Dynamic Ensembling of Textual and Structure-Based Models for Knowledge Graph Completion

Anonymous ACL submission

Abstract

We consider two popular approaches to Knowledge Graph Completion (KGC): textual models that rely on textual entity descriptions, and structure-based models that exploit the connectivity structure of the Knowledge Graph (KG). Preliminary experiments show that these approaches have complementary strengths: structure-based models perform exceptionally well when the gold answer is easily reachable from the query head in the KG, while textual models exploit descriptions to give good performance even when this is not the case. In response, we propose a novel method for learning query-dependent ensemble weights to combine these approaches by using the distributions of scores assigned by individual models to all candidate entities. Our ensemble baseline achieves state-of-the-art results on three standard KGC datasets, with up to 6.8 pt MRR and 8.3 pt Hits@1 gains over best individual models.

1 Introduction

The task of Knowledge Graph Completion (KGC) can be described as inferring missing links in a Knowledge Graph (KG) based on given triples $(\mathtt{h}, \mathtt{r}, \mathtt{t})$, where $\mathtt{r}$ is a relation that exists between the head entity $\mathtt{h}$ and the tail entity $\mathtt{t}$. Several KGC approaches, such as NBFNet (Zhu et al., 2021) and RGHAT (Zhang et al., 2020), exploit the underlying graph structure, often using GNNs. On the other hand, textual models such as SimKGC (Wang et al., 2022) and KG-BERT (Yao et al., 2019) leverage pre-trained large language models (LLMs) such as BERT to utilize textual descriptions of the KG entities and relations for KGC.

Our preliminary experiments suggest that when the gold answer $\mathtt{t}$ for query $(\mathtt{h}, \mathtt{r}, ?)$ is reachable from $\mathtt{h}$ using a path of reasonable length in the KG, structure-based models tend to outperform textual models. In contrast, textual models use textual descriptions to perform better than structure-based models when $\mathtt{t}$ is not easily reachable from $\mathtt{h}$. Motivated by our findings, we seek to explore how ensembling, an approach currently underrepresented in KGC literature, can effectively harness the complementary strengths of these models.

Consequently, we propose a novel, simple, model-agnostic and lightweight method for learning ensemble weights such that the weights are (i) query-dependent and (ii) learned from statistical features obtained from the distribution of scores assigned by individual models to all candidate entities. This approach results in a new state-of-the-art baseline when applied on two strong KGC models: SimKGC and NBFNet, which are textual and structure-based in nature, respectively.

On three KGC datasets, we find that applying our ensembling method to SimKGC and NBFNet consistently improves KGC performance, outperforming best individual models by up to 6.8 pt MRR and 8.3 pt Hits@1. To the best of our knowledge, our results are state-of-the-art for all three datasets. Further experiments (including a fourth dataset to which NBFNet does not scale) show that our approach generalises to ensembling with another KG embedding model, RotatE (Sun et al., 2019), with similar gains. We also demonstrate that our ensemble learning methodology outperforms conventional model-combination techniques such as static ensembling (where the ensemble weight is a tuned constant hyperparameter) and re-ranking. We release all code to guide future research.

2 Background and Related Work

Task: We are given an incomplete KG $\mathcal{K} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ consisting of entities $\mathcal{E}$, relation set $\mathcal{R}$ and set of triples $\mathcal{T} = \{(\mathtt{h}, \mathtt{r}, \mathtt{t})\}$ (where $\mathtt{h}, \mathtt{t} \in \mathcal{E}$ and $\mathtt{r} \in \mathcal{R}$). The goal of KGC is to answer queries of the form $(\mathtt{h}, \mathtt{r}, ?)$ or $(?, \mathtt{r}, \mathtt{t})$ to predict missing links, with corresponding answers $\mathtt{t}$ and $\mathtt{h}$. We model $(?, \mathtt{r}, \mathtt{t})$ as $(\mathtt{t}, \mathtt{r}^{-1}, ?)$ queries in this work.

Overview of Related Work: We focus on three types of KGC models. The first type consists of Graph Neural Network (GNN) based models such as NBFNet (Zhu et al., 2021), RGHAT (Zhang et al., 2020) that leverage neighborhood information to train distinct GNN architectures. The second type contains textual models such as KG-BERT (Yao et al., 2019) and SimKGC (Wang et al., 2022) which fine-tune a pre-trained LLM on textual descriptions of entities and relations for KGC. The third type involves models such as RotatE (Sun et al., 2019) and TransE (Bordes et al., 2013) that learn low-dimensional embeddings for entities and relations and compose them by employing unique scoring functions. Unification of these approaches has not been extensively studied in KGC literature, with previous works such as VEM2L (He et al., 2022) focusing on encouraging these models to learn from each other during training. Since our main experiments are based on NBFNet, SimKGC and RotatE, we describe these in some detail next.

NBFNet: Neural Bellman-Ford Network (NBFNet) is a path-based link prediction model that introduces neural functions into the Generalized Bellman-Ford (GBF) Algorithm (Baras and Theodorakopoulos, 2010), which in turn models the path between two nodes in the KG through generalized sum and product operators. This formulates a novel GNN framework that learns entity representations for each candidate tail $\mathtt{t}$ conditioned on $\mathtt{h}$ and $\mathtt{r}$ for each query $(\mathtt{h}, \mathtt{r}, ?)$. The score of any candidate $\mathtt{t}$ is then computed by applying an MLP to its embedding.

SimKGC: SimKGC is an LLM-based KGC model that employs a bi-encoder architecture to generate the score of a given triple $(\mathtt{h}, \mathtt{r}, \mathtt{t})$. The model considers two pre-trained BERT (Devlin et al., 2019) models. The first model is finetuned on a concatenation of textual descriptions of $\mathtt{h}$ and $\mathtt{r}$ to generate their joint embedding $\mathbf{e}_{hr}$ and the second model is finetuned on the textual description of $\mathtt{t}$ to generate the embedding $\mathbf{e}_t$. The score for the triple is the cosine similarity between $\mathbf{e}_{hr}$ and $\mathbf{e}_t$.

RotatE: RotatE is a KG Embedding model that maps entities and relations to a complex vector space and models each relation $\mathtt{r}$ as a complex rotation from the head $\mathtt{r}$ to the tail $\mathtt{t}$ for triple $(\mathtt{h}, \mathtt{r}, \mathtt{t})$. More specifically, the scoring function of RotatE is $\|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|$ where $\mathbf{h}, \mathbf{t} \in \mathbb{C}^k$ are the complex embedding of $\mathtt{h}$ and $\mathtt{t}$, and $\circ$ is the Hadamard product.

3 Dynamic Ensemble Methodology

Our goal is to dynamically ensemble k KGC models $\mathtt{M_i}$, which may be textual or structure-based, to maximize performance. Each model $\mathtt{M_i}$ assigns a score $\mathtt{M_i}(\mathtt{h}, \mathtt{r}, \mathtt{t})$ to all candidate tails $\mathtt{t} \in \mathcal{E}$ for query $\mathtt{q} = (\mathtt{h}, \mathtt{r}, ?)$. These models are trained independently and their parameters are frozen before ensembling. We formulate the ensemble $\mathtt{E}$ as:

$$\mathtt{E}(\mathtt{h}, \mathtt{r}, \mathtt{t}) = \sum_{i=1}^{k} \mathtt{w_i}(\mathtt{q}) \mathtt{M_i}(\mathtt{h}, \mathtt{r}, \mathtt{t})$$

where $\mathtt{E}(\mathtt{h}, \mathtt{r}, \mathtt{t})$ is the ensemble score for $\mathtt{t}$ given query $\mathtt{q} = (\mathtt{h}, \mathtt{r}, ?)$. We first normalize these scores as described below.

Normalization: To bring the distribution of scores assigned by each model $\mathtt{M_i}$ over all $\mathtt{t} \in \mathcal{E}$ in the same range for each query, we max-min normalize the scores obtained from all models $\mathtt{M_i}$ separately:

$$\mathtt{M_i}(\mathtt{h}, \mathtt{r}, \mathtt{t}) \leftarrow \mathtt{M_i}(\mathtt{h}, \mathtt{r}, \mathtt{t}) - \min_{\mathtt{t}' \in \mathcal{E}} \mathtt{M_i}(\mathtt{h}, \mathtt{r}, \mathtt{t}')$$

$$\mathtt{M_i}(\mathtt{h}, \mathtt{r}, \mathtt{t}) \leftarrow \frac{\mathtt{M_i}(\mathtt{h}, \mathtt{r}, \mathtt{t})}{\max_{t' \in \mathcal{E}} \mathtt{M_i}(\mathtt{h}, \mathtt{r}, \mathtt{t}')}$$

The scores obtained after normalization lie in the range [0,1] for all models. We next describe the simple model used to learn the query-dependent ensemble weights $\mathtt{w_i}$.

Model: We extract the following features from the score distribution of each model $\mathtt{M_i}$:

$$\mathtt{f}(\mathtt{M_i}, \mathtt{q}) = \underset{\mathtt{t}' \in \mathcal{E}}{\mathrm{mean}}(\mathtt{M_i}(\mathtt{h}, \mathtt{r}, \mathtt{t}')) || \underset{\mathtt{t}' \in \mathcal{E}}{\mathrm{var}}(\mathtt{M_i}(\mathtt{h}, \mathtt{r}, \mathtt{t}'))$$

In the above equations, $\mathrm{mean}()$ and $\mathrm{var}()$ are the standard mean and variance functions respectively, whose outputs are concatenated to obtain the feature. This choice is driven by the insight that the variance and mean of the distribution of scores computed by any model over $\mathcal{E}$ is correlated to the model confidence. A more detailed discussion, along with an exploration of other possible feature sets can be found in Appendix C.

Next, we concatenate these features for all $\mathtt{M_i}$ to obtain a final feature vector that is passed to an independent 2-layer MLP ($\mathtt{MLP_i}$) for each model $\mathtt{M_i}$ to learn query-dependent $\mathtt{w_i}$:

$$\mathtt{w_i}(\mathtt{q}) = \mathtt{MLP_i}(\mathtt{f}(\mathtt{M_1}, \mathtt{q}) || \mathtt{f}(\mathtt{M_2}, \mathtt{q}) || ... || \mathtt{f}(\mathtt{M_k}, \mathtt{q}))$$

Intuitively, this concatenation informs each MLP about the relative confidence of all models regarding their predictions, enhancing the ensemble weight computation for corresponding models. Note that our approach is agnostic to models $\mathtt{M_i}$.

Table 1: Results on four datasets for our baselines and approach. [NBF], [Sim] and [RotE] represent NBFNet, SimKGC and RotatE models. [NBF] does not scale up to YAGO3-10. Best individual model results are underlined.

Model	WN18RR			FB15k-237			CoDex-M			YAGO3-10		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
[Sim]	66.4	58.5	80.3	32.1	23.2	50.5	29.1	21.0	45.2	15.8	10.0	27.3
[NBF]	54.2	48.6	65.7	40.5	31.0	59.4	35.3	27.0	51.4	-	-	-
[RotE]	47.7	43.9	55.2	33.7	24.0	53.2	33.5	26.3	46.9	49.3	39.9	67.1
[Sim]+[NBF]	**73.2**	**66.9**	**85.7**	42.7	33.2	61.5	38.9	30.5	**54.8**	-	-	-
[Sim]+[RotE]	68.0	60.7	80.7	36.6	26.9	56.3	36.3	28.1	51.7	**50.6**	**41.3**	**67.9**
[Sim]+[NBF]+[RotE]	**73.2**	**66.9**	**85.7**	**43.0**	**33.4**	**62.0**	**40.0**	**31.2**	**54.8**	-	-	-

Our experiments in this paper involve only one textual model. Therefore, we learn the ensemble weights for the other models with respect to this textual model, which is assigned a fixed weight of 1. This decreases the parameter count while still being as expressive as learning distinct ensemble weights for all models. The method for learning these other weights is unchanged.

Loss Function: We train our model on the validation set (traditionally used to tune ensemble weights) using margin loss between the score of the gold entity and a set of negative samples. The train set is not used since all models are likely to give high-confidence predictions on its triples. If the gold entity is $\mathtt{t}^*$ and the set of negative samples is $\mathtt{N}$, the loss function $\mathcal{L}$ for query $\mathtt{q} = (\mathtt{h}, \mathtt{r}, ?)$ is:

$$\mathcal{L} = \sum_{\mathtt{t} \in \mathtt{N}} \max\left(\mathtt{E}(\mathtt{h}, \mathtt{r}, \mathtt{t}) - \mathtt{E}(\mathtt{h}, \mathtt{r}, \mathtt{t}^*) + \mathtt{m}, 0\right)$$

where $\mathtt{m}$ is the margin hyperparameter. This hyperparameter ensures that the generated ensemble weights do not blow up during training.

4 Experiments

Datasets: We use four datasets for evaluation: WN18RR (Dettmers et al., 2018), FB15k-237 (Toutanova and Chen, 2015), CoDex-M (Safavi and Koutra, 2020) and YAGO3-10 (Mahdisoltani et al., 2015). For each triple in the test set, we answer queries $(\mathtt{h}, \mathtt{r}, ?)$ and $(\mathtt{t}, \mathtt{r}^{-1}, ?)$ with answers $\mathtt{t}$ and $\mathtt{h}$. We report the Mean Reciprocal Rank (MRR) and Hits@k (H@1, H@10) under the filtered measures (Bordes et al., 2013). Details and data statistics are in Appendix A.

Baselines: We use SimKGC ([Sim] in tables) as a state of the art textual model baseline. NBFNet ([NBF] in tables) serves as a strong structure-based model baseline. We also present results with RotatE ([RotE] in tables) to showcase the generalisation of our method to KG embedding models. We have reproduced the numbers published by the original authors for these baselines. Since [NBF] does not scale up to YAGO3-10 with reasonable hyperparameters on our hardware, we omit those results. We represent ensembling by + in tables.

Experimental Setup: All baseline models are frozen after training using optimal configurations. Ensemble weights are trained on the validation split, using Adam as the optimizer with a learning rate of 5.0e-5. We use 10,000 negative samples per query. MLP hidden dimensions are set to 16 and 32 for ensemble of 2 and 3 models respectively. MLP weights are initialized uniformly in the range [0, 2]. Training of the ensemble converges in a single epoch, making our method fast and efficient.

Results: We report the results of dynamic ensembling in Table 1 (more details in Appendix B). We observe a notable increase in performance after ensembling with [Sim] for both [NBF] and [RotE], which shows that our approach works with both structure-based and KG embedding models. In particular, we obtain 6.8 pt MRR and 8.4 pt Hits@1 improvement with [Sim] + [NBF] over [Sim] on WN18RR. Ensembling with [Sim] results in substantial performance gains even when it is outperformed by the other models (on FB15k-237, CoDex-M and YAGO3-10). Notably, on YAGO3-10, where [RotE] outperforms [Sim] by 33.5 pt MRR and 29.9 pt Hits@1, we still obtain 1.3 pt MRR and 1.4 pt Hits@1 gain with [Sim] + [RotE] over [RotE]. Results for [Sim] + [NBF] + [RotE] show that ensembling with [RotE] results in marginal gains over [Sim] + [NBF], obtaining up to 1.1 pt MRR and 0.7 pt Hits@1 gain on CoDex-M. We hypothesize that the gains are marginal due to [RotE]'s ability to implicitly capture and exploit structural information (explored in more detail in Appendix D and E), making it somewhat redundant in the presence of [NBF]. To the best of our knowledge, our best results on WN18RR, FB15k-237 and CoDex-M datasets are state of the art.

5 Analysis

We perform three further analyses to answer the following questions: **Q1**. How does the behavior of textual and structure-based models vary with reachability? **Q2.** Do the weights learned by our approach follow expected trends with reachability? **Q3.** Does our approach improve performance over conventional model-combination techniques?

Reachability Ablation: To answer **Q1**, we divide the test set for each dataset into 'reachable' and 'unreachable' splits. A triple $(\mathtt{h}, \mathtt{r}, \mathtt{t})$ is part of the reachable split if $\mathtt{t}$ can be reached from $\mathtt{h}$ with a path of length at most $\mathtt{l}$ (= 2) in the KG. If not, it is put in the unreachable split. We present split-wise results for [NBF], [Sim] and [Sim]+[NBF] on the WN18RR and FB15k-237 datasets in Table 2.

Table 2: Results on Reachable and Unreachable Split of [NBF], [Sim] and [NBF] + [Sim] on WN18RR and FB15k-237. Best individual model results are underlined.

Dataset	Model	Reachable Split			Unreachable Split		
		MRR	H@1	H@10	MRR	H@1	H@10
WN18RR	[NBF]	89.7	86.8	95.7	26.0	18.3	41.8
	[Sim]	85.3	79.4	94.5	51.8	42.3	69.0
	[Sim]+[NBF]	**93.9**	**91.7**	**97.4**	**56.8**	**47.0**	**76.4**
FB15k−237	[NBF]	44.8	35.2	64.0	28.2	19.3	46.2
	[Sim]	31.5	22.6	49.6	30.0	21.2	48.2
	[Sim]+[NBF]	**46.5**	**36.8**	**65.3**	**32.3**	**23.1**	**50.6**

We observe that [Sim] outperforms [NBF] on the unreachable split (by up to 25.8 pt MRR and 24.0 pt Hits@1 for WN18RR), while [NBF] outperforms [Sim] on the reachable split (by up to 13.3 pt MRR and 12.6 pt Hits@1 for FB15k-237). This is because [NBF] can easily exploit knowledge of the KG structure on the reachable split, while [Sim] can instead use BERT to leverage textual descriptions to perform better on the unreachable split. The performance gap between [Sim] and [NBF] on the unreachable split is notably larger for WN18RR than for FB15k-237, which can be attributed to the sparsity of the WN18RR dataset, which also has several unseen entities in this split. In such cases, [Sim] achieves reasonable performance, whereas [NBF] lacks any paths for reasoning. Our ensemble obtains substantial gains over best individual models on both splits, with 4.2 pt MRR and 4.9 pt Hits@1 gain on the reachable split and 5.0 pt MRR and 4.7 pt Hits@1 gain on the unreachable split for WN18RR. More details in Appendix D.

Analysis of Ensemble Weights: To answer **Q2**, we study the mean of the ensemble weight $\mathtt{w}_2$ for [Sim] + [NBF] over the queries in the reachable and unreachable splits of the datasets we use. We observe that this mean is consistently larger (by a margin of up to 17% for WN18RR) on the reachable split than the unreachable split. This is because [NBF] tends to give better performance on the reachable split, and a larger $\mathtt{w}_2$ gives it more importance in the ensemble. More details and numbers are in Appendix F, which also shows the non-trivial standard deviation of $\mathtt{w}_2$, demonstrating our approach's ability to adjust it depending on individual queries.

Comparison with Conventional Techniques: To answer **Q3**, we present results for static ensembling and re-ranking using [Sim] and [NBF] for WN18RR and FB15k-237 datasets in Table 3. 'Static ensembling' involves manually tuning the ensemble weight as a constant on the validation set. For [NBF]-[Sim] re-ranking (Han et al., 2020), we consider the top 100 entities by score from [NBF] for each query and re-rank them according to their [Sim] score. The rest of the entities are ranked according to [NBF]. We also present results for [Sim]-[NBF] re-ranking for comparison.

Table 3: Comparison of Static and Dynamic Ensembling and Re-ranking. [X]-[Y] re-ranking indicates re-ranking of top 100 predictions from [X] using [Y].

Dataset	Approach	MRR	H@1	H@10
WN18RR	[NBF]-[Sim] Re-rank	63.5	57.1	74.9
	[Sim]-[NBF] Re-rank	60.7	53.3	76.0
	Static Ensemble	72.2	65.5	85.4
	Dynamic Ensemble	**73.2**	**66.9**	**85.7**
FB15k−237	[NBF]-[Sim] Re-rank	32.7	23.3	52.5
	[Sim]-[NBF] Re-rank	38.9	30.0	56.5
	Static Ensemble	41.9	32.7	60.1
	Dynamic Ensemble	**42.7**	**33.2**	**61.5**

We find that our 'dynamic ensemble' approach outperforms both re-ranking and 'static ensembling' across datasets. Notably, our approach beats re-ranking by 9.7 pt MRR and 9.8 pt Hits@1, and static ensembling by 1.0 pt MRR and 1.4 pt Hits@1 on the WN18RR dataset. This highlights the utility of our dynamic ensembling approach. A detailed analysis of this improvement is in Appendix G.

6 Conclusion and Future Work

We present a simple, novel, model-agnostic and lightweight dynamic ensembling approach for the task of KGC, while also highlighting the complementary strengths of textual and structure-based KGC models. Our state-of-the-art results for [Sim] + [NBF] over three standard KGC datasets creates a new competitive ensemble baseline for future KGC. We release all code for future research. Future work includes better and tighter unification methods for textual and structure-based KGC.

Limitations

We do not consider Neuro-Symbolic KGC approaches in this work, which have also recently started to give competitive results with other KGC approaches, through models such as RNNLogic (Qu et al., 2021). Our experiments consider ensembling of one textual model with multiple structural models. This is because most textual models in recent KGC literature are not competitive with SimKGC (Wang et al., 2022). The ensembling of multiple textual models with multiple structure-based models would be a possible future work. In models with substantial validation splits, learning query embeddings to augment the features we use to compute ensemble weights is also a possibility.

Ethics Statement

We anticipate no substantial ethical issues arising due to our work on ensembling textual and structure-based models for KGC. Our work relies on other baseline models for ensembling. This may propagate any bias present in these baseline models, however ensembling may also reduce these biases.

References

Baras S Baras and George Theodorakopoulos. 2010. Path Problems in Networks. *Synthetic Lectures in Communication Networks*, 3:1–77.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NeurIPS*. Curran Associates, Inc.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Shuguang Han, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2020. Learning-to-Rank with BERT in TF-Ranking. In *Arxiv*.

Tao He, Ming Liu, Yixin Cao, Tianwen Jiang, Zihao Zheng, Jingrun Zhang, Sendong Zhao, and Bing Qin. 2022. Vem^{2}l: A plug-and-play framework for fusing text and structure knowledge on sparse knowledge graph completion.

Farzaneh Mahdisoltani, Joanna Asia Biega, and Fabian M. Suchanek. 2015. Yago3: A Knowledge Base from Multilingual Wikipedias. In *Conference on Innovative Data Systems Research*.

Meng Qu, Junkun Chen, Louis-Pascal A. C. Xhonneux, Yoshua Bengio, and Jian Tang. 2021. RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs. In *ICLR*, pages 1–21.

Tara Safavi and Danai Koutra. 2020. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8328–8350, Online. Association for Computational Linguistics.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus Latent Features for Knowledge Base and Text Inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China. Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *ICML*, page 2071–2080. JMLR.org.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022. SimKGC: Simple Contrastive Knowledge Graph Completion with Pre-trained Language Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4281–4294, Dublin, Ireland. Association for Computational Linguistics.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for Knowledge Graph Completion. In *AAAI Conference on Artificial Intelligence*.

Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhi-Ping Shi, Hui Xiong, and Qing He. 2020. Relational Graph Neural Network with Hierarchical Attention for Knowledge Graph Completion. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 9612–9619. AAAI Press.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural Bellman-Ford Networks: A General Graph Neural Network Framework for Link Prediction. In *Advances in Neural Information Processing Systems*.

Table 4: Statistics of Knowledge Graph datasets

Datasets	#Entities	#Relations	#Training	#Validation	#Test
FB15k-237	14541	237	272,115	17,535	20,446
WN18RR	40,943	11	86,835	3,034	3,134
Yago3-10	123182	36	1,079,040	5000	5000
CoDex-M	17050	71	185584	10310	10311

Table 5: Results of on four datasets: WN18RR, FB15k-237, Yago3-10 and CoDex-M with ensemble of textual and structure-based models. [NBF], [Sim], [RotE] and [Comp] represents NBFNet, SimKGC, RotatE and CompleX models respectively. [NBF] does not scale to YAGO3-10. Best individual model results are underlined.

Model	**WN18RR**					**FB15k-237**				
	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
[Sim]	174.0	66.4	58.5	71.3	80.3	131.9	32.1	23.2	34.6	50.5
[NBF]	699.3	54.2	48.6	56.9	65.7	111.4	40.5	31.0	44.3	59.4
[RotE]	4730.7	47.7	43.9	49.1	55.2	176.6	33.7	24.0	37.4	53.2
[Comp]	5102.6	47.2	42.8	49.2	56.0	180.7	35.7	26.3	39.4	54.7
[Sim]+[NBF]	**56.6**	**73.2**	**66.9**	**76.5**	**85.7**	92.2	42.7	33.2	46.7	61.5
[Sim]+[RotE]	162.7	68.0	60.7	72.2	80.7	116.0	36.6	26.9	40.2	56.3
[Sim]+[Comp]	172.9	68.0	60.8	72.3	80.7	116.3	37.8	28.3	41.2	57.1
[Sim]+[NBF]+[RotE]	**56.6**	**73.2**	**66.9**	**76.5**	**85.7**	**91.5**	**43.0**	**33.4**	**47.0**	**62.0**
[Sim]+[NBF]+[Comp]	**56.6**	**73.2**	**66.9**	**76.5**	**85.7**	92.0	42.8	33.3	46.8	61.5
Model	**CoDex-M**					**Yago3-10**				
	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
[Sim]	284.2	29.1	21.0	31.5	45.2	**497.4**	15.8	10.0	16.2	27.3
[NBF]	337.5	35.3	27.0	39.0	51.4	-	-	-	-	-
[RotE]	502.6	33.5	26.3	36.8	46.9	1866.8	49.3	39.9	55.0	67.1
[Comp]	391.0	35.3	27.7	38.8	49.5	1578.1	49.2	40.1	53.8	66.7
[Sim]+[NBF]	252.1	38.9	30.5	42.7	**54.8**	-	-	-	-	-
[Sim]+[RotE]	293.4	36.3	28.1	40.0	51.7	610.6	**50.6**	**41.3**	**56.0**	**67.9**
[Sim]+[Comp]	296.3	37.5	29.6	41.0	52.4	515.9	49.5	40.5	54.2	66.6
[Sim]+[NBF]+[RotE]	**216.5**	**40.0**	**31.2**	**43.3**	**54.8**	-	-	-	-	-
[Sim]+[NBF]+[Comp]	293.3	37.6	29.8	41.1	52.5	-	-	-	-	-

Table 6: Results of [Sim], [NBF], [RotE], [Comp] and [Sim] + [NBF] on the Reachable and Unreachable splits of WN18RR, FB15k-237, and CoDex-M datasets. Best individual model results are underlined.

Dataset	**Model**	**Reachable Split**				**Unreachable Split**			
		MR	MRR	H@1	H@10	MR	MRR	H@1	H@10
WN18RR	[Sim]	29.7	85.3	79.4	94.5	288.5	51.8	42.3	69.0
	[NBF]	4.7	89.7	86.8	95.7	1250.7	26.0	18.3	41.8
	[RotE]	102.9	85.6	83.3	90.0	8404.8	17.5	12.6	1.1
	[Comp]	285.9	85.6	83.8	88.5	10526.6	16.2	12.1	23.7
	[Sim]+[NBF]	**2.7**	**93.9**	**91.7**	**97.4**	**99.4**	**56.8**	**47.0**	**76.4**
FB15k−237	[Sim]	131.8	31.5	22.6	49.6	153.8	30.0	21.2	48.2
	[NBF]	86.9	44.8	35.2	64.0	180.4	28.2	19.3	46.2
	[RotE]	131.8	35.6	25.5	56.3	303.2	28.1	19.8	44.5
	[Comp]	129.9	37.9	28.0	57.8	323.9	29.7	21.5	46.0
	[Sim]+[NBF]	**76.0**	**46.5**	**36.8**	**65.3**	**137.0**	**32.3**	**23.1**	**50.6**
CoDex−M	[Sim]	166.5	35.5	26.8	52.4	363.6	23.7	15.8	39.6
	[NBF]	150.1	47.8	39.5	63.2	458.1	27.2	18.9	43.5
	[RotE]	290.5	44.2	37.2	56.8	639.1	26.7	19.5	40.5
	[Comp]	187.1	46.5	38.8	60.4	519.0	28.2	20.8	42.6
	[Sim]+[NBF]	**112.8**	**51.2**	**40.5**	**66.1**	**339.6**	**31.4**	**23.0**	**47.6**

A Data Statistics and Evaluation Metrics

Table 4 outlines the statistics of the datasets utilized in our experimental section. We utilize the standard train, validation and test splits for all datasets.

Metrics: For each triplet $(\mathtt{h}, \mathtt{r}, \mathtt{t})$ in the KG, typically queries of the form $(\mathtt{h}, \mathtt{r}, ?)$ and $(?, \mathtt{r}, \mathtt{t})$ are created for evaluation, with corresponding answers $\mathtt{t}$ and $\mathtt{h}$. We represent the $(?, \mathtt{r}, \mathtt{t})$ query as $(\mathtt{t}, \mathtt{r}^{-1}, ?)$ with the same answer $\mathtt{h}$, where $\mathtt{r}^{-1}$ is the inverse relation for $\mathtt{r}$, for both training and testing. Given ranks for all queries, we report the Mean Reciprocal Rank (MRR) and Hit@k (H@k, k = 1, 10) under the filtered setting in the main paper and two additional metrics: Mean Rank (MR) and Hits@3 in the appendices.

B Detailed Results on Proposed Ensemble

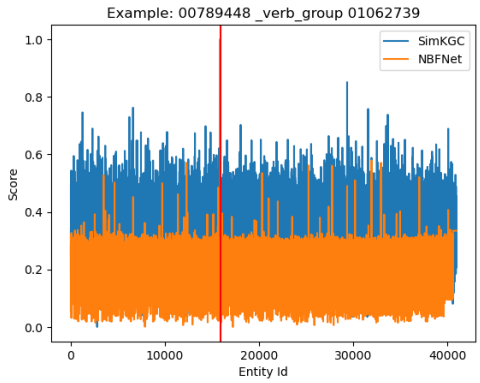
Here we present our experimental setup for the main results presented in Table 1. Since loading both `NBFNet` and the two BERT encoders from `SimKGC` into GPU at the same time is too taxing for our hardware, we dump the embeddings of all possible $(\mathtt{h}, \mathtt{r})$ and $\mathtt{t}$ from `SimKGC` to disk, and use them for training our ensemble. `SimKGC` is reliant on textual descriptions for performance. The original authors provide descriptions for WN18RR and FB15k-237, while descriptions for CoDex-M are available as part of the dataset. Since YAGO3-10 does not contain any descriptions, we treat the entity names as their descriptions. `SimKGC` also has a structural re-ranking step independent of its biencoder architecture, which we do not utilize as we expect our ensembling method to subsume it.

Next, we present results in Table 5 that are supplementary to results already presented in Table 1. In addition to MRR, Hits@1 and Hits@10 considered in Table 1, we also present numbers for Mean Rank (MR) and Hits@3 in Table 5. As before, the '+' sign represents our ensemble approach. We also consider an additional KG embedding model ComplEx (Trouillon et al., 2016) ([`Comp`] in tables) in this section and present complete results for it.

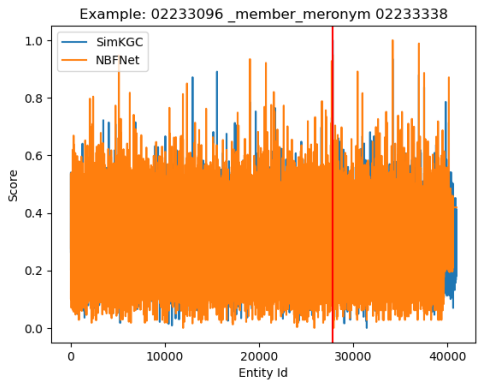
We observe that for the two new metrics considered in Table 5, we also obtain substantial performance gains on ensembling, notably a gain of 5.2 pt Hits@3 and 67.4% MR on the WN18RR dataset with [`Sim`] + [`NBF`] over [`Sim`]. Further, we observe that [`Sim`]+[`Comp`] consistently outperforms both [`Sim`] and [`Comp`], (by up to 2.2 pt MRR for CoDex-M). We also present complete numbers for [`Sim`]+[`NBF`]+[`Comp`] and [`Sim`]+[`NBF`]+[`RotE`] here.

C Feature Selection for Ensemble Weight Learning

In this section, we justify our choice of features for learning ensemble weights. We focus on `NBF` and `Sim` for this purpose. We claim that after our normalization procedure, a model has lower mean and variance when it is confident about the validity of its top predictions. To highlight this, we present distribution of the normalized scores over all candidate entities for `NBF` and `Sim` for two queries in the WN18RR dataset: one from the reachable split and the other from the unreachable split. The query for Figure 1a lies in the reachable split while the query for Figure 1b lies in the unreachable split. The entity id of the gold answer is marked with a red vertical line in both cases.



(a) Query in reachable split



(b) Query in unreachable split

Figure 1: Score distributions of [`NBF`] and [`Sim`] for two queries in WN18RR

We notice that for the query in the reachable split, [`NBF`] is very confident about its top prediction. Therefore, it scores the gold answer significantly higher than the other candidates. Upon normalization, this causes the other entities to have

comparatively smaller values (mostly in the range [0-0.4]), with a tighter spread. In comparison, for the query in the unreachable split, [NBF] cannot predict the gold answer confidently. This results in a much larger spread of scores across entities, with a lot of extreme values close to 1 indicating that the model is unable to conclusively determine which entity is the correct one. We choose the mean and variance as features because they will be able to distinguish between these two distributions, with their values being substantially smaller in the first case where [NBF] is confident about the predictions.

[Sim] also has these properties, albeit to a lesser degree. This is because SimKGC cannot exploit the KG structure, and therefore has to draw conclusions based on textual descriptions, which can point to several candidate answers of seemingly comparable validity. This results in the score distributions having a higher spread and a lower margin between the score of the top prediction and the other candidates. Therefore, the relative values of these mean and variance features can also inform the MLPs about the relative confidence of the models about their output, allowing them to compute ensemble weights for corresponding models as necessary.

As validation, we present the average of the mean and variance features from [NBF] over all test queries in the reachable and unreachable split for the WN18RR, FB15k-237 and CoDex-M datasets in Table 7.

Table 7: Average of[NBF] Features across Splits

Dataset	Reachable Split		Unreachable Split	
	Mean	Var	Mean	Var
WN18RR	0.277	0.008	0.353	0.127
FB15k-237	0.244	0.017	0.284	0.019
CoDex-M	0.492	0.015	0.566	0.017

We find that the average of the mean and variance features is up to 21% lower (for WN18RR) on the reachable split than the unreachable split, allowing the MLPs to distinguish between the splits based on score distribution statistics alone. We finally present results of experiments with other similar sets of features as input to the MLP in Table 8.

Table 8: Performance of [Sim] + [NBF] with different sets of input features to the MLP. + indicates concatenation here. Var and Std stand for variance and standard deviation. Zip stands for passing the entire output distribution from the base models as input to the MLP. Top 10 indicates using the top 10 scores from the output distribution as input features.

Dataset	Input Features	MRR	H@1	H@10
WN18RR	Mean + Var	**73.2**	**66.9**	**85.7**
	Mean + Std	73.1	66.8	85.1
	Std	67.1	59.2	80.5
	Mean	67.2	59.4	80.5
	Zip	66.6	58.7	80.3
	Top 10	72.1	65.5	84.8
CoDex-M	Mean + Var	**38.9**	**30.5**	**54.8**
	Mean + Std	38.1	29.9	54.1
	Std	32.5	23.4	48.5
	Mean	32.1	23.5	48.6
	Zip	31.6	23.3	48.1
	Top 10	31.7	23.3	48.4

We find that features that are created according to the reasoning above (Mean + Var and Mean + Std) perform better as compared to other features (Zip, Top 10) across datasets and metrics.

D Detailed Reachability Ablation

In this section we discuss further results of the experiment done to answer **Q1** in Section 5. The results presented in Table 6 are supplementary to the results presented in Table 2 where in addition to the MRR, Hits@1, Hits@10 metrics already presented in Table 2, we present results over one additional metric, MR. Additionally, we present the results on the 'reachable' and 'unreachable' split of CoDex-M, and for [RotE] and [Comp] on all datasets. We observe that [Sim] has up to 76% lower MR than [NBF] on the unreachable split while [NBF] has up to 83.3% lower MR than [Sim] on the reachable split over all the three datasets (both statistics mentioned are for WN18RR). The ensemble of [Sim]+[NBF] brings the MR down further, notably obtaining a gain of 42.5% on reachable split and 66% on unreachable split over best individual models for the WN18RR dataset. We also observe that [RotE] and [Comp] show similar variation of performance across splits when compared to [NBF], performing notably better on the reachable split as compared to the unreachable split across datasets. This indicates that these KG embedding models are also dependent on KG structure and paths between the head and gold tail to some extent for performance. We investigate this in more detail in Appendix E.

E RotatE as a Structure-Based Model

We claim that despite structure not being explicitly involved in the training of [RotE], it is still capable of capturing the structure of the KG to some extent in its relation embeddings by exploiting the compositionality inherent in its scoring function. Consider an example in which $(\mathtt{h}_1, \mathtt{r}_1, \mathtt{h}_2)$, $(\mathtt{h}_2, \mathtt{r}_2, \mathtt{h}_3)$ and $(\mathtt{h}_1, \mathtt{r}_3, \mathtt{h}_3)$ are all present in the KG. Let $\mathtt{T}_\mathtt{r}(\mathtt{h})$ be the vector obtained after rotating the embedding of $\mathtt{h}$ by the complex rotation defined by $\mathtt{r}$. During training, $\mathtt{T}_{\mathtt{r}_1}(\mathtt{h}_1)$ will be brought close to the embedding of $\mathtt{h}_2$ and $\mathtt{T}_{\mathtt{r}_2}(\mathtt{h}_2)$ will be brought close to the embedding of $\mathtt{h}_3$. As a result, $\mathtt{T}_{\mathtt{r}_2}(\mathtt{T}_{\mathtt{r}_1}(\mathtt{h}_1))$ will be brought close to $\mathtt{h}_3$. Upon training on $(\mathtt{h}_1, \mathtt{r}_3, \mathtt{h}_3)$, $\mathtt{T}_{\mathtt{r}_3}(\mathtt{h}_1))$ will also be brought close to $\mathtt{h}_3$. However, the relative positions of $\mathtt{h}_1$ and $\mathtt{h}_3$ on the complex plane already contain information about $\mathtt{T}_{\mathtt{r}_2} \circ \mathtt{T}_{\mathtt{r}_1}$, which is used while training $\mathtt{T}_{\mathtt{r}_3}$. As more such examples are seen over multiple epochs, $\mathtt{T}_{\mathtt{r}_3}$ will eventually be brought closer to the composed rotation $\mathtt{T}_{\mathtt{r}_2} \circ \mathtt{T}_{\mathtt{r}_1}$. Therefore, when query $(\mathtt{h}, \mathtt{r}_3, ?)$ is seen at test time, the model will be more likely to return candidates $\mathtt{t}$ which are connected to $\mathtt{h}$ through a path in the KG involving relations $\mathtt{r}_1$ and $\mathtt{r}_2$, making it structure dependent.

Of course, this phenomenon is not limited to paths of length 2, but can encode paths of longer length as well. We also expect only the most common paths to be captured through this mechanism, since multiple such paths have to be encoded by the same relation embedding. To validate these claims, we perform an experiment where we exhaustively mine the dataset for cases where $(\mathtt{h}_1, \mathtt{r}_1, \mathtt{h}_2)$ is present in the KG, alongside an entity $\mathtt{h}_3$ such that $(\mathtt{h}_1, \mathtt{r}_2, \mathtt{h}_3)$ and $(\mathtt{h}_3, \mathtt{r}_3, \mathtt{h}_2)$ are also present in the KG. This essentially considers all the cases where there is a path involving $\mathtt{r}_2$ and $\mathtt{r}_3$ that is closed by $\mathtt{r}_1$. We enumerate all such cases for each triple $(\mathtt{r}_1, \mathtt{r}_2, \mathtt{r}_3)$ and filter out those triples that have less than 20 occurrences in the KG. For each of the remaining triples, we take a random vector and transform it according to $\mathtt{T}_{\mathtt{r}_2} \circ \mathtt{T}_{\mathtt{r}_3}$. We then report the $\mathtt{r}$ such that transforming the same vector according to $\mathtt{T}_\mathtt{r}$ moves it closest to the result obtained on transforming it according to $\mathtt{T}_{\mathtt{r}_2} \circ \mathtt{T}_{\mathtt{r}_3}$. We expect $\mathtt{r}$ to be $\mathtt{r}_1$ for a majority of the triples based on our claims. We report accuracies obtained through this experiment for [RotE] on the WN18RR, FB15k-237 and CoDex-M datasets in Table 9.

Table 9: Structure Dependency of [RotE] and [Comp]

Dataset	Accuracy of Closest Relation [RotE]
WN18RR	38.1
FB15k-237	45.0
CoDex-M	68.2

We find that the accuracies are substantially better than the random baseline of $\frac{1}{\mathtt{NumberofRelations}}$ for all datasets (which is 9.1% for WN18RR, 0.4% for FB15k-237 and 1.4% for CoDex-M). [Sim] is not capable of capturing this notion, since it encodes $(\mathtt{h}, \mathtt{r})$ together using BERT, not as a composition of $\mathtt{h}$ and $\mathtt{r}$ embeddings. Therefore, we find that its behavior is independent of the split in which the query under consideration is present.

F Reachability Trends of Ensemble Weights

The aim of this section is to further discuss the results of the experiment done to answer **Q2** in Section 5. The results in Table 11 present the mean and standard deviation of ensemble weights $\mathtt{w}_2$ over the queries in the reachable and unreachable split for the WN18RR, CoDex-M and FB15k-237 datasets. The weight discussed in these tables is $\mathtt{w}_2$ in the ensemble defined as $\mathtt{w}_1[\mathtt{Sim}] + \mathtt{w}_2[\mathtt{NBF}]$ (with $\mathtt{w}_1 = 1$) according to Section 3. We observe that across all datasets, the average weight for reachable split is higher than the weight of unreachable split (up to 17% higher for WN18RR), thus reinforcing the fact that our approach gives more weightage to [NBF] on the reachable split across datasets. The standard deviation of $\mathtt{w}_2$ is also non-trivial on all splits of all datasets, showing that our approach is capable of adjusting it as required by individual queries.

Table 11: Mean and Standard Deviation (Std Dev in Table) of Ensemble Weights for [Sim] + [NBF]

Dataset	Reachable Split		Unreachable Split	
	Mean	Std Dev	Mean	Std Dev
WN18RR	0.61	0.04	0.52	0.07
CoDex-M	2.03	0.24	1.91	0.38
FB15k-237	2.64	0.22	2.57	0.24

To investigate why our ensemble weights are not binary and are quite consistent with each other for each split, we contrast [Sim] + [NBF] with a model that selects NBFNet on the reachable split and SimKGC on the unreachable split: $\mathtt{Split} - \mathtt{Select}$. We present results in Table 12.

Table 10: Results of paired student's t-test for dynamic ensemble and static ensemble on MRR with [Sim] + [NBF].

Dataset	Method	Split 1	Split 2	Split 3	Split 4	Split 5
WN18RR	Dynamic Ensemble	73.5	73.2	73.2	73.7	73.2
	Static Ensemble	72.0	72.5	71.9	72.3	71.9
	Difference	1.5	0.7	1.3	1.4	1.3
CoDex-M	Dynamic Ensemble	38.7	38.9	38.8	39.1	38.7
	Static Ensemble	37.1	37.8	38.0	37.8	38.0
	Difference	1.6	1.1	0.8	1.3	0.7

Table 12: Comparison of [Sim] + [NBF] and Split − Select on the WN18RR dataset.

Dataset	Approach	MRR	H@1	H@10
WN18RR	[Sim] + [NBF]	**73.2**	**66.9**	**85.7**
	Split − Select	68.4	61.8	81.0

We find that dynamic ensembling performs better than the oracle by 4.8 pt MRR. This is because structure-based models tend to rank more connected tails higher, while text-based models rank tails based solely on their textual descriptions. Therefore, a soft ensemble can take advantage of both structural and textual information to perform better than a mixture-of-experts model that simply selects one of the base models based on expected performance trends.

G Significance of Improvements with Dynamic Ensembling

We first perform a paired student's t-test on the MRR over a 5-fold split for [Sim] + [NBF] to confirm that the gains obtained by our approach over static ensembling are statistically significant. We present the results in Table 10.

We obtain a t-value of 8.9 for WN18RR and 6.7 for CoDex-M. With a p-value of 0.05, the reference value is 2.78. Therefore, the gains obtained by our model over static ensembling are indeed statistically significant.

The performance of an ensemble is ultimately dependent on the performance of the individual models. To obtain an estimate of the best possible performance that can be obtained from model fusion, we present results in Table 13 with [Sim] + [NBF] for a model that selects the most performant model for each query (BEST).

Table 13: Comparison of [Sim] + [NBF] and BEST on the WN18RR and CoDex-M datasets.

Dataset	Approach	MRR	H@1	H@10
WN18RR	[Sim] + [NBF]	73.2	66.9	85.7
	BEST	**74.1**	**67.6**	**86.1**
CoDex-M	[Sim] + [NBF]	38.9	30.5	54.8
	BEST	**41.2**	**32.6**	**57.9**

We find that the results for our dynamic ensemble are only up to 2.3 MRR pts behind a theoretical oracle that always knows the best model for each query, indicating that most of the potential for improvement through late fusion techniques has been obtained through dynamic ensembling.