

Automated Recommender System (RecSys)

Quanming Yao^{1,2}, Yong Li¹, Chen Gao¹, Huan Zhao², Yongqi Zhang²

¹*Department of Electronic Engineering, Tsinghua University*

²*4Paradigm Inc.*

<https://quanmingyao.github.io/AutoML.github.io/ijcai21-tutorial.html>



Tutorial Outline

- I. An introduction to Automated Machine Learning (AutoML)
 - Background on technical tools from machine learning
2. Why AutoML is Needed in RecSys and Recent Advances
 - Exemplar works introducing AutoML into RecSys
3. Automated Graph Representation Learning for RecSys
 - Explore neural architecture search for GNN based RecSys
4. Automated Knowledge Graph (KG) Embedding
 - Explore AutoML for KG Embedding based RecSys

Schedule at a Glance

Time	Event
0:00-0:40 minutes	Part 1: An introduction to Automated Machine Learning (AutoML) Speaker: Quanming Yao
0:40-1:20 minutes	Part 2: Why AutoML is Needed in RecSys and Recent Advances Speaker: Chen Gao
1:20-1:30 minutes	Break
1:30-2:10 minutes	Part 3: Automated Graph Neural Network for RecSys Speaker: Huan Zhao
2:10-2:50 minutes	Part 4: Automated Knowledge Graph Embedding Speaker: Yongqi Zhang
2:50-3:00 minutes	Part 5: Discussion

Automated Recommender System (RecSys) Tutorial

Part 4: Automated Knowledge Graph Embedding

Yongqi Zhang

4Paradigm Inc.

zhangyongqi@4paradigm.com

Outline

I. Background

- KG for RecSys
- KG embedding

2. Important aspects in KGE

3. Scoring function design

- Overview
- AutoSF

4. Summary

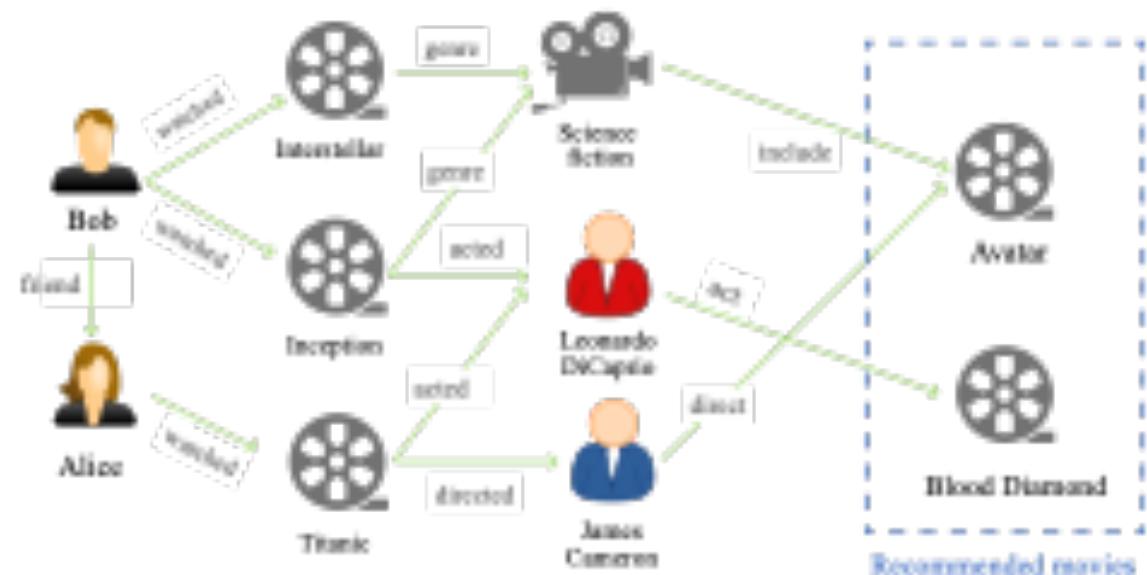
KG for recommendation

Knowledge Graph

- introducing external information for RecSys

Benefits

- improve accuracy
- provide explanation
- alleviate cold start



[Guo et. al. 2020]

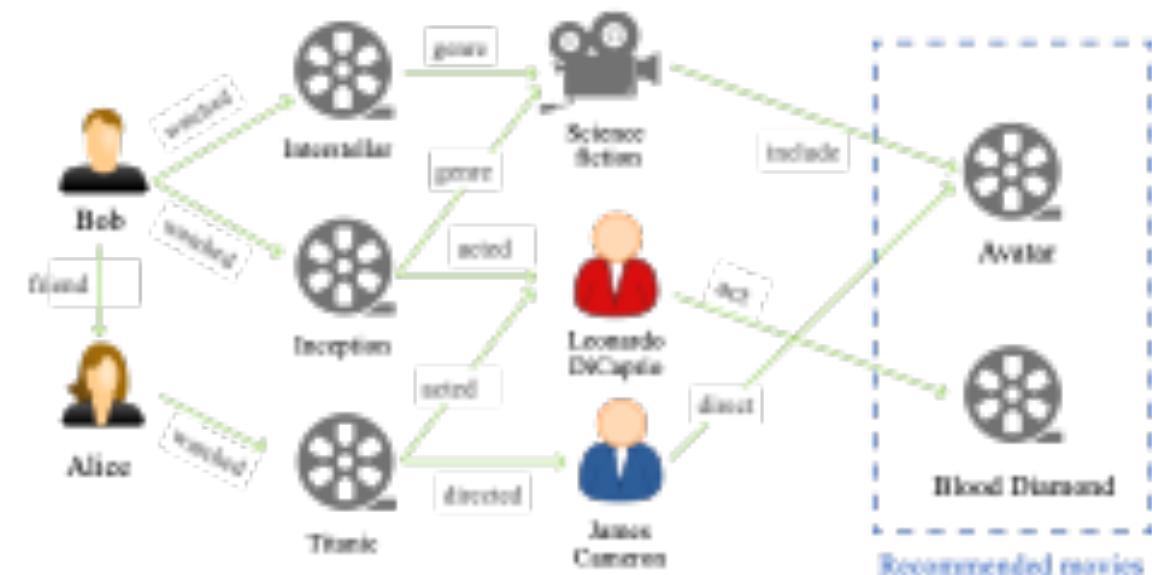
KG for recommendation

Entities

- users, movies, actors, directors, genres

Relations

- interaction, belonging, acting,
- directing, friendship



[Guo et. al. 2020]

KG for recommendation

User-item interactions:

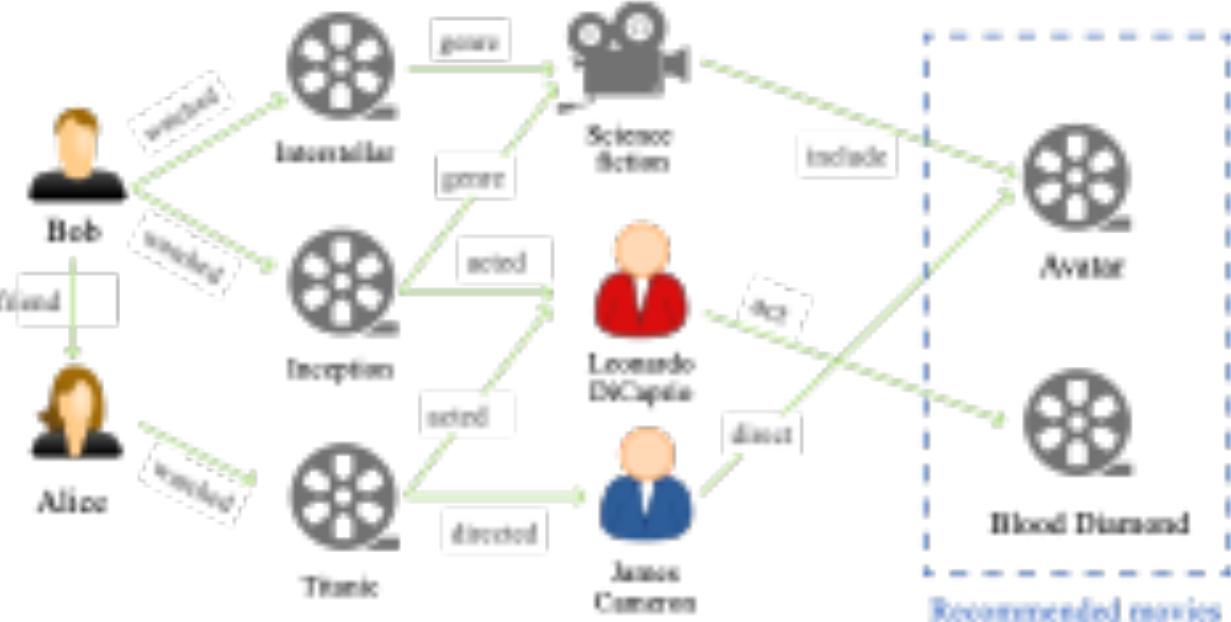
- (Bob, watched, Intestellar)

User-user connections:

- Social relationships
- (Bob, isFriendof, Alice)

Item-item connections:

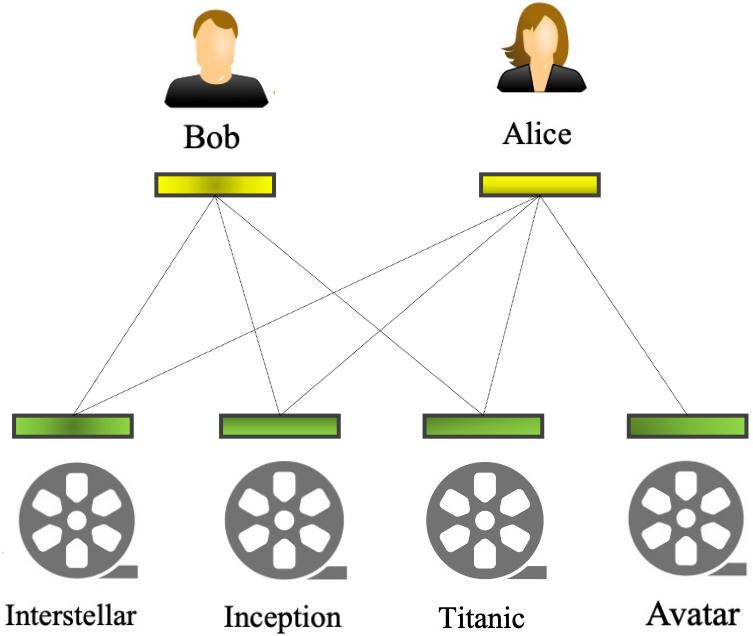
- provide additional information
- (Cameron, directed, Titanic)
- (Inception, genre, Science_fiction)



[Guo et. al. 2020]

Rich heterogeneous information

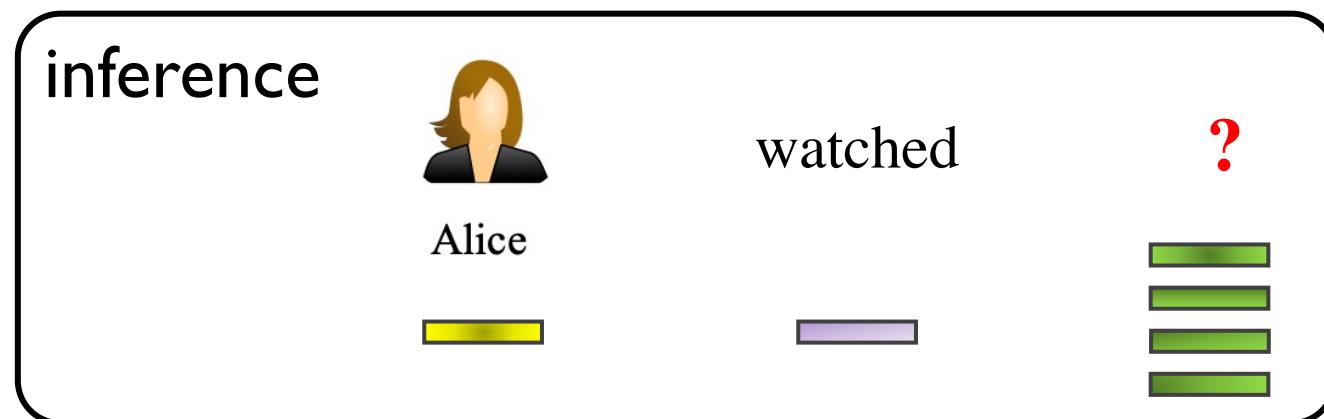
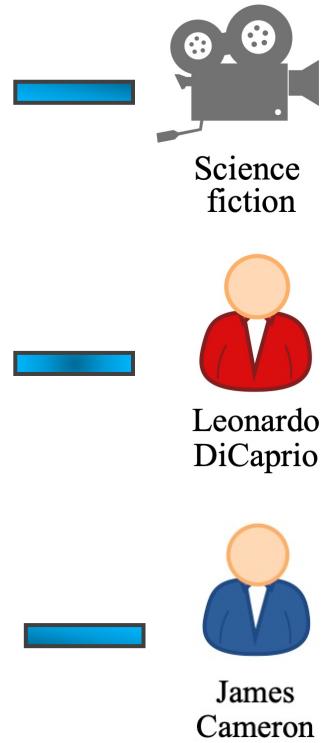
KGE based RecSys



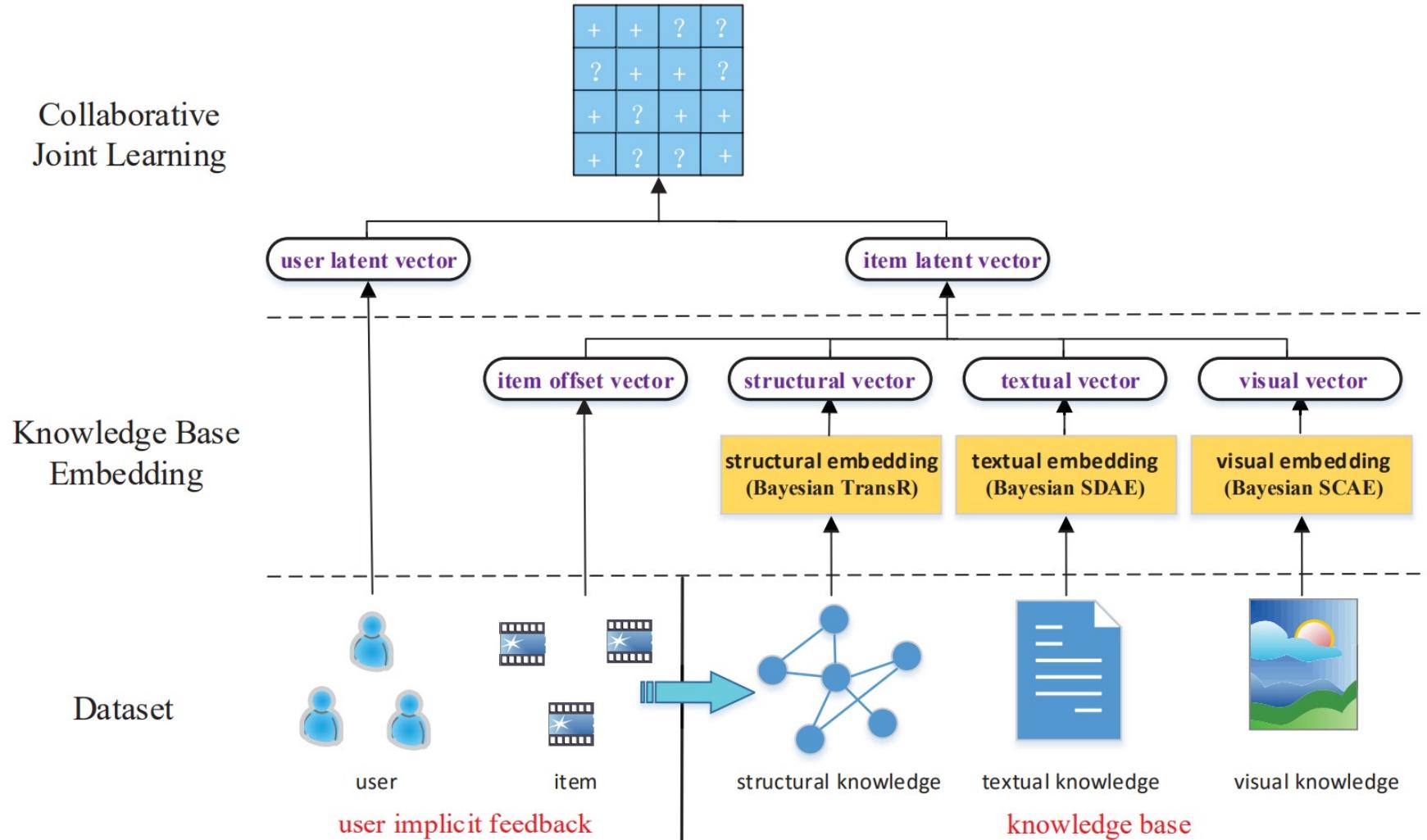
improve



- [purple bar] watched
- [purple bar] genre
- [purple bar] directed
- [purple bar] act
- [purple bar] include
- [purple bar] direct



Collaborative KGE based RecSys



Outline

I. Background

- KG for RecSys
- KG embedding

2. Important aspects in KGE and AutoML

3. Scoring function design

- Overview
- AutoSF

4. Summary

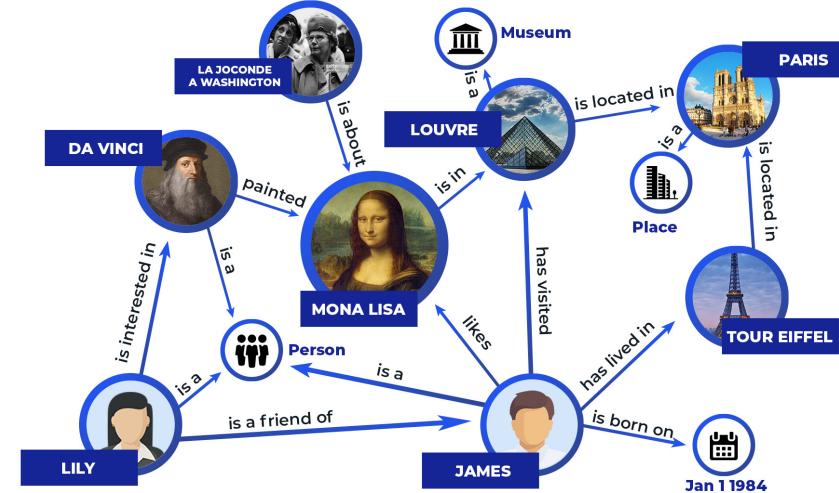
Knowledge graph

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

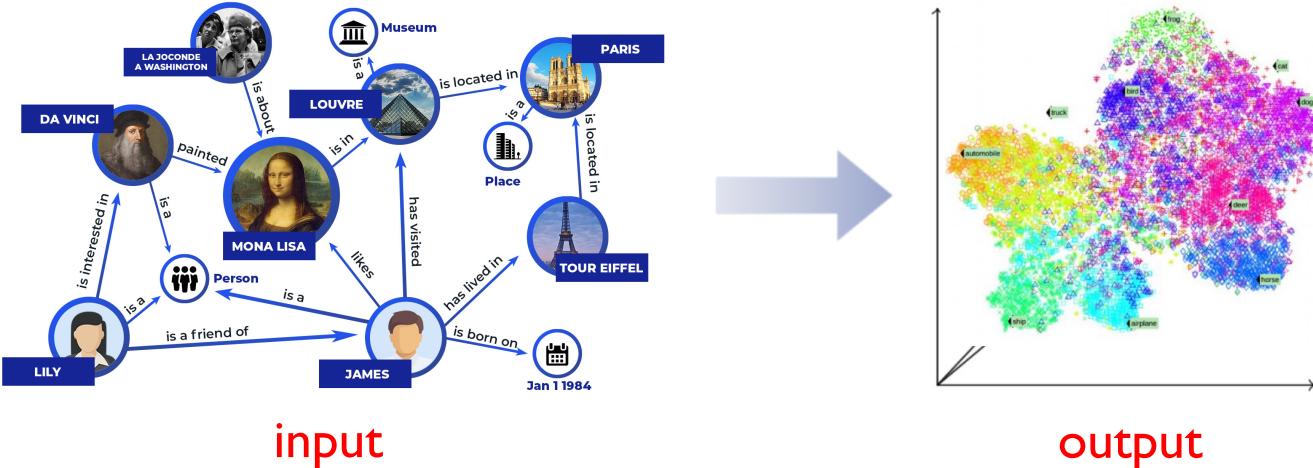


Freebase™

BIO2RDF

Embedding (representation)

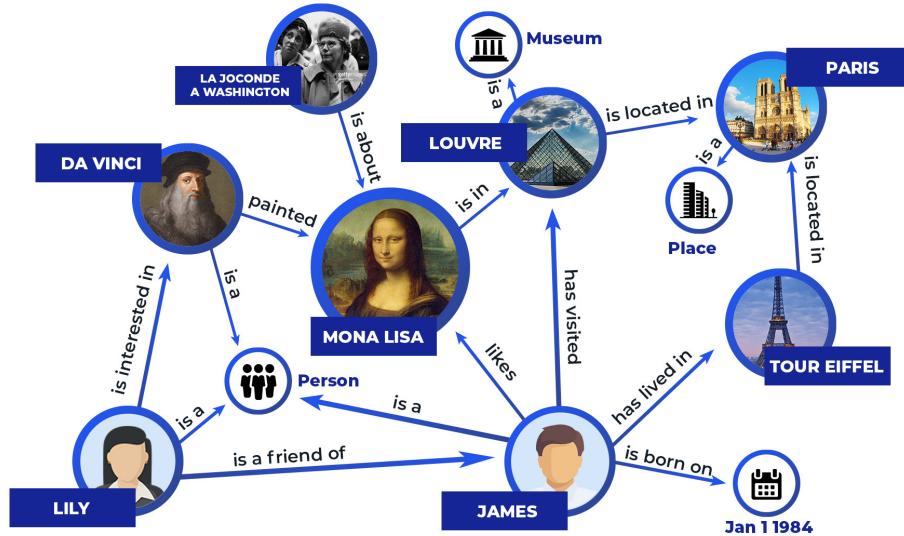
Encode **entities** and **relations** in KG into low-dimensional **vector spaces** \mathbb{R}^{d_1} and \mathbb{R}^{d_2} , while capturing nodes' and edges' connection properties.



Advantages:

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

Representation learning



Observed triple S^+ :
maximize score

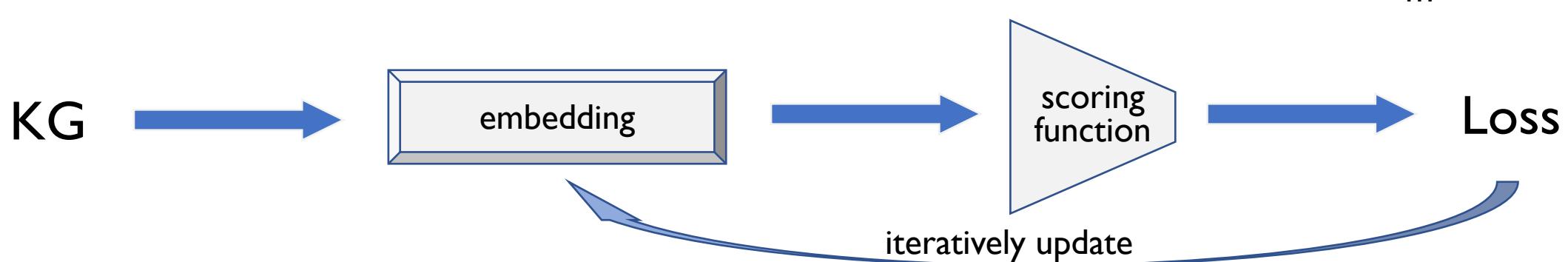
(Lily, isA, Person)
(James, likes, MonaLisa)
(Louver, isLocatedIn, Paris)
(Lily, isInterestedIn, DaVinci)

...

Unobserved triple S^- :
minimize score

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

...

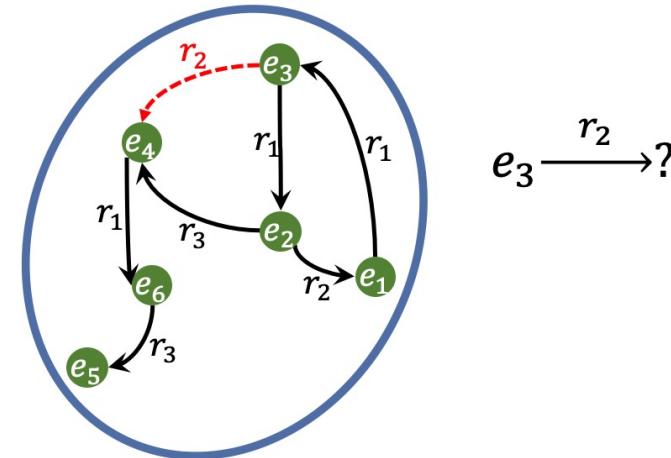


➤ Objective: Preserve as much information on original graph as possible.

Evaluation

- Key task:
 - Link prediction (cater to RecSys)
- 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:

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



Quick summary

- KG provides external information for RecSys that can improve **accuracy, explanation** and alleviate **cold start**.
- KG embedding is a strong technique in learning KGs for **efficient** and **accurate** prediction.

Outline

I. Background

- KG for RecSys
- KG embedding

2. Important aspects in KGE

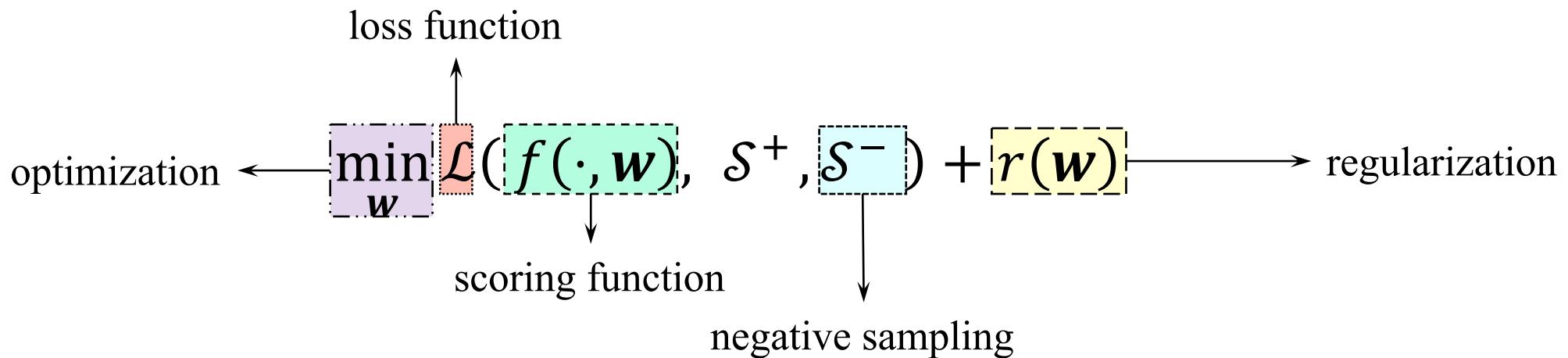
3. Scoring function design

- Overview
- AutoSF

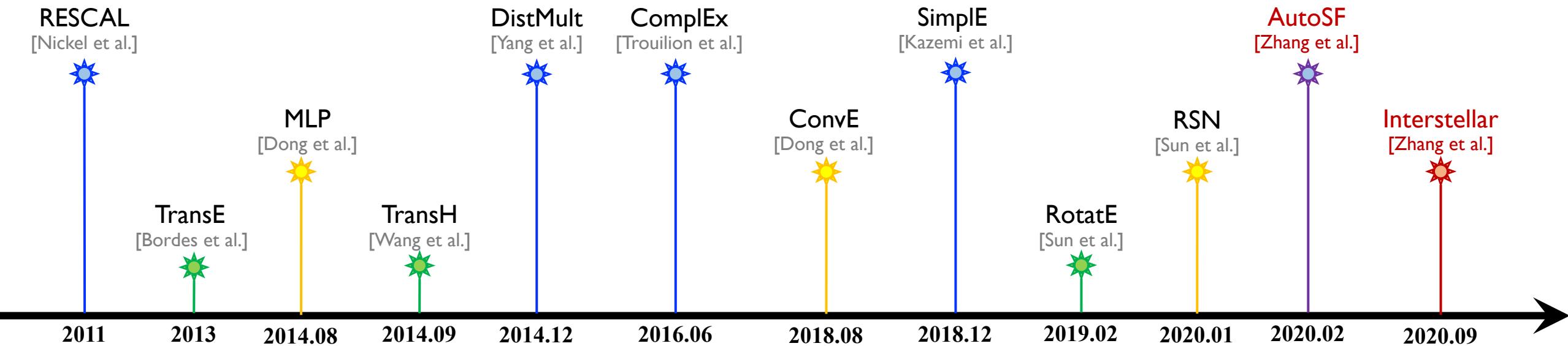
4. Summary

Learning problem in KGE

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



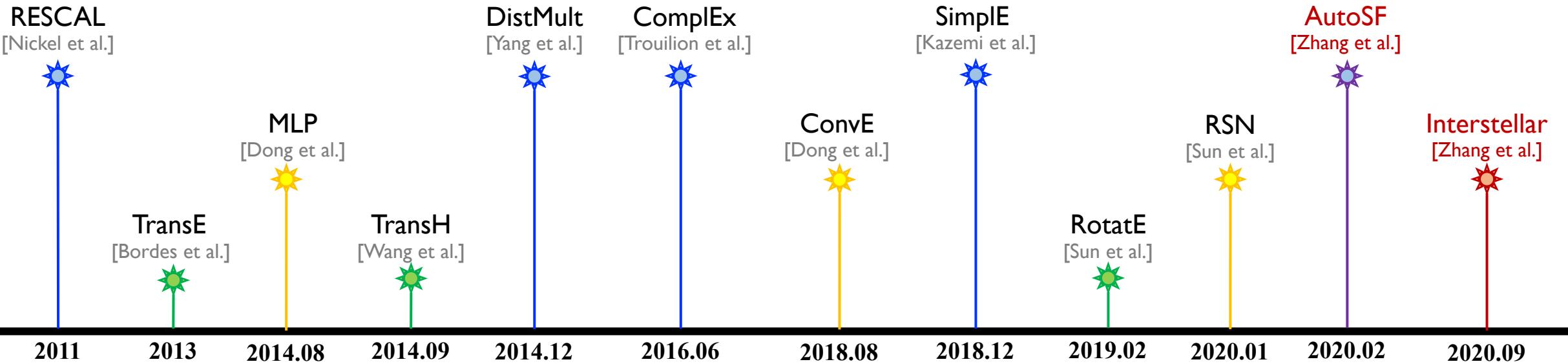
Scoring function



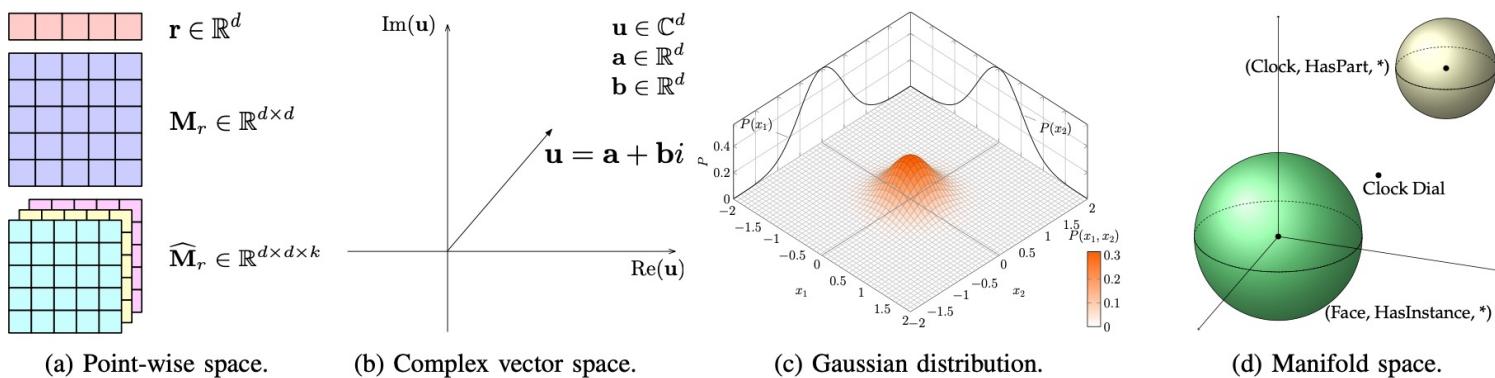
Design patterns:

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

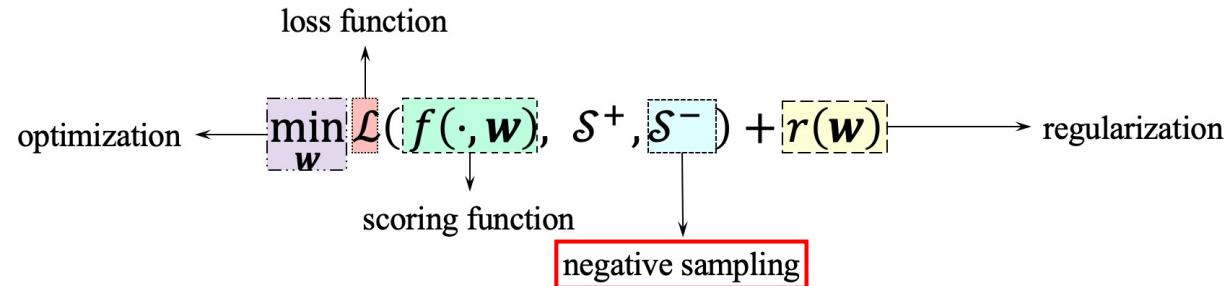
Scoring function



Embedding spaces



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:

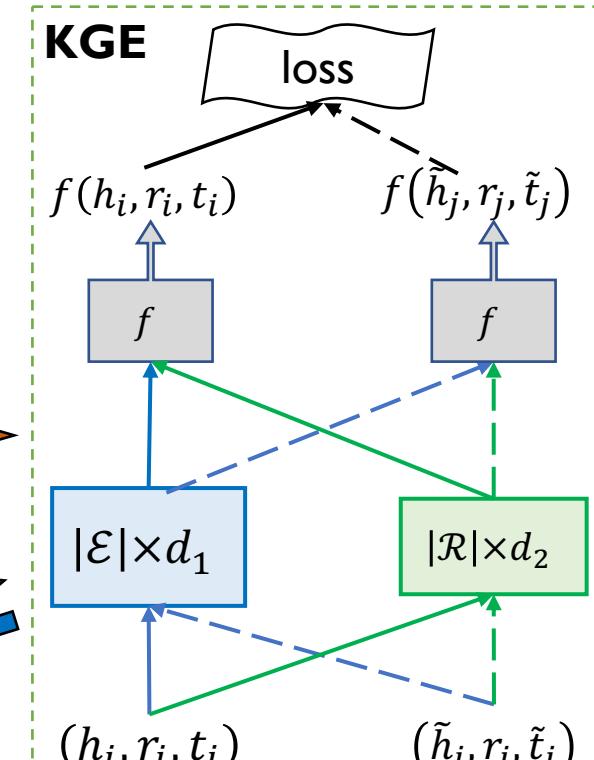
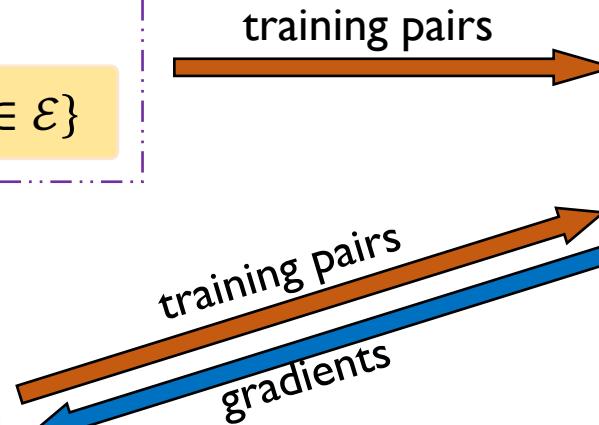
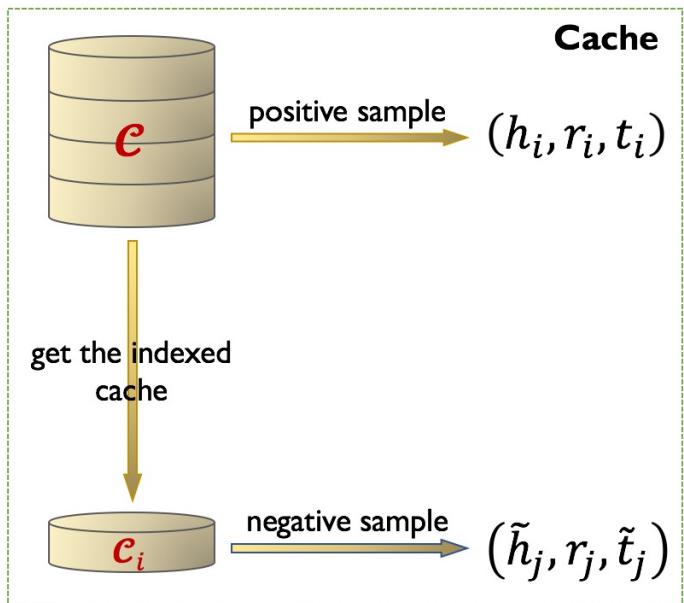
(h, r, t)

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

not dynamic, quality is poor.

Cache-based:

[Zhang et al. ICDE 2019]

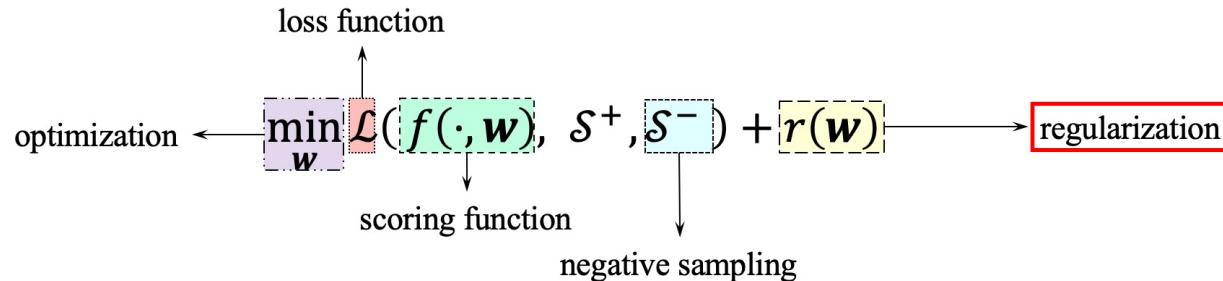


GAN-based:

[Wang et. al.
AAAI 2018]

[Cai et. al.
NAACL 2018]

Regularization



- The models may overfit on the training data.

expressiveness $\xleftarrow{\text{trade off}}$ complexity

- General approaches:

- Directly constrain the embedding norm

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

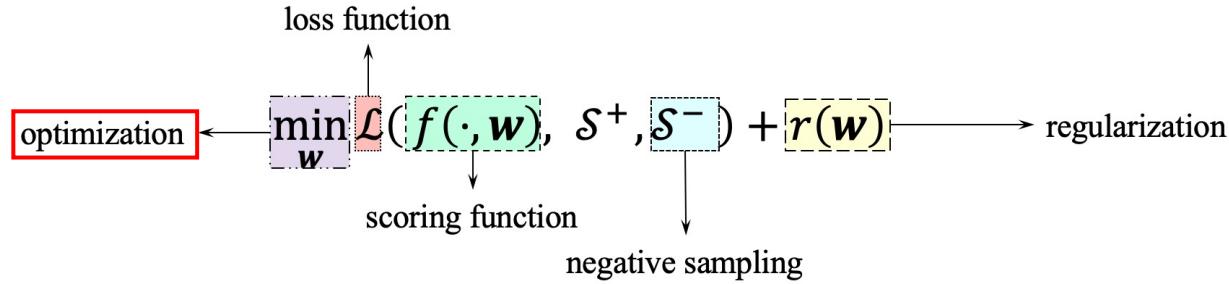
- Add L_p regularizers

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

- Dropout on the embeddings



Optimization



- Parameters 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 ;

- 1: initialize the embedding parameters \mathbf{e}, \mathbf{r} for each $e \in \mathcal{E}$ and $r \in \mathcal{R}$.
 - 2: **for** $i = 1, \dots, T$ **do**
 - 3: sample a mini-batch $\mathcal{S}_{\text{batch}} \in \mathcal{S}$ of size m ;
 - 4: **for** each $(h, r, t) \in \mathcal{S}_{\text{batch}}$ **do**
 - 5: 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) ;
 - 6: 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 ;
 - 7: update embeddings based on loss ℓ using selected positive and negative triplets;
 - 8: **end for**
 - 9: **end for**
 - 10: **return** embeddings of entities in \mathcal{E} and relations in \mathcal{R} .
-

AutoML – Recall



I. 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

Search Space

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

s. t.

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

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

2. Design or select proper search algorithm

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

AutoML – KGE



Search space:

hyper-parameter, scoring function

Search objective:

ranking-based metrics

Search constraint:

number of model training

Training objective:

loss function

Search algorithm:

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

Outline

I. Background

- KG for RecSys
- KG embedding

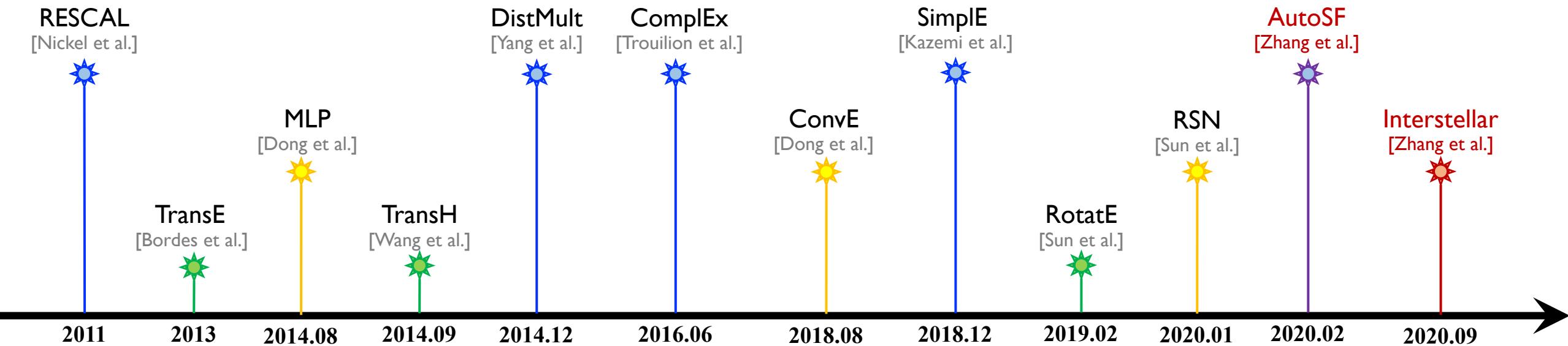
2. Important aspects in KGE

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...
- Too many existing ones, hard to design.

Outline

I. Background

- KG for RecSys
- KG embedding

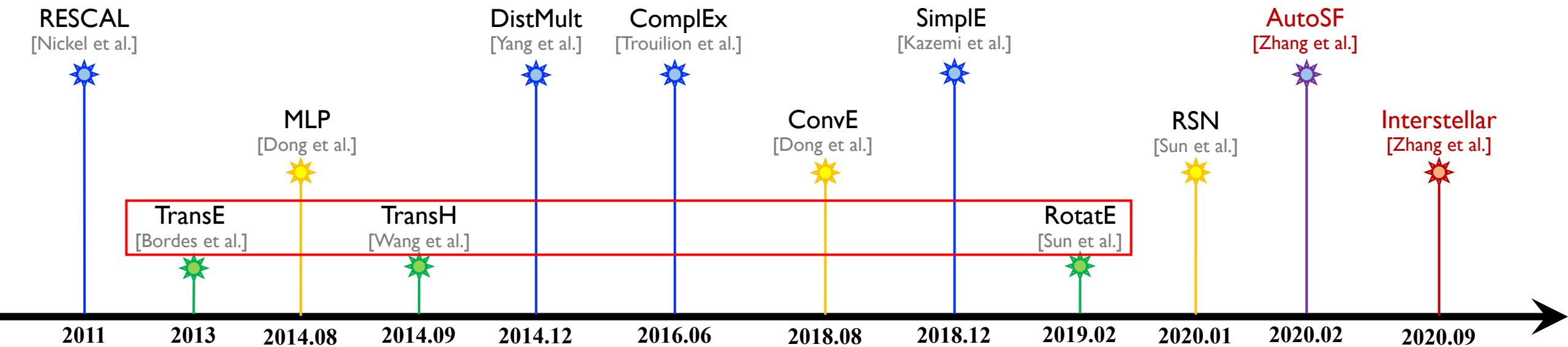
2. Important aspects in KGE

3. Scoring function design

- Overview
- AutoSF

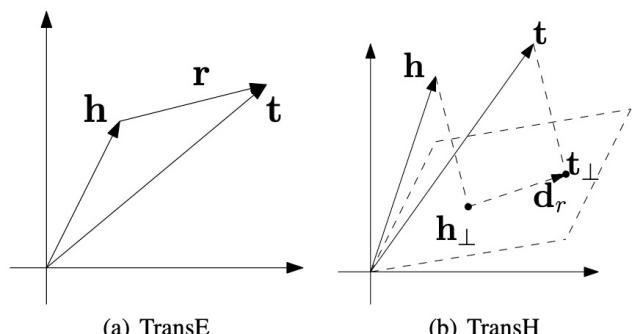
4. Summary

Translational Distance Model (TDM)

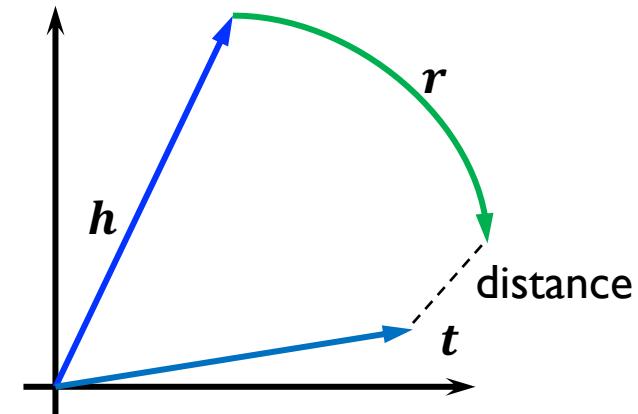


$$\text{TransE: } f(h, r, t) = -\|(\mathbf{h} + \mathbf{r}) - \mathbf{t}\|_p$$

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

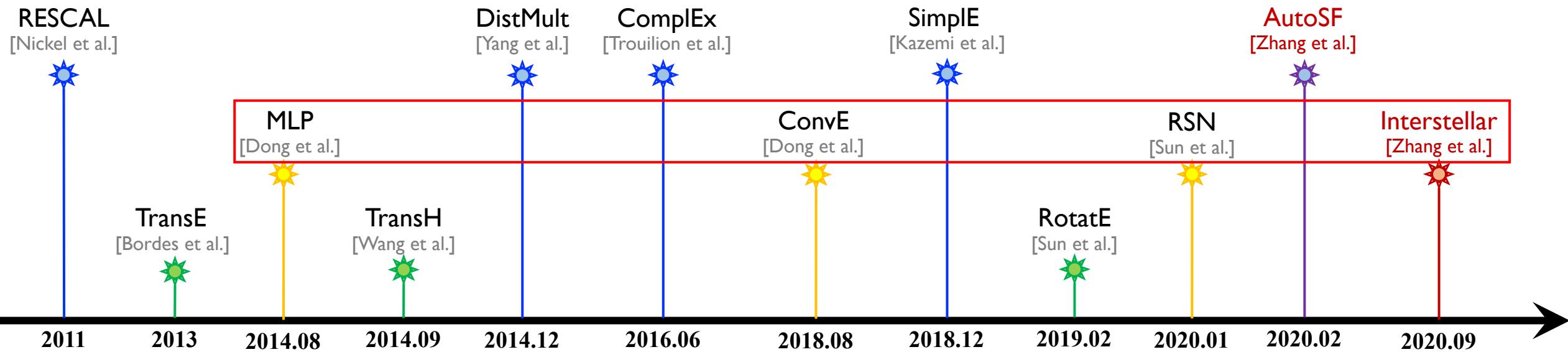


$$\text{RotatE: } f(h, r, t) = -\|(\mathbf{h} \circ \mathbf{r}) - \mathbf{t}\|_p$$

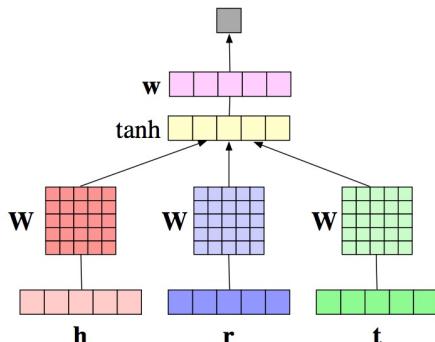


less expressive

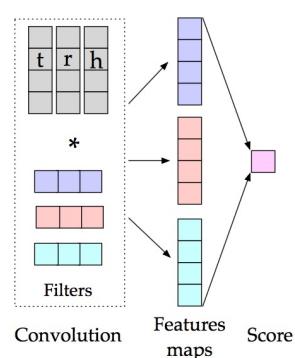
Neural Network Model (NNM)



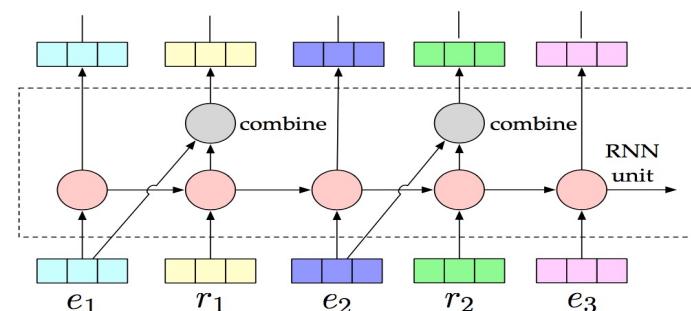
MLP



ConvE

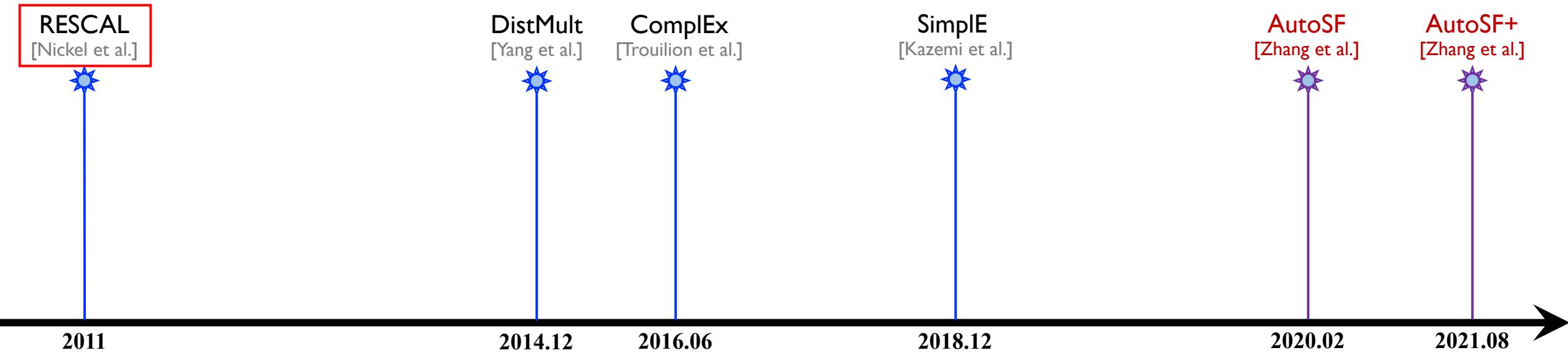


RSN



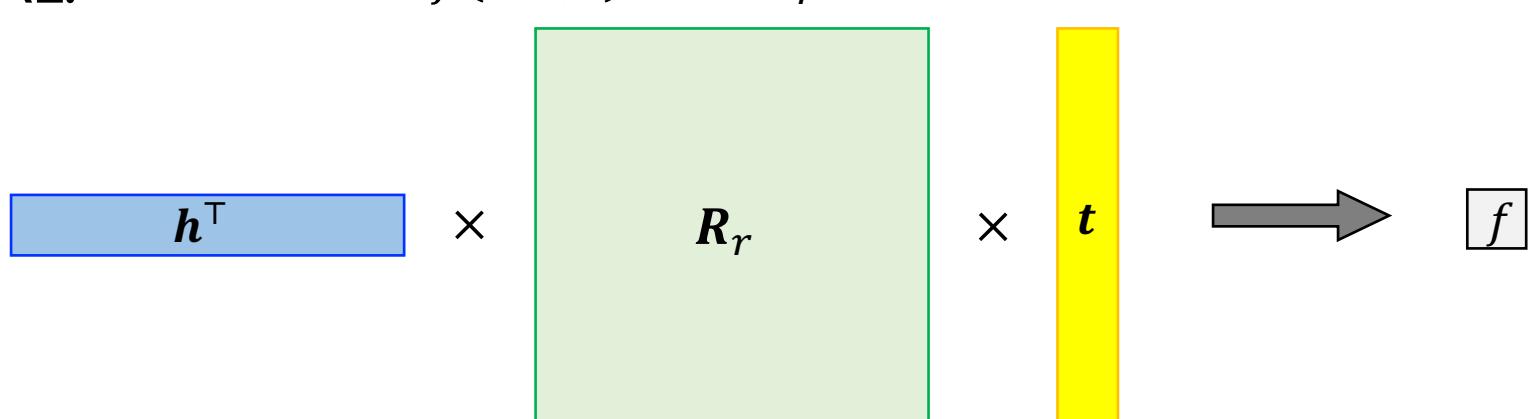
complex
and
difficult to train

BiLinear Model (BLM)

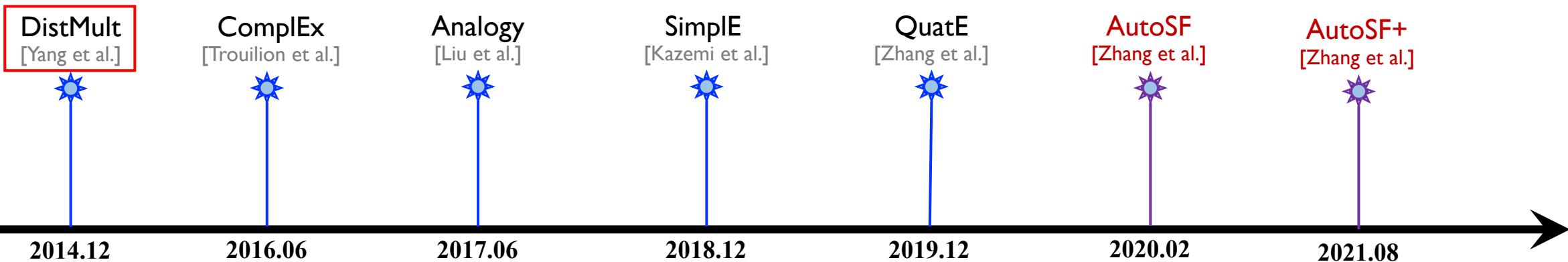


RESCAL:

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

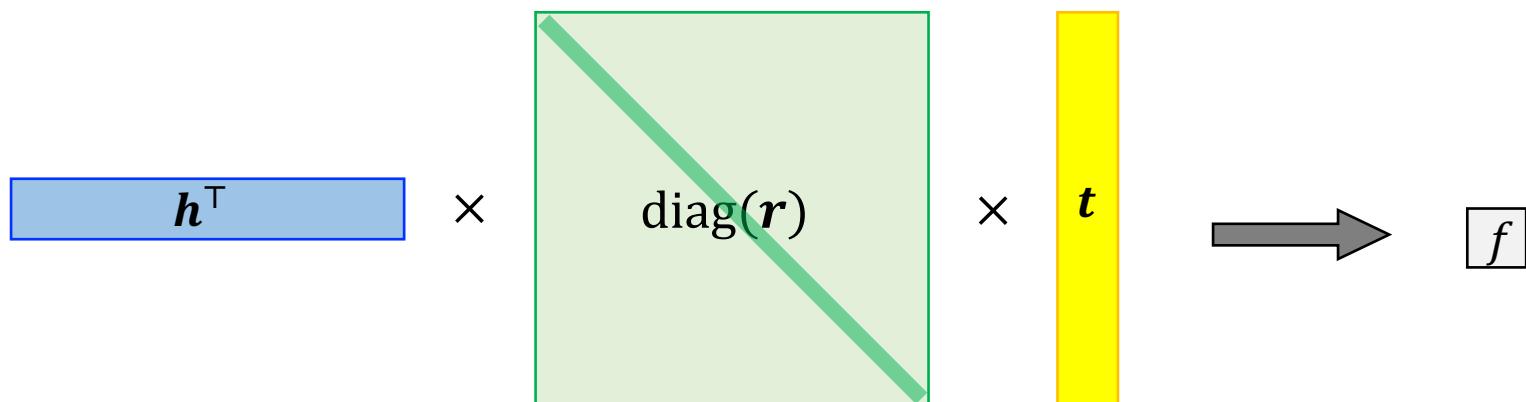


DistMult

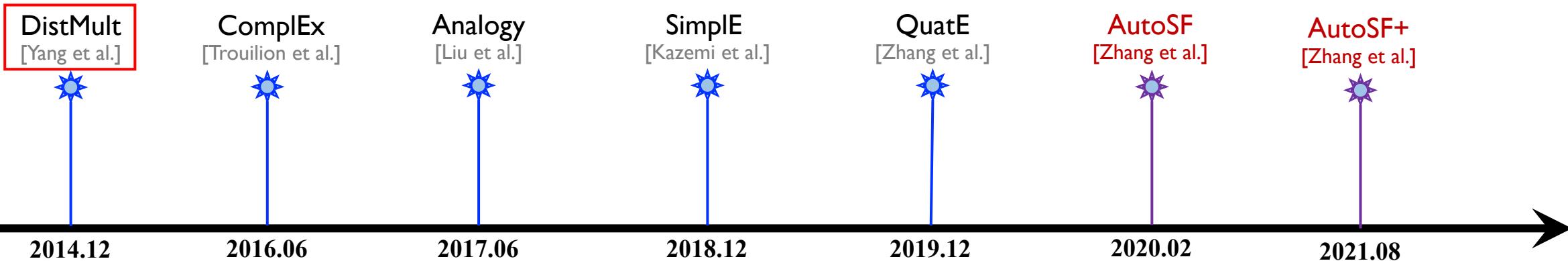


DistMult:

$$f(\mathbf{h}, \mathbf{r}, \mathbf{t}) = \mathbf{h}^\top \text{diag}(\mathbf{r}) \mathbf{t} = \langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle$$

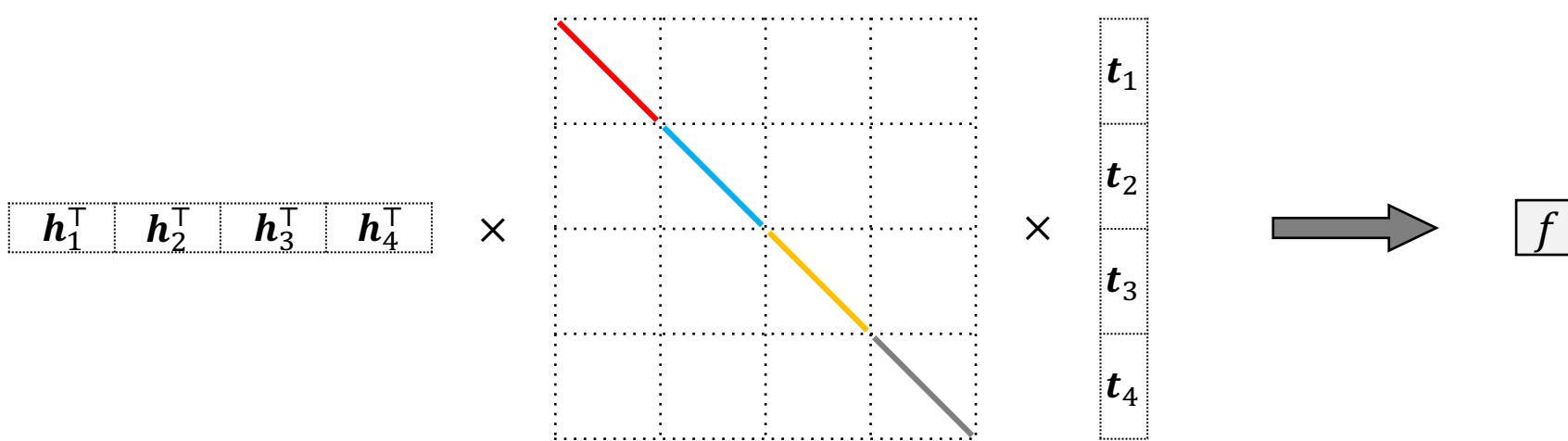


DistMult – splitition

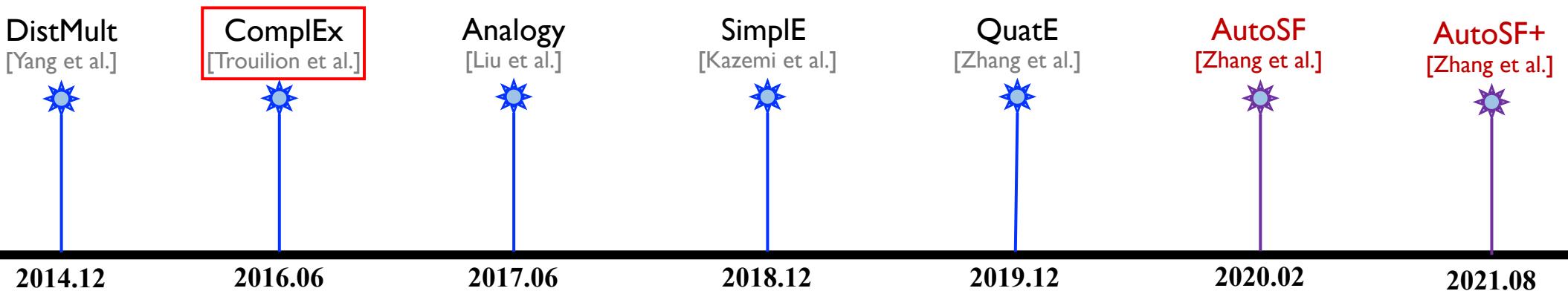


DistMult:

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

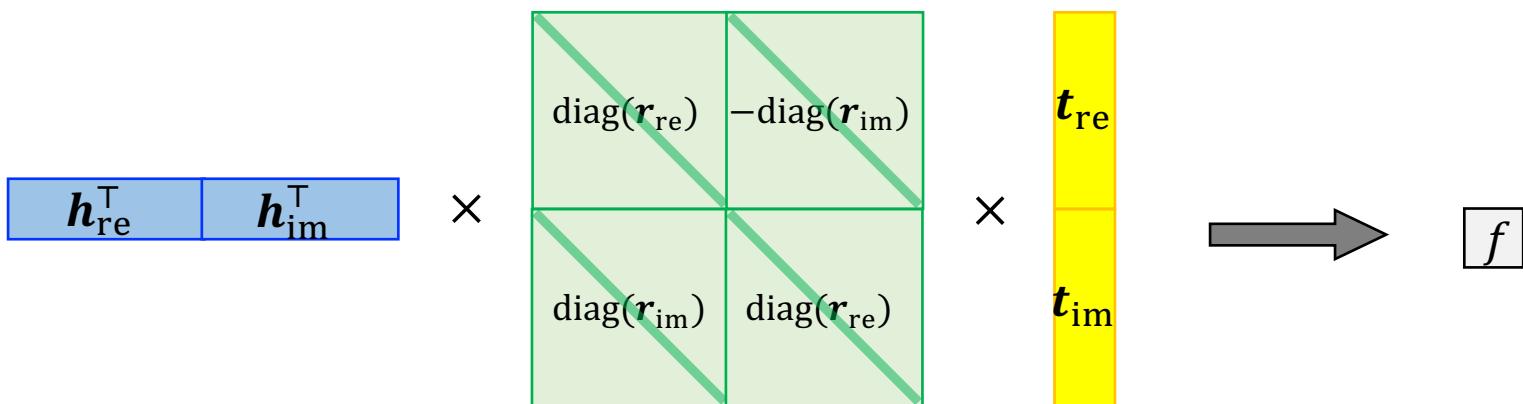


ComplEx

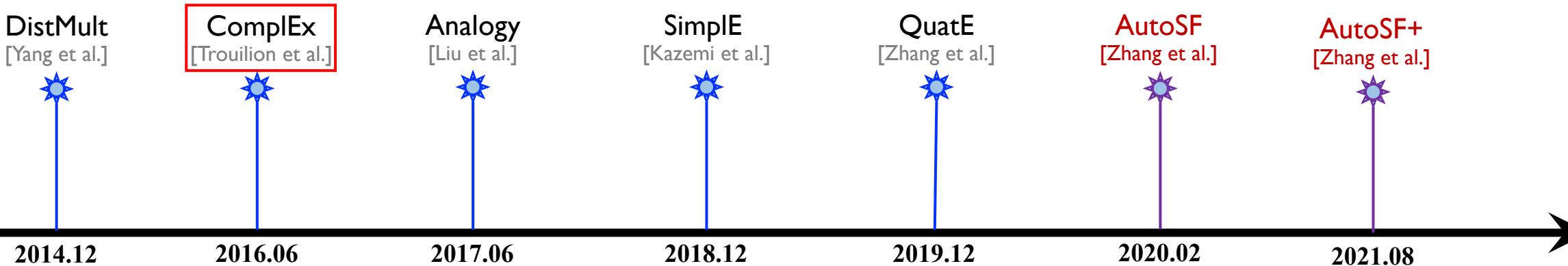


ComplEx:

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

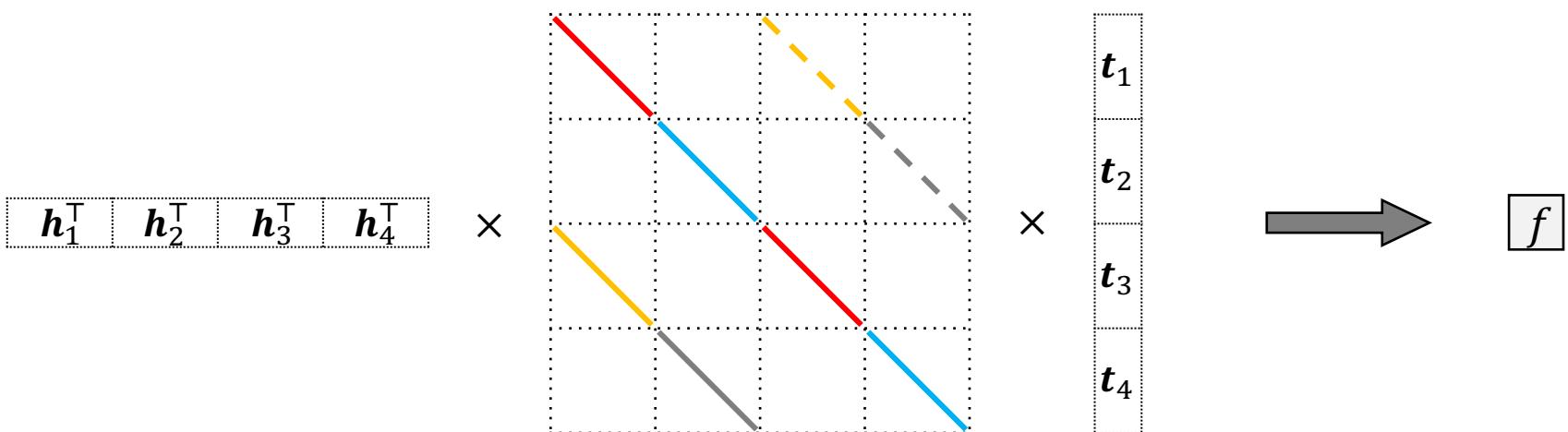


ComplEx – splition

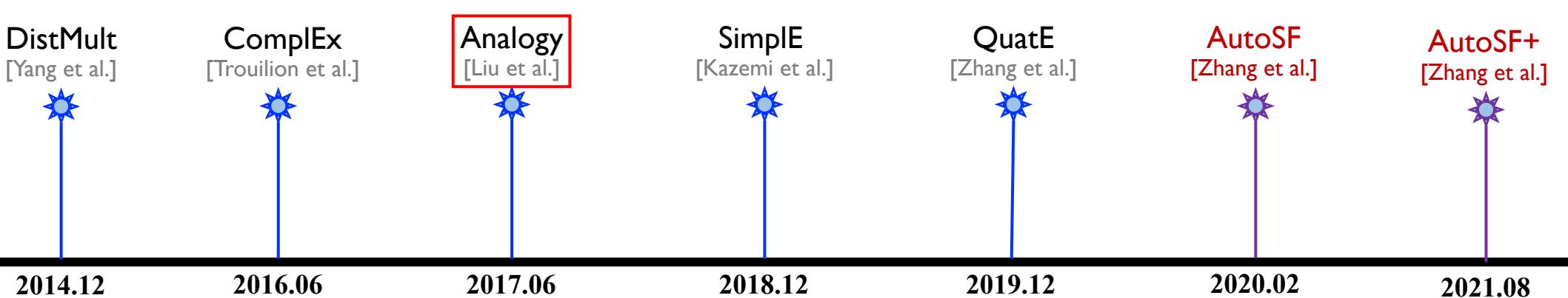


ComplEx:

$$\begin{aligned} f(h, r, t) &= \langle \mathbf{h}_{\text{re}}, \mathbf{r}_{\text{re}}, \mathbf{t}_{\text{re}} \rangle + \langle \mathbf{h}_{\text{im}}, \mathbf{r}_{\text{re}}, \mathbf{t}_{\text{im}} \rangle + \langle \mathbf{h}_{\text{re}}, \mathbf{r}_{\text{im}}, \mathbf{t}_{\text{im}} \rangle - \langle \mathbf{h}_{\text{im}}, \mathbf{r}_{\text{im}}, \mathbf{t}_{\text{re}} \rangle \\ &= \langle \mathbf{h}_1, \mathbf{r}_1, \mathbf{t}_1 \rangle + \langle \mathbf{h}_2, \mathbf{r}_2, \mathbf{t}_2 \rangle + \langle \mathbf{h}_3, \mathbf{r}_1, \mathbf{t}_3 \rangle + \langle \mathbf{h}_4, \mathbf{r}_2, \mathbf{t}_4 \rangle \\ &\quad + \langle \mathbf{h}_1, \mathbf{r}_3, \mathbf{t}_3 \rangle + \langle \mathbf{h}_2, \mathbf{r}_4, \mathbf{t}_4 \rangle - \langle \mathbf{h}_3, \mathbf{r}_3, \mathbf{t}_1 \rangle - \langle \mathbf{h}_4, \mathbf{r}_4, \mathbf{t}_2 \rangle \end{aligned}$$

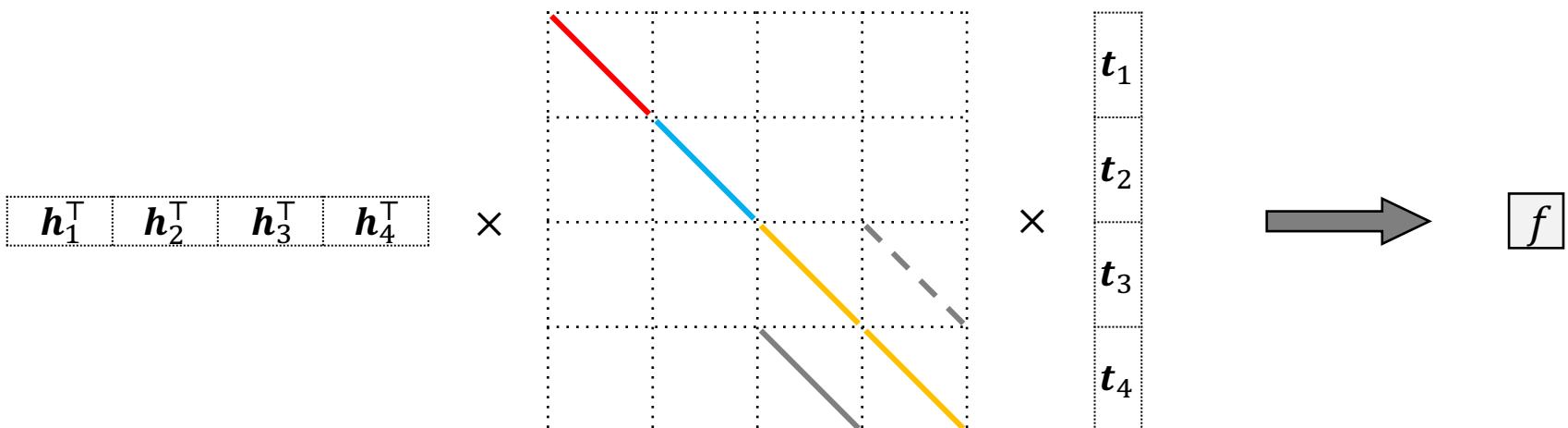


Analogy – splitition

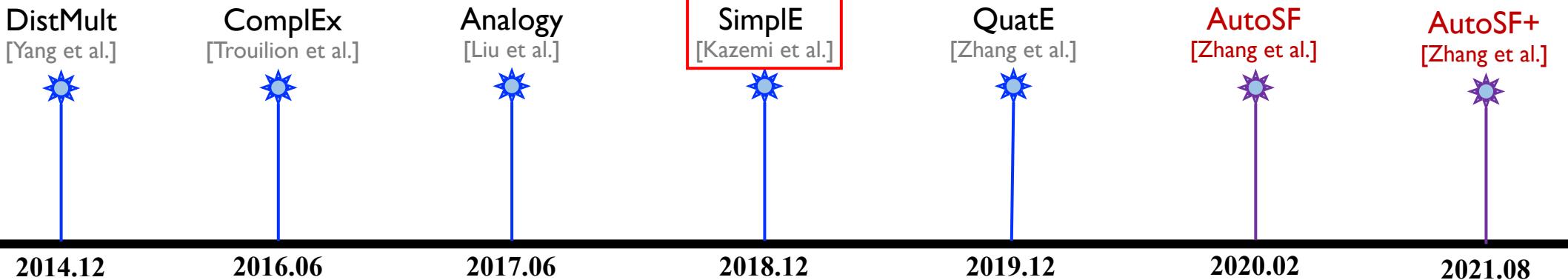


Analogy:

$$\begin{aligned} f(h, r, t) &= \langle \hat{h}, \hat{r}, \hat{t} \rangle + \text{Re}(\langle \underline{h}, \underline{r}, \underline{t} \rangle) \\ &= \langle h_1, r_1, t_1 \rangle + \langle h_2, r_2, t_2 \rangle \\ &\quad + \langle h_3, r_3, t_3 \rangle + \langle h_4, r_4, t_4 \rangle + \langle h_4, r_3, t_4 \rangle - \langle h_4, r_4, t_3 \rangle \end{aligned}$$

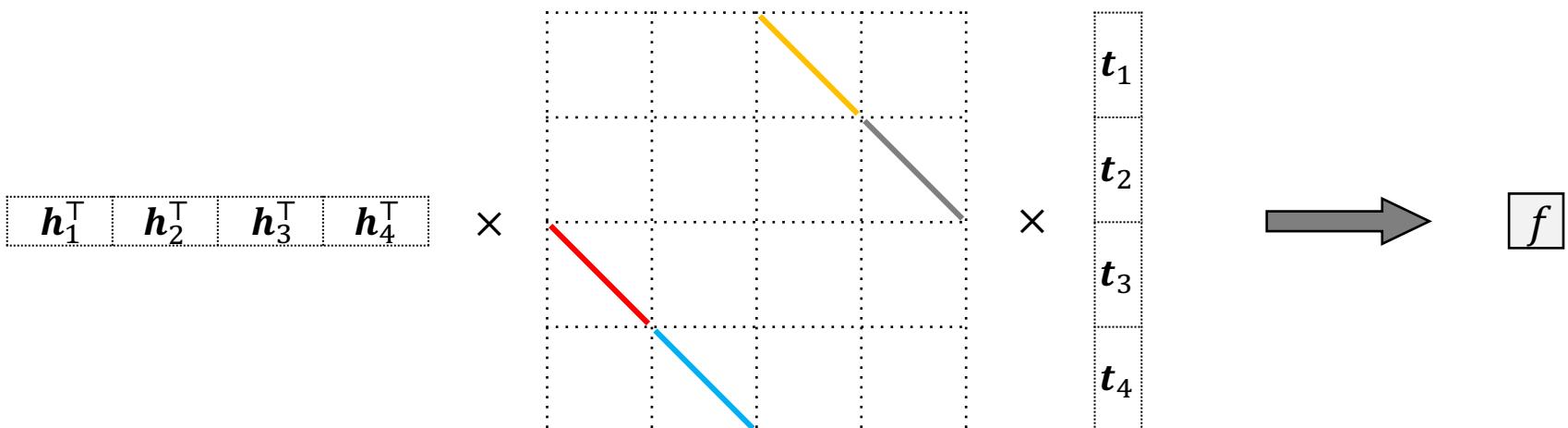


SimPL-E – splition



SimPL-E:

$$\begin{aligned} f(h, r, t) &= \langle \hat{h}, \hat{r}, \underline{t} \rangle + \langle \underline{t}, \underline{r}, \hat{h} \rangle \\ &= \langle h_1, r_1, t_3 \rangle + \langle h_2, r_2, t_4 \rangle + \langle h_3, r_3, t_1 \rangle + \langle h_4, r_4, t_2 \rangle \end{aligned}$$



QuatE

DistMult [Yang et al.]	ComplEx [Trouillon et al.]	Analogy [Liu et al.]	Simple [Kazemi et al.]	QuatE [Zhang et al.]	AutoSF [Zhang et al.]	AutoSF+ [Zhang et al.]
---------------------------	-------------------------------	-------------------------	---------------------------	-------------------------	--------------------------	---------------------------

2014.12

2016.06

2017.06

2018.12

2019.12

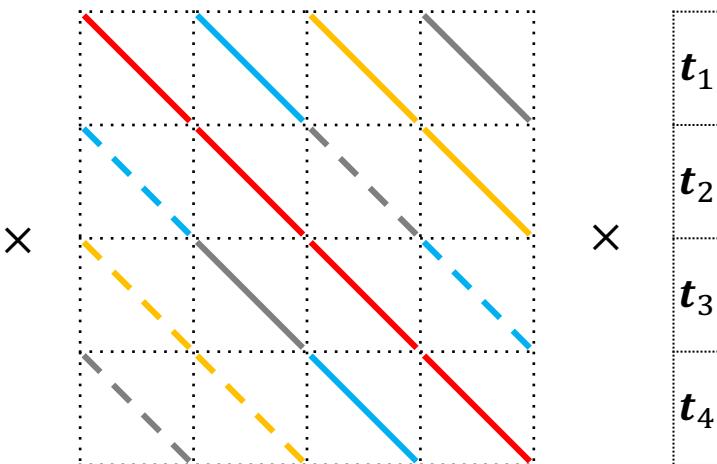
2020.02

2021.08

QuatE:

$$\begin{aligned}
 f(h, r, t) = & \langle h_1, r_1, t_1 \rangle - \langle h_1, r_2, t_2 \rangle - \langle h_1, r_3, t_3 \rangle - \langle h_1, r_4, t_4 \rangle \\
 & + \langle h_2, r_2, t_1 \rangle + \langle h_2, r_1, t_2 \rangle + \langle h_2, r_4, t_3 \rangle - \langle h_2, r_3, t_4 \rangle \\
 & + \langle h_3, r_3, t_1 \rangle - \langle h_3, r_4, t_2 \rangle + \langle h_3, r_1, t_3 \rangle + \langle h_3, r_2, t_4 \rangle \\
 & + \langle h_4, r_4, t_1 \rangle + \langle h_4, r_3, t_2 \rangle - \langle h_4, r_2, t_3 \rangle + \langle h_4, r_1, t_4 \rangle
 \end{aligned}$$

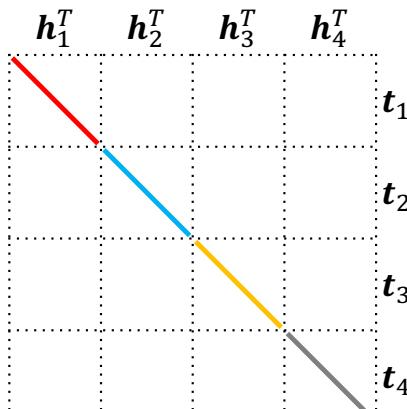
$$\begin{bmatrix} h_1^\top \\ h_2^\top \\ h_3^\top \\ h_4^\top \end{bmatrix}$$



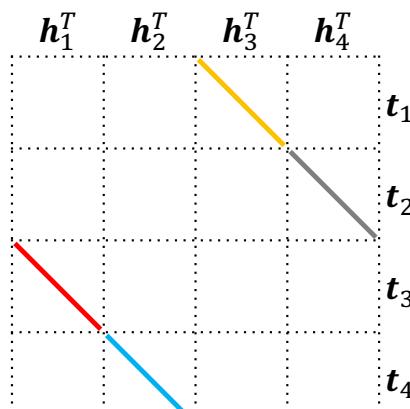
BLMs – unified form

- With unified representation.
- State-of-the-art performance.
- Fully expressive.
- No absolute winner as KGs have different properties.
- How to balance expressiveness and generalization.

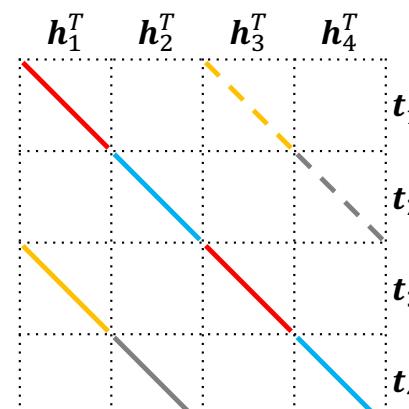
searching!



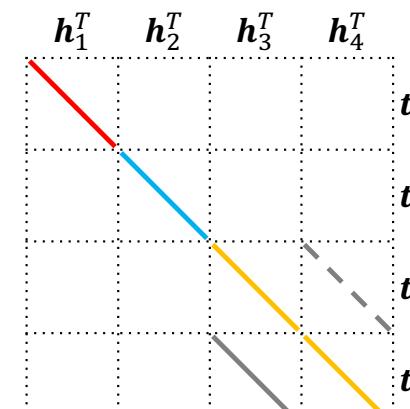
(a) DistMult.



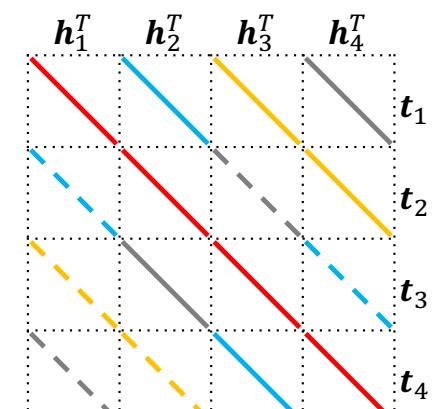
(b) SimplE.



(c) ComplEx.



(d) Analogy.



(e) QuatE.

Outline

I. Background

- KG for RecSys
- KG embedding

2. Important aspects in KGE

3. Scoring function design

- Overview
- AutoSF

4. Summary

Search space

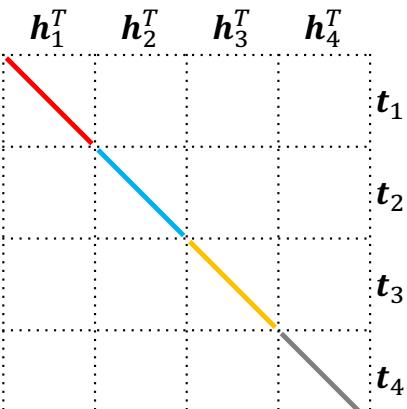
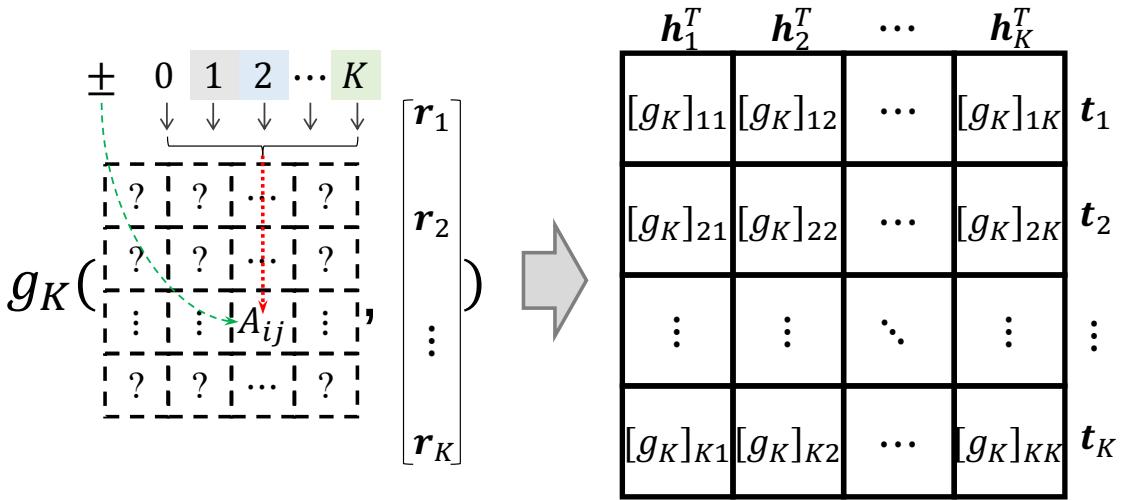
Definition 3 (Unified BiLinear Model). *The desired scoring function is of the form*

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

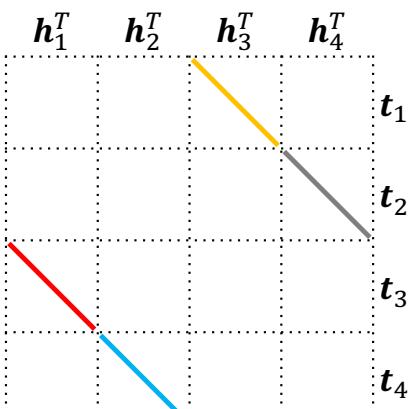
where

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

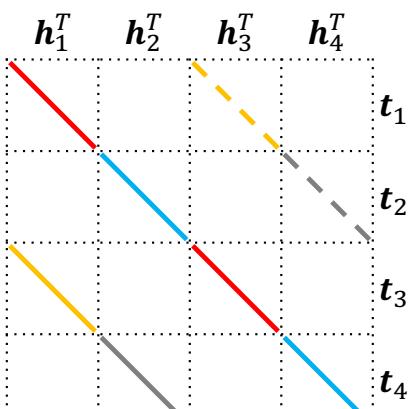
is called the structure matrix. Here, we define $\mathbf{r}_0 \equiv \mathbf{0}$, and $\text{sign}(0) = 0$.



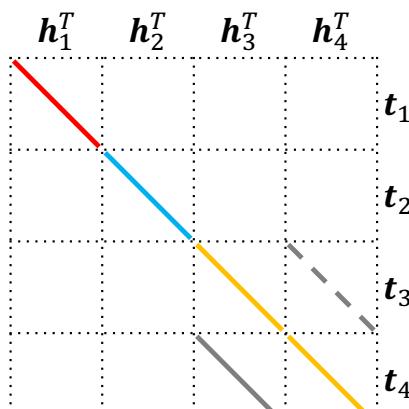
(a) DistMult.



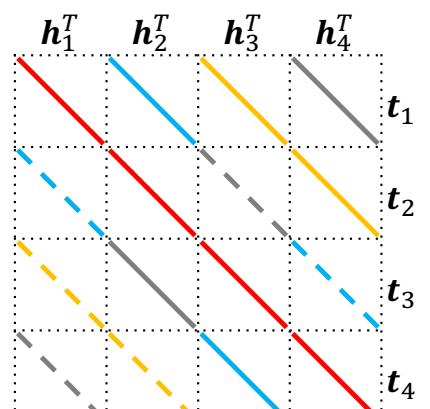
(b) SimplE.



(c) ComplEx.



(d) Analogy.



(e) QuatE.

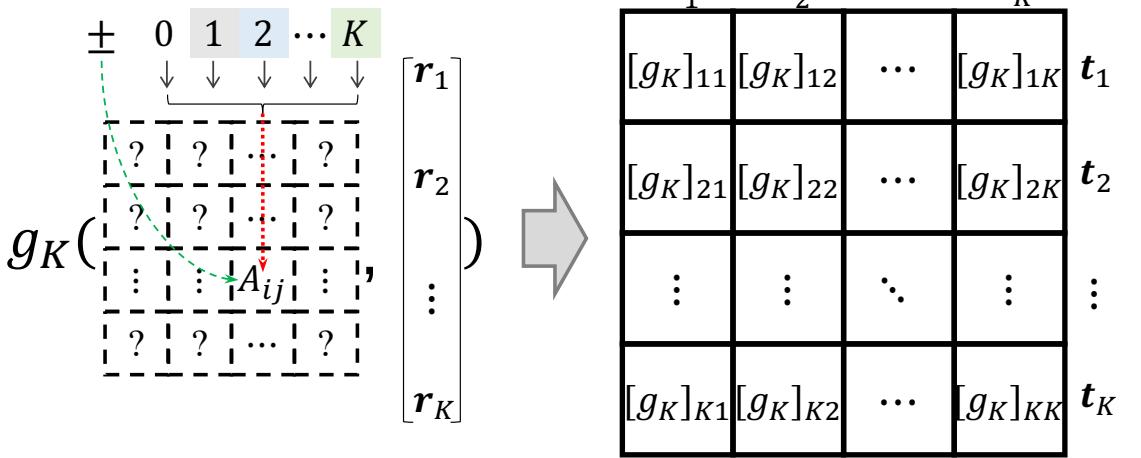
Search problem

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

Definition 4 (Bilinear Model Search (AutoBLM)). Let $F(\mathbf{P}; \mathbf{A})$ be a KG embedding model (where \mathbf{P} includes the entity embedding matrix \mathbf{E} and relation embedding matrix \mathbf{R} , and \mathbf{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:

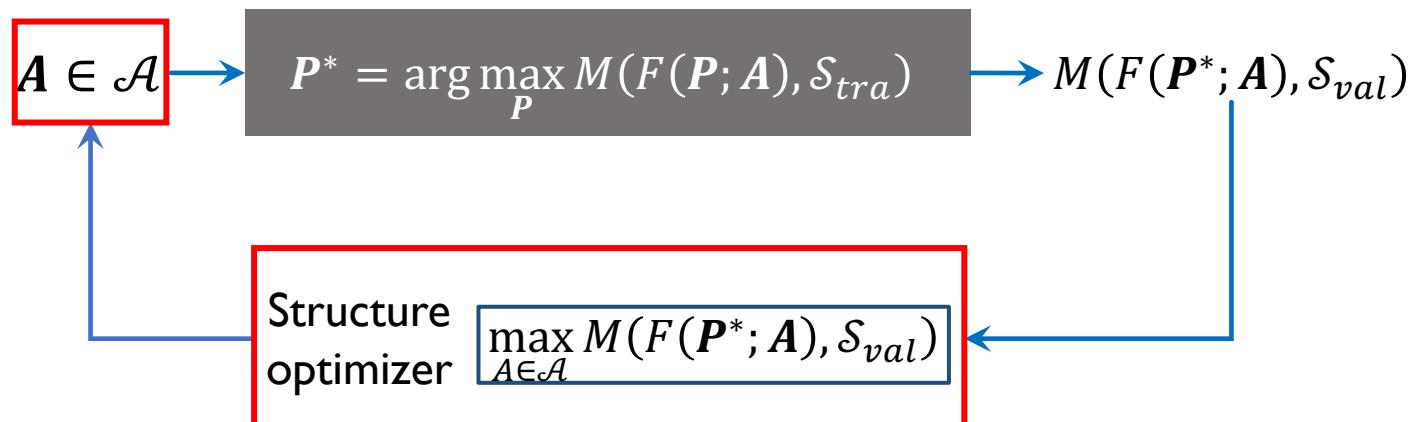
$$\mathbf{A}^* \in \operatorname{Arg\,max}_{\mathbf{A} \in \mathcal{A}} M(F(\mathbf{P}^*; \mathbf{A}), \mathcal{S}_{val}) \quad (9)$$

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



Search space:
What to be searched

Validation
performance



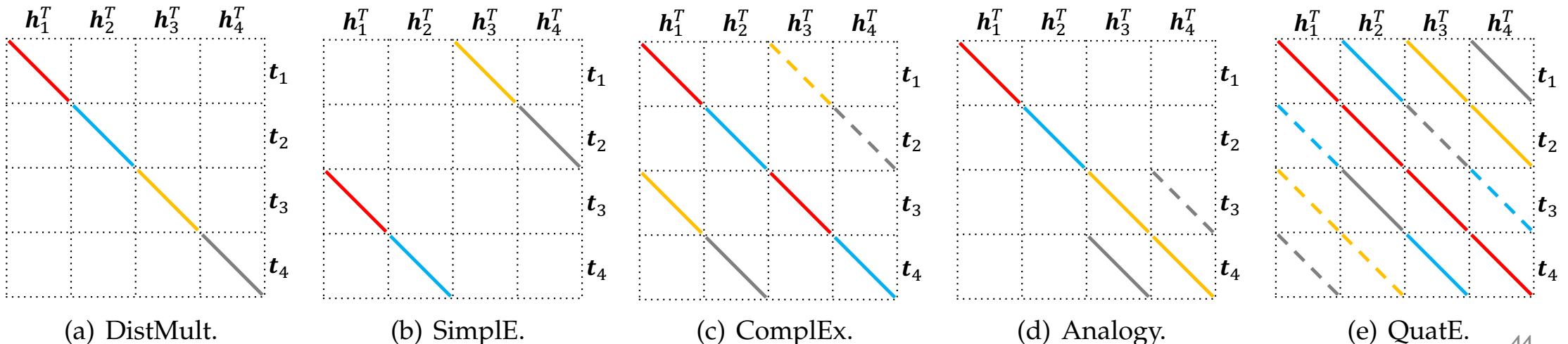
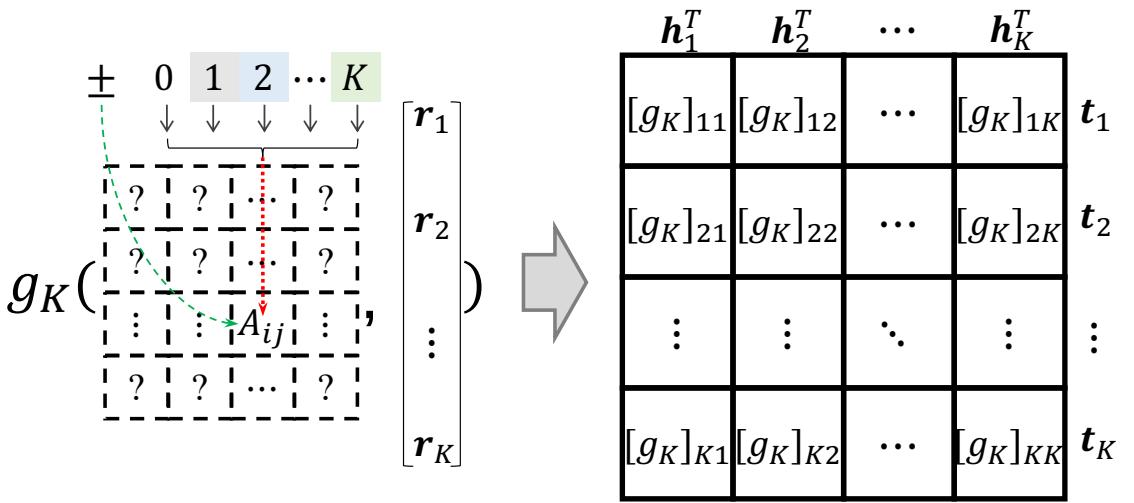
Fully expressive

Proposition 1. Let

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

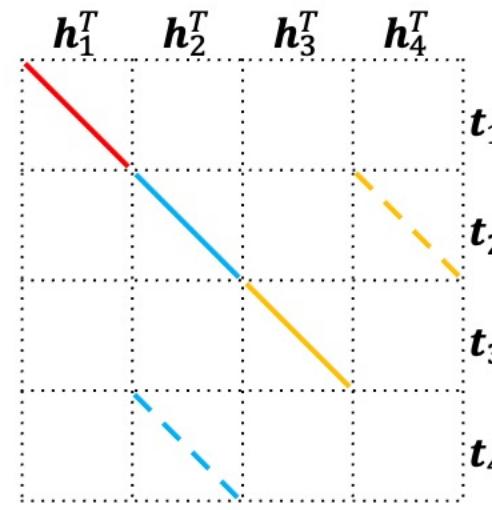
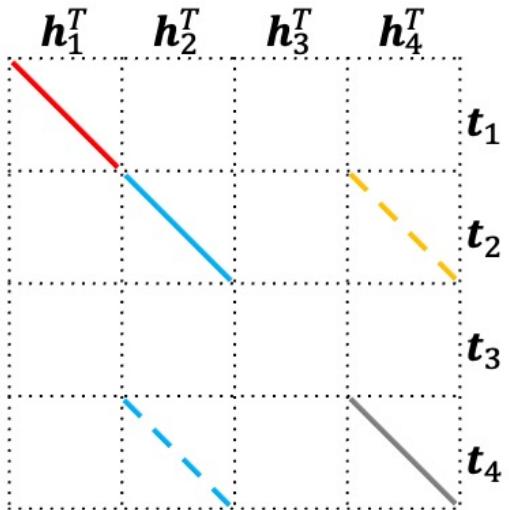
Given an \mathbf{A} in (5), the bilinear model with scoring function (7) is fully expressive if

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



Degenerate structures

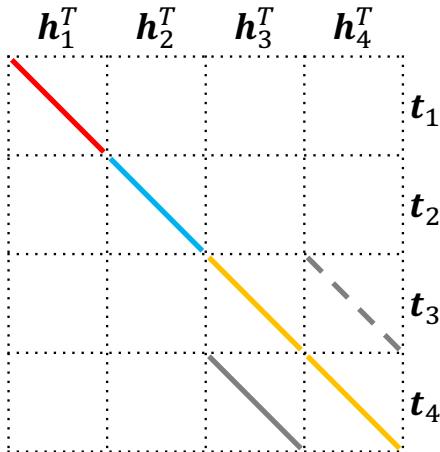
Definition 5 (Degenerate structure). *Matrix A is degenerate if (i) there exists $\mathbf{h} \neq \mathbf{0}$ such that $\mathbf{h}^\top g_K(A, \mathbf{r})\mathbf{t} = 0, \forall \mathbf{r}, \mathbf{t}$; or (ii) there exists $\mathbf{r} \neq \mathbf{0}$ such that $\mathbf{h}^\top g_K(A, \mathbf{r})\mathbf{t}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



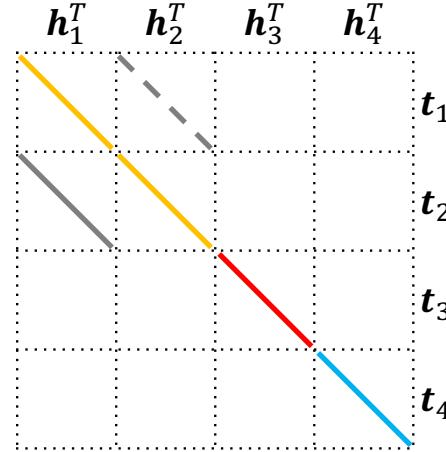
Proposition 2. *A is not degenerate if and only if $\text{rank}(A) = K$ and $\{1, \dots, K\} \subset \{|A_{ij}| : i, j = 1, \dots, K\}$.*

Equivalent structures

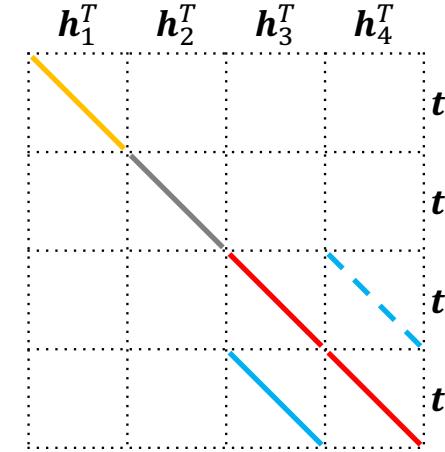
Definition 6 (Equivalence). Let $\mathbf{P}^* = \arg \max_{\mathbf{P}} M(F(\mathbf{P}; \mathbf{A}), \mathcal{S})$ and $\mathbf{P}'^* = \arg \max_{\mathbf{P}'} M(F(\mathbf{P}'; \mathbf{A}'), \mathcal{S})$. If $\mathbf{A} \neq \mathbf{A}'$ but $M(F(\mathbf{P}^*; \mathbf{A}), \mathcal{S}) = M(F(\mathbf{P}'^*;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



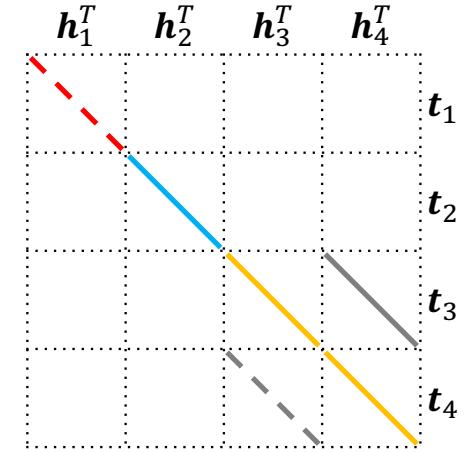
Analogy.



Permuting rows
and columns.

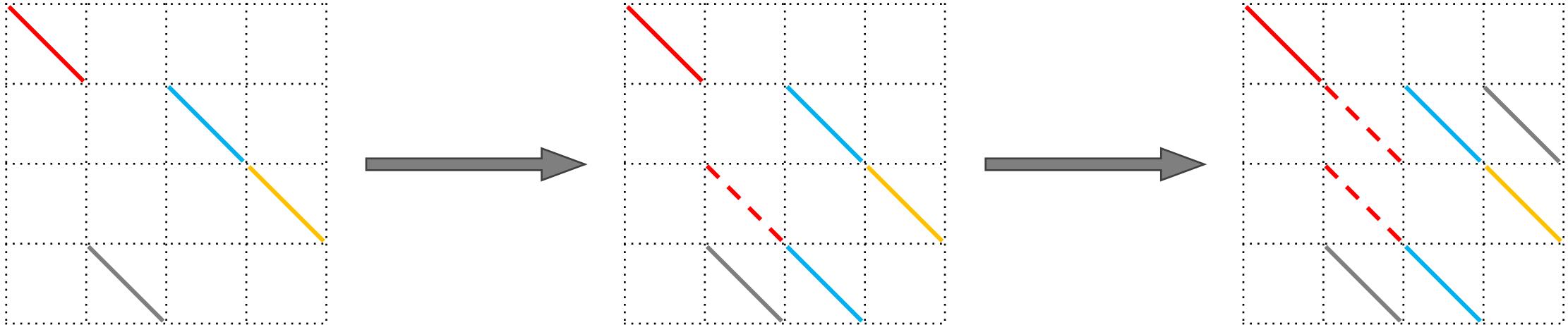


Permuting values.



Flipping signs.

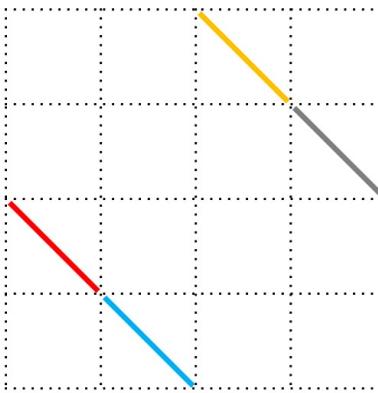
Greedy algorithm – AutoSF



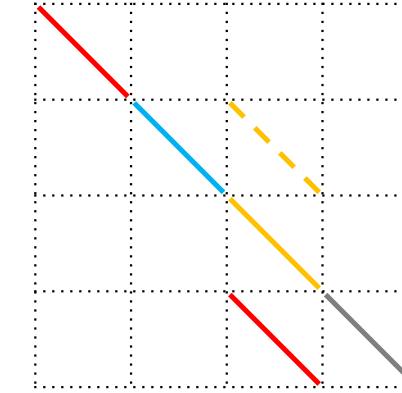
- Progressively adding values and adjusting the search space.
- Greedily preserving the **top- I** structures with b non-zero elements.

Evolutionary algorithm – AutoSF+

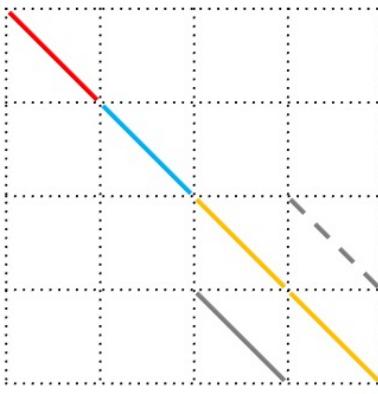
SimpIE:



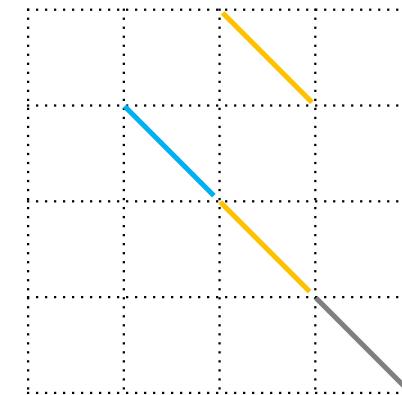
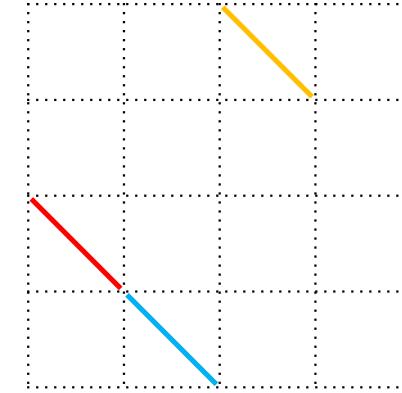
mutation:



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	

- 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.

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

Expressiveness – standard benchmarks

Measurements

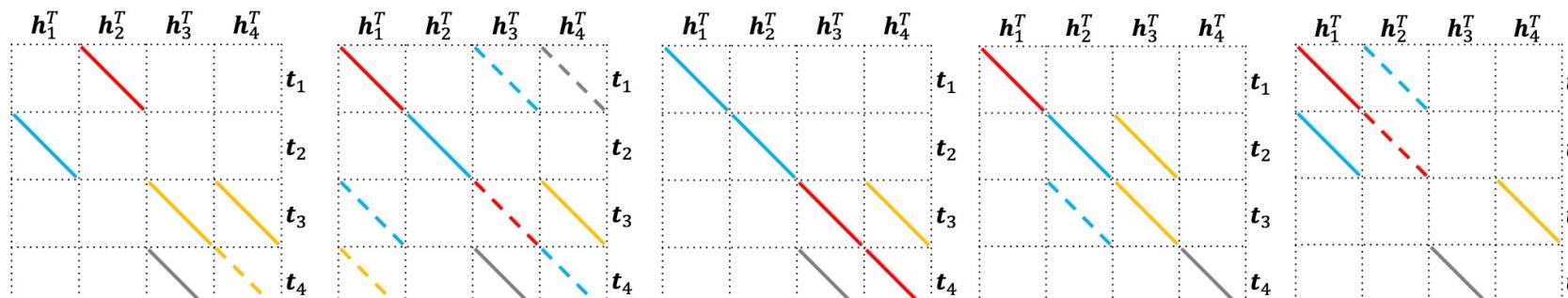
- Given a triplet (h, r, t) ;
- Compute the score of $(h', r, t), \forall h' \in \mathcal{E}$;
- Get the rank of h among all h' ;

Metrics

$$\text{MRR: } \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \frac{1}{\text{rank}_i}$$

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

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	<u>0.952</u>	<u>94.7</u>	96.1	<u>0.853</u>	<u>82.1</u>	<u>91.0</u>	<u>0.490</u>	<u>45.1</u>	56.7	<u>0.360</u>	<u>26.7</u>	<u>55.2</u>	<u>0.571</u>	<u>50.1</u>	71.5	
AutoSF+	<u>0.952</u>	<u>94.7</u>	96.1	0.861	<u>83.2</u>	<u>91.3</u>	0.492	<u>45.2</u>	56.7	0.364	<u>27.0</u>	<u>55.3</u>	<u>0.577</u>	<u>50.2</u>	<u>71.5</u>	



(a) WN18.

(b) FB15k.

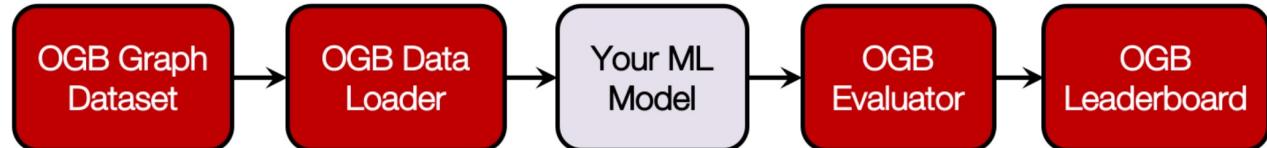
(c) WN18RR.

(d) FB15k237.

(e) YAGO3-10.

OGB – results

- Open graph benchmark:



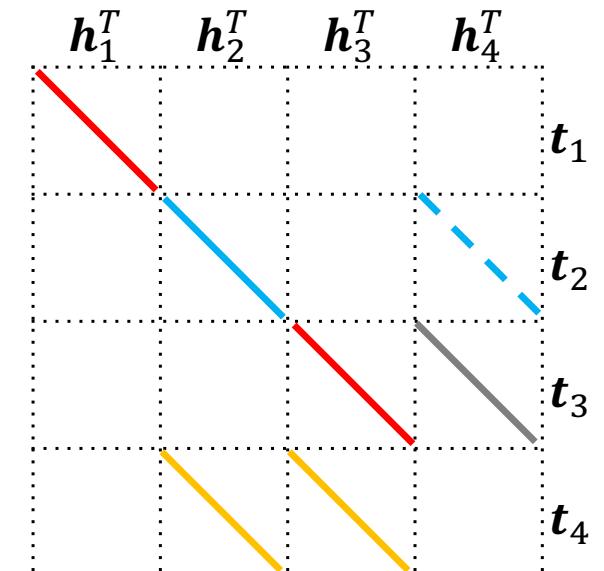
Leaderboard for [ogbl-wikikg2](#)

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	Validation		Contact	References	#Params	Hardware	Date
		Test MRR	MRR					
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
		0.0035	0.0047					
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
		0.0052	0.0063					
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
		0.0027	0.0020					
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
		0.0025	0.0028					
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
		0.0030	0.0030					



OGB – results

- Open graph benchmark:

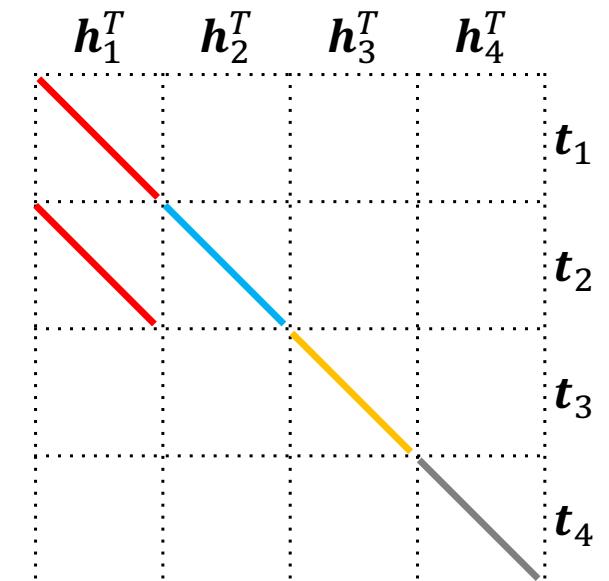


Leaderboard for ogbl-biokg

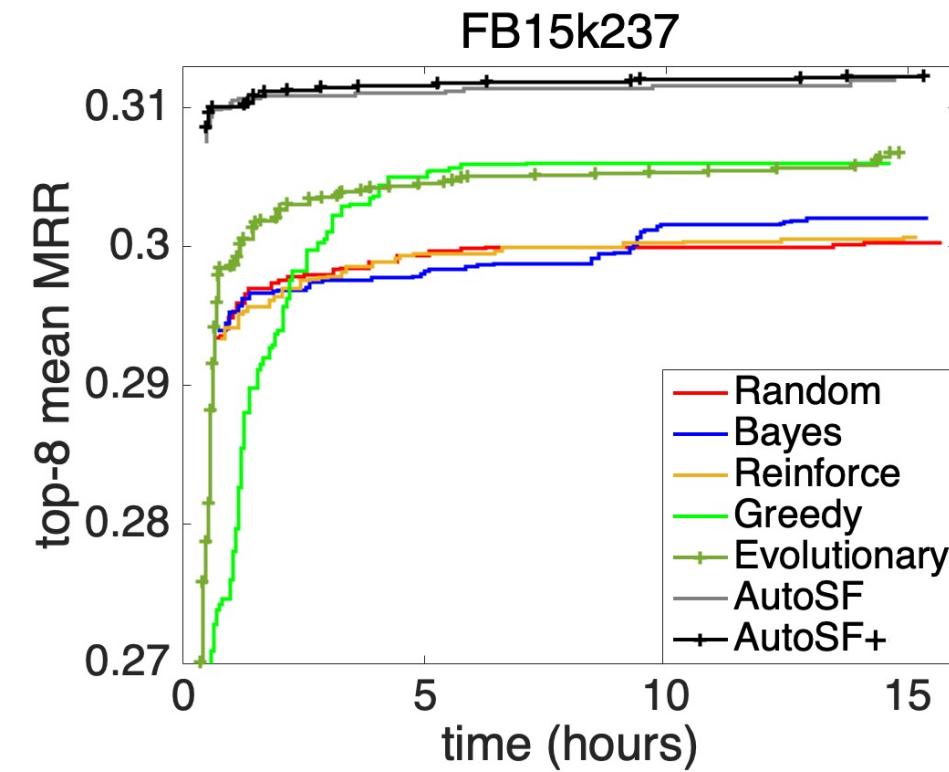
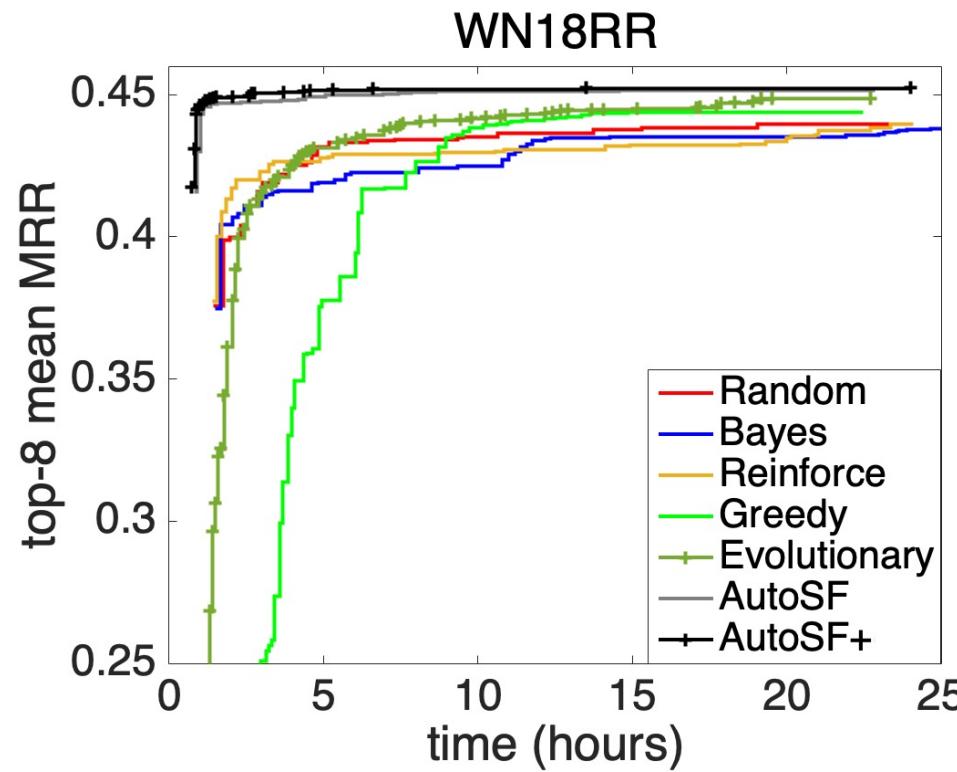
The MRR score on the test and validation sets. The higher, the better.

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.

$$\begin{aligned}\mathbf{e}_i^{\ell+1} &= \sigma(\mathbf{W}_0^\ell \mathbf{e}_i^\ell + \sum_{(r,j):(i,r,j) \in \mathcal{S}_{\text{tra}}} \mathbf{W}^\ell \phi(\mathbf{e}_j^\ell, \mathbf{r}^\ell)), \\ \phi(\mathbf{e}_j^\ell, \mathbf{r}^\ell) &= g_K(\mathbf{A}, \mathbf{r}^\ell) \mathbf{e}_j^\ell\end{aligned}$$

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

I. Background

- KG for RecSys
- KG embedding

2. Important aspects in KGE

3. Scoring function design

- Overview
- AutoSF

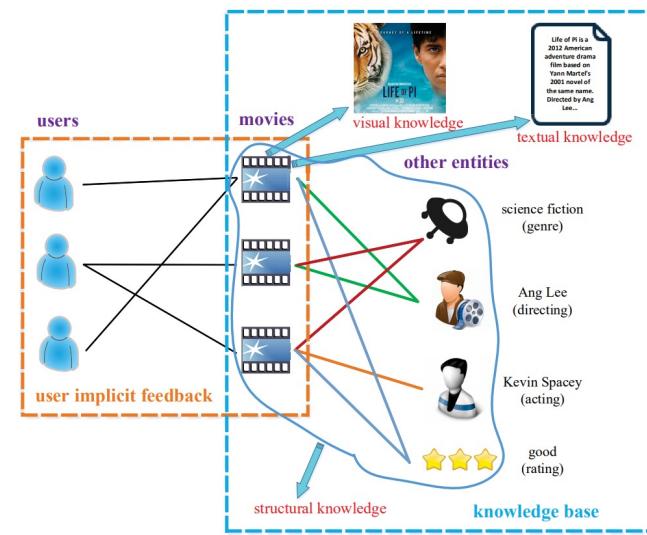
4. Summary

Take away

- KG benefits RecSys 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

- Model search for different auxiliary information.
- Sample search to balance the usage of auxiliary information.
- Scoring function search for user-user, user-item, item-item interaction.



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-4Paradigm>

Thank you!

Site: <https://quanmingyao.github.io/AutoML.github.io/ijcai21-tutorial.html>

Email: zhangyongqi@4paradigm.com