{h}^\intercal \mathbf{M}_r \mathbf{t}$, a transformation on any pair of entities $\mathbf{h}$,$\mathbf{t}$ $\in \mathbb{R}^d$ which indicates the relation $\mathbf{M_r}$ $\in \mathbb{R}^{d \times d}$. In recent years, KGE has allured much attention and many methods have been proposed, mostly varing in the score functions used. They can be classified into three types[5, 25], translation-based model, latent semantics-based, and neural network-based.

The basic idea of translation-based model is to represent relations as translations from subjects to objects. Starting from TransE, many models of this type are developed. Semantic matching models, also called tensor factorization-based[5], match latent semantics of entities and relations embedded in the vector space to calculate the plausibility of facts. They have their self-defined scoring functions respectively. Examples are RESCAL[16], DistMult[27], and ComplEx[24], which also take parts in our experiments for reproduction purposes.

In recent years, inspired by the nature of convolutional neural networks(CNNs), neural network-based models have also been studied. For example, ConvE[6] uses 2D convolution over embeddings and multiple layers of nonlinear features. The embeddings of subject and relation are reshaped and concatenated into an input matrix and fed to the convolution layer. Table 3.1 presents the entity and relation representations and scoring functions[8] of the model involved in our work.

Model	Ent. embed.	Rel. embed.	Scoring Function $f_r(h, t)$
RESCAL	$\mathbf{h}$,$\mathbf{t}$ $\in \mathbb{R}^d$	$\mathbf{M_r}$ $\in \mathbb{R}^{d \times d}$	$\mathbf{h}^\intercal \mathbf{M}_r \mathbf{t}$
DistMult	$\mathbf{h}$,$\mathbf{t}$ $\in \mathbb{R}^d$	$\mathbf{r}$ $\in \mathbb{R}^{d \times d}$	$\mathbf{h}^\intercal diag(\mathbf{M}_r)\mathbf{t}$
ComplEx	$\mathbf{h}$,$\mathbf{t}$ $\in \mathrm{C}^d$	$\mathbf{r}$ $\in \mathrm{C}^d$	Re($\mathbf{h}^\intercal$ diag($\mathbf{r}$)$\bar{\mathbf{t}}$)
ConvE	$\mathbf{M}_h$ $\in \mathbb{R}^{d_w \times d_h}$,$\mathbf{t}$ $\in \mathbb{R}^d$	$\mathbf{M}_r$ $\in \mathbb{R}^{d_w \times d_h}$	σ(vec(σ([$\mathbf{M}_h$,$\mathbf{M}_r$]*ω))$\mathbf{W}$)$\mathbf{t}$

Table 3.1: Embeddings and Scoring Functions

One approach to evaluate the performance of a KGE model is to measure how good the model is at entity classification, which refers to predicting the tail entity given the head entity in the query relation, *R*(*h*, ?), and predicting the head entity given the tail entity in the query relation, *R*(?, *t*), We apply a *score-based ranking model*[26] for evaluating such tasks. We quote a well defined definition on *score-based ranking model* below from [26] with partial adjustments in notations and descriptions to be consistent with the rest of the content.

A *score-based ranking model* associates to a score function ${f_r}^m(i, j)$

$\in\mathbb{R}^d$ with each subject-relation-object triple of a KGE model *m*. Denote by $S_k^m \in\mathbb{R}^{N\times N}$ the corresponding $scoreingmatrix$ for relation k, i.e., $[S_k^m]$ = ${f_r}^m(i,j)$, Denote by $S^m \in\mathbb{R}^{N\times N\times K}$ the *score tensor* of *m*, i.e., the tensor with frontal slices $S_{(k)}^m = S_k^m$.

Score-based models are used to rank (pairs of) entities by their predicted truthfulness, given a query of form *R*(?, *j*), *R*(*i*,?), or *R*(?,?). Generally, a result with a higher score is considered more likely to be correct, We say that an $N \times N$ matrix is a ranking matrix if all its entries are in $\{1,2,...,N^2\}$ and whenever there is any entry with value s-1. Denoted by $\Pi(S)$ the unique ranking matrix associated with score matrix S, where $\pi_{ij}(S) \stackrel{def}{=} [\pi(S)]_{ij}$ is the $denserank$ of s_{ij} in the multiset of the entries of S. For every pair of tuples $(i, j) \in\mathrm{N}\times N$ and $(i', j') \in\mathrm{N}\times N$, we have

$$s_{ij} \leq s_{i'j'} \Longleftrightarrow \pi_i j(S) \geq \pi_{i'j'}(S)$$

After applying *score-based ranking model*, we attain the rankings of all entities given a query, where the rankings also stand for the comparative plausibility of all entities comparing each individual to others. There are two common standards used to evaluate the performance of a KGE model, **HITS@K** and **Mean Reciprocal Rank**(MRR), where the former measures the percentage of cases in which the true triple appears in the top k ranked triples, and the later takes the average of the reciprocal rank assigned to the true triple.

To reach a better performance, KGE models can be trained in three methods. We first consider positive triples $T^+ \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ given a data set, where $E = |\mathcal{E}| \leq 2$ is the number of entities. One is to train the model with all positive triples T^+ and the negative triples built by the complete set of *subject* or *object*-corrupted triples, which are exactly all the other triples besides it self for a query task as *R*(*s*, ?) or *R*(?,*o*). As it turns out, there will be one positive triple and $2N$-2 negative triples used to train the embeddings of each T^+. This method is referred as *1vsAll*. For example, there are two triples, (s_1, p_1, o_1), $(s_1, p_1, o_2) \in\mathrm{T}^+$, Based on the definition of $1vsAll$, for a query task as $R_1(s_1, ?)$, the later triple is consider as a negative sample, although it is also a fact in T^+. In contrast, *KvsAll* is a training method considering a positive triple set with T^+ and triples that can possibly result in completion, and the negative triple set is a set excluding those triples. So, for the same query task, triple (s_1, p_1, o_2) is considered as positive triple by *KvsAll* method. Derived from *1vsAll*, *negative sampling* allows one to assign the number of negative samples and the sampling method applied to sample out the negative samples. With those settings, one sets up for training with positive triples T^+ and the self-defined negative samples applying *negative sampling* method.

3.3 Ensemble Learning with Knowledge Graph Embeddings

To relate the concept of ensemble learning[29] to KGEs, ensemble learning with Knowledge Graph Embeddings refers to have many models, commonly referred as base learners and trained by different score functions(see Table 3.1 for examples) on the same problem, combined as a ensemble model to get better results.

In the following, we first introduce what is ensemble learning, For reproduction purposes, we then focus on Stacking, a ensemble learning framework we applied with KGEs. Finally, we talk about joint learning.

3.3.1 Ensemble Learning

"Ensemble learning is a machine learning paradigm where multiple learners are trained to solve the same problem. In contrast to ordinary machine learning approaches which try to learn one hypothesis from training data, ensemble methods try to construct a set of hypotheses and combine them to use." by Zhi-Hua Zhou[22].

Those multiple learners are commonly referred as *base learners*, which are assembled by *meta learner* to form an ensemble model. There has been a number of machine learning competitions, where the winning solutions used ensemble methods[22]. Three frameworks commonly used are : *Boosting*, *Bagging*, and *Stacking*[29]. As our work focus on reproduction study, we then present Stacking in more details, For future work, one could applied other frameworks.

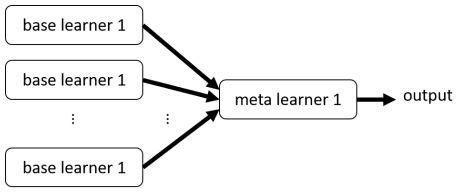

Figure 3.1: The mechanism of ensemble learning

Stacking, which is applied in the present work, is a technique to combine multiple base learners through a meta learner. At first, we train the base learners independently on the same task. Then we use the outputs from each base learner to train the meta learner. The pseudo-code of Stacking[29] is shown in Figure 3.2. The training process shown in Figure 3.2 for base learners is in sequence, which, in practical, can be trained in parallel so to be more efficient.

Input: Data Set $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_m, y_m)\}$;
First-level with base learners with score functions $L_1, ..., L_T$;
Second-level with meta learner L.

Process:

for $t = 1, ..., T$:
  $h_t = L_t(D)$ %Train a first-level individual learner h_t by applying the first-level
end; %learning algorithm $L_t$to the original data set D
$D' = \emptyset$; % Generate a new data set
for $i = 1, ..., m$:
  for $t = 1, ..., T$:
    $z_{it} = h_t(\mathbf{x}_i)$ % Use h_t to classify the training example $\mathbf{x}_i$
  end;
  $D' = D' \cup \{((z_{i1}, z_{i2}, ..., z_{iT}), y_i)\}$
end;
$h' = L(D')$ %Train the second-level learner h' by applying the second-level
%training meta learner L to the new data set D'

Output: $H(\mathbf{x}) = h'(h_1(\mathbf{x}), ..., h_T(\mathbf{x}))$

Figure 3.2: The Stacking algorithm

3.3.2 Stacking

In this section, we introduce the stacking we applied in the KBC task. To begin, let's assume we have a KG with k triples, q_e entities, and q_r relations. As introduced, a stacking involves base learners, which are N different KGE models with the scoring functions, respectively:

$$\theta^1_{s,p,o}, \theta^2_{s,p,o}, ..., \theta^N_{s,p,o}.$$

We apply logistic regression model as the meta learner, which aggregates the scores into one:

$$P(x_{s,p,o} = 1 | \theta^1_{s,p,o}, ..., \theta^N_{s,p,o}) = \frac{1}{1 + exp\{-(\Sigma\omega_i \theta^i_{s,p,o} + \omega_0)\}}$$

where ω_i and ω_0 stand for coefficients, bias, respectively, and $\theta^i_{s,p,o}$ represents the score functions of each base learner.

To train the stacking, it begin with training the base learners. The training for individual KGE model involves the setting in the scoring function $\theta^n_{s,p,o}$ of the model, regularization term, training methods, batch size, learning rate, optimizer,

and many others. In practice, we apply the settings of the best trained model from [19].

With the trained base learners, we now build the training data set for training the meta learner. With different mechanism take place while scoring, the scores from different KGE models vary greatly, where one model produce the score set with values ranging from -100 to 100, the other can generate the score set with values ranging from -50 to 50.

In order to fairly evaluate the contributions of each models in the later work, in each triple, given a *relation* and one of *subject* or *object*, we score on all entities opposed to the incomplete triple, and then perform Min-max normalization[3] to the set of scores from each query. We name the scores after Min-max normalization *transformed scores* in the following content.

The new formed training set then is applied to train the logistic regression by *1vsAll*.

3.3.3 Joint Learning

In contrast to stacking, joint learning refers to training all the models in the production pipeline as one task with the purpose to reduce error propagation. Some studies[18][15][17][28] have applied such approaches in the area related to extraction of entities and relations. In our setting, only difference comparing to sacking is that training the meta learner and training the base learners take place in one training pipeline, meaning ω_i and the embeddings of all base learners will be updated in each iterations. The backward propagation will also update the corresponding parameters of the loss function as a pipeline starting from meta learner down to all base learners.

Chapter 4

Experiments

4.1 Stacking at Model Level

Stacking can be built in different structure. We begin the experiments on a stacking at model level with meta learner as logistic regression, a comparatively simple structure. The goal is to test if ensembles with better trained base learners still outperform the best single model.

For building up trained base learners, we used the pretrained models from [19]. For building up the trained meta learner, we used validation set with batch size 512 to train `SGDClassifier(loss='log')` [2] with default settings from the **Scikit-Learn** library. For evaluating, we used test set.

For reproduction purposes, We used the FB15K-237 [23](extracted from Freebase, more challenging than Freebase) and WNRR[6](extracted from WordNet, more challenging than WNRR) in our study. Table 4.1 shows the statistics of the data sets.

data set	#Entity	#Relation	#training	#validation	#testing
FB15K-237	14,541	237	272,115	17,535	20,466
WNRR	40,943	11	86,835	3,034	3,134

Table 4.1: Statistics of the experimental data sets

Table 4.2 showing the best performance of KGE models on both FB15K-237 and WNRR[19] serves as a reference for comparison in the following content. In the following context, the model will be presented with single letter as the letter noted in the round bracket next to the name of KGE model in Table 4.2. This also applies to the subscript of parameters.

	FB15K-237		WNRR	
model	HITS@10	MRR	HITS@10	MRR
RESCAL (R)	**54.06**	**35.59**	51.72	46.66
DistMult (D)	53.13	34.28	53.05	45.18
ComplEx (C)	53.60	34.77	**54.74**	**47.49**
ConvE (V)	52.07	33.86	50.45	44.16

*By column, the largest values are in bold.

Table 4.2: Best performance from Ruffinelli et al.(2020)[19](in %)

Table 4.3 shows the performances on WNRR from averaging the transformed scores (from 2^{nd} to 3^{rd} column) and Stacking with meta learner (from 4^{th} to 5^{th} column), with the differences between them (from 6^{th} to 7^{th} column). The performance from averaging the scores is here served as an another baseline, where we take on the statistical concept that the **maen** represents how all things are in general or how all things should generally perform. So we believe it is a good reference to compare to while trying new approaches, as applying logistic regression to aggregate in our case.

Comparing to single KGE model's performance shown in Table 4.2, results from simply averaging the scores is already good, where the increase can be up to 2.57 for HITS@10 and 1.92 for MRR. This can also be observed in the work from Kadlec et al.[10](see Table 4.4, where the simple averaging approach advance the performance on HITS@1 by 5.6 percentage point(ppt) and 25.1 ppt on WN18 and FB15K, respectively. The performance doesn't improve much by applying meta learner. Also, for HITS@10, comparing to averaging scores, stacking with meta learner involving three base learners or more seems to have progressed much on average than that involving just two base learners. This trend can also be observed in the work from Wang et al.(2018)[26](see Table 4.5), which makes sense with the thought of being together is stronger. However, it is worse in general for MRR.

Stacking	(averaging)		(meta learner)		(c - a)	(d - b)
	HITS@10 (a)	MRR (b)	HITS@10 (c)	MRR (d)		
R + D	55.06	47.72	55.33	47.35	0.27	-0.37
R + C	56.27	49.04	56.16	48.97	-0.11	-0.07
R + V	54.21	48.07	54.08	48.06	-0.13	-0.01
D + C	56.33	48.61	56.32	48.45	-0.01	-0.16
D + V	54.48	46.40	54.40	46.28	-0.08	-0.12
C + V	56.22	48.57	56.13	48.56	-0.09	-0.01
R + D + C	57.07	**49.41**	57.29	49.35	0.22	-0.06
R + D + V	55.81	48.10	55.93	47.79	0.12	-0.31
R + C + V	57.02	49.37	57.04	**49.46**	0.02	0.09
D + C + V	56.60	48.70	56.78	48.65	0.18	-0.05
R + D + C + V	**57.31**	49.34	**57.32**	49.30	0.01	-0.04

*By column, the largest values are in bold.

Table 4.3: Stacking on WNRR(in %)

Model	HITS@1	
	WN18	FB15K
DistMult	72.8	54.6
DistMult ensemble	**78.4**	**79.7**

*By column, the largest values are in bold.

Table 4.4: Performance from Kadlec et al.(2018)[10](in %)

data set	WN18		FB15K	
Model	HITS@10	MRR	HITS@10	MRR
HolE	94.1	93.8	72.6	50.2
TransE	94.5	43.9	79.5	34.4
RESCAL	87.8	79.9	59.6	38.1
RESCAL+TransE	94.8	87.3	79.7	51.1
RESCAL+HolE	94.4	**94.0**	79.1	57.5
HolE+TransE	94.9	93.8	84.6	61.0
RESCAL+HolE+TransE	**95.0**	**94.0**	**85.1**	**62.8**

*By column, the largest values are in bold.

Table 4.5: Performance from Wang et al.(2018)[26](in %)

Stacking	ω_R	ω_D	ω_C	ω_V	ω_0
R + D	0.0520	0.0735			-10.568
R + C	0.0608		0.0767		-10.641
R + V	0.0518			0.0442	-10.568
D + C		0.0740	0.0698		-10.577
D + V		0.0729		0.0433	-10.560
C + V			0.0667	0.0410	-10.552
R + D + C	0.0514	0.0731	0.0687		-10.591
R + D + V	0.0527	0.0740		0.0453	-10.596
R + C + V	0.0466		0.0647	0.0383	-10.558
D + C + V		0.0701	0.0645	0.0381	-10.555
R + D + C + V	0.0449	0.0632	0.0632	0.0364	-10.563

Table 4.6: Parameters of meta learner for WNRR trained by validation set

Table 4.6 shows the trained coefficients and intercept of the logistic regression, the meta learner for each stacking on WNRR. In general, ω_0 increases as the number of base learner increase. The pattern of coefficients is not so clear while trained by Validation set. However, if we apply train set, where the size is much more than that of Validation set, we see in Table 4.7 that the coefficients do not vary much up to 2 decimal places for any particular KGE model, independent of the number and the type of base learners that are involved in the same stacking, where DistMult is the most emphasized and ConvE is the least emphasized among all. This can be that the larger number of samples in training set contributes to how coefficients better represent the emphasis of each base learner in overall performance. Com-

paring to using validation set for training, the coefficients are less centralized. We think transforming (min-maxing) the scores before aggregating makes the model emphasis transparent on coefficients. Furthermore, we can also observe from Table 4.7 that, for any particular base learner's coefficient, its value gradually decreases as more base learner joining the ensemble, which makes sense in terms of the distribution of the efforts. It is also interesting that the coefficient fluctuates more for ConvE comparing to other models, which may be caused by the comparatively less stable scores from ConvE.

Stacking	ω_R	ω_D	ω_C	ω_V	ω_0
R + D	0.146	0.184			-10.692
R + C	0.146		0.162		-10.691
R + V	0.144			0.131	-10.687
D + C		0.182	0.161		-10.670
D + V		0.182		0.130	-10.665
C + V			0.162	0.132	-10.693
R + D + C	0.145	0.183	0.162		-10.726
R + D + V	0.143	0.181		0.129	-10.716
R + C + V	0.143		0.160	0.130	-10.726
D + C + V		0.182	0.160	0.130	-10.715
R + D + C + V	0.141	0.180	0.160	0.128	-10.752

Table 4.7: Parameters of meta learner for WNRR trained by training set

For another data set, Table 4.8 shows the performances in the same structure as in Table 4.3 on FB15K-237 from averaging the scores and stacking with meta learner. Comparing to single KGE model's performance, simply by averaging the scores is already good also, where the increase can be up to 0.96 for HITS@10 and 0.48 for MRR. It advance on HITS@10 by 0.5 and MRR by 0.62 the most by applying meta learner as logistic regress. Also, comparing to averaging scores, there is no significant improvement on Stackings with meta learner neither as the number of base learners increases. In contrast to WNRR, the performance on FB15K-237 seems to be indifferent to the number of base learners in ensembles, whereas it even slightly gets better for ensembles with less base learners. The MRR increase and decrease as how HITS@10 perform in each ensembles in general.

Table 4.9 shows the trained coefficients and intercept of the logistic regression, the meta learner for each stacking on FB15K-237. In general, not like WNRR, ω_0 doesn't increases as the number of base learner increase. But the pattern of coefficients is not so clear while trained by Validation set, as the case in WNRR.

However, if we apply training set, we see in Table 4.10 that the coefficients also do not vary much up to 2 decimal places for any particular KGE model, independent of the number and the type of base learners that are involved in the same stacking, where again DistMult is the most emphasized and ConvE is the least emphasized among all. Also, the ensembles involving DistMult and not RESCAL, such as D+C, D+V, and D+C+V, have smaller ω_0, which is the same for Validation set. As in WNRR, in Table 4.10 we see that, for any particular base learner, the coefficient of a particular model gradually decreases as more base learner jointing that ensemble.

Stacking	(averaging)		(meta learner)		(c - a)	(d - b)
	HITS@10 (a)	MRR (b)	HITS@10 (c)	MRR (d)		
R + D	54.66	35.64	55.07	36.26	0.42	0.62
R + C	54.91	35.89	55.24	36.29	0.32	0.40
R + V	54.34	35.77	54.84	36.15	0.50	0.38
D + C	54.21	35.12	54.28	35.16	0.08	0.04
D + V	54.17	35.23	53.74	35.11	-0.43	-0.12
C + V	54.30	35.44	54.17	35.26	-0.13	-0.18
R + D + C	54.92	35.81	**55.36**	36.30	0.45	0.49
R + D + V	54.90	35.99	55.20	36.28	0.30	0.30
R + C + V	55.01	**36.07**	55.31	**36.33**	0.29	0.25
D + C + V	54.61	35.53	54.45	35.58	-0.16	0.04
R + D + C + V	**55.02**	36.01	55.34	36.32	0.32	0.32

*By column, the largest values are in bold.

Table 4.8: Stacking on FB15K-237(in %)

Stacking	ω_R	ω_D	ω_C	ω_V	ω_0
R + D	0.0403	0.0086			-9.6030
R + C	0.0399		0.0138		-9.6042
R + V	0.0394			0.0242	-9.6082
D + C		0.0101	0.0158		-9.5928
D + V		0.0107		0.0276	-9.5999
C + V			0.0176	0.0289	-9.6057
R + D + C	0.0399	0.0082	0.0137		-9.6066
R + D + V	0.0411	0.0091		0.0256	-9.6152
R + C + V	0.0397		0.0137	0.0245	-9.6137
D + C + V		0.0092	0.0148	0.0258	-9.6006
R + D + C + V	0.0143	0.0406	0.0406	0.0252	-9.6189

Table 4.9: Parameters of meta learner for FB15K-237 trained by validation set

Table 4.9 shows the trained coefficients and intercept of the logistic regression, the meta learner for each stacking on FB15K-237. In general, ω_0 also increases as the number of base learner increase. The pattern of coefficients is not so clear while trained by Validation set. However, if we apply training set, where the size is much more than that of Validation set, we see in Table 4.10 that the coefficients also do not vary much up to 2 decimal places for a particular KGE model, independent of the number and the type of base learners that are involved in the same stacking, where again DistMult is the most emphasized and ConvE is the least emphasized among all.

Stacking	ω_R	ω_D	ω_C	ω_V	ω_0
R + D	0.3346	0.4223			-9.5309
R + C	0.3354		0.3990		-9.5322
R + V	0.3340			0.3712	-9.5361
D + C		0.4208	0.3968		-9.5207
D + V		0.4224		0.3718	-9.5278
C + V			0.3981	0.3714	-9.5336
R + D + C	0.3309	0.4180	0.3940		-9.5345
R + D + V	0.3299	0.4185		0.3673	-9.5431
R + C + V	0.3304		0.395	0.3679	-9.5416
D + C + V		0.4175	0.3937	0.3677	-9.5286
R + D + C + V	0.3266	0.4143	0.3905	0.3642	-9.5468

Table 4.10: Parameters of meta learner for FB15K-237 trained by training set

We then run five times on the best Stackings on HITS@10 and MRR for WNRR and FB15K-237 respectively. Table 4.11 and Table 4.12 show the mean and standard deviation of the validation data performance.

data set	Stacking	mean & std. (5 times)
WNRR	R + D +C + V	HITS@10:57.32±0.01
		MRR: 49.33±0.01
FB15K-237	R + D + C	HITS@10:55.38±0.01
		MRR: 36.30±0.00

Table 4.11: Statistics of best Stacking on HITS@10

data set	Stacking	mean & std. (5 times)
WNRR	R + C + V	HITS@10:57.11±0.11
		MRR: 49.43±0.05
FB15K-237	R + C + V	HITS@10:55.32±0.01
		MRR: 36.33±0.02

Table 4.12: Statistics of best Stacking on MRR

4.2 Stacking at Relation Level

In this section, we experiment on Stackings at relation level. A stacking at relation level consists of many Stackings at model level trained by data set constrained on a relation. For example, there are 11 relations in WNRR. To build a Stacking at relation level for WNRR, we first build a Stacking at model level, and train the model with the data sample in WNRR where the relation is the 1^{st} relation. Next, we repeat the process but with the 2^{nd} relation. To continue till all 11 Stackings at model level are well trained, we define a Stacking at relation level consisting of these 11 Stackings. This is how to train Stackings at relation level.

For evaluating the performance, the Stacking will assign baby Stacking at model level to a query task where the relation corresponds to that used to train the baby Stacking. From a structural point of view, a Stacking at relation level is an ensemble of many ensembles.

With the same goal to test if ensembles with better trained base learners still outperform the best single model, this section shows the results on Stackings at relation level. Additionally, As described in the steps of how to train and evaluate, we expect Stacking at relation level to outperform Stacking at model level and averaging approach, since the Stacking is formed as a set of specialists solving different type of query with the skilled predictor.

For building up trained base learners, here we also used the pretrained models from [19]. To build up the trained meta learner, we train the number of meta learners corresponding to the number Stackings at model level we have to build. So for WNRR, we trained 11 meta learners for the 11 Stackings at model level. In WNRR, the validation set has at least one sample for each relation. However, it is not the case in FB15K-237, where some relations has no sample in its validation set. To be consistent, we train Stacking at relation level with training set for both WNRR and FB15K-237. We train the meta learner on $relation_i$ on training set, and with batch size = max(512, number of samples with $relation_i$), `SGDClassifier(loss='log')`[2] is also applied here. For evaluation, we used test set.

Table 4.13 shows the HITS@10 on WNRR for Stacking at relation level (4^{th} column), comparing to that of averaging and model level. Note that the values on model level (3^{rd} column) is from the ensemble where the meta learner are also trained with training data, so to have meaningful comparison. Values on 2^{nd} is acquired by averaging the transformed scores from base learners, the same as presented in the previous section. The differences are presented in 4^{th} to 6^{th} column. Table 4.14, Table 4.15, and Table 4.16 are presented in the same format on differet data set (WNRR and FB15K-237) and different evaluation (HITS@10 and MRR) respectively.

The results in all tables show not much advance by building up a set of skilled predictors, and, also, averaging the scores, a rather simple approach, has perform nearly as good as other two approaches.

However, if we compare the 6^{th} column in Table 4.3 to the 5^{th} column in Table 4.13, where the only difference is the data used to train the meta learner, we find also not much difference worth mentioning. As the validation set often are used to fine tune the model, this brings up a thinking that the power of validation set in tuning the Stacking for better output is seemingly less than expected. One can have similar observations by comparing the 7^{th} column in Table 4.3 to the 5^{th} column in Table 4.14, comparing the 6^{th} column in Table 4.8 to the 5^{th} column in Table 4.15, and comparing the 7^{th} column in Table 4.8 to the 5^{th} column in Table 4.16.

Stacking	averaging (a)	model level (b)	relation level (c)	(b - a)	(c - a)	(c - b)
R + D	55.06	55.25	55.30	00.19	00.24	00.05
R + C	56.27	56.25	56.16	-00.02	-00.11	-00.10
R + V	54.21	54.18	54.15	-00.03	-00.06	-00.03
D + C	56.33	56.25	56.27	-00.08	-00.06	00.02
D + V	54.48	54.42	54.42	-00.06	-00.06	00.00
C + V	56.22	56.21	56.21	-00.02	-00.02	00.00
R + D + C	57.07	57.12	57.05	00.05	-00.02	-00.06
R + D + V	55.81	55.84	55.90	00.03	00.10	00.06
R + C + V	57.02	57.12	57.12	00.10	00.10	00.00
D + C + V	56.60	56.81	56.78	00.21	00.18	-00.03
R + D + C + V	**57.31**	**57.39**	**57.37**	00.08	00.06	-00.02

*By column, the largest values are in bold.

Table 4.13: Comparing HITS@10 of stacking on WNRR(in %)

Stacking	averaging (a)	model level (b)	relation level (c)	(b - a)	(c - a)	(c - b)
R + D	47.72	47.46	47.43	-00.25	-00.28	-00.03
R + C	49.04	49.01	48.97	-00.03	-00.07	-00.04
R + V	48.07	48.08	48.09	00.01	00.02	00.01
D + C	48.61	48.34	48.31	-00.27	-00.30	-00.04
D + V	46.40	46.39	46.37	-00.01	-00.04	-00.02
C + V	48.57	48.69	48.64	00.12	00.06	-00.05
R + D + C	**49.41**	49.34	49.30	-00.07	-00.11	-00.04
R + D + V	48.10	47.85	47.84	-00.25	-00.26	-00.01
R + C + V	49.37	**49.47**	**49.45**	00.10	00.08	-00.02
D + C + V	48.70	48.63	48.58	-00.07	-00.12	-00.05
R + D + C + V	49.34	49.29	49.28	-00.05	-00.05	-00.01

*By column, the largest values are in bold.

Table 4.14: Comparing MRR of stacking on WNRR(in %)

Stacking	averaging (a)	model level (b)	relation level (c)	(b - a)	(c - a)	(c - b)
R + D	54.66	54.51	54.51	-00.15	-00.15	-00.01
R + C	54.91	54.87	54.87	-00.04	-00.04	00.00
R + V	54.34	54.20	54.20	-00.14	-00.14	00.00
D + C	54.21	54.21	54.23	00.01	00.02	00.01
D + V	54.17	54.15	54.13	-00.02	-00.03	-00.02
C + V	54.30	54.29	54.32	-00.01	00.02	00.03
R + D + C	54.92	54.78	54.77	-00.14	-00.15	00.00
R + D + V	54.90	54.83	54.81	-00.07	-00.09	-00.02
R + C + V	**55.02**	**54.97**	**54.97**	-00.05	-00.05	00.00
D + C + V	54.61	54.61	54.59	00.01	-00.02	-00.02
R + D + C + V	55.01	54.94	54.95	-00.08	-00.07	00.01

*By column, the largest values are in bold.

Table 4.15: Comparing HITS@10 of stacking on WNRR(in %)

Stacking	averaging (a)	model level (b)	relation level (c)	(b - a)	(c - a)	(c - b)
R + D	35.64	35.48	35.50	-00.16	-00.14	00.02
R + C	35.89	35.80	35.81	-00.10	-00.08	00.01
R + V	35.77	35.71	35.71	-00.07	-00.06	00.00
D + C	35.12	35.12	35.13	-00.01	00.00	00.01
D + V	35.23	35.21	35.20	-00.02	-00.03	-00.01
C + V	35.44	35.45	35.44	00.01	00.00	-00.01
R + D + C	35.81	35.70	35.73	-00.11	-00.09	00.03
R + D + V	35.99	35.87	35.88	-00.11	-00.10	00.01
R + C + V	**36.07**	**36.05**	**36.02**	-00.02	-00.05	-00.03
D + C + V	35.53	35.50	35.52	-00.03	-00.02	00.01
R + D + C + V	36.01	35.92	35.93	-00.09	-00.08	00.01

*By column, the largest values are in bold.

Table 4.16: Comparing MRR of stacking on FB15K-237(in %)

To extend, we look into details of how Stacking at relation level perform on different type of relation. Table 4.18 shows how models perform tasks as predicting subject and predicting object, where the tasks are devided into 1:1, 1:N, N:1 and N:N, representing ”1 to 1”, ”1 to N”, ”N to 1” and ”N to N” relation type, respectively. The improvements done by Stackings comparing to the best single model are presented in the last row of the table. We can see the maximum progress is up to 8.43 ppt on ”1 to N” relation type for predicting subject. The progress on ”N to 1” relation type also reaches nearly 7 ppt. However, the same evaluation made on model level shown in Table 4.19 has pretty close achievements. Again, even digging down into relation type tasks, the approach as forming a set of skilled predictors fails to meet our expectation. For FB15K-237, the evaluations on both relation and model level also result in very close performances, without significant improvement in relation type categories. Nevertheless, it is as expected that ensemble does generate better output than best single model in general, as We can see in the work from Wang et al.(2018)[26](see Table 4.20).

Task	Predict subject				Predict object			
Relations	1:1	1:N	N:1	N:N	1:1	1:N	N:1	N:N
R	95.24	42.32	16.95	94.69	95.24	22.32	31.20	94.60
D	**97.62**	43.79	17.22	**95.49**	**97.62**	21.26	35.04	**95.40**
C	**97.62**	**45.89**	**21.72**	95.31	**97.62**	**26.53**	**35.51**	95.31
V	**97.62**	43.16	11.84	95.04	**97.62**	18.53	31.20	94.96

*By column, the largest values are in bold.

Table 4.17: Detail results on single model on WNRR(in %)

Task	Predict subject				Predict object			
Relations	1:1	1:N	N:1	N:N	1:1	1:N	N:1	N:N
R + D	**97.62**	48.00	21.12	95.58	97.62	24.84	38.13	95.31
R + C	**97.62**	50.11	22.66	95.49	**97.62**	27.16	38.94	95.22
R + V	**97.62**	47.37	19.23	95.22	**97.62**	24.00	36.18	94.96
D + C	**97.62**	50.32	21.92	95.49	**97.62**	26.74	40.08	95.40
D + V	**97.62**	47.58	18.09	95.75	**97.62**	24.21	37.53	95.49
C + V	**97.62**	51.58	20.91	95.66	**97.62**	28.21	39.74	95.49
R + D + C	**97.62**	52.63	**23.00**	95.66	**97.62**	27.16	41.22	95.49
R + D + V	**97.62**	50.53	20.85	95.58	**97.62**	26.74	39.54	95.31
R + C + V	**97.62**	**54.32**	22.53	95.58	**97.62**	**28.84**	41.09	95.31
D + C + V	**97.62**	52.63	21.32	**95.84**	**97.62**	27.79	41.43	**95.49**
R + D + C + V	**97.62**	**54.32**	22.33	95.75	**97.62**	28.00	**42.43**	95.40
best single	97.62	45.89	21.72	95.49	97.62	26.53	35.51	95.40
best ensemble	97.62	54.32	23.00	95.84	97.62	28.84	42.43	**95.49**
improvement	0.00	8.43	1.28	0.35	0.00	2.31	6.92	0.09

*By column, the largest values are in bold.

Table 4.18: Detail results on Stacking at relation level on WNRR(in %)

Task	Predict subject				Predict object			
Relations	1:1	1:N	N:1	N:N	1:1	1:N	N:1	N:N
R	95.24	42.32	16.95	94.69	95.24	22.32	31.20	94.60
D	**97.62**	43.79	17.22	95.49	**97.62**	21.26	35.04	95.40
C	**97.62**	45.89	21.72	95.31	**97.62**	26.53	35.51	95.31
V	**97.62**	43.16	11.84	95.04	**97.62**	18.53	31.20	94.96
R+D	**97.62**	48.21	21.12	95.58	**97.62**	24.84	37.86	95.31
R+C	**97.62**	50.95	22.66	95.49	**97.62**	27.16	39.07	95.22
R+V	**97.62**	47.79	19.17	95.22	**97.62**	24.21	36.18	94.96
D+C	**97.62**	50.53	21.86	95.49	**97.62**	26.74	40.01	95.40
D+V	**97.62**	47.37	18.16	95.75	**97.62**	24.21	37.53	95.49
C+V	**97.62**	52.21	20.91	95.66	**97.62**	27.58	39.81	95.40
R+D+C	**97.62**	52.84	**23.07**	95.66	**97.62**	27.37	41.29	95.49
R+D+V	**97.62**	50.53	20.85	95.58	**97.62**	26.74	39.34	95.22
R+C+V	**97.62**	**54.53**	22.39	95.58	**97.62**	**28.63**	41.22	95.31
D+C+V	**97.62**	52.84	21.39	**95.84**	**97.62**	27.79	41.36	**95.58**
R+D+C+V	**97.62**	54.32	22.33	95.75	**97.62**	28.21	**42.43**	95.40
best single	97.62	45.89	21.72	95.49	97.62	26.53	35.51	95.40
best ensemble	97.62	54.53	23.07	95.84	97.62	28.63	42.43	95.58
improvement	0.00	8.64	1.35	0.35	0.00	2.10	6.92	0.18

*By column, the largest values are in bold.

Table 4.19: Detail results on Stacking at model level on WNRR(in %)

Task	Predict subject				Predict object			
Relations	1:1	1:N	N:1	N:N	1:1	1:N	N:1	N:N
TransE	75.8	91.9	41.4	82.2	75.5	51.1	91.9	84.7
HolE	80.4	69.5	44.7	77.4	79.0	57.8	59.1	79.0
RESCAL	43.1	75.7	17.7	62.0	42.4	21.3	79.2	65.8
R+H+T	**87.5**	**94.3**	**55.2**	**86.7**	**87.0**	**65.0**	**93.3**	**89.4**

Table 4.20: Detail results on Stacking at relation level on FB15K from Wang et al.(2018)[26](in %)

4.3 Fine-tuning

The early experiments have shown that ensembles do outperform well trained base learners, which answers the question we set up the present work for. More over, we have tested many ensembles and seen how better they perform. Since the training process for those ensembles runs separately on base learners and meta learners, we wonder whether the performance will improve by continuing training on them. In other words, we want to fine-tune the ensembles with well trained base learners

and meta learner and try to enhance those already good results.

Fine-tuning proceeds with settings with one batch size and one training type as one process, which is the same for joint learning in Section 4.4. The following experiments on fine-tuning and joint learning involve 5 Stackings with settings and notations presented in Table 4.21, applying the extended LibKGE. Also, in fine-tuning, we apply the pretrained KGE models as base learners, which assemble into 5 Stackings with the same setting on batch size and training type respectively. The idea is to test whether the Stacking will perform better with all the base learners at their best shapes. Because an issue related to the scoring functions of ConvE remained unsolved during the implementation, the experiments do not include ConvE.

data set	batch size	training type	Stacking
FB15K-237	1024	NegSam	S_1
	512	1vsAll	S_2
WNRR	128	KvsAll	S_3
	1024	KvsAll	S_4
	512	1vsAll	S_5

Table 4.21: Settings for experiments

Table 4.22 showing the best performance of KGE models with different batch size and training type settings on both FB15K-237 and WNRR[19] serves as a reference for comparison in the following content.

			RESCAL		DistMult		ComplEx	
data set	batch size	training type	HITS@10	MRR	HITS@10	MRR	HITS@10	MRR
FB15K-237	1024	NegSam	44.50	**53.13**	**53.60**	23.69	34.28	34.77
	512	1vsAll	**54.06**	**52.09**	52.15	35.59	33.74	33.98
WNRR	128	KvsAll	**51.72**	**49.86**	47.64	46.66	44.03	43.59
	1024	KvsAll	**50.34**	**53.05**	48.55	45.55	45.18	44.65
	512	1vsAll	49.95	**50.29**	**54.74**	44.01	44.06	47.49

*By row, the largest values are in bold; *NegSam* stands for *Negative Sampling*.

Table 4.22: Best performance from Ruffinelli et al.(2020)[19](in %)

To perform meaningful comparison, we also build up the baselines, where we simply average the scores and aggregate the scores by meta learner. Table 4.23

shows the performance of the Stackings in Table 4.21. By simply averaging the scores, the performance is better than the best single KGE model on WNRR, but worse on FB15K-237. Between averaging and applying meta learner, there is no significant difference on performance, with the most 0.32 ppt advance with meta learner by *1vsAll* on WNRR.

Stacking	(averaging)		(meta learner)		(c - a)	(d - b)
	HITS@10 (a)	MRR (b)	HITS@10 (c)	MRR (d)		
S_1	53.16	34.09	53.10	34.07	-0.06	-0.01
S_2	53.74	35.29	53.66	35.25	-0.07	-0.04
S_3	55.22	48.45	55.28	48.49	0.06	0.04
S_4	54.99	48.20	55.01	48.06	0.02	-0.14
S_5	55.47	48.17	55.79	48.27	0.32	0.10

Table 4.23: Stackings on WNRR as baseline for fine-tune(in %)

To fine-tune, we experiment on three settings:

- Setting 1 (Set_1)
  - apply best configurations of all base learners
  - apply pretrained embeddings of all base learners
  - set 1 as base learners' coefficients and 0 as intercept for meta learner
  - run exact 50 epochs

- Setting 2 (Set_2)
  - apply best configurations of all base learners
  - apply pretrained embeddings of all base learners
  - set 1 as base learners' coefficients and 0 as intercept for meta learner
  - run exact 100 epochs

- Setting 3 (Set_3)
  - apply best configurations of all base learners
  - apply pretrained embeddings of all base learners
  - apply the pretrained meta learners
  - run exact 50 epochs

Stacking	average (a)	Set_1	Set_2 (b)	(b - a)	meta learner	Set_3 (c)	(c - a)
S_1	53.16	53.18	53.18	0.01	53.10	53.10	-0.06
S_2	53.74	54.02	54.02	0.28	53.66	53.66	-0.07
S_3	55.22	53.94	53.94	-1.28	55.28	55.28	0.06
S_4	54.99	54.96	54.96	-0.03	55.01	55.01	0.02
S_5	55.47	55.28	55.28	-0.19	55.79	55.79	0.32

Table 4.24: Performance on HITS@10 on fine-tune settings(in %)

Stacking	average (a)	Set_1	Set_2 (b)	(b - a)	meta learner	Set_3 (c)	(c - a)
S_1	34.09	34.03	34.03	-0.05	34.07	34.07	-0.01
S_2	35.29	35.5	35.5	0.21	35.25	35.25	-0.04
S_3	48.45	47.84	47.84	-0.61	48.49	48.49	0.04
S_4	48.20	48.18	48.18	-0.01	48.06	48.06	-0.14
S_5	48.17	47.99	47.99	-0.18	48.27	48.27	0.1

Table 4.25: Performance on MRR on Fine-Tune Settings(in %)

In both Table 4.24 and Table 4.25 , we can evaluate the values from 2^{nd} to 4^{th} column as a group, since the weights for base learners are the same (before fine-tuning) in all 3 cases. Also, we can make similar comments on values from 6^{th} to 7^{th} column, where the results from both cases are applied with trained meta learner (before fine-tuning).

Experiments on Set_1 and Set_2 result the same, which means the best model attained are the same in both 50-epoch run and 100-epoch run. In other words, the training starting from 51^{st} to 100^{th} contribute no better output. As for Set_3, which starts the fine-tune with trained meta learner, there is no better output during the first 50 epoch run. Table 4.26 shows the performance with the last epoch in each experimental settings. For example, in Set_1 experiment, the values in column *last* is the performance from the 50^{th} epoch. In these settings, the performance from *last* is worse than the *best*.

		HITS@10		MRR	
Setting	Stacking	best	last	best	last
Set_1	S_1	53.18	49.64	34.03	30.10
	S_2	54.02	53.76	35.50	35.10
	S_3	53.94	50.13	47.84	45.42
	S_4	54.96	53.91	48.18	47.73
	S_5	55.28	52.04	47.99	45.74
Set_2	S_1	53.18	50.00	34.03	30.47
	S_2	54.02	53.85	35.50	35.22
	S_3	53.94	51.47	47.84	46.44
	S_4	54.96	53.06	48.18	46.86
	S_5	55.28	43.17	47.99	38.71
Set_3	S_1	53.10	50.81	34.07	31.60
	S_2	53.66	52.78	35.25	34.26
	S_3	55.28	55.28	48.49	48.47
	S_4	55.01	54.96	48.06	48.17
	S_5	55.79	55.41	48.27	48.06

*Column *last* shows the results from the last epoch.

Table 4.26: Comparing with the last epoch(in %)

4.4 Joint Learning

In this section, we run joint learning with the goal to see how the performance will become if train ensembles from scratch, which means to train ensembles without pretrained base learners and meta learner.

We experiment on the Stackings defined in Table 4.21 with the setting as below:

- apply the best configurations of all base learners,
- set 1 as base learners' coefficients and 0 as intercept for meta learner, and
- run exact 400 epochs.

	HITS@10		MRR	
Stacking	best	last	best	last
S_1	0.1393	0.0757	0.1085	0.0766
S_2	0.1099	0.1075	0.0921	0.0891
S_3	0.0160	0.0160	0.0233	0.0219
S_4	0.0000	0.0000	0.0172	0.0170
S_5	0.0000	0.0000	0.0213	0.0208

*Column *last* shows the results from the last epoch.

Table 4.27: Performance of joint learning(in %)

Table 4.27 shows the results from joint learning, where we also present the results from the last epoch. The column *best* shows the best ensemble trained during the 400 epoch run; the column *last* shows the ensemble trained on the 400^{th} epoch, where HITS@10 and MRR are worse than the *best*. From this table, we suppose that the model reached to a peak performance during the 400 iterations, and than became worse afterwards. Also, joint learning perform poorly especially on HITS@10 with WNRR.

Chapter 5

Conclusion

5.1 Summary

In the present work, we extend LibKGE and perform experiments on Stacking at both model and relation level. We also try fine-tuning and joint learning in KGEs, which results in some interesting discoveries.

- In Stacking experiments, the performance is already good by averaging the scores, whereas applying logistic regression as meta learning doesn't enhance the performance much.
- In the case where we apply training set to train the meta learner, indifferent to the numbers of base learners in an ensemble, the coefficients of models are centralized to the same values rounding up to 2 decimals. We suppose min-maxing the scores leads to the phenomena. With the models contributing the scores on the same basis, the weights bringing out the best performance should reasonably be robust.
- The performances of the stacking at relational level are similar to that of the stacking at model level, which is not as expected. Since ensemble at relation level aims at bringing the best out of the base learners, we suppose the results should be slightly better, if not significantly.
- The fine-tune with trained meta learner doesn't affect much on the performance. We suppose that, with the trained meta learner, the ensemble may has all its parameters fallen on a saddle point as the iterations go on in the preceding training.
- With fine-tuning, the Stackings without trained meta learner perform better than those with. We suppose that this setting Unleashes the possibility of

better output. However, this can also result in getting negative coefficients for the meta learner, which leads to ensembles with less explainability.

- The performance from joint learning is really poor. We suppose that the minimum efforts required to level up the performance of the base learners to a threshold are too much. With the best configurations but newly initialized embeddings for the base learners, along with the coefficients from the meta learner being updated among each iterations, the path to the better performances are rather challenging. Also, the number of parameters from all base learners, taking the embedding dimension into account, are too high, which also could have contributed to this results. In other words, all the settings bring too much stress on ONE training process with the quest for the best performance.

5.2 Future Work

Aside from Stacking, Boosting and Bagging are also applied in many areas related to machine learning. One could also test the feasibility of other two frameworks in KGEs. Also, there are other training approaches could be experimented on ensembles in KGEs, i.e., alternative training. It would be interesting to find a way to train ensemble in KGEs from scratch and have the performance as good as that of Stacking.

Bibliography

[1] Amit singhal. 2012. introducing the knowledge graph: things, not strings. google blog. https://github.com/uma-pi1/kge.

[2] Sgdclassifier from scikit-learnlibrary. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html.

[3] Luai Al Shalabi, Zyad Shaaban, and Basel Kasasbeh. Data mining: A preprocessing engine. *Journal of Computer Science*, 2(9):735–739, 2006.

[4] 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*, pages 1247–1250, 2008.

[5] Yuanfei Dai, Shiping Wang, Neal N Xiong, and Wenzhong Guo. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics*, 9(5):750, 2020.

[6] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. *arXiv preprint arXiv:1707.01476*, 2017.

[7] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. Knowledge graphs. *arXiv preprint arXiv:2003.02320*, 2020.

[8] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388*, 2020.

[9] Damir Juric, Giorgos Stoilos, André Melo, Jonathan Moore, and Mohammad Khodadadi. A system for medical information extraction and verification from unstructured text. In *AAAI*, pages 13314–13319, 2020.

[10] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. Knowledge base completion: Baselines strike back. *arXiv preprint arXiv:1705.10744*, 2017.

[11] Denis Krompaß and Volker Tresp. Ensemble solutions for link-prediction in knowledge graphs. In *2nd Workshop*, 2015.

[12] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195, 2015.

[13] Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In *International Semantic Web Conference*, pages 3–20. Springer, 2018.

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

[15] Makoto Miwa and Yutaka Sasaki. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, Doha, Qatar, October 2014. Association for Computational Linguistics.

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

[17] Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1015–1024, 2017.

[18] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, 2013.

[19] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*, 2020.

[20] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[21] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, 2007.

[22] Andreas Töscher, Michael Jahrer, and Robert M Bell. The bigchaos solution to the netflix grand prize. *Netflix prize documentation*, pages 1–52, 2009.

[23] 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*, pages 57–66, 2015.

[24] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML), 2016.

[25] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[26] Yanjie Wang, Rainer Gemulla, and Hui Li. On multi-relational link prediction with bilinear models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[27] 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.

[28] Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. Joint extraction of entities and relations based on a novel tagging scheme. *arXiv preprint arXiv:1706.05075*, 2017.

[29] Zhi-Hua Zhou. Ensemble learning. *Encyclopedia of biometrics*, 1:270–273, 2009.

Declaration of Academic Integrity

I, Kuan Min Chen, hereby declare that the master's thesis entitled "Revisiting Ensembles for Knowledge Base Completion" is entirely my own work, and that I have employed no sources or aids other than the ones listed. I have clearly marked and acknowledged all ideas and illustrations that have been taken directly or indirectly from the works of others. I also confirm that this thesis has not been submitted in this form, or a similar form, to any other academic institution.

Mannheim, den 04.09.2020 Signature