Automated Learning form Graph-Structured Data

Quanming Yao[1,2], Huan Zhao[2], Yongqi Zhang[2]

[1]*Department of Electronic Engineering, Tsinghua University*

[2]*4Paradigm Inc.*

https://quanmingyao.github.io/AutoML.github.io/acml21-tutorial.html



Tutorial Outline

1. What is Automated Machine Learning (AutoML)
   - Background on technical tools from machine learning
2. Automated Graph Neural Networks (GNN)
   - Explore neural architecture search for GNN
3. Automated Knowledge Graph (KG) Embedding
   - Explore AutoML for KG Embedding

Schedule at a Glance

Time	Event
00-45 minutes	**Part 1**: An introduction to Automated Machine Learning (AutoML)
	Speaker: Quanming Yao
45-90 minutes	**Part 2**: Automated Graph Neural Networks (GNN)
	Speaker: Huan Zhao
90-135 minutes	**Part 3**: Automated Knowledge Graph (KG) Embedding
	Speaker: Yongqi Zhang
135-150 minutes	**Part 4**: Discussion

Automated Learning from Graph Structured Data Tutorial

Part 3: Automated Knowledge Graph Embedding

Yongqi Zhang

4Paradigm Inc.

zhangyongqi@4paradigm.com

Outline

1. Background
   - Knowledge Graph (KG)
   - KG embedding
2. Important aspects in KGE
3. Scoring function design
   - Overview
   - AutoSF
4. Summary

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

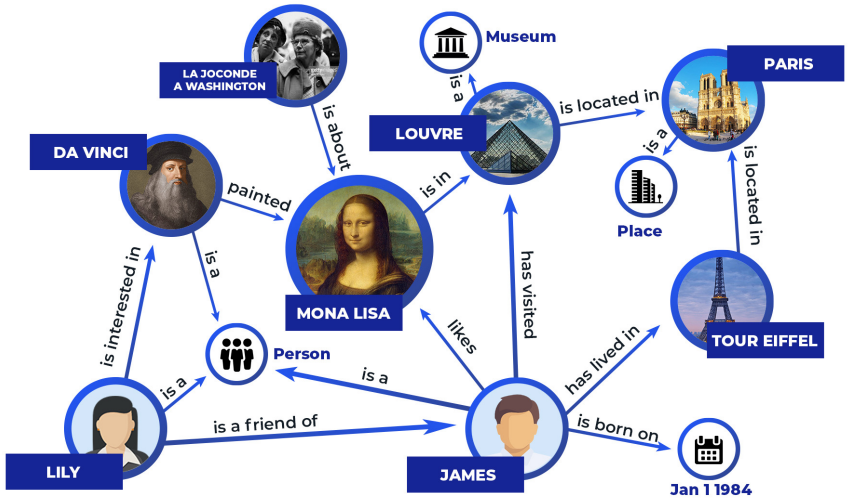
- How does a knowledge graph work?
- Why is knowledge graph important?
- What is Knowledge Graph in SEO?
- How do you make a knowledge graph?

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

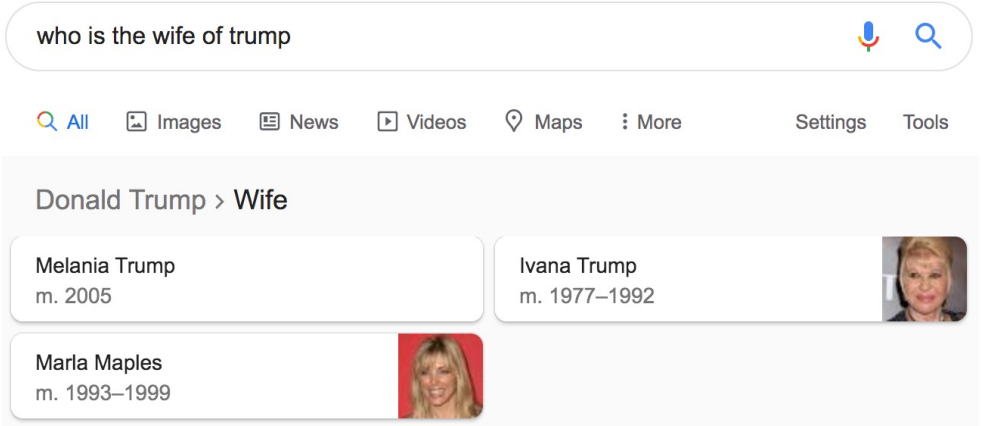
WIKIDATA

Freebase™

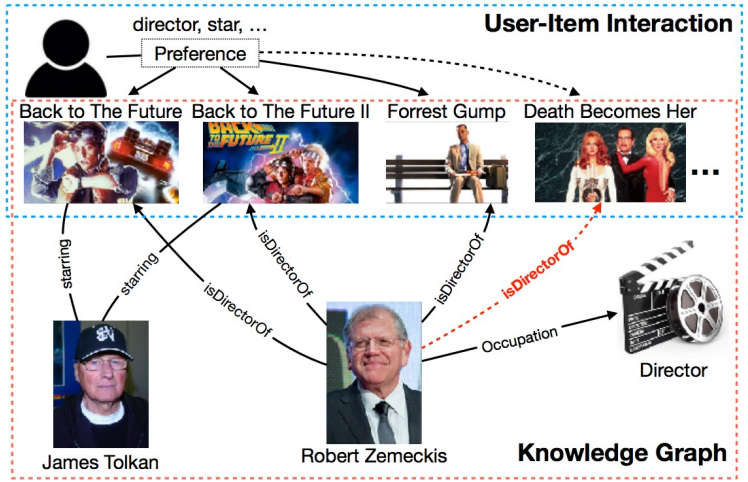
BIO2RDF

KG – important applicaitons

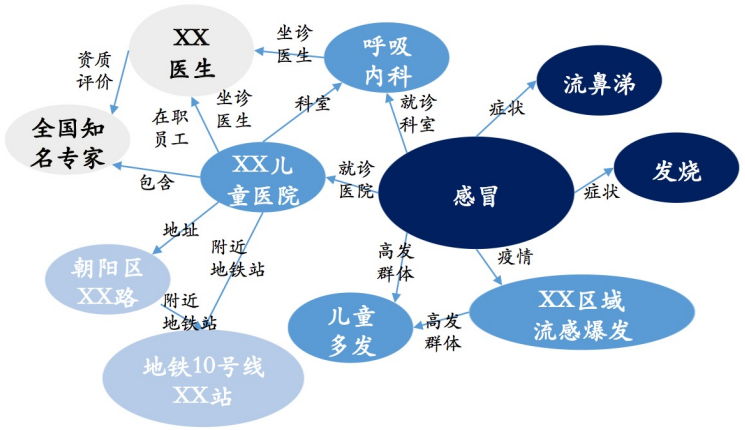
KGQA:



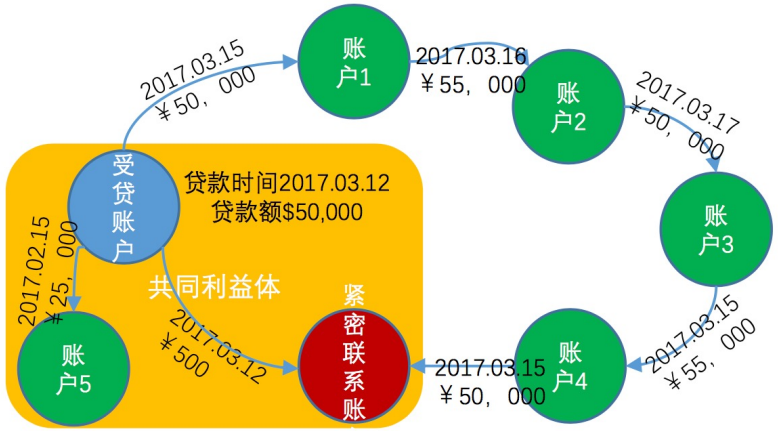
Recommendation:



Medical diagnostic:



Anti-fraud:

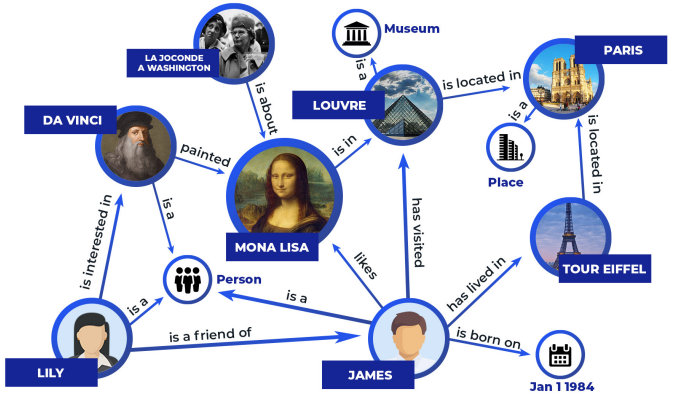


Outline

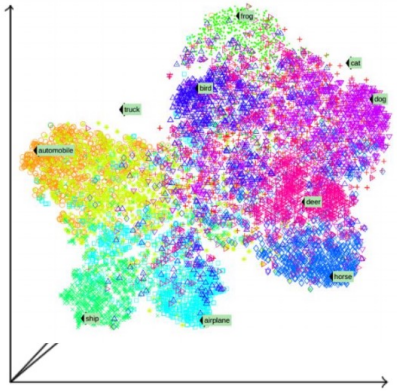
1. Background
   - Knowledge Graph (KG)
   - KG embedding
2. Important aspects in KGE and AutoML
3. Scoring function design
   - Overview
   - AutoSF
4. Summary

Embedding (representation)

Encode entities and relations in KG into low-dimensional vector spaces $\mathbb{R}^{d_1}$ and $\mathbb{R}^{d_2}$, while capturing nodes' and edges' connection properties.



input

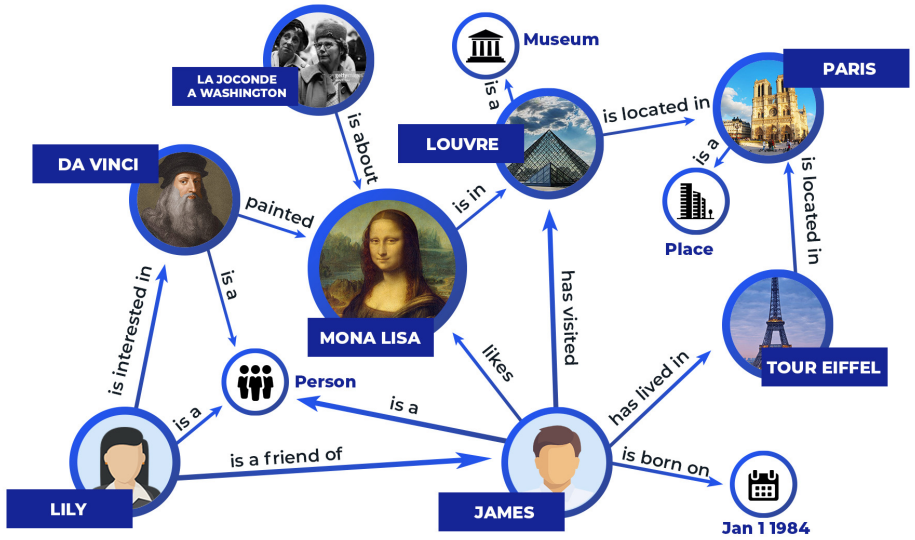


output

Advantages:

- Inject into downstream ML pipelines.
- Provide efficient similarity search.
- Discover latent properties in missing links.

Representation learning

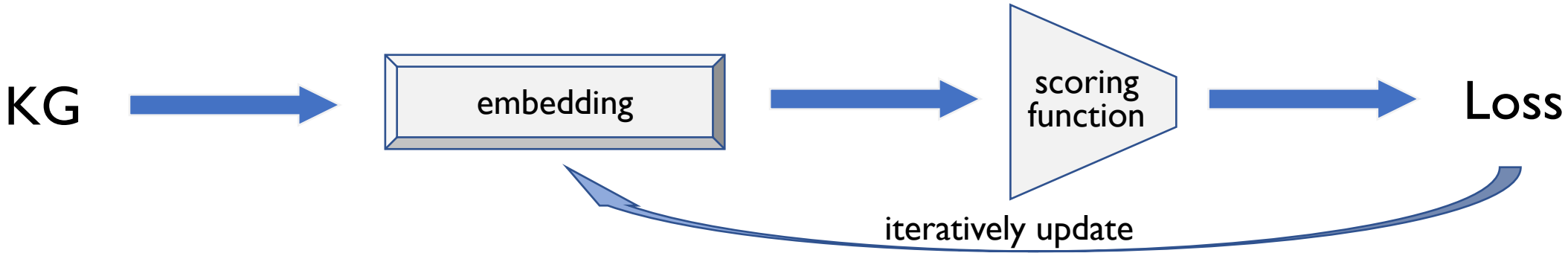


Observed triple S^+:
maximize score

(Lily, isA, Person)
(James, likes, MonaLisa)
(Louver, isLocatedIn, Paris)
(Lily, isInterestedIn, DaVinci)
…

Unobserved triple S^-:
minimize score

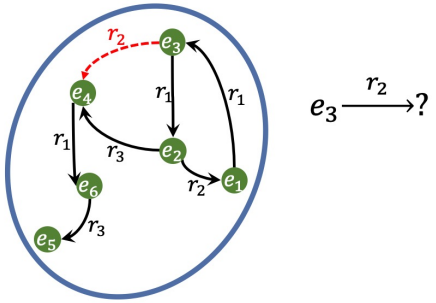
(Lily, isA, Place)
(James, likes, Lily)
(Lily, isLocatedIn, Paris)
(Lily, isA, DaVinci)
…



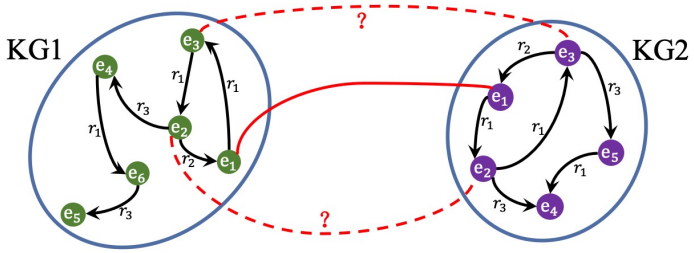
➢Objective: Preserve as much information on original graph as possible.

Important tasks

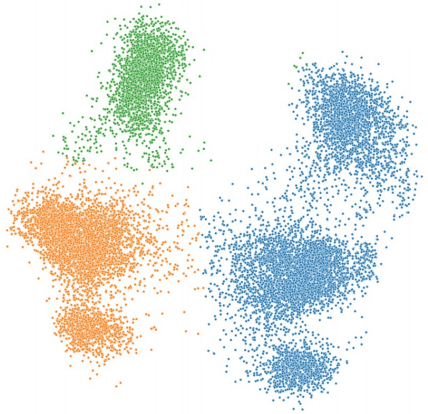
- Knowledge graph completion
  - Link prediction / triple classification: predicting missing links



- 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:

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

Quick summary

- KG provides external information that can improve accuracy, explanation, and alleviate cold start for many applications.

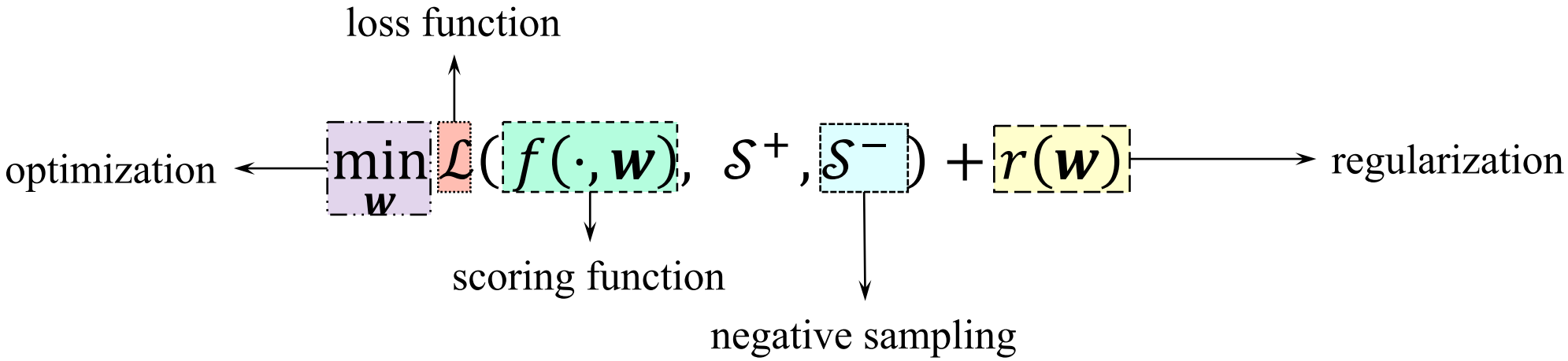
- KG embedding is a strong technique in learning KGs for efficient and accurate prediction.

Outline

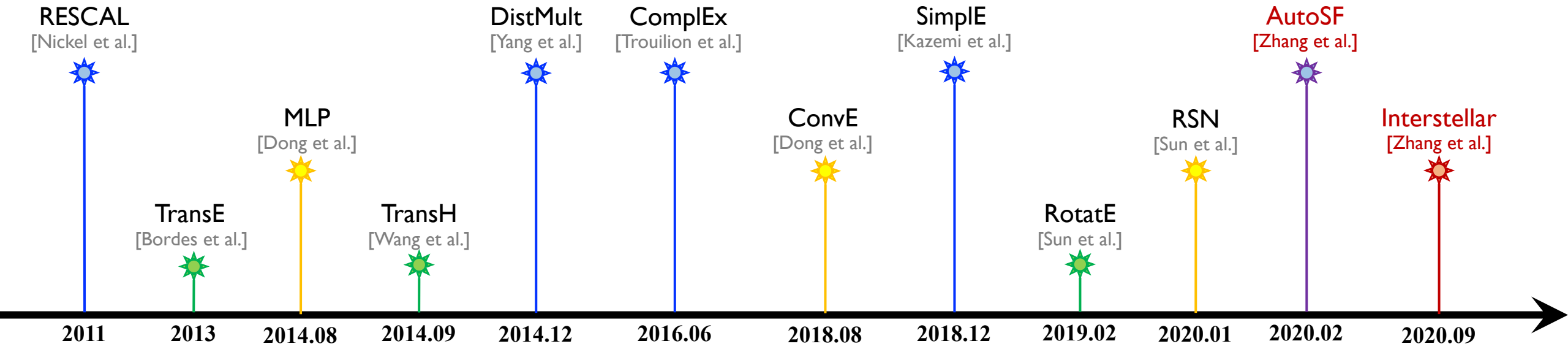
1. Background
   - Knowledge Graph (KG)
   - KG embedding
2. Important aspects in KGE
3. Scoring function design
   - Overview
   - AutoSF
4. Summary

Learning problem in KGE

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



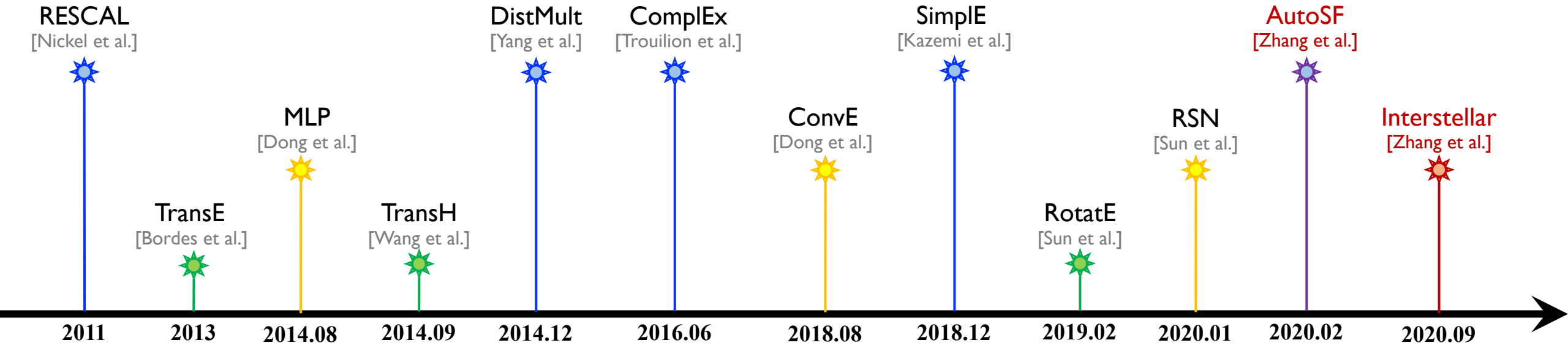
Scoring function



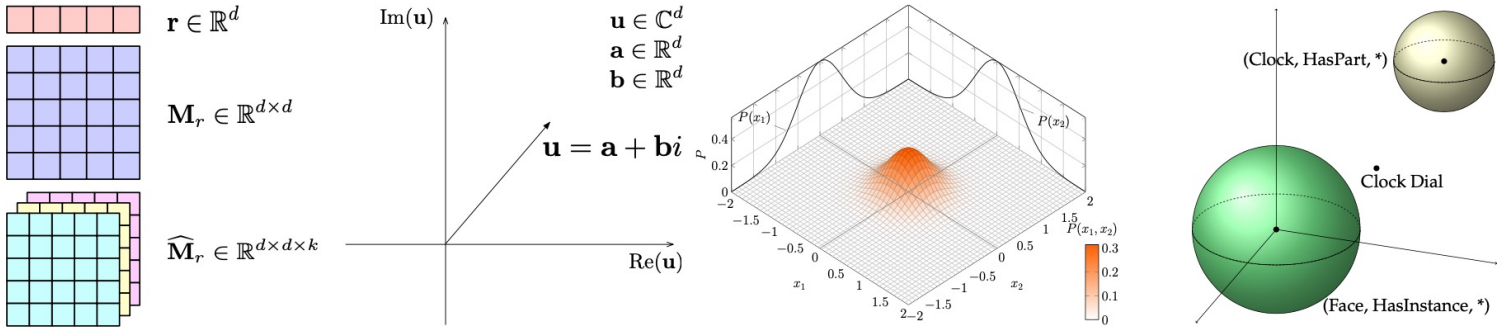
Design patterns:

Symmetric:	<Lily, friendOf, James>
Anti-symmetric:	<Lily, olderThan, Bob>
Inverse:	<Bob, youngerThan, Lily>
Asymmetric:	<Lily, interestedIn, DaVinci>
Composition:	workIn ∧ locatedIn ⟶ liveIn

Scoring function



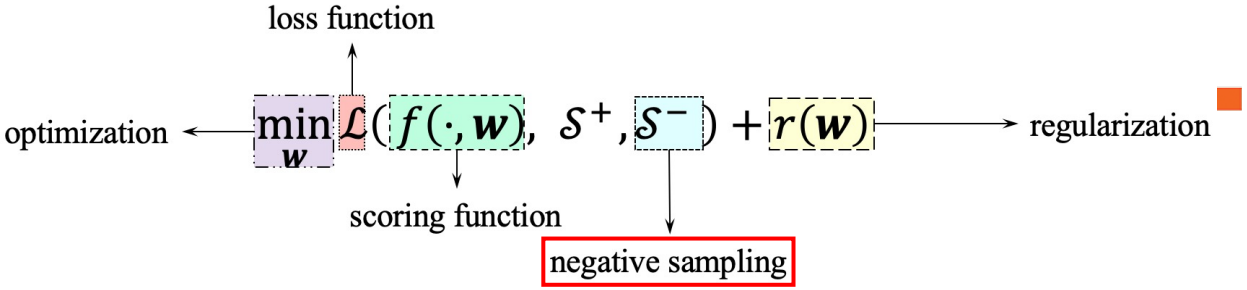
Embedding spaces



(a) Point-wise space. (b) Complex vector space. (c) Gaussian distribution. (d) Manifold space.

[Ji et. al. 2021]

Negative sampling



- KG is sparse, the full set of negative triples is much larger than the set of positive triples.

- Given the positive triple (h, r, t), randomly replacing the head or tail entity from the set

$$\mathcal{S}^-_{(h,r,t)} = \{(h', r, t) \notin \mathcal{S}^+ : h' \in \mathcal{E}\} \cup \{(h, r, t') \notin \mathcal{S}^+ : t' \in \mathcal{E}\}$$

Negative sampling

Standard approaches:

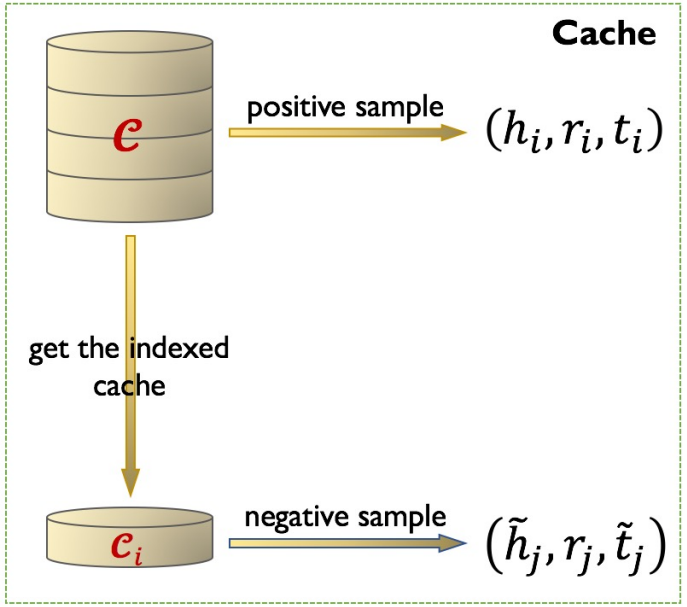
$$(h, r, t)$$

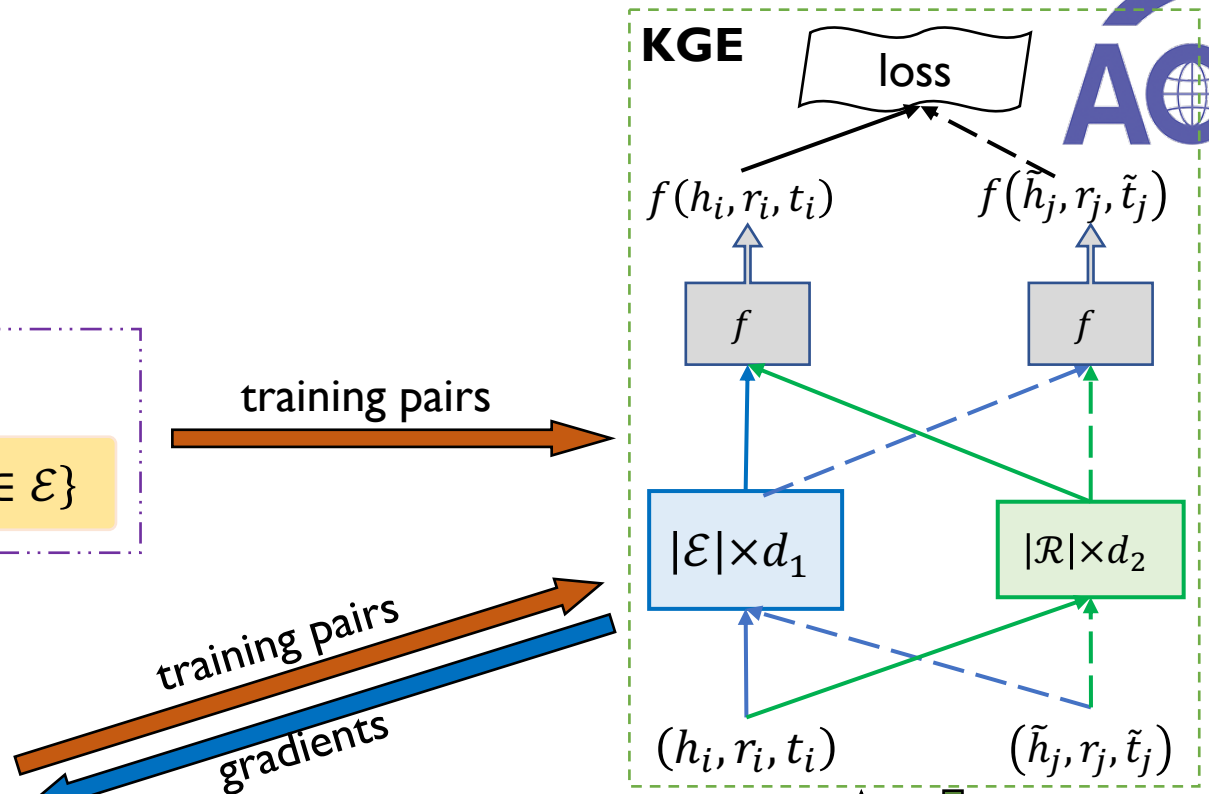
$$\tilde{S}_{(h,r,t)} = \{(\tilde{h}, r, t) \notin S | \tilde{h} \in \mathcal{E}\} \cup \{(h, r, \tilde{t}) \notin S | \tilde{t} \in \mathcal{E}\}$$

not dynamic, quality is poor.

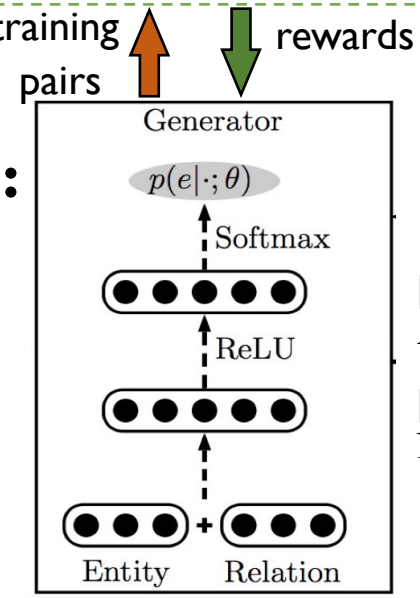
Cache-based:

[Zhang et al. ICDE 2019]





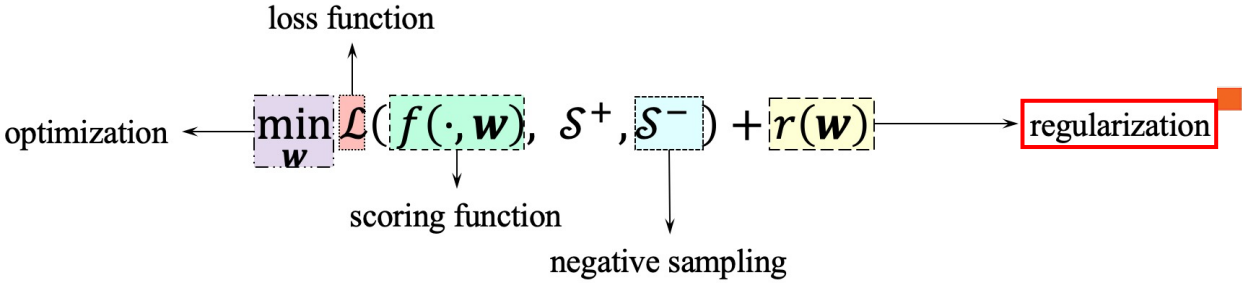
GAN-based:



[Wang et. al. AAAI 2018]

[Cai et. al. NAACL 2018]

Regularization



- The models may overfit on the training data.

expressiveness ⟷ (trade off) complexity

- General approaches:
  - Directly constrain the embedding norm

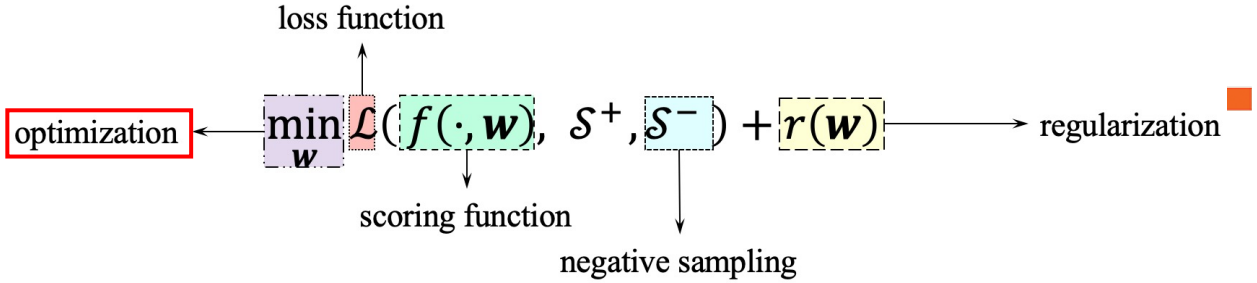
$$\|\boldsymbol{e}\|_p \leq 1, \|\boldsymbol{r}\|_p \leq 1$$

  - Add Lp regularizers

$$r(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|_p^p$$

  - Dropout on the embeddings

Optimization



- Parameters $\boldsymbol{w}$: embeddings, neural networks
- Stochastic gradient descent (Adam, Adagrad) to optimize the parameters

Algorithm 1 Stochastic training of KGE [34].

Require: training set $\mathcal{S} = \{(h, r, t)\}$, scoring function f and loss function ℓ;

initialize the embedding parameters $\mathbf{e}, \mathbf{r}$ for each $e \in \mathcal{E}$ and $r \in \mathcal{R}$.
for $i = 1, \cdots, T$ **do**
  sample a mini-batch $\mathcal{S}_{\text{batch}} \in \mathcal{S}$ of size m;
  for each $(h, r, t) \in \mathcal{S}_{\text{batch}}$ **do**
    sample $\tilde{m}$ negative triplets $\tilde{\mathcal{S}}_{(h,r,t)} \equiv \{(\tilde{h}_j, r, \tilde{t}_j)\}$ for the positive triplet (h, r, t);
    compute the scores of positive triplet (h, r, t) and negative triplets $\{(\tilde{h}_j, r, \tilde{t}_j)\}$ based on the scoring function f;
    update embeddings based on loss ℓ using selected positive and negative triplets;
  end for
end for
return embeddings of entities in $\mathcal{E}$ and relations in $\mathcal{R}$.

Learning rate, initializer, batch size, dimension size, early stop…

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 →

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

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

2. Design or select proper search algorithm

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

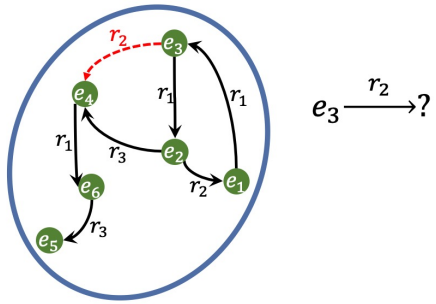
Why AutoML for KGE?

- Various benchmarks

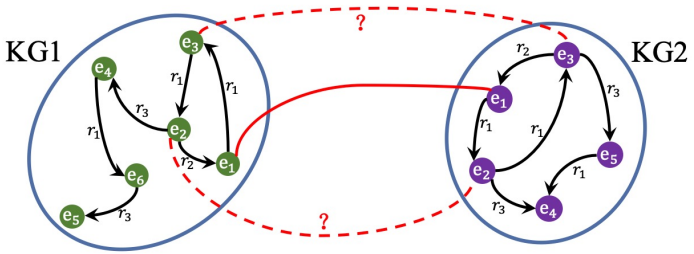
	WN18	FB15k	WN18RR	FB15k-237	YAGO3-10	ogbl-biokg	ogbl-wikikg2
# entities	41k	15k	41k	15k	123k	94k	2,500k
# relations	18	1345	11	237	37	51	535
# triplets	151k	593k	93k	210k	1,080k	5,089k	17,136k

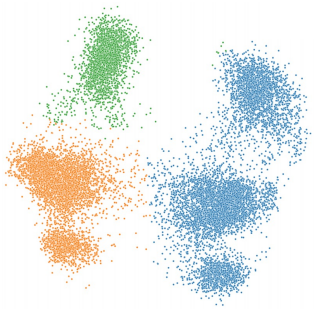
- Different relation patterns:
  - symmetric, anti-symmetric, inverse, asymmetric, composition…

- Different tasks:



$e_3 \xrightarrow{r_2} ?$





AutoML – KGE

Search space:

hyper-parameter, scoring function

Search objective:

ranking-based metrics

Training objective:

loss function

Search constraint:

overall time for searching

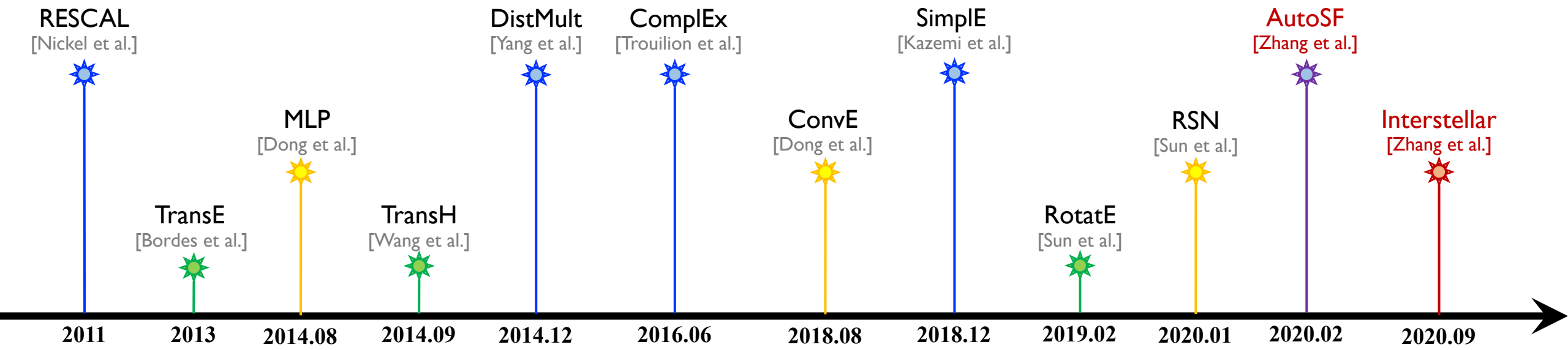
Search algorithm:

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

Outline

1. Background
   - Knowledge Graph (KG)
   - KG embedding
2. Important aspects in KGE
3. Scoring function design
   - Overview
   - AutoSF
4. Summary

Scoring function

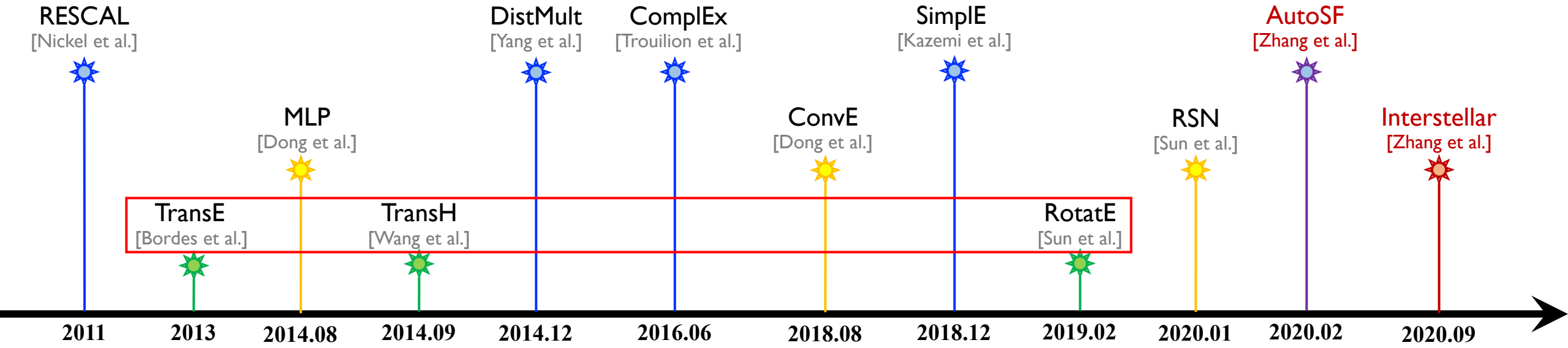


- A large amount of scoring functions (SFs) $f(h, r, t)$ are defined to measure the plausibility of triplets $\{(h, r, t)\}$ in KG.
- Capture important properties:
  - symmetric, anti-symmetric, inverse, asymmetric…
- No one model can perform good on all datasets.

Outline

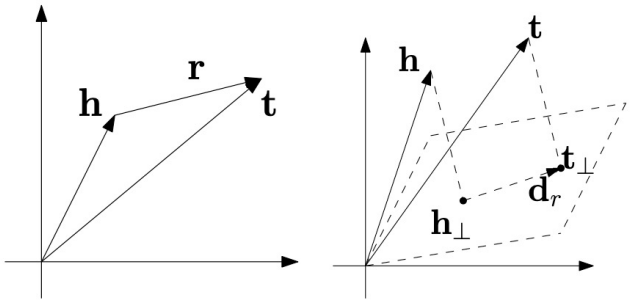
1. Background
   - Knowledge Graph (KG)
   - KG embedding
2. Important aspects in KGE
3. Scoring function design
   - Overview
   - AutoSF
4. Summary

Translational Distance Model (TDM)



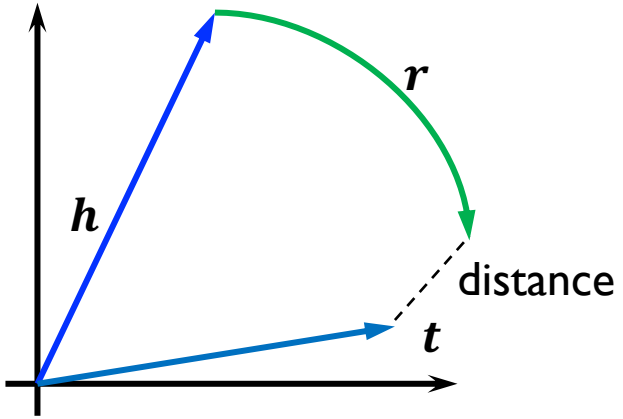
TransE: $f(h,r,t) = -\|(\boldsymbol{h}+\boldsymbol{r})-\boldsymbol{t}\|_p$

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



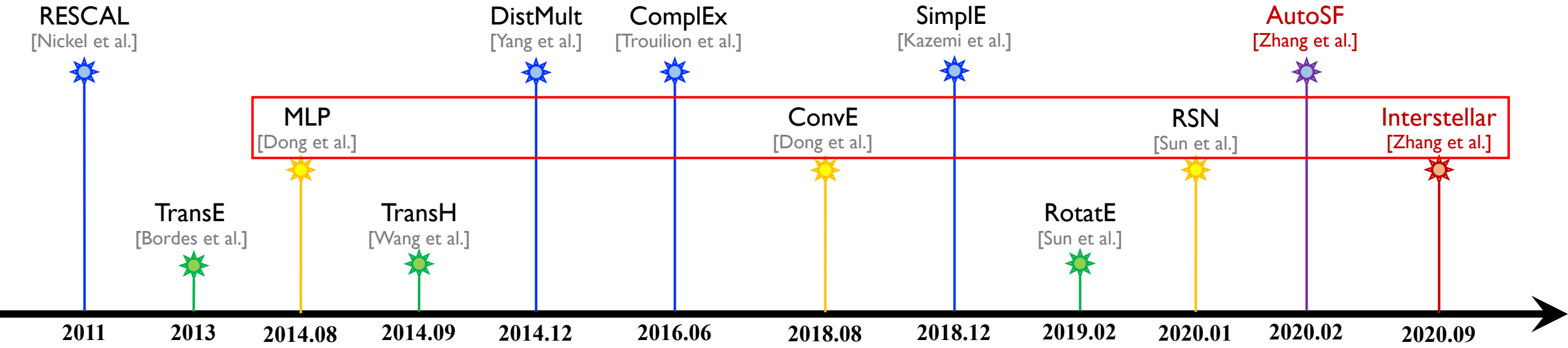
(a) TransE (b) TransH

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

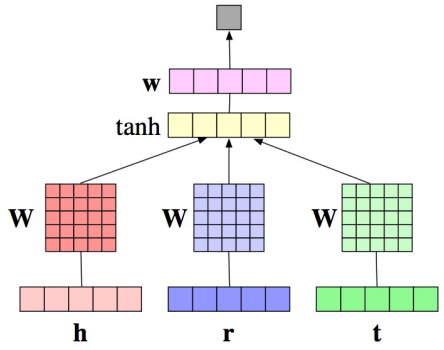


less expressive

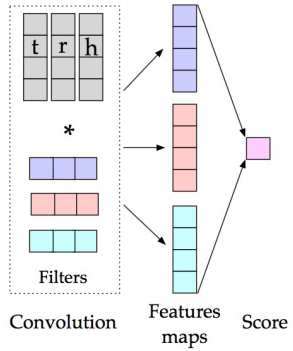
Neural Network Model (NNM)



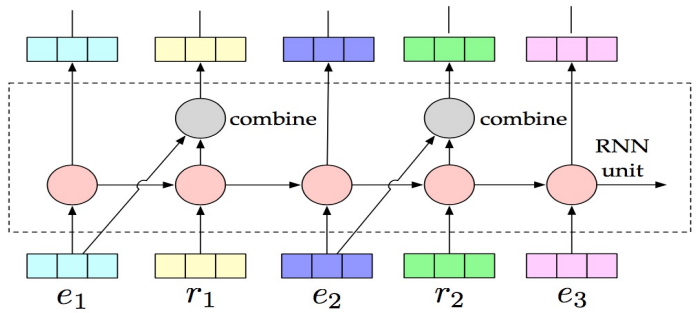
MLP



ConvE

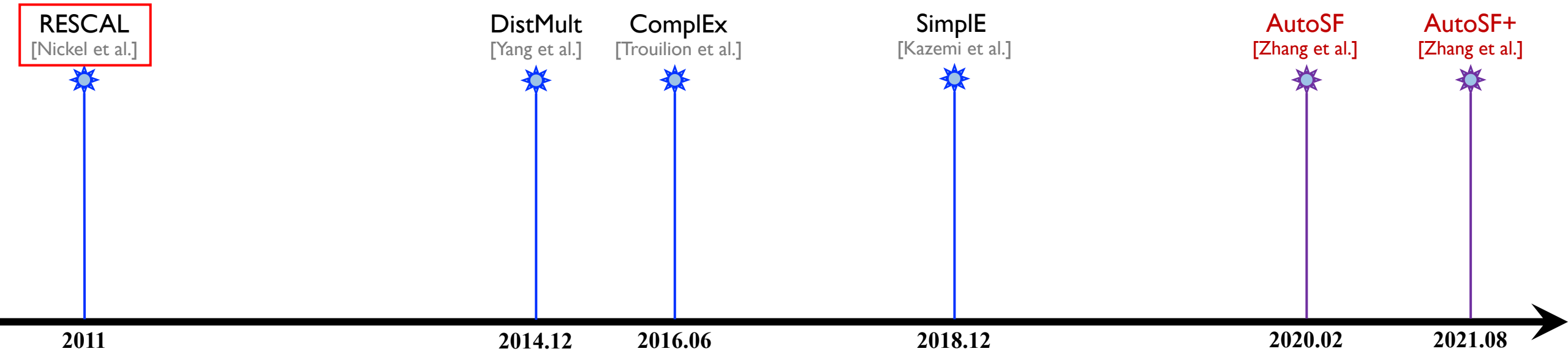


RSN

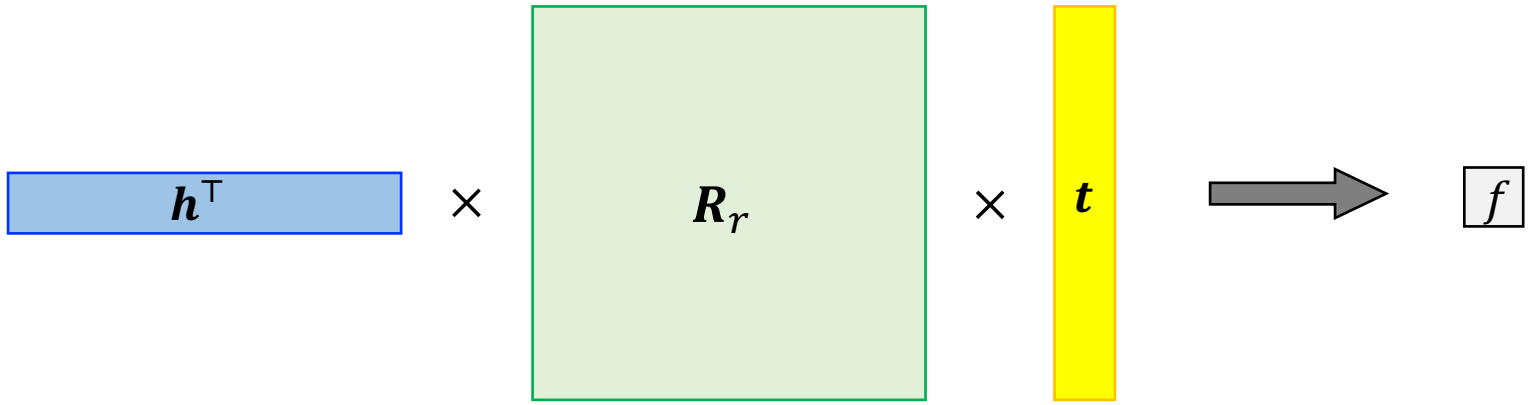


complex
and
difficult to train

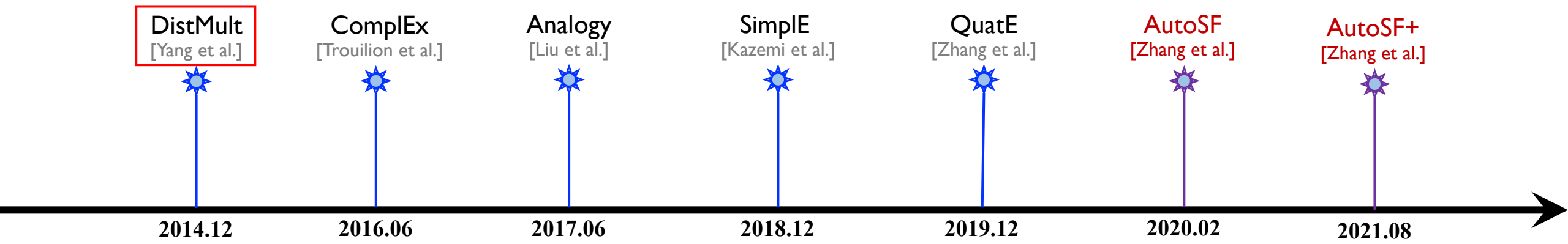
BiLinear Model (BLM)



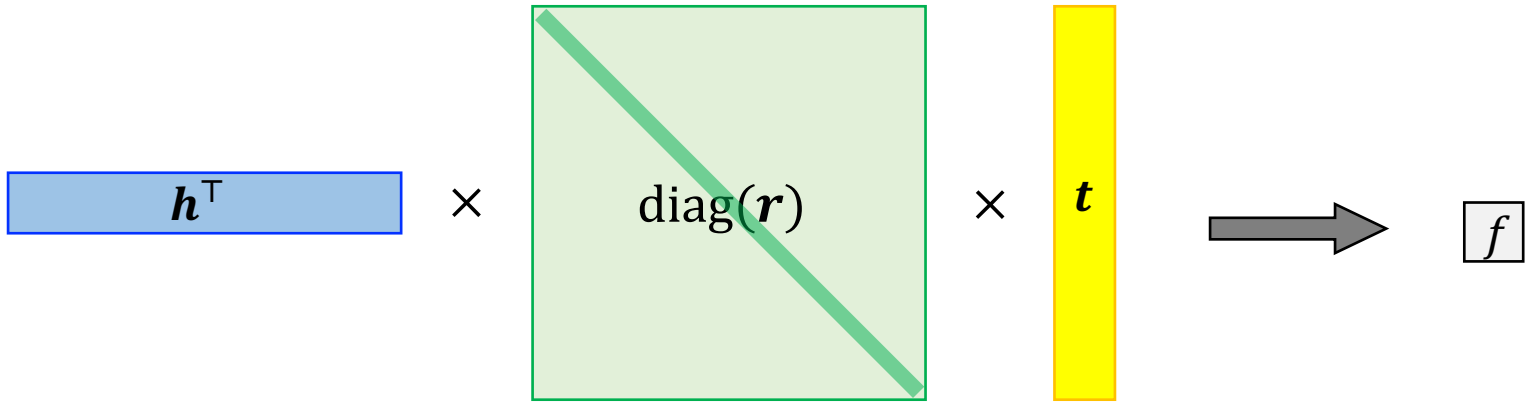
RESCAL: $f(h, r, t) = \boldsymbol{h}^\top \boldsymbol{R}_r \boldsymbol{t}$



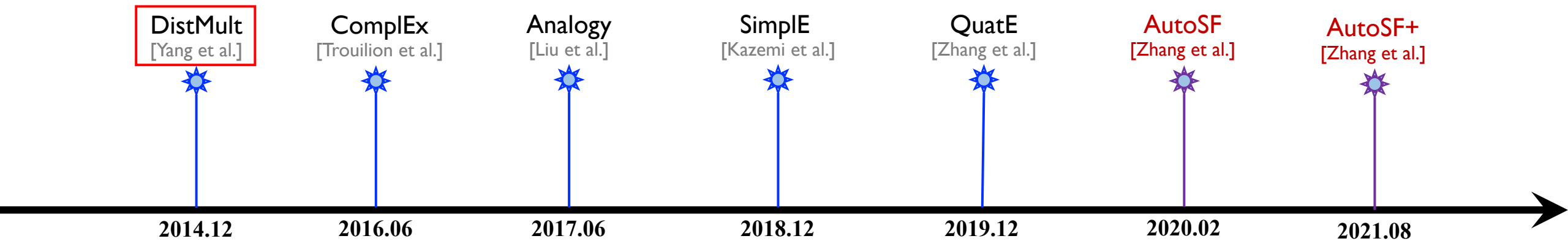
DistMult



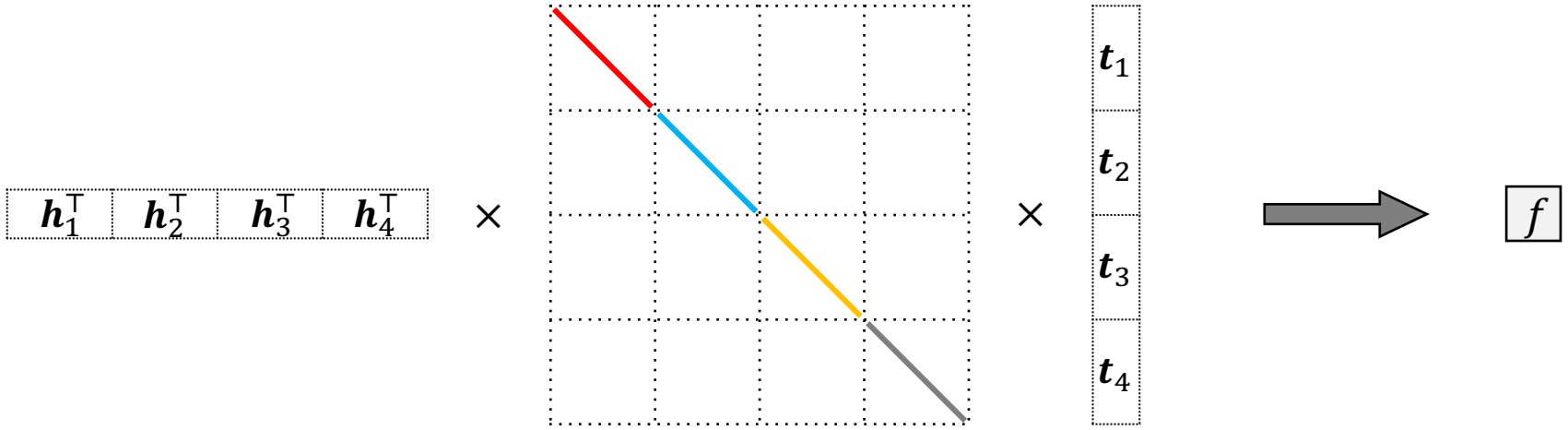
DistMult: $f(h, r, t) = \boldsymbol{h}^\top \mathrm{diag}(\boldsymbol{r})\boldsymbol{t} = \langle \boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \rangle$



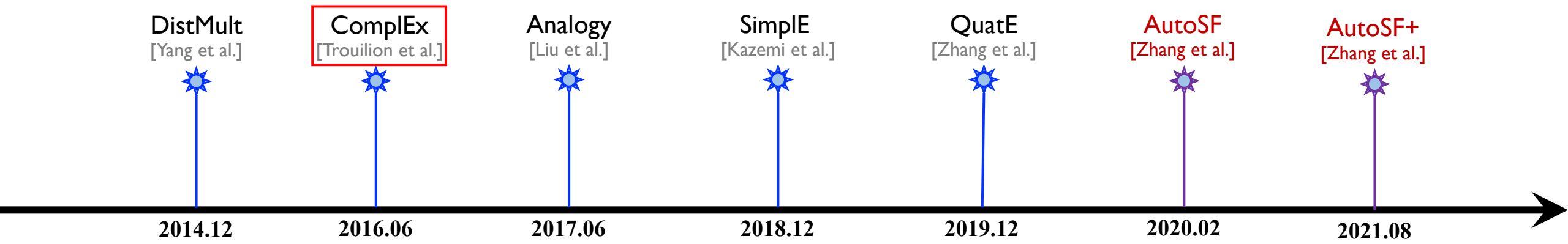
DistMult – splition



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

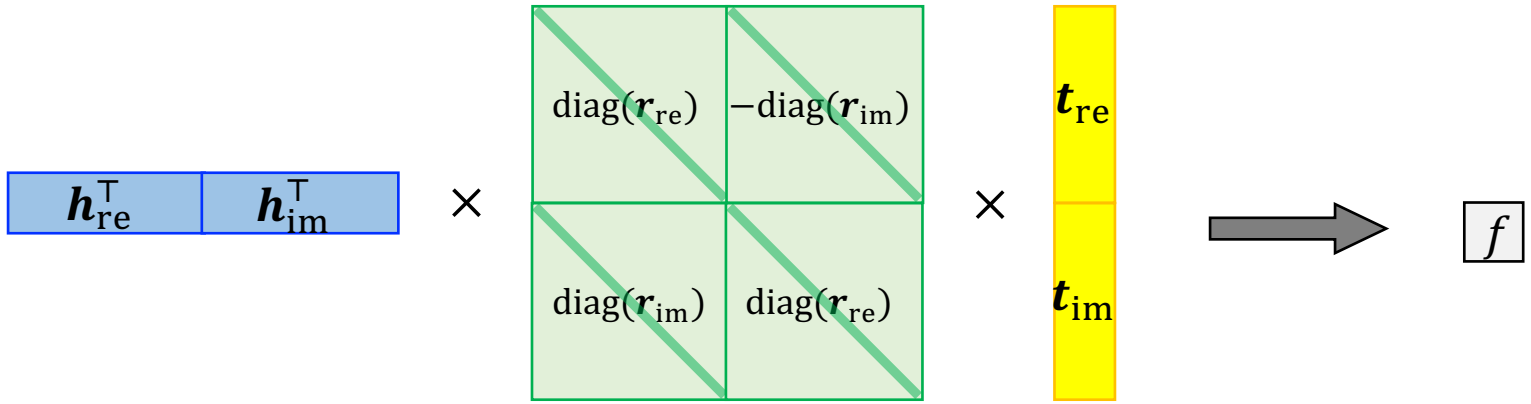


ComplEx

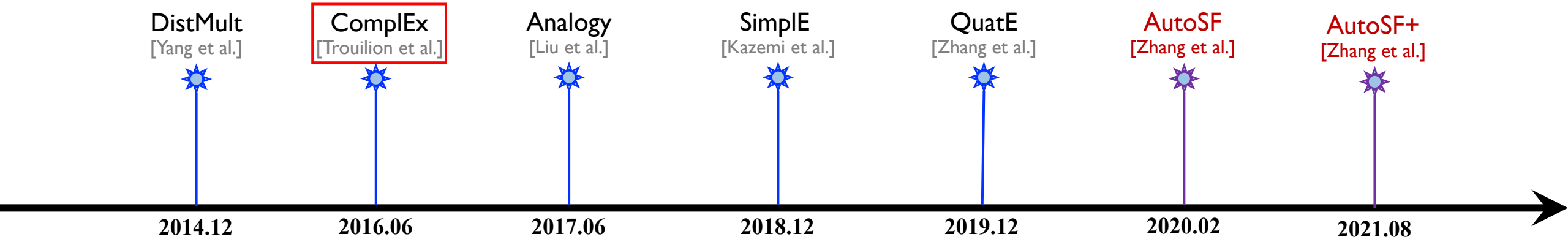


ComplEx:

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

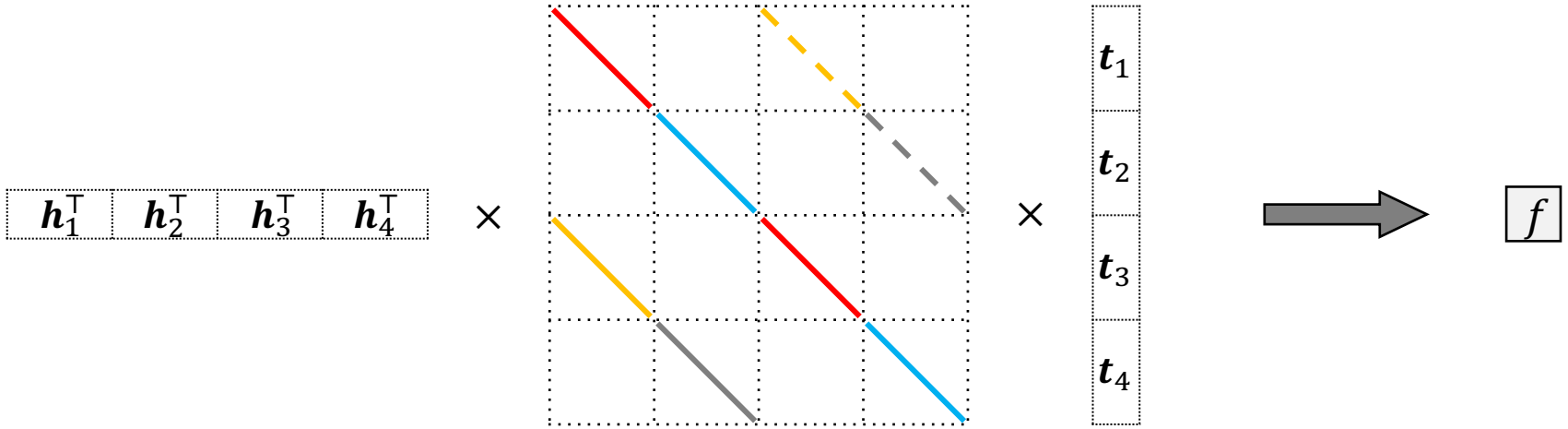


ComplEx – splition

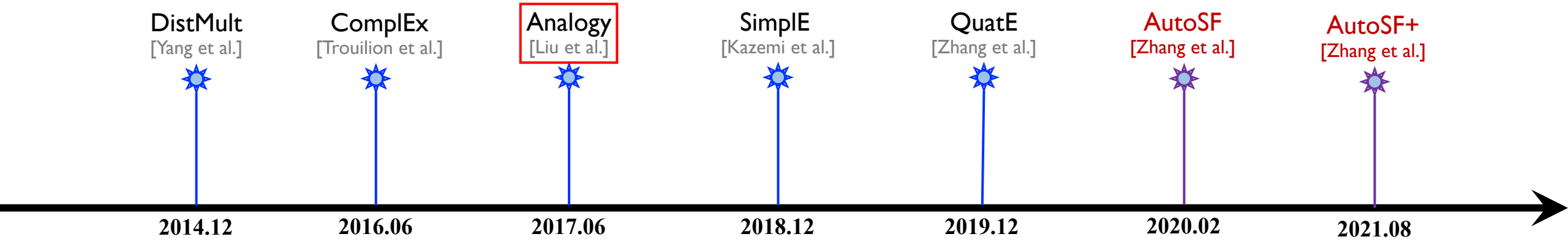


ComplEx:

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

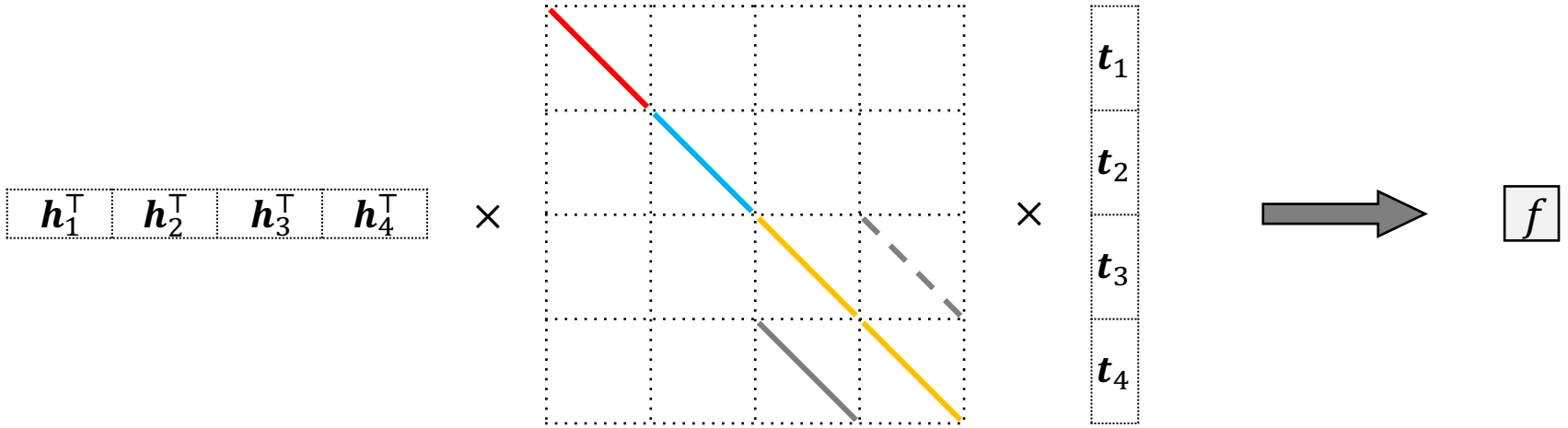


Analogy – splition

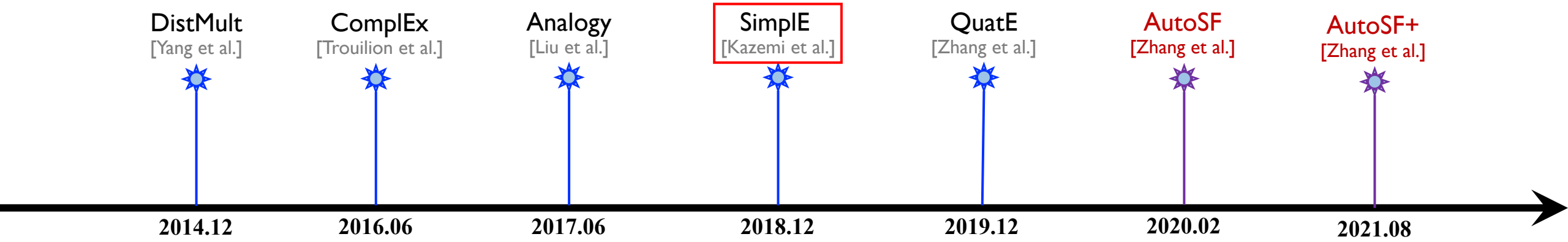


Analogy:

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

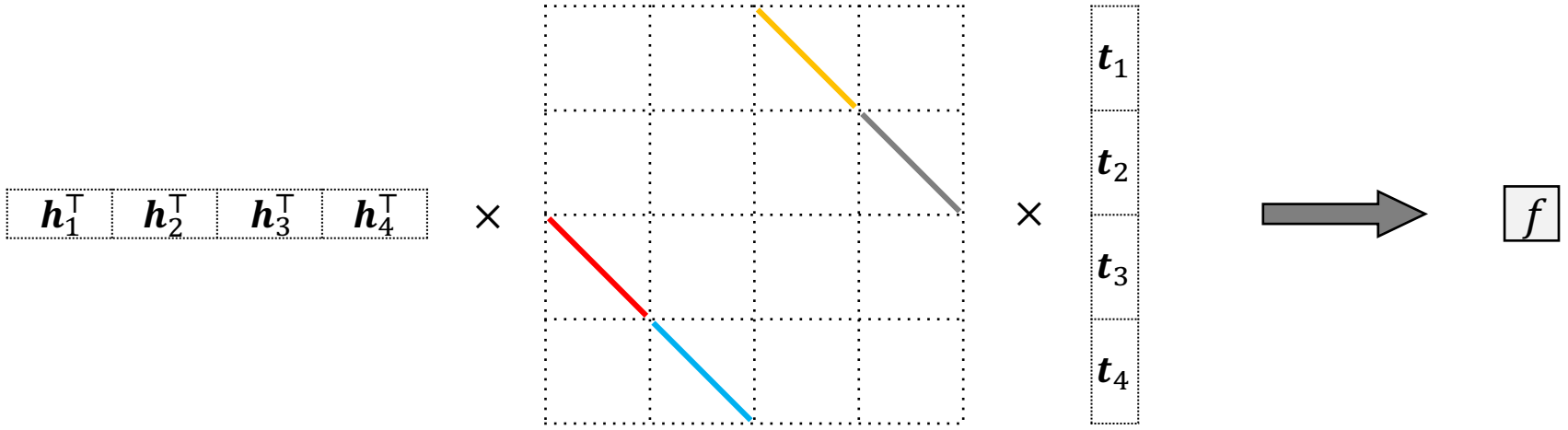


SimplE – splition

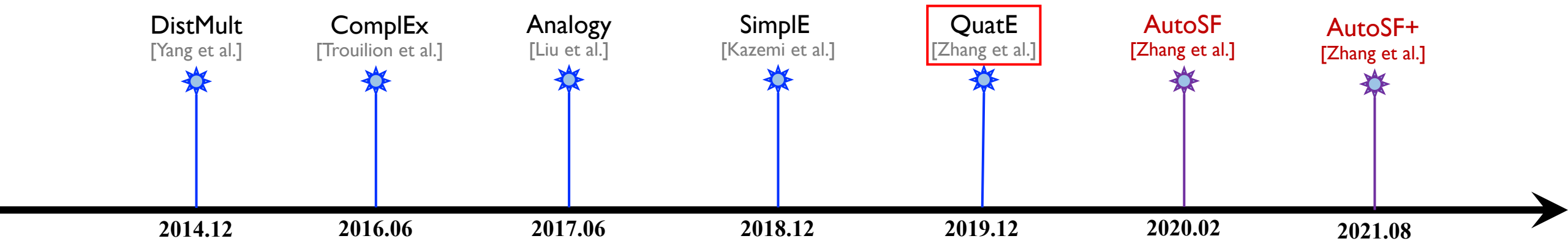


SimplE:

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

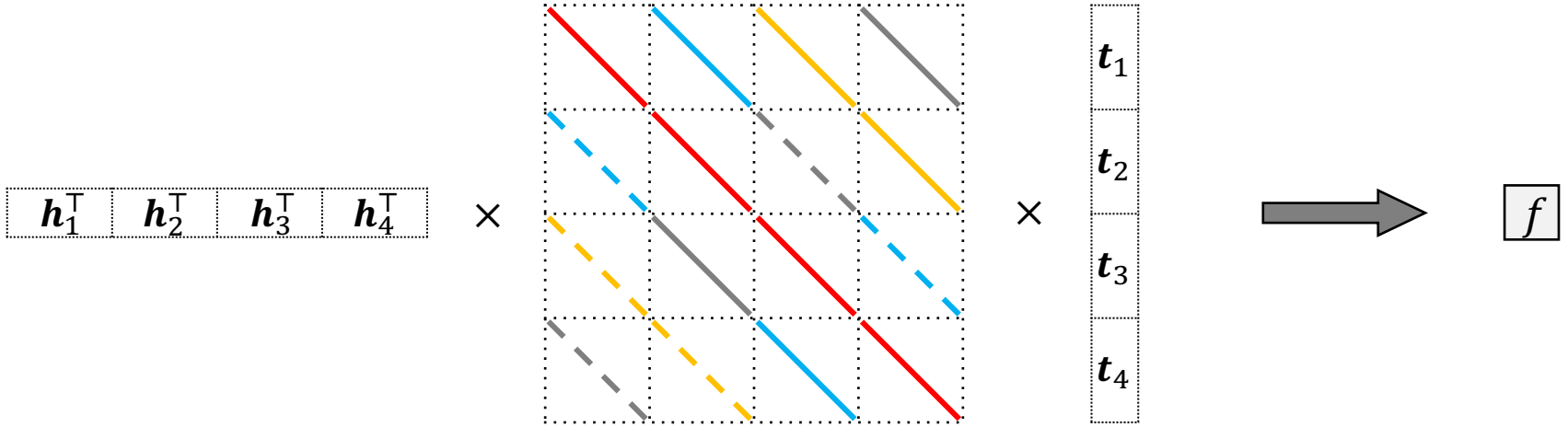


QuatE



QuatE:

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

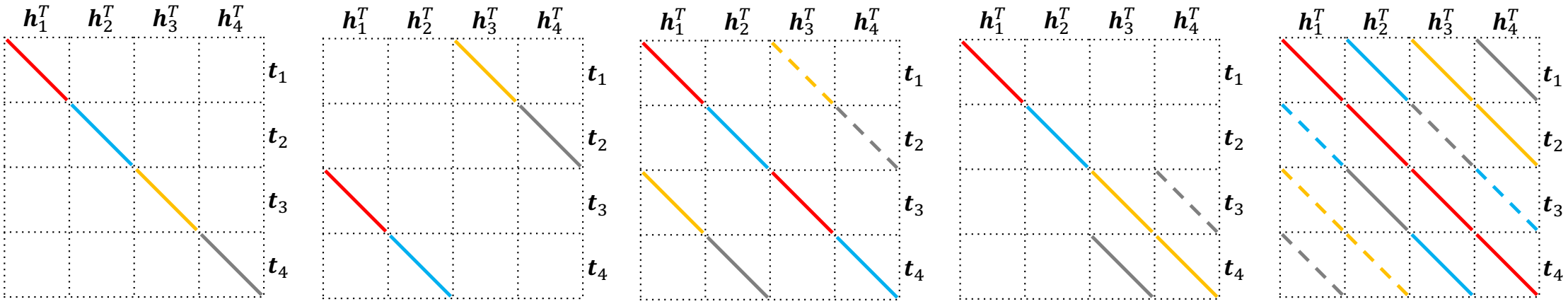


BLMs – unified form

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

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

searching!



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

Outline

1. Background
   - Knowledge Graph (KG)
   - KG embedding
2. Important aspects in KGE
3. Scoring function design
   - Overview
   - AutoSF
4. Summary

Search space

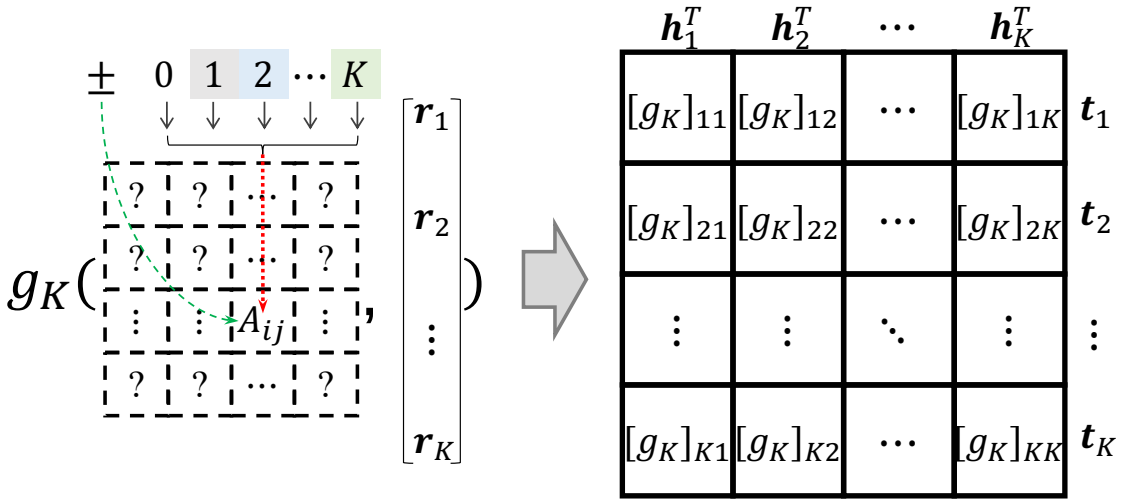
Definition 3 (Unified BiLinear Model). *The desired scoring function is of the form*

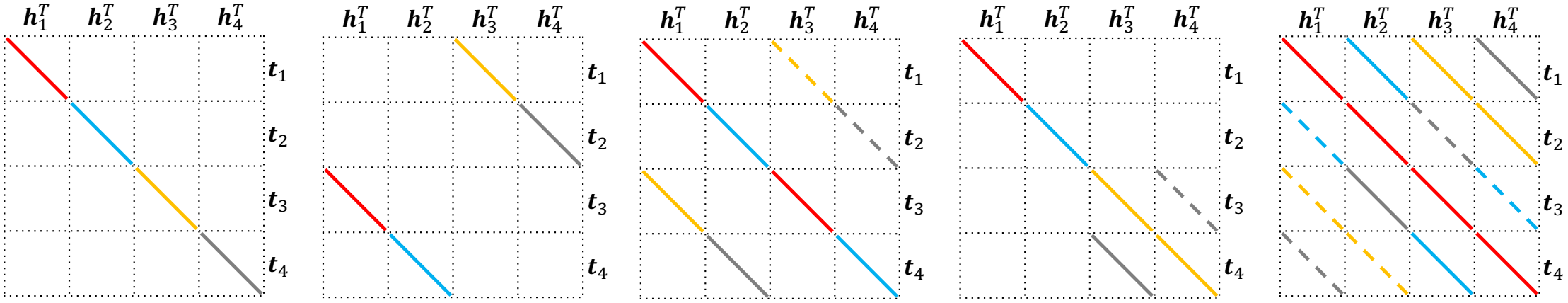
$$f_{\boldsymbol{A}}(h,r,t) = \sum_{i,j=1}^{K} \text{sign}(A_{ij}) \langle \boldsymbol{h}_i, \boldsymbol{r}_{|A_{ij}|}, \boldsymbol{t}_j \rangle , \quad (4)$$

where

$$\boldsymbol{A} \in \{0, \pm 1, \ldots, \pm K\}^{K \times K} \quad (5)$$

is called the structure matrix. *Here, we define* $\boldsymbol{r}_0 \equiv \boldsymbol{0}$, *and* $\text{sign}(0) = 0$.





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

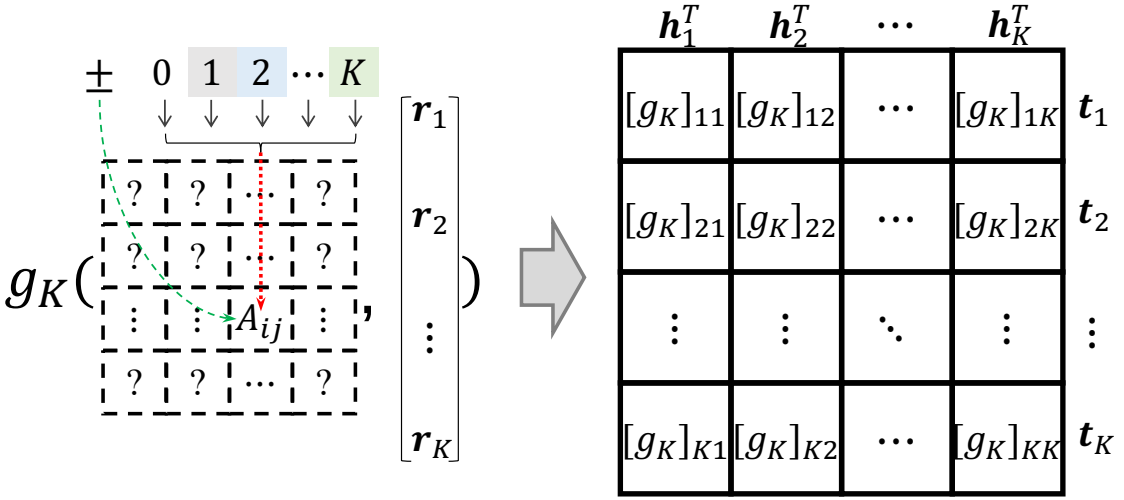
Search problem

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

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:*

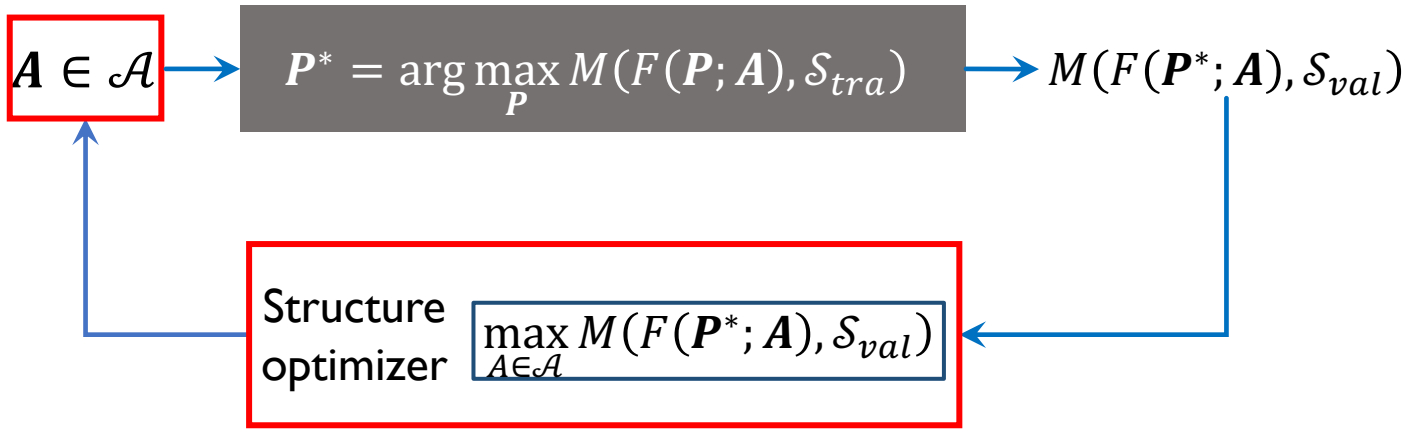
$$\boldsymbol{A}^* \in \operatorname{Arg\,max}_{\boldsymbol{A}\in\mathcal{A}} M\left(F(\boldsymbol{P}^*;\boldsymbol{A}),\mathcal{S}_{val}\right) \quad (9)$$

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



Search space:
What to be searched

Validation performance



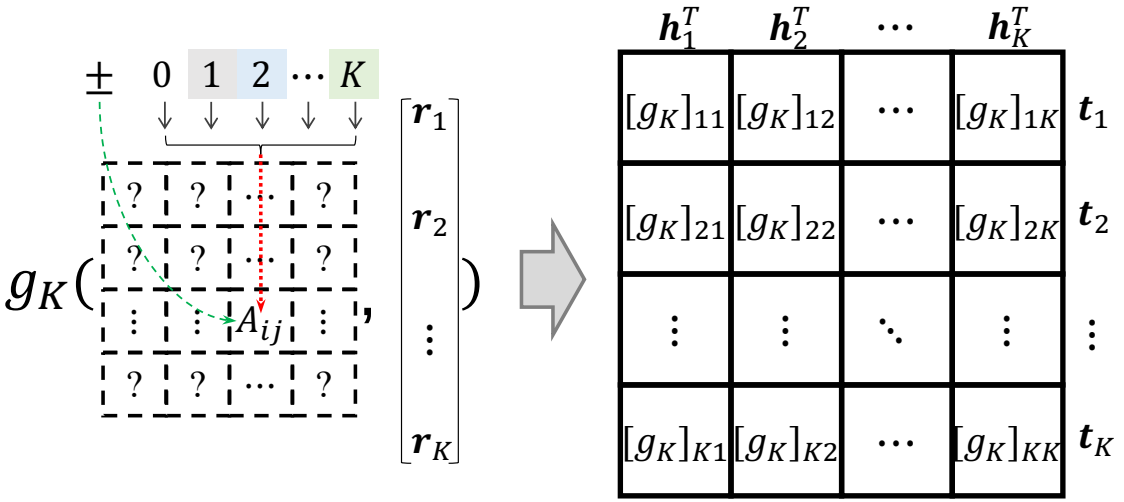
Fully expressive

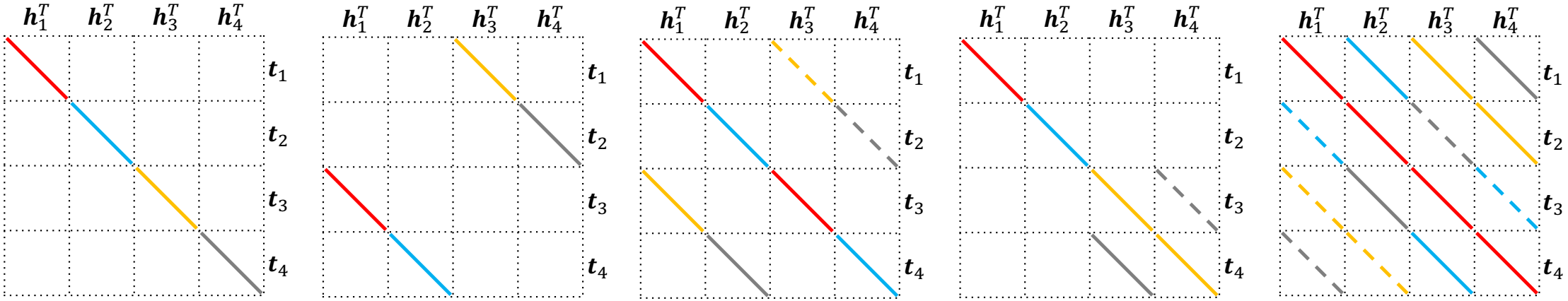
Proposition 1. *Let*

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

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.,* $g_K(\boldsymbol{A}, \check{\boldsymbol{r}})^\top = -g_K(\boldsymbol{A}, \check{\boldsymbol{r}})$*).*

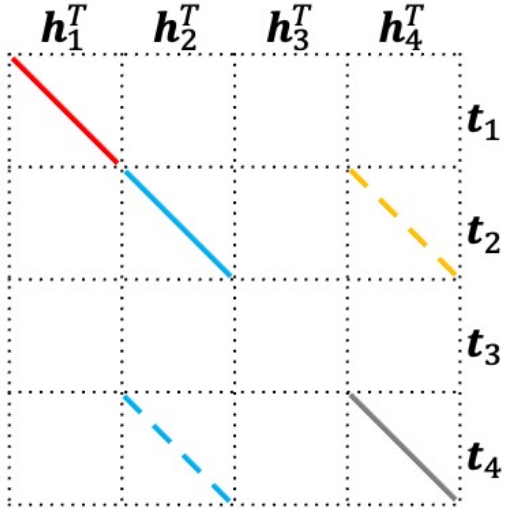


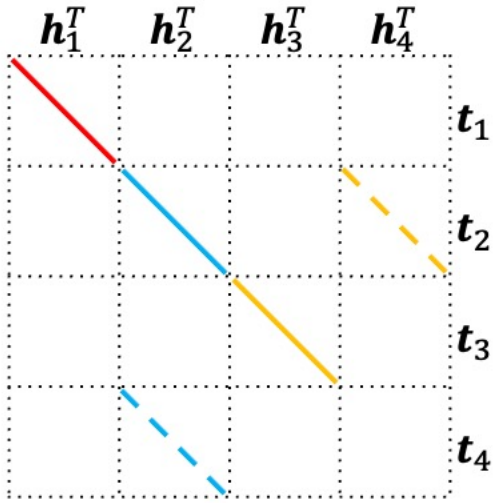


(a) DistMult. (b) SimplE. (c) ComplEx. (d) Analogy. (e) QuatE.

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* $\boldsymbol{h}^\top g_K(\boldsymbol{A}, \boldsymbol{r})\boldsymbol{t} = 0, \forall \boldsymbol{h}, \boldsymbol{t}$.

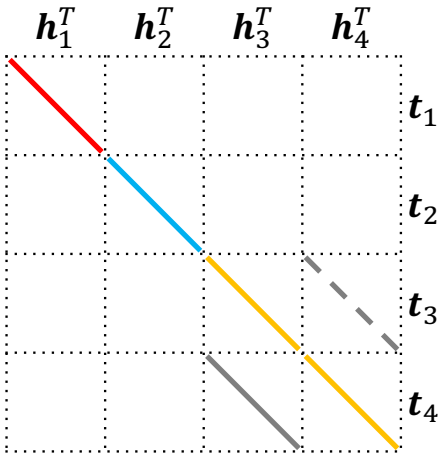




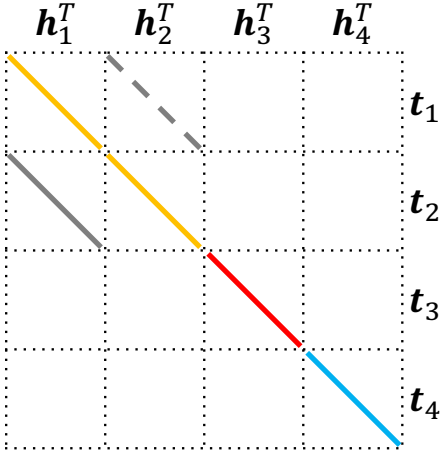
Proposition 2. $\boldsymbol{A}$ *is not degenerate if and only if* $\text{rank}(\boldsymbol{A}) = K$ *and* $\{1, \ldots, K\} \subset \{|A_{ij}| : i, j = 1, \ldots, K\}$.

Equivalent structures

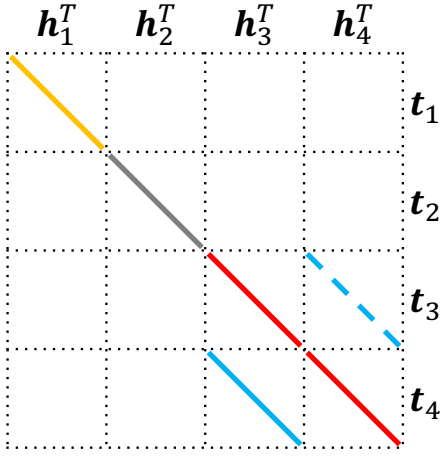
Definition 6 (Equivalence). *Let* $\boldsymbol{P}^* = \arg\max_{\boldsymbol{P}} M(F(\boldsymbol{P}; \boldsymbol{A}), \mathcal{S})$ *and* $\boldsymbol{P}'^* = \arg\max_{\boldsymbol{P}'} M(F(\boldsymbol{P}'; \boldsymbol{A}'), \mathcal{S})$. *If* $\boldsymbol{A} \neq \boldsymbol{A}'$ *but* $M\big(F(\boldsymbol{P}^*;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



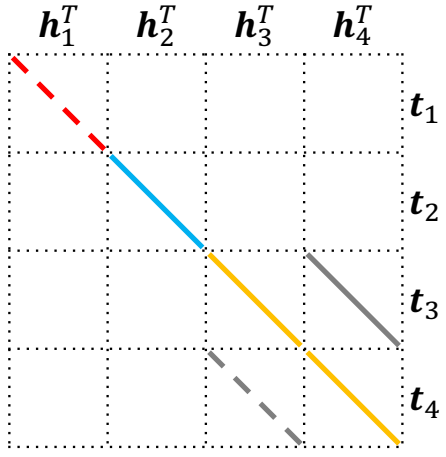
Analogy.



Permuting rows and columns.

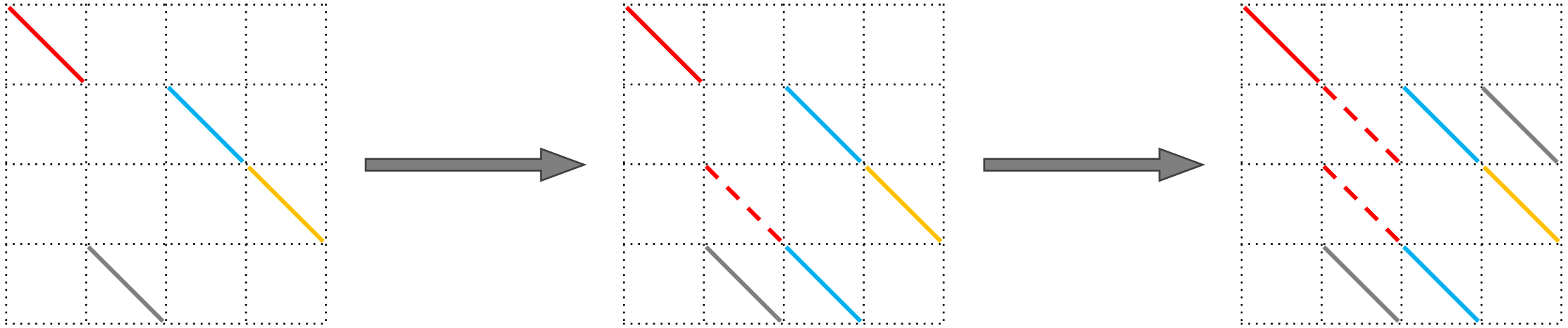


Permuting values.



Flipping signs.

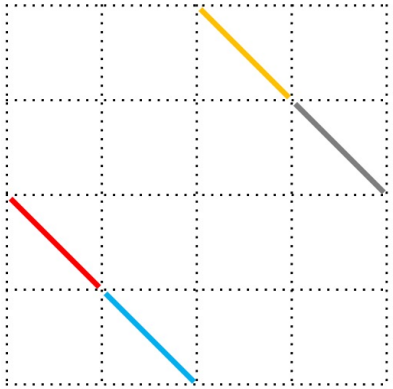
Greedy algorithm – AutoSF



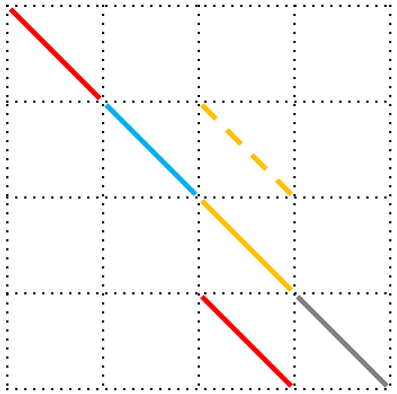
- Progressively adding values and adjusting the search space.
- Greedily preserving the top-I structures with b non-zero elements.

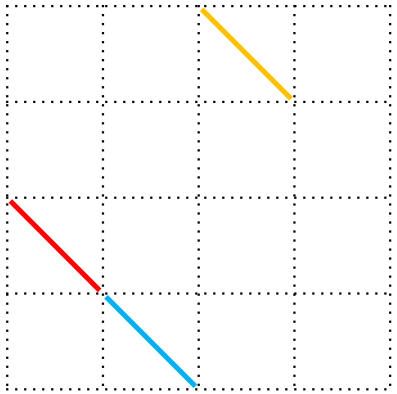
Evolutionary algorithm – AutoSF+

SimplE:

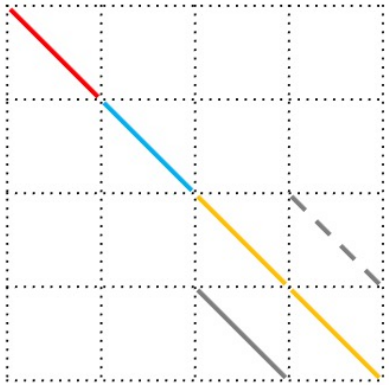


mutation:

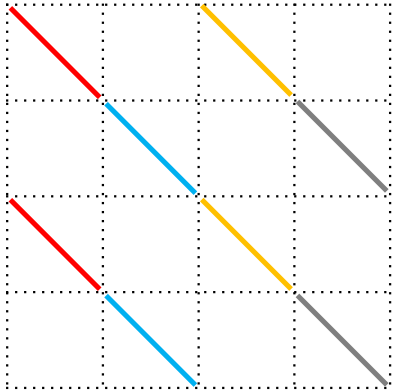


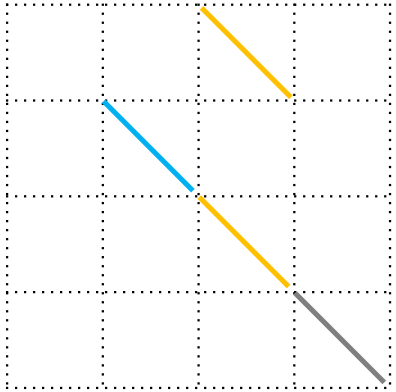


Analogy:



crossover:





- Preserving the overall top-*I* structures during searching.
- More flexible than progressive algorithm.

Structure selection

- Degenerate and equivalent structures
  - Using Filter to directly remove these structures.

- Effectiveness
  - Symmetric related features (SRF): captures how A can be symmetric and skew-symmetric
  - A predictor is learned to map SRF to performance.

- Key idea:
  - select better SFs based on matrix structure to train and evaluate.

Expressiveness – standard benchmarks

	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**

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';

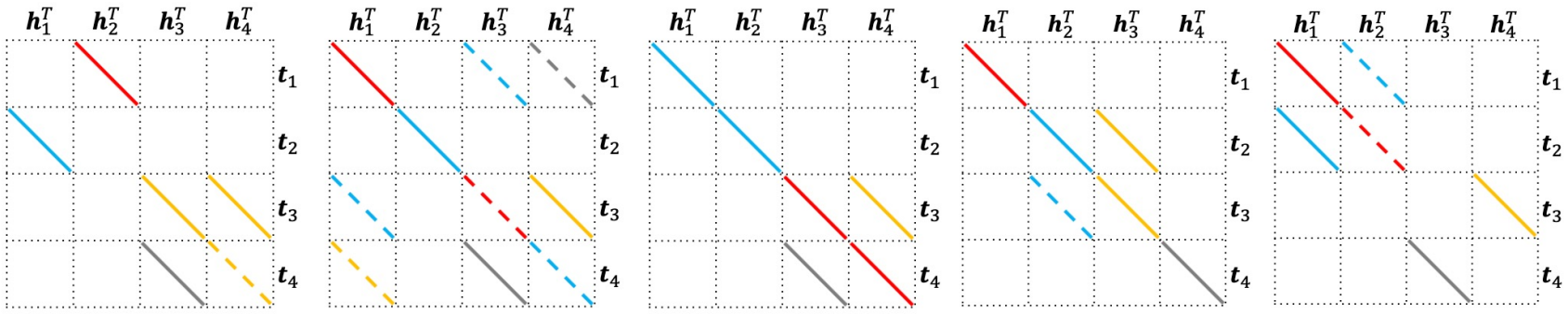
Metrics

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

- BLMs are better than the other models.
- There is no absolute winner among the BLMs.
- Compared with human-designed ones, the SFs searched by AutoSF+ always lead the performance.

Expressiveness – standard benchmarks

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

OGB – results

- Open graph benchmark:

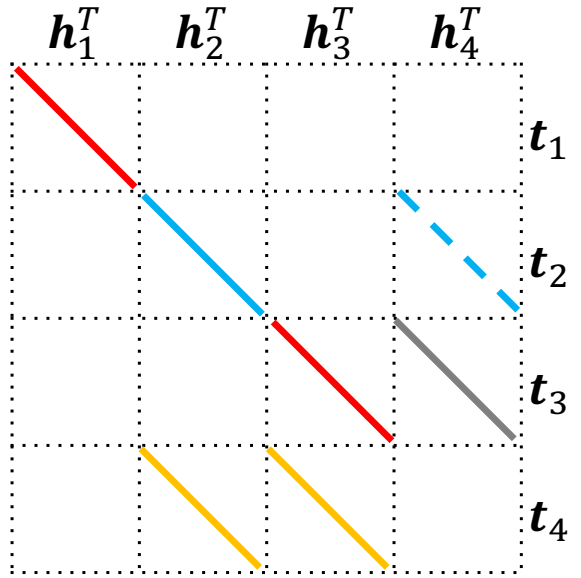
Leaderboard for ogbl-wikikg2

The MRR score on the test and validation sets. The higher, the better.

Package: >=1.2.4

Deprecated `ogbl-wikikg` leaderboard can be found here.

Rank	Method	Test MRR	Validation MRR	Contact	References	#Params	Hardware	Date
1	**NodePiece + AutoSF**	0.5703 ± 0.0035	0.5806 ± 0.0047	Mikhail Galkin (Mila)	Paper, Code	6,860,602	Tesla V100 (32 GB)	Jul 17, 2021
2	**AutoSF**	0.5458 ± 0.0052	0.5510 ± 0.0063	Yongqi Zhang (4Paradigm)	Paper, Code	500,227,800	Quadro RTX 8000 (45GB GPU)	Apr 2, 2021
3	**PairRE (200dim)**	0.5208 ± 0.0027	0.5423 ± 0.0020	Linlin Chao	Paper, Code	500,334,800	Tesla P100 (16GB GPU)	Jan 28, 2021
4	RotatE (250dim)	0.4332 ± 0.0025	0.4353 ± 0.0028	Hongyu Ren – OGB team	Paper, Code	1,250,435,750	Quadro RTX 8000 (45GB GPU)	Jan 23, 2021
5	TransE (500dim)	0.4256 ± 0.0030	0.4272 ± 0.0030	Hongyu Ren – OGB team	Paper, Code	1,250,569,500	Quadro RTX 8000 (45GB GPU)	Jan 23, 2021



OGB – results

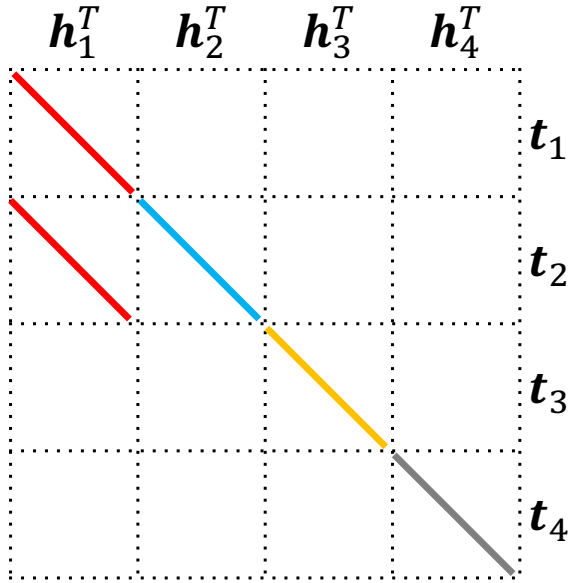
- Open graph benchmark:

Leaderboard for ogbl-biokg

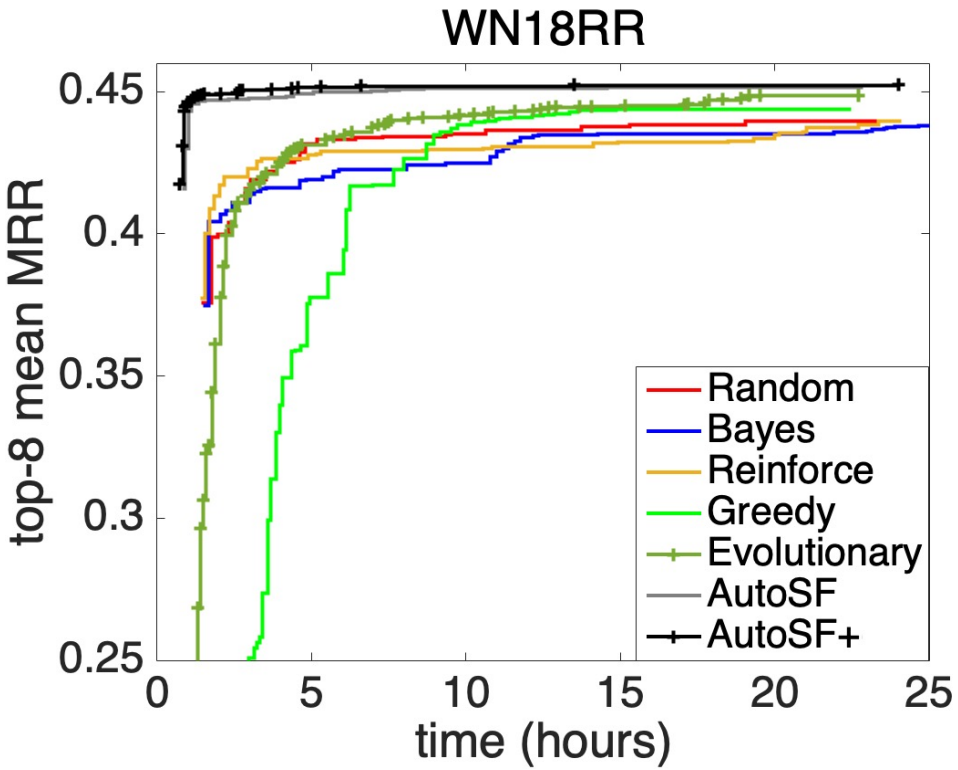
The MRR score on the test and validation sets. The higher, the better.

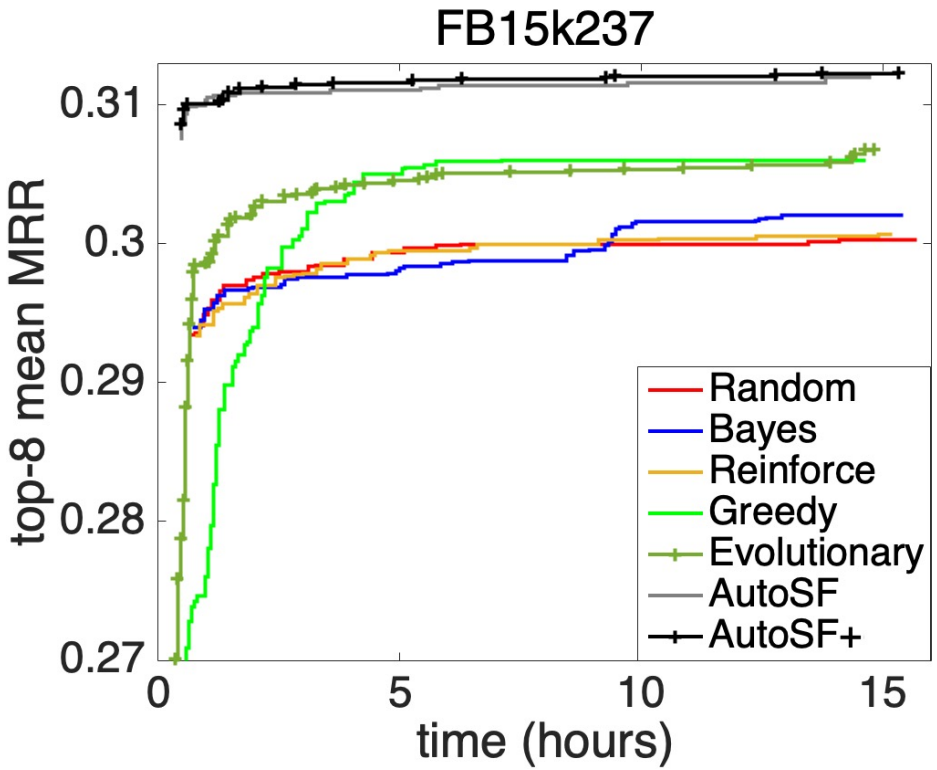
Package: >=1.2.0

Rank	Method	Test MRR	Validation MRR	Contact	References	#Params	Hardware	Date
1	**AutoSF**	0.8309 ± 0.0008	0.8317 ± 0.0007	Yongqi Zhang (4Paradigm)	Paper, Code	93,824,000	GeForce RTX 2080 (11GB GPU)	Apr 2, 2021
2	**PairRE**	0.8164 ± 0.0005	0.8172 ± 0.0005	LinlinChao (AntGroup KG&NLP)	Paper, Code	187,750,000	Tesla P100 (16GB GPU)	Nov 9, 2020
3	ComplEx	0.8095 ± 0.0007	0.8105 ± 0.0001	Hongyu Ren – OGB team	Paper, Code	187,648,000	GeForce RTX 2080 (11GB GPU)	Jun 10, 2020
4	DistMult	0.8043 ± 0.0003	0.8055 ± 0.0003	Hongyu Ren – OGB team	Paper, Code	187,648,000	GeForce RTX 2080 (11GB GPU)	Jun 10, 2020
5	RotatE	0.7989 ± 0.0004	0.7997 ± 0.0002	Hongyu Ren – OGB team	Paper, Code	187,597,000	GeForce RTX 2080 (11GB GPU)	Jun 10, 2020
6	TransE	0.7452 ± 0.0004	0.7456 ± 0.0003	Hongyu Ren – OGB team	Paper, Code	187,648,000	GeForce RTX 2080 (11GB GPU)	Jun 10, 2020



Efficiency





- Greedy and Evolutionary algorithms are better than Random, Bayes and Reinforce.
- The Filter and Predictor significantly improve efficiency.

Entity classification task

AutoSF as the message function in GNN.

$$\boldsymbol{e}_i^{\ell+1} = \sigma\left(\boldsymbol{W}_0^{\ell}\boldsymbol{e}_i^{\ell} + \sum_{(r,j):(i,r,j)\in\mathcal{S}_{\text{tra}}} \boldsymbol{W}^{\ell}\phi(\boldsymbol{e}_j^{\ell}, \boldsymbol{r}^{\ell})\right),$$

$$\phi\left(\boldsymbol{e}_j^{\ell}, \boldsymbol{r}^{\ell}\right) = g_K\left(\boldsymbol{A}, \boldsymbol{r}^{\ell}\right)\boldsymbol{e}_j^{\ell}$$

Table 7: Results on entity classification task.

dataset	AIFB	MUTAG	BGS
GCN	86.67±2.33	68.83±2.18	73.79±3.08
R-GCN	92.78±2.48	74.12±2.67	82.97±0.47
CompGCN	90.6±0.2	85.3±1.2	84.14±1.89
AutoSF+	**96.66±1.24**	**85.88±1.32**	**86.17±1.31**

Summary

- AutoSF(+)
  - An AutoML approach to design bilinear scoring functions.
- Search space:
  - Unified bilinear models
  - Guaranteed with expressiveness
- Search algorithm:
  - Filter, remove degenerate and equivalent
  - Predictor, predict based on expressiveness
  - Greedy and Evolutionary algorithm

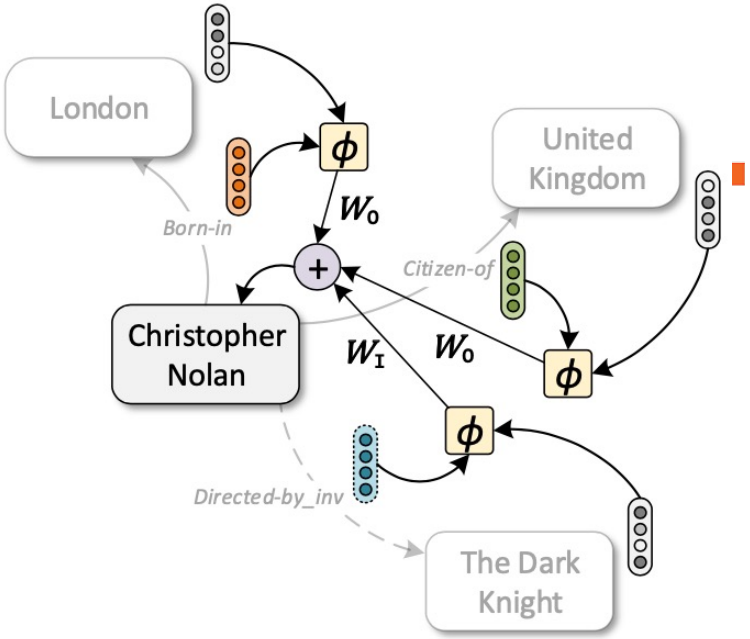
Outline

1. Background
   - Knowledge Graph (KG)
   - KG embedding
2. Important aspects in KGE
3. Scoring function design
   - Overview
   - AutoSF
4. Summary

Take away

- KG benefits real-world applications by providing auxiliary information.
- KG embedding cares about semantic modelling of relations.
- Bilinear model is the SOTA scoring function.
- Searching Bilinear models can be both effective and efficient.

Future work



- GNN architecture design for KG.
- Efficient hyperparameter search for KG embedding.
- Generative model for biological, physical KGs.

Related publications

Collaborative Knowledge Base Embedding for Recommender Systems. F. Zhang, J. Yuan, D. Lian, X. Xie and W. Ma. KDD 2016.

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.

AutoSF+: Bilinear Scoring Function Search for Knowledge Graph Learning. **Y. Zhang**, Q. Yao, Y. Li and J. Kwok. Under review

https://github.com/AutoML-Research

Next Works

Research Topic

Automated Learning from Graph Structured Data

Manage topic | Submit your abstract | Submit your manuscript | Participate

Special Issue in: **Frontiers in artificial intelligence**

Jointly hold by: Quanming Yao, Huan Zhao

Rex Ying, and Xia Hu

Overview | Articles | Authors | Impact

Your submission is welcome!

About this Research Topic

Machine learning is an important technique to learn from graph structured data (GSD). However, since it is knowledge- and labor-intensive to pursue good learning performance, humans are heavily involved in every aspect of learning from GSD. Examples are property prediction for molecular graphs, product recommendations from heterogeneous information networks, and logical queries from knowledge base. To make machine learning techniques easier to apply and reduce the demand for experienced human experts, automated machine learning (AutoML) has emerged as a hot topic with great interest. At the same time, a special type of deep network model has been recently proposed, called graph neural networks (GNNs), which capture the dependence of graphs via message passing between the nodes of graphs and allow learning from GSD in a more principled manner. Moreover, they allow learning from molecular dynamics and meteorological simulation, which further extend the classical scope of GSD.

However, again due to such diversities in GSD, there is no unitary GSD learning model that can perform consistently across different tasks and datasets. As a consequence, how to design adaptive methods to learn from GSD in a task-aware and data-specific manner is an important problem. Fortunately, there are emerging technical tools from machine learning communities that have the potential to solve the above problems. Examples are automated machine learning, neural architecture search, meta-learning, and learning to lean. These methods can all help generalize learning methods that can exhibit abilities to learn well across different datasets and tasks. Thus, by proposing this Research Topic, we hope to draw interest from both academia and industry, with the goal to push learning methods for GSD to the next level.

Submission Deadlines

19 December 2021	Abstract
17 February 2022	Manuscrip

Next Works

AH2: Automated Learning form Graph-Structured Data
Quanming Yao, Huan Zhao and Yongqi Zhang

Graph-structured data (GSD) is ubiquitous in real-life applications, which appears in many learning applications such as property prediction for molecular graphs, product recommendations from heterogeneous information networks, and logical queries from knowledge graphs. Recently, learning from graph-structured data has also become a research focus in the machine learning community. However, again due to such diversities in GSD, there are no universal learning models that can perform well and consistently across different learning applications based on graphs. In sharp contrast to this, convolutional neural networks work well on natural images, and transformers are good choices for text data. In this tutorial, we will talk about using automated machine learning (AutoML) as a tool to design learning models for GSD. Specifically, we will elaborate on what is AutoML, what kind of prior information from graphs can be explored by AutoML, and how insights can be generated from the searched models.

Quanming Yao

Tsinghua University

Dr. Quanming Yao is a tenure-track assistant professor at EE, Tsinghua University. He obtained his Ph.D. degree at the CSE of HKUST and was the founding leader of 4Paradigm INC's machine learning research team. He is a recipient of Forbes-30-Under-30 (China), Young Scientist Awards (HKIS), and Google Fellowship (machine learning).

Huan Zhao

4Paradigm

Dr. Huan Zhao is a senior researcher in 4Paradigm, leading the research on automated graph representation learning (AutoGraph) and real-world applications like retailing recommendation and bioinformatics. He has published various papers on top-tier venues like KDD, CIKM, AAAI, and TKDE. He obtained his Doctor Degree from HKUST in Jan. 2019.

Yongqi Zhang

4Paradigm

Dr. Yongqi Zhang is a senior researcher in 4Paradigm. He obtained the Ph.D. degree at CSE of HKUST. He has published many top-tier conference/journal papers as first-author in NeurIPS, ICDE, VLDB-J. His research interests focus on KG embedding and AutoML. He has been a reviewer for AAAI, IJCAI, CIKM and TKDE.

AAAI-22 Tutorial Forum

Thirty-Sixth Conference on Artificial Intelligence

February 23, 2022

Vancouver, BC, Canada

Our next stop

- more recent works will be included

Thank you!