Neural Architecture Search

- Another Example from Knowledge Graph

Dr. Quanming Yao

Assistant professor, EE Tsinghua

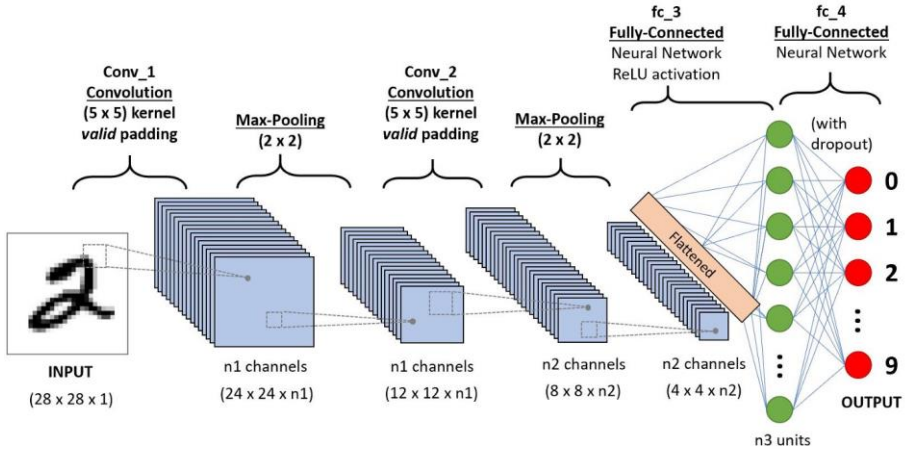
qyaoaa@tsinghua.edu.cn

https://lars-group.github.io/index.html

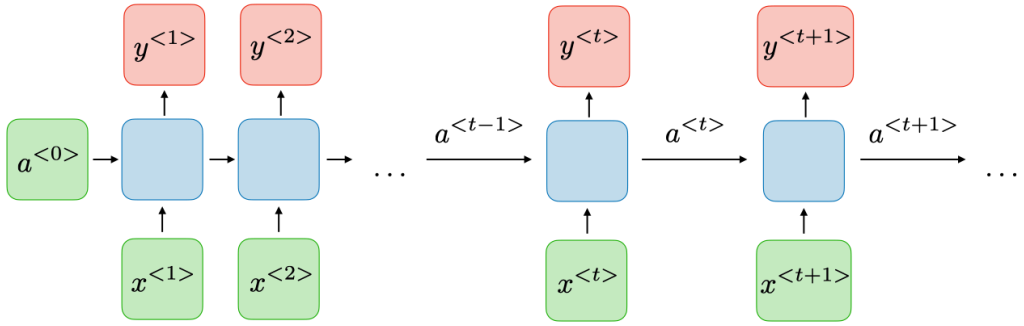
Outline

- What's Neural Architecture Search (NAS)
- Architectures for Knowledge Graph (KG) Learning
- Searching KG Learning Architectures
- Summary and Future Works

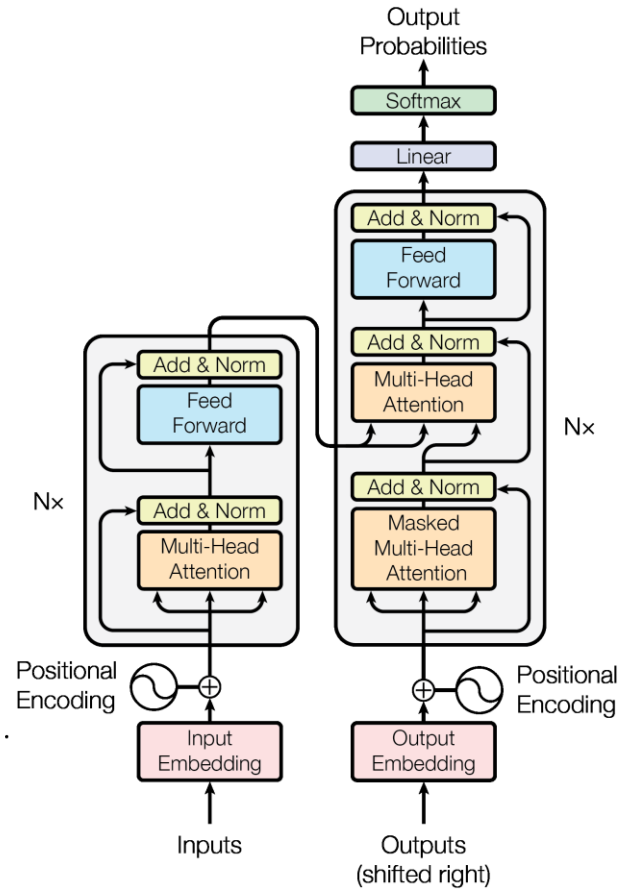
What are Neural Architectures?



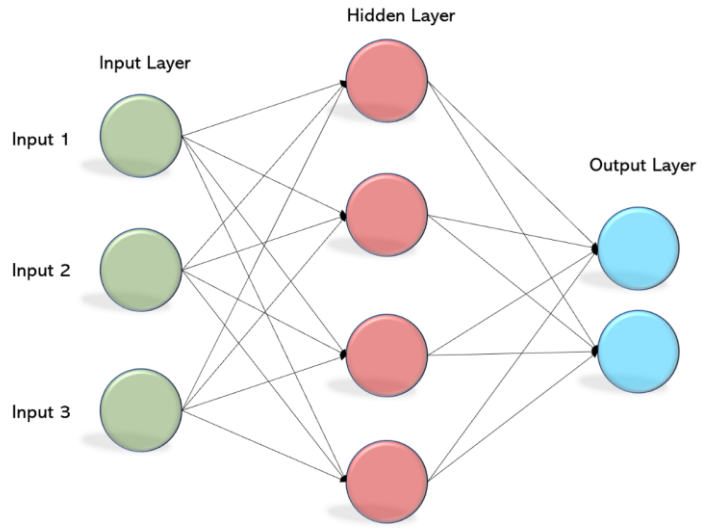
Convolutional neural network (CNN)



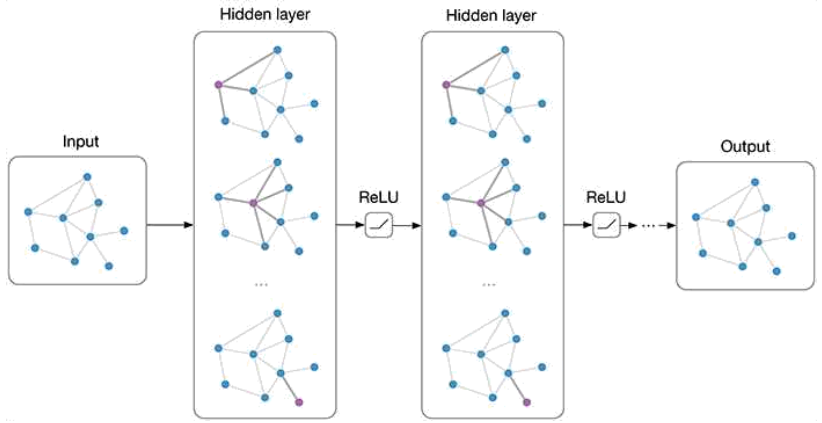
Recurrent Neural Networks (RNN)



Transformer



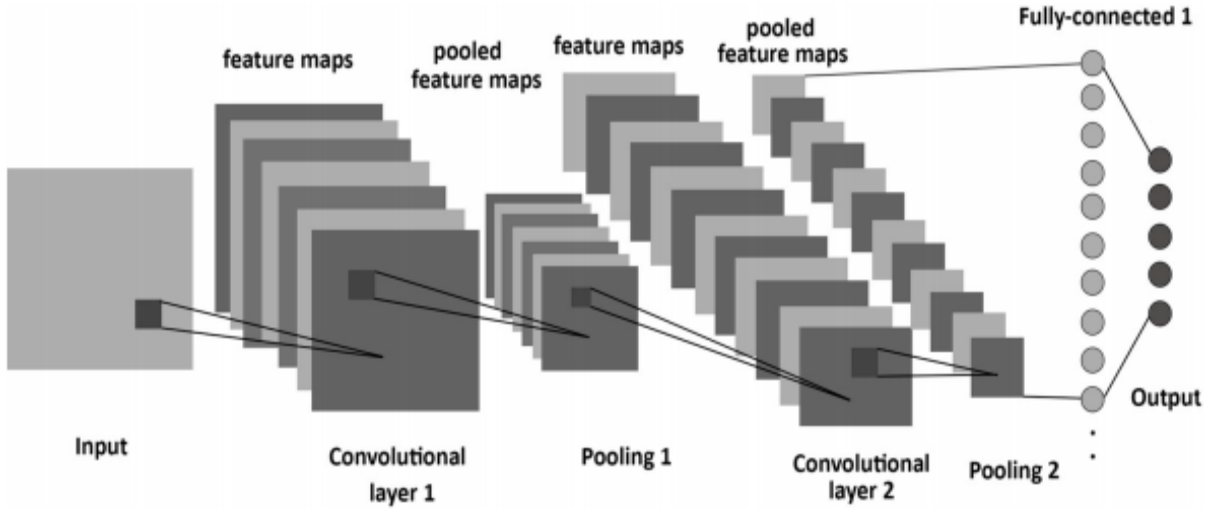
Multilayer perceptron (MLP)



Graph Neural Network (GNN)

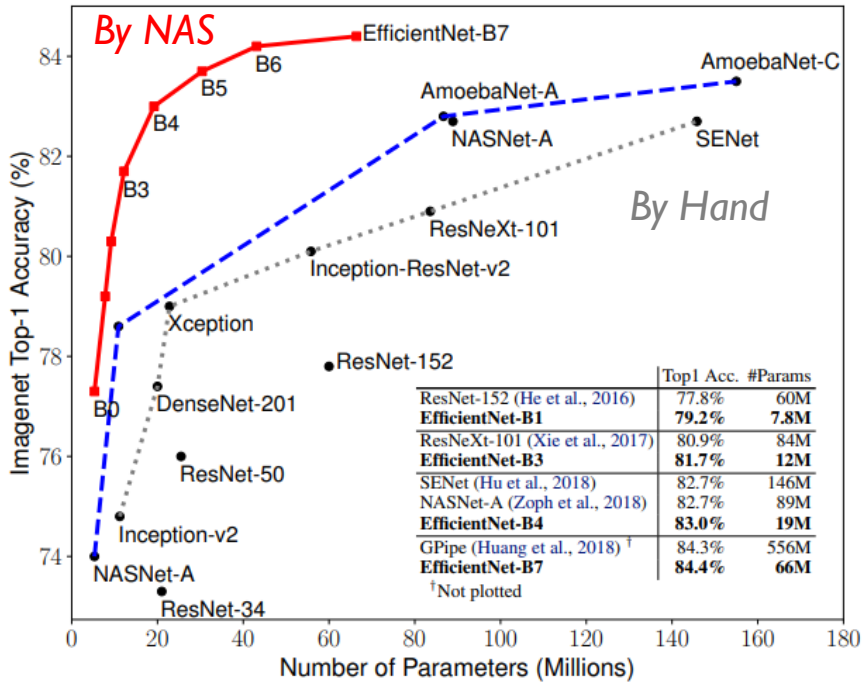
Why need to Search Architectures?

Architecture of networks are critical to deep learning's performance but hard to fine-tune



Design choice in each layer

- number of filters
- filter height
- filter width
- stride height
- stride width
- skip connections



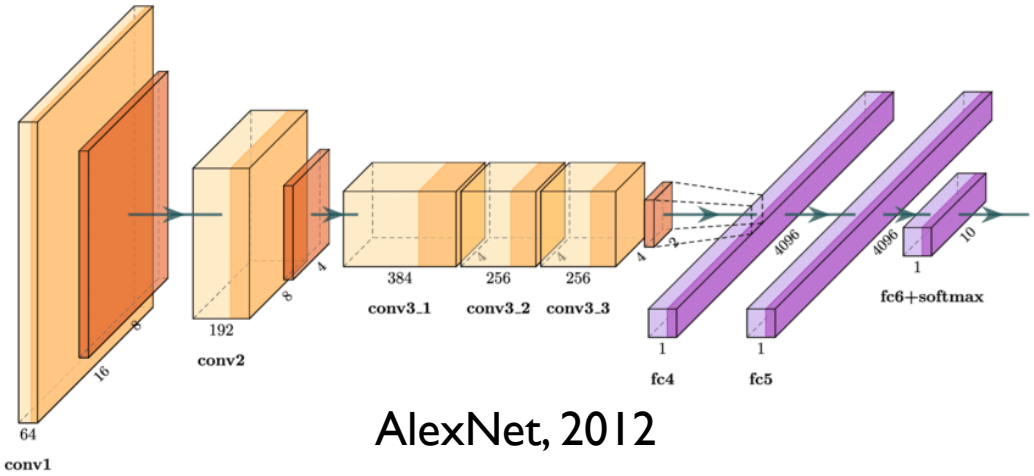
Much better than hand-designed ones

Neural Architecture Search (**NAS**) tries to directly optimize network architecture using validation data sets

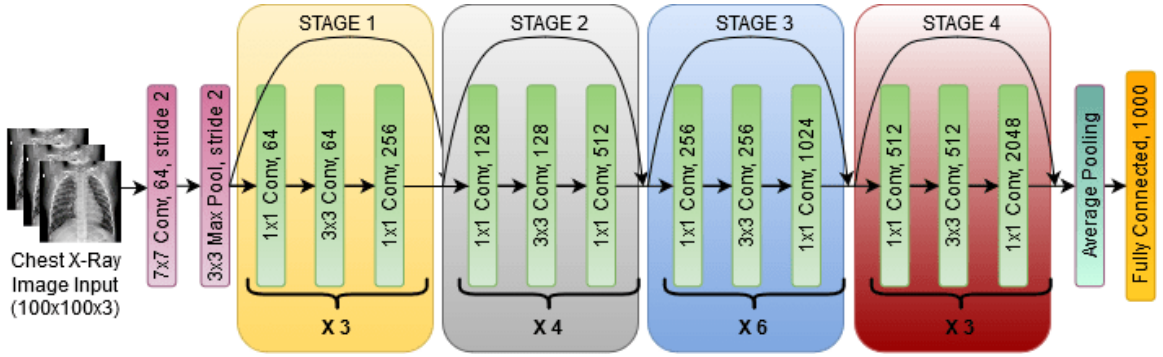
Right figure: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML 2019
Left figure: Gradient-based learning applied to document recognition. 1998

Evolving of Hand-design Architectures

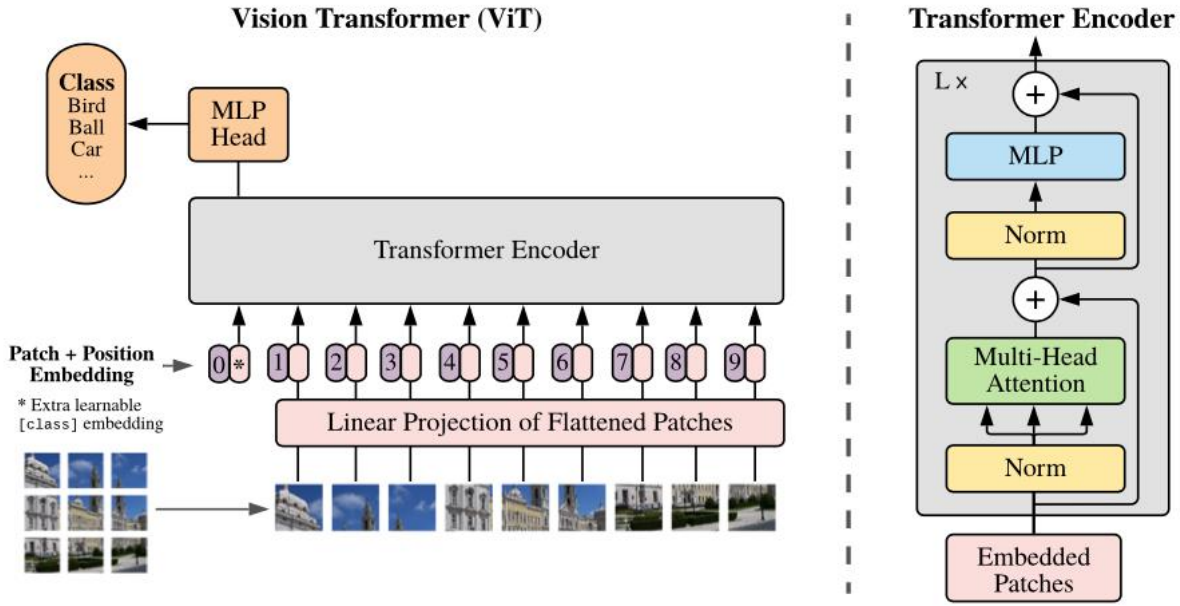
Architectures for image classification



AlexNet, 2012



ResNet, 2015



ViT, 2020

How to Search? Bilevel – again!

1. Define an AutoML (NAS) problem

- Derive a search space from insights in specific domains
- Search objective is usually validation performance
- Search constraint is usually resource budgets
- Training objective usually comes from classical learning models

$$\min_{\lambda \in \mathcal{S}} M(F(w^*; \lambda), D_{\text{val}})$$ ← Search Objective

Search Space →

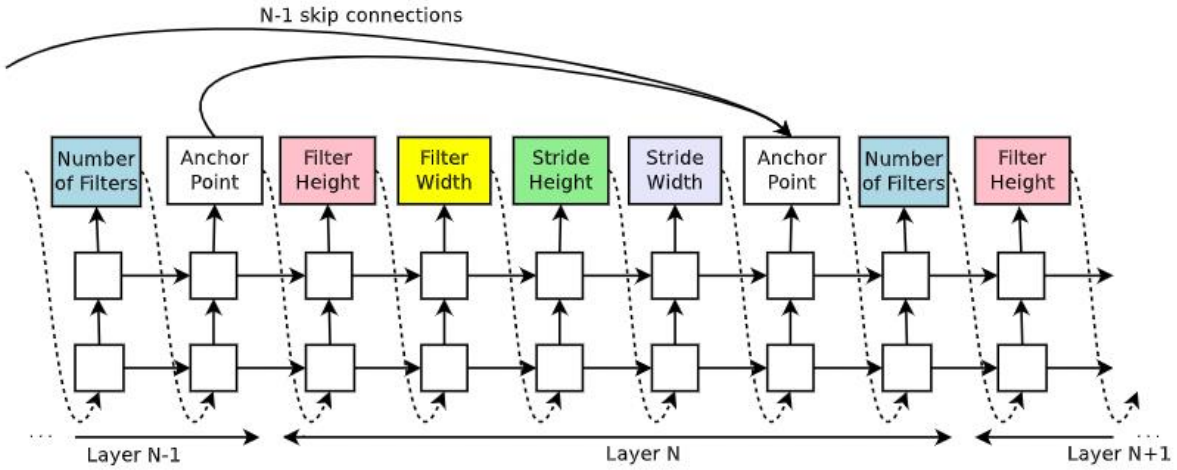
$$\text{s.t.} \quad \min_{w} L(F(w; \lambda), D_{\text{tra}})$$ ← Training Objective

$$G(\lambda) \leq C$$ ← Search Constraints

2. Design or select proper search algorithm

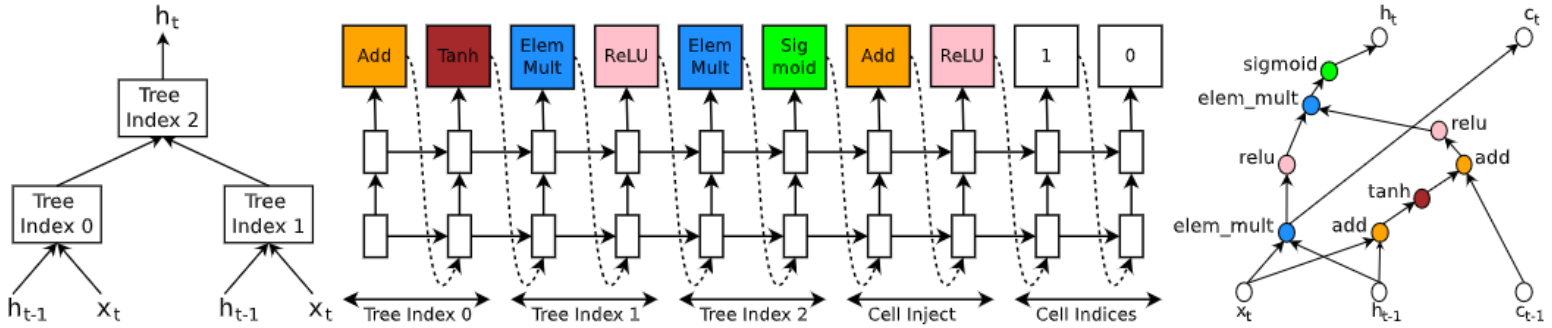
- Reduce model training cost (time to get w^*)

Pioneering Work – Define the problem



How controller RNN samples a simple CNN

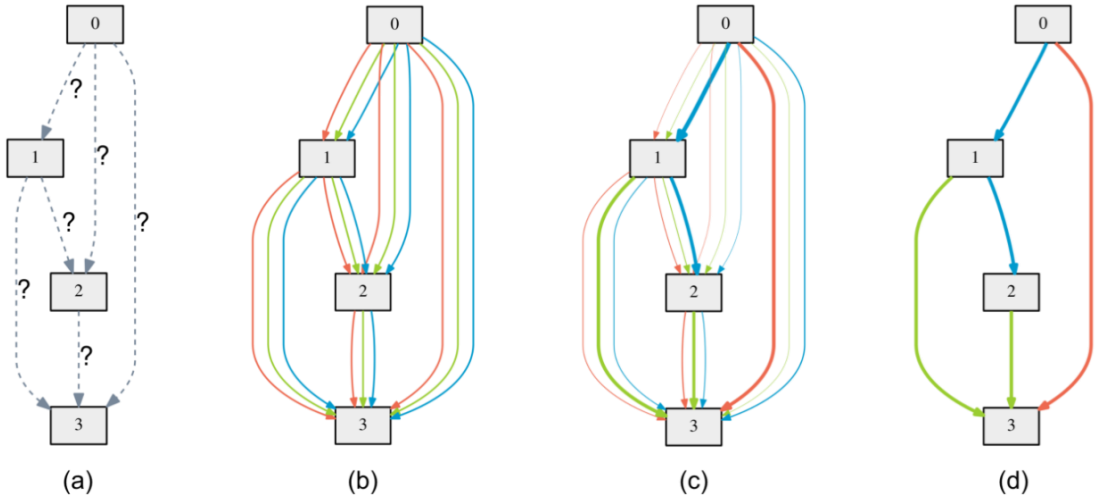
- Training with REINFORCE
- Consume a lot of computation resources!
  - 1e4 CPU days for language models on PTB



Example: a recurrent cell constructed from a tree

Neural Architecture Search with Reinforcement Learning, ICLR 2017

Pioneering Work – Make it fast



Overview of DARTS

(a). unknown operations on edges

(b). continuous relaxation of the search space

(c). joint optimization of the mixing probabilities and the network weights

(d). inducing the final architecture

0.5 GPU days for 1st order & 1 GPU days for 2nd order

(b). continuous relaxation

learnable parameter

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$$

candidate operations

(c). joint optimization

objective

$$\min_{\alpha} \quad \mathcal{L}_{val}(w^*(\alpha), \alpha)$$
$$\text{s.t.} \quad w^*(\alpha) = \text{argmin}_w \ \mathcal{L}_{train}(w, \alpha)$$

algorithm: SGD

$$\nabla_\alpha \mathcal{L}_{val}(w^*(\alpha), \alpha)$$
$$\approx \nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha)$$

(d). final architecture: most likely operation

$$o^{(i,j)} = \text{argmax}_{o \in \mathcal{O}} \ \alpha_o^{(i,j)}$$

Algorithm 1: DARTS – Differentiable Architecture Search

Create a mixed operation $\bar{o}^{(i,j)}$ parametrized by $\alpha^{(i,j)}$ for each edge (i, j)

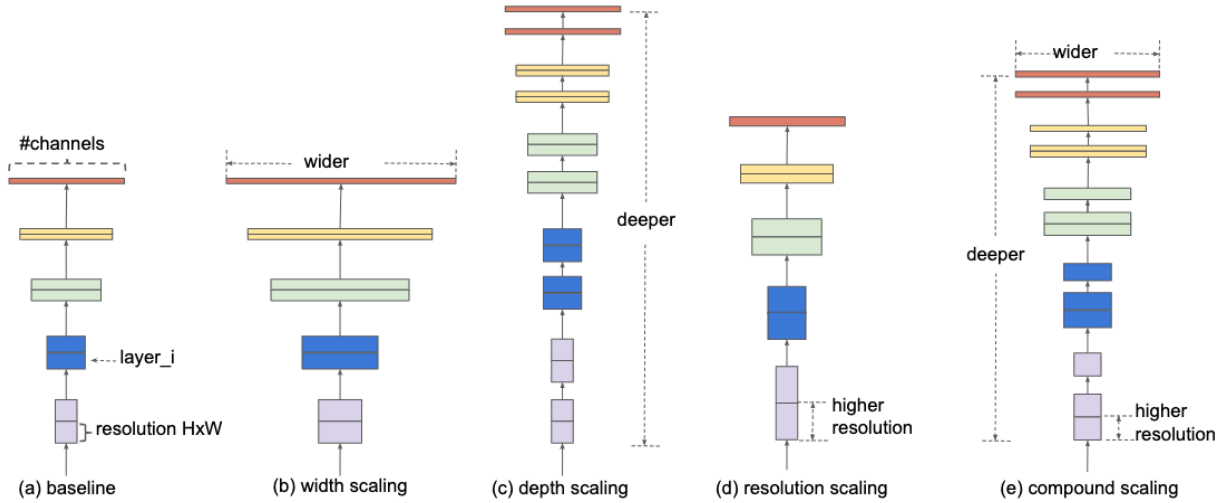
while *not converged* **do**

1. Update architecture α by descending $\nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha)$ ($\xi = 0$ if using first-order approximation)
2. Update weights w by descending $\nabla_w \mathcal{L}_{train}(w, \alpha)$

Derive the final architecture based on the learned α.

DARTS: Differentiable Architecture Search, ICLR 2019

Pioneering Work – Make it large



model scaling

Optimization objective

$$\max_{d,w,r} \quad Accuracy\big(\mathcal{N}(d,w,r)\big)$$

$$s.t. \quad \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\big(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\big)$$

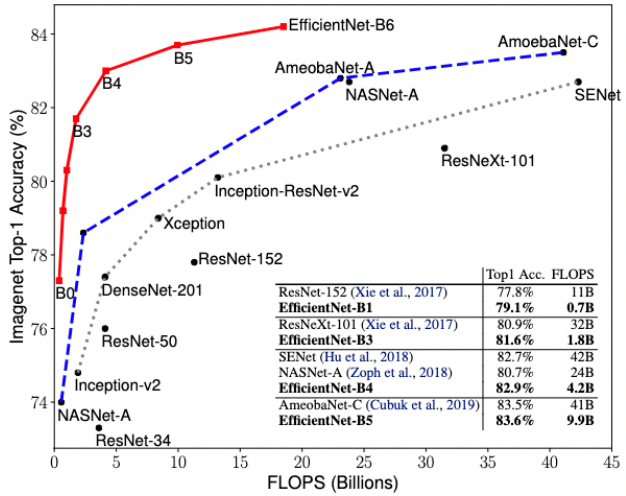
$$\text{Memory}(\mathcal{N}) \le \text{target\_memory}$$

$$\text{FLOPS}(\mathcal{N}) \le \text{target\_flops}$$

maximize network accuracy with appropriate depth, width, and resolution

EfficientNet

- search for a group of hyperparameters based on NAS algorithm
- find optimal hyperparameters: depth (d), width (w), and resolution (r)
- search for hyperparameters with grid search



EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, ICML 2019

Some Recent Progress

- Convolutional Neural Network
  - GAEA (Li, et al. 2021, ICLR): combine with different types of weight-sharing algorithms (DARTS, PC-DARTS, etc) and reduce the impact of the architecture discretization step due to finding sparser solutions
- Transformer
  - NASViT (Gong, et al. 2022, ICLR): identify one key issue of ViT supernet training that the supernet gradients and the sub-network gradients are likely to disagree with each other, and propose gradient conflict aware training
- Graph Neural Network
  - SANE (Zhao, et al. 2021, ICDE): search for aggregation functions in graph neural networks, and introduce differentiable algorithms into automated graph neural networks for the first time
  - PAS (Wei, et al. 2021, CIKM): searching for adaptive pooling structures for graph classification using bilevel optimization
  - AutoBLM (Zhang et al. 2022, TPAMI): bilinear scoring function search for knowledge graph learning

Outline

- What's Neural Architecture Search (NAS)
- Architectures for Knowledge Graph (KG) Learning
  - What's KG Learning
  - Triplet-based Models
- Searching KG Learning Architectures
- Summary and Future Works

Knowledge graph by Google

what is knowledge graph

All Images News Videos Maps More Settings Tools

About 358,000,000 results (0.49 seconds)

en.wikipedia.org › wiki › Knowledge_Graph

Knowledge Graph - Wikipedia

The **Knowledge Graph** is a **knowledge** base used by Google and its services to enhance its search engine's results with information gathered from a variety of sources. ... Information from the **Knowledge Graph** is used to answer direct spoken questions in Google Assistant and Google Home voice queries.

Knowledge base · Knowledge engine · Ontology (information science) · Freebase

People also ask

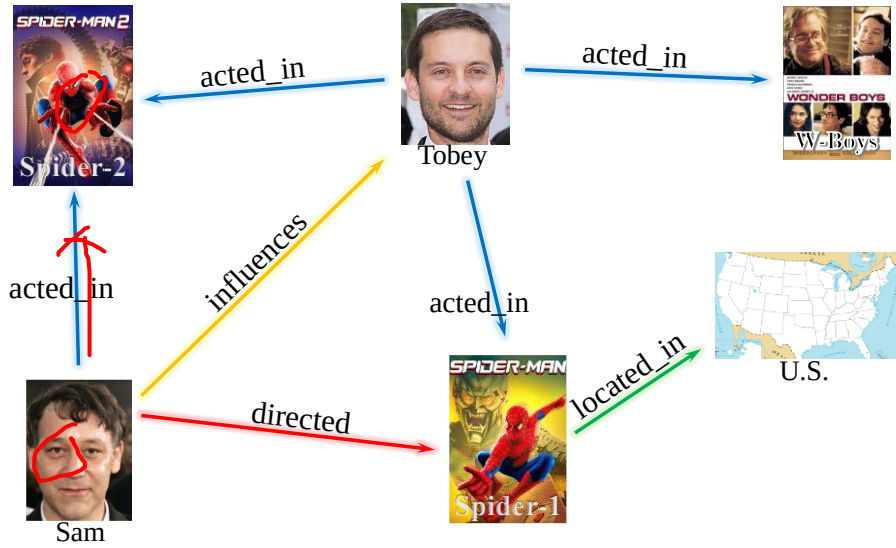
- How does a knowledge graph work?
- Why is knowledge graph important?
- What is Knowledge Graph in SEO?
- How do you make a knowledge graph?

Knowledge Graph

The Knowledge Graph is a knowledge base used by Google and its services to enhance its search engine's results with information gathered from a variety of sources. The information is presented to users in an infobox next to the search results. Wikipedia

Feedback

Knowledge graph (KG)



Graph representation: $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{S})$.

Entities $\mathcal{E}$: real world objects or abstract concepts.

Relations $\mathcal{R}$: interactions between/among entities.

Fact/triples $\mathcal{S}$: the basic unit in form of (head entity, relation, tail entity), (h, r, t).

Other related information:

- Types/attributes of entities/relations.
- Text descriptions on entities and relations.
- Ontologies: concept level description.
- Logic rules: regular expressions.

Google Knowledge Graph

范式星图 地址知识图谱工具

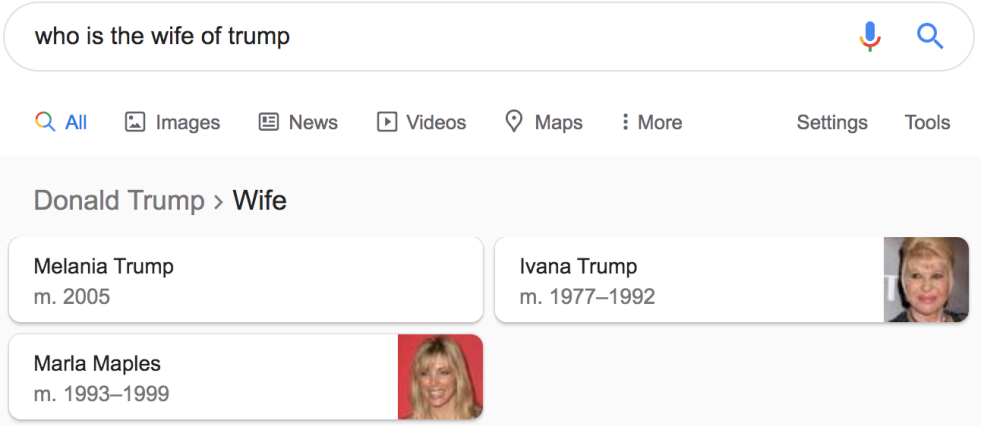
WIKIDATA

Freebase™

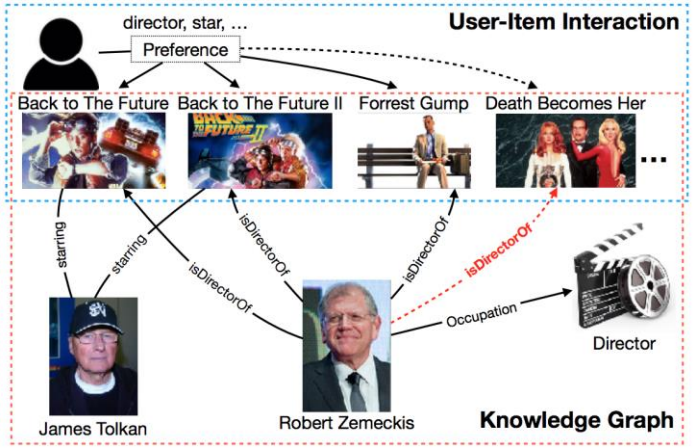
BIO2RDF

KG – important applications

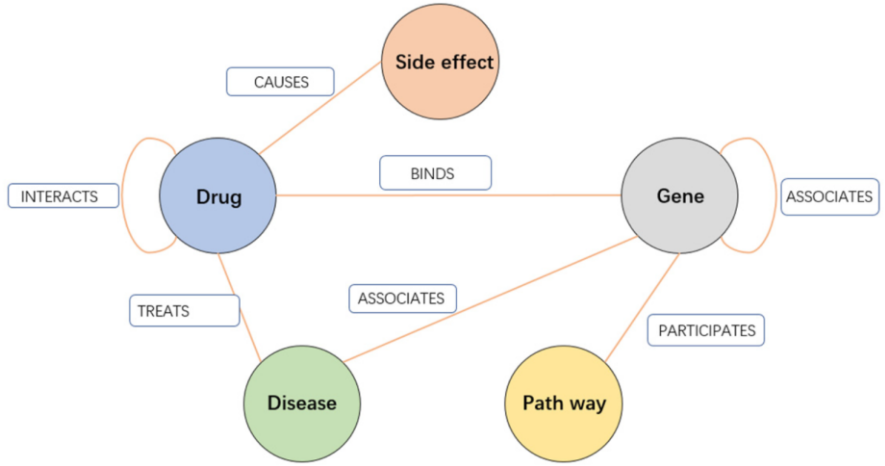
KGQA:



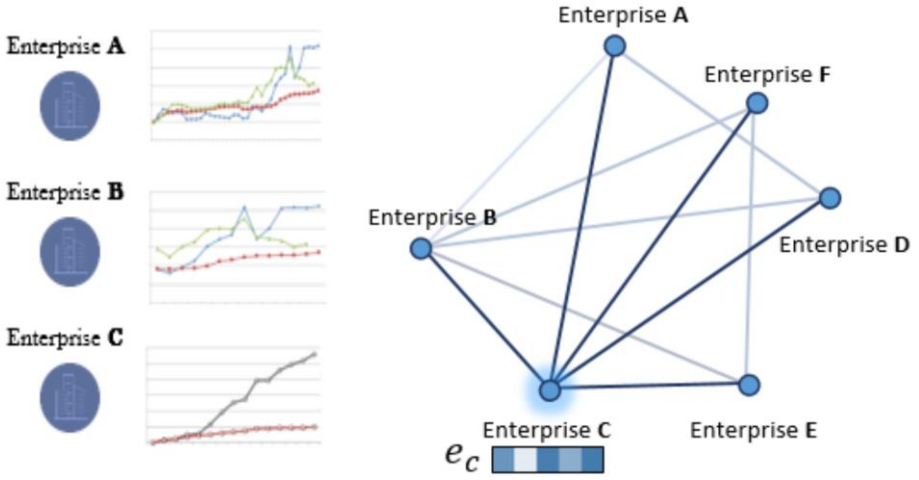
Recommendation:



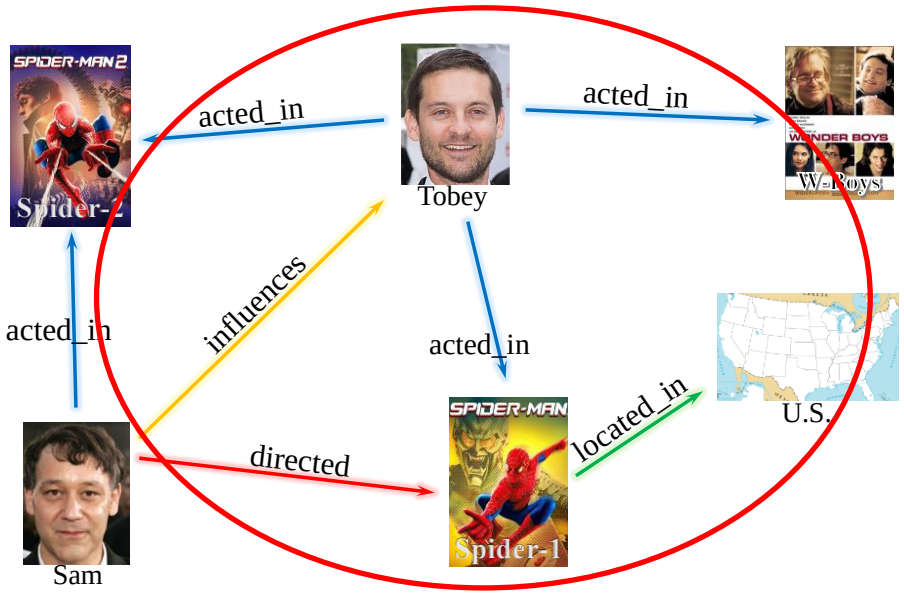
Drug discovery:



Stock prediction:



KG Learning

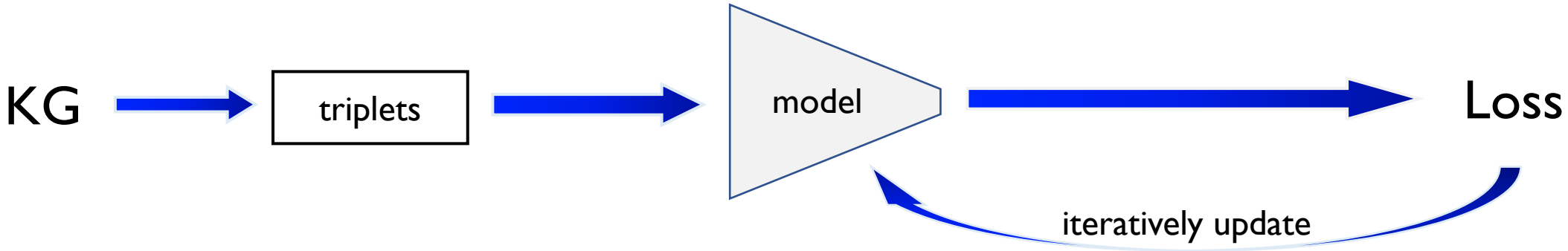


Observed triplets S^+:
maximize score

(Sam, directed, Spider-1)
(Tobey, acted_in, Spider-2)
(Spider-1, located_in, U.S.)
…

Unobserved triplets S^-:
minimize score

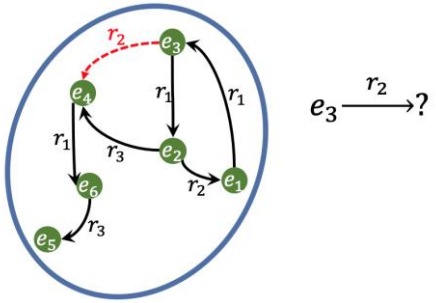
(Tobey, directed, Spider-1)
(Tobey, acted_in, U.S.)
(Spider-1, directed, U.S.)
…



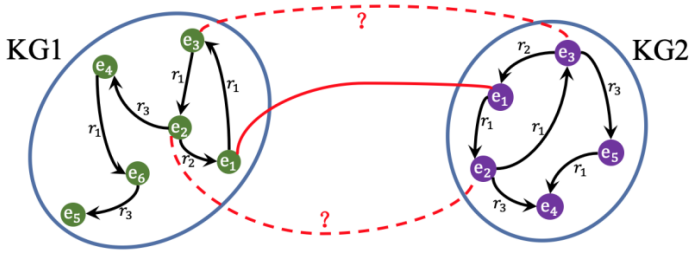
➢Objective: Preserve as much information on original graph as possible.

Basic tasks

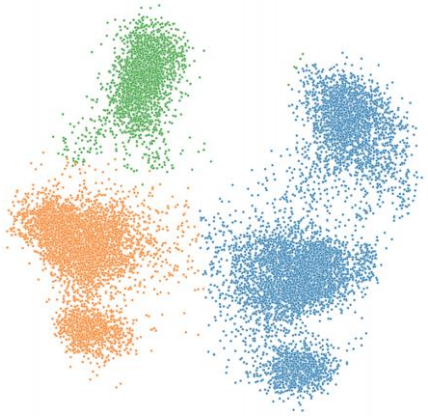
- Knowledge graph completion
  - Link prediction / triple classification: predicting missing links



- Entity alignment
  - Align the same entities in two KGs, e.g. cross-lingual KGs



- Entity classification
  - Predict the type of entities



Evaluation

- Link prediction as an example.

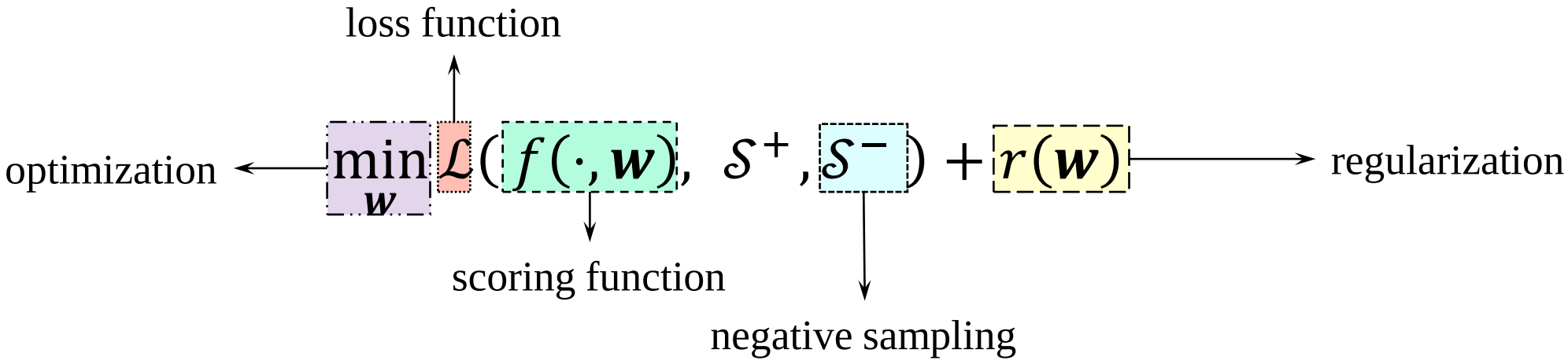
- Evaluation on (h, r, t):
  - head/tail prediction: $(?, r, t)$ or $(h, r, ?)$
  - get the rank of (h, r, t) over $\{(e, r, t): e \in \mathcal{E}\}$ or $\{(h, r, e): e \in \mathcal{E}\}$

- Metrics:

$$MR = \frac{1}{|\mathcal{S}_{\text{tst}}|} \sum_{i \in \mathcal{S}_{tst}} \text{rank}_i \qquad MRR = \frac{1}{|\mathcal{S}_{\text{tst}}|} \sum_{i \in \mathcal{S}_{tst}} \frac{1}{\text{rank}_i} \qquad H@K = \frac{|\{i \in \mathcal{S}_{\text{tst}}: \text{rank}_i \leq K\}|}{|\mathcal{S}_{\text{tst}}|}$$

Learning problem in KGR

- As a learning problem, the KG reasoning problem contains the following important components:



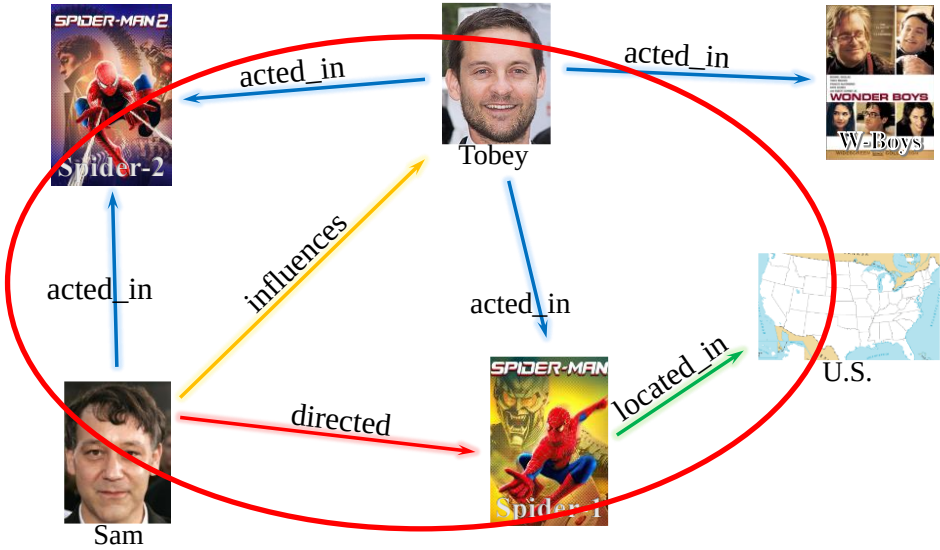
Outline

- What's Neural Architecture Search (NAS)
- Architectures for Knowledge Graph (KG) Learning
  - What's KG Learning
  - Triplet-based Models
- Searching KG Learning Architectures
- Summary and Future Works

Models

Query: (Sam, *directed*, ?)

Answer: Spider-1, Spider-2



triplet-based

(Sam, *directed*, Spider-2)

RESCAL [Nickel et. al. 2011]
TransE [Bordes et. al. 2013]
ComplEx [Trouillion et. al. 2017]
ConvE [Dettmers et. al.]
AutoSF [Zhang et. al. 2020]

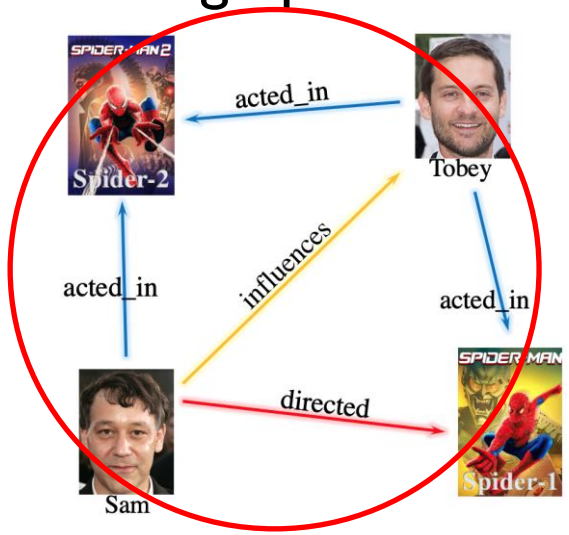
path-based

$$\text{Sam} \xrightarrow{directed} \text{Spider-1} \xrightarrow{acted\_in} \text{Tobey} \xrightarrow{acted\_in} \text{Spider-2}$$

$$\text{Sam} \xrightarrow{influences} \text{Tobey} \xrightarrow{acted\_in} \text{Spider-2}$$

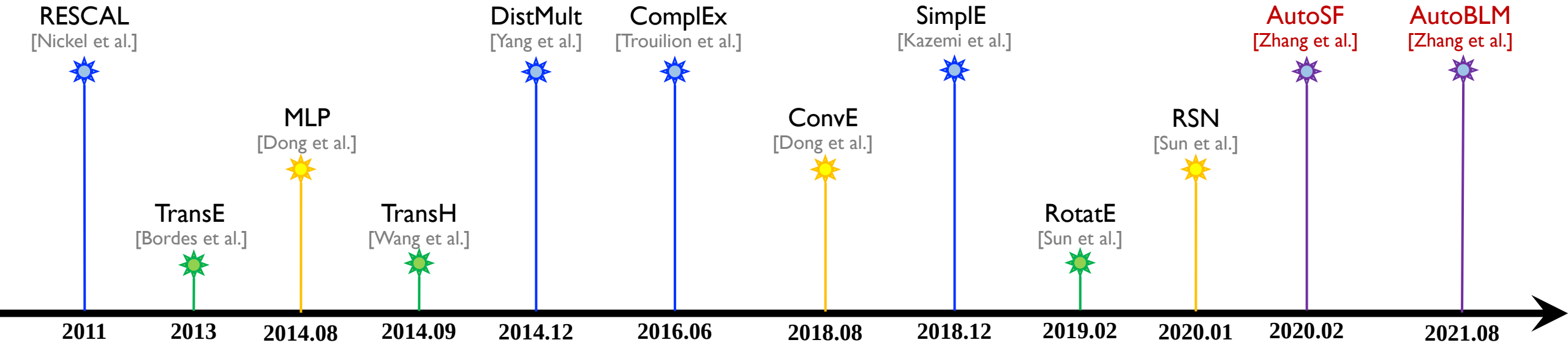
PTransE [Lin et. al. 2015]
RSN [Dettmers et. al.]
Interestellar [Zhang et. al. 2020]

subgraph-based



GraIL [Teru et. al. 2019]
NBFNet [Zhu et. al. 2021]
RED-GNN [Zhang et. al. 2022]

Triplet-based models

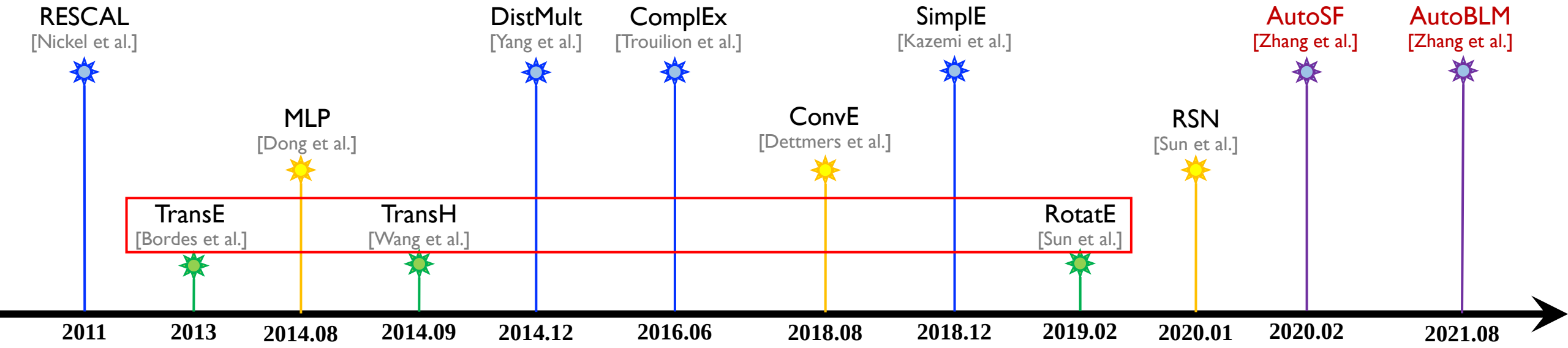


A large amount of scoring functions (SFs) $f(h, r, t)$ are defined to measure the plausibility of triplets $\{(h, r, t)\}$ in KG.

Capture important properties:

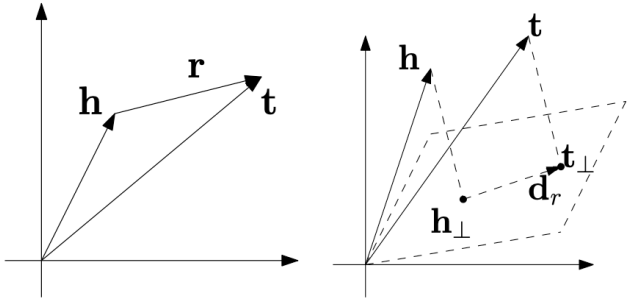
- symmetric, anti-symmetric, inverse, asymmetric…

Translational Distance Model (TDM)



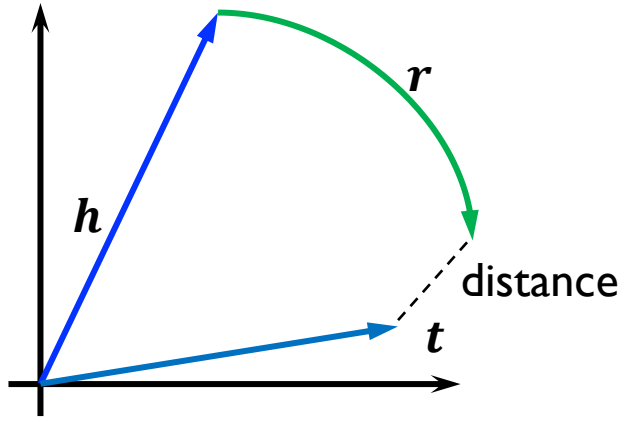
TransE: $f(h,r,t) = -\|(\boldsymbol{h}+\boldsymbol{r})-\boldsymbol{t}\|_p$

TransH: $f(h,r,t) = -\|(\boldsymbol{h}'+\boldsymbol{r})-\boldsymbol{t}'\|_p$



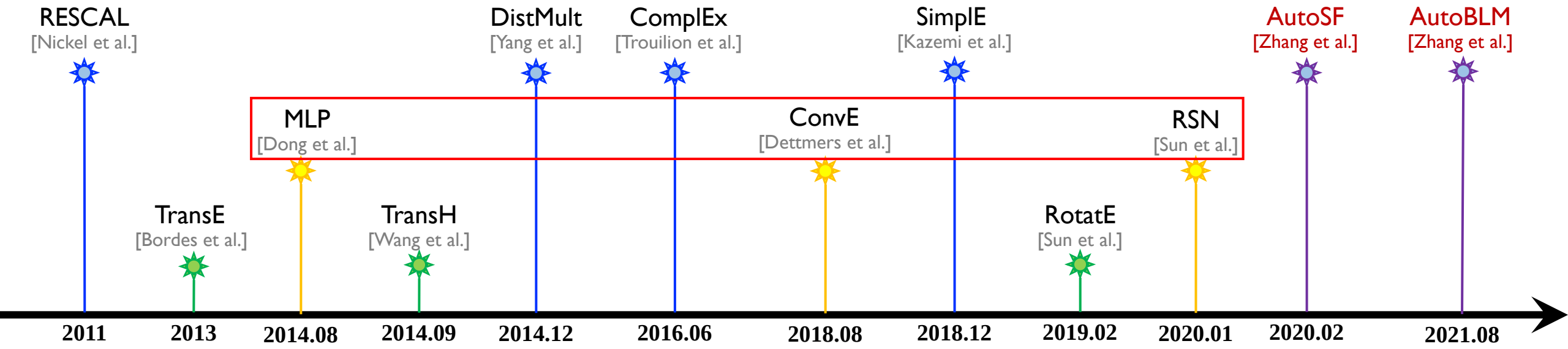
(a) TransE (b) TransH

RotatE: $f(h,r,t) = -\|(\boldsymbol{h}\circ\boldsymbol{r})-\boldsymbol{t}\|_p$

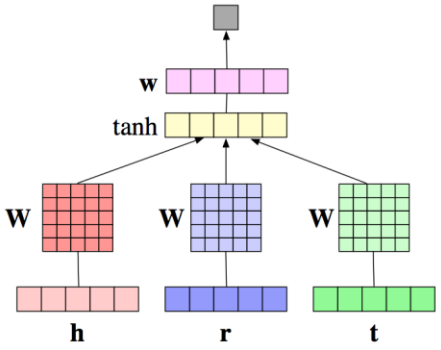


less expressive

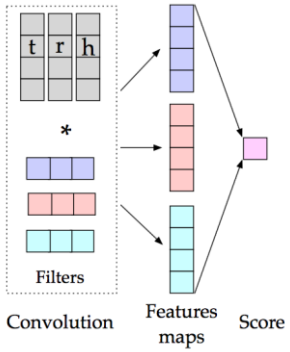
Neural Network Model (NNM)



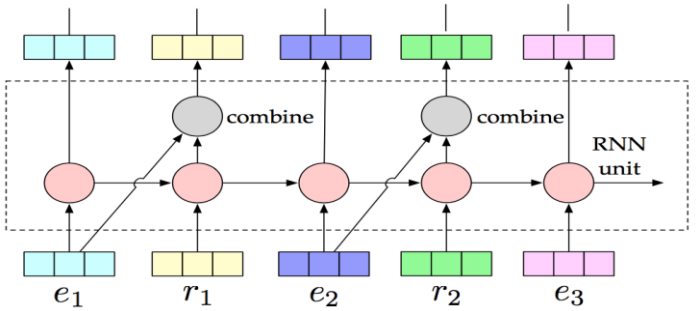
MLP



ConvE

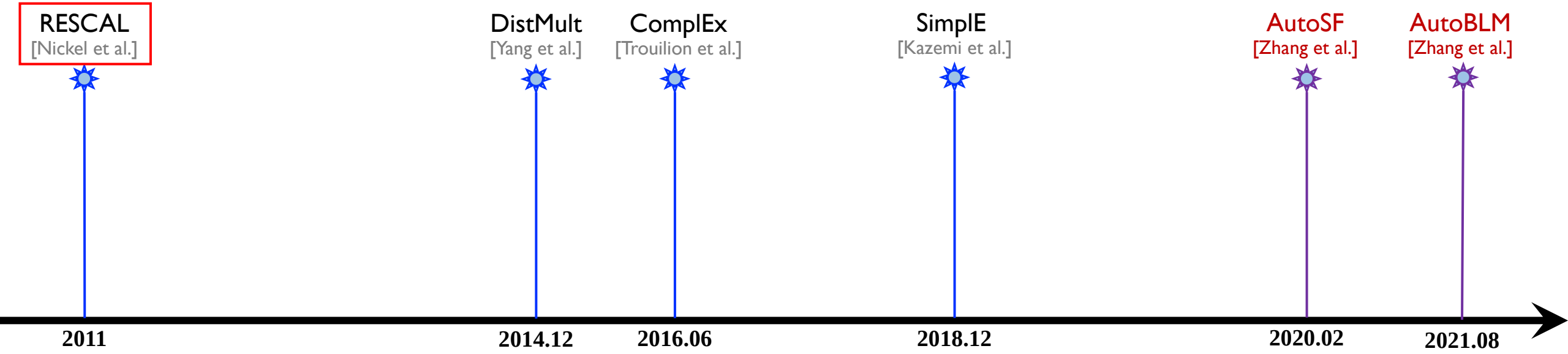


RSN

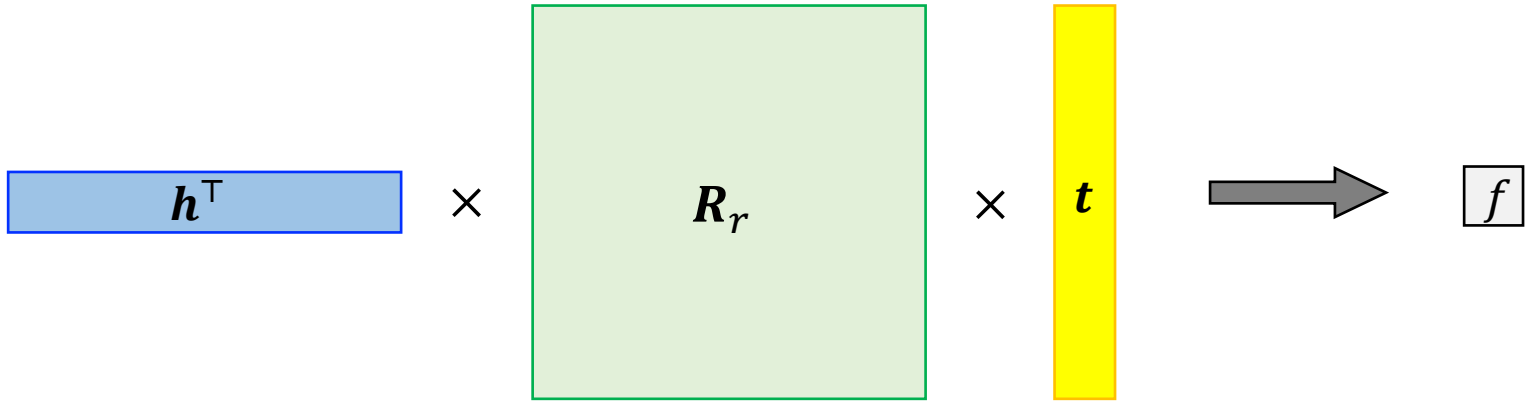


complex
and
difficult to train

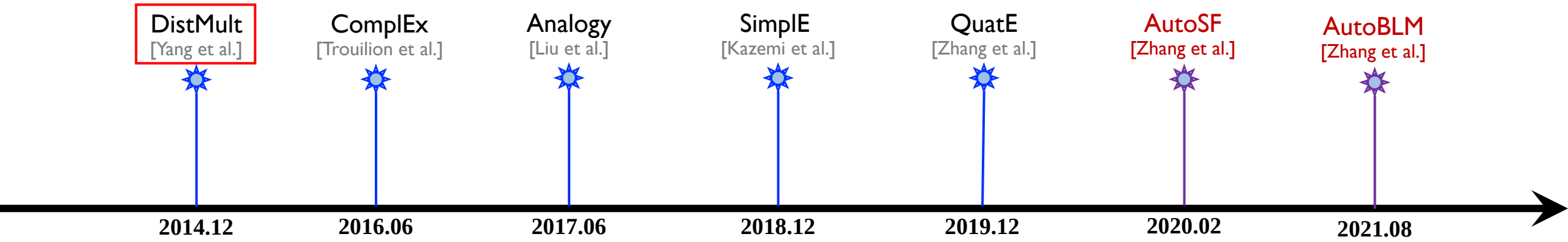
BiLinear Model (BLM)



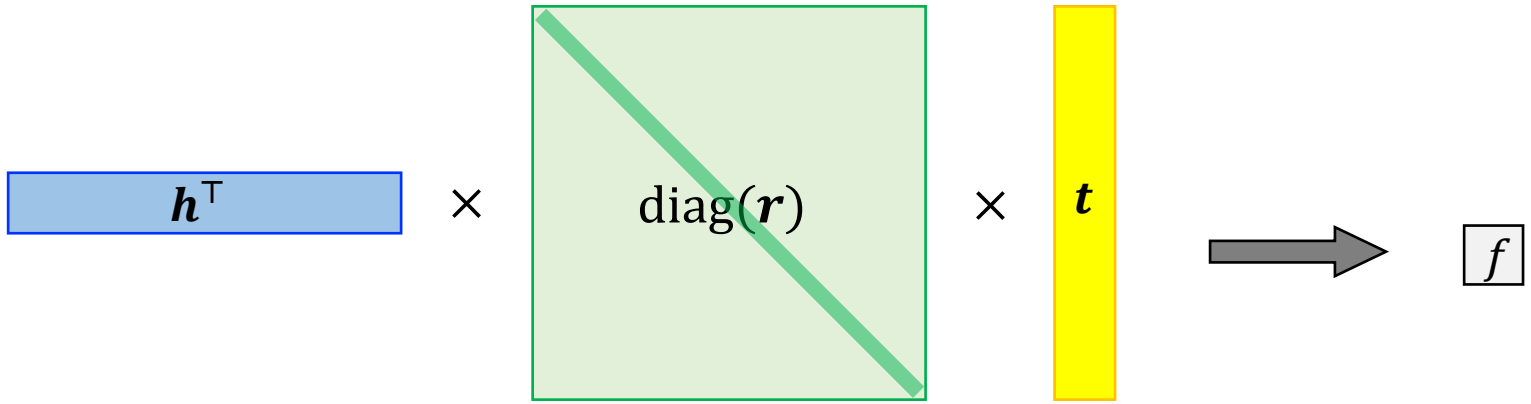
RESCAL: $f(h, r, t) = \boldsymbol{h}^\top \boldsymbol{R}_r \boldsymbol{t}$



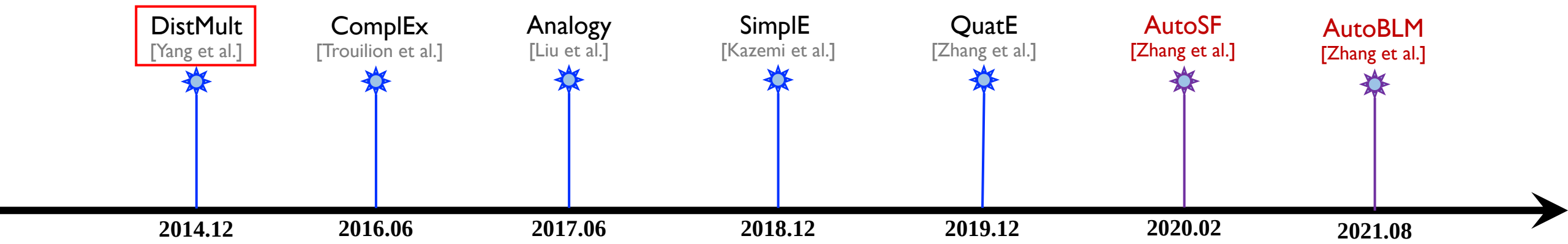
DistMult



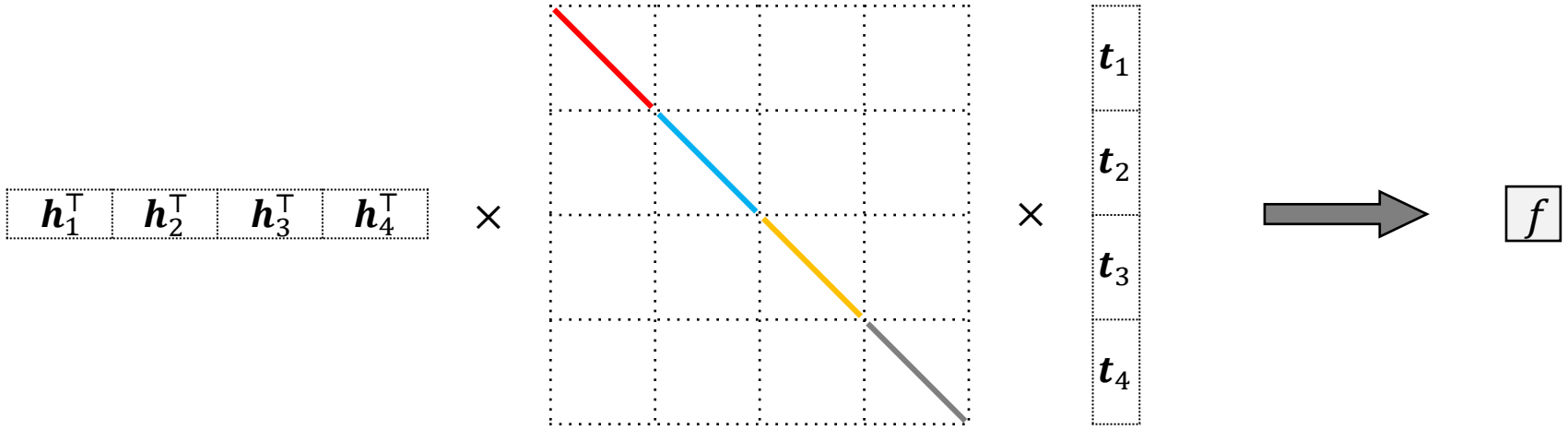
DistMult: $f(h,r,t) = \boldsymbol{h}^\top \mathrm{diag}(\boldsymbol{r})\boldsymbol{t} = \langle \boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \rangle$



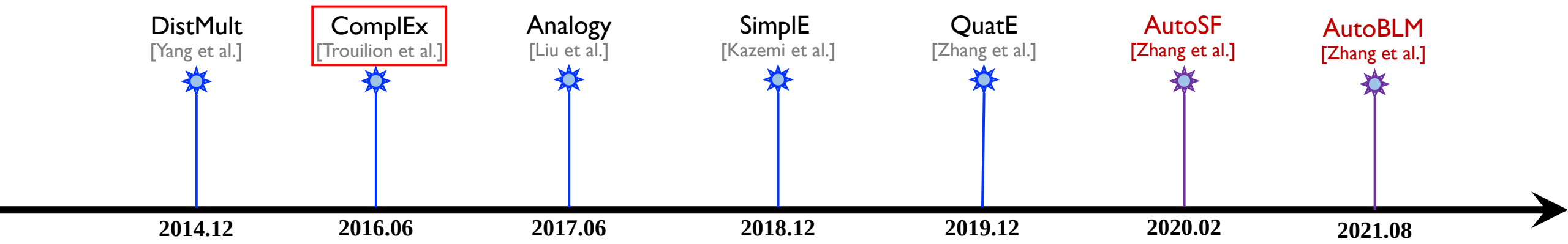
DistMult - splition



DistMult: $f(h,r,t) = \langle \boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \rangle = \langle \boldsymbol{h}_1, \boldsymbol{r}_1, \boldsymbol{t}_1 \rangle + \langle \boldsymbol{h}_2, \boldsymbol{r}_2, \boldsymbol{t}_2 \rangle + \langle \boldsymbol{h}_3, \boldsymbol{r}_3, \boldsymbol{t}_3 \rangle + \langle \boldsymbol{h}_4, \boldsymbol{r}_4, \boldsymbol{t}_4 \rangle$

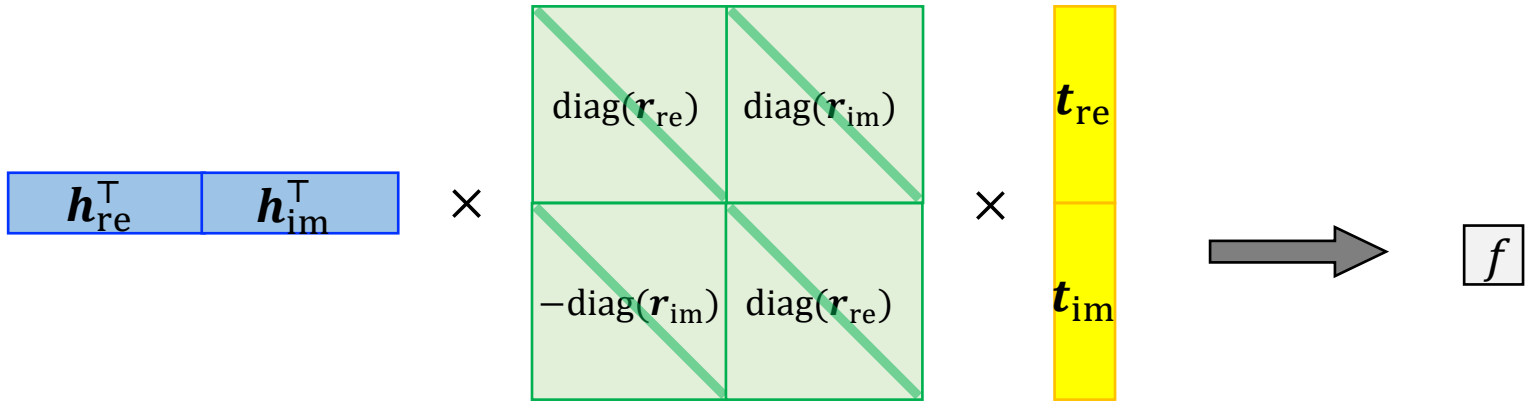


ComplEx

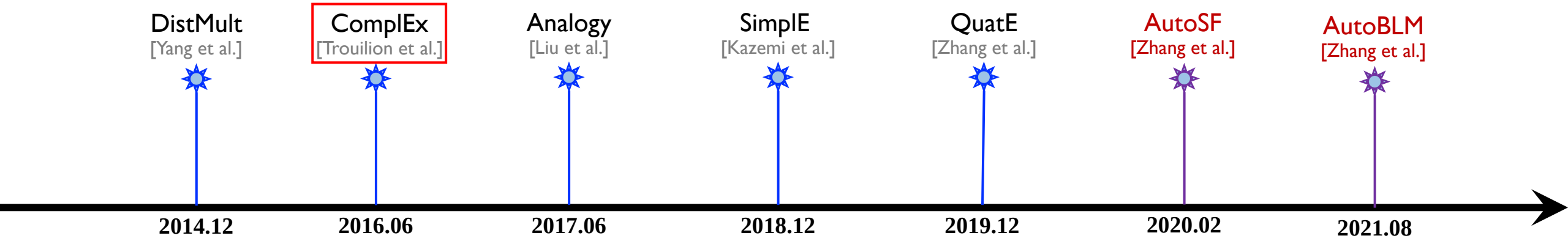


ComplEx:

$$f(h,r,t) = \mathrm{Re}(\boldsymbol{h}^\top \mathrm{diag}(\boldsymbol{r})\bar{\boldsymbol{t}}) = \langle \boldsymbol{h}_{\mathrm{re}}, \boldsymbol{r}_{\mathrm{re}}, \boldsymbol{t}_{\mathrm{re}} \rangle + \langle \boldsymbol{h}_{\mathrm{im}}, \boldsymbol{r}_{\mathrm{re}}, \boldsymbol{t}_{\mathrm{im}} \rangle + \langle \boldsymbol{h}_{\mathrm{re}}, \boldsymbol{r}_{\mathrm{im}}, \boldsymbol{t}_{\mathrm{im}} \rangle - \langle \boldsymbol{h}_{\mathrm{im}}, \boldsymbol{r}_{\mathrm{im}}, \boldsymbol{t}_{\mathrm{re}} \rangle$$

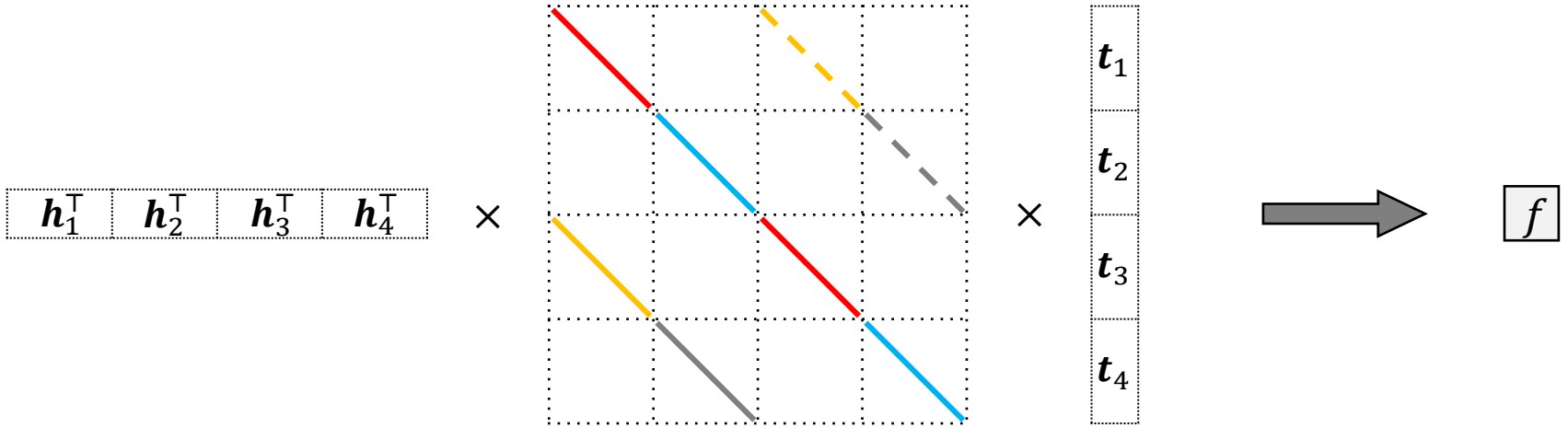


ComplEx-splition

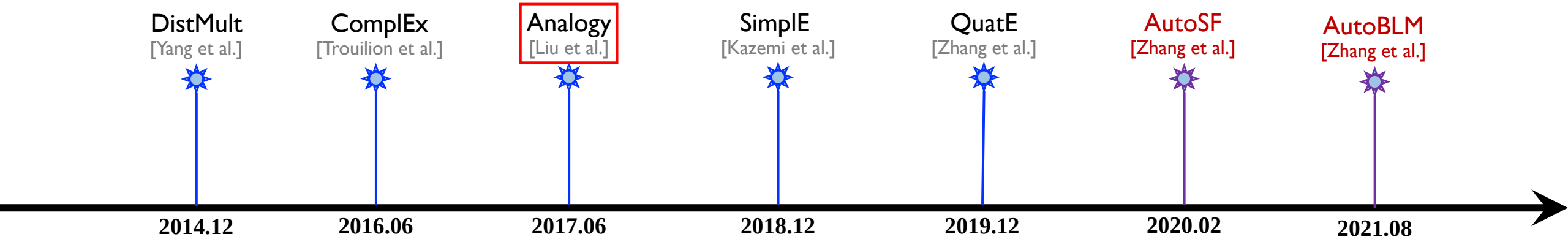


ComplEx:

$$f(h,r,t) = \langle \boldsymbol{h}_{\mathrm{re}}, \boldsymbol{r}_{\mathrm{re}}, \boldsymbol{t}_{\mathrm{re}} \rangle + \langle \boldsymbol{h}_{\mathrm{im}}, \boldsymbol{r}_{\mathrm{re}}, \boldsymbol{t}_{\mathrm{im}} \rangle + \langle \boldsymbol{h}_{\mathrm{re}}, \boldsymbol{r}_{\mathrm{im}}, \boldsymbol{t}_{\mathrm{im}} \rangle - \langle \boldsymbol{h}_{\mathrm{im}}, \boldsymbol{r}_{\mathrm{im}}, \boldsymbol{t}_{\mathrm{re}} \rangle$$
$$= \langle \boldsymbol{h}_1, \boldsymbol{r}_1, \boldsymbol{t}_1 \rangle + \langle \boldsymbol{h}_2, \boldsymbol{r}_2, \boldsymbol{t}_2 \rangle + \langle \boldsymbol{h}_3, \boldsymbol{r}_1, \boldsymbol{t}_3 \rangle + \langle \boldsymbol{h}_4, \boldsymbol{r}_2, \boldsymbol{t}_4 \rangle$$
$$+ \langle \boldsymbol{h}_1, \boldsymbol{r}_3, \boldsymbol{t}_3 \rangle + \langle \boldsymbol{h}_2, \boldsymbol{r}_4, \boldsymbol{t}_4 \rangle - \langle \boldsymbol{h}_3, \boldsymbol{r}_3, \boldsymbol{t}_1 \rangle - \langle \boldsymbol{h}_4, \boldsymbol{r}_4, \boldsymbol{t}_2 \rangle$$

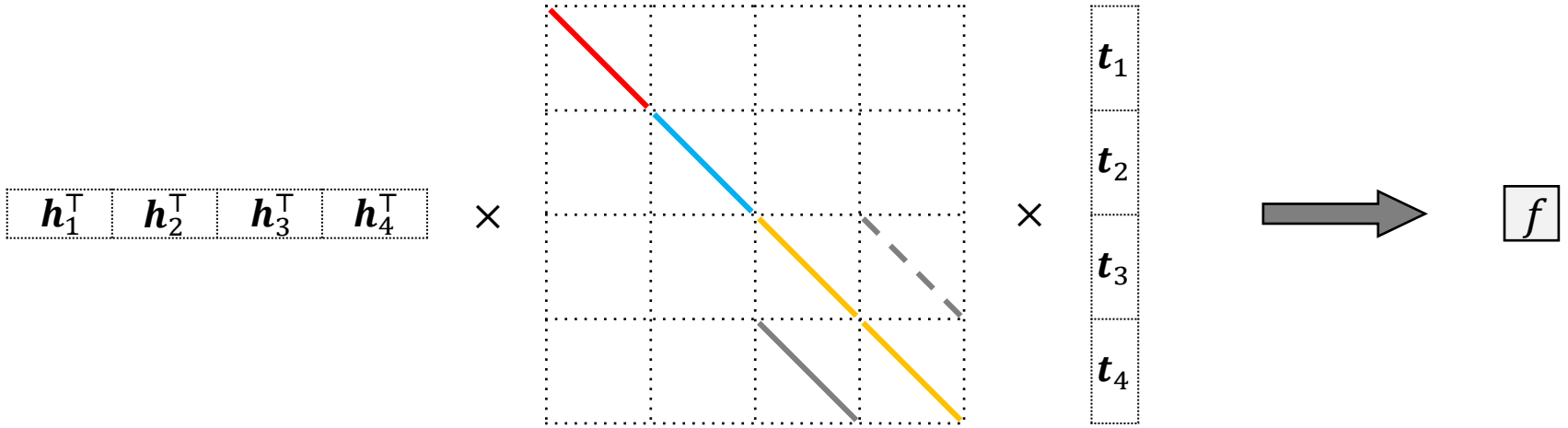


Analogy-splition

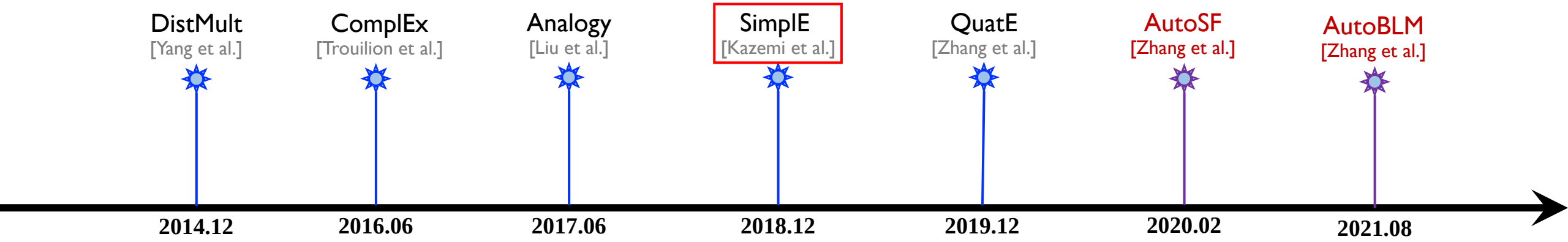


Analogy:

$$f(h,r,t) = \langle \hat{\boldsymbol{h}}, \hat{\boldsymbol{r}}, \hat{\boldsymbol{t}} \rangle + Re\big(\langle \underline{\boldsymbol{h}}, \underline{\boldsymbol{r}}, \bar{\boldsymbol{t}} \rangle\big)$$
$$= \langle \boldsymbol{h}_1, \boldsymbol{r}_1, \boldsymbol{t}_1 \rangle + \langle \boldsymbol{h}_2, \boldsymbol{r}_2, \boldsymbol{t}_2 \rangle$$
$$+ \langle \boldsymbol{h}_3, \boldsymbol{r}_3, \boldsymbol{t}_3 \rangle + \langle \boldsymbol{h}_4, \boldsymbol{r}_4, \boldsymbol{t}_4 \rangle + \langle \boldsymbol{h}_4, \boldsymbol{r}_3, \boldsymbol{t}_4 \rangle - \langle \boldsymbol{h}_4, \boldsymbol{r}_4, \boldsymbol{t}_3 \rangle$$

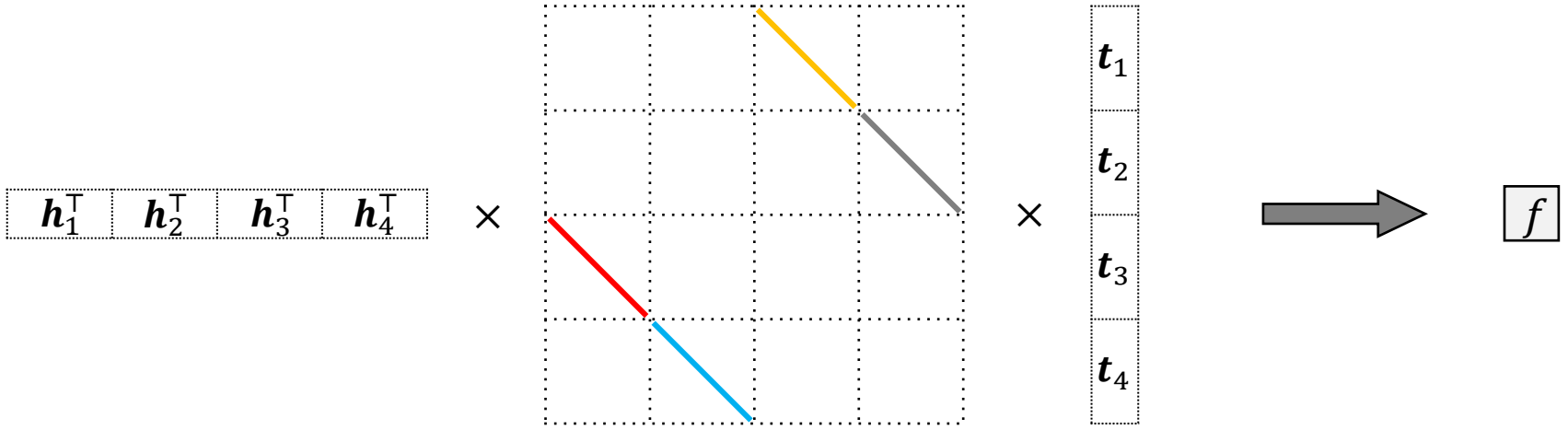


SimplE-splition

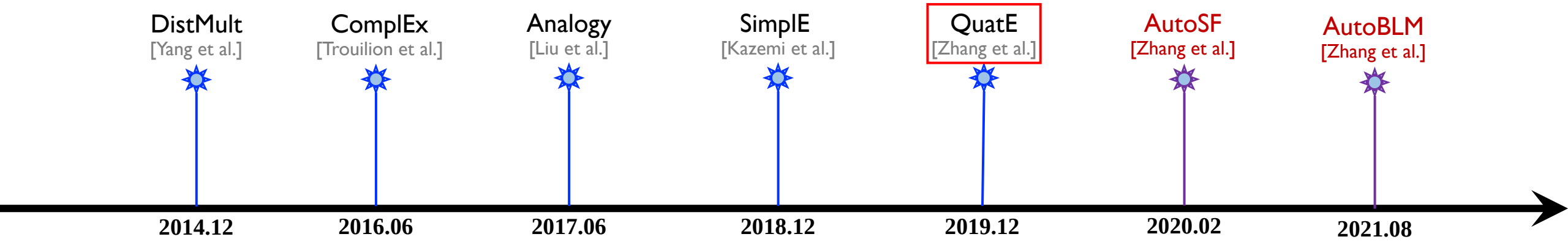


SimplE:

$$f(h,r,t) = \langle \hat{\boldsymbol{h}}, \hat{\boldsymbol{r}}, \underline{\boldsymbol{t}} \rangle + \langle \underline{\boldsymbol{t}}, \underline{\boldsymbol{r}}, \hat{\boldsymbol{h}} \rangle$$
$$= \langle \boldsymbol{h}_1, \boldsymbol{r}_1, \boldsymbol{t}_3 \rangle + \langle \boldsymbol{h}_2, \boldsymbol{r}_2, \boldsymbol{t}_4 \rangle + \langle \boldsymbol{h}_3, \boldsymbol{r}_3, \boldsymbol{t}_1 \rangle + \langle \boldsymbol{h}_4, \boldsymbol{r}_4, \boldsymbol{t}_2 \rangle$$

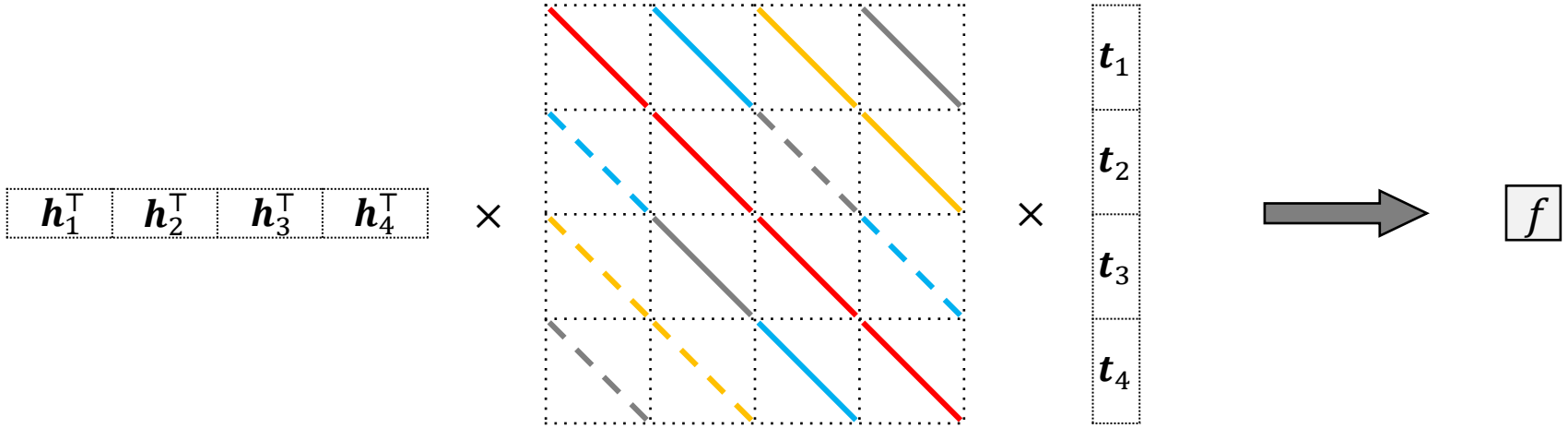


QuatE-splition



QuatE:

$$
\begin{aligned}
f(h,r,t) = &\langle \boldsymbol{h}_1, \boldsymbol{r}_1, \boldsymbol{t}_1 \rangle - \langle \boldsymbol{h}_1, \boldsymbol{r}_2, \boldsymbol{t}_2 \rangle - \langle \boldsymbol{h}_1, \boldsymbol{r}_3, \boldsymbol{t}_3 \rangle - \langle \boldsymbol{h}_1, \boldsymbol{r}_4, \boldsymbol{t}_4 \rangle \\
&+ \langle \boldsymbol{h}_2, \boldsymbol{r}_2, \boldsymbol{t}_1 \rangle + \langle \boldsymbol{h}_2, \boldsymbol{r}_1, \boldsymbol{t}_2 \rangle + \langle \boldsymbol{h}_2, \boldsymbol{r}_4, \boldsymbol{t}_3 \rangle - \langle \boldsymbol{h}_2, \boldsymbol{r}_3, \boldsymbol{t}_4 \rangle \\
&+ \langle \boldsymbol{h}_3, \boldsymbol{r}_3, \boldsymbol{t}_1 \rangle - \langle \boldsymbol{h}_3, \boldsymbol{r}_4, \boldsymbol{t}_2 \rangle + \langle \boldsymbol{h}_3, \boldsymbol{r}_1, \boldsymbol{t}_3 \rangle + \langle \boldsymbol{h}_3, \boldsymbol{r}_2, \boldsymbol{t}_4 \rangle \\
&+ \langle \boldsymbol{h}_4, \boldsymbol{r}_4, \boldsymbol{t}_1 \rangle + \langle \boldsymbol{h}_4, \boldsymbol{r}_3, \boldsymbol{t}_2 \rangle - \langle \boldsymbol{h}_4, \boldsymbol{r}_2, \boldsymbol{t}_3 \rangle + \langle \boldsymbol{h}_4, \boldsymbol{r}_1, \boldsymbol{t}_4 \rangle
\end{aligned}
$$



Outline

- What's Neural Architecture Search (NAS)
- Architectures for Knowledge Graph (KG) Learning
- Searching KG Learning Architectures
  - Search bilinear structure
  - Search recurrent structure
- Summary and Future Works

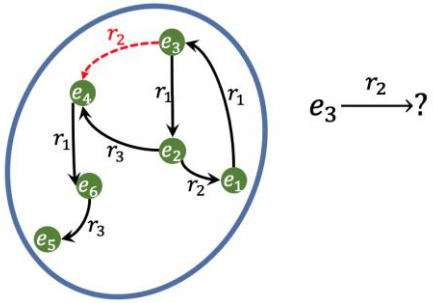
Why AutoML for KGR?

- Various benchmarks

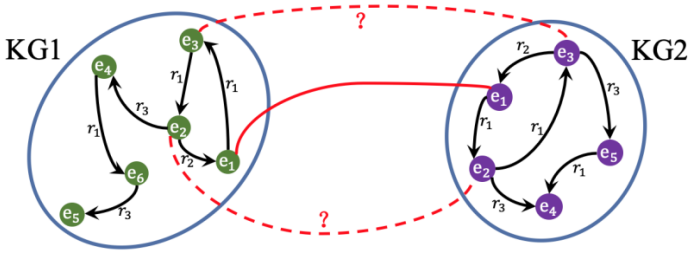
	WN18	FB15k	WN18RR	FB15k-237	YAGO3-10	ogbl-biokg	ogbl-wikikg2
# entities	41k	15k	41k	15k	123k	94k	2,500k
# relations	18	1345	11	237	37	51	535
# triplets	151k	593k	93k	210k	1,080k	5,089k	17,136k

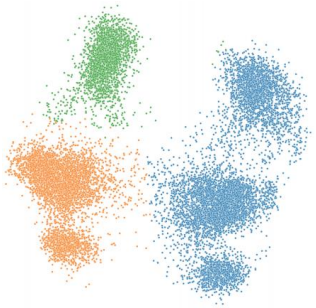
- Different relation patterns:
  - symmetric, anti-symmetric, inverse, asymmetric, composition…

- Different tasks:



$e_3 \xrightarrow{r_2} ?$





AutoML – Recall

1. Define an AutoML problem

- Derive a **search space** from insights in specific domains
- **Search objective** is usually validation performance
- **Search constraint** is usually resource budgets
- **Training objective** usually comes from classical learning models

$$\min_{\lambda \in \mathcal{S}} M(F(w^*; \lambda), D_{\text{val}})$$ ← Search Objective

Search Space →

$$\min_{w} L(F(w; \lambda), D_{\text{tra}})$$ ← Training Objective

s. t.

$$G(\lambda) \leq C$$ ← Search Constraints

2. Design or select proper search algorithm

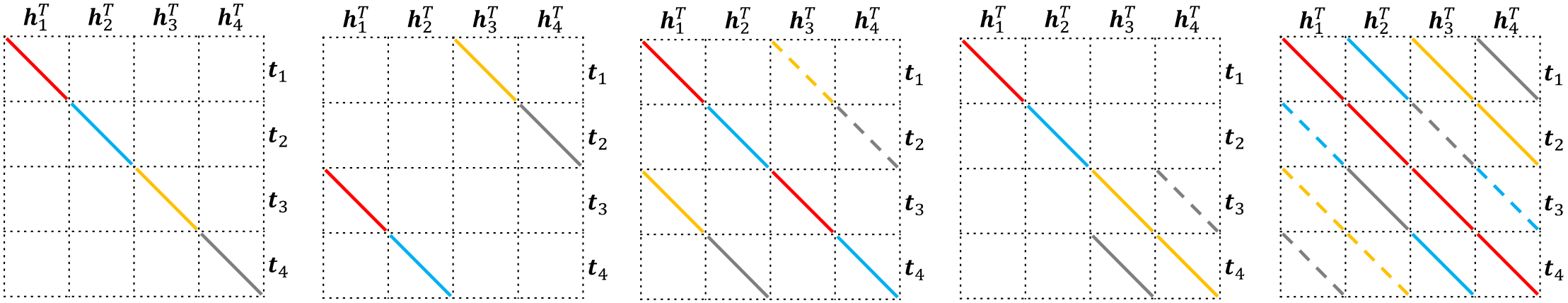
- Reduce model training cost (time to get w^*)

BLMs – unified form

- With unified representation.
- State-of-the-art performance.
- Fully expressive.

- No absolute winner as KGs have different properties.
- How to balance expressiveness and generalization.

AutoBLM: searching!



(a) DistMult. (b) SimplE. (c) ComplEx. (d) Analogy. (e) QuatE.

Y. Zhang, Q. Yao, J. Kwok. Bilinear Scoring Function Search for Knowledge Graph Learning. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI).

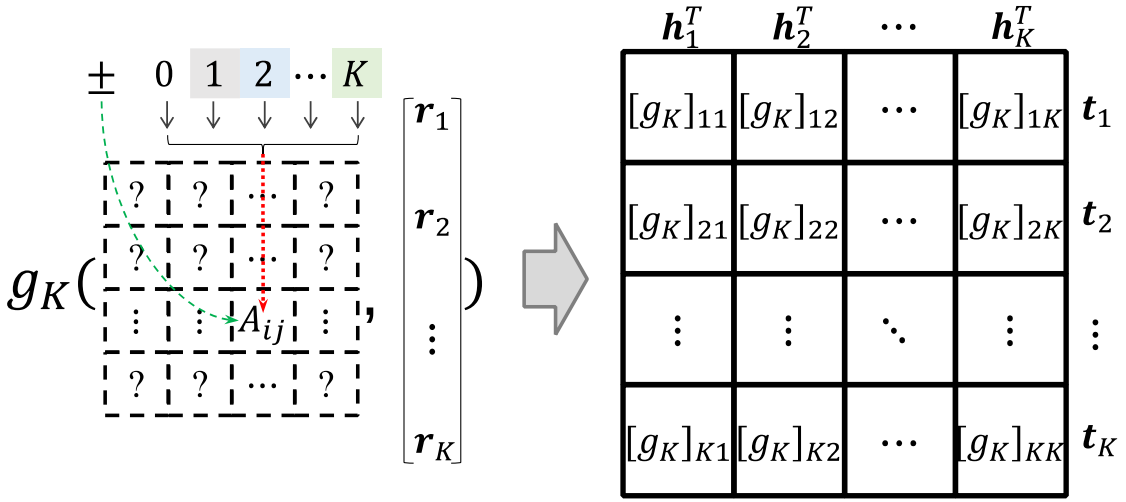
Search problem

Definition 4 (Bilinear Model Search (AutoBLM)). *Let $F(\boldsymbol{P};\boldsymbol{A})$ be a KG embedding model (where $\boldsymbol{P}$ includes the entity embedding matrix $\boldsymbol{E}$ and relation embedding matrix $\boldsymbol{R}$, and $\boldsymbol{A}$ is the structure matrix) and $M(F,\mathcal{S})$ be the performance of F on triples $\mathcal{S}$ (the higher the better). The AutoBLM problem is formulated as:*

$$\boldsymbol{A}^* \in \operatorname{Arg\,max}_{\boldsymbol{A}\in\mathcal{A}} M\left(F(\boldsymbol{P}^*;\boldsymbol{A}),\mathcal{S}_{val}\right) \quad (9)$$

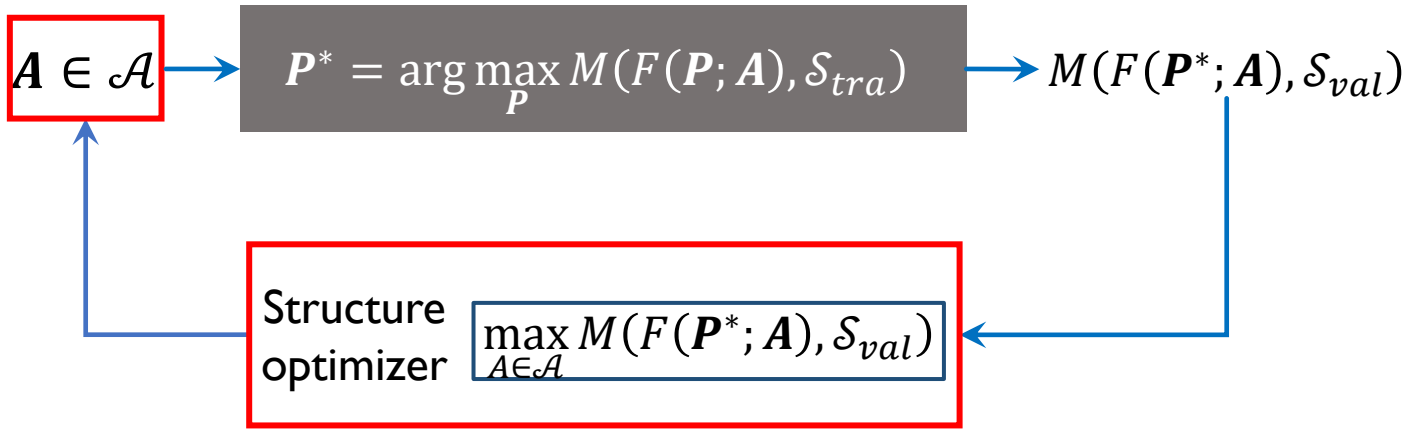
$$s.t. \quad \boldsymbol{P}^* = \arg\max_{\boldsymbol{P}} M\left(F(\boldsymbol{P};\boldsymbol{A}),\mathcal{S}_{tra}\right), \quad (10)$$

$(2K+1)^{K^2}$



Search space:
What to be searched

Validation performance

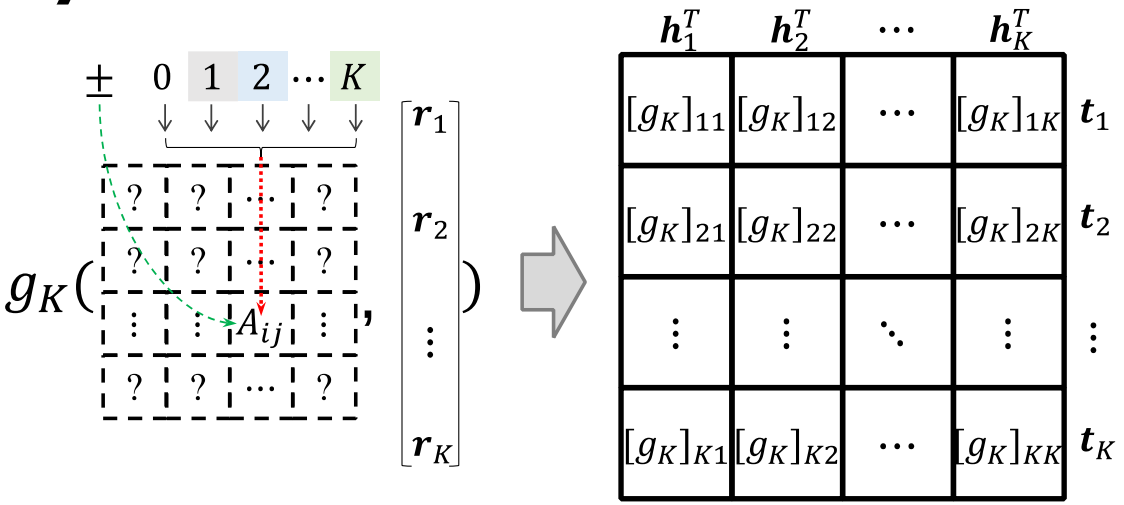


How to search efficiently?

Definition 4 (Bilinear Model Search (AutoBLM)). *Let $F(\boldsymbol{P};\boldsymbol{A})$ be a KG embedding model (where $\boldsymbol{P}$ includes the entity embedding matrix $\boldsymbol{E}$ and relation embedding matrix $\boldsymbol{R}$, and $\boldsymbol{A}$ is the structure matrix) and $M(F,\mathcal{S})$ be the performance of F on triples $\mathcal{S}$ (the higher the better). The AutoBLM problem is formulated as:*

$$\boldsymbol{A}^* \in \operatorname{Arg\,max}_{\boldsymbol{A}\in\mathcal{A}} M\left(F(\boldsymbol{P}^*;\boldsymbol{A}),\mathcal{S}_{val}\right) \quad (9)$$

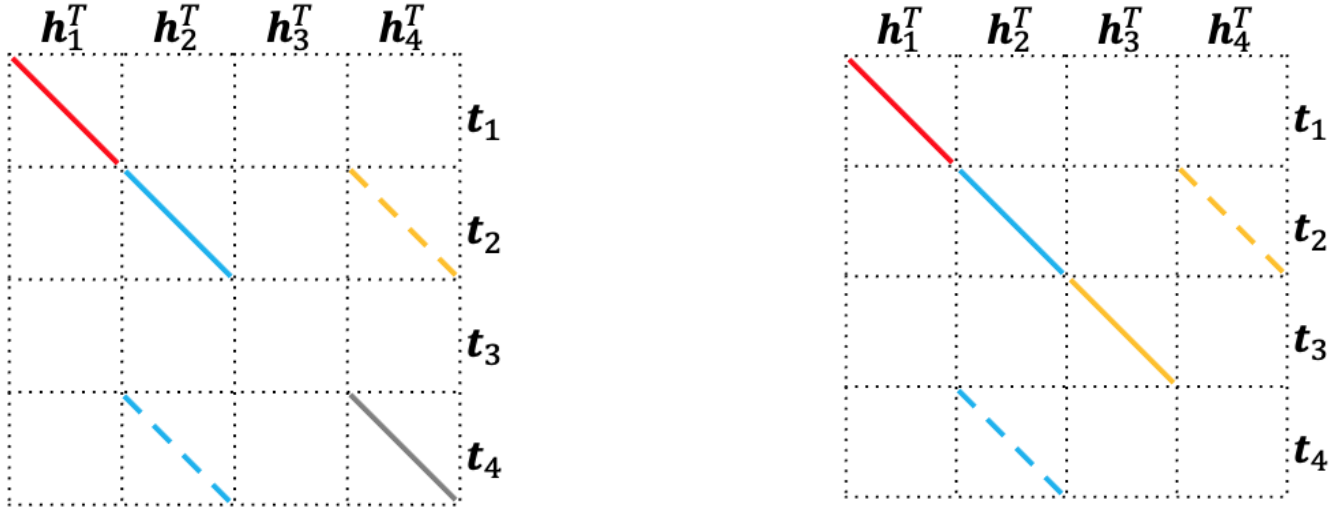
$$s.t. \quad \boldsymbol{P}^* = \arg\max_{\boldsymbol{P}} M\left(F(\boldsymbol{P};\boldsymbol{A}),\mathcal{S}_{tra}\right), \quad (10)$$



- Reduce the size of search space $\mathcal{A}$ --- filter
- Reduce the cost of training $\arg\max M$ --- predictor

Degenerate structures

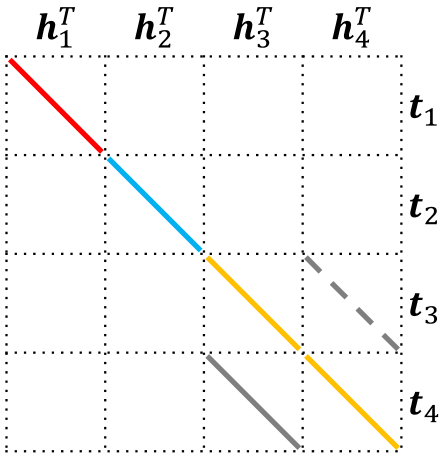
Definition 5 (Degenerate structure). *Matrix* $\boldsymbol{A}$ *is degenerate if (i) there exists* $\boldsymbol{h} \neq \boldsymbol{0}$ *such that* $\boldsymbol{h}^\top g_K(\boldsymbol{A}, \boldsymbol{r})\boldsymbol{t} = 0, \forall \boldsymbol{r}, \boldsymbol{t}$*; or (ii) there exists* $\boldsymbol{r} \neq \boldsymbol{0}$ *such that* $\boldsymbol{h}^\top g_K(\boldsymbol{A}, \boldsymbol{r})\boldsymbol{t} = 0, \forall \boldsymbol{h}, \boldsymbol{t}$.



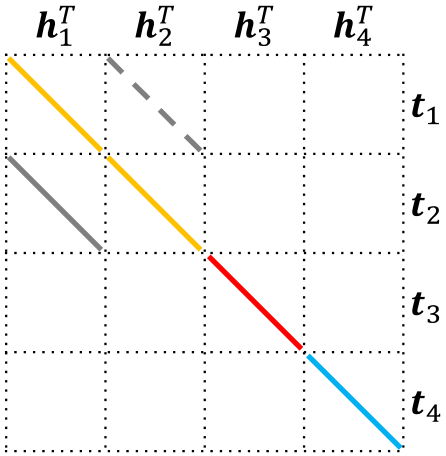
Proposition 2. $\boldsymbol{A}$ *is not degenerate if and only if* $\text{rank}(\boldsymbol{A}) = K$ *and* $\{1, \ldots, K\} \subset \{|A_{ij}| : i, j = 1, \ldots, K\}$.

Equivalent structures

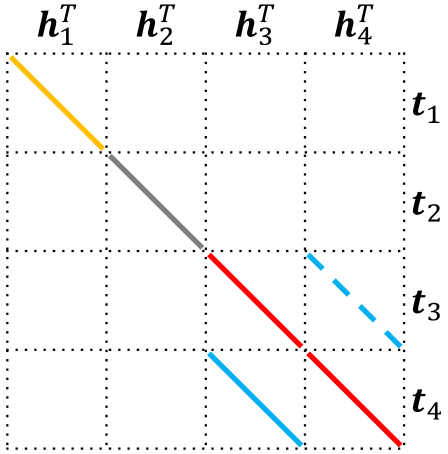
Definition 6 (Equivalence). *Let* $\boldsymbol{P}^* = \arg\max_{\boldsymbol{P}} M(F(\boldsymbol{P}; \boldsymbol{A}), \mathcal{S})$ *and* $\boldsymbol{P}'^* = \arg\max_{\boldsymbol{P}'} M(F(\boldsymbol{P}'; \boldsymbol{A}'), \mathcal{S})$. *If* $\boldsymbol{A} \neq \boldsymbol{A}'$ *but* $M\big(F(\boldsymbol{P}^*; \boldsymbol{A}), \mathcal{S}\big) = M\big(F(\boldsymbol{P}'^*; \boldsymbol{A}'), \mathcal{S}\big)$ *for all* $\mathcal{S}$, *then* $\boldsymbol{A}$ *is equivalent to* $\boldsymbol{A}'$ *(denoted* $\boldsymbol{A} \equiv \boldsymbol{A}'$*).*



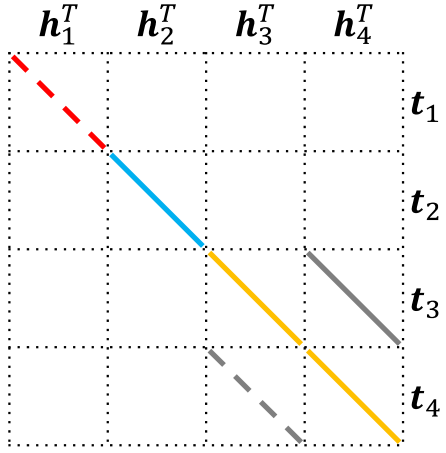
Analogy.



Permuting rows and columns.



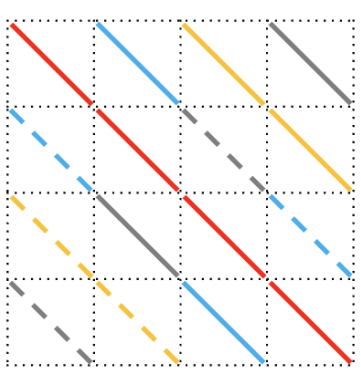
Permuting values.

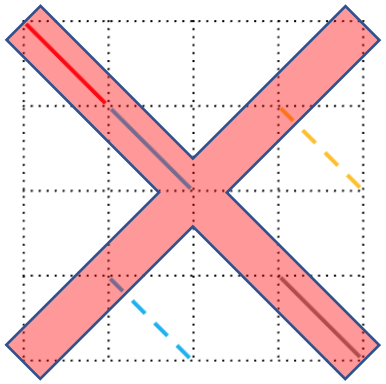


Flipping signs.

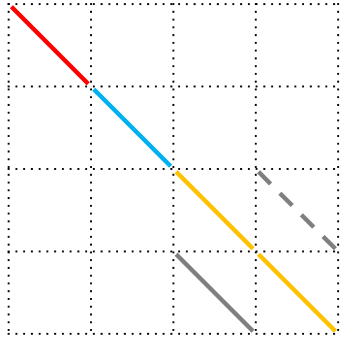
Filter

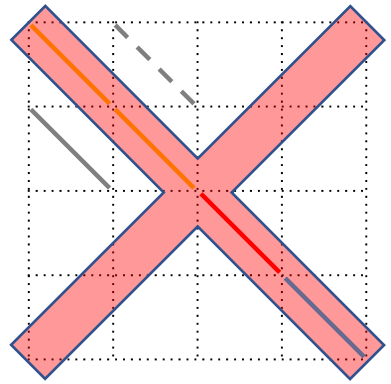
- Given a set of structures, directly remove the degenerate ones and equivalent ones to avoid spending time in evaluating them.





degenerate





equivalent

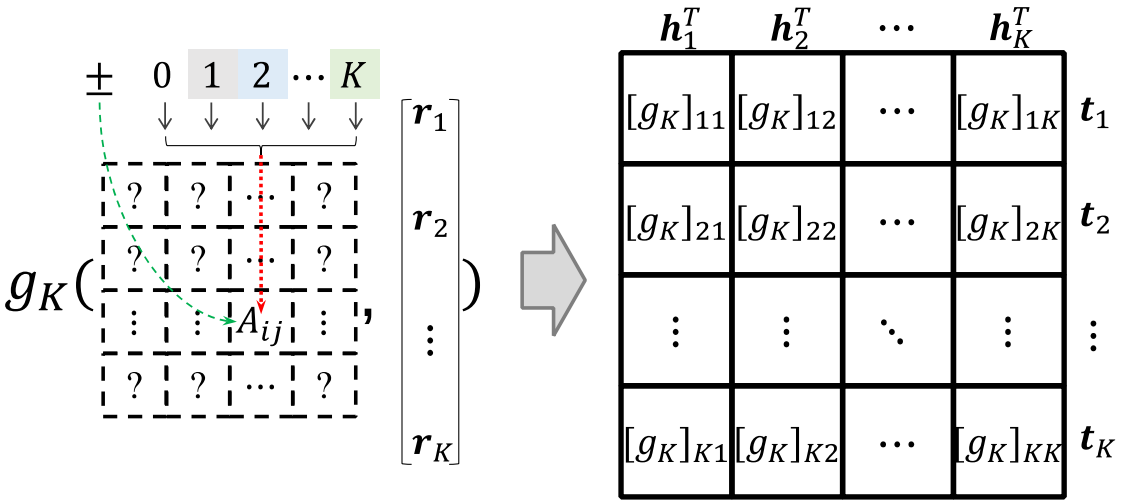
Expressiveness

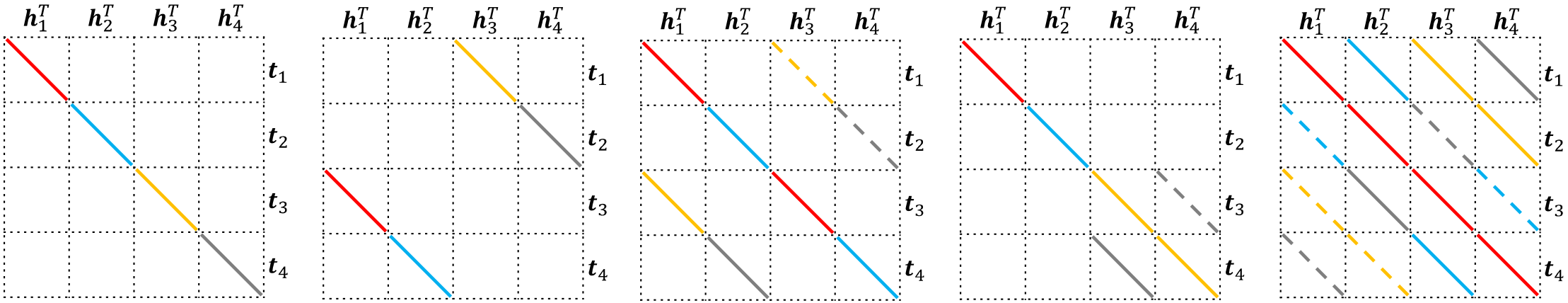
Proposition 1. *Let*

$$\mathcal{C} \equiv \{\boldsymbol{r} \in \mathbb{R}^K \mid \boldsymbol{r} \neq \mathbf{0}, \\ r[i] \in \{0, \pm 1, \ldots, \pm K\}, i = 1, \ldots, K\}. \quad (8)$$

Given an $\boldsymbol{A}$ in (5), the bilinear model with scoring function (7) is fully expressive if

1) $\exists \hat{\boldsymbol{r}} \in \mathcal{C}$ *such that* $g_K(\boldsymbol{A}, \hat{\boldsymbol{r}})$ *is* symmetric *(i.e.,* $g_K(\boldsymbol{A}, \hat{\boldsymbol{r}})^\top = g_K(\boldsymbol{A}, \hat{\boldsymbol{r}})$*), and*
2) $\exists \check{\boldsymbol{r}} \in \mathcal{C}$ *such that* $g_K(\boldsymbol{A}, \check{\boldsymbol{r}})$ *is* skew-symmtric *(i.e.,* $g_K(\boldsymbol{A}, \check{\boldsymbol{r}})^\top = -g_K(\boldsymbol{A}, \check{\boldsymbol{r}})$*).*



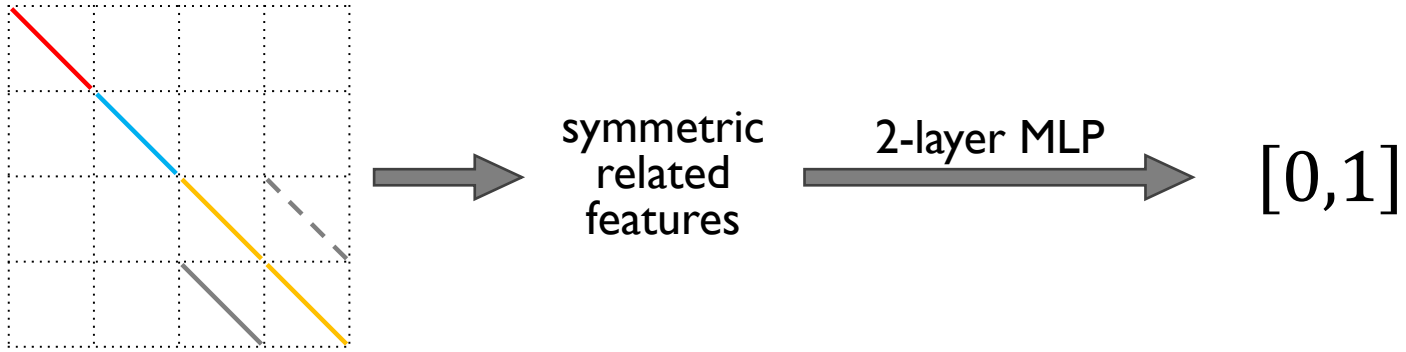


(a) DistMult. (b) SimplE. (c) ComplEx. (d) Analogy. (e) QuatE.

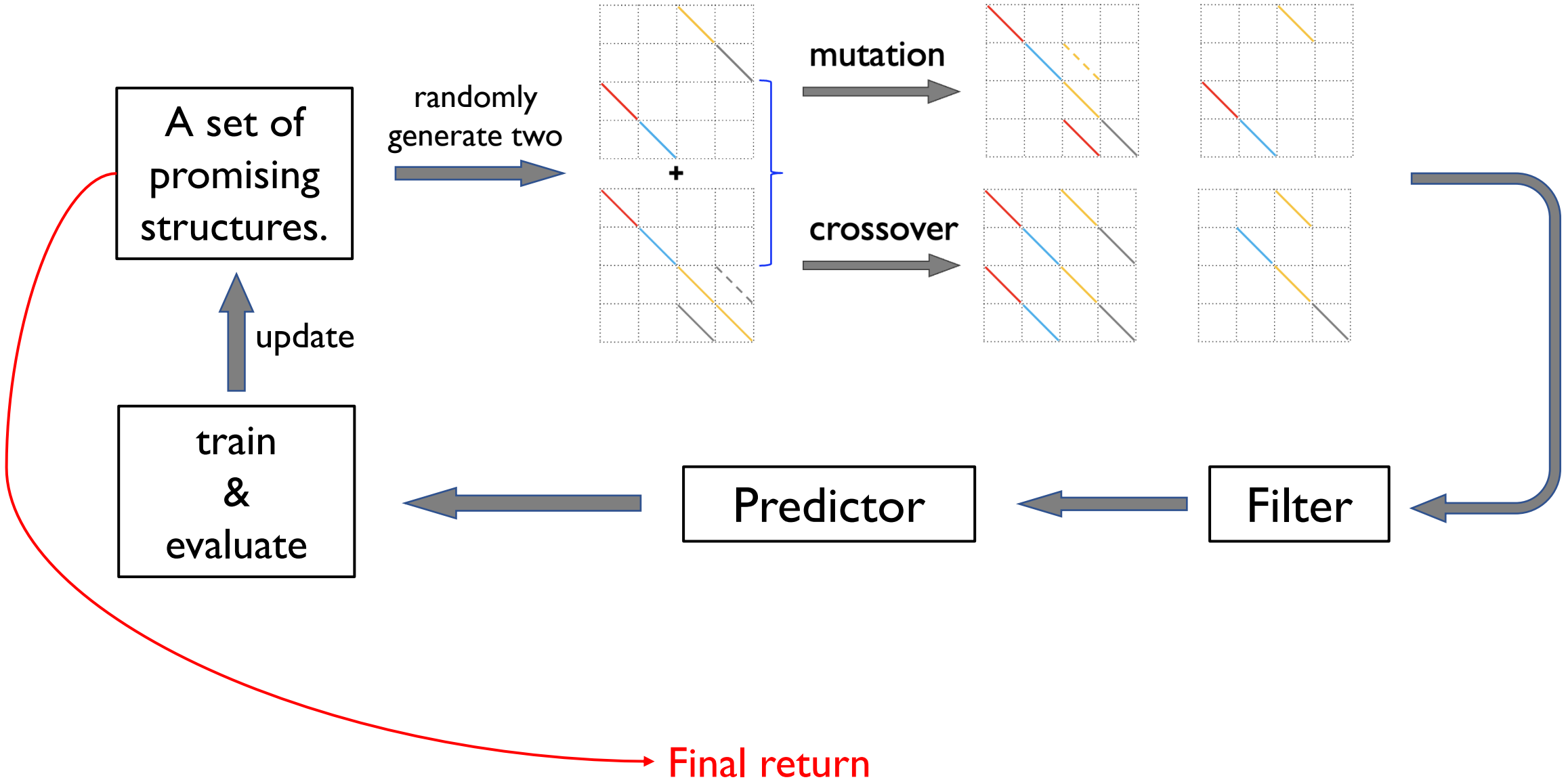
Predictor

- Approximately predict the performance of structures based on the symmetric related properties.

- Estimate the mapping from structures to performance.

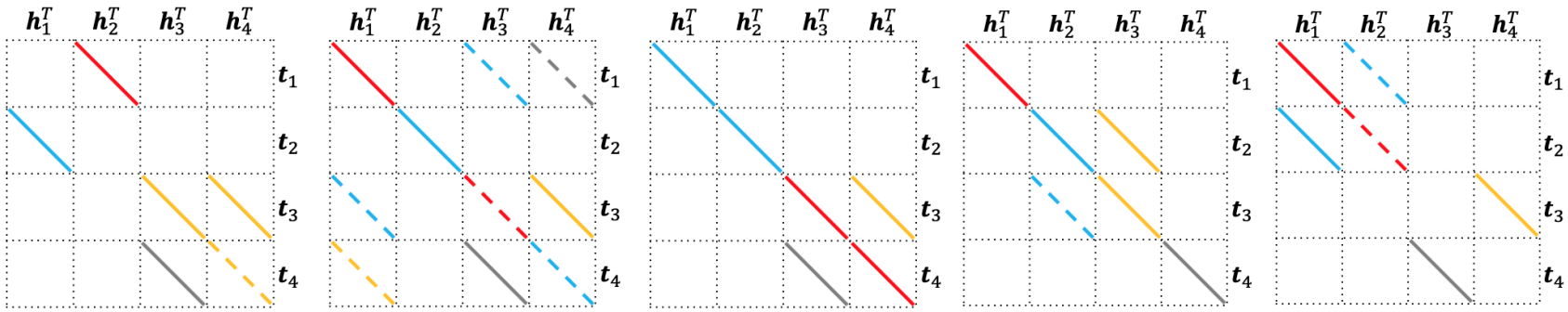


Search algorithms (evolutionary search)



Experiments

	model	WN18			FB15k			WN18RR			FB15k237			YAGO3-10		
		MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
(TDM)	TransH	0.521	—	94.5	0.452	—	76.6	0.186	—	45.1	0.233	—	40.1	—	—	—
	RotatE	—	—	—	0.797	74.6	88.4	0.476	42.8	57.1	0.338	24.1	53.3	0.488	39.6	66.3
	PairRE	—	—	—	0.811	76.5	89.6	—	—	—	0.351	25.6	54.4	—	—	—
(NNM)	ConvE	0.942	93.5	95.5	0.745	67.0	87.3	0.46	39.	48.	0.316	23.9	49.1	0.52	45.	66.
	RSN	0.94	92.2	95.3	—	—	—	—	—	—	0.28	20.2	45.3	—	—	—
	CompGCN	—	—	—	—	—	—	0.479	44.3	54.6	0.355	26.4	53.5	—	—	—
(BLM)	TuckER	**0.953**	**94.9**	95.8	0.795	74.1	89.2	0.470	44.3	52.6	0.358	26.6	54.4	—	—	—
	DistMult	0.821	71.7	95.2	0.775	71.4	87.2	0.443	40.4	50.7	0.352	25.9	54.6	0.552	47.1	68.9
	SimplE/CP	0.950	94.5	95.9	0.826	79.4	90.1	0.462	42.4	55.1	0.350	26.0	54.4	0.565	49.1	71.0
	HolE/ComplEx	0.951	94.5	95.7	0.831	79.6	90.5	0.471	43.0	55.1	0.345	25.3	54.1	0.563	49.0	70.7
	Analogy	0.950	94.6	95.7	0.816	78.0	89.8	0.467	42.9	55.4	0.348	25.6	54.7	0.557	48.5	70.4
	QuatE	0.950	94.5	95.9	0.782	71.1	90.0	0.488	43.8	**58.2**	0.348	24.8	55.0	0.556	47.4	70.4
	AutoSF	0.952	94.7	**96.1**	0.853	82.1	91.0	0.490	45.1	56.7	0.360	26.7	55.2	0.571	50.1	**71.5**
	AutoSF+	0.952	94.7	**96.1**	**0.861**	**83.2**	**91.3**	**0.492**	**45.2**	56.7	**0.364**	**27.0**	**55.3**	**0.577**	**50.2**	**71.5**



(a) WN18. (b) FB15k. (c) WN18RR. (d) FB15k237. (e) YAGO3-10.

Measurements

- Given a triplet (h, r, t);
- Compute the score of $(h', r, t), \forall h' \in \mathcal{E}$;
- Get the rank of h among all h';

Metrics

- MRR: $\frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|}\frac{1}{\text{rank}_i}$
- Hit@k: $\frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|}\mathbb{I}(\text{rank}_i < 10)$

Summary

- Bilinear models are the best choice in triplet-based models.
- AutoBLM: An AutoML approach to design bilinear scoring functions.
- Search space:
  - Unified bilinear models
  - Guaranteed with *expressiveness*
- Search algorithm:
  - *Progressive* and *Evolutionary* algorithm
  - Tackle domain properties by filter and predictor

Outline

- What's Neural Architecture Search (NAS)
- Architectures for Knowledge Graph (KG) Learning
- Searching KG Learning Architectures
  - Search bilinear structure
  - Search recurrent structure
- Summary and Future Works

Relational path

Triplets (s, r, o)

Relational path

consecutively connected

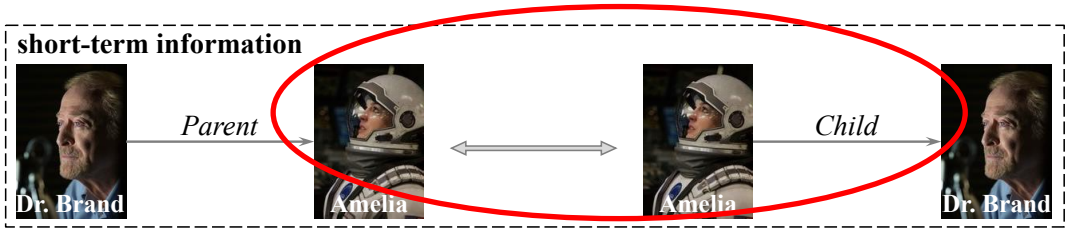
$s_1 \xrightarrow{r_1} s_2 \xrightarrow{r_2} s_3 \cdots s_L \xrightarrow{r_L} s_{L+1}$

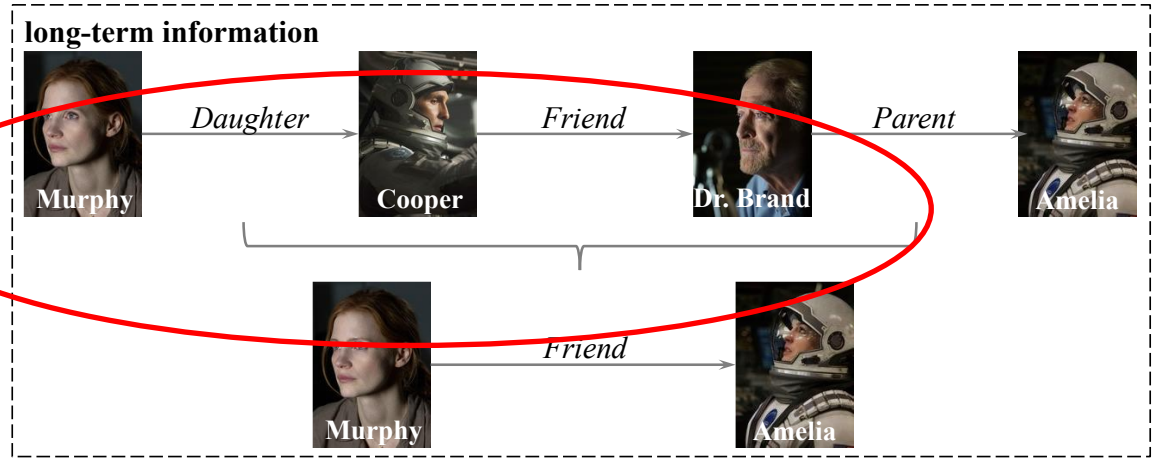
[Guu et al. 2015, Lin et al. 2015, Guo et al. 2019]

short-term information inside triplets.

composition of relations.

long-term information across multiple triplets.





PTransE [Lin et al. 2015]

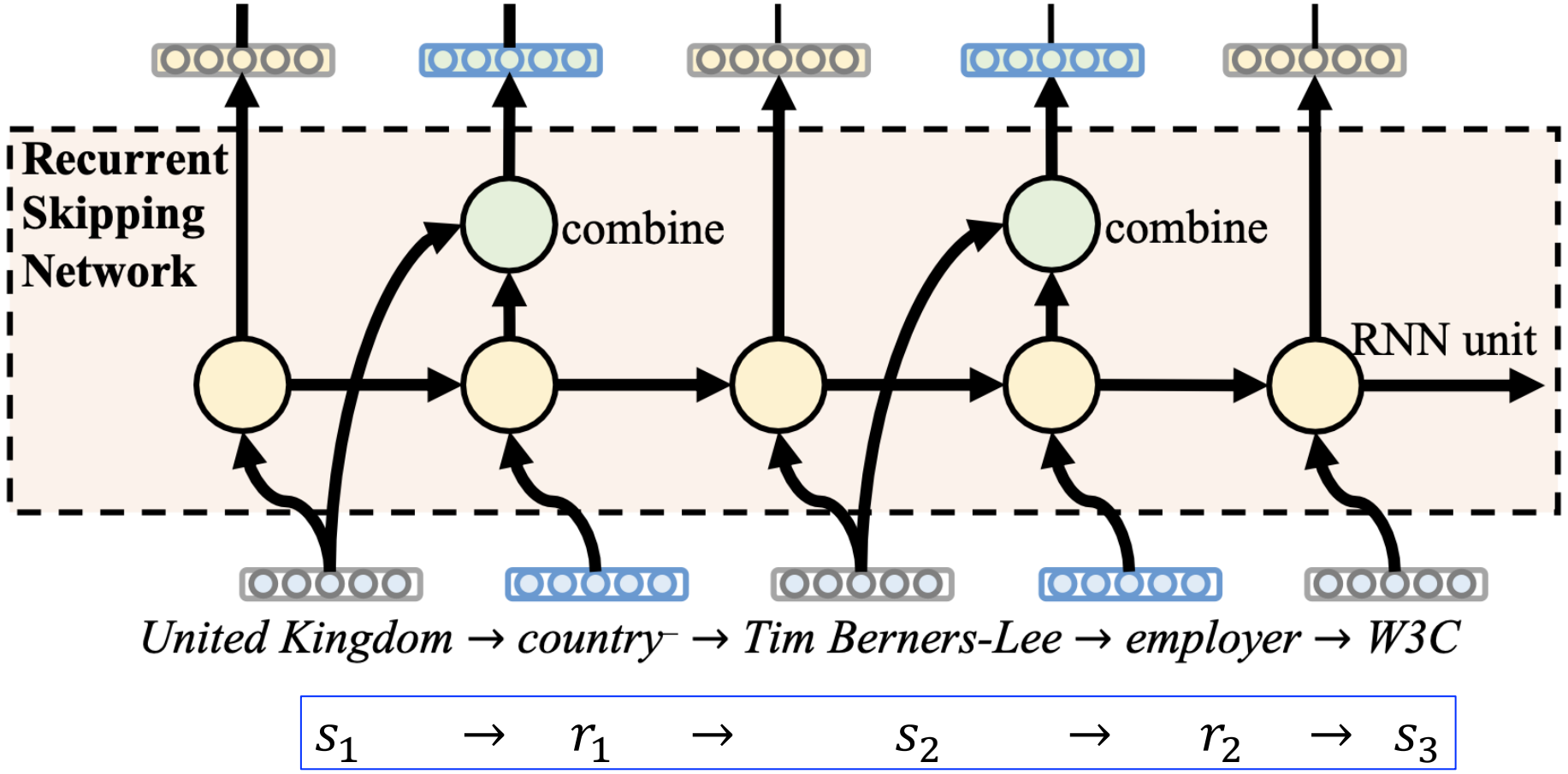
- Models composition of relations

$$s_1 \xrightarrow{r_1} \cdot \xrightarrow{r_2} \cdots \xrightarrow{r_{L-1}} s_L$$

- With condition

$$\boldsymbol{s}_L \approx \boldsymbol{s}_1 + \boldsymbol{r}_1 + \boldsymbol{r}_2 + \cdots \boldsymbol{r}_{L-1}$$

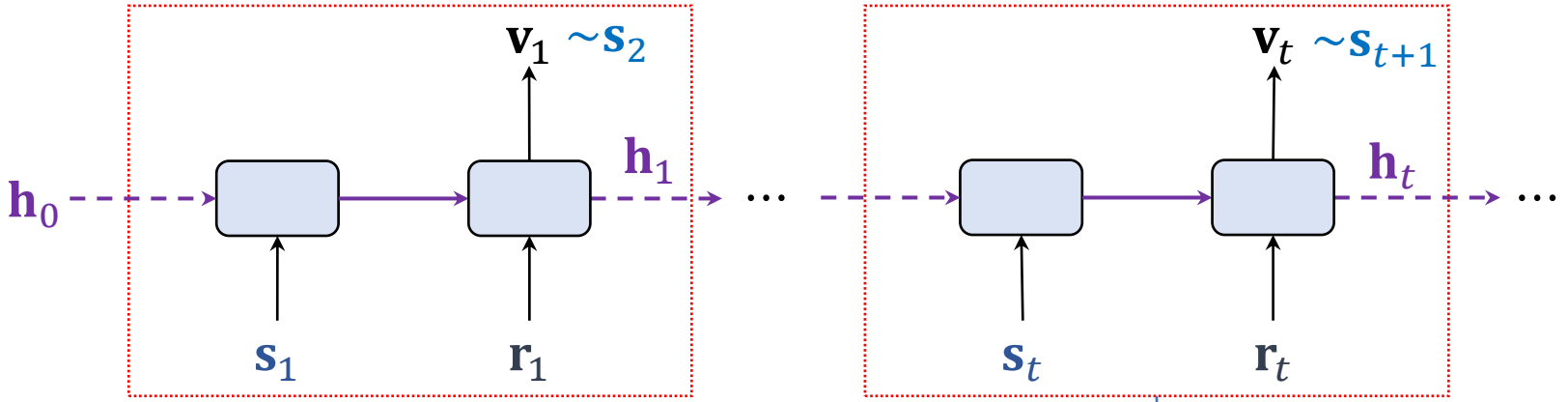
RSN [Guo et al. 2019]

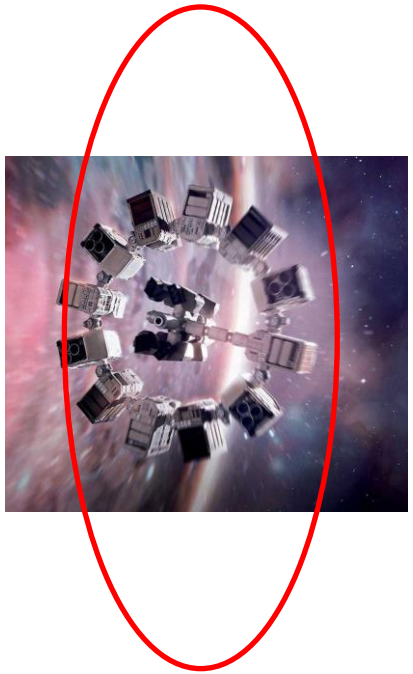


Models long-term information well, but is limited in modeling short-term information.

Interstellar [Zhang et al. 2020]

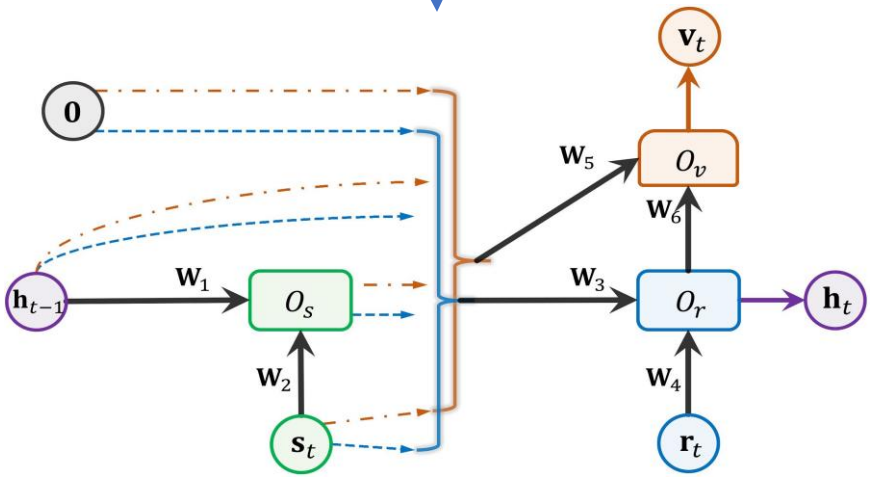
Recurrently process the path by $[\mathbf{v}_t, \mathbf{h}_t] = f(\mathbf{s}_t, \mathbf{r}_t, \mathbf{h}_{t-1}), \quad \forall t = 1 \ldots L$





Searching !

macro-level	connections	$\mathbf{h}_{t-1}, O_s, \mathbf{0}, \mathbf{s}_t$
$\hat{\boldsymbol{\alpha}} \in \hat{\mathcal{A}}$	combinators	$+, \odot, \otimes$, gated
micro-level	activation	identity, tanh, sigmoid
$\check{\boldsymbol{\alpha}} \in \check{\mathcal{A}}$	weight matrix	$\{\mathbf{W}_i\}_{i=1}^{6}, \mathbf{I}$



Hybrid search algorithm

Search appropriate $\alpha \in \mathcal{A}$ that maximize the validation performance

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha} \in \mathcal{A}} \mathcal{M}\left(f(\boldsymbol{F}^*; \boldsymbol{\alpha}), \mathcal{G}_{\text{val}}\right), \quad \text{s.t.} \quad \boldsymbol{F}^* = \arg\min_{\boldsymbol{F}} \mathcal{L}\left(f(\boldsymbol{F}; \boldsymbol{\alpha}), \mathcal{G}_{\text{tra}}\right)$$

Stand-alone approach:

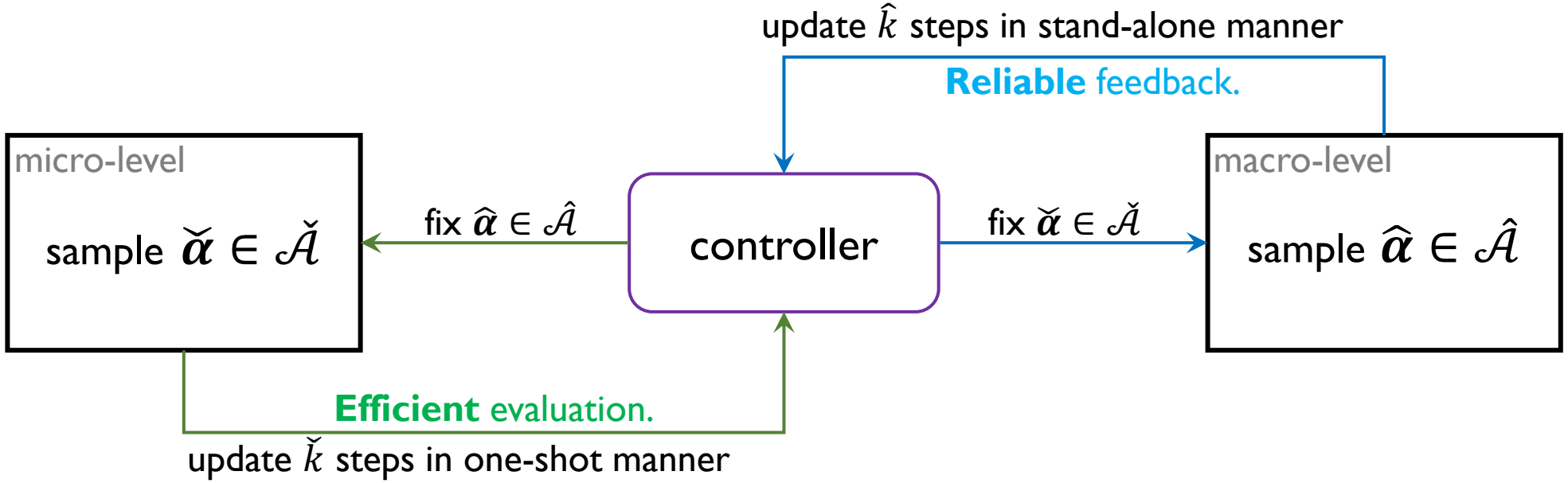
- $\mathcal{M}$ is accurate;
- $\boldsymbol{F}^*$ needs high cost.

[Zoph and Le 2017]

One-shot approach:

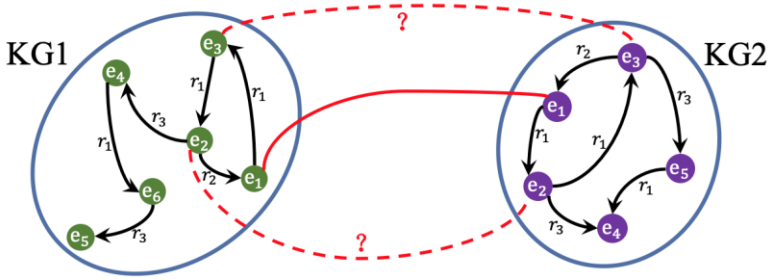
- $\boldsymbol{F}^*$ is shared and efficient;
- $\mathcal{M}$ is not always reliable.

[Pham et al. 2018, Liu et al. 2019]

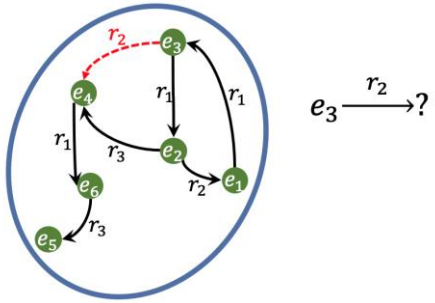


Experiments

	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**



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**



Outline

- What's Neural Architecture Search (NAS)
- Architectures for Knowledge Graph (KG) Learning
- Searching KG Learning Architectures
  - Search bilinear structure
  - Search recurrent structure
- Summary and Future Works

Summary

Search space:

hyper-parameter, scoring function

Search objective:

ranking-based metrics

Training objective:

loss function

Search constraint:

overall time for searching

Search algorithm:

- grid search, random search, greedy algorithm, evolutionary algorithm
- Bayesian optimization, Reinforcement learning, Gradient descent

Future work

- Automated GNN architecture design for KG.
- Efficient reasoning on large-scaled KGs, e.g., ogbl-wikikg.
- Reasoning on dynamic KGs.

Related papers

Knowledge Graph Embedding: A Survey of Approaches and Applications. Q. Wang, Z. Mao, B. Wang and L. Guo. TKDE 2017.

Taking the Human out of Learning Applications: A Survey on Automated Machine Learning. Q. Yao, M. Wang, Y. Chen et. al. 2019.

NSCaching: Simple and Efficient Negative Sampling for Knowledge Graph Embedding. **Y. Zhang**, Q. Yao, Y. Shao and L. Chen. ICDE 2019.

Simple and Automated Negative Sampling for Knowledge Graph Embedding. **Y. Zhang**, Q. Yao and L. Chen. VLDB-J 2021.

AutoSF: Searching Scoring Functions for Knowledge Graph Embedding. **Y. Zhang**, Q. Yao, W. Dai and L. Chen. ICDE 2020.

Interstellar: Searching Recurrent Architecture for Knowledge Graph Embedding. **Y. Zhang**, Q. Yao and L. Chen. NeurIPS 2020 (spotlight).

Efficient Relation-aware Search for Scoring Functions in Knowledge Graph Embedding. S. Di, Q. Yao, **Y. Zhang** and L. Chen. ICDE 2021.

Knowledge Graph Reasoning with Relational Directed Graph. **Y. Zhang**, Q. Yao. WWW 2022

AutoBLM: Bilinear Scoring Function Search for Knowledge Graph Learning. **Y. Zhang**, Q. Yao, Y. Li and J. Kwok. In process.

https://github.com/AutoML-Research

Thank you!