Interstellar: Searching Recurrent Architecture for Knowledge Graph Embedding

Yongqi Zhang[1,3] **Quanming Yao**[1,2] **Lei Chen**[3]
[1]4Paradigm Inc.
[2]Department of Electronic Engineering, Tsinghua University
[3]Department of Computer Science and Engineering, HKUST
{zhangyongqi,yaoquanming}@4paradigm.com, leichen@cse.ust.hk

Abstract

Knowledge graph (KG) embedding is well-known in learning representations of KGs. Many models have been proposed to learn the interactions between entities and relations of the triplets. However, long-term information among multiple triplets is also important to KG. In this work, based on the relational paths, which are composed of a sequence of triplets, we define the Interstellar as a recurrent neural architecture search problem for the short-term and long-term information along the paths. First, we analyze the difficulty of using a unified model to work as the Interstellar. Then, we propose to search for recurrent architecture as the Interstellar for different KG tasks. A case study on synthetic data illustrates the importance of the defined search problem. Experiments on real datasets demonstrate the effectiveness of the searched models and the efficiency of the proposed hybrid-search algorithm. [1]

1 Introduction

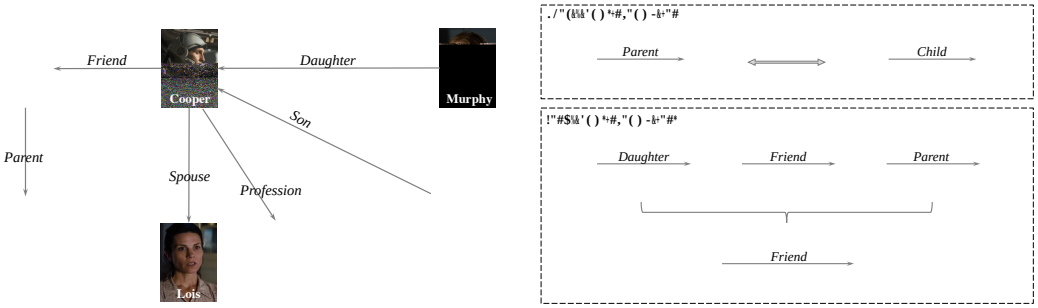
Knowledge Graph (KG) [4, 43, 52] is a special kind of graph with many relational facts. It has inspired many knowledge-driven applications, such as question answering [31, 38], medical diagnosis [60], and recommendation [29]. An example of the KG is in Figure 1(a). Each relational fact in KG is represented as a triplet in the form of *(subject entity, relation, object entity)*, abbreviated as $(s; r; o)$. To learn from the KGs and benefit the downstream tasks, embedding based methods, which learn low-dimensional vector representations of the entities and relations, have recently developed as a promising direction to serve this purpose [8, 19, 45, 52].

Many efforts have been made on modeling the plausibility of triplets $(s; r; o)$s through learning embeddings. Representative works are triplet-based models, such as TransE [8], ComplEx [49], ConvE [14], RotatE [44], AutoSF [61], which define different embedding spaces and learn on single triplet $(s; r; o)$. Even though these models perform well in capturing short-term semantic information inside the triplets in KG, they still cannot capture the information among multiple triplets.

In order to better capture the complex information in KGs, the relational path is introduced as a promising format to learn composition of relations [20, 28, 39] and long-term dependency of triplets [27, 12, 19, 51]. As in Figure 1(b), a relational path is defined as a set of L triplets $(s_1; r_1; o_1); (s_2; r_2; o_2); \cdots; (s_L; r_L; o_L)$, which are connected head-to-tail in sequence, i.e. $o_i = s_{i+1}; \forall i = 1 \cdots L - 1$. The paths not only preserve every single triplet but also can capture the dependency among a sequence of triplets. Based on the relational paths, the triplet-based models can be compatible by working on each triplet $(s_i; r_i; o_i)$ separately. TransE-Comp [20] and PTransE [28] learn the composition relations on the relational paths. To capture the long-term information

[1]Code is available at `https://github.com/AutoML-4Paradigm/Interstellar`, and correspondence is to Q. Yao.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.



(a) An example of Knowledge Graph. (b) Examples of relational paths.

Figure 1: Short-term information is represented by a single triplet. Long-term information passes across multiple triplets. The two kinds of information in KGs can be preserved in the relational path.

in KGs, Chains [12] and RSN [19] design customized RNN to leverage all the entities and relations along path. However, the RNN models still overlook the semantics inside each triplet [19]. Another type of models leverage Graph Convolution Network (GCN) [25] to extract structural information in KGs, e.g. R-GCN, GCN-Align [53], CompGCN [50]. However, GCN-based methods do not scale well since the entire KG needs to be processed and it has large sample complexity [16].

In this paper, we observe that the relational path is an important and effective data structure that can preserve both short-term and long-term information in KG. Since the semantic patterns and the graph structures in KGs are diverse [52], how to leverage the short-term and long-term information for a specific KG task is non-trivial. Inspired by the success of neural architecture search (NAS) [15], we propose to search recurrent architectures as the *Interstellar* to learn from the relational path. The contributions of our work are summarized as follows:

1. We analyze the difficulty and importance of using the relational path to learn the short-term and long-term information in KGs. Based on the analysis, we define the Interstellar as a recurrent network to process the information along the relational path.
2. We formulate the above problem as a NAS problem and propose a domain-specific search space. Different from searching RNN cells, the recurrent network in our space is specifically designed for KG tasks and covers many human-designed embedding models.
3. We identify the problems of adopting stand-alone and one-shot search algorithms for our search space. This motivates us to design a hybrid-search algorithm to search efficiently.
4. We use a case study on the synthetic data set to show the reasonableness of our search space. Empirical experiments on entity alignment and link prediction tasks demonstrate the effectiveness of the searched models and the efficiency of the search algorithm.

Notations. We denote vectors by lowercase boldface, and matrix by uppercase boldface. A KG $\mathcal{G} = (\mathcal{E}; \mathcal{R}; \mathcal{S})$ is defined by the set of entities $\mathcal{E}$, relations $\mathcal{R}$ and triplets $\mathcal{S}$. A triplet $(s; r; o) \in \mathcal{S}$ represents a relation r that links from the subject entity s to the object entity o. The embeddings in this paper are denoted as boldface letters of indexes, e.g. $\mathbf{s}; \mathbf{r}; \mathbf{o}$ are embeddings of $s; r; o$. "$\odot$" is the element-wise multiply and "$\otimes$" is the Hermitian product [49] in complex space.

2 Related Works

2.1 Representation Learning in Knowledge Graph (KG)

Given a single triplet $(s; r; o)$, TransE [8] models the relation r as a translation vector from subject entity s to object entity o, i.e., the embeddings satisfy $\mathbf{s} + \mathbf{r} \approx \mathbf{o}$. The following works DistMult [55], ComplEx [49], ConvE [14], RotatE [44], etc., interpret the interactions among embeddings $\mathbf{s}; \mathbf{r}$ and $\mathbf{o}$ in different ways. All of them learn embeddings based on single triplet.

In KGs, a relational path is a sequence of triplets. PTransE [28] and TransE-Comp [20] propose to learn the composition of relations $(r_1; r_2; \ldots; r_n)$. In order to combine more pieces of information in KG, Chains [12] and RSN [19] are proposed to jointly learn the entities and relations along the relational path. With different connections and combinators, these models process short-term and long-term information in different ways.

This verifies our analysis that it is difficult to use a unified model that can adapt to the short-term and long term information for different KG tasks.

4.3 Comparison with State-of-the-art KG Embedding Methods

Entity Alignment. The entity alignment task aims to align entities in different KGs referring the same instance. In this task, long-term information is important since we need to propagate the alignment across triplets [19, 45, 62]. We use four cross-lingual and cross-database subset from DBpedia and Wikidata generated by [19], i.e. DBP-WD, DBP-YG, EN-FR, EN-DE. For fair comparison, we follow the same path sampling scheme and the data set splits in [19].

Table 4: Performance comparison on entity alignment task. H@k is short for Hit@k. The results of TransD [22], BootEA [45], IPTransE [62], GCN-Align [53] and RSN [19] are copied from [19].

models		DBP-WD			DBP-YG			EN-FR			EN-DE		
		H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR
triplet	TransE	18.5	42.1	0.27	9.2	24.8	0.15	16.2	39.0	0.24	20.7	44.7	0.29
	TransD*	27.7	57.2	0.37	17.3	41.6	0.26	21.1	47.9	0.30	24.4	50.0	0.33
	BootEA*	32.3	63.1	0.42	31.3	62.5	0.42	31.3	62.9	0.42	44.2	70.1	0.53
GCN	GCN-Align	17.7	37.8	0.25	19.3	41.5	0.27	15.5	34.5	0.22	25.3	46.4	0.22
	VR-GCN	19.4	55.5	0.32	20.9	55.7	0.32	16.0	50.8	0.27	24.4	61.2	0.36
	R-GCN	8.6	31.4	0.16	13.3	42.4	0.23	7.3	31.2	0.15	18.4	44.8	0.27
path	PTransE	16.7	40.2	0.25	7.4	14.7	0.10	7.3	19.7	0.12	27.0	51.8	0.35
	IPTransE*	23.1	51.7	0.33	22.7	50.0	0.32	25.5	55.7	0.36	31.3	59.2	0.41
	Chains	32.2	60.0	0.42	35.3	64.0	0.45	31.4	60.1	0.41	41.3	68.9	0.51
	RSN*	38.8	65.7	0.49	40.0	67.5	0.50	34.7	63.1	0.44	48.7	72.0	0.57
	Interstellar	**40.7**	**71.2**	**0.51**	**40.2**	**72.0**	**0.51**	**35.5**	**67.9**	**0.46**	**50.1**	**75.6**	**0.59**

Table 4 compares the testing performance of the models searched by Interstellar and human-designed ones on the *Normal* version datasets [19] (the *Dense* version [19] in Appendix B.1). In general, the path-based models are better than the GCN-based and triplets-based models by modeling long-term dependencies. BootEA [45] and IPTransE [62] win over TransE [8] and PTransE [28] respectively by iteratively aligning discovered entity pairs. Chains [12] and RSN [19] outperform graph-based models and the other path-based models by explicitly processing both entities and relations along path. In comparison, Interstellar is able to search and balance the short-term and long-term information adaptively, thus gains the best performance. We plot the learning curve on DBP-WD of some triplet, graph and path-based models in Figure 4(a) to verify the effectiveness of the relational paths.

Link Prediction. In this task, an incomplete KG is given and the target is to predict the missing entities in unknown links [52]. We use three famous benchmark datasets, WN18-RR [14] and FB15k-237 [48], which are more realistic than their superset WN18 and FB15k [8], and YAGO3-10 [32], a much larger dataset.

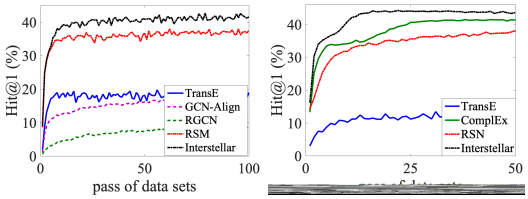
Table 5: Link prediction results.

models	WN18-RR			FB15k-237			YAGO3-10		
	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR
TransE	12.5	44.5	0.18	17.3	37.9	0.24	10.3	27.9	0.16
ComplEx	41.4	49.0	0.44	22.7	49.5	0.31	40.5	62.8	0.48
RotatE*	43.6	54.2	0.47	**23.3**	50.4	**0.32**	40.2	63.1	0.48
R-GCN	-	-	-	15.1	41.7	0.24	-	-	-
PTransE	27.2	46.4	0.34	20.3	45.1	0.29	12.2	32.3	0.19
RSN	38.0	44.8	0.40	19.2	41.8	0.27	16.4	37.3	0.24
Interstellar	**44.0**	**54.8**	**0.48**	**23.3**	**50.8**	**0.32**	**42.4**	**66.4**	**0.51**

We search architectures with dimension 64 to save time and compare the models with dimension 256. Results of R-GCN on WN18-RR and YAGO3-10 are not available due to out-of-memory issue. As shown in Table 5, PTransE outperforms TransE by modeling compositional relations, but worse than ComplEx and RotatE since the adding operation is inferior to $\otimes$ when modeling the interaction between entities and relations [49]. RSN is worse than ComplEx/RotatE since it pays more attention to long-term information rather than the inside semantics. Interstellar outperforms the path-based methods PTransE and RSN by searching architectures that model fewer steps (see Appendix C.4). And it works comparable with the triplet-based models, i.e. ComplEx and RotatE, which are specially designed for this task. We show the learning curve of TransE, ComplEx, RSN and Interstellar on WN18-RR in Figure 4(b).

4.4 Comparison with Existing NAS Algorithms

In this part, we compare the proposed Hybrid-search algorithm in Interstellar with the other NAS methods. First, we compare the proposed algorithm with stand-alone NAS methods. Once an architecture is sampled, the parameters are initialized and trained into converge to give reliable feedback. Random search *(Random)*, Reinforcement learning *(Reinforce)* [63] and Bayes optimization *(Bayes)* [7] are chosen as the baseline algorithms. As shown in Figure 5 (entity alignment on DBP-WD and link prediction on WN18-RR), the Hybrid algorithm in Interstellar is more efficient since it takes advantage of the one-shot approach to search the micro architectures in $\mathcal{A}_2$.



(a) Entity alignment. (b) Link prediction.

Figure 4: Single model learning curve.

(a) Entity alignment. (b) Link prediction.

Figure 5: Compare with NAS methods.

Then, we compare with one-shot NAS methods with PS on the entire space. DARTS [30] and ASNG [2] are chosen as the baseline models. For DARTS, the gradient is obtained from loss on training set since validation metric is not differentiable. We show the performance of the best architecture found by the two one-shot algorithms. As shown in the dashed lines, the architectures found by one-shot approach are much worse than that by the stand-alone approaches. The reason is that PS is not reliable in our complex recurrent space (more experiments in Appendix B.2). In comparison, Interstellar is able to search reliably and efficiently by taking advantage of both the stand-alone approach and the one-shot approach.

4.5 Searching Time Analysis

We show the clock time of Interstellar on entity alignment and link prediction tasks in Table 6. Since the datasets used in entity alignment task have similar scales (see Appendix A.4), we show them in the same column. For each task/dataset, we show the computation cost of the macro-level and micro-level in Interstellar for 50 iterations between step 2-5 in Algorithm 1 (20 for YAGO3-10 dataset); and the fine-tuning procedure after searching for 50 groups of hyper-parameters, i.e. learning rate, decay rate, dropout rate, L2 penalty and batch-size (details in Appendix A.5). As shown, the entity alignment tasks take about 15-25 hours, while link prediction tasks need about one or more days due to the larger data size. The cost of the search process is at the same scale with that of the fine-tuning time, which shows the search process is not expensive.

Table 6: Comparison of searching and fine-tuning time (in hours) in Algorithm 1.

procedure		entity alignment		link prediction		
		Normal	Dense	WN18-RR	FB15k-237	YAGO3-10
search	macro-level (line 2-3)	9.9±1.5	14.9±0.3	11.7±1.9	23.2±3.4	91.6±8.7
	micro-level (line 4-5)	4.2±0.2	7.5±0.6	6.3± 0.9	5.6±0.4	10.4±1.3
fine-tune (line 7)		11.6±1.6	16.2±2.1	44.3±2.3	67.6±4.5	> 200

5 Conclusion

In this paper, we propose a new NAS method, Interstellar, to search RNN for learning from the relational paths, which contain short-term and long-term information in KGs. By designing a specific search space based on the important properties in relational path, Interstellar can adaptively search promising architectures for different KG tasks. Furthermore, we propose a hybrid-search algorithm that is more efficient compared with the other the state-of-art NAS algorithms. The experimental results verifies the effectiveness and efficiency of Interstellar on various KG embedding benchmarks. In future work, we can combine Interstellar with AutoSF [61] to give further improvement on the embedding learning problems. Taking advantage of data similarity to improve the search efficiency on new datasets is another extension direction.

Broader impact

Most of the attention on KG embedding learning has been focused on the triplet-based models. In this work, we emphasis the benefits and importance of using relational paths to learn from KGs. And we propose the path-interstellar as a recurrent neural architecture search problem. This is the first work applying neural architecture search (NAS) methods on KG tasks.

In order to search efficiently, we propose a novel hybrid-search algorithm. This algorithm addresses the limitations of stand-alone and one-shot search methods. More importantly, the hybrid-search algorithm is not specific to the problem here. It is also possible to be applied to the other domains with more complex search space [63, 47, 42].

One limitation of this work is that, the Interstellar is currently limited on the KG embedding tasks. Extending to reasoning tasks like DRUM [39] is an interesting direction.

Acknowledgment

This work is partially supported by the Hong Kong RGC GRF Project 16202218, CRF Project C6030-18G, C1031-18G, C5026-18G, AOE Project AoE/E-603/18, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX. Lei Chen is partially supported by Microsoft Research Asia Collaborative Research Grant, Didi-HKUST joint research lab project, and Wechat and Webank Research Grants.

References

[1]

[2] Y. Akimoto, S. Shirakawa, N. Yoshinari, K. Uchida, S. Saito, and K. Nishida. Adaptive stochastic natural gradient method for one-shot neural architecture search. In *ICML*, pages 171–180, 2019.

[3] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In *Semantic Web*, pages 722–735, 2007.

[5] B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcement learning. In *ICLR*, 2017.

[6] G. Bender, P. Kindermans, B. Zoph, V. Vasudevan, and Q. Le. Understanding and simplifying one-shot architecture search. In *ICML*, pages 549–558, 2018.

[7] J. S Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *NeurIPS*, pages 2546–2554, 2011.

[8] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NeurIPS*, pages 2787–2795, 2013.

[9] G. Bouchard, S. Singh, and T. Trouillon. On approximate reasoning capabilities of low-rank vector spaces. In *AAAI Spring Symposium Series*, 2015.

[10] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Gated feedback recurrent neural networks. In *ICML*, pages 2067–2075, 2015.

[11] A. Conn, K. Scheinberg, and L. Vicente. *Introduction to derivative-free optimization*, volume 8. Siam, 2009.

[12] R. Das, A. Neelakantan, D. Belanger, and A. McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *ACL*, pages 132–141, 2017.

[13] B. Dasgupta and E. Sontag. Sample complexity for learning recurrent perceptron mappings. In *NIPS*, pages 204–210, 1996.

[14] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel. Convolutional 2D knowledge graph embeddings. In *AAAI*, 2017.

[15] T. Elsken, Jan H. Metzen, and F. Hutter. Neural architecture search: A survey. *JMLR*, 20(55):1–21, 2019.

[16] V. Garg, S. Jegelka, and T. Jaakkola. Generalization and representational limits of graph neural networks. In *ICML*, 2020.

[17] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *TPAMI*, (6):721–741, 1984.

[18] A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *SIGKDD*, pages 855–864. ACM, 2016.

[19] L. Guo, Z. Sun, and W. Hu. Learning to exploit long-term relational dependencies in knowledge graphs. In *ICML*, 2019.

[20] K. Guu, J. Miller, and P. Liang. Traversing knowledge graphs in vector space. In *EMNLP*, pages 318–327, 2015.

[21] F. Hutter, L. Kotthoff, and J. Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2018.

[22] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao. Knowledge graph embedding via dynamic mapping matrix. In *ACL*, volume 1, pages 687–696, 2015.

[23] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. Xing. Neural architecture search with bayesian optimisation and optimal transport. In *NeurIPS*, pages 2016–2025, 2018.

[24] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

[25] T. N Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2016.

[26] T. Lacroix, N. Usunier, and G. Obozinski. Canonical tensor decomposition for knowledge base completion. In *ICML*, 2018.

[27] N. Lao, T. Mitchell, and W. Cohen. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, pages 529–539, 2011.

[28] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu. Modeling relation paths for representation learning of knowledge bases. Technical report, arXiv:1506.00379, 2015.

[29] C. Liu, L. Li, X. Yao, and L. Tang. A survey of recommendation algorithms based on knowledge graph embedding. In *CSEI*, pages 168–171, 2019.

[30] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019.

[31] D. Lukovnikov, A. Fischer, J. Lehmann, and S. Auer. Neural network-based question answering over knowledge graphs on word and character level. In *WWW*, pages 1211–1220, 2017.

[32] F. Mahdisoltani, J. Biega, and F. M Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*, 2013.

[33] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur.

[34] Y. Ollivier, L. Arnold, A. Auger, and N. Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. *JMLR*, 18(1):564–628, 2017.

[35] R. Pascanu and Y. Bengio. Revisiting natural gradient for deep networks. Technical report, arXiv:1301.3584, 2013.

[36] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *ICLR*, 2017.

[37] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameters sharing. In *ICML*, pages 4095–4104, 2018.

[38] H. Ren, W. Hu, and J. Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *ICLR*, 2020.

[39] A. Sadeghian, M. Armandpour, P. Ding, and D. Wang. DRUM: End-to-end differentiable rule mining on knowledge graphs. In *NeurIPS*, pages 15321–15331, 2019.

[40] M. Schlichtkrull, T. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *ESWC*, pages 593–607, 2018.

[41] C. Sciuto, K. Yu, M. Jaggi, C. Musat, and M. Salzmann. Evaluating the search phase of neural architecture search. In *ICLR*, 2020.

[42] D. So, Q. Le, and C. Liang. The evolved transformer. In *ICML*, pages 5877–5886, 2019.

[43] R. Socher, D. Chen, C. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *NeurIPS*, pages 926–934, 2013.

[44] Z. Sun, Z. Deng, J. Nie, and J. Tang. RotatE: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2019.

[45] Z. Sun, W. Hu, Q. Zhang, and Y. Qu. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*, pages 4396–4402, 2018.

[46] M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *CISCA*, 2012.

[47] M. Tan and Q. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *ICML*, pages 6105–6114, 2019.

[48] K. Toutanova and D. Chen. Observed versus latent features for knowledge base and text inference. In *Workshop on CVSMC*, pages 57–66, 2015.

[49] T. Trouillon, C. Dance, E. Gaussier, J. Welbl, S. Riedel, and G. Bouchard. Knowledge graph completion via complex tensor factorization. *JMLR*, 18(1):4735–4772, 2017.

[50] S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar. Composition-based multi-relational graph convolutional networks. *ICLR*, 2020.

[51] H. Wang, H. Ren, and J. Leskovec. Entity context and relational paths for knowledge graph completion. Technical report, arXiv:1810.13306, 2020.

[52] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *TKDE*, 29(12):2724–2743, 2017.

[53] Z. Wang, Q. Lv, X. Lan, and Y. Zhang. Cross-lingual knowledge graph alignment via graph convolutional networks. In *EMNLP*, pages 349–357, 2018.

[54] S. Xie, H. Zheng, C. Liu, and L. Lin. SNAS: stochastic neural architecture search. In *ICLR*, 2019.

[55] B. Yang, W. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*, 2015.

[56] Q. Yao and M. Wang. Taking human out of learning applications: A survey on automated machine learning. Technical report, arXiv:1810.13306, 2018.

[57] Q. Yao, J. Xu, W. Tu, and Z. Zhu. Efficient neural architecture search via proximal iterations. In *AAAI*, 2020.

[58] R. Ye, X. Li, Y. Fang, H. Zang, and M. Wang. A vectorized relational graph convolutional network for multi-relational network alignment. In *IJCAI*, pages 4135–4141, 2019.

[59] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020.

[60] C. Zhang, Y. Li, N. Du, W. Fan, and P. Yu. On the generative discovery of structured medical knowledge. In *KDD*, pages 2720–2728, 2018.

[61] Y. Zhang, Q. Yao, W. Dai, and L. Chen. AutoSF: Searching scoring functions for knowledge graph embedding. In *ICDE*, pages 433–444, 2020.

[62] H. Zhu, R. Xie, Z. Liu, and M. Sun. Iterative entity alignment via joint knowledge embeddings. In *IJCAI*, pages 4258–4264, 2017.

[63] B. Zoph and Q. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

A Supplementary Details

A.1 Details of the search space

Operators Given two input vectors a and b with d-dimensions and d is an even number. Two basic combinators are (a). adding ($+$): $\mathsf{o}_i = \mathsf{a}_i + \mathsf{b}_i$; and (b).multiplying ($\odot$): $\mathsf{o}_i = \mathsf{a}_i \cdot \boldsymbol{b}_i$. In order to cover ComplEx [49] in the search space, we use Hermitian product ($\otimes$) as in [49]

$$\mathsf{o}_i = \begin{array}{ll} \mathsf{a}_i \cdot \mathsf{b}_i - \mathsf{a}_{i+d/2} \cdot \mathsf{b}_{i+d/2} & \text{if } i < d/2 \\ \mathsf{a}_{i-d/2} \cdot \mathsf{b}_i - \mathsf{a}_i \cdot \mathsf{b}_{i-d/2} & \text{otherwise} \end{array}$$

As for the gated unit, we define it as $\mathsf{o}_i = \mathsf{g}_i \cdot \mathsf{a}_i + (1 - \mathsf{g}_i) \cdot \mathsf{b}_i$, where the gate function $\mathsf{g} = \text{sigmoid}\,(\mathsf{W}_a\mathsf{a} + \mathsf{W}_b\mathsf{b})$ with trainable parameters W_a; $\mathsf{W}_b \in \mathbb{R}^{d \times d}$, to imitate the gated recurrent function.

Space size Based on the details in Table 2, the inputs of O_r and O_v are chosen from h_{t-1}; O_s; $\mathsf{0}$; s_t. Then the three operators O_s; O_r; O_v select one combinator from $+$; $\odot$; $\otimes$ and gated. Thus, we have $4^2 \times 4^3 = 1024$ architectures in the macro space $\mathcal{A}_1$. For the micro space, we only use activation functions after O_s; O_r since we empirically observe that activation function on O_v is bad for the embedding spaces. In this way, the size of micro space $\mathcal{A}_2$ is $3^2 \times 2^6 = 576$. Totally, there should be 6×10^5 candidates with the combination of macro-level space and micro-level space.

A.2 Details of the search algorithm

Compare with existing NAS problem. In Section 3.2, we have already talked about the problems of existing search algorithms and propose a hybrid-search algorithm that can search fast and accurate. Here, we give an overview of the different NAS methods in Table 7.

Table 7: Comparison of state-of-the-art NAS methods for general RNN with the proposed Interstellar.

	Existing NAS Algorithms		Interstellar
	Stand-alone approach	One-shot approach	
space	complex structures	micro cells	KG-specific
algorithm	reinforcement learning [63], Bayes optimization [7, 23]	direct gradient descent [30], stochastic relaxation [2, 54]	natural policy gradient
evaluation	full model training	parameter-sharing	hybrid

Implementation of $\mathcal{M}(f(\boldsymbol{F}^*; \boldsymbol{\alpha}); \mathcal{G}_{\text{val}})$. For both tasks, the training samples are composed of relational paths. But the validation data format is different. For the entity alignment task, the validation data is a set of entity pairs, i.e. $\mathcal{S}_{val} = \{(e_1; e_2) | e_1 \in \mathcal{E}_1; e_2 \in \mathcal{E}_2\}$. The performance is measured by the cosine similarity of the embeddings e_1 and e_2. Thus, the architecture $\boldsymbol{\alpha}$ is not available in $\mathcal{M}(f(\boldsymbol{F}^*; \boldsymbol{\alpha}); \mathcal{G}_{\text{val}})$. Instead, we use the negative loss function on a training mini-batch $-\mathcal{L}(f(\boldsymbol{F}^*; \boldsymbol{\alpha}); \mathcal{B}_{\text{tra}})$ as an alternative of $\mathcal{M}(f(\boldsymbol{F}^*; \boldsymbol{\alpha}); \mathcal{G}_{\text{val}})$. For the link prediction task, the validation data is set of single triplets, i.e. 1-step paths. Different from the entity alignment task, the path is available for validation measurement here. Therefore, once we sample a mini-batch $\mathcal{B}_{\text{val}}$ from $\mathcal{G}_{\text{val}}$, we can either use the loss function $-\mathcal{L}(f(\boldsymbol{F}^*; \boldsymbol{\alpha}); \mathcal{B}_{\text{val}})$ or the direct evaluation metric (MRR or Hit@k) $\mathcal{M}(f(\boldsymbol{F}^*; \boldsymbol{\alpha}); \mathcal{B}_{\text{val}})$ on the mini-batch $\mathcal{B}_{\text{val}}$.

Implementation of natural policy gradient. Recall that the optimization problem of architectures is changed from optimizing $\boldsymbol{\alpha}$ by $\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha} \in \mathcal{A}} \mathcal{M}(f(\boldsymbol{F}^*; \boldsymbol{\alpha}); \mathcal{G}_{\text{val}})$ to optimizing the distribution parameter $\boldsymbol{\theta}$ by $\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\alpha} \sim p_{\boldsymbol{\theta}}(\boldsymbol{\alpha})} [\mathcal{M}(f(\boldsymbol{F}^*; \boldsymbol{\alpha}); \mathcal{G}_{\text{val}})]$: As discussed in Section 3.2.2, we use the natural policy gradient to solve $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} J(\ )$. To begin with, we need to define the distribution $p_{\boldsymbol{\theta}}$. Same as [2, 34], we refer to the exponential family $h(\boldsymbol{\alpha}) \cdot \exp\ (\boldsymbol{\theta})^\top T(\boldsymbol{\alpha}) - A(\boldsymbol{\theta})\ $, where $h(\boldsymbol{\alpha})$, $T(\boldsymbol{\alpha})$ and $A(\boldsymbol{\theta})$ are known functions depending on the target distribution. The benefit of using exponential family is that the inverse Fisher information matrix $\boldsymbol{H}^{-1}(\boldsymbol{\theta})$ is easily obtained. For simplicity, we set $h(\boldsymbol{\alpha}) = 1$ and choose the expectation

parameters $\boldsymbol{\theta} = \mathbb{E}_{p_{\boldsymbol{\theta}}}[T(\boldsymbol{\alpha})]$ as in [19]. Then the gradient reduces to $\nabla_{\boldsymbol{\theta}} \ln(p_{\boldsymbol{\theta}}(\boldsymbol{\alpha})) = T(\boldsymbol{\alpha}) - \boldsymbol{\theta}$, and the inverse Fisher information matrix $\boldsymbol{H}^{-1}(\boldsymbol{\theta}) = \mathbb{E}_{p_{\boldsymbol{\theta}}}[(T(\boldsymbol{\alpha}) - \ )(T(\boldsymbol{\alpha}) - \ )^\top]$ is computed with the finite approximation. The iteration is,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{1}{m} \sum_{i=1}^{m} \left[T(\boldsymbol{\alpha}^{(i)}) - \boldsymbol{\theta}_t \right] \left[T(\boldsymbol{\alpha}^{(i)}) - \boldsymbol{\theta}_t \right]^\top \cdot \mathcal{M} \left[f(\boldsymbol{F}^{(i)^*}; \boldsymbol{\alpha}^{(i)}); \mathcal{G}_{\text{val}} \right] ;$$

where $\boldsymbol{\alpha}^{(i)}$'s are sampled from $p_{\boldsymbol{\theta}_t}$. In this paper, we set the finite sample m as 2 to save time. Since the architecture parameters are categorical, we model $p_{\boldsymbol{\theta}}$ as categorical distributions. For a component with K categories, we use a K dimensional vector $\boldsymbol{\theta}$ to model the distribution $p_{\boldsymbol{\theta}}$. The probability of sampling the i-th category is $\boldsymbol{\theta}_i$ and $_K = 1 - \sum_{j=1}^{K-1} \boldsymbol{\theta}_j$. Note that, the NG is only used to update the first $K-1$ dimension of $\boldsymbol{\theta}$, and $T(\boldsymbol{\alpha}_i)$ is a one-hot vector (without dimension K) of the category $\boldsymbol{\alpha}_i$. For more details, please refer to Section 2.4 in [19] and the published code.

A.3 Training details

Following [18, 19], we sample paths from biased random walks. Take Figure 6 as an example and the path walked from e_0 to e_1 just now. In conventional random walk, all the neighbors of e_1 have the same probability to be sampled as the next step. In biased random walk, we sample the neighbor that can go deeper or go to another KG with larger probability. For single KG, we use one parameter $\in (0.5; 1)$ to give the depth bias. Specifically, the neighbors that are two steps away from the previous entity e_0, like e_2 and e_4 in Figure 6, have the bias of . And the bias of others, i.e. e_0; e_3, are $1 -$. Since > 0.5, the path is more likely to go deeper. Similarly, we have another parameter $\in (0.5; 1)$ for two KGs to give cross-KG bias. Assume e_0; e_1; and e_3 belongs to the same KG $\mathcal{G}_1$ and e_2; e_4 are parts of $\mathcal{G}_2$. Then, the next step is encouraged to jump to entities in another KG, namely to the entities e_2 and e_4 in Figure 6. Since the aligned pairs are usually rare in the training set, encouraging cross-KG walk can learn more about the aligned entities in the two KGs.

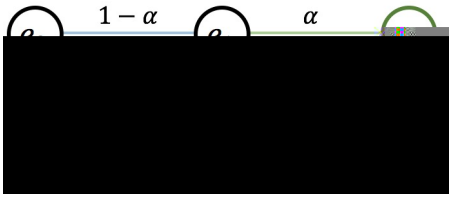


Figure 6: An example of the biased random walks.

Since the KGs are usually large, we sample two paths for each triplet in $\mathcal{S}_{tra}$. The length of paths is 7 for entity alignment task on two KGs and 3 for link prediction task on single KG. The paths are fixed for each dataset to keep a fair comparison among different models. The parameters used for sampling path are summarized in Table 8.

Table 8: Parameter used for sampling path.

parameters			length
entity alignment	0.9	0.9	7
link prediction	0.7	–	3

Once the paths are sampled, we start training based on the relational path. The loss for a path with length L, i.e. containing L triplets, is given in given as

$$\mathcal{L}_{tra} = \sum_{t=1}^{L} \left(-\mathbf{v}_t \cdot \mathbf{o}_t + \log \sum_{o_i \in \mathcal{E}} \exp\left(\mathbf{v}_t \cdot \mathbf{o}_i\right) \right) \tag{4}$$

In the recurrent step t, we focus on one triplet $(s_t; r_t; o_t)$. The subject entity embedding $\mathbf{s}_t$ and relation embedding $\mathbf{r}_t$ are processed along with the proceeding information $\mathbf{h}_{t-1}$ to get the output $\mathbf{v}_t$. The output $\mathbf{v}_t$ is encouraged to approach the object entity embedding $\mathbf{o}_t$. Thus, the objective in (4) can be regarded as a multi-class log-loss [26] where the object o_t is the true label.

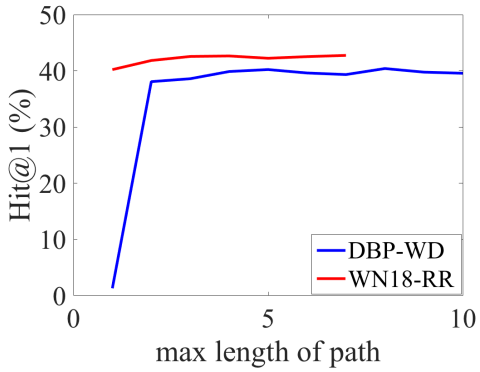
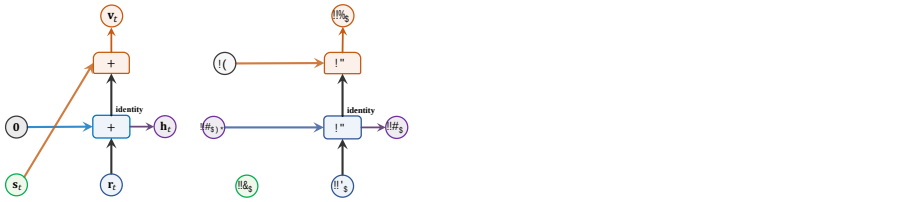


Figure 9: The influence of path length.

C Searched Models

C.1 Architectures in literature

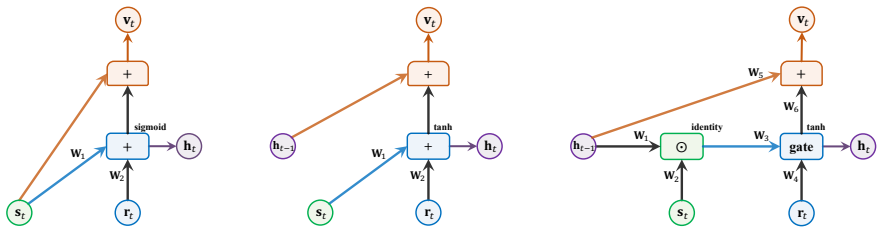


(a) TransE. (b) PTransE. (c) Chains. (d) RSN.

Figure 10: Graphical illustration of TransE, PTransE, Chains, RSN. TransE is a triplet-based model thus it is non-recurrent. PTransE recurrently processes the relational path without the intermediate entities. Chains simultaneously combines the entities and relations along the path, and RSN designs a customized variant of RNN. Each edge is an identity mapping, except an explicit mark of the weight matrix W_i. Activation functions are shown in the upper right corner of the operators.

C.2 Countries

We give the graphical illustration of architectures searched in Countries dataset in Figure 11. The search procedure is conducted in the whole search space rather than the four patterns P1-P4 in Figure 3. We can see that in Figure 11, (a) belongs to P1, (b) belongs to P2 and (c) belongs to P4. These results further verify that Interstellar can adaptively search architectures for specific KG tasks and datasets.



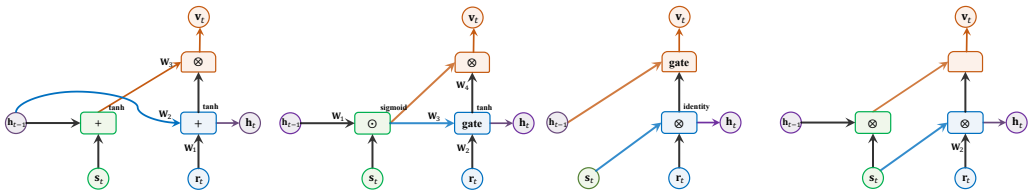
(a) S1 (b) S2 (c) S3

Figure 11: Graphical representation of the searched f on countries dataset.

C.3 Entity Alignment

The searched models by Interstellar, which consistently perform the best on each dataset, are graphically illustrated in Figure 12. As shown, all of the searched recurrent networks processing recurrent information in $\mathbf{h}_{t-1}$, subject entity embedding $\mathbf{s}_t$ and relation embedding $\mathbf{r}_t$ together. They have different connections, different composition operators and different activation functions, even though the searching starts with the same random seed *1234*.

More interestingly, the searched architectures in DBP-WD and DBP-YG can model multiple triplets simultaneously. While the architectures in EN-FR and EN-DE models two adjacent triplets each time. The reason is that, in the first two datasets, the KGs are from different sources, thus the distribution is various [19]. In comparison, EN-FR and EN-DE are two cross-lingual datasets, which should have more similar distributions in each KG. Therefore, we need to model the longer term information on DBP-WD and DBP-YG. In comparison, modeling the shorter term information like two triplets together is better for EN-FR and EN-DE. In this way, the model can focus more on learning short-term semantic in a range of two triplets.



(a) DBP-WD (b) DBP-YG (c) EN-FR (d) EN-DE

Figure 12: Graphical representation of the searched recurrent network f on each datasets in entity alignment task (Normal version).

(a) DBP-WD (b) DBP-YG (c) EN-FR (d) EN-DE

Figure 13: Graphical representation of the searched recurrent network f on each datasets in entity alignment task (Dense version).

C.4 Link Prediction

The best architectures searched in link prediction tasks are given in Figure 14. In order to illustrate the models we searched, we make a statistics of the distance, i.e. the shortest path distance when regarding the KG as a simple undirected graph, between two entities in the validation and testing set in Table 13. As shown, most of the triplets have distance less than 3. Besides, as indicated by the performance on the two datasets, we infer that triplets far away from 3-hop are very challenging to model. At least in this task, triplets less or equal than 3 hops are the main focus for different models. This also explains why RSN, which processes long relational path, does not perform well in the link prediction task. The searched models in Figure 14 do not directly consider long-term structures. Instead, the architecture on WN18-RR models one triplet, and the architecture on FB15k-237 focuses on modeling two consecutive triplets. These are consistent with the statistics in Table 13, where WN18-RR has more one-hop and FB15k-237 has more two-hop triplets.