



Reasoning Beyond Triples: Recent Advances in Knowledge Graph Embeddings



Bo Xiong ¹



Mojtaba Nayyeri ¹



Daniel Daza ^{2,3}



Michael Cochez ²

¹ University of Stuttgart ²Vrije Universiteit Amsterdam ³ University of Amsterdam

Saturday 21 October, 09:00 - 17:30 (GMT+1) @ University of Birmingham - Birmingham, UK

Materials and Slack channel

- Material will be online: <https://kg-beyond-triple.github.io/>
- Join our slack channel for Q&A



Raise your hand if you...

- are familiar with **knowledge graphs (KGs)** ?
- are familiar with **knowledge graph embeddings (KGEs)** ?
- are familiar with any **non-triple based KGs**?
 - Temporal KGs
 - Hyper-relational/n-ary KGs
 - Multimodal, literal/numerical, LLMs, etc.
- are familiar with **complex query answering**?

Agenda

Part 1. Introduction (Bo & Mojtaba, 9:00 - 9:30)

- 1.1 Tutorial Overview (Bo)
- 1.2 Introduction to KG Embeddings
- 1.3 Motivation to go beyond triple-based KGs

Part 2. Temporal KG Embeddionsg (Mojtaba, 9:30 – 10:30)

- 2.1 Background
 - 2.2 In sample timestamp KGE
 - 2.3 Out of sample timestamp
- Coffee Break (10:30 – 11:00)*

Part 3. Hyper-relational & N-ary KG embeddings (Bo, 11:00 – 12:30)

- 3.1 Introduction to hyper-relational & N-Ary KGs
 - 3.2 N-ary relational embeddings
 - 3.3 Hyper-relational embeddings
- Lunch Break (12:30 – 14:00)*

Part 4. KG embeddings with literal and text (Daniel, 14:00 – 15:00)

- 4.1 Incorporating entity descriptions
- 4.2 Incorporate numeric literals as features
- 4.3 Incorporating relation descriptions

Part 5. Advanced Topic (Michael, 15:00-15:30 & 16:00 – 16:30)

- 5.1 Complex query embeddings (Daniel)
- Coffee Break (15:30 – 16:00)*
- 5.2 Neural Graph Databases
- 5.3 Inductive learning settings
- 5.4 Incorporating ontology, LLM, etc.

Part 6. Conclusion & Future Works (Michael, 16:30 – 17:00)

- 6.1 Conclusion
- 6.2 Future Works



Part 1. Introduction

Agenda

Part 1. Introduction (Bo & Mojtaba, 9:00 - 9:30)

- 1.1 Tutorial Overview (Bo)
- 1.2 Introduction to KG Embeddings
- 1.3 Motivation to go beyond triple-based KGs

Part 2. Temporal KG Embeddionsg (Mojtaba, 9:30 – 10:30)

- 2.1 Background
 - 2.2 In sample timestamp KGE
 - 2.3 Out of sample timestamp
- Coffee Break (10:30 – 11:00)*

Part 3. Hyper-relational & N-ary KG embeddings (Bo, 11:00 – 12:30)

- 3.1 Introduction to hyper-relational & N-Ary KGs
 - 3.2 N-ary relational embeddings
 - 3.3 Hyper-relational embeddings
- Lunch Break (12:30 – 14:00)*

Part 4. KG embeddings with literal and text (Daniel, 14:00 – 15:00)

- 4.1 Incorporating entity descriptions
- 4.2 Incorporate numeric literals as features
- 4.3 Incorporating relation descriptions

Part 5. Advanced Topic (Michael, 15:00-15:30 & 16:00 – 16:30)

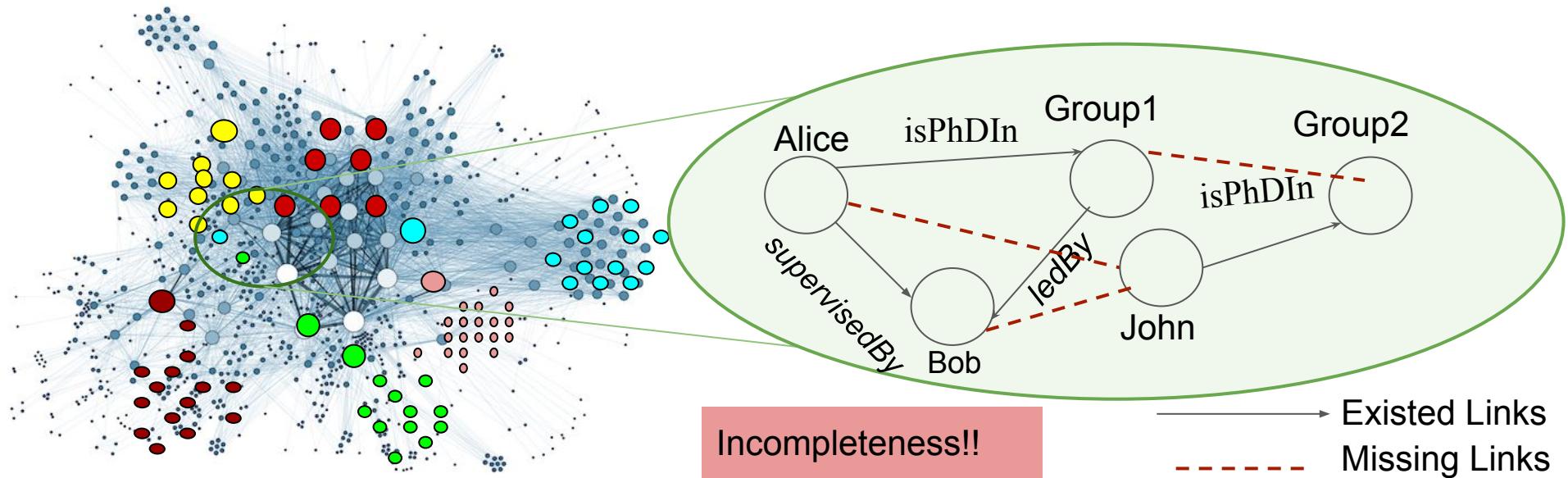
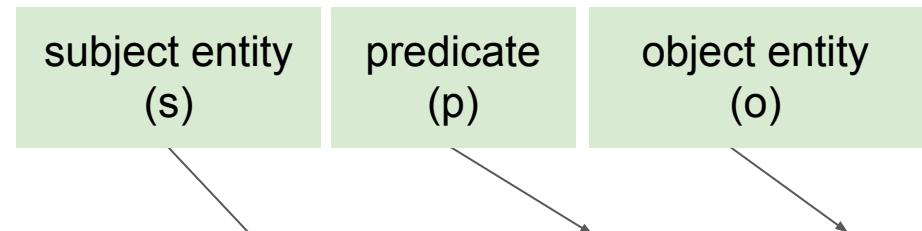
- 5.1 Complex query embeddings (Daniel)
- Coffee Break (15:30 – 16:00)*
- 5.2 Neural Graph Databases
- 5.3 Inductive learning settings
- 5.4 Incorporating ontology, LLM, etc.

Part 6. Conclusion & Future Works (Michael, 16:30 – 17:00)

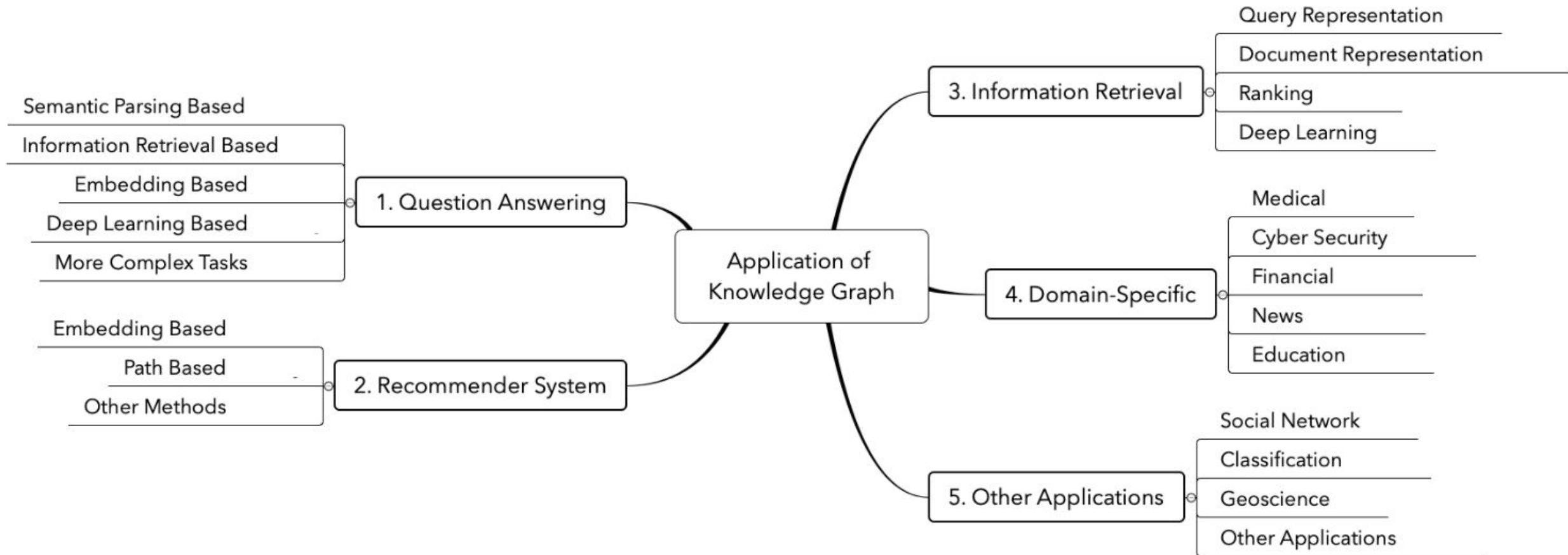
- 6.1 Conclusion
- 6.2 Future Works

Background: Knowledge Graphs

- Symbolic information
- Relational and structured data
- Representation of facts in the form of triples: (Alice, supervisedBy, Bob)



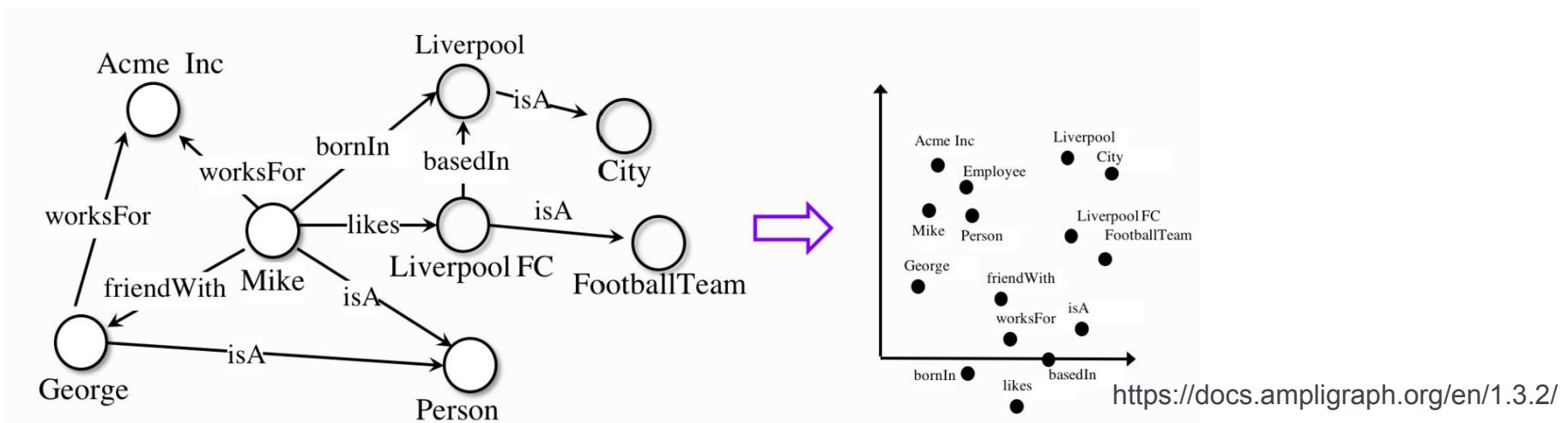
Background: Knowledge Graphs Applications



Zou, Xiaohan. "A survey on application of knowledge graph." In *Journal of Physics: Conference Series*, vol. 1487, no. 1, p. 012016. IOP Publishing, 2020.

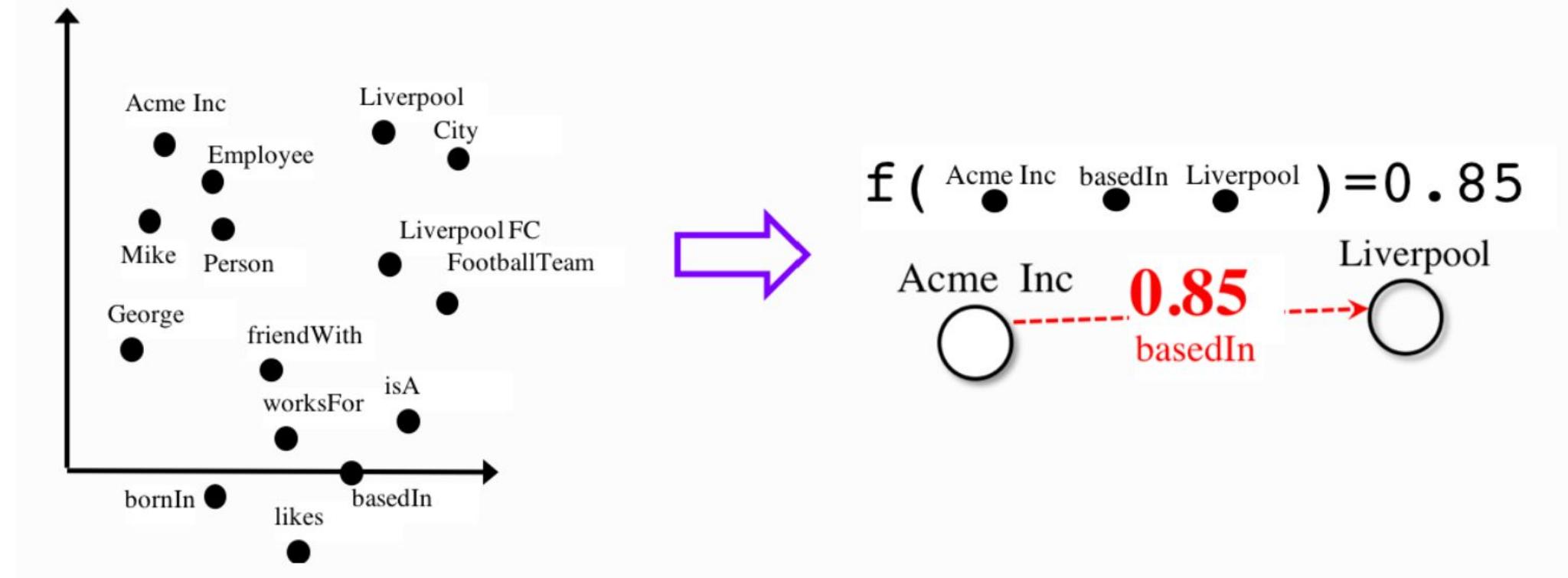
Background: Knowledge Graphs Embeddings

- Knowledge graphs are incomplete
- Solution for KG incompleteness: Link prediction
- Machine learning based link prediction:
 - KG mainly contain symbolic data, ML mainly acts on numerical data
 - Solution: Map KG to a vector space (embeddings)
 - define function to measure likelihood of links



Background: Knowledge Graphs Embeddings

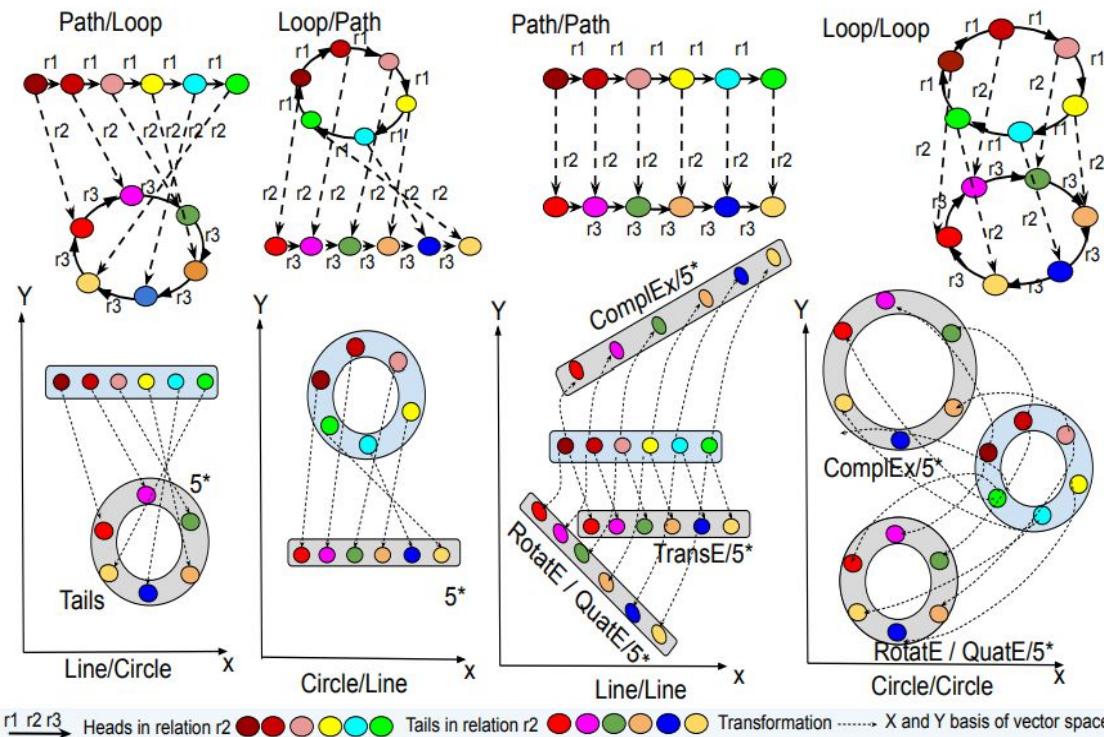
- KGEs measure likelihood of links in the vector space



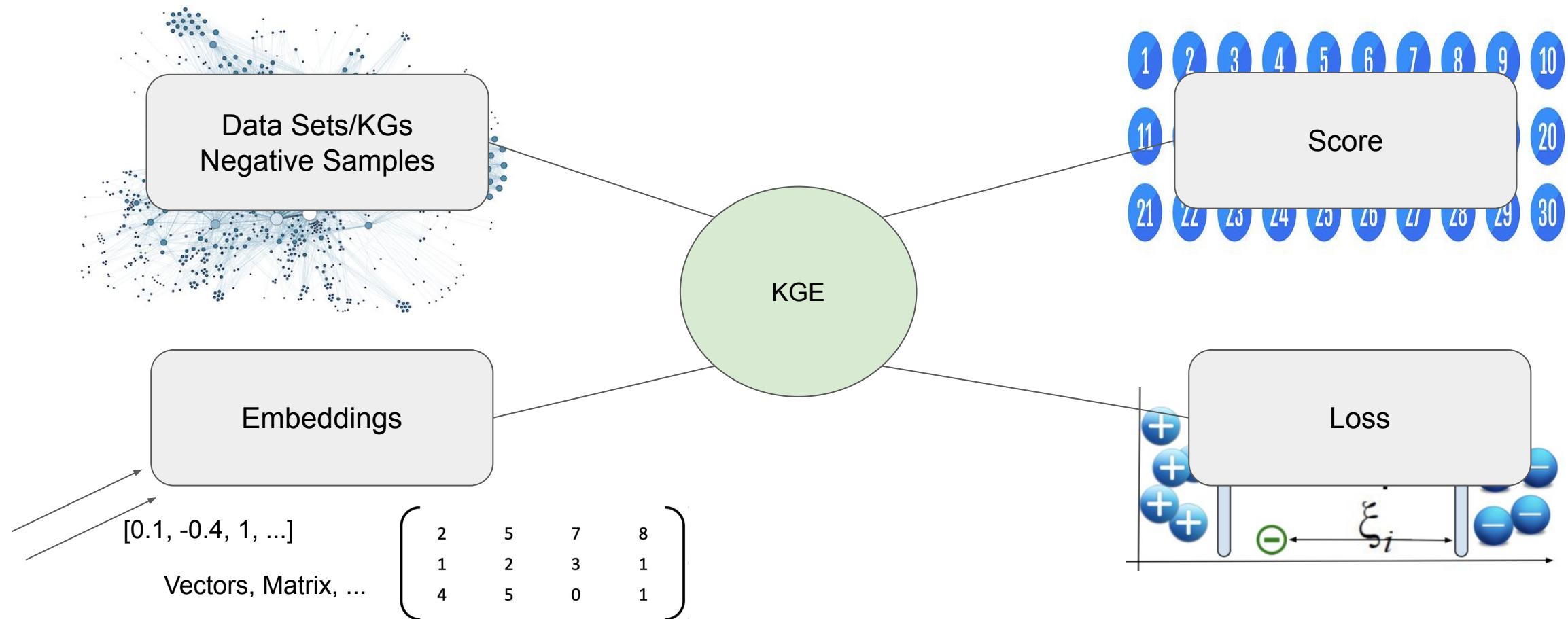
<https://docs.ampligraph.org/en/1.3.2/>

Background: Knowledge Graphs Embeddings

- Knowledge graph embedding:
 - Learn the representation of knowledge graphs in the vector space
 - Assumption: Embedding space can represent the graph characteristics
 - E.g., preserve graph structures Such as loop, path



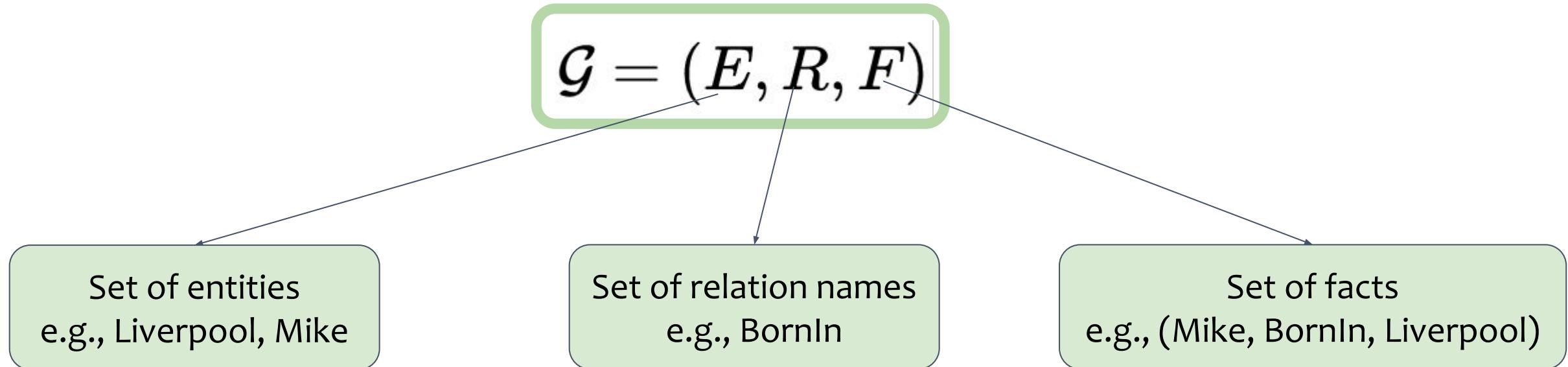
Background: Knowledge Graph Embeddings



<https://docs.ampligraph.org/en/1.3.2/>

Background: Knowledge Graph Embeddings

- Training data: Knowledge Graph (positive samples)



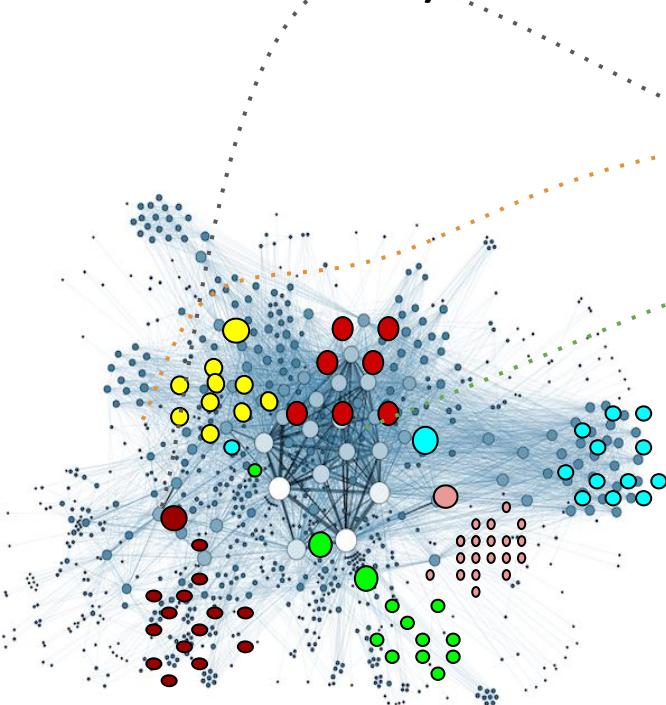
Background: Knowledge Graph Embeddings

- **Training data:**
 - Knowledge graphs include positive samples
 - ML models require both positive and negative samples
 - **Solution:** generate random negative samples
 - **Negative sampling:**
 - Given a positive triple $t=(s, p, o)=(Berlin, CapitalOf, Germany)$
 - Take a random entity, e.g., $e'=France$, from E
 - Generate negative sample from the positive one
 - $t'=(s, p, e')=(Berlin, CapitalOf, France)$
 - For each positive sample, generate k negative samples

Background: Score Function

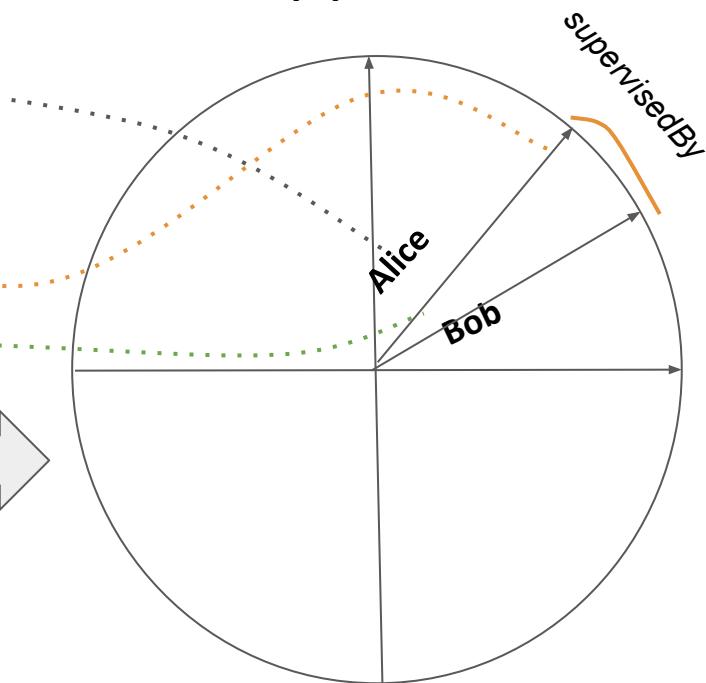
Vectors representation of entities and relations: (s, p, o)

subject \circlearrowright relation = object (\circlearrowright is an operator which applies rotation)



$$f(s, p, o) = -\| s \circlearrowright p - o \|$$

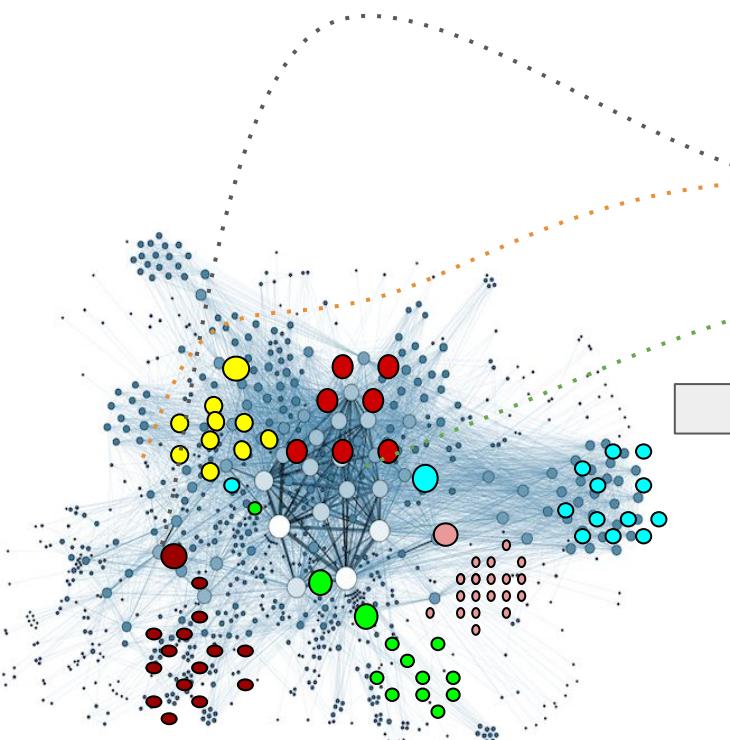
query: (John, supervisedBy,?)



Background: TransE Score Function

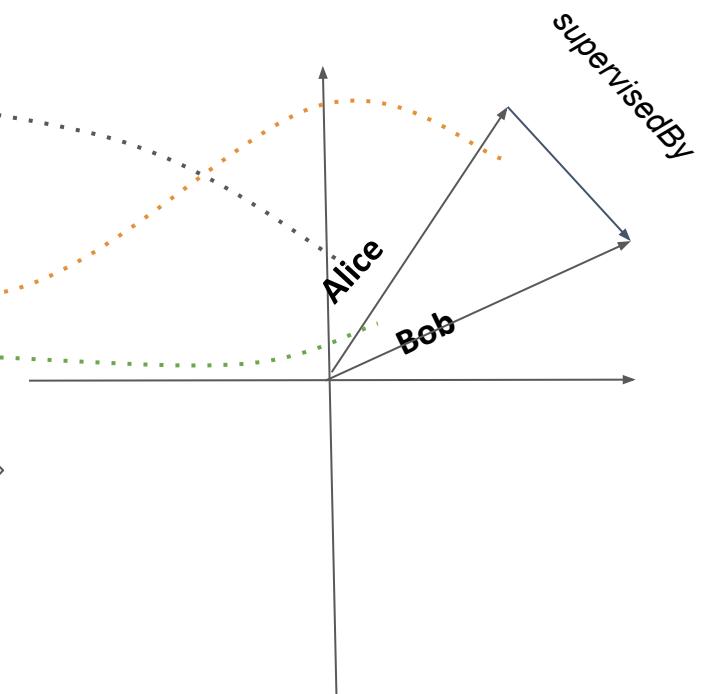
Vectors representation of entities and relations: (s, p, o)

$$\text{subject} \xrightarrow{\quad} + \xrightarrow{\quad} \text{relation} = \text{object}$$



$$f(s, p, o) = -\| s + p - o \|$$

query: (John, supervisedBy,?)



Knowledge Graphs

- Ranking metrics:
 - Rank: for a given test triple (s, p, o) ,
 - **Right rank:** “ o ” is replaced with all entities in the KG
 - score of all triples $\{(s, p, o')\}$ are computed
 - all triples are sorted and ranked based on their scores
 - Rank of the original triple (s, p, o') is returned as right rank
 - **Left rank:** “ s ” is replaced with all entities in the KG
 - same process is done
 - **Rank** of a triple (s, p, o) is the average of left and right ranks
 - **Mean rank (MR):** average rank over all test triples
 - **Mean Reciprocal Rank (MRR):** average of reciprocal of ranks
 - **Hits@k:** the percentages that test triples that their ranks are lower than k

Knowledge Graphs

- Loss functions:
 - Margin ranking loss:

$$\mathcal{L}_m = \sum_{(\mathbf{s}, \mathbf{p}, \mathbf{o}) \in \mathcal{F}} \left[\lambda + f(\mathbf{s}, \mathbf{p}, \mathbf{o}) - \sum_{\mathbf{o}' \in E} f(\mathbf{s}, \mathbf{p}, \mathbf{o}') \right]_+$$

- Cross-entropy loss:

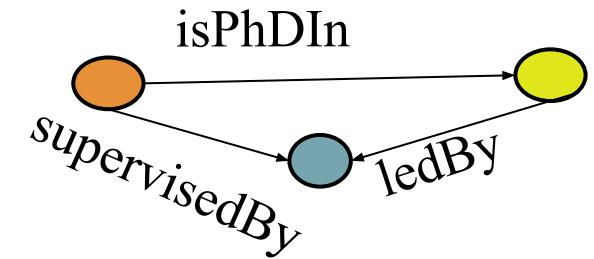
$$\begin{aligned} \mathcal{L}_c = & - \log \left(\frac{\exp(f(\mathbf{s}, \mathbf{p}, \mathbf{o}))}{\sum_{\mathbf{s}' \in E} \exp(f(\mathbf{s}', \mathbf{p}, \mathbf{o}))} \right) \\ & - \log \left(\frac{\exp(f(\mathbf{o}, \mathbf{p}^{-1}, \mathbf{s}))}{\sum_{\mathbf{o}' \in E} \exp(f(\mathbf{o}', \mathbf{p}^{-1}, \mathbf{s}))} \right) \end{aligned}$$

Relational Patterns

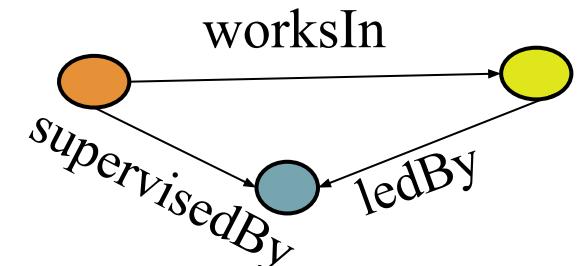
Conclusion \leftarrow Premise

Composition

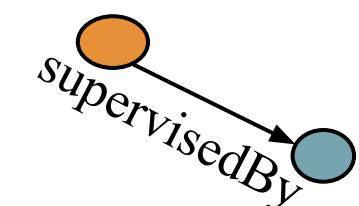
supervisedBy(X,Y) \leftarrow isPhdIn(X,Z), ledBy (Z,Y).



supervisedBy(X,Y) \leftarrow worksIn(X,Z), ledBy (Z,Y).



$\neg \text{supervisedBy}(Y,X) \leftarrow \text{supervisedBy}(X,Y).$

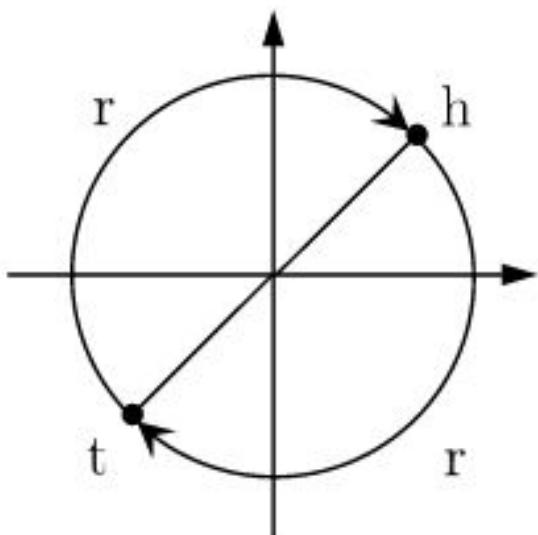


Antisymmetry

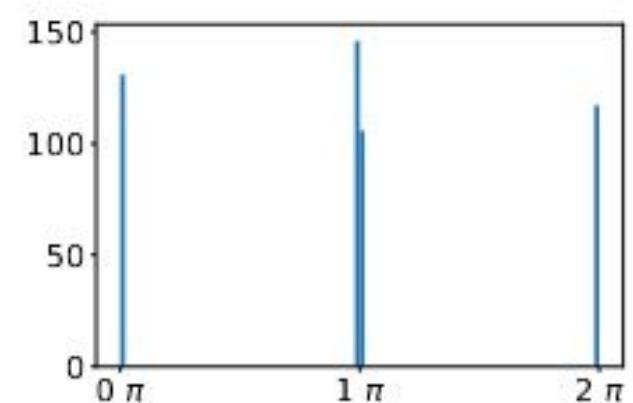
Inferring Relational Patterns

Conclusion ← Premise

- Pattern inference: if a model represent premise as true, then it automatically represent conclusion as true



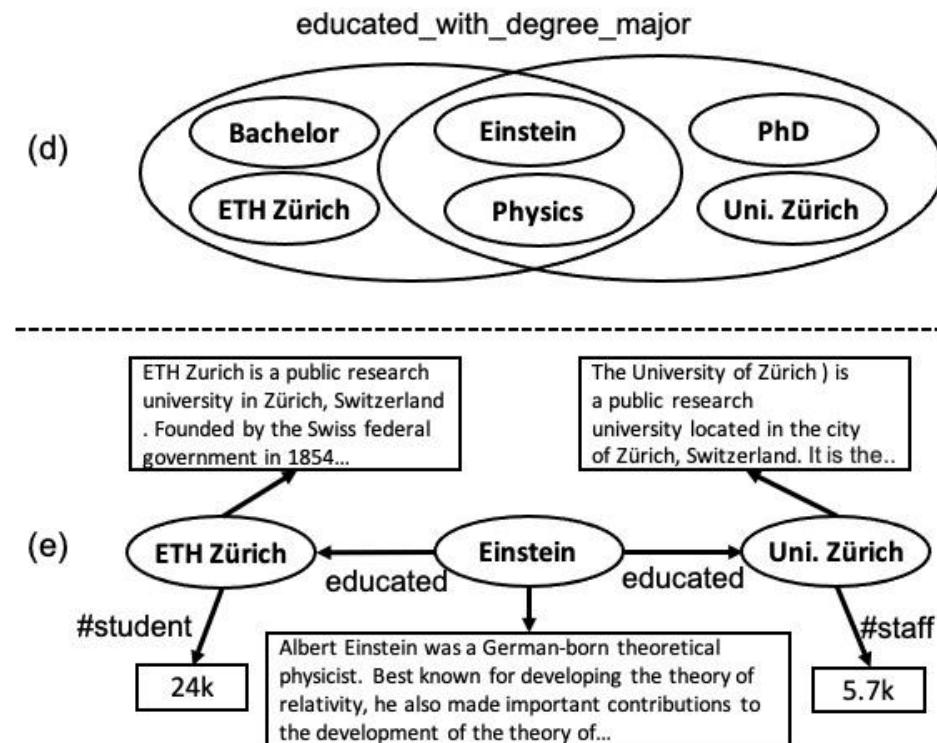
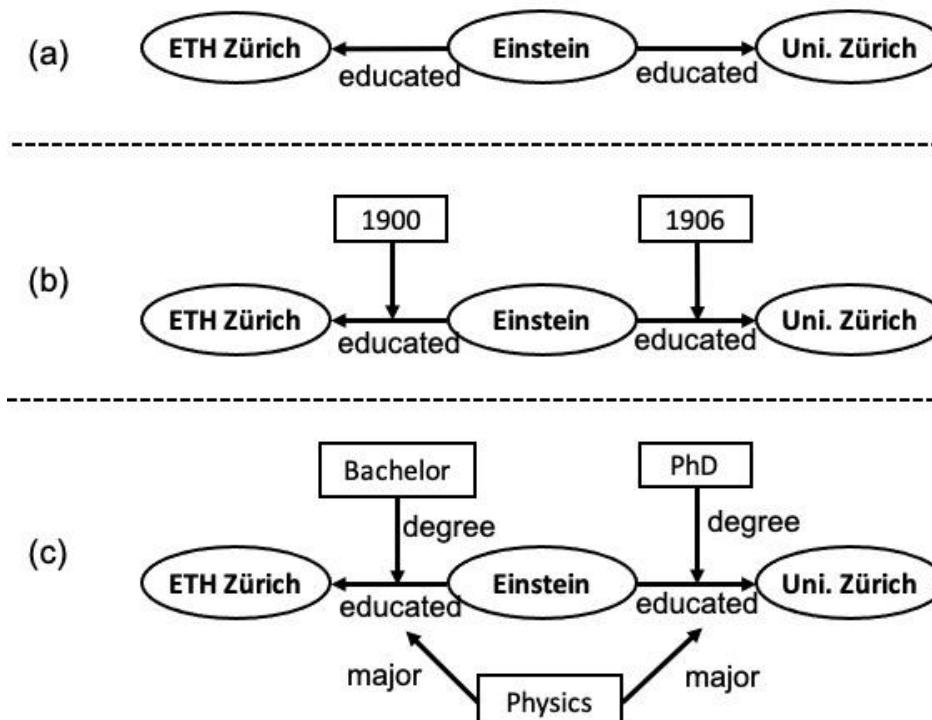
Inferring Symmetry
by RotatE



(a) `similar_to`

Motivation

- Knowledge graphs depict explicit knowledge over the world
- But are limited to the triple representations (subject, predicate, object)
- This tutorial goes beyond these vanilla representation





Part 2. Temporal KG Embeddings

Agenda

Part 1. Introduction (Bo & Mojtaba, 9:00 - 9:30)

- 1.1 Tutorial Overview (Bo)
- 1.2 Introduction to KG Embeddings
- 1.3 Motivation to go beyond triple-based KGs

Part 2. Temporal KG Embeddionsg (Mojtaba, 9:30 – 10:30)

- 2.1 Background
 - 2.2 In sample timestamp KGE
 - 2.3 Out of sample timestamp KGE
- Coffee Break (10:30 – 11:00)*

Part 3. Hyper-relational & N-ary KG embeddings (Bo, 11:00 – 12:30)

- 3.1 Introduction to hyper-relational & N-Ary KGs
 - 3.2 N-ary relational embeddings
 - 3.3 Hyper-relational embeddings
- Lunch Break (12:30 – 14:00)*

Part 4. KG embeddings with literal and text (Daniel, 14:00 – 15:00)

- 4.1 Incorporating entity descriptions
- 4.2 Incorporate numeric literals as features
- 4.3 Incorporating relation descriptions

Part 5. Advanced Topic (Michael, 15:00-15:30 & 16:00 – 16:30)

- 5.1 Complex query embeddings (Daniel)
- Coffee Break (15:30 – 16:00)*
- 5.2 Neural Graph Databases
- 5.3 Inductive learning settings
- 5.4 Incorporating ontology, LLM, etc.

Part 6. Conclusion & Future Works (Michael, 16:30 – 17:00)

- 6.1 Conclusion
- 6.2 Future Works

Background: Temporal Knowledge Graphs

- Temporal KG:

- E is the set of entities
- R is the set of relation names
- T is a set of time stamps
- X is a set of quadruple

$$\mathcal{G} = (E, R, T, X)$$

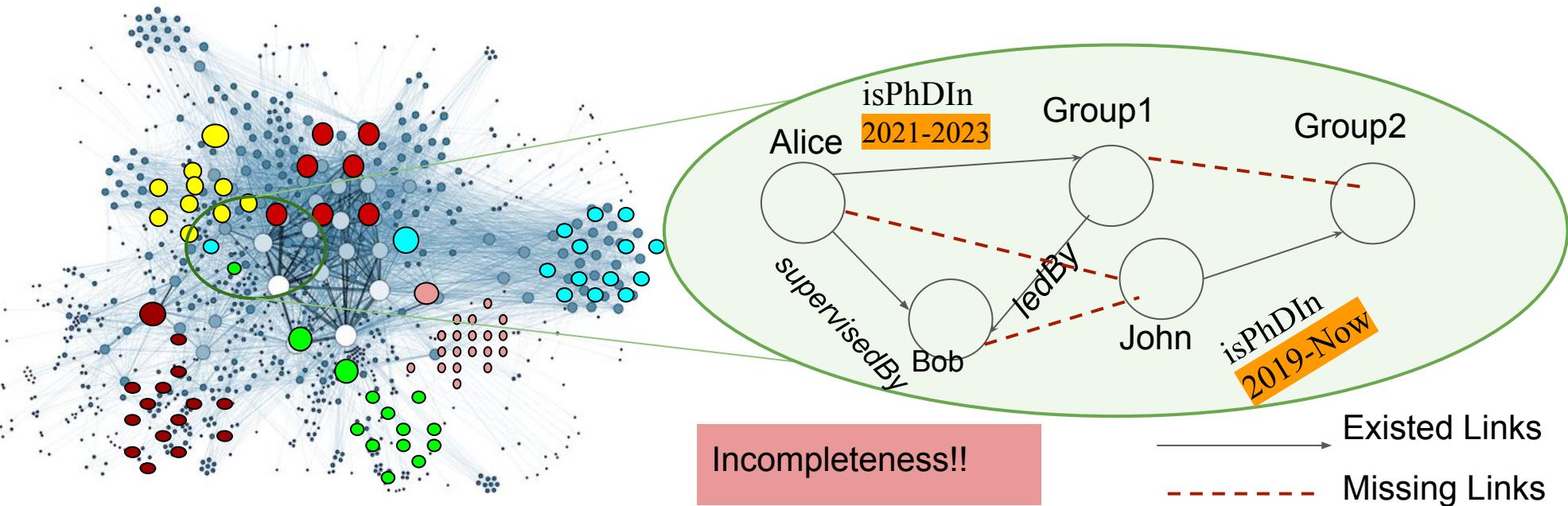
$$X \in E * R * E * T$$

- Time stamp:

- time point, e.g., (Alice, *visited*, Bob, **21 April 2021**)
- time interval, e.g., (Alice, *supervisedBy*, Bob, **2021-2023**)

Background: Temporal Knowledge Graphs

- Relational and structured data
- Representation of temporal facts: (Alice, *supervisedBy*, Bob, **2021-2023**)



Background: Temporal Knowledge Graphs

- **Prediction in TKGs:**
 - Subject prediction ($?, p, o, t$)
 - Relation prediction ($s, ?, o, t$)
 - Object prediction ($s, p, ?, t$)
 - Time prediction ($s, p, o, ?$)
- **Evaluation metrics:**
 - Mean Rank (MR)
 - Mean Reciprocal Rank (MRR)
 - Hits@k (e.g., $k=1, 3, 10$)

Background: Temporal Knowledge Graphs

- **Filtering:**
 - time-aware filtering:
 - Filter quadruples appeared in train/valid/test
 - time unaware filtering:
 - Ignore time during filtering (filtering based on triples)
- Time-aware filtering is more suitable for evaluation
- **Challenge of TKGE:** how to effectively integrate the temporal validity of facts into the models to properly capture temporal dynamics of entities, relations, and the underlying graph for the prediction

Background: Temporal Knowledge Graphs

- TKG Dataset:

Datasets	#Entities	#Relations	#Timestamps	#Time Span	#Training	#Validation	#Test	#Granularity	#Category
ICEWS14	6,869	230	365	01/01/2014 – 12/31/2014	72,826	8,941	8,963	24 hours	Interpolation
ICEWS05-15	10,094	251	4,017	01/01/2005 – 12/31/2015	368,962	46,275	46,092	24 hours	Interpolation
GDELT	500	20	366	04/01/2015 – 03/31/2016	2,735,685	341,961	341,961	24 hours	Interpolation
YAGO11k	10,623	10	70	-431 – 2844	16,406	2,050	2,051	–	Interpolation
YAGO15k	15,403	34	198	1553 – 2017	29,381	3,635	3,685	–	Interpolation
Wikidata12k	12,554	24	81	1709 – 2018	32,497	4,062	4,062	–	Interpolation
ICEWS14	6,869	230	365	01/01/2014 – 12/31/2014	72,826	8,941	8,963	24 hours	Extrapolation
ICEWS18	23,033	256	304	01/01/2018 – 10/31/2018	373,018	45,995	49,545	24 hours	Extrapolation
GDELT	7,691	240	2,751	01/01/2018 – 01/31/2018	1,734,399	238,765	305,241	15 mins	Extrapolation
WIKI	12,554	24	232	1786 – 2018	539,286	67,538	63,110	1 year	Extrapolation
YAGO	10,623	10	189	1830 – 2019	161,540	19,523	20,026	1 year	Extrapolation

Background: Temporal Knowledge Graphs

- **TKG Dataset:**

- The ICEWS and GDELT datasets contain events with time points, which are respectively extracted from the Integrated Crisis Early Warning System repository and the Global Database of Events, Language, and Tone.
- WIKIDATA contains events extracted from the Wikidata knowledge base, with timestamps as time ranges like “occursSince 2013”.
- YAGO15K also contain time range.

Overview of Temporal Knowledge Graph Embeddings

- In sample timestamp KGE
 - Time-included Tensor Decomposition
 - Time-based Transformation
 - Synthetic Time-dependent Relation TKGE
 - Timestamps-specific functions-based TKGE
 - Dynamic Embedding
- Out of sample timestamp KGE
 - Interpolation-based TKGE
 - Extrapolation-based TKGE

Agenda

Part 1. Introduction (Bo & Mojtaba, 9:00 - 9:30)

- 1.1 Tutorial Overview (Bo)
- 1.2 Introduction to KG Embeddings
- 1.3 Motivation to go beyond triple-based KGs

Part 2. Temporal KG Embeddionsg (Mojtaba, 9:30 – 10:30)

- 2.1 Background
 - 2.2 In sample timestamp KGE
 - 2.3 Out of sample timestamp KGE
- Coffee Break (10:30 – 11:00)*

Part 3. Hyper-relational & N-ary KG embeddings (Bo, 11:00 – 12:30)

- 3.1 Introduction to hyper-relational & N-Ary KGs
 - 3.2 N-ary relational embeddings
 - 3.3 Hyper-relational embeddings
- Lunch Break (12:30 – 14:00)*

Part 4. KG embeddings with literal and text (Daniel, 14:00 – 15:00)

- 4.1 Incorporating entity descriptions
- 4.2 Incorporate numeric literals as features
- 4.3 Incorporating relation descriptions

Part 5. Advanced Topic (Michael, 15:00-15:30 & 16:00 – 16:30)

- 5.1 Complex query embeddings (Daniel)
- Coffee Break (15:30 – 16:00)*
- 5.2 Neural Graph Databases
- 5.3 Inductive learning settings
- 5.4 Incorporating ontology, LLM, etc.

Part 6. Conclusion & Future Works (Michael, 16:30 – 17:00)

- 6.1 Conclusion
- 6.2 Future Works

In sample timestamp Temporal KGE

- The quadruples in the test set include timestamps that have been observed during training
 - e.g., Facts in the training set cover all years between 2000 and 2022
 - All new facts in test set contain timestamp between 2000 and 2022
- Most TKGE models are in sample timestamp, classified into the following:
 - Time-included Tensor Decomposition
 - Time-based Transformation
 - Synthetic Time-dependent Relation TKGE
 - Timestamps-specific functions-based TKGE
 - Dynamic Embedding

Time-included Tensor Decomposition

- They represent Temporal KG as 4-way tensor
- They decompose the tensor via a tensor factorization

Complex space

T-ComplEx
(ICLR'20)

Hypercomplex space

TeLM (NAACL'21)

Tucker tensor
decomposition

TuckERT (KBS'22)

Cai, Borui, et al. "Temporal knowledge graph completion: A survey." *arXiv preprint arXiv:2201.08236* (2022).

Time-included Tensor Decomposition

- Represent Temporal KG as 4-way tensor
- Decompose the tensor via a tensor factorization

$$\mathcal{G} = (E, R, T, X)$$

$$X \in \mathbb{R}^{|E| \times |R| \times |E| \times |T|}$$

$$X \approx \hat{X} = \sum_{\alpha=1}^d \mathbf{E}_{:, \alpha} \otimes \mathbf{R}_{:, \alpha} \otimes \mathbf{E}_{:, \alpha} \otimes \mathbf{T}_{:, \alpha}$$



Lin, Lifan, and Kun She. "Tensor decomposition-based temporal knowledge graph embedding." *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2020.

Time-included Tensor Decomposition

$$\hat{X} = \sum_{\alpha=1}^d \mathbf{E}_{:, \alpha} \otimes \mathbf{R}_{:, \alpha} \otimes \mathbf{E}_{:, \alpha} \otimes \mathbf{T}_{:, \alpha} = [\![\mathbf{E}, \mathbf{R}, \mathbf{E}, \mathbf{T}]\!] \iff \forall (i, j, k, l), X_{i,j,k,l} = \sum_{\alpha=1}^d \mathbf{e}_{i,\alpha} \mathbf{r}_{i,\alpha} \mathbf{e}_{i,\alpha} \mathbf{t}_{i,\alpha} = \langle \mathbf{e}_i, \mathbf{r}_j, \mathbf{e}_k, \mathbf{t}_l \rangle$$

Matrix form

Vector form

T-ComplEx (ICLR'20)

- Tensor decomposition with complex embeddings

Score function

$$X \approx \hat{X}(E, R, T) = \text{Re}([\![\mathbf{E}, \mathbf{R}, \bar{\mathbf{E}}, \mathbf{T}]\!]) \iff \hat{X}(E, R, T)_{i,j,k,l} = \text{Re}(\langle e_i, r_j, \bar{e}_k, t_l \rangle)$$

Complex Matrices

Complex vectors

T-ComplEx (ICLR'20)

- Advantage of T-ComplEx:
 - T-ComplEx can handle symmetry and antisymmetry
 - Temporal embeddings scale with the number of timestamps
 - Expressive model
- Note: the timestamp can be used to equivalently modulate the objects, predicates or subjects to obtain time-dependent representation

$$\langle \mathbf{e}_i, \mathbf{r}_j, \overline{\mathbf{e}_k}, \mathbf{t}_l \rangle = \langle \mathbf{e}_i \odot \mathbf{t}_l, \mathbf{r}_j, \overline{\mathbf{e}_k} \rangle = \langle \mathbf{e}_i, \mathbf{r}_j \odot \mathbf{t}_l, \overline{\mathbf{e}_k} \rangle = \langle \mathbf{e}_i, \mathbf{r}_j, \overline{\mathbf{e}_k \odot \mathbf{t}_l} \rangle$$

TNT-ComplEx (ICLR'20)

- Disadvantage of T-ComplEx:
 - Some relationships might not be affected by time
 - e.g., “daughters of” does not change by time, but “hasOccupation” may change by time
- Solution: add static relation

$$\hat{X} = \text{Re}([\mathbf{E}, \mathbf{R}^t, \bar{\mathbf{E}}, \mathbf{T}] + [\mathbf{E}, \mathbf{R}, \bar{\mathbf{E}}, \mathbf{1}]) \iff \hat{X}_{i,j,k,l} = \text{Re}(\langle \mathbf{e}_i, \mathbf{r}_j^t \odot \mathbf{t}_l + \mathbf{r}_j, \bar{\mathbf{e}}_k \rangle)$$

Temporal
relation

Static
relation

Temporal
relation

Static
relation

- Beyond complex space
 - TeLM learns multivector representations of entities, relations, and time
 - e.g., 2-grade multivector representations
$$M_b = b_0 + b_1e_1 + b_2e_2 + b_{12}e_1e_2, e_1e_1 = e_2e_2 = 1, e_1e_2 = -e_2e_1$$
 - Complex product is replaced by n-grade geometric product
 - Advantage: T-ComplEx is a special case of TeLM
- TeLM handles both time point and time intervals:
 - for a fact with interval timestamp $(s, p, o, [t_b, t_e])$
 - split it into two parts $(s, p, o, [t_b, -]), (s, p, o, [-, t_e])$
 - Score
$$f(s, r, o, [t_b, t_e]) = \frac{1}{2}(f(s, r_b, o, t_b) + f(s, r_e, o, t_e))$$

TuckERT (KBS'22)

- TuckERT is a TKGE based on Tucker decomposition
 - It is expressive
 - It captures interactions between different elements of TKG

$$\hat{X}(E, R, T)_{i,j,k,l} = \langle \mathcal{M}; \mathbf{e}_i, \mathbf{r}_j, \mathbf{e}_k, \mathbf{t}_l \rangle = \sum_{\alpha_1} \sum_{\alpha_2} \sum_{\alpha_3} \sum_{\alpha_4} \mathcal{M}^{(\alpha_1 \alpha_2 \alpha_3 \alpha_4)} \mathbf{e}_{\alpha_1 i} \circ \mathbf{r}_{\alpha_2 j} \circ \mathbf{e}_{\alpha_3 k} \circ \mathbf{t}_{\alpha_4 l}$$

Core tensor

- Disadvantage: Inefficient in terms of number of parameters

Temporal Regulizer

- Temporal smoothness penalties are used to ensure neighboring timestamps learn similar representations

$$\mathcal{L}_{\tau_1} = \sum_{i=1}^{|T|-1} \|\mathbf{t}_{i+1} - \mathbf{t}_i\|_p^p$$

Temporal smoothness
penalty

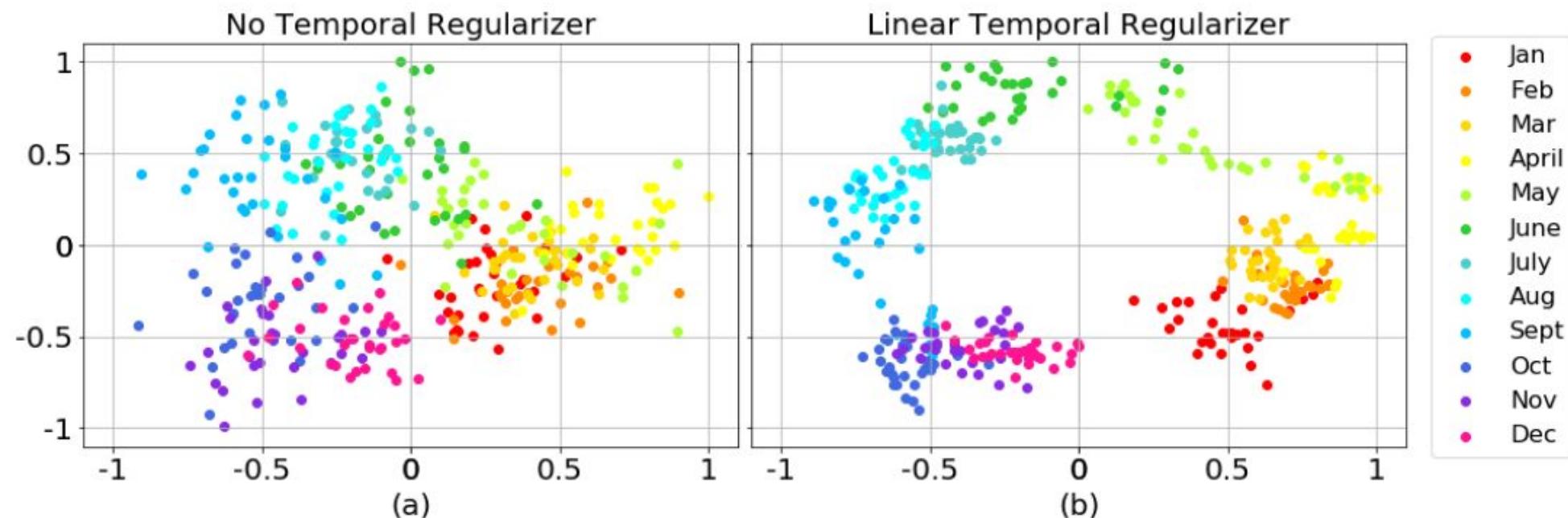
$$\mathcal{L}_{\tau_2} = \frac{1}{|T|-1} \sum_{i=1}^{|T|-1} \|\mathbf{t}_{i+1} - \mathbf{t}_i + \mathbf{b}_t\|_3^3$$

Temporal smoothness
with bias

$$\|\mathbf{t}_{i+m} - \mathbf{t}_i\| > \|\mathbf{t}_{i+1} - \mathbf{t}_i\| \text{ when } m \gg 1$$

Temporal Regulizer

- 2D PCA of the learned temporal embedding with and without temporal regularizer



Xu, Chengjin, et al. "Temporal knowledge graph completion using a linear temporal regularizer and multivector embeddings." NAACL 2021

Summary

- TKGE based on Tucker decomposition
 - They are expressive
 - They can capture interactions between different elements of TKG
- Disadvantages:
 - They may not infer all patterns well.
 - e.g., composition

In sample timestamp Temporal KGE

- The quadruples in the test set include timestamps that have been observed during training
 - e.g., Facts in the training set cover all years between 2000 and 2022
 - All new facts in test set contain timestamp between 2000 and 2022
- Most TKGE models are in sample timestamp, classified into the following:
 - Time-included Tensor Decomposition
 - Time-based Transformation
 - Synthetic Time-dependent Relation TKGE
 - Timestamps-specific functions-based TKGE
 - Dynamic Embedding

Time-based Transformation

- Synthetic Time-dependent Relation TKGes
 - They model temporal relations
 - (s, p, o, t) is converted to $(s, p:t, o)$
- Timestamp-specific function-based TKGes
 - They represent time as a simple parameterized function
 - They parametrize time as a sophisticated function, e.g., deep NN

Time-based Transformation

- Synthetic Time-dependent Relation TKGEs
 - They model temporal relations
 - (s, p, o, t) is converted to $(s, g(p, t), o)$
- Timestamp-specific function-based TKGEs
 - (s, p, o, t) is converted to $(g_t(s), g_t(p), g_t(o))$
 - g is a timestamp-specific function

Synthetic Time-dependent Relation TKGes

(Lakers; championOf; NBA; 2010)



(Lakers; championOf:2010; NBA)

TTransE (WWW'18)

(Lakers; championOf; NBA; 2010)



(Lakers; championOf:2010; NBA)

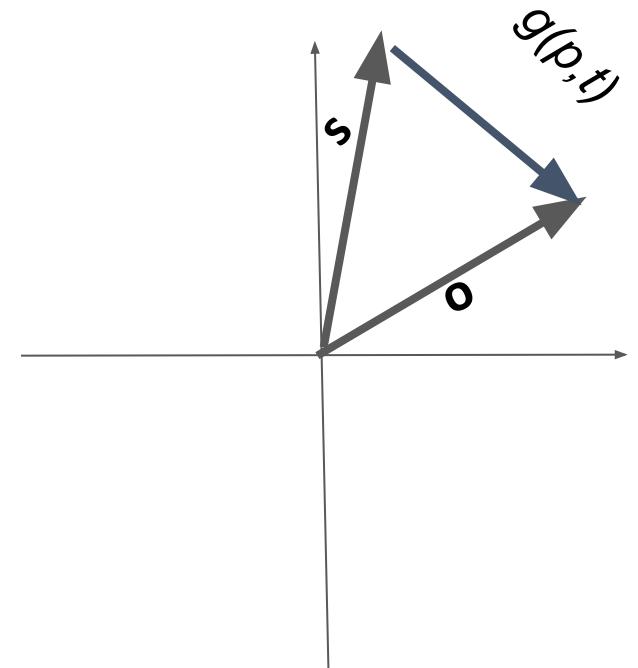
$$f(s, p, o) = -\|s + g(p, t) - o\|$$

Synthetic Predicates

$$g(p, t) = \mathbf{p} : \mathbf{t}$$

$$g(p, t) = \mathbf{p} + \mathbf{t}$$

$$g(p, t) = \mathbf{p} * \mathbf{t}$$



Leblay, Julien, and Melisachew Wudage Chekol. "Deriving validity time in knowledge graph." *Companion Proceedings of the The Web Conference 2018*.

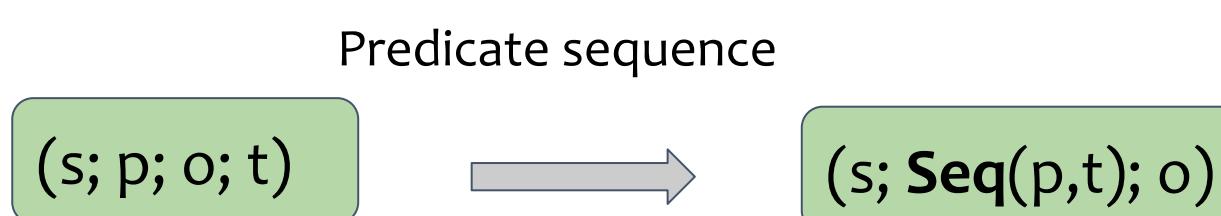
- Disadvantage:
 - Temporal relations might appear diversity in a dataset

Examples:

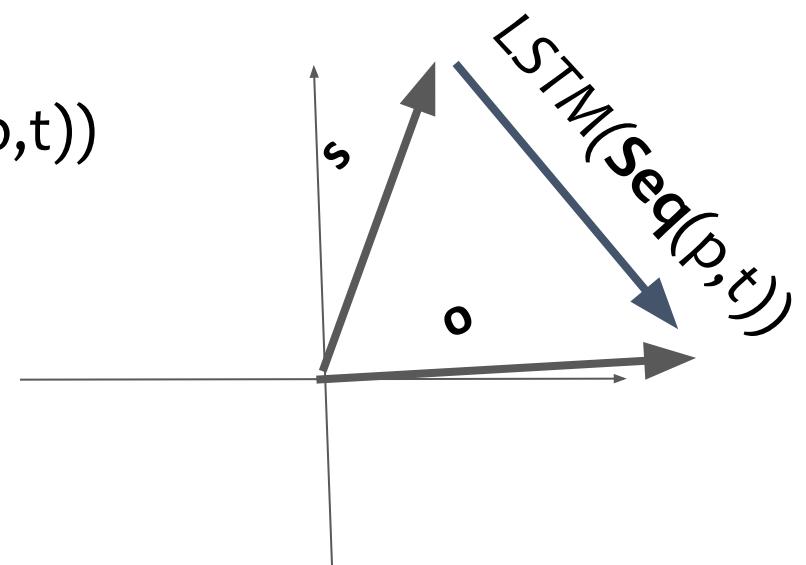
Fact
(Barack Obama, country, US)
(Barack Obama, born, US, 1961)
(Barack Obama, president, US, since, 2009-01)

- TTransE cannot handle these diverse temporal relations simultaneously

TA-TransE (EMNLP'18)



$\text{LSTM}(\text{Seq}(p,t))$



Examples of mapping predicate to predicate sequences

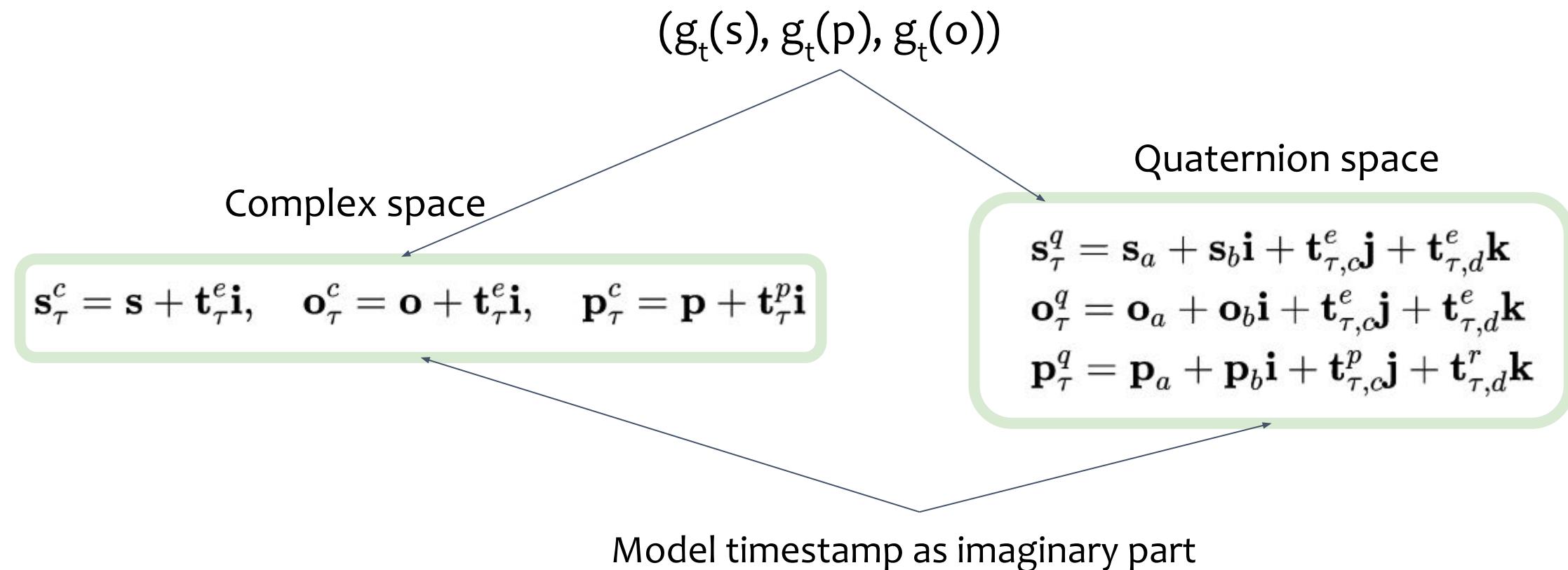
Fact	Predicate Sequence
(Barack Obama, country, US)	[country]
(Barack Obama, born, US, 1961)	[born, 1y, 9y, 6y, 1y]
(Barack Obama, president, US, since, 2009-01)	[president, since, 2y, 0y, 0y, 9y, 01m]

Garcia-Duran, Alberto, Sebastijan Dumančić, and Mathias Niepert. "Learning Sequence Encoders for Temporal Knowledge Graph Completion." *EMNLP 2018*

Summary of Synthetic Time-based Relation Models

- Synthetic time-based relation KGEs represent temporal relations.
- They can take advantage of existing static KGE models
- Disadvantage:
 - they do not model temporal entity evolutions

- Synthetic time-based entity and relation models



In sample timestamp Temporal KGE

- The quadruples in the test set include timestamps that have been observed during training
 - e.g., Facts in the training set cover all years between 2000 and 2022
 - All new facts in test set contain timestamp between 2000 and 2022
- Most TKGE models are in sample timestamp, classified into the following:
 - Time-included Tensor Decomposition
 - Time-based Transformation
 - Synthetic Time-dependent Relation TKGE
 - Timestamps-specific functions-based TKGE
 - Dynamic Embedding

Timestamp-specific function-based TKGEs

(Lakers; championOf; NBA; 2010)



(Lakers:2010; championOf:2010; NBA:2010)

$$\mathbf{s}_t = \mathcal{P}_t(s), \mathbf{p}_t = \mathcal{P}_t(p), \mathbf{o}_t = \mathcal{P}_t(o)$$

(Lakers; championOf; NBA; 2010)



(Lakers:2010; championOf:2010; NBA:2010)

Projection on time-specific hyperplane

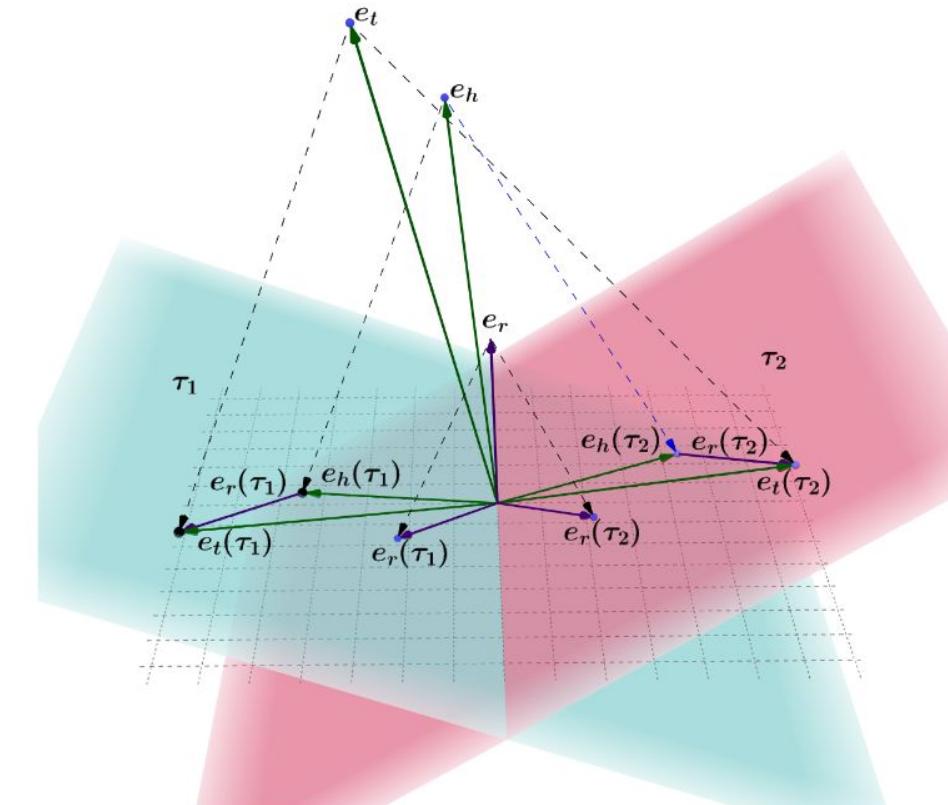
$$P_\tau(s) = \mathbf{s} - (w_\tau^\top \mathbf{s}) w_\tau$$

$$P_\tau(o) = \mathbf{o} - (w_\tau^\top \mathbf{o}) w_\tau$$

$$P_\tau(p) = \mathbf{p} - (w_\tau^\top \mathbf{p}) w_\tau$$

Score function

$$f_\tau(s, p, o) = \|P_\tau(s) + P_\tau(p) - P_\tau(o)\|_{l_1/l_2}$$



Dasgupta, Shib Sankar, Swayambhu Nath Ray, and Partha Talukdar. "Hyte: Hyperplane-based temporally aware knowledge graph embedding." EMNLP 2018.

- Disadvantage:
 - TKGs exhibit patterns
 - This model inherits limitations of translation-based KGE models in inferring patterns.

TeRo (Colling'20), RotateQVS (ACL'22)

- Temporal KGs exhibit temporal patterns:

Definition 1. A relation p is symmetric, if $\forall s, o, t, p(s, o, t) \wedge p(o, s, t)$ holds True.

Definition 2. A relation p is asymmetric, if $\forall s, o, t, p(s, o, t) \wedge \neg p(o, s, t)$ holds True.

Definition 3. Relation p_1 is the inverse of p_2 , if $\forall s, o, t, p_1(s, o, t) \wedge p_2(o, s, t)$ holds True.

Definition 4. Relation p_1 and p_2 are evolving over time from t_1 to t_2 , if $\forall s, o, p_1(s, o, t_1) \wedge p_2(s, o, t_2)$ holds True.

- Rotation operator is suitable for modeling these patterns

TeRo (Colling'20), RotateQVS (ACL'22)

(Lakers; championOf; NBA; 2010)



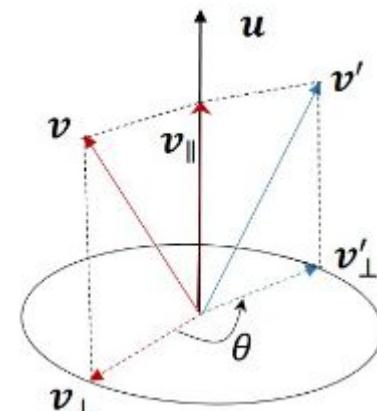
(Lakers:2010; championOf:2010; NBA:2010)

TeRO: time-specific 2D rotation in Complex space

$$\mathcal{P}_t(\mathbf{s}) = \mathbf{s} \circ \mathbf{t}, \quad \mathcal{P}_t(\mathbf{o}) = \mathbf{o} \circ \mathbf{t}$$

$$\mathbf{s}, \mathbf{p}, \mathbf{o}, \mathbf{t} \in \mathbb{C}^d, \quad \mathbf{t}_j \in \mathbb{C}, \text{ to be } |\mathbf{t}_j| = 1, \quad \text{that is } \mathbf{t}_j = e^{i\theta_{t,j}}$$

Rotation of \mathbf{v} around the axis \mathbf{u}



RotateQVS: time-specific 3D rotation in Quaternion space

$$\mathcal{P}_t(\mathbf{s}) = \mathbf{t} \mathbf{s} \mathbf{t}^{-1}, \quad \mathcal{P}_t(\mathbf{o}) = \mathbf{t} \mathbf{o} \mathbf{t}^{-1}$$

$$\mathbf{s}, \mathbf{p}, \mathbf{o}, \mathbf{t} \in \mathbb{Q}^d, \quad \mathbf{t}_j \in \mathbb{Q}, \text{ to be } |\mathbf{t}_j| = 1, \quad \text{that is } \mathbf{t}_j = e^{\frac{\theta_{t,j}}{2}(t_{j,x}i + t_{j,y}j + t_{j,z}k)}$$

RotateQVS (ACL'22)

- Advantages:

Lemma 1. RotateQVS can model the symmetric pattern for TKG.

Lemma 2. RotateQVS can model the asymmetric pattern for TKG.

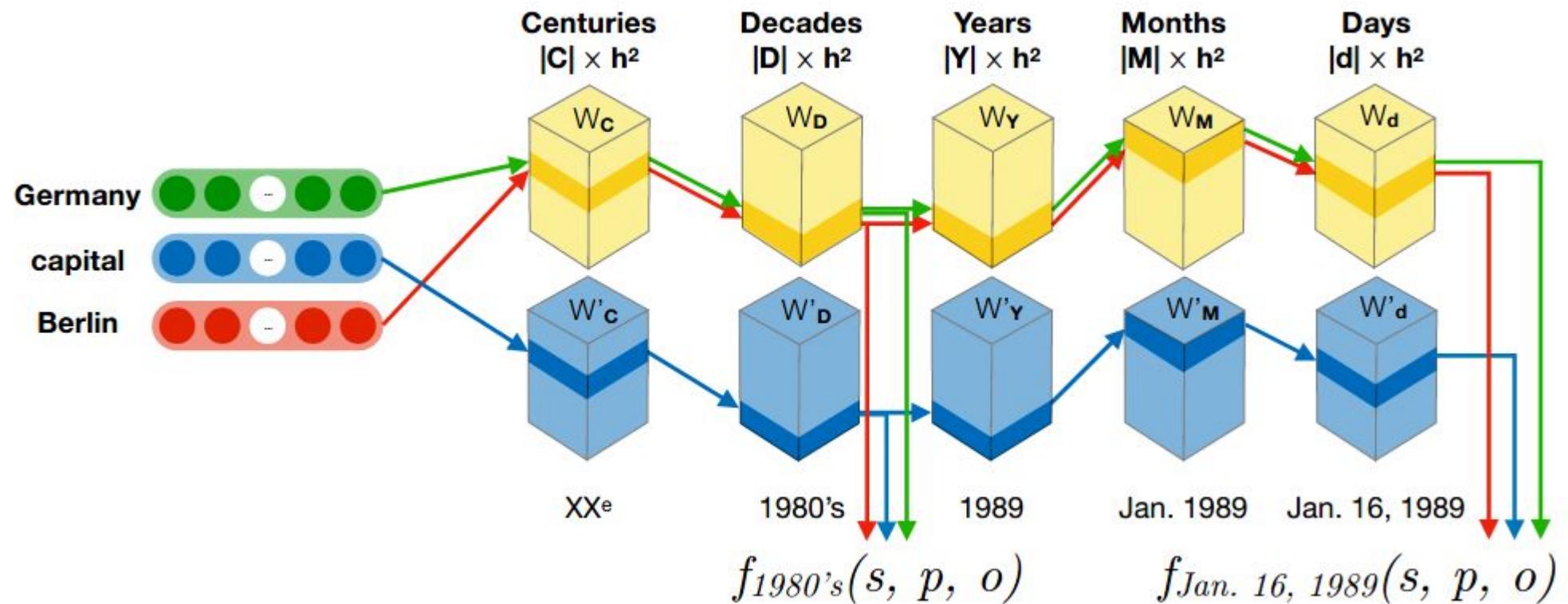
Lemma 3. RotateQVS can model the inversion pattern for TKG.

Lemma 4. RotateQVS can model the temporalevolution pattern for *TKG*.

- Disadvantage:

- It cannot handle arbitrary time resolution.

- This model handles arbitrary time resolution



- Define time scope

- Each year has 4 quarters, each quarter has 3 months, each month has 5 weeks, each week has 7 days

Quarters				Months			Weeks					Days					
1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0

- Example: the interval [2nd week Apr 2019; 24 May 2019]

- Month granularity

Quarters				Months			Weeks					Days					
0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

- Day granularity

Quarters				Months			Weeks					Days					
0	1	0	0	1	0	0	0	1	1	1	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	1	0	1	1	1	1	0	0

- Computes temporal embedding of entities

$$\mathcal{P}_t(\mathbf{e}) = W_{0 \rightarrow j}[t](e) = W_j[u_j](W_{j-1}[u_{j-1}](\dots W_0[u_0](e))), \quad t = (u_0, \dots, u_j)$$

Matrices for each granularity

u_i denotes slot set to 1 at granularity t_i

Summary of Time-based Transformation Models

- They models a time-based transformation function to represent temporal entities/relations.
- Disadvantage:
 - The context of an entity/relation might change over time.
 - They cannot properly model the dynamic evolution of entities over time, because
 - there is a global time transformation applied to all entities in the same way, while
 - the change over time might be different per entities.

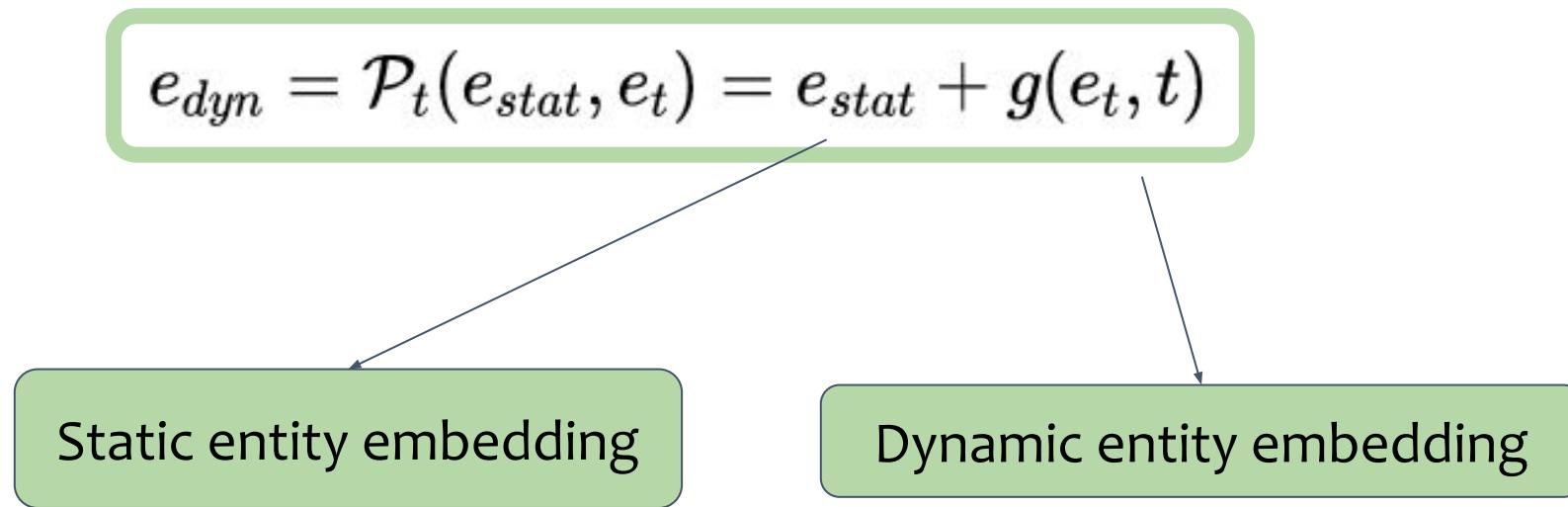
In sample timestamp Temporal KGE

- The quadruples in the test set include timestamps that have been observed during training
 - e.g., Facts in the training set cover all years between 2000 and 2022
 - All new facts in test set contain timestamp between 2000 and 2022
- Most TKGE models are in sample timestamp, classified into the following:
 - Time-included Tensor Decomposition
 - Time-based Transformation
 - Synthetic Time-dependent Relation TKGE
 - Timestamps-specific functions-based TKGE
 - Dynamic Embedding

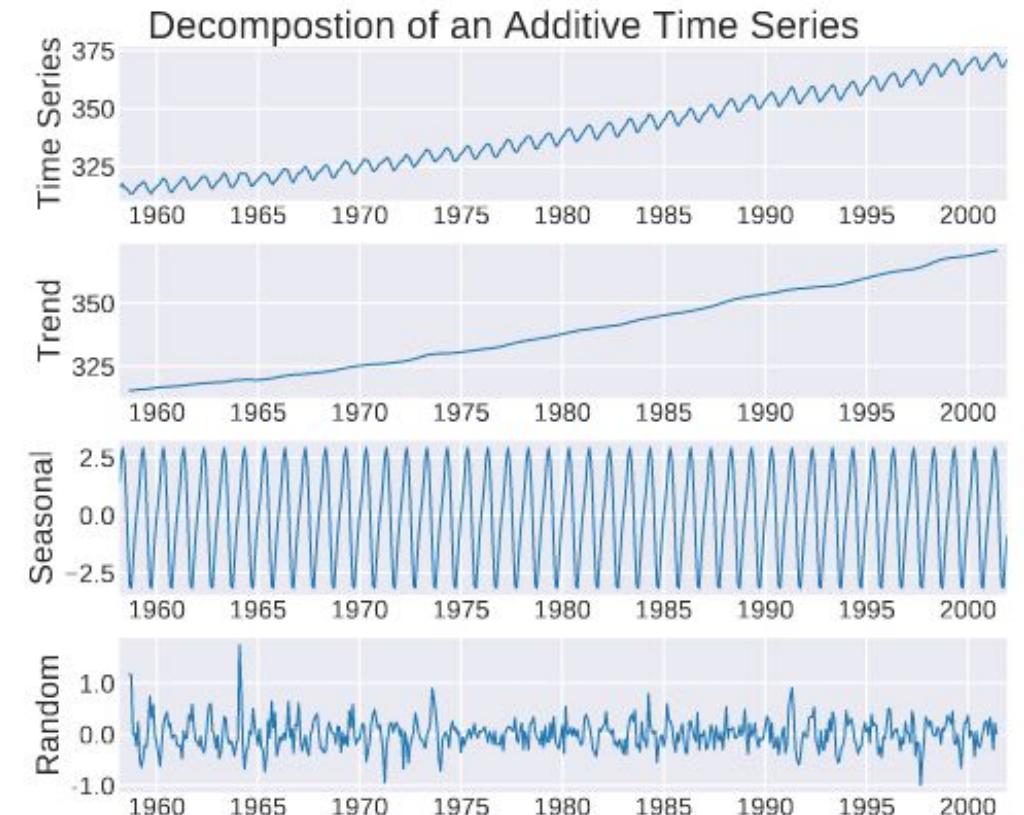
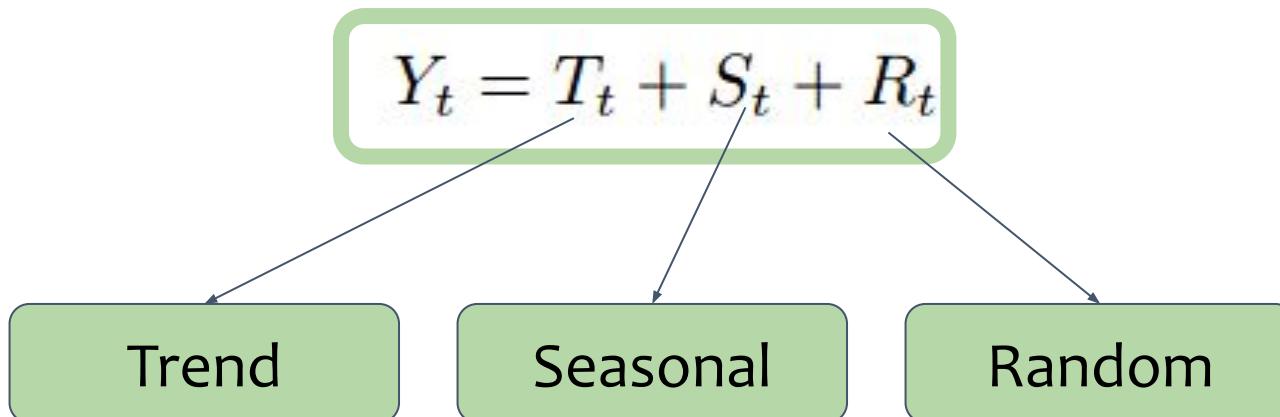
Dynamic Embedding

- Entities/relations' context change by time
- e.g., a person might have different experiences and lifestyles during childhood, adolescence, youth, adulthood
- Entity/relation evolution over time should be reflected in the embedding space in dynamic manner

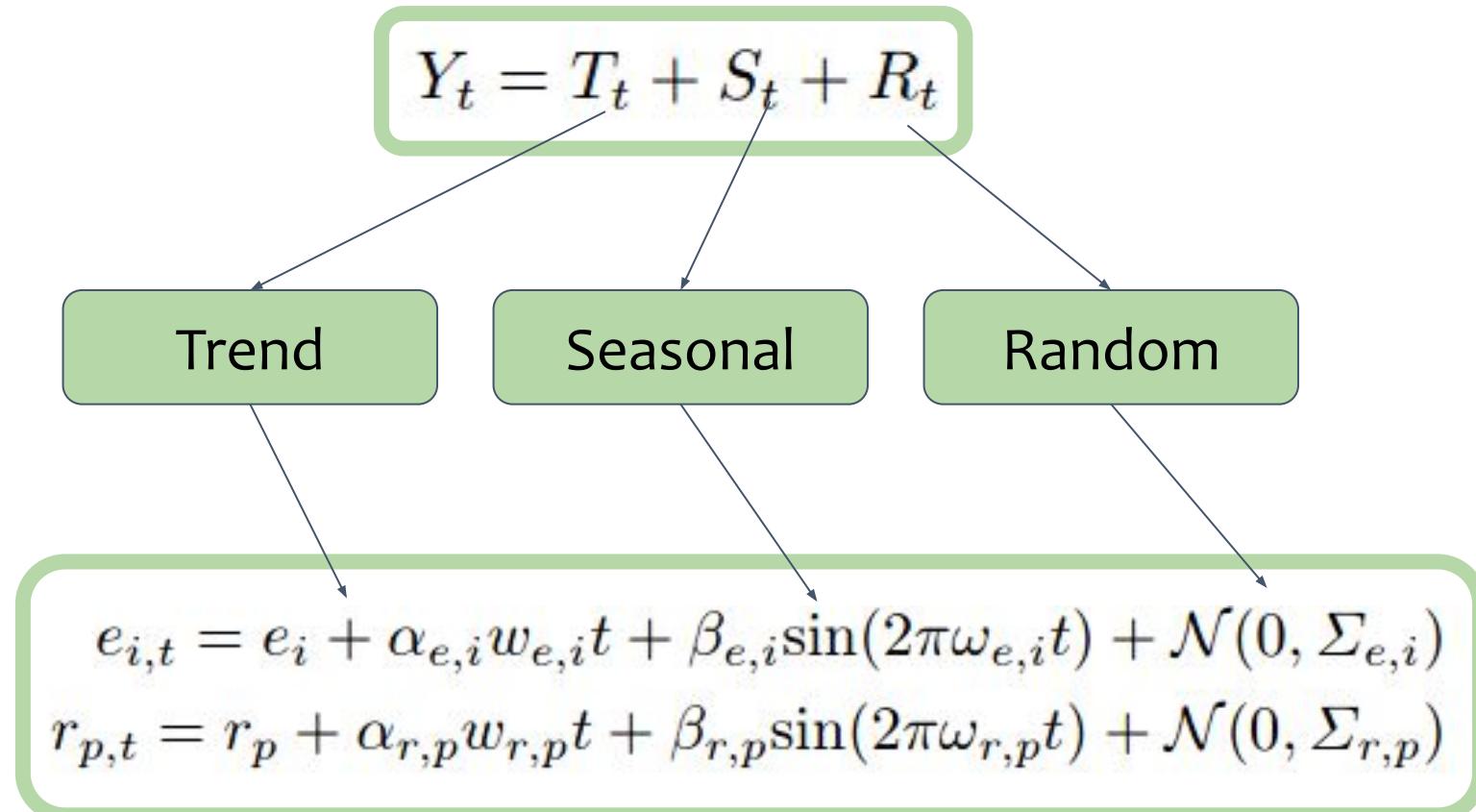
Dynamic Embedding



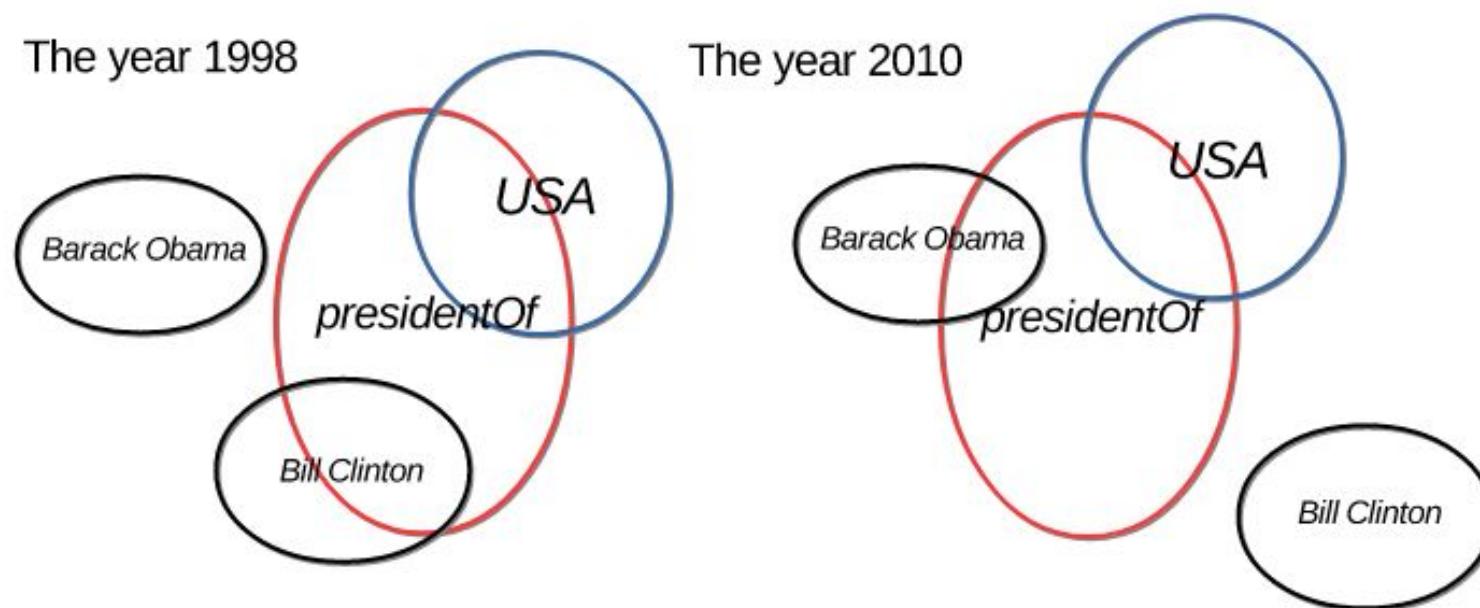
- Models entities as time series evolving in the vector space
- Time-series decomposition



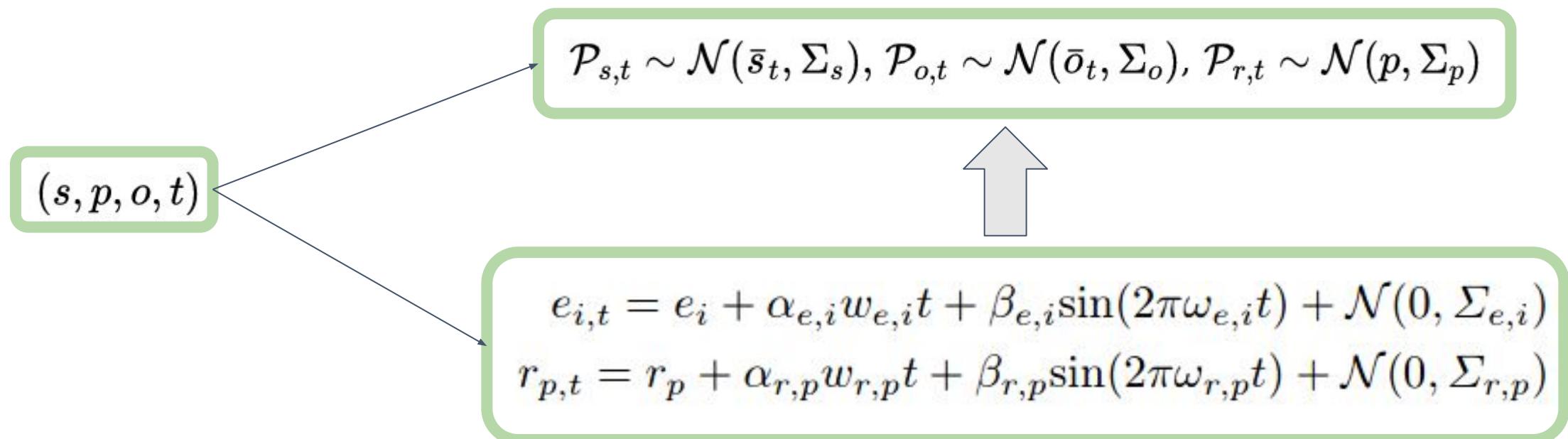
- Entity/relation temporal evolution



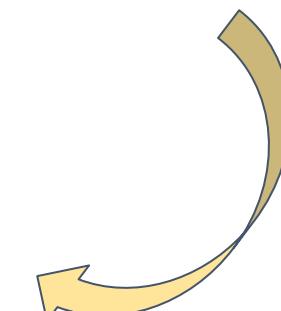
- Models uncertainty in entity/relation representation



- Models uncertainty in entity/relation representation



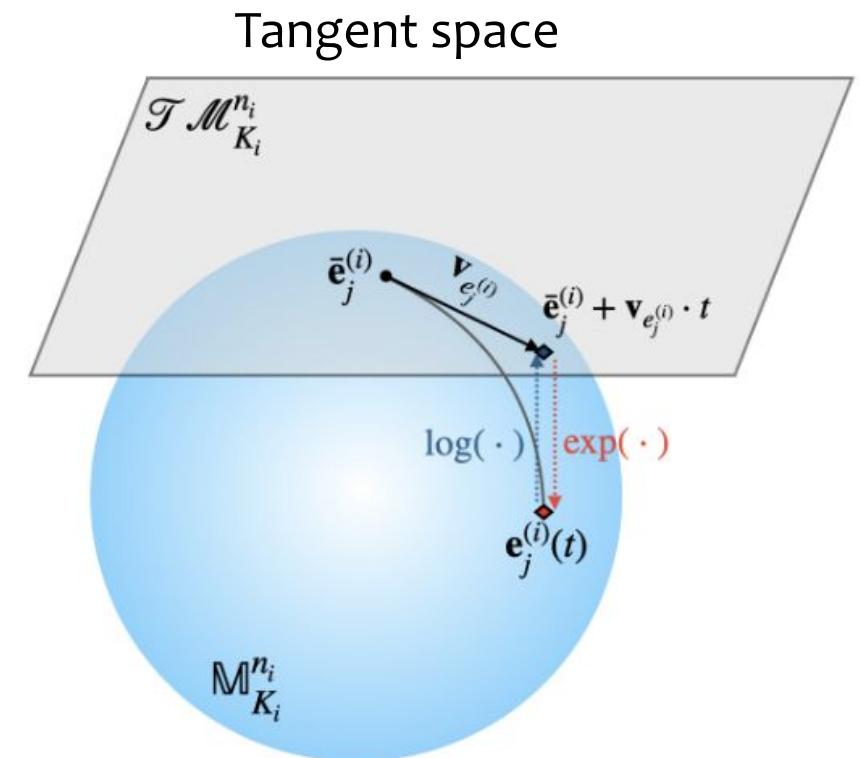
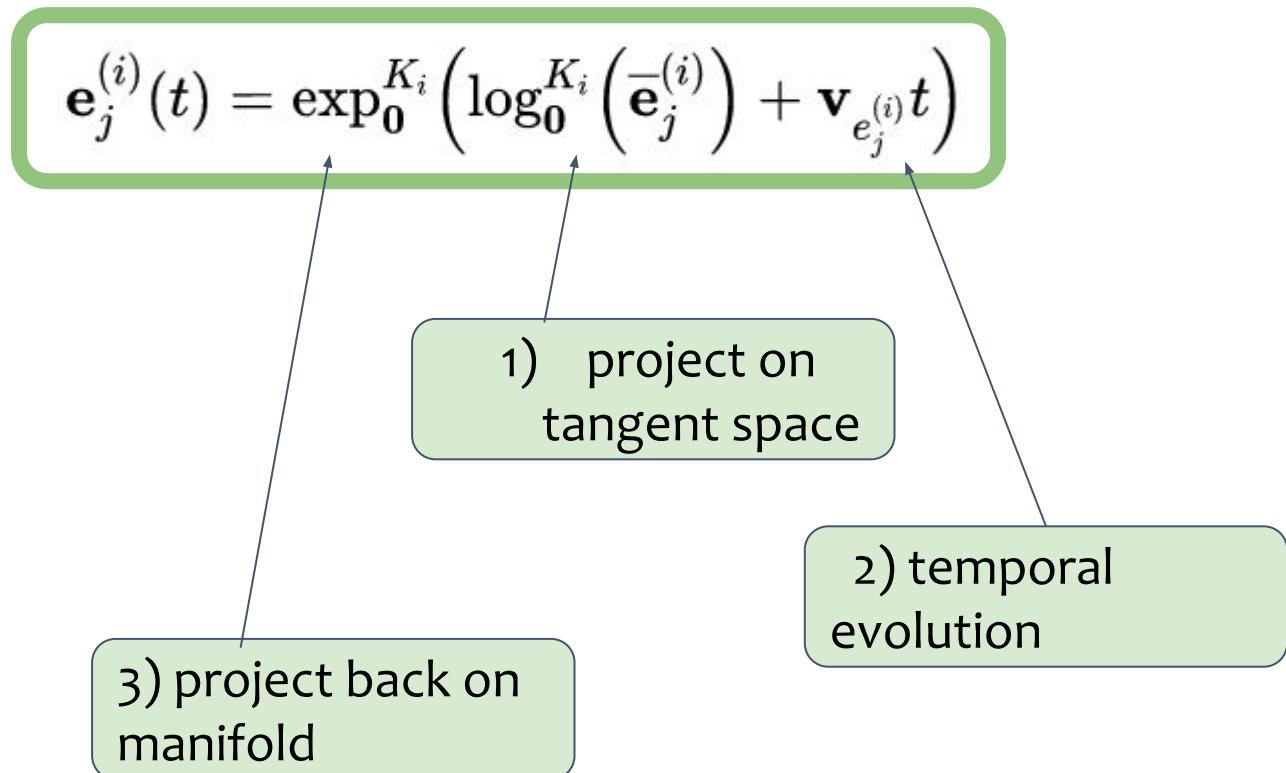
- Score function is inspired by TransE

$$\mathbf{s} - \mathbf{o} \approx \mathbf{p} \longrightarrow \mathcal{P}_{e,t} = \mathcal{P}_{s,t} - \mathcal{P}_{o,t} \approx \mathcal{P}_{r,t}$$
$$f_t(s, p, o) = \frac{1}{2}(\mathcal{D}_{KL}(P_{p,t}, P_{e,t}) + \mathcal{D}_{KL}(P_{e,t}, P_{p,t}))$$


- Disadvantage:
 - Embedding lie on Euclidean space
 - It is not suitable for modeling various structures such as hierarchy

DyERNIE (EMNLP'20)

- Models entity evolution on manifolds



Han, Zhen, et al. "DyERNIE: Dynamic Evolution of Riemannian Manifold Embeddings for Temporal Knowledge Graph Completion." EMNLP 2020

$$\mathbf{e}^t[n] = \begin{cases} \mathbf{a}_e[n]\sigma(\mathbf{w}_e[n]t + \mathbf{b}_e[n]), & \text{if } 1 \leq n \leq \gamma d \\ \mathbf{a}_e[n], & \text{if } \gamma d < n \leq d \end{cases}$$

Entity-specific static parameters

Entity-specific temporal
parameters

Summary of Insample TKGEs

- These models represent temporal entities/relations via time-specific mapping
- The mapping can be a simple or sophisticated function.
- Disadvantage:
 - They assume that timestamps in the inference time are already observed during training.

Agenda

Part 1. Introduction (Mojtaba, 9:00 - 9:30)

- 1.1 Tutorial Overview
- 1.2 Introduction to KG Embeddings
- 1.3 Motivation to go beyond triple-based KGs

Part 2. Temporal KG Embeddionsg (Mojtaba, 9:30 – 10:30)

- 2.1 Background
- 2.2 In sample timestamp KGE
- 2.3 Out of sample timestamp
 - 2.3.1 Interpolation-based TKGE
 - 2.3.2 Extrapolation-based TKGE

Coffee Break (10:30 – 11:00)

Part 3. Hyper-relational & N-ary KG embeddings (Bo, 11:00 – 12:30)

- 3.1 Introduction to hyper-relational & N-Ary KGs
 - 3.2 Functional embedding models
 - 3.2 Graph neural networks models
- Lunch Break (12:30 – 13:30)

Part 4. KG embeddings with literal and text (Daniel, 14:00 – 15:00)

- 4.1 Incorporating entity descriptions
- 4.2 Incorporate numeric literals as features
- 4.3 Incorporating relation descriptions

Part 5. Advanced Topic (Michael Cochez, 15:00-15:30 & 16:00 – 16:30)

- 5.1 Complex query embeddings
- 5.2 Inductive learning settings
- 5.3 Incorporating ontology, LLM, etc.

Part 6. Conclusion & Future Works (Michael Cochez, 16:30 – 17:00)

- 6.1 Conclusion
- 6.2 Future Works

Out of sample timestamp Temporal KGE

- The quadruples in the test set include timestamps that have **NOT** been observed during training
- Out of sample TKGE are classified into the following:
 - Interpolation-based TKGE
 - e.g., Facts in the training set cover all years between 2000 and 2022 except the years 2005 and 2006
 - All new facts in test set contain timestamps of 2005 and 2006
 - Extrapolation-based TKGE
 - e.g., Facts in the training set cover all years between 2000 and 2022
 - All new facts in test set contain timestamps after 2022

Out of sample timestamp Temporal KGE

- Interpolation-based TKGE
 - Some of models introduced in in-sample timestamp TKGEs that can handle also missing timestamps
 - e.g., DE-DistMult reports results on missing timestamps.
 - A variant of the ICEWS14 dataset is created by including every fact except those on the 5th, 15th, and 25th day of each month in the train set.
 - DE-DistMult outperformed DistMult

Out of sample timestamp Temporal KGE

○ Results:

Model	Variation	MRR	Hit@1	Hit@3	Hit@10
DE-TransE	No variation (Activation function: <i>Sine</i>)	0.326	12.4	46.7	68.6
DE-DistMult	No variation (Activation function: <i>Sine</i>)	0.501	39.2	56.9	70.8
DE-DistMult	Activation function: <i>Tanh</i>	0.486	37.5	54.7	70.1
DE-DistMult	Activation function: <i>Sigmoid</i>	0.484	37.0	54.6	70.6
DE-DistMult	Activation function: <i>Leaky ReLU</i>	0.478	36.3	54.2	70.1
DE-DistMult	Activation function: <i>Squared Exponential</i>	0.501	39.0	56.8	70.9
DE-TransE	Diachronic embedding for both entities and relations	0.324	12.7	46.1	68.0
DE-DistMult	Diachronic embedding for both entities and relations	0.502	39.4	56.6	70.4
DistMult	Generalizing to unseen timestamps	0.410	30.2	46.2	62.0
DE-DistMult	Generalizing to unseen timestamps	0.452	34.5	51.3	65.4
DE-DistMult	$a_v[n] = 1$ for $1 \leq n \leq \gamma d$ for all $v \in \mathcal{V}$	0.458	34.4	51.8	68.5
DE-DistMult	$w_v[n] = 1$ for $1 \leq n \leq \gamma d$ for all $v \in \mathcal{V}$	0.470	36.4	53.1	67.1
DE-DistMult	$b_v[n] = 0$ for $1 \leq n \leq \gamma d$ for all $v \in \mathcal{V}$	0.498	38.9	56.2	70.4

Out of sample timestamp Temporal KGE

- The quadruples in the test set include timestamps that have **NOT** been observed during training
- Out of sample TKGE are classified into the following:
 - Interpolation-based TKGE
 - e.g., Facts in the training set cover all years between 2000 and 2022 except the years 2005 and 2006
 - All new facts in test set contain timestamps of 2005 and 2006
 - Extrapolation-based TKGE
 - e.g., Facts in the training set cover all years between 2000 and 2022
 - All new facts in test set contain timestamps after 2022

Out of sample timestamp Temporal KGE

- Extrapolation-based TKGEs
 - They are also known as future link prediction methods
 - They mainly represent TKG as different snapshots

$$\mathcal{G} = \{G_{t_1}, \dots, G_{t_n}\} \quad G_{t_i} = \{(s, p, o, t_i)\}$$

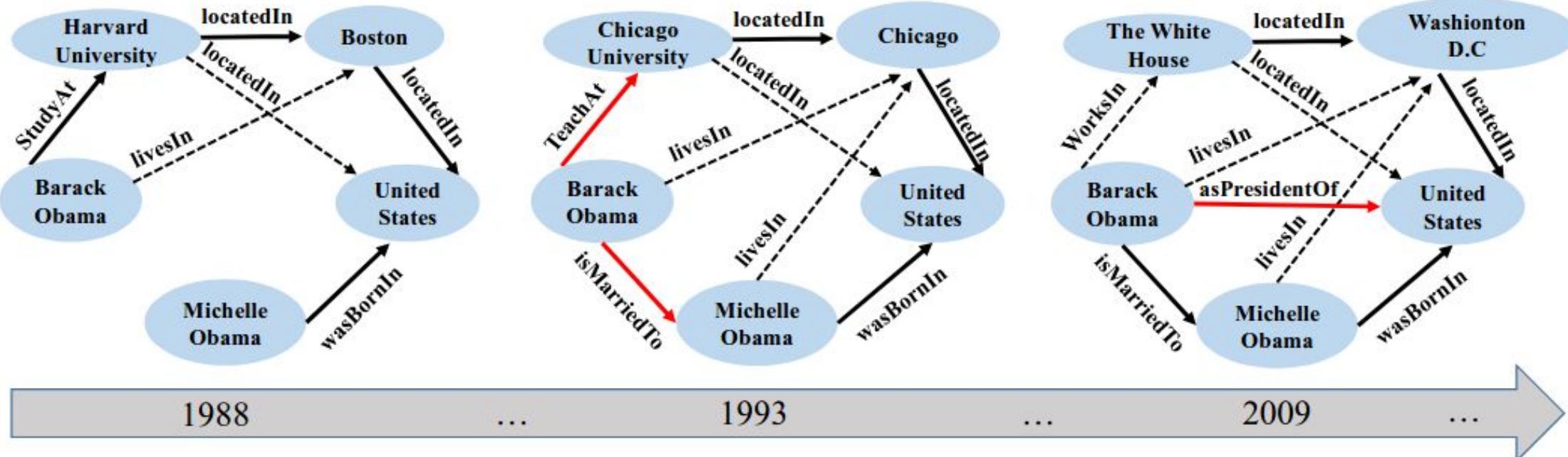
- **Markov process models:** next snapshot state is depended on the previous snapshot

$$\mathbf{E}_t \rightarrow \mathbf{E}_{t+1}$$

- **Autoregressive models:** Next snapshot state depends on m previous snapshots

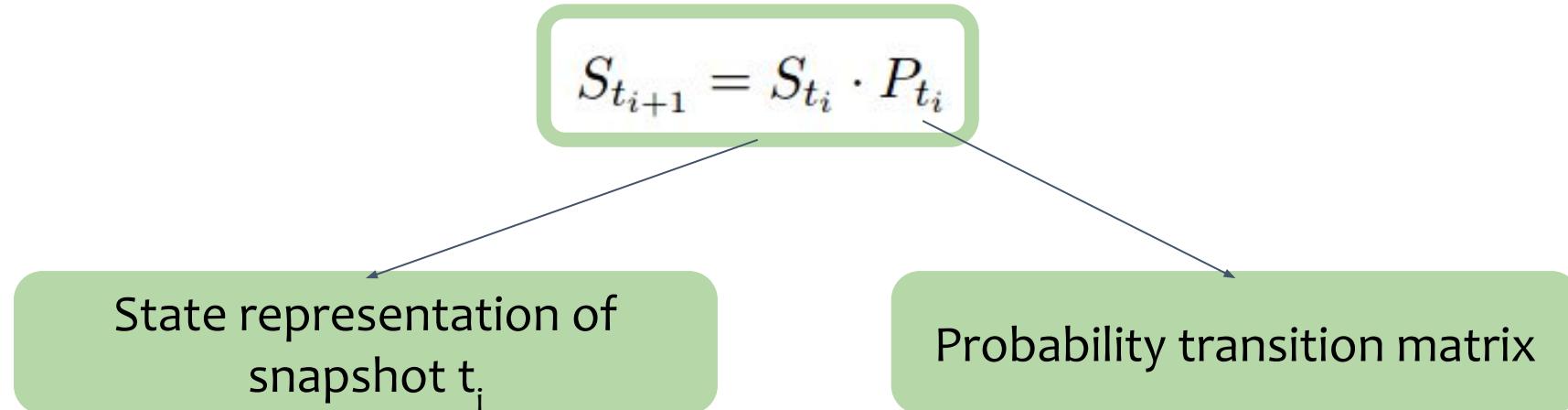
$$\mathbf{E}_{t-m}, \dots, \mathbf{E}_{t-2}, \mathbf{E}_{t-1}, \mathbf{E}_t \rightarrow \mathbf{E}_{t+1}$$

Out of sample timestamp Temporal KGE

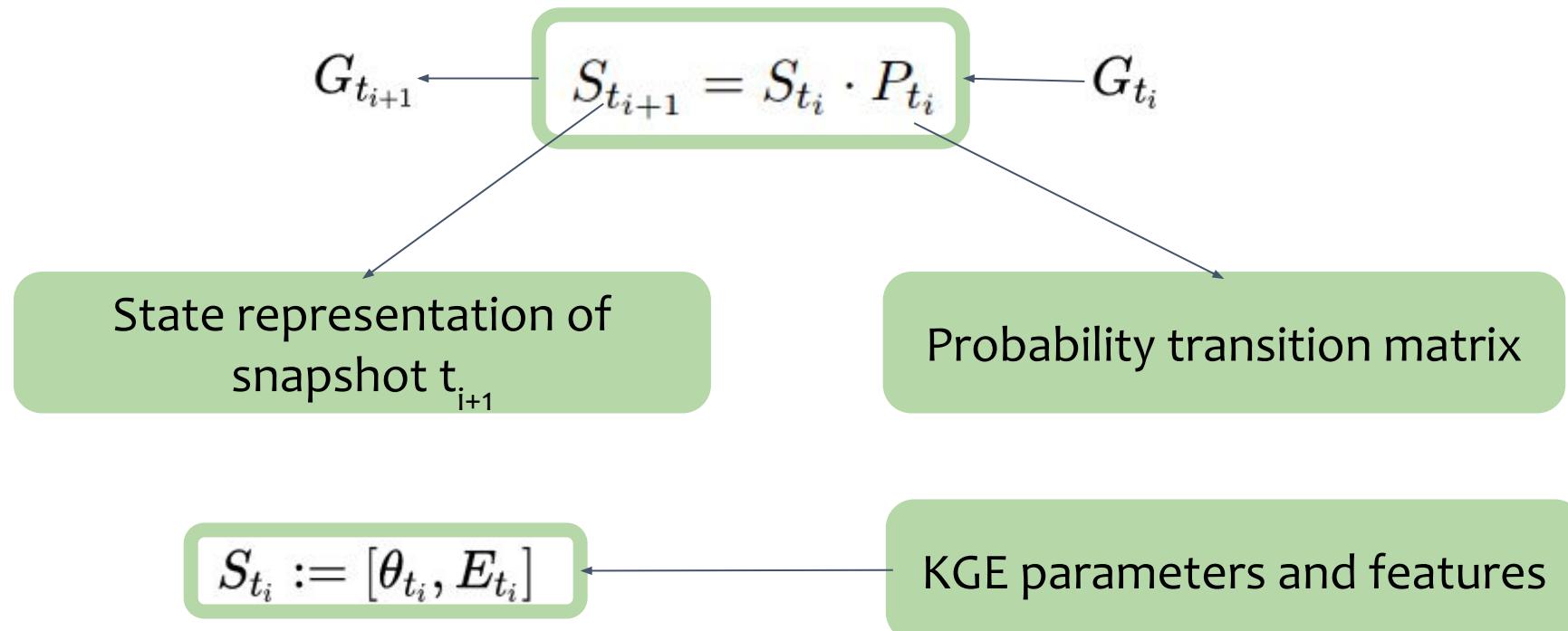


Markov process models (RTFE (NAACL'21))

- It represents TKG as sequences of graphs
- It treats the sequence of graphs as a Markov chain with transitions from the previous state to the next state



- KGE parameters and features form state vector

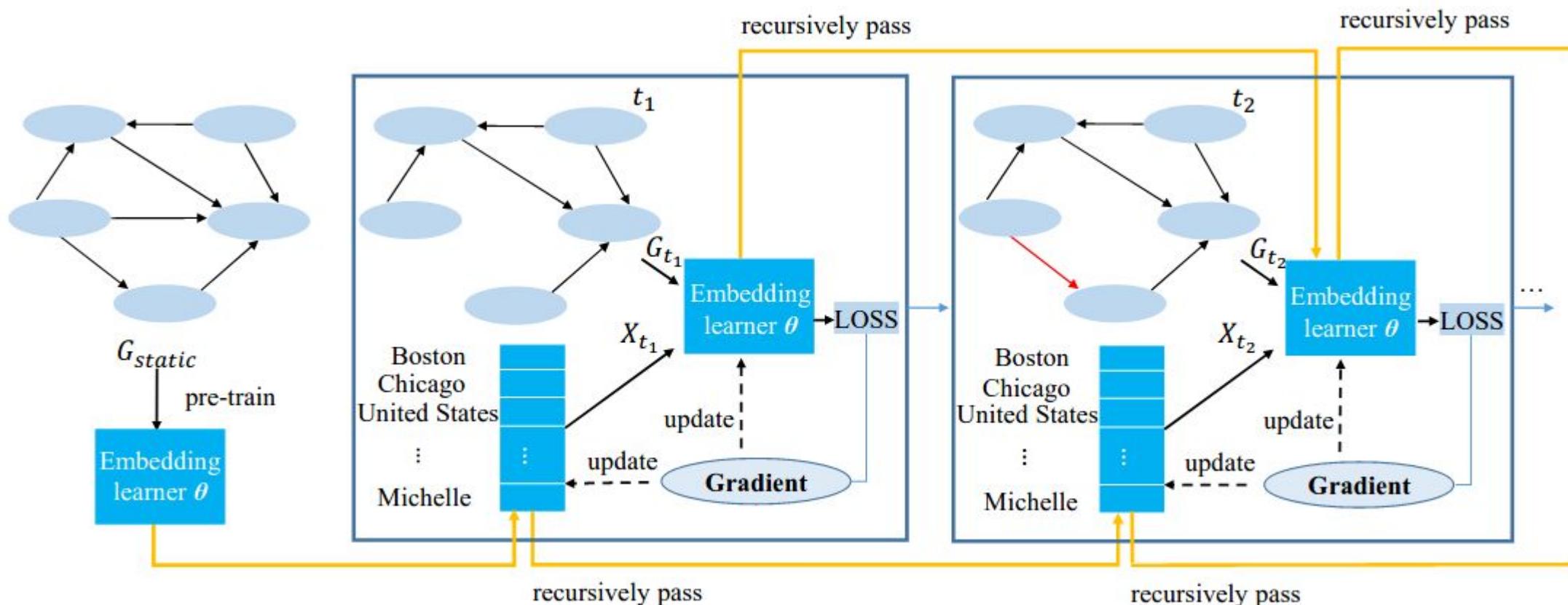


- It defines probability of the future snapshot state given previous snapshot state

$$\begin{aligned} P(X_{t_{i+1}}, \theta_{t_{i+1}} \mid E_{t_1}, \dots, E_{t_i}; \theta_{t_1}, \dots, \theta_{t_i}) \\ = P(E_{t_{i+1}}, \theta_{t_{i+1}} \mid E_{t_i}, \theta_{t_i}) \end{aligned}$$

Step1: represent TKG as static KG and train with static KGEs as initialization

Step n: use previous state as initialization to obtain next state



- Update Formula:

$$\begin{aligned}\theta_{t_{i+1}} &= \theta_{t_i} - \alpha \cdot \nabla_{\theta} l(\theta_{t_i}, X_{t_i}, G_{t_i}) \\ X_{t_{i+1}} &= X_{t_i} - \alpha \cdot \nabla_X l(\theta_{t_i}, X_{t_i}, G_{t_i})\end{aligned}$$

Static KGE

Train per snapshot TKGE

Algorithm 1 training and testing of RTFE

Input: TKG $G_{train} = \{G_1, \dots, G_n\}$, $G_{test} = \{G'_1, \dots, G'_n\}$,
 epochs_{static} , epochs_{tem}

- 1: **initialize:** $\mathbf{X}_{static}^{(0)} \leftarrow \text{random_initialize}$, $\theta_{static}^{(0)} \leftarrow \text{random_initialize}$
- 2: $G_{static} := \bigcup_{i=1}^n G_i$
- 3: **for** $i = 1, 2, \dots, \text{epochs}_{static}$ **do**
- 4: $\theta_{static}^{(i)} = \theta_{static}^{(i-1)} - \alpha \nabla_{\theta} l(\theta_{static}^{(i-1)}, \mathbf{X}_{static}^{(i-1)}, G_{static})$
- 5: $\mathbf{X}_{static}^{(i)} = \mathbf{X}_{static}^{(i-1)} - \alpha \nabla_{\mathbf{X}} l(\theta_{static}^{(i-1)}, \mathbf{X}_{static}^{(i-1)}, G_{static})$
- 6: **end for**
- 7: $\mathbf{X}_t^{(0)} := \mathbf{X}_{static}^{(i)}$
- 8: $\theta_t^{(0)} := \text{random_initialize}$
- 9: **for** $t = 1, 2, \dots, n$ **do**
- 10: **for** $i = 1, 2, \dots, \text{epochs}_{tem}$ **do**
- 11: $\theta_t^{(i)} = \theta_t^{(i-1)} - \alpha \nabla_{\theta} l(\theta_t^{(i-1)}, \mathbf{X}_t^{(i-1)}, G_t)$
- 12: $\mathbf{X}_t^{(i)} = \mathbf{X}_t^{(i-1)} - \alpha \nabla_{\mathbf{X}} l(\theta_t^{(i-1)}, \mathbf{X}_t^{(i-1)}, G_t)$
- 13: **end for**
- 14: metric _{t} \leftarrow test G'_t using $\theta_t^{(i)}$ and $\mathbf{X}_t^{(i)}$
- 15: $\theta_{t+1}^{(0)} := \theta_t^{(i)}$
- 16: $\mathbf{X}_{t+1}^{(0)} := \mathbf{X}_t^{(i)}$
- 17: **end for**
- 18: metric $= (\sum_{t=1}^n \|G'_t\| * \text{metric}_t) / \sum_{t=1}^n \|G'_t\|$
- 19: **return** metric

Update in each
snapshot

Summary of Markov process models

- They use previous snapshot to obtain representation of current snapshot
- Disadvantage:
 - The evolution sometimes dependent on more than one snapshot before.
 - Simple assumption may loose information

Out of sample timestamp Temporal KGE

- Extrapolation-based TKGEs

- They are also known as future link prediction methods
- They mainly represent TKG as different snapshots

$$\mathcal{G} = \{G_{t_1}, \dots, G_{t_n}\} \quad G_{t_i} = \{(s, p, o, t_i)\}$$

- **Markov process models:** next snapshot state is depended on the previous snapshot

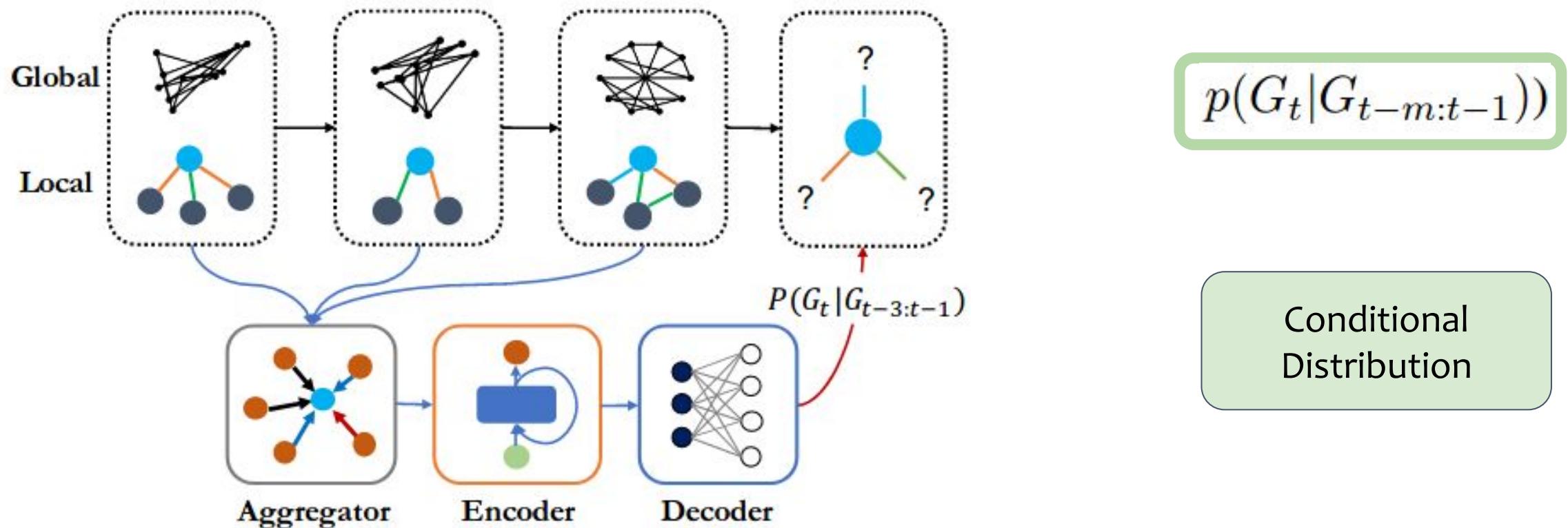
$$\mathbf{E}_t \rightarrow \mathbf{E}_{t+1}$$

- **Autoregressive models:** Next snapshot state depends on m previous snapshots

$$\mathbf{E}_{t-m}, \dots, \mathbf{E}_{t-2}, \mathbf{E}_{t-1}, \mathbf{E}_t \rightarrow \mathbf{E}_{t+1}$$

RE-NET (EMNLP'20)

- Overall Architecture: each snapshot depends on m previous snapshots

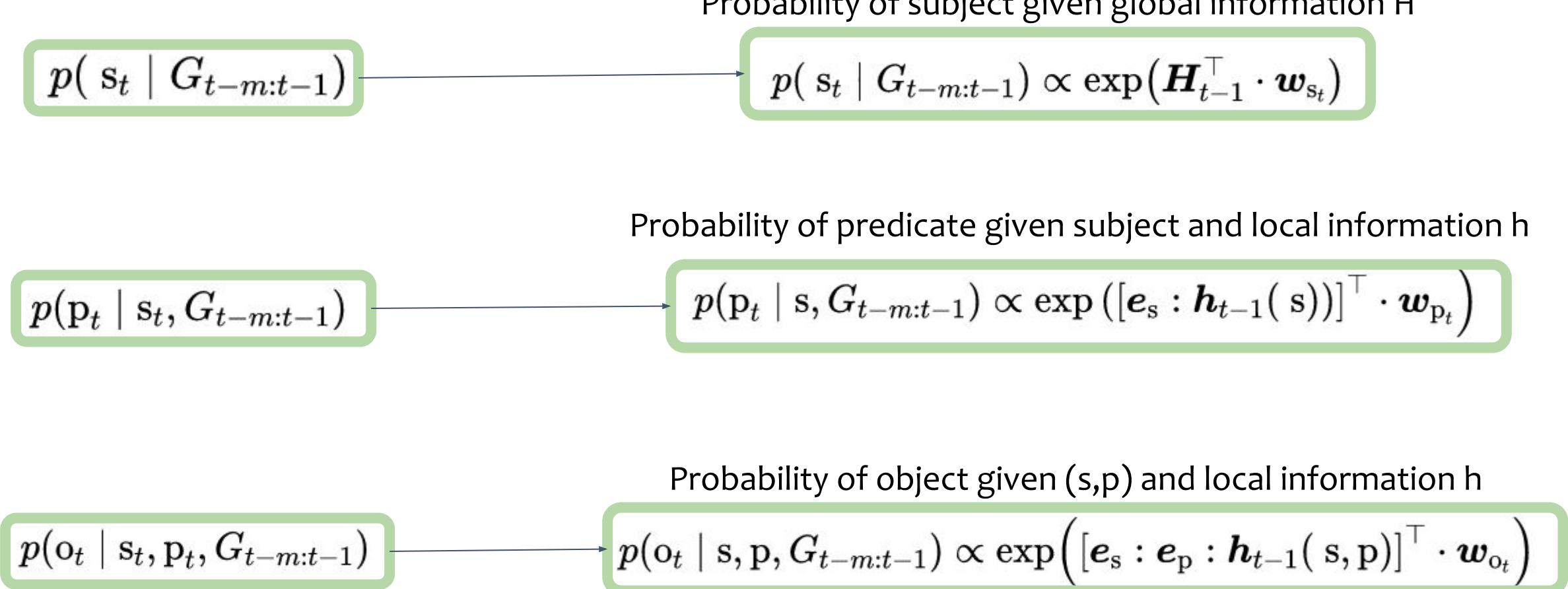


RE-NET (EMNLP'20)

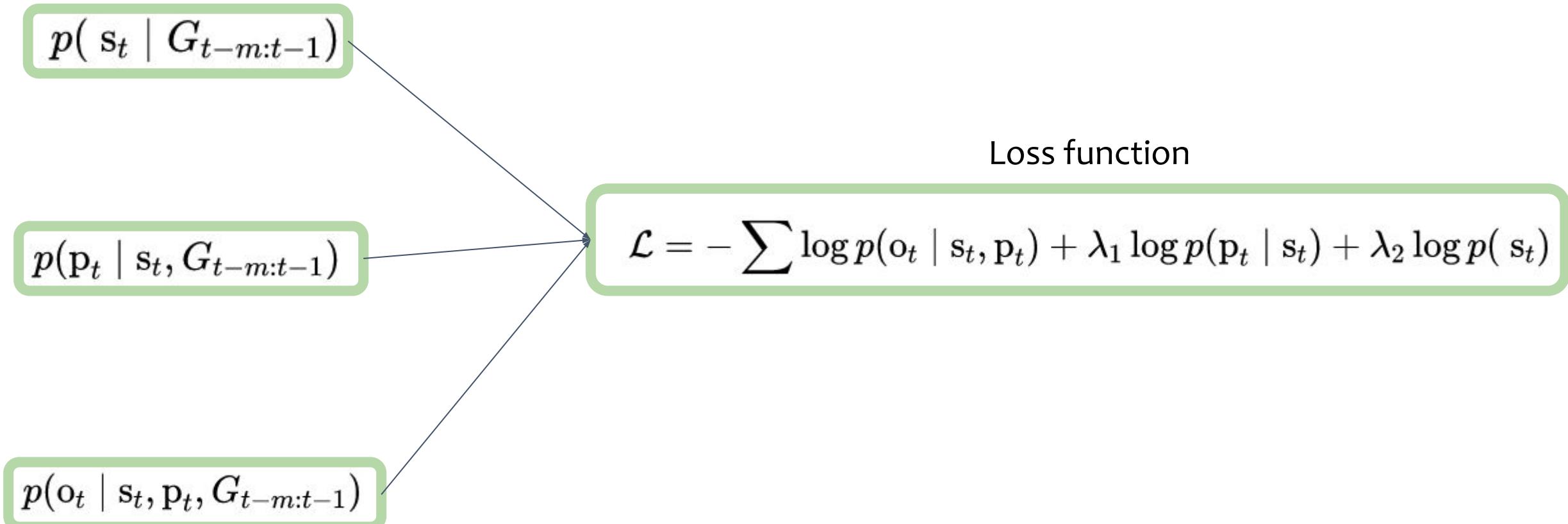
Joint distribution
of all snapshots

Used assumption:
Independency of snapshots
and facts

$$\begin{aligned} p(G) &= \prod_t \prod_{(s_t, p_t, o_t) \in G_t} p(s_t, p_t, o_t \mid G_{t-m:t-1}) \\ &= \prod_t \prod_{(s_t, p_t, o_t) \in G_t} p(s_t \mid G_{t-m:t-1}) p(p_t \mid s_t, G_{t-m:t-1}) \\ &\quad \cdot p(o_t \mid s_t, p_t, G_{t-m:t-1}). \end{aligned}$$



RE-NET (EMNLP'20)



RE-NET (EMNLP'20)

Probability of predicate given subject and local information h

$$p(p_t | s, G_{t-m:t-1}) \propto \exp ([e_s : h_{t-1}(s)])^\top \cdot w_{p_t}$$

Probability of subject given global information H

$$p(s_t | G_{t-m:t-1}) \propto \exp (H_{t-1}^\top \cdot w_{s_t})$$

Probability of object given (s,p) and local information h

$$p(o_t | s, p, G_{t-m:t-1}) \propto \exp ([e_s : e_p : h_{t-1}(s, p)]^\top \cdot w_{o_t})$$

global information

$$\begin{aligned} H_t &= \text{RNN}^1(g(G_t), H_{t-1}) \\ h_t(s, p) &= \text{RNN}^2\left(g\left(N_t^{(s)}\right), H_t, h_{t-1}(s, p)\right) \\ h_t(s) &= \text{RNN}^3\left(g\left(N_t^{(s)}\right), H_t, h_{t-1}(s)\right) \end{aligned}$$

local information

RE-NET (EMNLP'20)

$$\begin{aligned}\mathbf{H}_t &= \text{RNN}^1(g(G_t), \mathbf{H}_{t-1}) \\ \mathbf{h}_t(s, p) &= \text{RNN}^2\left(g\left(\mathcal{N}_t^{(s)}\right), \mathbf{H}_t, \mathbf{h}_{t-1}(s, p)\right) \\ \mathbf{h}_t(s) &= \text{RNN}^3\left(g\left(\mathcal{N}_t^{(s)}\right), \mathbf{H}_t, \mathbf{h}_{t-1}(s)\right)\end{aligned}$$

Aggregation over all events at time t

Neighborhood aggregation

$$g(\mathcal{N}_t^{(s,r)}) = \sum_{o \in \mathcal{N}_t^{(s,r)}} \alpha_o \mathbf{e}_o$$

Other Approaches

- TKGE and generally TK completion approaches are very broad.
- Here are list of the topics we did not cover:
 - Temporal Logic
 - Contextual temporal reasoning
 - Temporal complex query embedding



Reasoning beyond Triples: Recent Advances in Knowledge Graph Embeddings



Bo Xiong ¹



Mojtaba Nayyeri ¹



Daniel Daza ^{2,3}



Michael Cochez ²

¹ University of Stuttgart ² Vrije Universiteit Amsterdam ³ University of Amsterdam

Saturday 21 October, 09:00 - 17:30 (GMT+1) @ University of Birmingham - Birmingham, UK



Part 3. N-Ary & Hyper-relational KG Embeddings

Agenda

Part 1. Introduction (Bo & Mojtaba, 9:00 - 9:30)

- 1.1 Tutorial Overview (Bo)
- 1.2 Introduction to KG Embeddings
- 1.3 Motivation to go beyond triple-based KGs

Part 2. Temporal KG Embeddings (Mojtaba, 9:30 – 10:30)

- 2.1 Background
- 2.2 Interpolation-based TKGE
- 2.3 Extrapolation-based TKGE
- *Coffee Break (10:30 – 11:00)*

Part 3. N-ary & Hyper-relational embeddings (Bo, 11:00 – 12:30)

- 3.1 Introduction to hyper-relational & N-Ary KGs
- 3.2 N-ary relational embeddings
- 3.2 Hyper-relational embeddings
- *Lunch Break (12:30 – 14:00)*

Part 4. KG embeddings with literal and text (Daniel, 14:00 – 15:00)

- 4.1 Incorporating entity descriptions
- 4.2 Incorporate numeric literals as features
- 4.3 Incorporating relation descriptions

Part 5. Advanced Topic (Michael, 15:00-15:30 & 16:00 – 16:30)

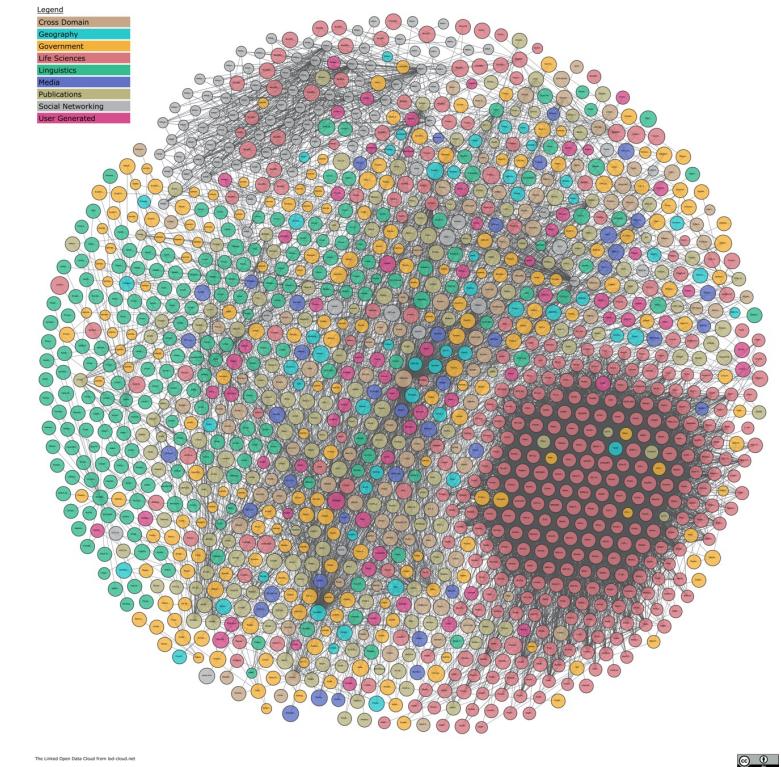
- 5.1 Complex query embeddings (Daniel)
- *Coffee Break (15:30 – 16:00)*
- 5.2 Neural Graph Databases
- 5.3 Inductive learning settings
- 5.3 Incorporating ontology, LLM, etc.

Part 6. Conclusion & Future Works (Michael, 16:30 – 17:00)

- 6.1 Conclusion
- 6.2 Future Works

Binary relations

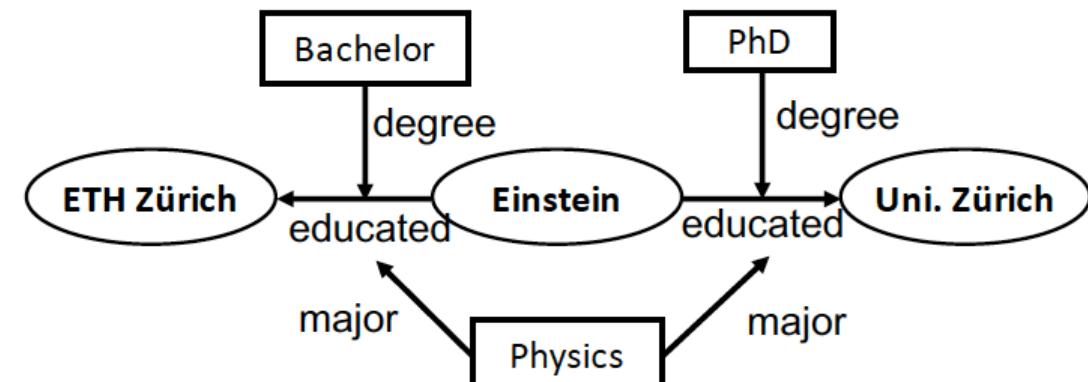
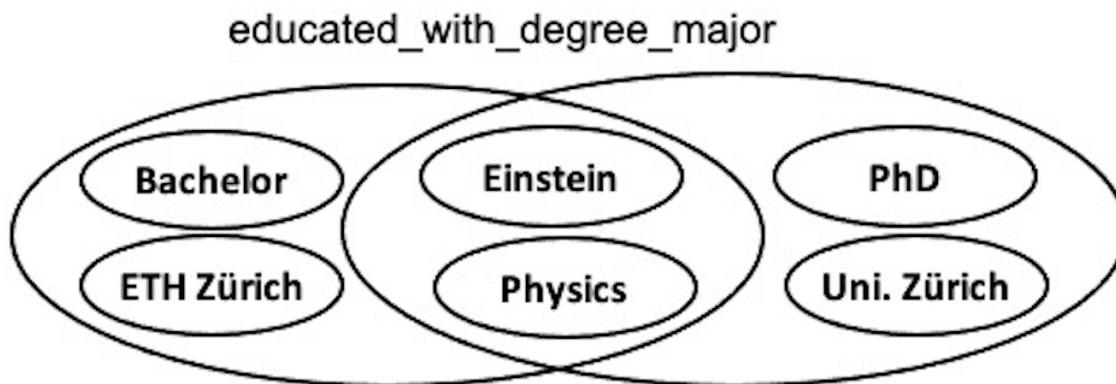
- Existing embeddings assume that relational data are instances of **binary relations**
 - e.g., *buy(Alice, Coffee)*
- In reality, a large portion of the relational data are from **non-binary relations**
 - e.g., *trade(Alice, Coffee, Bob, \$5)*
- In Freebase, more than **1/3 of the entities** participate in non-binary relations



<https://lod-cloud.net/>

N-Ary Relation vs Hyper-Relation

- N-Ary relational facts
 - Representation: **n-tuple** or **dictionary**
 - No primal relations and entities
 - Incompatible with **RDF-star**
- Hyper-relational facts
 - Representation: **triple + qualifiers**
 - Distinguish between primal triple and qualifiers
 - Compatible with **RDF-star**



Multi-fold (N-Ary) Relations

- N-Ary relational facts

John buys a "Lenny the Lion" book from books.example.com for \$15 as a gift

- Named N-Tuple

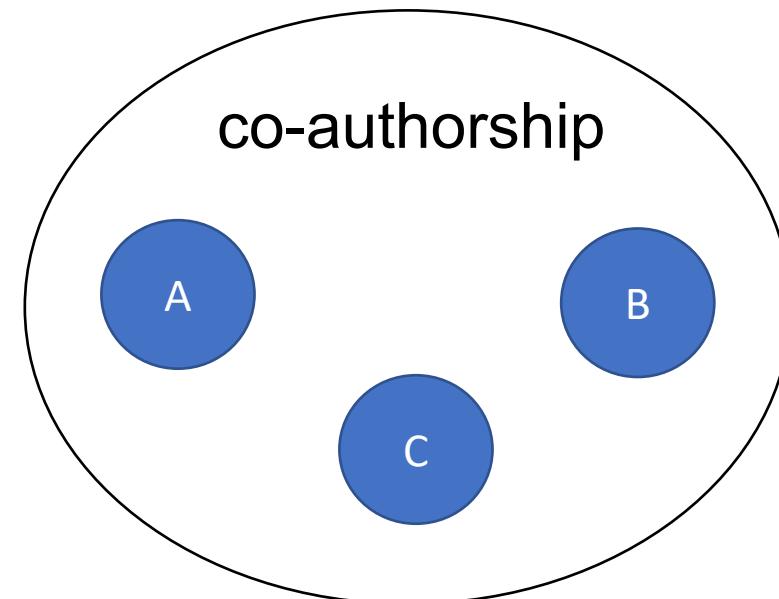
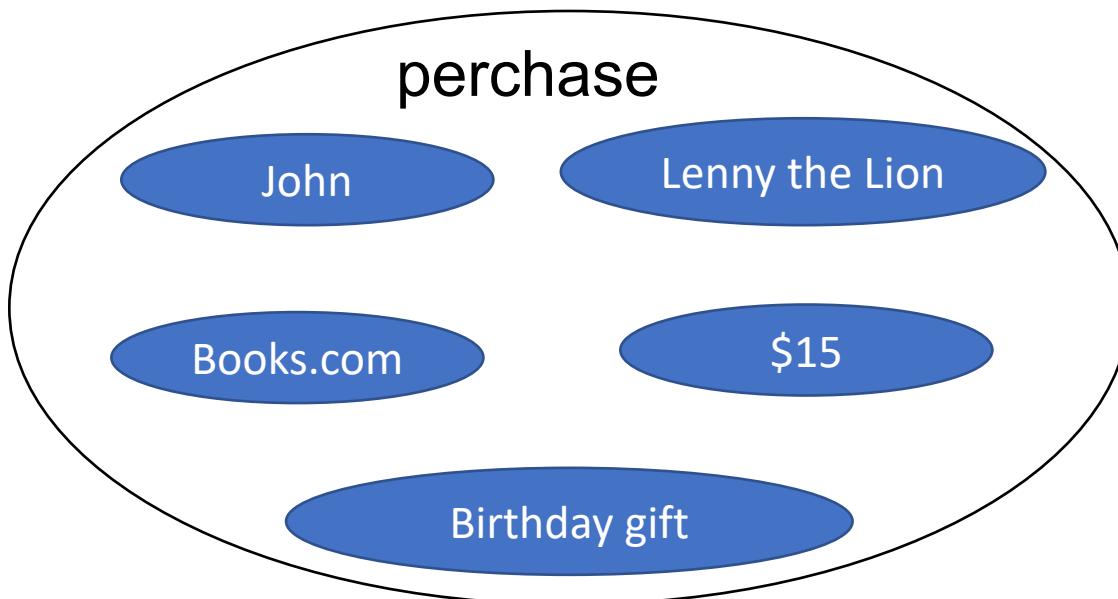
purchase(John, Lenny the Lion, books.com, \$15, gift)

- Dictionary (key-value pairs)

{buyer:John, item: Lenny the Lion, seller: books.com, price: \$15, purpose: gift”}

(Directed) Knowledge Hypergraph

- **(Directed) knowledge hypergraph**
 - The order of entities matters (anti-symmetric)
 - Otherwise symmetric relations, e.g., co-authorship

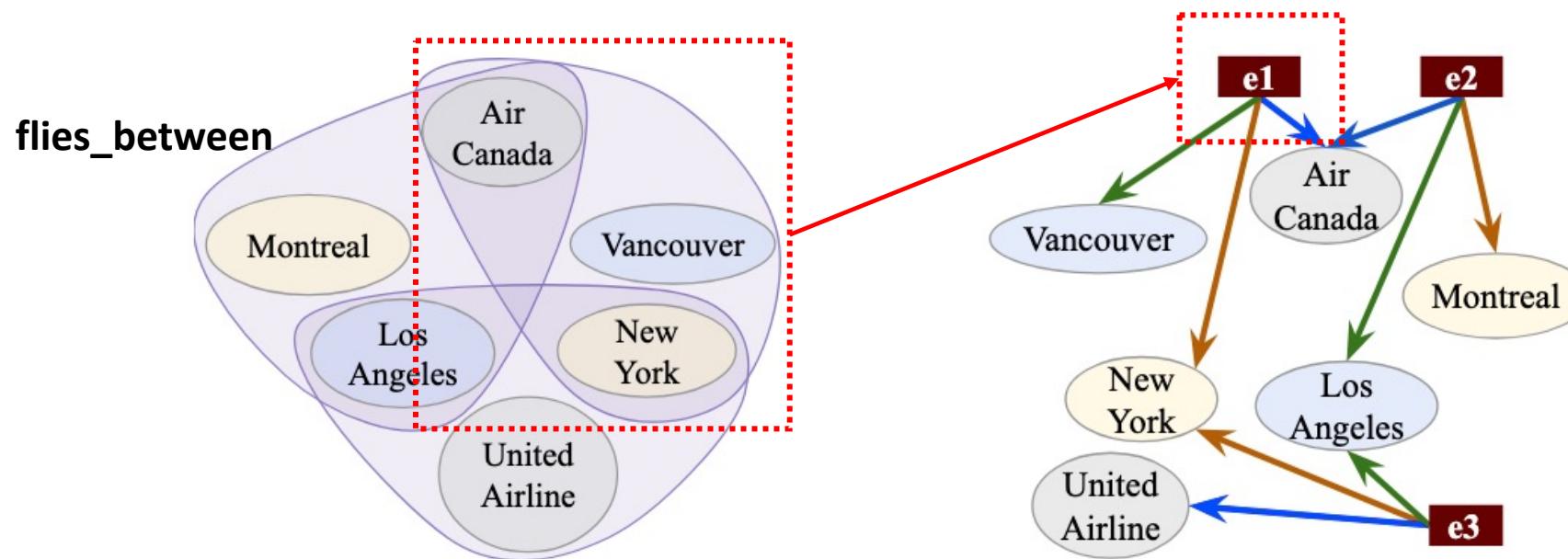


N-Ary KG vs Knowledge Hypergraph

- N-Ary KG (**tuple-based**) is equivalent to knowledge hypergraph
 - Knowledge hypergraph is a **graph structure** of N-ary KG (tuple)
- N-Ary KG (**key-value pairs**) is more powerful than knowledge hypergraph
- Both are not equivalent to hyper-relational KGs!
- All of them can be **reified** to triple-based KGs
 - but reification suffers from key issues

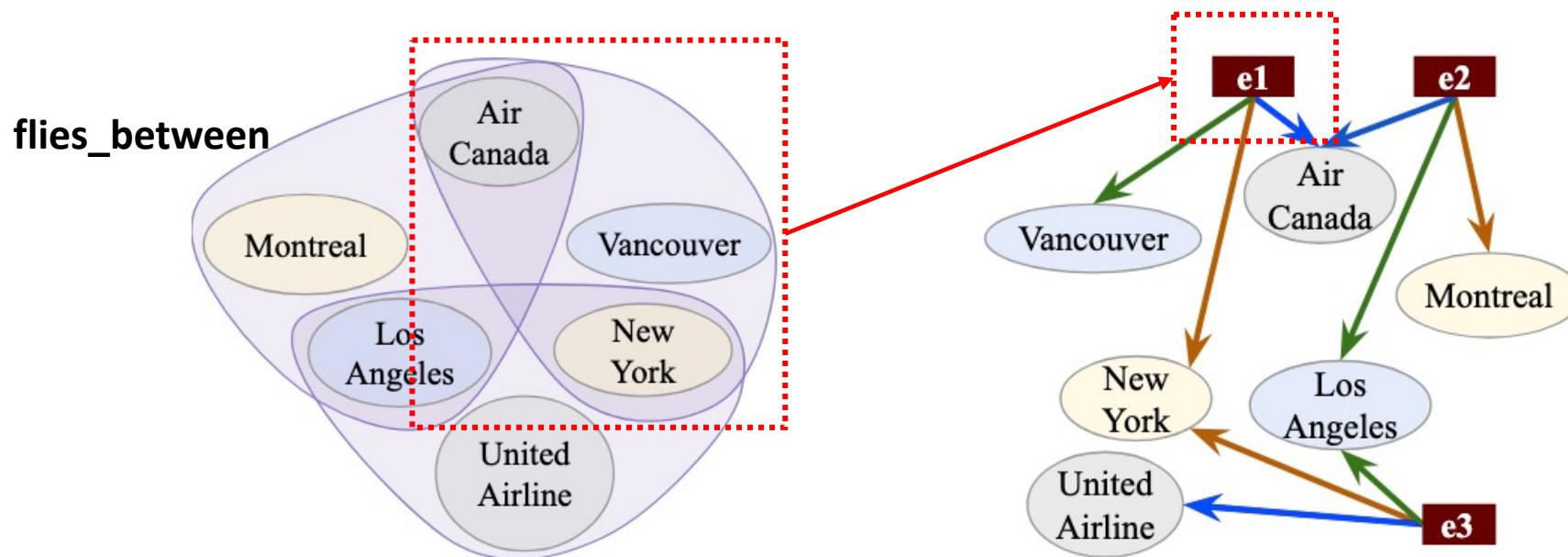
Standard Reification

- Converting higher-arity relations into binary ones
 1. for each relation, creating k new binary relations, one for each position
 2. for each fact, creating a new entity e
 3. connecting e to each of the k entities using the k binary relations



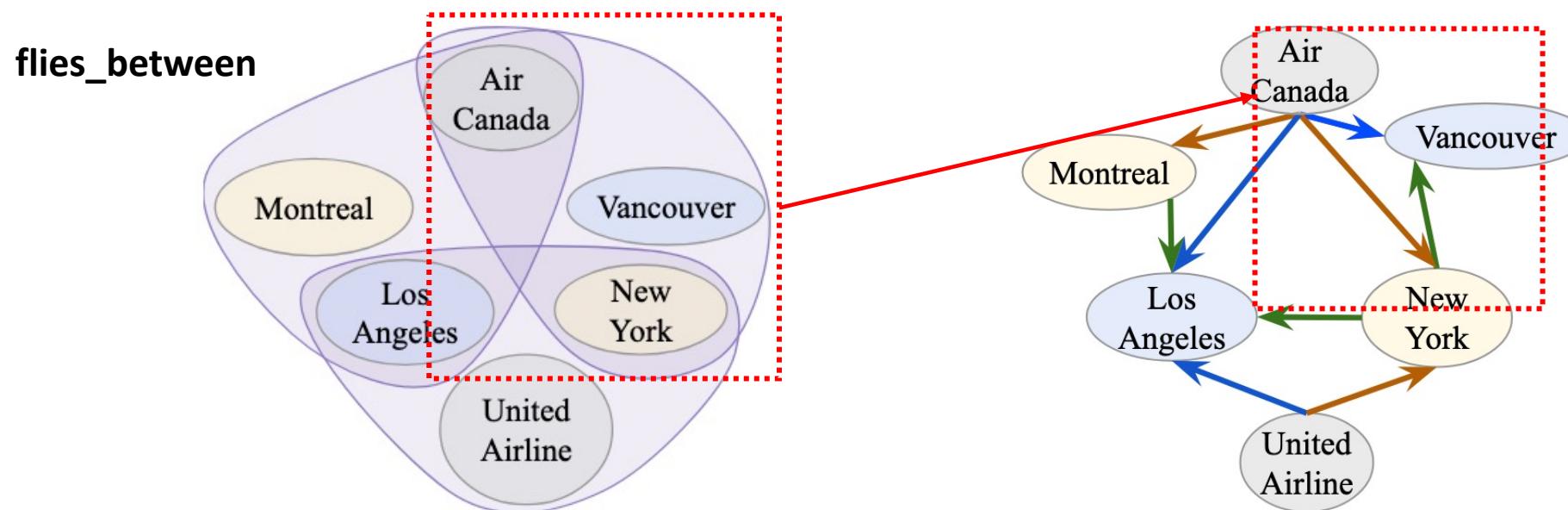
Standard Reification

- The conversion does not lose information
- But introduces new entities that model never encounters during training
 - Problematic in the test stage!



Star-to-Clique Reification

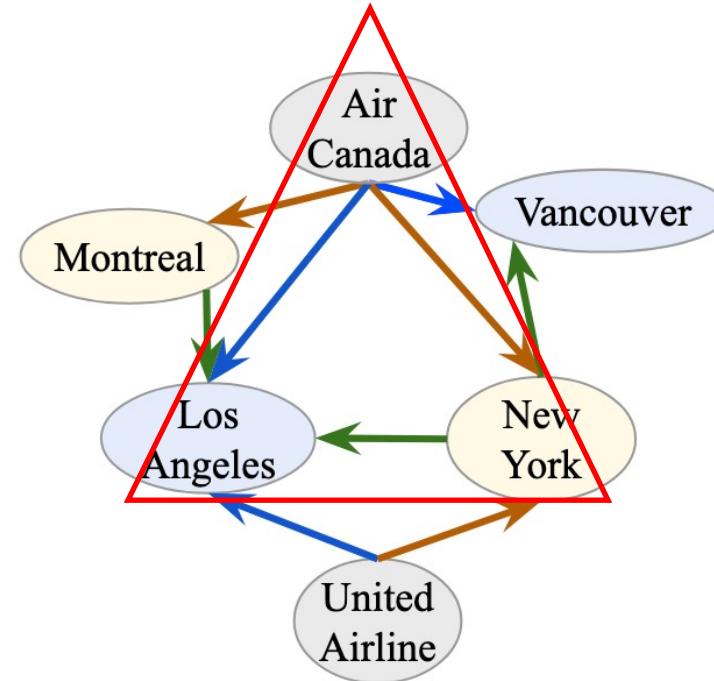
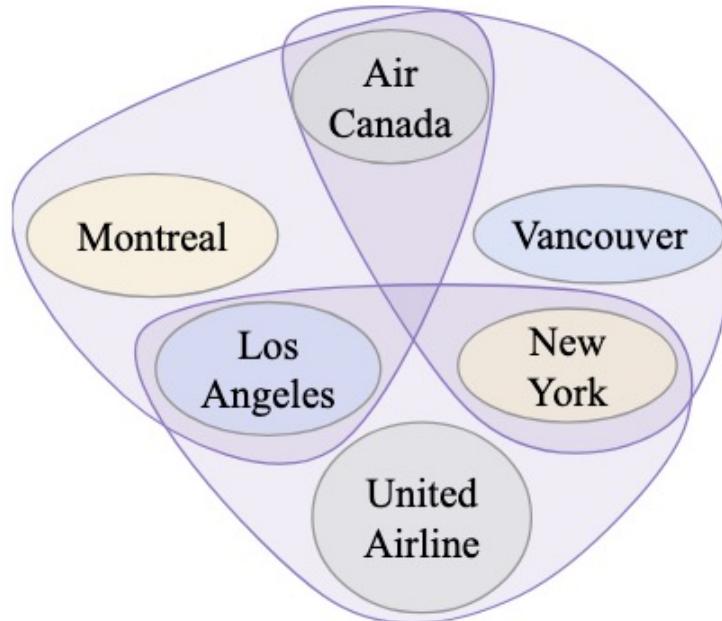
- Converting higher-arity relations into binary ones
 1. For each relation defined on k entities e_1, e_2, \dots, e_k
 2. converts a tuple defined on k entities into triples with distinct relations between all pairwise entities in the tuple



e.g., FB15K results from applying a star-to-clique conversion

Star-to-Clique Reification

- Does not introduce new entities
- **but loses structural information**



Agenda

Part 1. Introduction (Bo & Mojtaba, 9:00 - 9:30)

- 1.1 Tutorial Overview (Bo)
- 1.2 Introduction to KG Embeddings
- 1.3 Motivation to go beyond triple-based KGs

Part 2. Temporal KG Embeddings (Mojtaba, 9:30 – 10:30)

- 2.1 Background
- 2.2 Interpolation-based TKGE
- 2.3 Extrapolation-based TKGE
- *Coffee Break (10:30 – 11:00)*

Part 3. N-ary & Hyper-relational embeddings (Bo, 11:00 – 12:30)

- 3.1 Introduction to hyper-relational & N-Ary KGs
- 3.2 N-ary relational embeddings
- 3.2 Hyper-relational embeddings
- *Lunch Break (12:30 – 14:00)*

Part 4. KG embeddings with literal and text (Daniel, 14:00 – 15:00)

- 4.1 Incorporating entity descriptions
- 4.2 Incorporate numeric literals as features
- 4.3 Incorporating relation descriptions

Part 5. Advanced Topic (Michael, 15:00-15:30 & 16:00 – 16:30)

- 5.1 Complex query embeddings (Daniel)
- *Coffee Break (15:30 – 16:00)*
- 5.2 Neural Graph Databases
- 5.3 Inductive learning settings
- 5.3 Incorporating ontology, LLM, etc.

Part 6. Conclusion & Future Works (Michael, 16:30 – 17:00)

- 6.1 Conclusion
- 6.2 Future Works

N-Ary Relational KG embeddings

- Translational models
 - **mTransH** (IJCAI'16)
 - **RAE** (WWW'18)
 - **BoxE** (NeurIPS'20)
- Tensor decomposition models
 - **HSimpleE & HypE** (IJCAI'20)
 - **GETD** (WWW'20)
 - **S2S** (WWW'21)

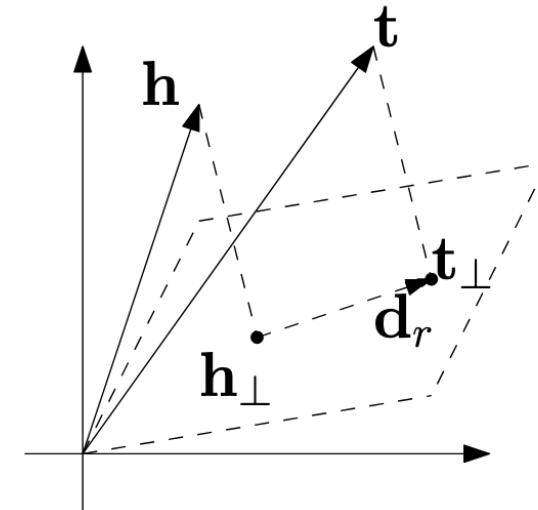
MTransH (IJCAI'16)

- TransH projects entities onto relation-specific hyperplanes

$$f_r(\mathbf{x}, \mathbf{y}) := \|\mathbb{P}_{\mathbf{n}_r}(\mathbf{x}) + \mathbf{b}_r - \mathbb{P}_{\mathbf{n}_r}(\mathbf{y})\|^2$$

where $\mathbb{P}_{\mathbf{n}_r}$ is the projection operator on the hyperplane.

- Overcomes the **one-to-N** problem of TransE



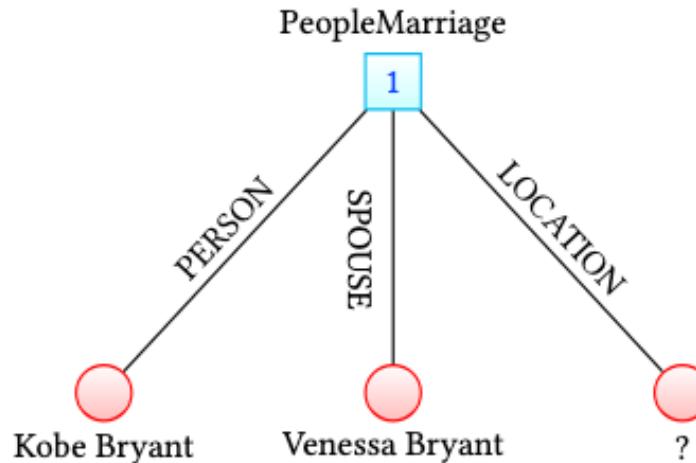
- mTransH generalizes TransH to multi-fold relation by attaching each n-ary relation with a hyperplane and projects entities onto relation-specific hyperplanes

$$f_r(\mathbf{x}_{\rho_1}, \mathbf{x}_{\rho_2}, \dots, \mathbf{x}_{\rho_{|\mathcal{M}(r)|}}) := \left\| \sum_{\rho \in \mathcal{M}(R_r)} \mathbf{a}_\rho \mathbb{P}_{\mathbf{n}_r}(\mathbf{x}_\rho) + \mathbf{b}_r \right\|^2$$

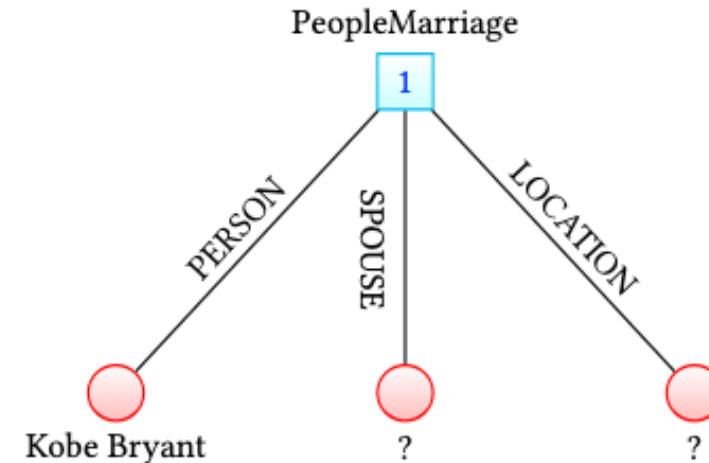
- m-TransH reduces to TransH for binary relations

Instance reconstruction

- **Link prediction** only predicts a single entity in a fact
- **Instance reconstruction** recovers multiple missing entities in a fact



(a) Link Prediction



(b) Instance Reconstruction

- **MTransH** suffers from **scalability issues** for instance construction
 - Time complexity: $O(N^{m-1})$

- **Intuition:** using **relatedness of entities** to filter all possible candidate entity pairs
 - **Relatedness:** two entities participate in the same fact are considered related
- Relatedness is modeled as a **binary classification** problem
 - related $p(x,y) = 1$
 - unrelated $p(x, y) = 0$
$$p(x \sim y) := \text{MLP}(\phi(x), \phi(y)).$$
- Overall RAE Model

$$E(\Theta_I, \Theta_C, \phi) := E_I(\Theta_I, \phi) + \lambda E_C(\Theta_C, \phi),$$



- **Schema based filtering**
 - Leverages type requirements on the entities dictated by the schema of a relation
- **Relatedness based filtering**
 - Filtering entities by comparing $p(x \sim y)$ with a threshold τ to predict if the two entities are related

Algorithm 1: SIR Algorithm

```
for every role  $\rho \in \mathcal{M}(r) \setminus \{\rho^*\}$  do
    |  $\mathcal{N}(x^*; \rho) := \text{SCH-FILTER}(x^*, r, \rho);$ 
end
 $\mathcal{N}(x^*; \rho^*) := \{x^*\};$ 
for every two roles  $\rho_i, \rho_j \in \mathcal{M}(r)$  with  $i < j$  do
    |  $\mathcal{P}(x^*; \rho_i, \rho_j) := \text{REL-FILTER}(\mathcal{N}(x^*; \rho_i), \mathcal{N}(x^*; \rho_j));$ 
end
 $\mathcal{I}(x^*) := \text{SPICE}(\{\mathcal{P}(x^*; \rho_i, \rho_j)\}, x^*);$ 
 $L(x^*) := \text{RANK}(\mathcal{I}(x^*));$ 
```

- Link prediction

Table 3: Link Prediction Performance

	FB15K		JK17K	
	TransH	RAE	mTransH	RAE
HIT@10	44.30%	44.27%	49.74%	50.35%
Mean Rank	171.24	171.03	237.82	229.27

- Relatedness prediction

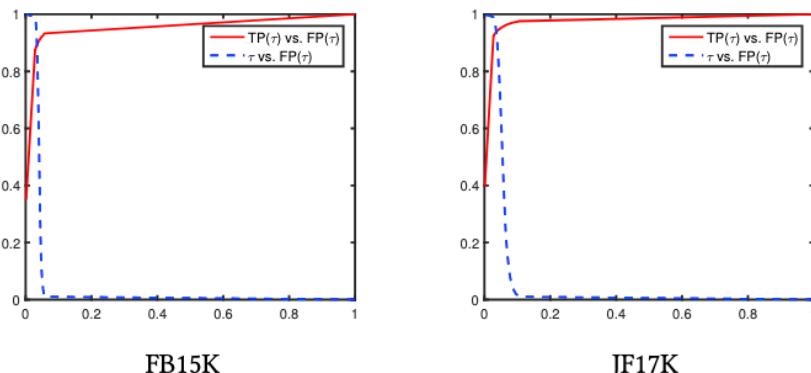


Figure 4: Relatedness prediction performances. Left: FB15K, TP(0.5)=90.38%, FP(0.5)=4.37%. Right: JF17K, TP(0.5)=95.53%, FP(0.5)=5.44%

- Instance construction

Table 4: Instance Reconstruction Performance of SIR on JF17K. $W(x^*)$ in SIR-Covered form. Coverage = 88.54%

arity	Top1	Top10	Top20	Top50	RNK
2	25.07%	52.68%	60.45%	74.63%	47.63
3	12.42%	27.29%	32.52%	41.21%	610.09
4	11.56%	29.48%	34.67%	41.88%	34844.11
5	2.78%	8.33%	9.72%	13.89%	10229.41
6	20.00%	60.00%	80.00%	100.00%	10.80
overall	19.24%	41.29%	47.89%	59.45%	2981.81

Full Expressiveness

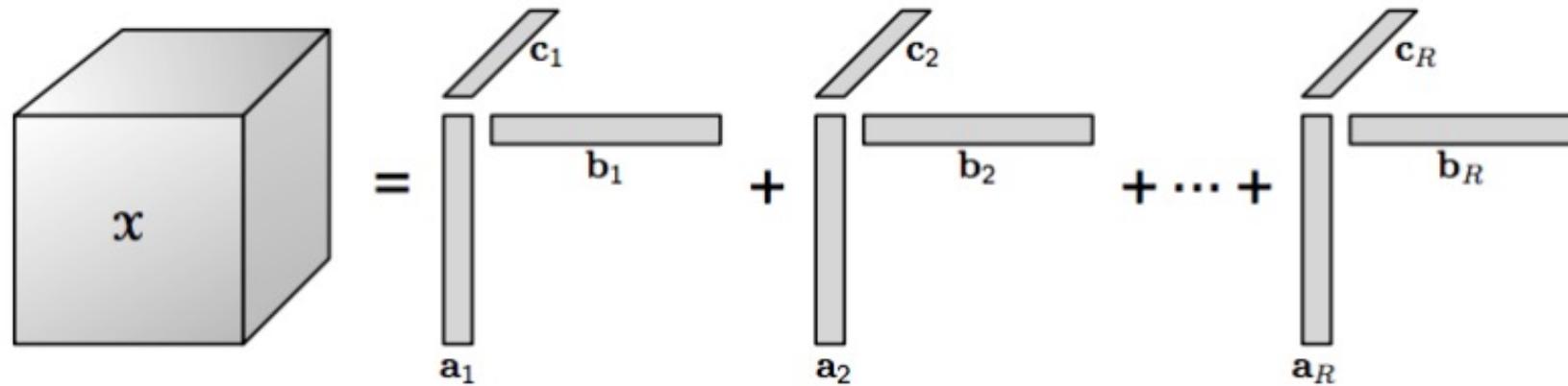
- TransH based methods cannot model certain types of relations
 - e.g., symmetry
- **Full Expressiveness:** an embedding model is fully expressive if for any ground truth over all entities and relations, there exist embeddings that accurately separate the true relational facts from the false ones.
- Full expressiveness guarantees the **completeness**

N-Ary Relational KG embeddings

- Translational models
 - **mTransH** (IJCAI'16)
 - **RAE** (WWW'18)
 - **BoxE** (NeurIPS'20)
- Tensor decomposition models
 - **HSimpleE & HypE** (IJCAI'20)
 - **GETD** (WWW'20)
 - **S2S** (WWW'21)

Tensor decomposition

- Approximate a higher-order tensor via a set of lower-order tensors



- Knowledge graph is three-order tensor
 - Head entities: represented along one mode.
 - Relations: represented along a second mode.
 - Tail entities: represented along a third mode.

Tensor decomposition

- Canonical Polyadic (CP) decomposition, **Simple (NeurIPS'18)**

$$\mathcal{T} \approx \sum_{r=1}^R \mathbf{u}_r^{(1)} \otimes \mathbf{u}_r^{(1)} \otimes \cdots \otimes \mathbf{u}_r^{(N)}.$$

- Tucker Decomposition, **TuckER (EMNLP'18)**

$$\begin{aligned}\mathcal{X} &\approx \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_n \mathbf{A}^{(n)} \\ &= \sum_{j_1=1}^{J_1} \sum_{j_2=1}^{J_2} \cdots \sum_{j_n=1}^{J_n} g_{j_1 j_2 \cdots j_n} \mathbf{a}_{j_1}^{(1)} \circ \mathbf{a}_{j_2}^{(2)} \circ \cdots \circ \mathbf{a}_{j_n}^{(n)},\end{aligned}$$

- Tensor Ring (TR) Decomposition, **GETD (WWW'20)**

$$x_{i_1 i_2 \cdots i_n} \approx \text{trace}\{Z_1(i_1) Z_2(i_2) \cdots Z_n(i_n)\} = \text{trace}\{\prod_{k=1}^n Z_k(i_k)\},$$

HSimplE (IJCAI'20)

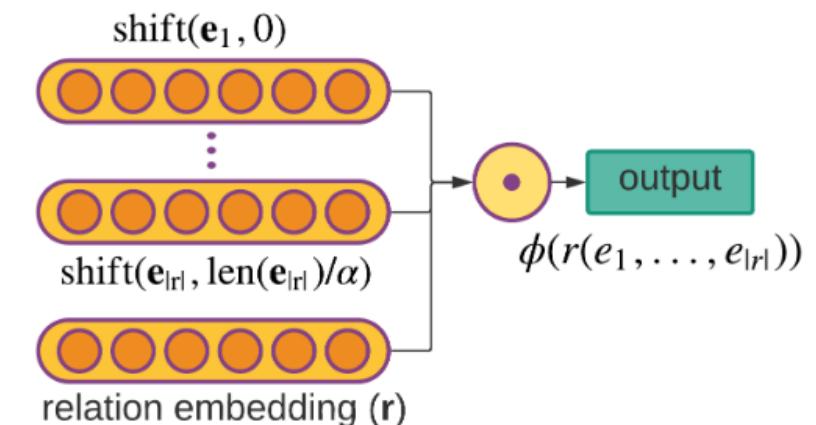
- **SimplE** learns
 - two embedding vectors $\mathbf{e}^{(1)}$ and $\mathbf{e}^{(2)}$ for each entity (one for each position)
 - two embedding vectors $\mathbf{r}^{(1)}$ and $\mathbf{r}^{(2)}$ for each relation (original and inverse)
 - and computes the score of a triple as the element-wise product (i.e. the average of the CP scores of the original triple and the inverse triple)

$$\phi(r(e_1, e_2)) = \odot(\mathbf{r}^{(1)}, \mathbf{e}_1^{(1)}, \mathbf{e}_2^{(2)}) + \odot(\mathbf{r}^{(2)}, \mathbf{e}_2^{(1)}, \mathbf{e}_1^{(2)}).$$

- **HSimplE** extends SimplE to high-order tensor

- Entity embedding is shifted based on the position

$$\phi(r(e_i, e_j, \dots, e_k)) = \odot(\mathbf{r}, \mathbf{e}_i, \text{shift}(\mathbf{e}_j, \text{len}(\mathbf{e}_j)/\alpha), \dots, \text{shift}(\mathbf{e}_k, \text{len}(\mathbf{e}_k) \cdot (\alpha - 1)/\alpha))) \quad (1)$$

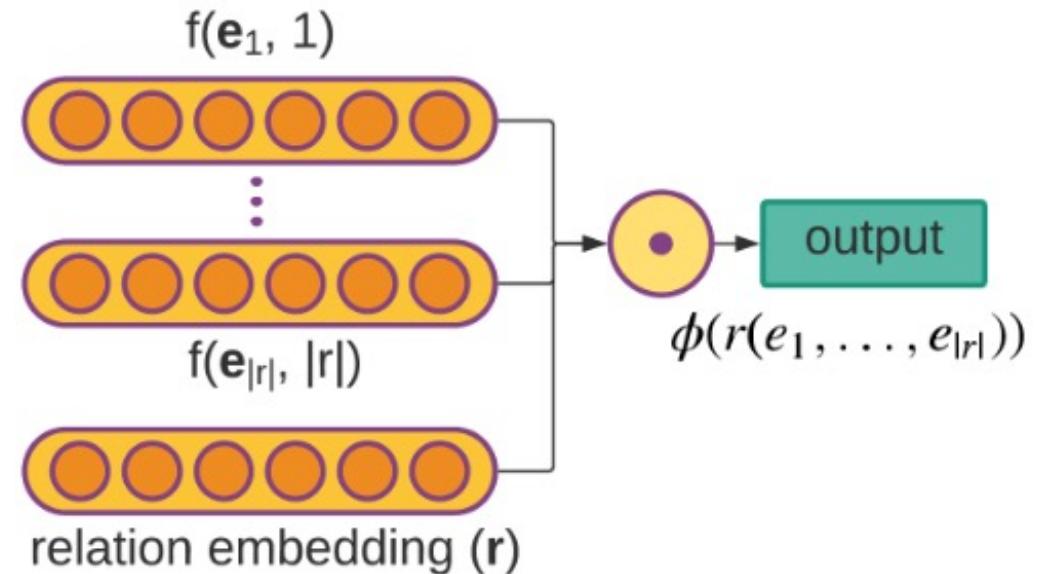
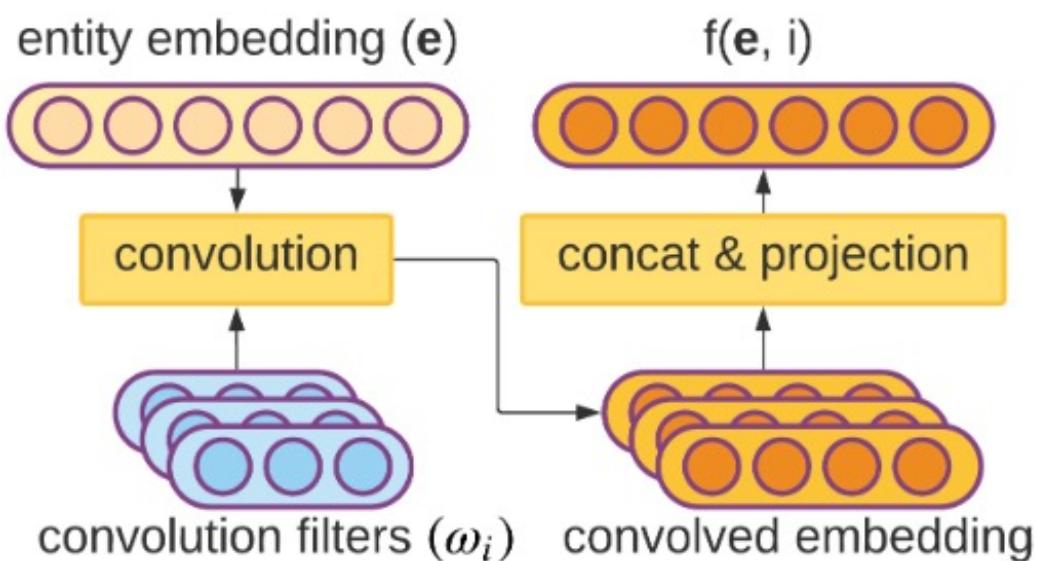


Seyed Mehran Kazemi et al. SimplE Embedding for Link Prediction in Knowledge. NeurIPS 2018.

Bahare Fatemi et al. Knowledge hypergraphs: Prediction beyond binary relations. IJCAI 2020.

HypE (IJCAI'20)

- Entity embedding e is transformed via a **positional convolution filters** $f_w(e, i)$
- These transformed entity embeddings are combined via the relation embedding and produce a final score



$$\phi(r(e_1, \dots, e_{|r|})) = \odot(\mathbf{r}, f(\mathbf{e}_1, 1), \dots, f(\mathbf{e}_{|r|}, |r|))$$

HSimplE & HypE (IJCAI'20)

- HypE and HSimplE are both **full expressive**
 - For any ground truth over entities and relations, there exists a HypE and a HSimplE model with embedding vectors

- Experimental results

Model	Arity			
	2	3	4-5-6	All
r-SimplE	0.478	0.025	0.017	0.168
m-DistMult	0.495	0.648	0.809	0.634
m-CP	0.409	0.563	0.765	0.560
m-TransH [Wen <i>et al.</i> , 2016]	0.411	0.617	0.826	0.596
HSimplE (Ours)	0.497	0.699	0.745	0.645
HypE (Ours)	0.466	0.693	0.858	0.656
# train tuples	36,293	18,846	6,772	61,911
# test tuples	10,758	10,736	3,421	24,915

Table 4: Breakdown performance of Hit@10 across relations with different arities on JF17K dataset along with their statistics.

n-TuckER & GETD (WWW'20)

- n-TuckER generalize **TuckER** to high-order tensor
 - But has exponential model complexity w.r.t the arity
- GETD further integrates **Tensor Ring (TR)** decomposition
 - Factorizing the higher-order core tensor into a seq. of 3-order tensors

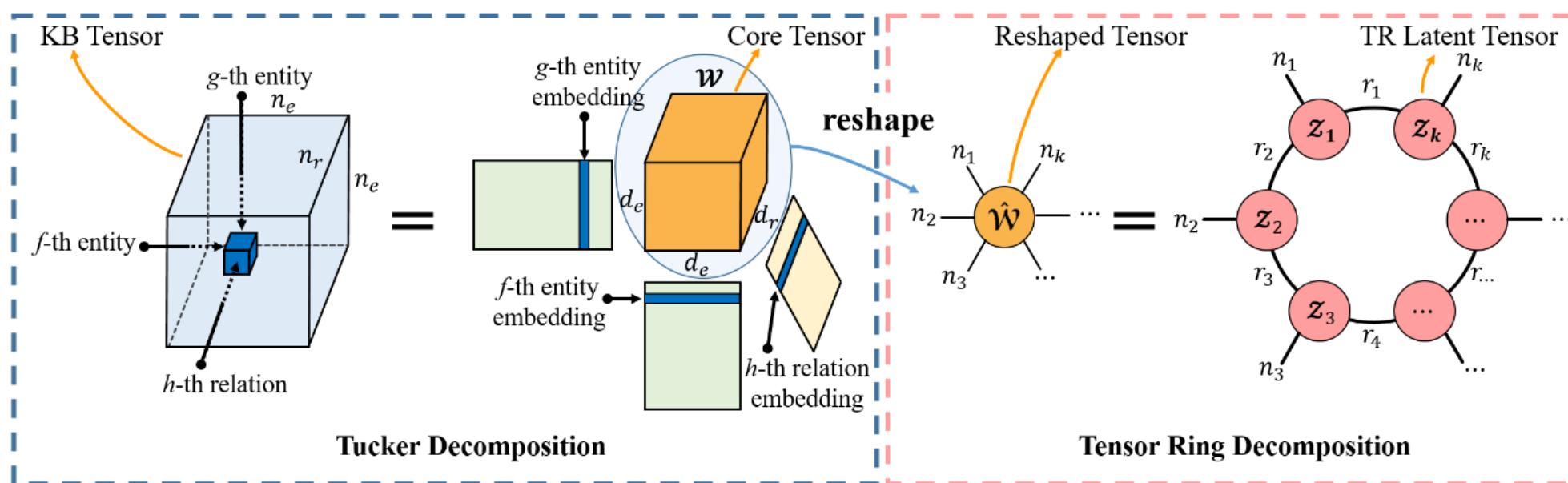


Figure 1: The main framework of GETD model.

Comparison of previous models

- Tensor decomposition models are more expressive

Type	Models	Effectiveness		Efficiency	
		Expressive	Mixed-arity	Time	Space
Translational Models	m-TransH [39]	✗	✓	$O(d)$	$O(n_e d_e + n_r d_r)$
	RAE [48]	✗	✓	$O(d^2)$	$O(n_e d_e + n_r d_r)$
Neural Network Models	NaLP [15]	unknown	✓	$O(d^2)$	$O(n_e d_e + N n_r d_r)$
	HINGE [30]	unknown	✓	$O(d^2)$	$O(n_e d_e + N n_r d_r)$
	NeuInfer [14]	unknown	✓	$O(d^2)$	$O(n_e d_e + N n_r d)$
Tensor Decomposition Models	n-TuckER [25]	✓	✗	$O(d^{n+1})$	$O(n_e d_e + n_r d_r + d_e^n d_r)$
	GETD [25]	✓	✗	$O(d^3)$	$O(n_e d_e + n_r d_r + c d_{\max}^3)$
	S2S	✓	✓	$O(d)$	$O(n_e d_e + n_r d_r)$

Comparison of previous models

- Tensor decompositions cannot handle mixed-arity

Type	Models	Effectiveness		Efficiency	
		Expressive	Mixed-arity	Time	Space
Translational Models	m-TransH [39]	✗	✓	$O(d)$	$O(n_e d_e + n_r d_r)$
	RAE [48]	✗	✓	$O(d^2)$	$O(n_e d_e + n_r d_r)$
Neural Network Models	NaLP [15]	unknown	✓	$O(d^2)$	$O(n_e d_e + N n_r d_r)$
	HINGE [30]	unknown	✓	$O(d^2)$	$O(n_e d_e + N n_r d_r)$
	NeuInfer [14]	unknown	✓	$O(d^2)$	$O(n_e d_e + N n_r d)$
Tensor Decomposition Models	n-TuckER [25]	✓	✗	$O(d^{n+1})$	$O(n_e d_e + n_r d_r + d_e^n d_r)$
	GETD [25]	✓	✗	$O(d^3)$	$O(n_e d_e + n_r d_r + c d_{\max}^3)$
	S2S	✓	✓	$O(d)$	$O(n_e d_e + n_r d_r)$

Comparison of previous models

- Over-parameterization

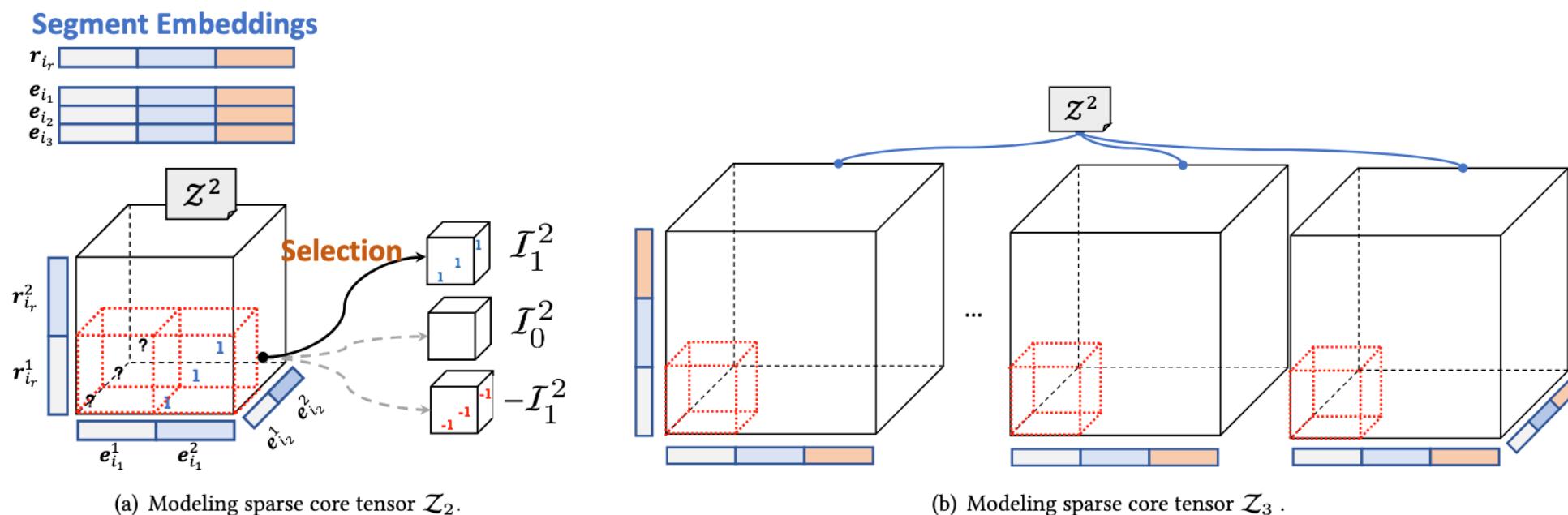
Type	Models	Effectiveness		Efficiency	
		Expressive	Mixed-arity	Time	Space
Translational Models	m-TransH [39]	✗	✓	$O(d)$	$O(n_e d_e + n_r d_r)$
	RAE [48]	✗	✓	$O(d^2)$	$O(n_e d_e + n_r d_r)$
Neural Network Models	NaLP [15]	unknown	✓	$O(d^2)$	$O(n_e d_e + N n_r d_r)$
	HINGE [30]	unknown	✓	$O(d^2)$	$O(n_e d_e + N n_r d_r)$
	NeuInfer [14]	unknown	✓	$O(d^2)$	$O(n_e d_e + N n_r d)$
Tensor Decomposition Models	n-TuckER [25]	✓	✗	$O(d^{n+1})$	$O(n_e d_e + n_r d_r + d_e^n d_r)$
	GETD [25]	✓	✗	$O(d^3)$	$O(n_e d_e + n_r d_r + c d_{\max}^3)$
	S2S	✓	✓	$O(d)$	$O(n_e d_e + n_r d_r)$

Comparison of previous models

- Over-parameterization

Type	Models	Effectiveness			Efficiency	
		Expressive	Mixed-arity	Time	Space	
Translational Models	m-TransH [39]	✗	✓	$O(d)$	$O(n_e d_e + n_r d_r)$	
	RAE [48]	✗	✓	$O(d^2)$	$O(n_e d_e + n_r d_r)$	
Neural Network Models	NaLP [15]	unknown	✓	$O(d^2)$	$O(n_e d_e + N n_r d_r)$	
	HINGE [30]	unknown	✓	$O(d^2)$	$O(n_e d_e + N n_r d_r)$	
	NeuInfer [14]	unknown	✓	$O(d^2)$	$O(n_e d_e + N n_r d)$	
Tensor Decomposition Models	n-TuckER [25]	✓	✗	$O(d^{n+1})$	$O(n_e d_e + n_r d_r + d_e^n d_r)$	
	GETD [25]	✓	✗	$O(d^3)$	$O(n_e d_e + n_r d_r + c d_{\max}^3)$	
	S2S	✓	✓	$O(d)$	$O(n_e d_e + n_r d_r)$	

- Searching to Sparsify (S2S) Core Tensor
 - partially share embeddings across arities and jointly learn embeddings with mixed arity
 - sparsify the dense core tensors using **neural architecture search (NAS)** techniques to avoid over-parameterization



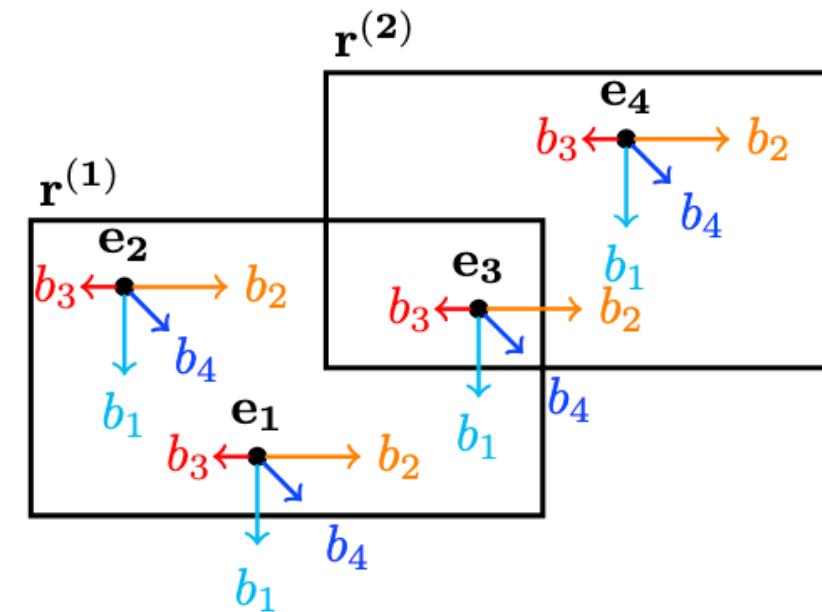
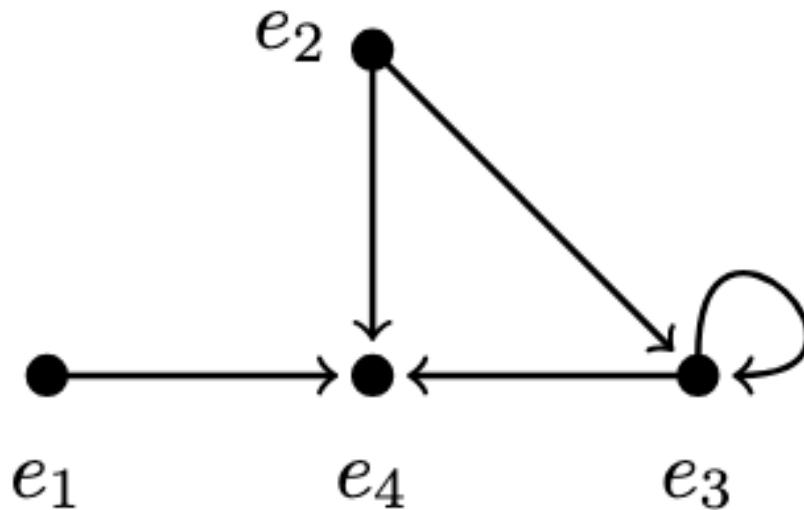
BoxE (NeurIPS'20)

- Full expressiveness only guarantee completeness but not inductive capacity
- Inference pattern is a specification of a logical property that if learned, enables further principled inferences from existing facts
- Existing n-ary embeddings can partially capture inference patterns

Inference pattern	BoxE	TransE	RotatE	DistMult	ComplEx
Symmetry: $r_1(x, y) \Rightarrow r_1(y, x)$	✓/✓	X/X	✓/✓	✓/✓	✓/✓
Anti-symmetry: $r_1(x, y) \Rightarrow \neg r_1(y, x)$	✓/✓	✓/✓	✓/✓	X/X	✓/✓
Inversion: $r_1(x, y) \Leftrightarrow r_2(y, x)$	✓/✓	✓/X	✓/✓	X/X	✓/✓
Composition: $r_1(x, y) \wedge r_2(y, z) \Rightarrow r_3(x, z)$	X/X	✓/X	✓/X	X/X	X/X
Hierarchy: $r_1(x, y) \Rightarrow r_2(x, y)$	✓/✓	X/X	X/X	✓/X	✓/X
Intersection: $r_1(x, y) \wedge r_2(x, y) \Rightarrow r_3(x, y)$	✓/✓	✓/X	✓/X	X/X	X/X
Mutual exclusion: $r_1(x, y) \wedge r_2(x, y) \Rightarrow \perp$	✓/✓	✓/✓	✓/✓	✓/X	✓/X

BoxE (NeurIPS'20)

- An n-ary relation is encoded by N boxes r^1, r^2, \dots, r^N
- Each entity e has an embedding e_i , and defines a bump b_j on other entities
- Entity representation is **dynamic**, the final entity embeddings depend on the relative facts



BoxE (NeurIPS'20)

- KBC Performance

Table 3: KBC results on JF17K and FB-AUTO.

Model	JF17K		FB-AUTO	
	MRR	H@10	MRR	H@10
m-TransH	.446	.614	.728	.728
m-DistMult	.460	.635	.784	.845
m-CP	.392	.560	.752	.837
HypE	.492	.650	.804	.856
HSimplE	.472	.649	.798	.855
BoxE(u)	.553	.711	.837	.895
BoxE(a)	.560	.722	.844	.898

- Rule injection



Figure 3: The SportsNELL ontology.

Table 4: Rule injection experiment results on the SportsNELL full and filtered evaluation sets.

Model	Full Set			Filtered Set		
	MR	MRR	H@10	MR	MRR	H@10
BoxE	17.4	.577	.780	19.1	.713	.824
BoxE+RI	1.74	.979	.997	5.11	.954	.984

Agenda

Part 1. Introduction (Bo & Mojtaba, 9:00 - 9:30)

- 1.1 Tutorial Overview (Bo)
- 1.2 Introduction to KG Embeddings
- 1.3 Motivation to go beyond triple-based KGs

Part 2. Temporal KG Embeddings (Mojtaba, 9:30 – 10:30)

- 2.1 Background
- 2.2 Interpolation-based TKGE
- 2.3 Extrapolation-based TKGE
- *Coffee Break (10:30 – 11:00)*

Part 3. N-ary & Hyper-relational embeddings (Bo, 11:00 – 12:30)

- 3.1 Introduction to hyper-relational & N-Ary KGs
- 3.2 N-ary relational embeddings
- 3.3 Hyper-relational embeddings
- *Lunch Break (12:30 – 14:00)*

Part 4. KG embeddings with literal and text (Daniel, 14:00 – 15:00)

- 4.1 Incorporating entity descriptions
- 4.2 Incorporate numeric literals as features
- 4.3 Incorporating relation descriptions

Part 5. Advanced Topic (Michael, 15:00-15:30 & 16:00 – 16:30)

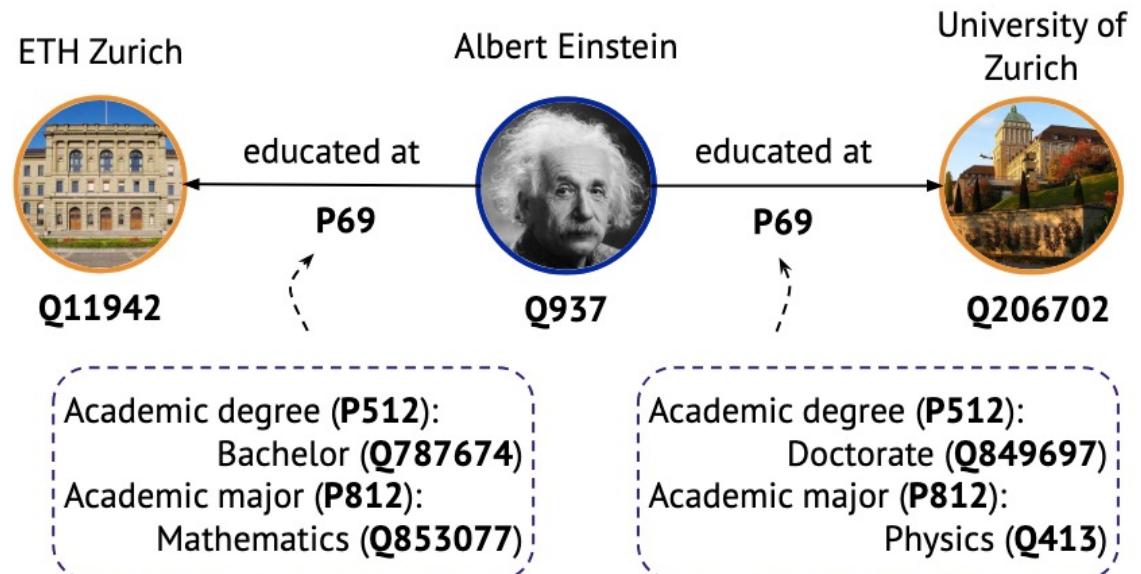
- 5.1 Complex query embeddings (Daniel)
- *Coffee Break (15:30 – 16:00)*
- 5.2 Neural Graph Databases
- 5.3 Inductive learning settings
- 5.3 Incorporating ontology, LLM, etc.

Part 6. Conclusion & Future Works (Michael, 16:30 – 17:00)

- 6.1 Conclusion
- 6.2 Future Works

Hyper-Relational Know Graphs

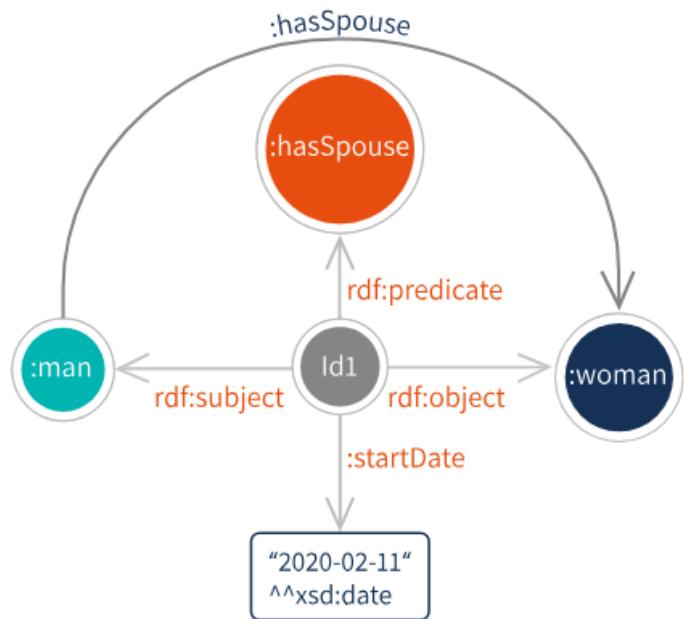
- A hyper-relational fact consists of
 - A **primal triple** (h, r, t)
 - A set of **qualifiers** (k, v)



- **Primal triple** depicts the main statement (subject, verb, object)
- **Qualifiers** contextualize/specialize the meaning of the statement
e.g., attributive, adverbial

Standard Reification

- Transforming hyper-relational facts into triple facts
 - Create a **ID entity** for each fact, and a **new entity** for each predicate
 - Create **three triples** that connect the **subject**, **object** and **predicate**, respectively
 - Create a **triple** for each of the qualifier entity with the corresponding key relation



```
:man :hasSpouse :woman .  
:id1 rdf:type rdf:Statement ;  
  rdf:subject :man ;  
  rdf:predicat :hasSpouse ;  
  rdf:object :woman ;  
  :startDate "2020-02-11"^^xsd:date .
```

- **Disadvantages:** create heterogeneity/inefficiency/new entities in the test stage

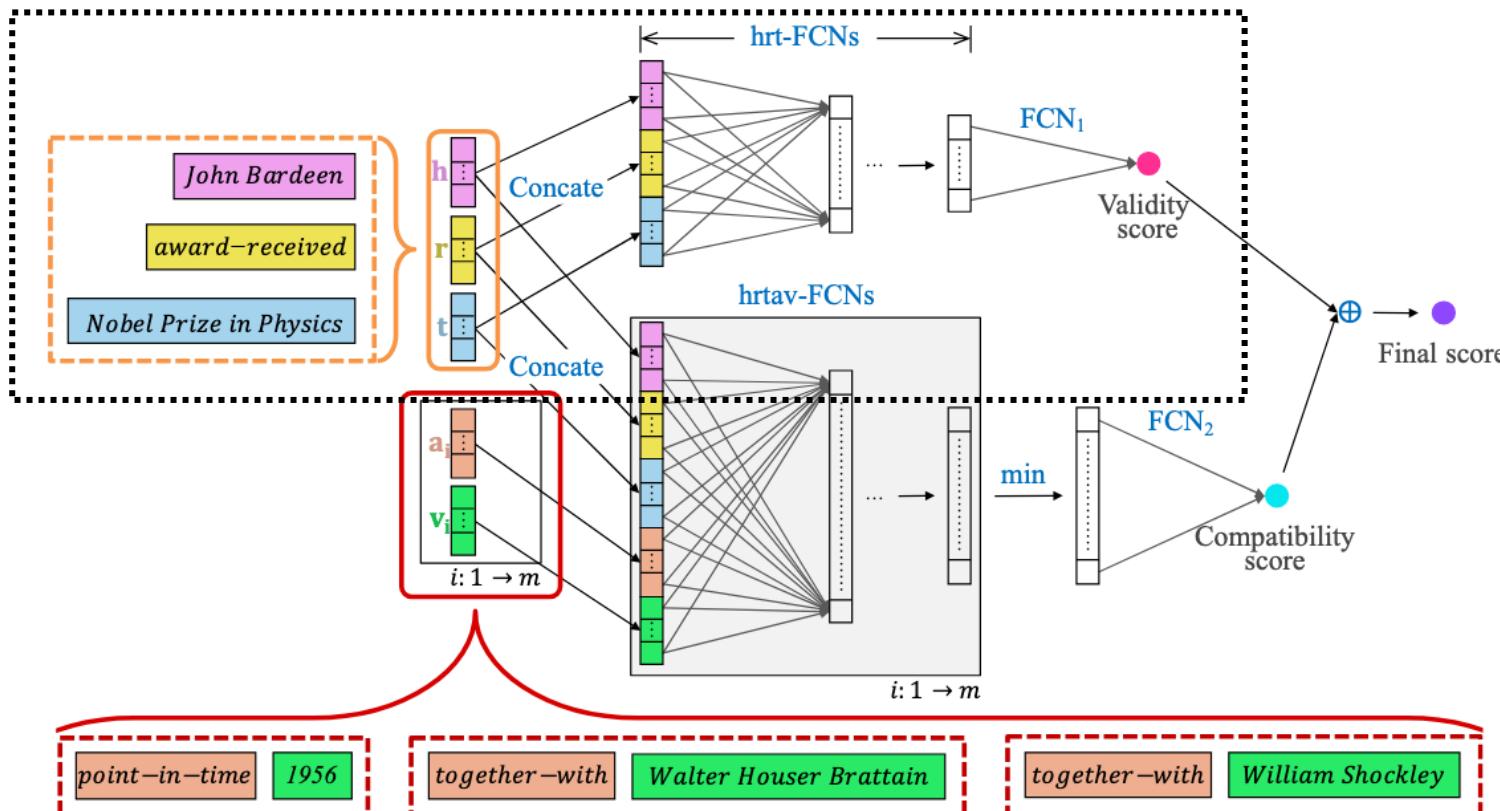
Ref: <https://www.ontotext.com/knowledgehub/fundamentals/what-is-rdf-star/>

Hyper-Relational KG embeddings

- Neural network models
 - **NeurInfer** (ACL'20) – shallow MLP layers
 - **HINGE** (WWW'20) – 2D Convolutional layers
 - **StarE** (EMNLP'20) – GNN layers + Transformer
 - **Hy-Transformer** (AAAI'22-DLG) – Transformer
 - **GRAN** (ACL-IJCNLP'21) – GNN layers
- Translational models
 - **ShrinkE** (ACL'23)

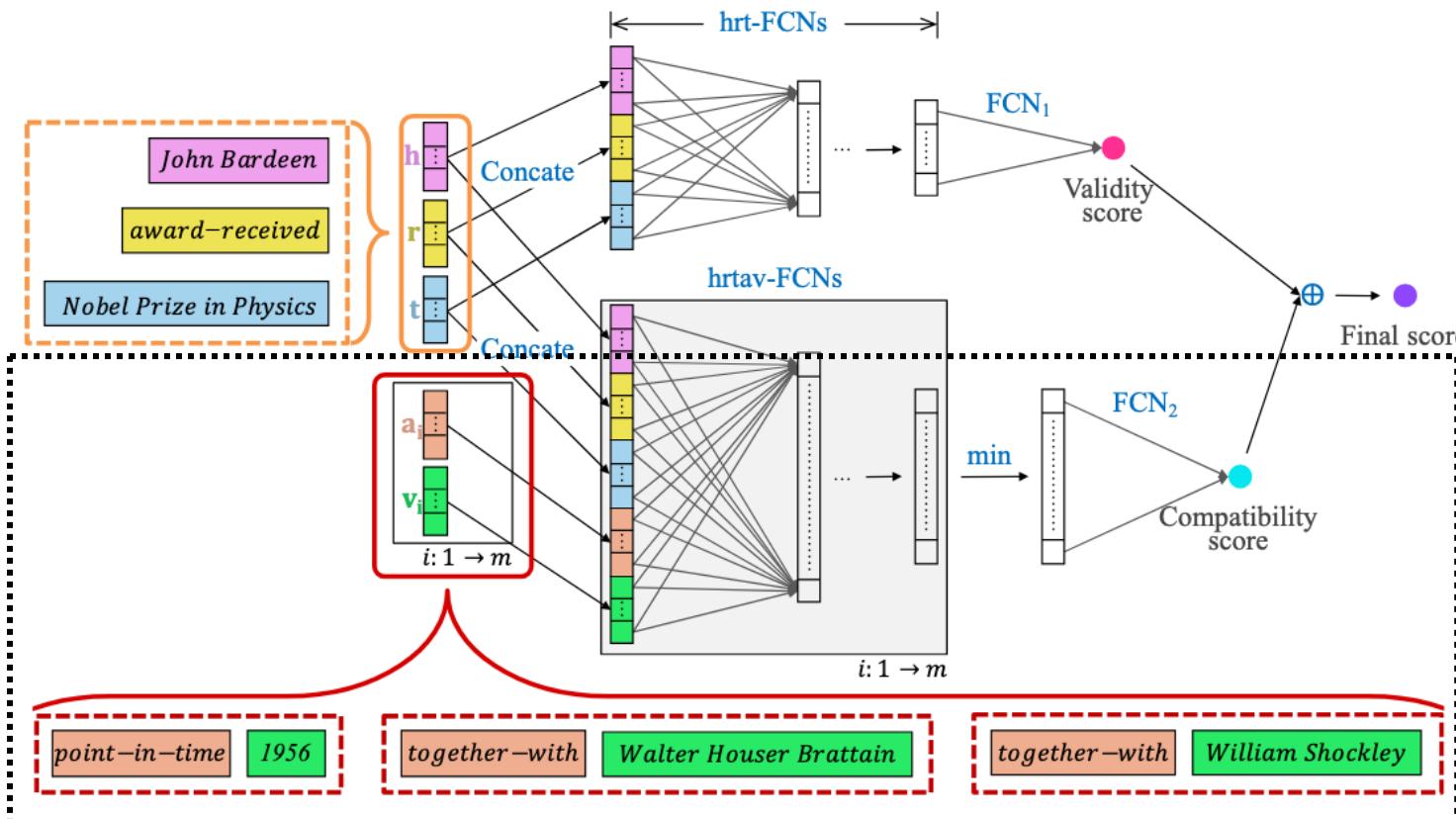
NeurInfer (ACL'20)

- The plausibility of a fact jointly depends on
 - the **validity** of the primal triple (h, r, t)
 - and its **compatibility** with each qualifier (a, v)



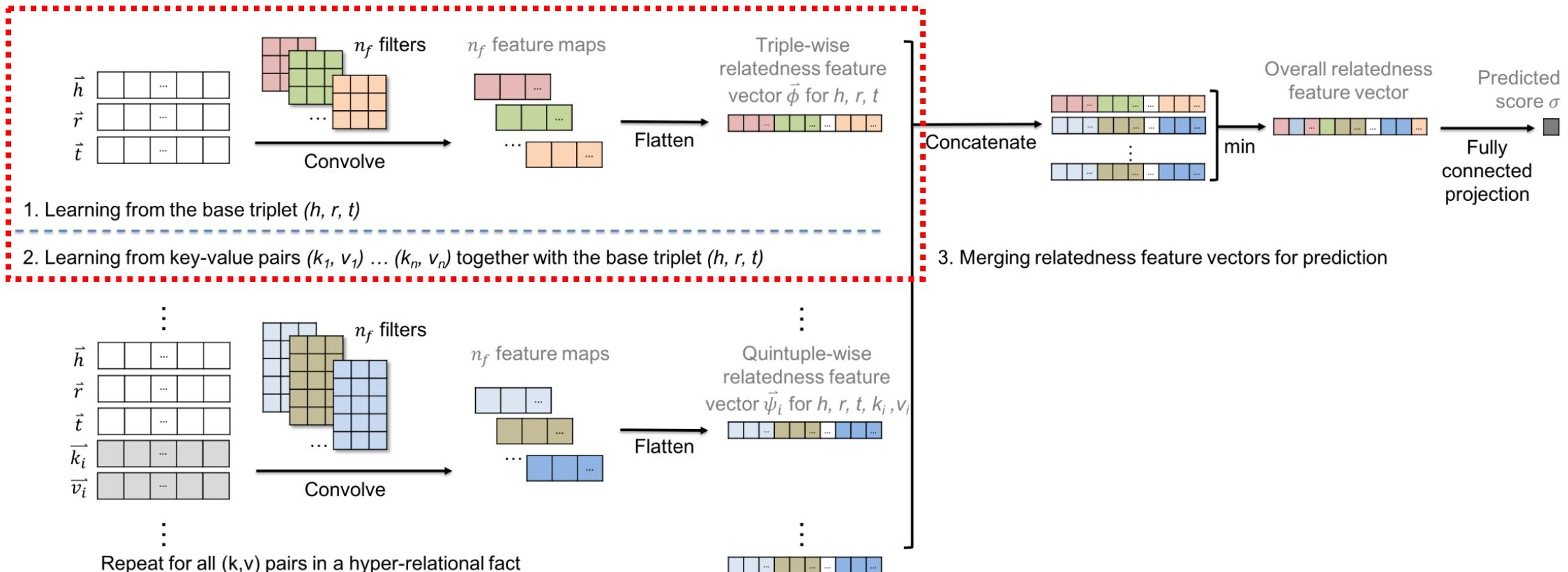
NeurInfer (ACL'20)

- The plausibility of a fact jointly depends on
 - the **validity** of the primal triple (h, r, t)
 - and its **compatibility** with each qualifier (a, v)



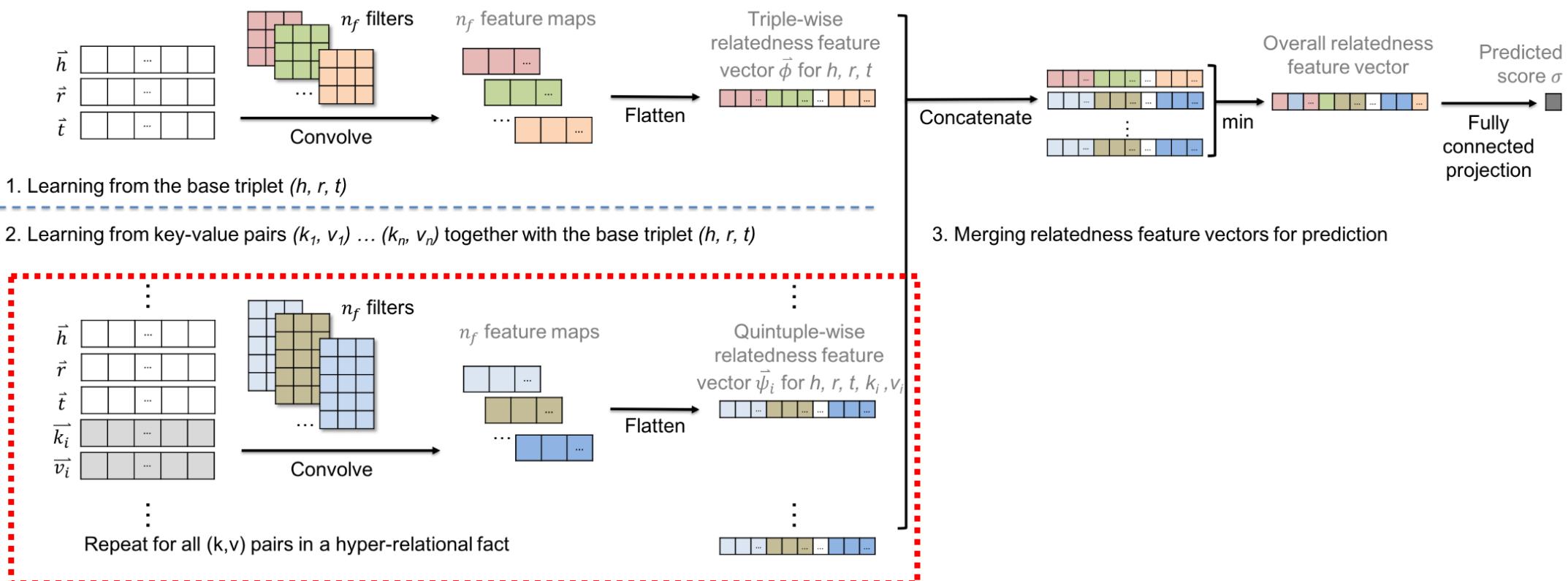
HINGE (WWW'20)

- Primal triplet modeling
 - Generating **relatedness vector** of (h, r, t) via convolutional neural layers



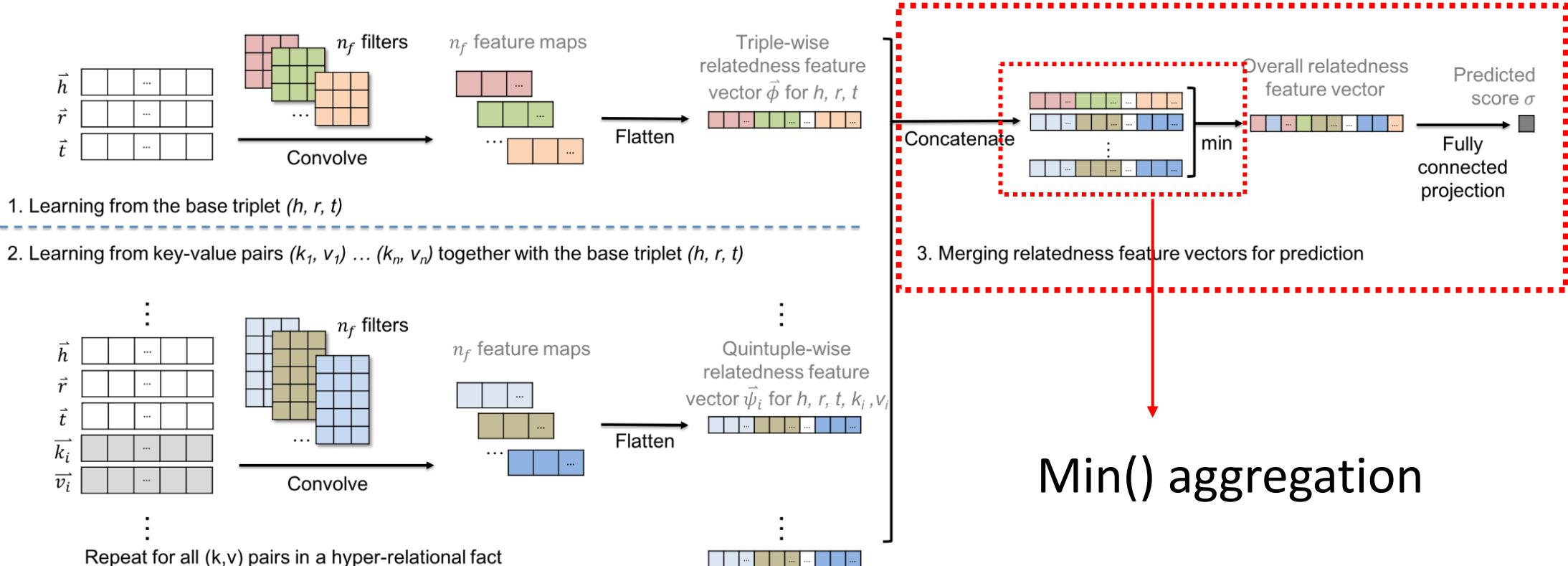
HINGE (WWW'20)

- Qualifier modeling
 - Generating **relatedness vector** of (h, r, t, k, v)



HINGE (WWW'20)

- Measuring the plausibility of a fact by combining
 - the **relatedness** of the (h, r, t) in the primal triple
 - and the **relatedness** between the triple and each qualifier

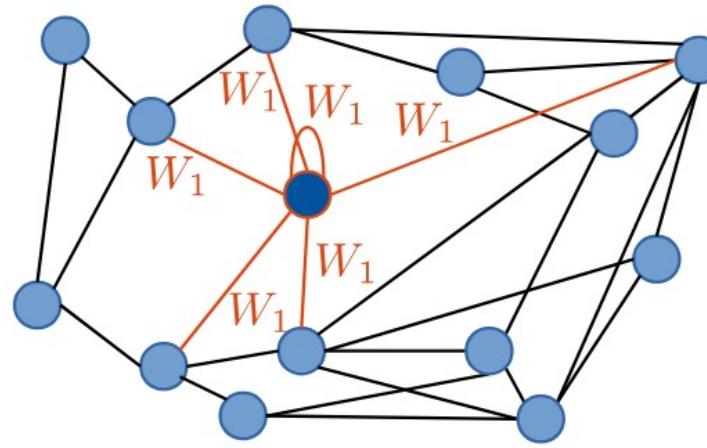


- Link prediction

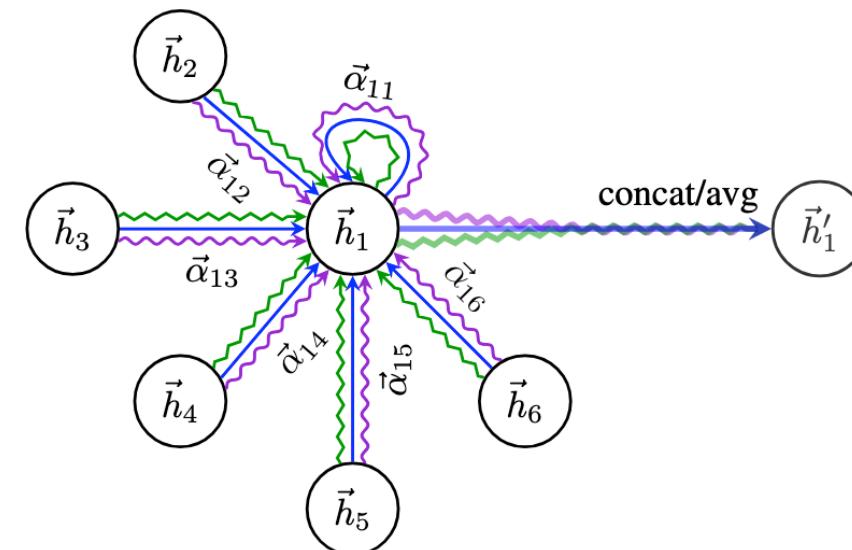
Dataset Transformation Setting	Method	WikiPeople						JF17K					
		Head/Tail Prediction			Relation Prediction			Head/Tail Prediction			Relation Prediction		
		MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1
Reification	TransE	0.3207	0.5977	0.1224	0.3253	0.3850	0.2747	0.2285	0.3806	0.1503	0.8793	0.9187	0.8559
	TransH	0.3244	0.6011	0.1242	0.3160	0.3873	0.2694	0.2302	0.3815	0.1538	0.8774	0.9218	0.8506
	TransR	0.3225	0.6002	0.1225	0.2396	0.2968	0.1817	0.2838	0.4722	0.1914	0.8751	0.9157	0.8510
	TransD	0.2123	0.5253	0.0195	0.5293	0.8611	0.3821	0.0950	0.2121	0.0366	0.5562	0.6610	0.5010
	Rescal	0.2751	0.4815	0.1430	0.7684	0.8816	0.7053	0.1354	0.2608	0.0759	0.6958	0.7620	0.6556
	DistMult	0.2276	0.4867	0.0519	0.5902	0.6611	0.5422	0.1523	0.2888	0.0875	0.1135	0.3251	0.0332
	ComplEx	0.2365	0.4795	0.0614	0.5375	0.5882	0.5039	0.1325	0.2552	0.0760	0.1311	0.2451	0.0717
	Analogy	0.2478	0.4901	0.0718	0.5838	0.6277	0.5562	0.1329	0.2594	0.0742	0.1548	0.2983	0.0852
	ConvE	0.4657	0.6434	0.3559	N/A			0.3469	0.5410	0.2500	N/A		
Original	m-TransH	0.0633	0.3006	0.0633	N/A			0.2060	0.4627	0.2060	N/A		
	RAE	0.0586	0.3064	0.0586	N/A			0.2153	0.4668	0.2153	N/A		
	NALP	0.4084	0.5461	0.3311	0.4818	0.8516	0.3198	0.2209	0.3310	0.1650	0.6391	0.8215	0.5472
	NALP-Fix	0.4202	0.5564	0.3429	0.8200	0.9757	0.7197	0.2446	0.3585	0.1852	0.7469	0.8921	0.6665
	HINGE	0.4763	0.5846	0.4154	0.9500	0.9977	0.9159	0.4489	0.6236	0.3611	0.9367	0.9894	0.9014

Graph Neural Networks

- Graph Convolutional Networks (GCN)
 - Feature transformation
 - Neighborhood aggregation (average/sum/max)
- Graph Attention Networks (GAT)
 - Aggregation by multi-head attention mechanism



$$\mathbf{h}_v^{(k)} = f \left(\sum_{u \in \mathcal{N}(v)} \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)} \right)$$

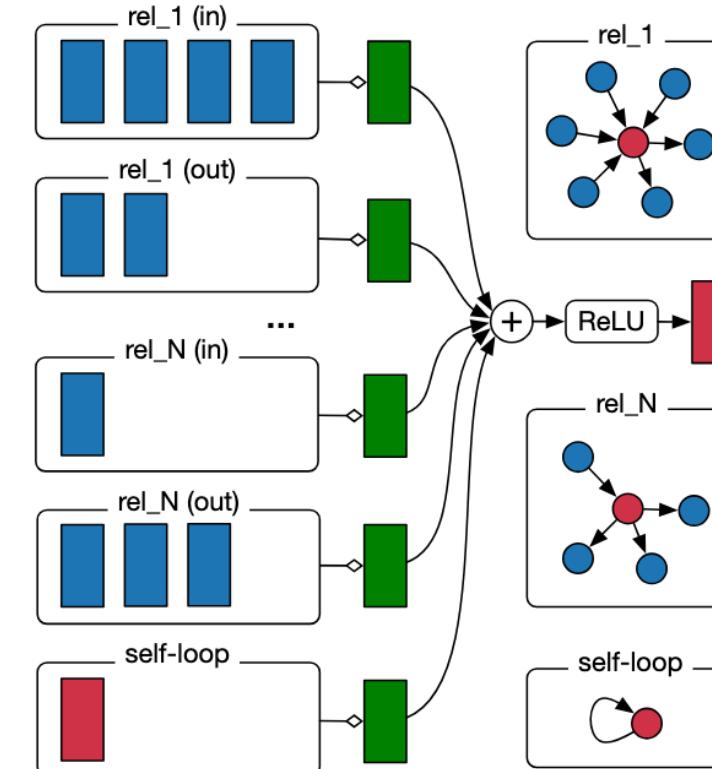


$$\vec{h}'_i = \parallel_{k=1}^K \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right)$$

Relational Graph Convolutional Networks (RGCN)

- RGCN extends GCNs to multi-relational graphs
 - By applying relation-specific linear transformation

$$h_i^{(l+1)} = \sigma \left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)} \right)$$

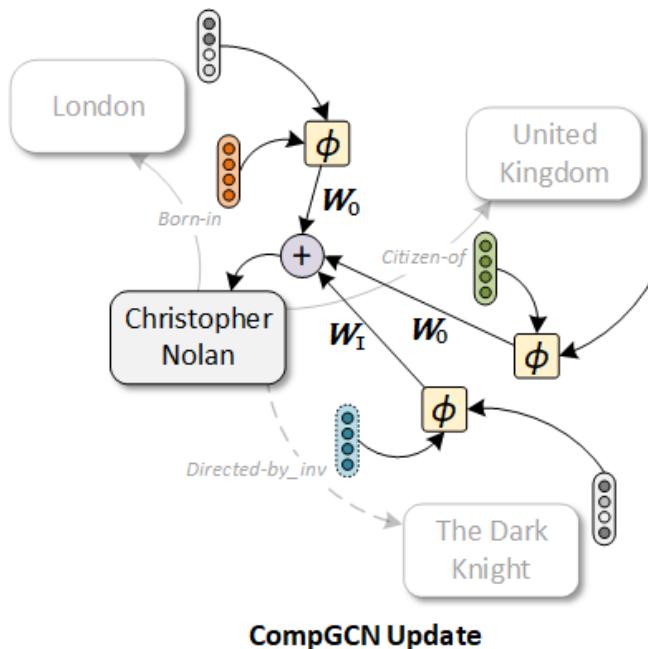
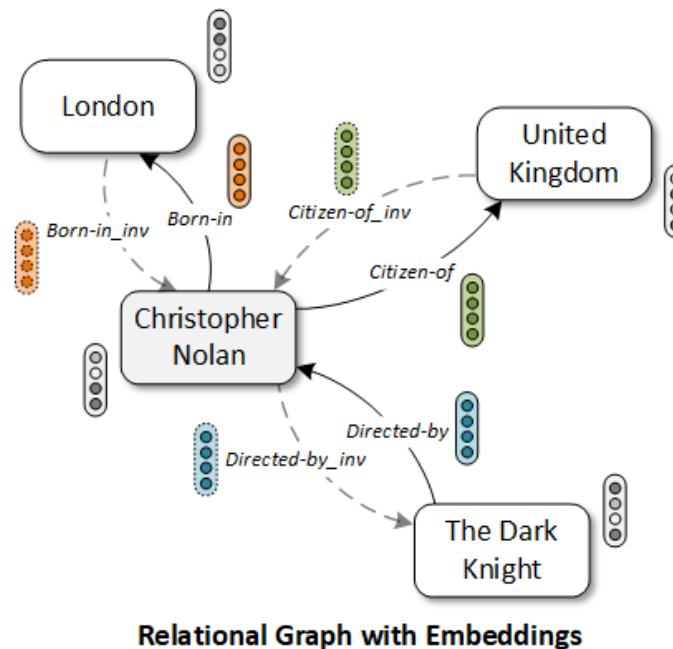


- Over-parameterization and only learn node representations

Michael Schlichtkrull et al. Modeling Relational Data with Graph Convolutional Networks. ESWC 2018.

Composition-based RGCN (CompGCN)

- CompGCN overcomes the limitations of RGCN via
 - Jointly embeds both nodes and relations
 - Composition operation over each edge (e.g., subtraction or multiplication)
 - Direction-specific **shared parameter** for incoming/outgoing/self-looping relations



$$\mathbf{h}_v^{(k)} = f \left(\sum_{(u,r) \in \mathcal{N}(v)} \mathbf{W}_{\lambda(r)}^{(k)} \phi(\mathbf{h}_u^{(k-1)}, \mathbf{h}_r^{(k-1)}) \right) \quad (4)$$

Composition-based RGCN (CompGCN)

- CompGCN is a comprehensive design and is more parameter efficient

Methods	Node Embeddings	Directions	Relations	Relation Embeddings	Number of Parameters
GCN Kipf & Welling (2016)	✓				$\mathcal{O}(Kd^2)$
Directed-GCN Marcheggiani & Titov (2017)	✓	✓			$\mathcal{O}(Kd^2)$
Weighted-GCN Shang et al. (2019)	✓		✓		$\mathcal{O}(Kd^2 + K \mathcal{R})$
Relational-GCN Schlichtkrull et al. (2017)	✓	✓	✓		$\mathcal{O}(\mathcal{B}Kd^2 + \mathcal{B}K \mathcal{R})$
COMPGCN (Proposed Method)	✓	✓	✓	✓	$\mathcal{O}(Kd^2 + \mathcal{B}d + \mathcal{B} \mathcal{R})$

Methods	$\mathbf{W}_{\lambda(r)}^k$	$\phi(\mathbf{h}_u^k, \mathbf{h}_r^k)$
Kipf-GCN (Kipf & Welling, 2016)	\mathbf{W}^k	\mathbf{h}_u^k
Relational-GCN (Schlichtkrull et al., 2017)	\mathbf{W}_r^k	\mathbf{h}_u^k
Directed-GCN (Marcheggiani & Titov, 2017)	$\mathbf{W}_{\text{dir}(r)}^k$	\mathbf{h}_u^k
Weighted-GCN (Shang et al., 2019)	\mathbf{W}^k	$\alpha_r^k \mathbf{h}_u^k$

StarE (EMNLP'20)

- Modified CompGNN for Hyper-Relational Knowledge Graphs

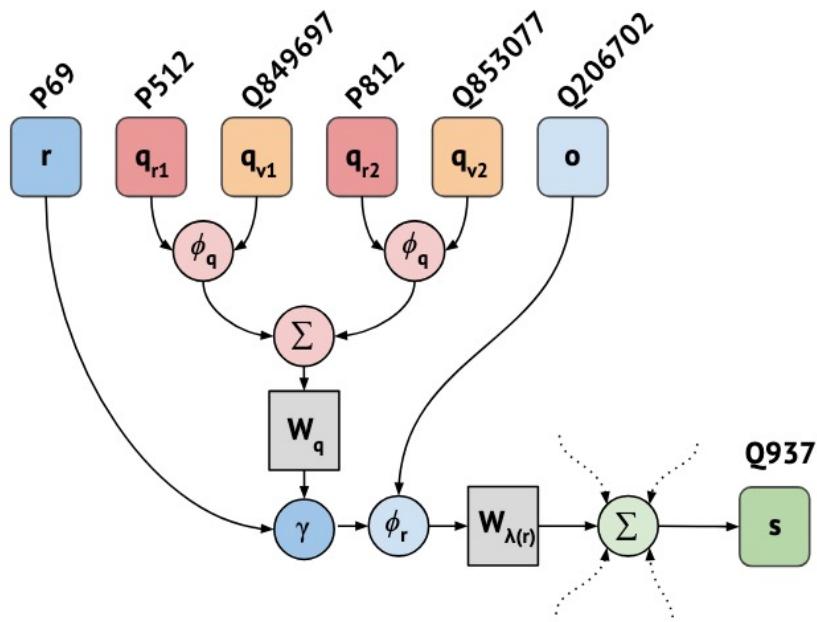


Figure 2: The mechanism in which STARE encodes a hyper-relational fact from Fig. 1.B. Qualifier pairs are passed through a composition function ϕ_q , summed and transformed by W_q . The resulting vector is then merged via γ , and ϕ_r with the relation and object vector, respectively. Finally, node $Q937$ aggregates messages from this and other hyper-relational edges.

- Aggregation

$$\mathbf{h}_v = f \left(\sum_{(u,r) \in \mathcal{N}(v)} \mathbf{W}_{\lambda(r)} \phi_r(\mathbf{h}_u, \gamma(\mathbf{h}_r, \mathbf{h}_q)_{vu}) \right) \quad (5)$$

$\gamma(\cdot)$ combines the main **relation representation** with its **qualifiers representation**

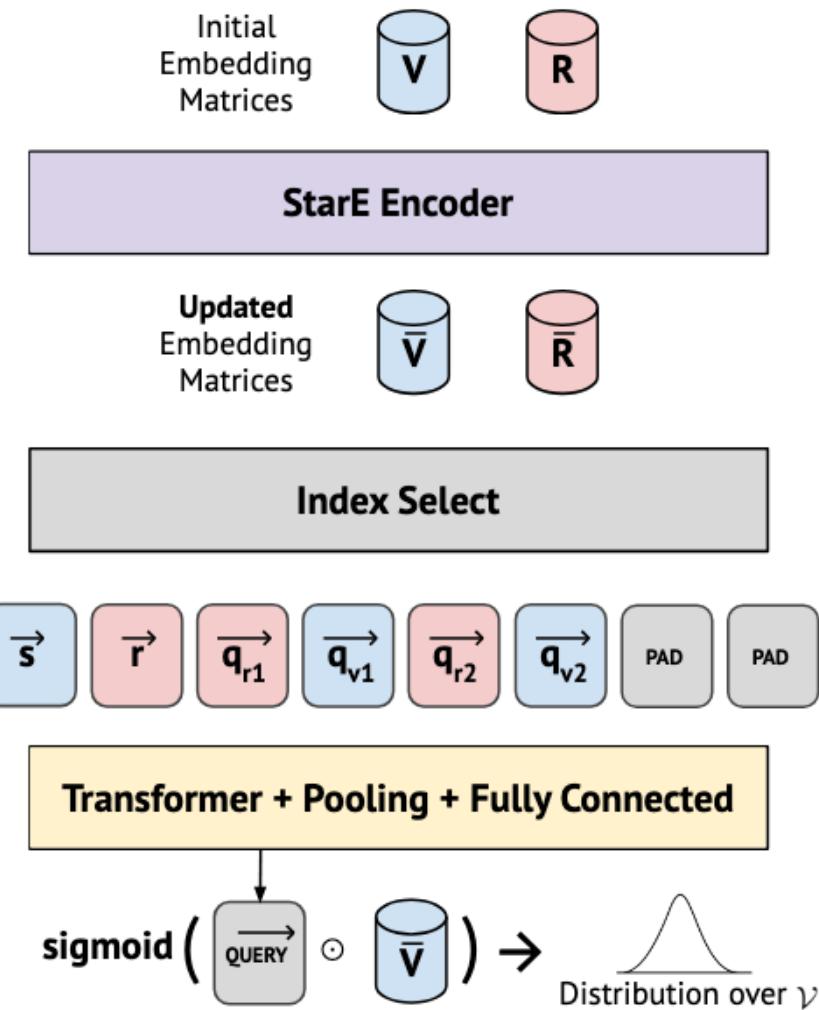
$$\gamma(\mathbf{h}_r, \mathbf{h}_q) = \alpha \odot \mathbf{h}_r + (1 - \alpha) \odot \mathbf{h}_q$$

$\phi(\cdot)$ combines the embeddings of qualifier relation and qualifier entity

$$\mathbf{h}_q = \mathbf{W}_q \sum_{(qr,qv) \in Q_{jrvu}} \phi_q(\mathbf{h}_{qr}, \mathbf{h}_{qv})$$

StarE (EMNLP'20)

- StarE for link prediction
 - Modified CompGCN as encoder
 - Transformer as decoder



StarE (EMNLP'20)

- StarE for link prediction

Table 2: Link prediction on WikiPeople and JF17K. Results of m-TransH, RAE, NaLP-Fix and HINGE are taken from (Rosso et al., 2020). Best results among hyper-relational models are in **bold**.

Exp #	Method	WikiPeople				JF17K			
		MRR	H@1	H@5	H@10	MRR	H@1	H@5	H@10
1	m-TransH	0.063	0.063	-	0.300	0.206	0.206	-	0.463
1	RAE	0.059	0.059	-	0.306	0.215	0.215	-	0.469
1	NaLP-Fix	0.420	0.343	-	0.556	0.245	0.185	-	0.358
1	HINGE	0.476	0.415	-	0.585	0.449	0.361	-	0.624
1,4	Transformer (H)	0.469	0.403	0.538	0.586	0.512	0.434	0.593	0.665
1,4	STARE (H) + Transformer(H)	0.491	0.398	0.592	0.648	0.574	0.496	0.658	0.725
4	Transformer (T)	0.474	0.419	0.532	0.575	0.537	0.473	0.606	0.663
4	STARE (T) + Transformer (T)	0.493	0.400	0.592	0.648	0.562	0.493	0.637	0.702

StarE (EMNLP'20)

- StarE with different decoder
 - StarE + ConvE: ConvE-like decoder but expanded for statements with qualifiers
 - StarE + ConvKB: ConvKB-like decoder adjusted for statements with qualifiers
 - **StarE + MskTrf** denotes a Transformer decoder with an explicit [MASK] token at the object position of each query

Table 6: Effect of different decoders on the link prediction task over WD50K, and its variations.

Dataset →	WD50K			WD50K (33)			WD50K (66)			WD50K (100)		
	MRR	H@1	H@10									
Method ↓												
STARE + Trf	0.349	0.271	0.496	0.331	0.268	0.451	0.481	0.420	0.594	0.654	0.588	0.777
STARE + ConvE	0.341	0.260	0.496	0.323	0.254	0.456	0.460	0.392	0.590	0.627	0.550	0.772
STARE + ConvKB	0.323	0.241	0.479	0.316	0.247	0.448	0.448	0.377	0.584	0.621	0.544	0.763
STARE + MskTrf	0.341	0.262	0.489	0.324	0.260	0.446	0.479	0.417	0.595	0.649	0.579	0.774

Hy-Transformer (AAAI'22-DLG)

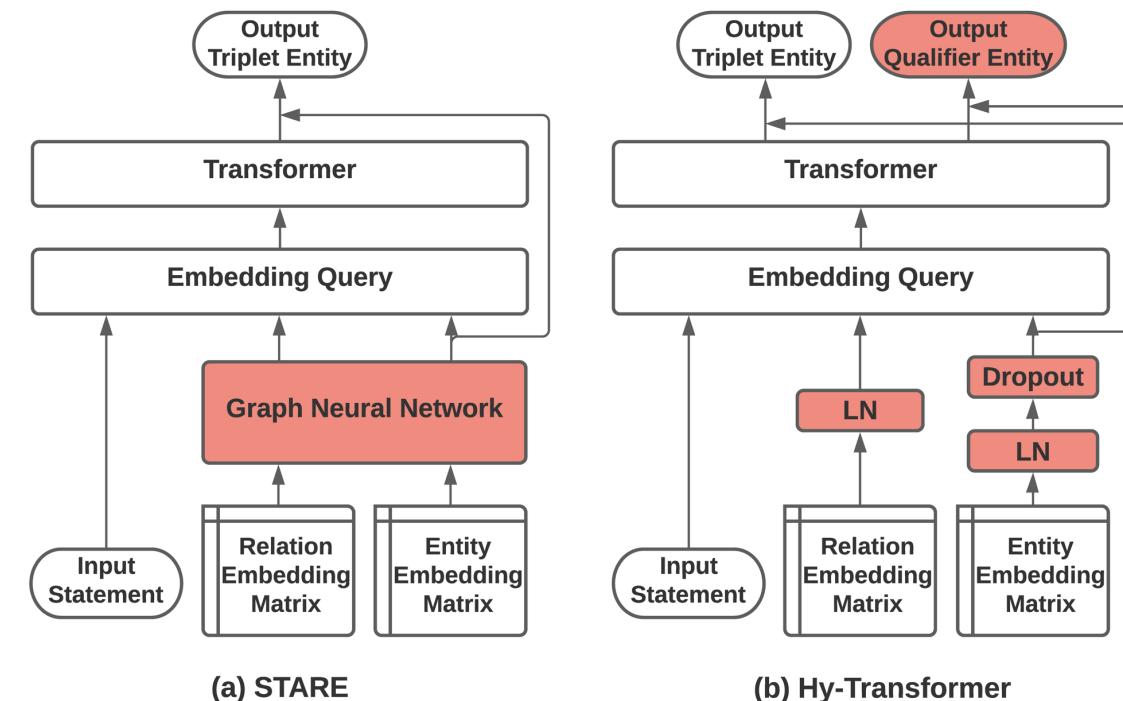
- Hy-Transformer improves StarE by
 - Replacing GNN with light-weight entity/relation embedding
 - Adding an qualifier-oriented auxiliary training task

- use **layer normalization (LN)** and **dropout** for entity and relation embedding

$$\hat{E} = \text{Dropout}(\text{LN}(E)) \in \mathbb{R}^{(N+1) \times d}$$

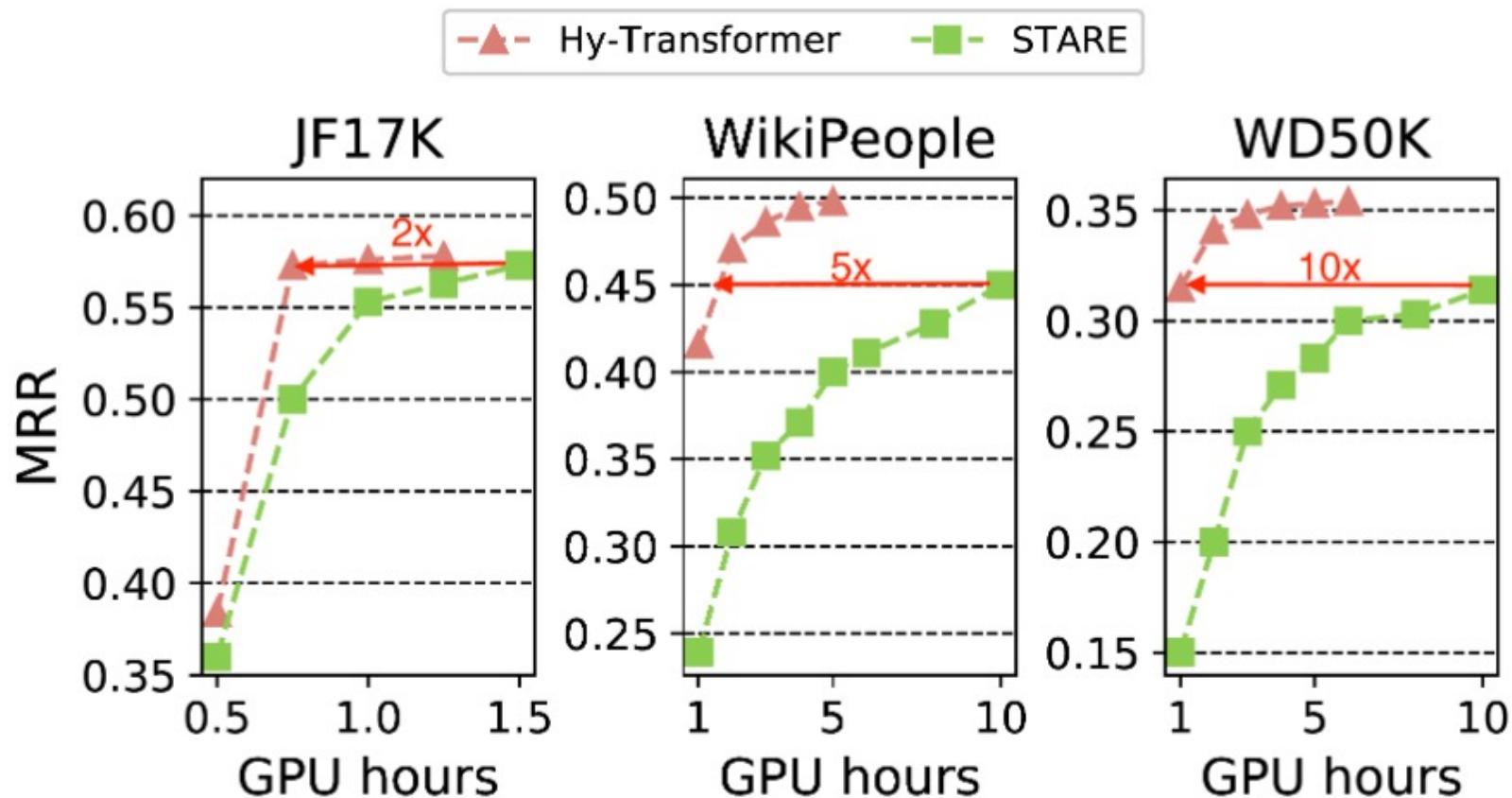
$$\hat{R} = \text{LN}(R) \in \mathbb{R}^{M \times d}$$

- Auxiliary training task
 - Predicting missing qualifier entities



Hy-Transformer (AAAI'22-DLG)

- Performance and efficiency



GRAN (ACL-IJCNLP'21 Findings)

- Graph representation

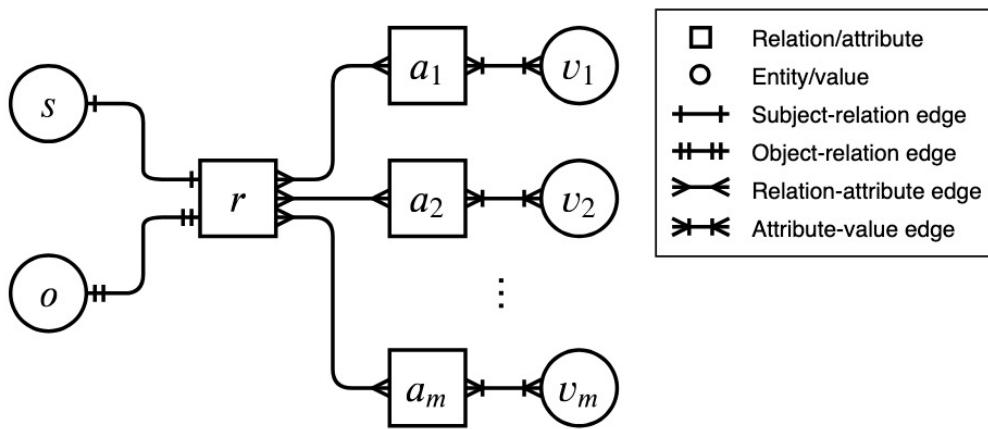


Figure 1: An n-ary fact as a heterogeneous graph, with relations/attributes and entities/values as vertices, and four types of edges designed between the vertices.

- Graph Learning

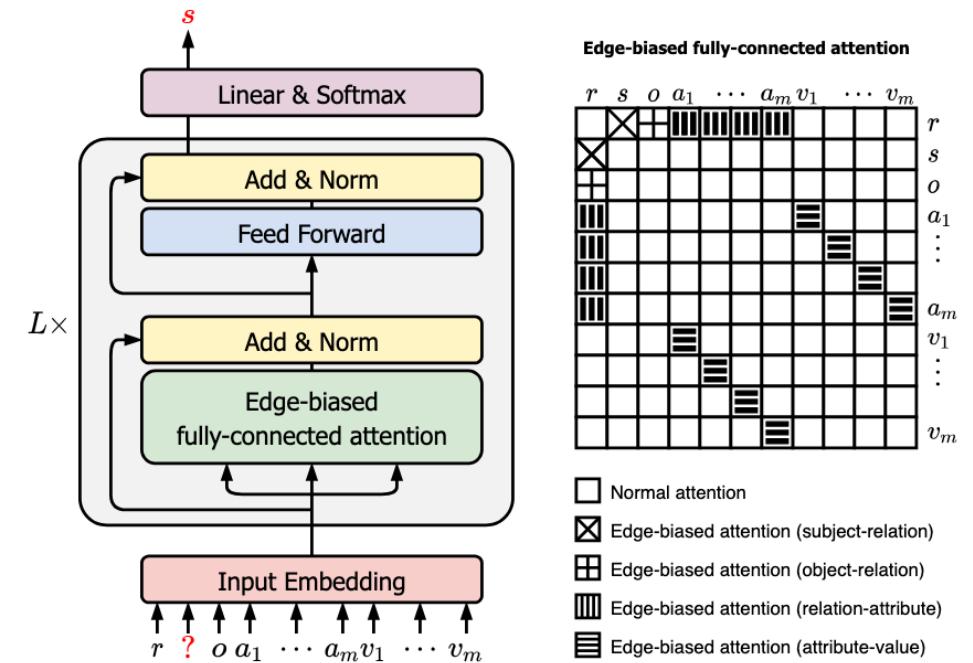


Figure 2: Overview of the graph learning process, with edge-biased fully-connected attention illustrated.

GRAN (ACL-IJCNLP'21 Findings)

- **GRAN-hete** encode both graph structure (whether there is an edge) and heterogeneity (which type the edge is)
- **GRAN-homo** retains graph structure but ignores heterogeneity
- **GRAN-complete** considers neither graph structure nor heterogeneity

	JF17K All Entities			JF17K Subject/Object			WikiPeople All Entities			WikiPeople Subject/Object		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
RAE	.310	.219	.504	.215	.215	.467	.172	.102	.320	.059	.059	.306
NaLP	.366	.290	.516	.221	.165	.331	.338	.272	.466	.408	.331	.546
HINGE	—	—	—	.449	.361	.624	—	—	—	.476	.415	.585
NeuInfer	.517	.436	.675	—	—	—	.350	.282	.467	—	—	—
RAM	.539	.463	.690	—	—	—	.380	.279	.539	—	—	—
STARE	—	—	—	.574	.496	.725	—	—	—	.491	.398	.648
Hy-Transformer	—	—	—	.582	.501	.742	—	—	—	.501	.426	.634
GRAN-hete	.656	.582	.799	.617	.539	.770	.479	.410	.604	.503	.438	.620
GRAN-homo	.650	.576	.795	.611	.533	.767	.465	.386	.602	.487	.410	.618
GRAN-complete	.622	.546	.774	.591	.510	.753	.460	.381	.601	.489	.413	.617

Hyper-Relational KG embeddings

- Neural network models
 - **NeurInfer** (ACL'20) – shallow MLP layers
 - **HINGE** (WWW'20) – 2D Convolutional layers
 - **StarE** (EMNLP'20) – GNN layers + Transformer
 - **Hy-Transformer** (AAAI'22-DLG) – Transformer
 - **GRAN** (ACL-IJCNLP'21) – GNN layers
- Translational models
 - **ShrinkE** (ACL'23)

ShrinkE (ACL'23)

- **Monotonicity:** by adding any qualifier to a query, the answer set may only shrink but never enlarge

Example: $((Einstein, educated_at, ?), \{degree:PhD, \})$

- **Qualifier-level logical patterns**

- **Qualifier implication:** if q_1 implies q_2 , then by attaching q_2 to any fact that contains q_1 , the truth of the fact does not change

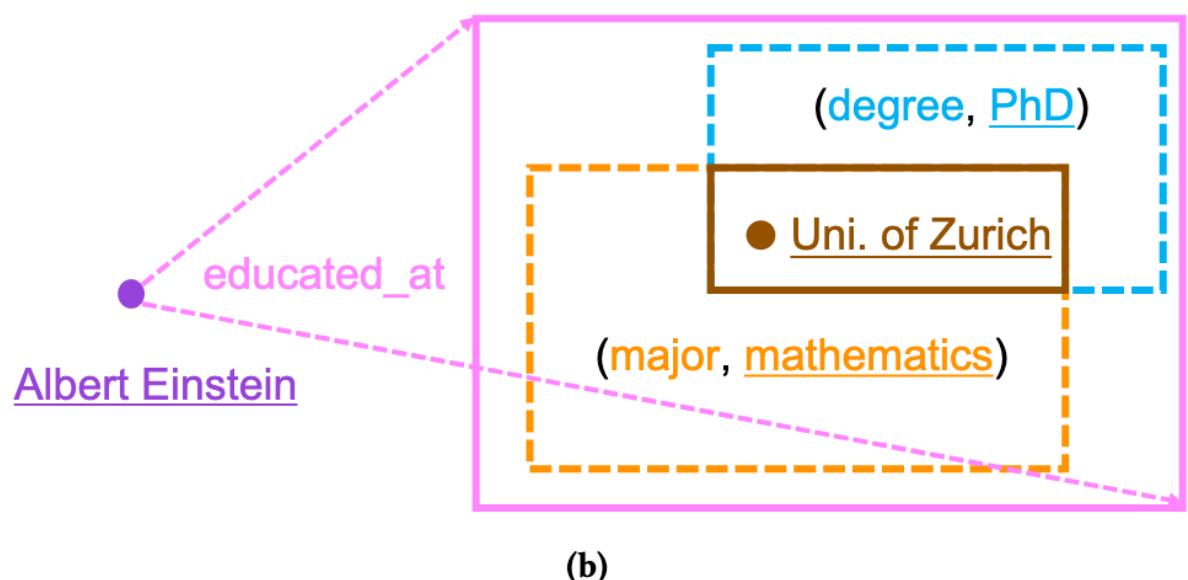
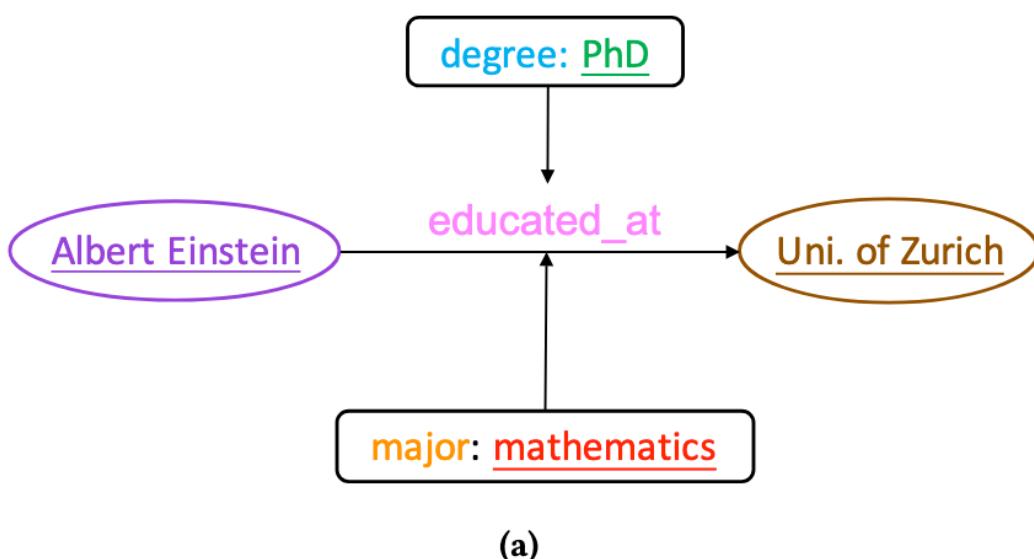
Example: $(located_at, Washington) \rightarrow (located_at, USA)$

- **Qualifier exclusion:** if q_1 is incompatible with q_2 , then by attaching (q_1, q_2) to the same fact, the fact must be false

Example: $(start_in, 2020) \oplus (end_in, 2018)$

ShrinkE (ACL'23)

- Modeling the **primal triple** as a spatial **spanning** (from a point to a box)
- Modeling each **qualifier** as a spatial (**monotonically**) **shrinking** of the box
- Qualifier **implication** and **exclusion** are geometrically modeled as **box containment** and **disjointedness**



ShrinkE (ACL'23)

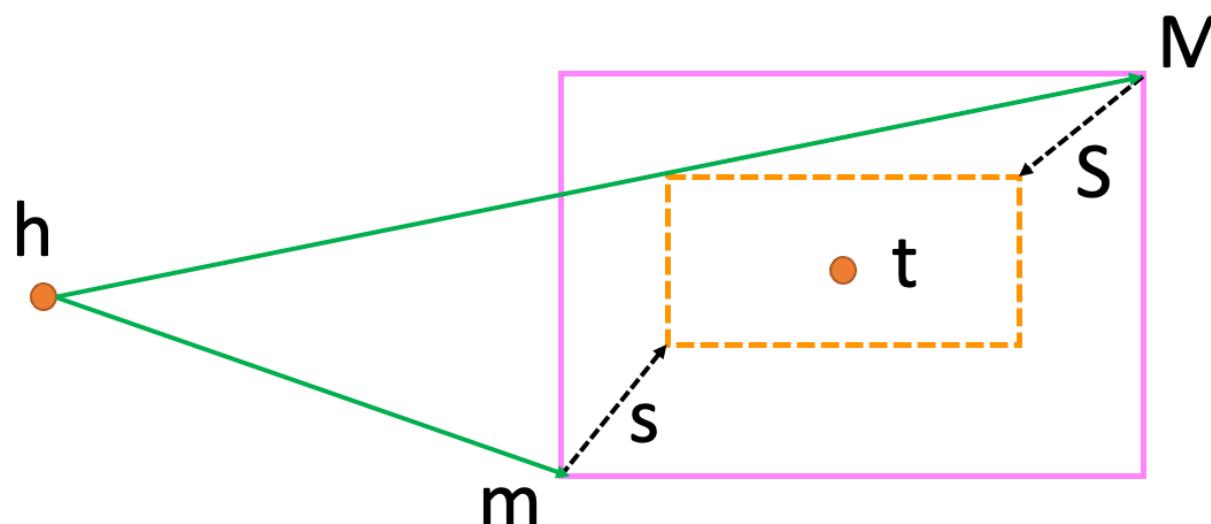
- Box embedding

$$\text{Box}^d(\mathbf{m}, \mathbf{M}) = \{x \in \mathbb{R}^d \mid \mathbf{m}_i \leq x_i \leq \mathbf{M}_i \quad i = 1, \dots, d\}$$

- Box shrinking is a box-to-box transform that monotonically shrinks the size

$$\mathcal{K}_{r,a,v}(\text{Box}^n(\mathbf{m}, \mathbf{M})) = \text{Box}^n(\mathbf{m} + \mathbf{s}_{r,a,v}, \mathbf{M} - \mathbf{S}_{r,a,v})$$

where $\mathbf{s}_{r,a,v}, \mathbf{S}_{r,a,v} \geq 0$.



ShrinkE (ACL'23)

Method	WikiPeople (2.6)			JF17K (45.9)			WD50K (13.6)		
	MRR	H@1	H@10	MRR	H@ 1	H@ 10	MRR	H@ 1	H@ 10
m-TransH	0.063	0.063	0.300	0.206	0.206	0.463	—	—	—
RAE	0.059	0.059	0.306	0.215	0.215	0.469	—	—	—
NaLP-Fix	0.420	0.343	0.556	0.245	0.185	0.358	0.177	0.131	0.264
NeuInfer	0.350	0.282	0.467	0.451	0.373	0.604	—	—	—
HINGE	0.476	0.415	0.585	0.449	0.361	0.624	0.243	0.176	0.377
Transformer	0.469	0.403	0.586	0.512	0.434	0.665	0.264	0.194	0.401
BoxE	0.395	0.293	0.503	0.560	0.472	0.722	—	—	—
StarE	0.491	0.398	0.648	0.574	0.496	0.725	0.349	<u>0.271</u>	0.496
ShrinkE	<u>0.485</u>	0.431	<u>0.601</u>	0.589	0.506	0.749	<u>0.345</u>	0.275	<u>0.482</u>

Summary

- N-Ary KG embeddings
 - Translational embeddings
 - Tensor decomposition embeddings
- Hyper-relational KG embeddings
 - Neural network models
 - Functional embeddings

Other interesting approaches

- N-Ary relational KGs
 - Dictionary based modeling, **NaLP** (WWW'19), **RAM** (WWW'21)
 - Modeling relational algebra (e.g., selection), **ReAIE** (JMLR'23)
 - Incorporating type information, **tNaLP** (TKDE'21)
 - Recursive hyperedges, **G-MPNN** (NeurIPS'20)
 - ...
- Hyper-relational KGs
 - Hyper-relational KG with numeric qualifiers, **HyNT** (KDD'23)
 - Temporal Hype-relational KG, **HypeTKG** (ArXiv'23)
 - Link prediction via masked language model, **Hyper-ELC** (ACL'22-DeeLIO)
 - Inductive learning settings (afternoon session)
 - Complex queries (afternoon session)
 - ...



Reasoning Beyond Triples: Recent Advances in Knowledge Graph Embeddings



Bo Xiong ¹



Mojtaba Nayyeri ¹



Daniel Daza ^{2,3}



Michael Cochez ²

¹ University of Stuttgart ² Vrije Universiteit Amsterdam ³ University of Amsterdam

Saturday 21 October, 09:00 - 17:30 (GMT+1) @ University of Birmingham - Birmingham, UK

Agenda

Part 1. Introduction (Bo & Mojtaba, 9:00 - 9:30)

- 1.1 Tutorial Overview (Bo)
- 1.2 Introduction to KG Embeddings
- 1.3 Motivation to go beyond triple-based KGs

Part 2. Temporal KG Embeddings (Mojtaba, 9:30 – 10:30)

- 2.1 Functional models
 - 2.2 Factorization models
 - 2.3 Neural embedding models
- Coffee Break (10:30 – 11:00)*

Part 3. Hyper-relational & N-ary KG embeddings (Bo, 11:00 – 12:30)

- 3.1 Introduction to hyper-relational & N-Ary KGs
 - 3.2 Functional embedding models
 - 3.2 Graph neural networks models
- Lunch Break (12:30 – 13:30)*

Part 4. KG embeddings with literal and text (Daniel, 14:00 – 15:00)

- 4.1 Incorporating entity descriptions
- 4.2 Incorporate numeric literals as features

Part 5. Advanced Topic (Michael, 15:00-15:30 & 16:00 – 16:30)

- 5.1 Complex query embeddings (Daniel)
- Coffee Break (15:30 – 16:00)*
- 5.2 Neural Graph Databases
- 5.3 Inductive learning settings
- 5.4 Incorporating ontologies and LLMs

Part 6. Conclusion & Future Works (Michael, 16:30 – 17:00)

- 6.1 Conclusion
- 6.2 Future Works

Part 4. KG embeddings with literal and text (Daniel, 14:00 – 15:00)

4.1 Incorporating entity descriptions

- Early approaches: DKRL
- KGloVe with literals
- BLP / BioBLP

4.2 Incorporate numeric literals as features

- Incorporating other modalities, BioBLP
- multi-modal GNN (Xander)
- Other multi-modal approaches, e.g.
<https://aclanthology.org/D18-1359/>

4.3 Incorporating relation descriptions

- Delft question answering [link](#) ?
- KG-BERT, SimKGC, Statik

10 minutes Q/A

(slides mehwish)

<https://arxiv.org/abs/1910.12507>

Part 5. Advanced Topic (Michael, 15:00-15:30 & 16:00 – 16:30)

5.1a Complex query embeddings (Daniel)

- Neural query encoders
 - GQE, MPQE, StarQE, GNN-QE
- Decomposition approaches
 - CQD, CQD-A
 - LitCQD

Coffee Break (15:30 – 16:00)

16:00-16:30

5.1b NGDB? 10 minutes or so

- evaluation + framework we are designing
- query expressivity + gaps

5.2 Inductive learning settings

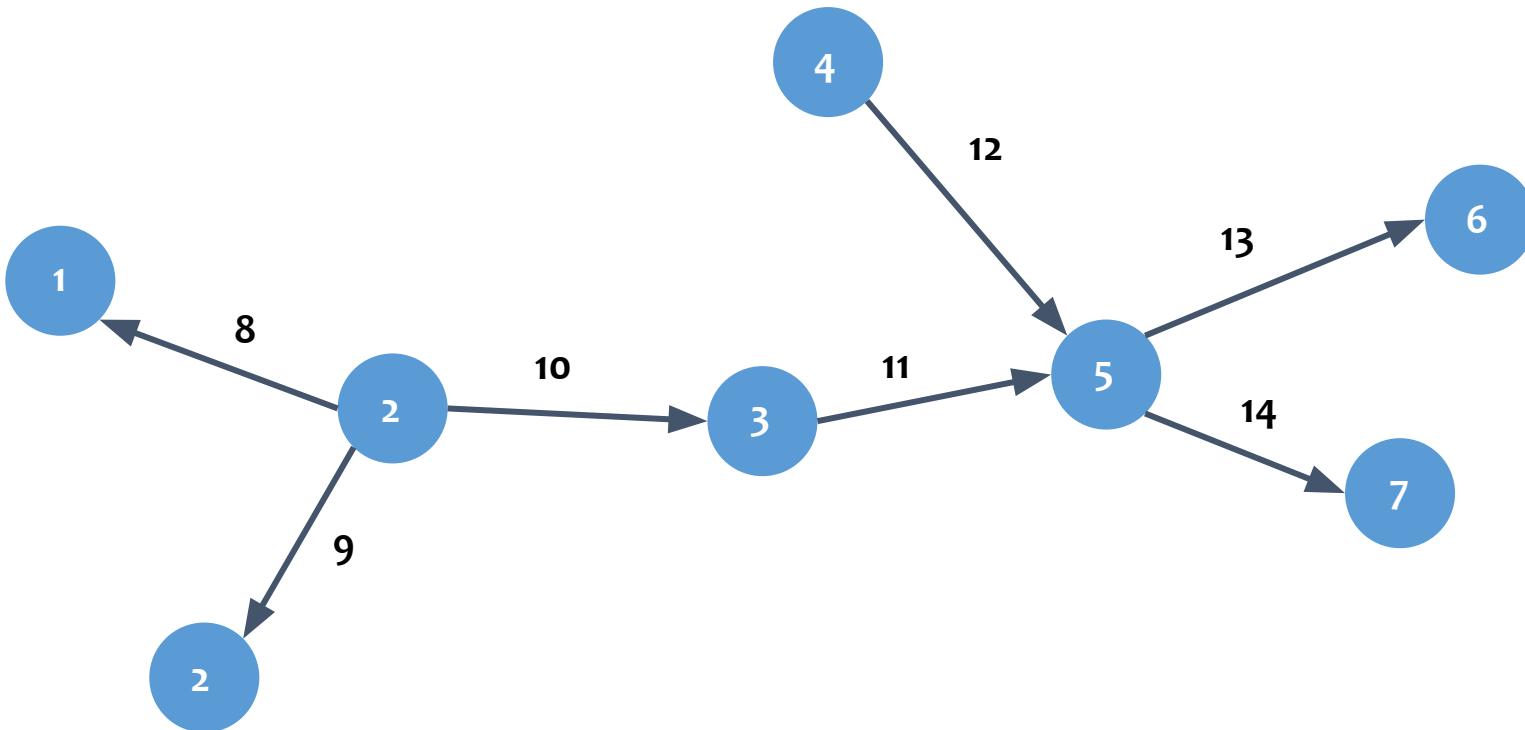
- NodePiece + extensions ??? Not beyond triples.
- Inductive query answering
- mention BLP/BioBLP/ relation description

5.3 Incorporating ontology, LLM, etc.

- Faithful embeddings for EL++
- Paper Pascal hitzler
- Paper D'Amato <https://openreview.net/pdf?id=C9g-pwemYxd>



Part 4. KG embeddings with attribute data



Introducing the Knowledge Graph: things, not strings

“The Knowledge Graph enables you to search for things, people or places that Google knows about—**landmarks, celebrities, cities, sports teams, buildings, geographical features, movies, celestial objects, works of art and more**—and instantly get information that’s relevant to your query.

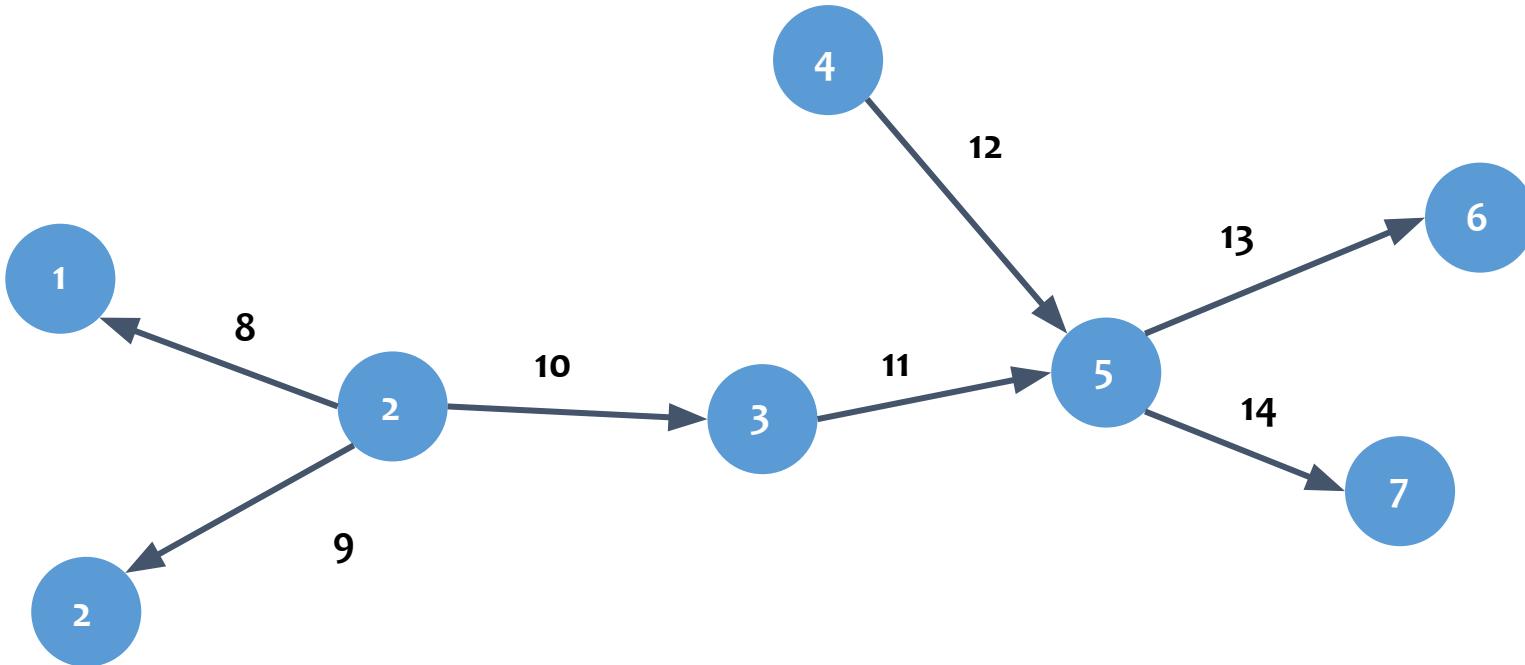
<https://blog.google/products/search/introducing-knowledge-graph-things-not/>

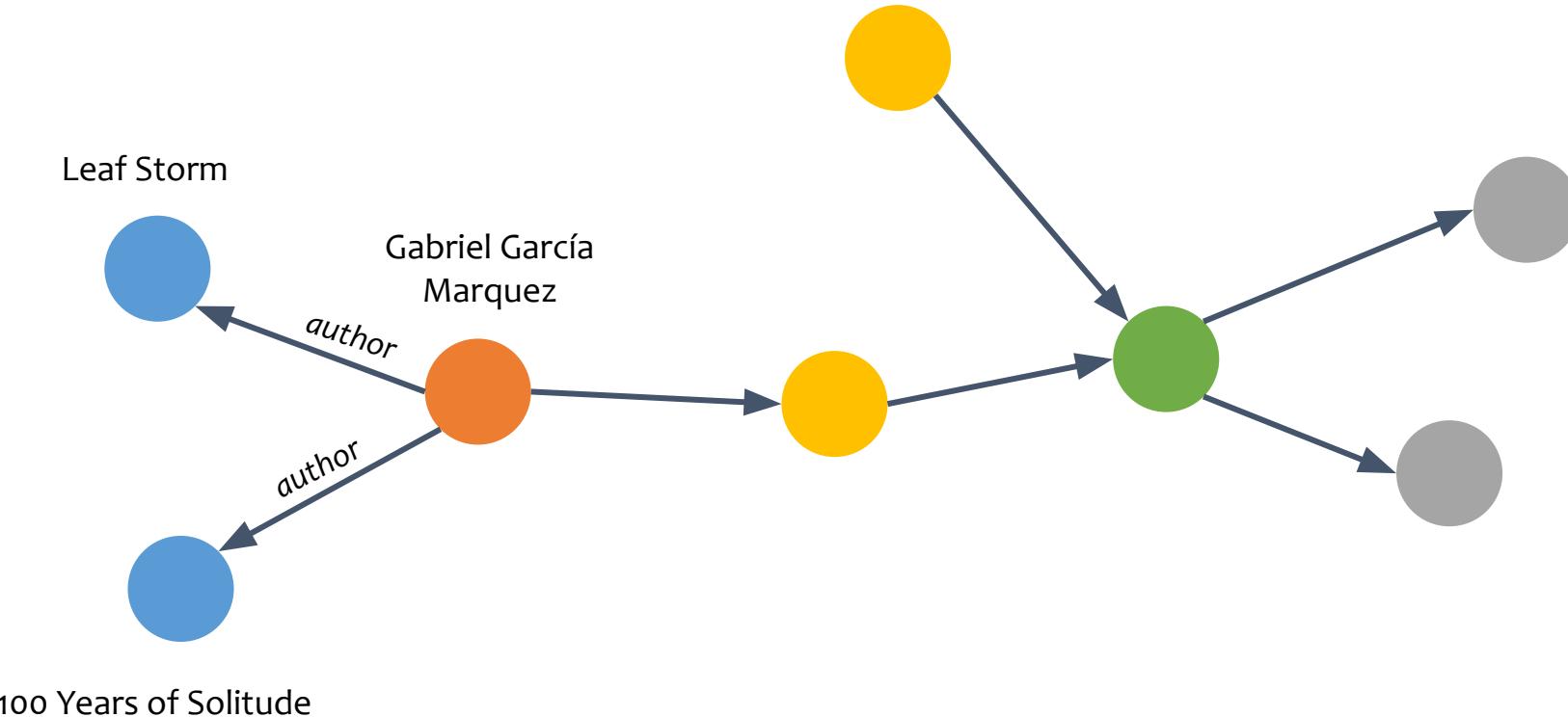
Introducing the Knowledge Graph: things, not strings

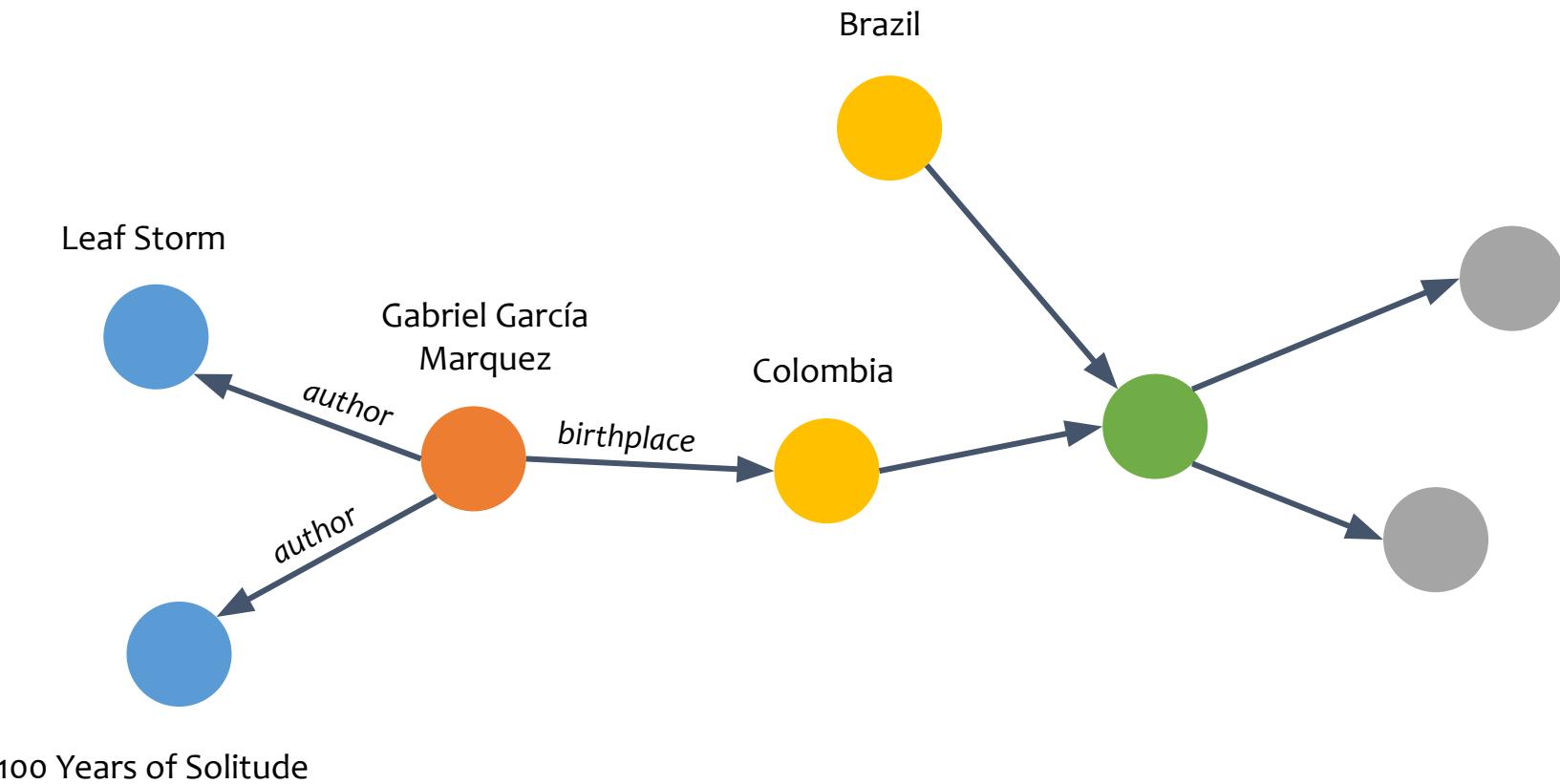
“The Knowledge Graph enables you to search for things, people or places that Google knows about—**landmarks, celebrities, cities, sports teams, buildings, geographical features, movies, celestial objects, works of art and more**—and instantly get information that’s relevant to your query.

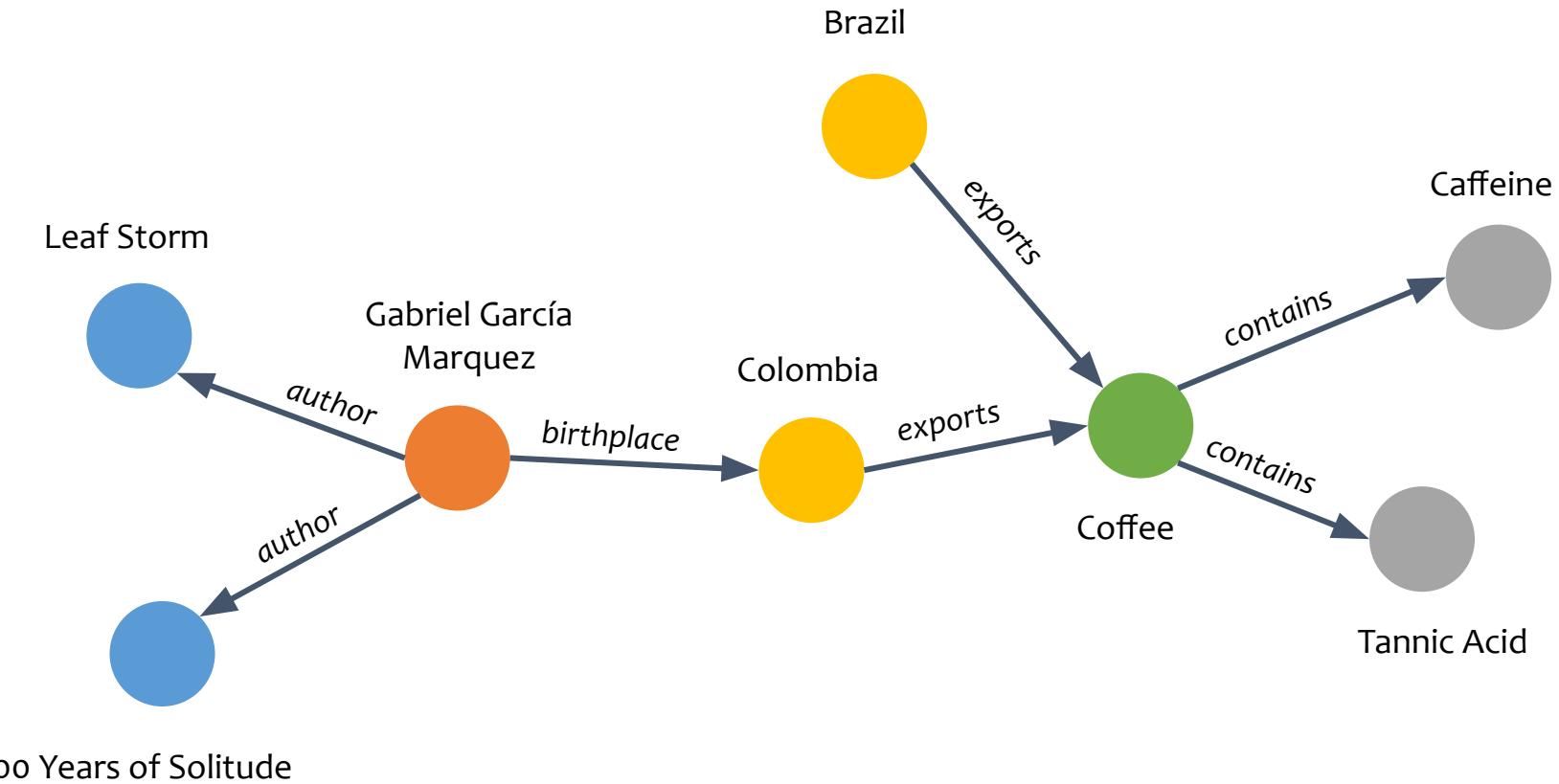
This is a critical first step towards building the next generation of search, which taps into the collective intelligence of the web and **understands the world a bit more like people do.**”

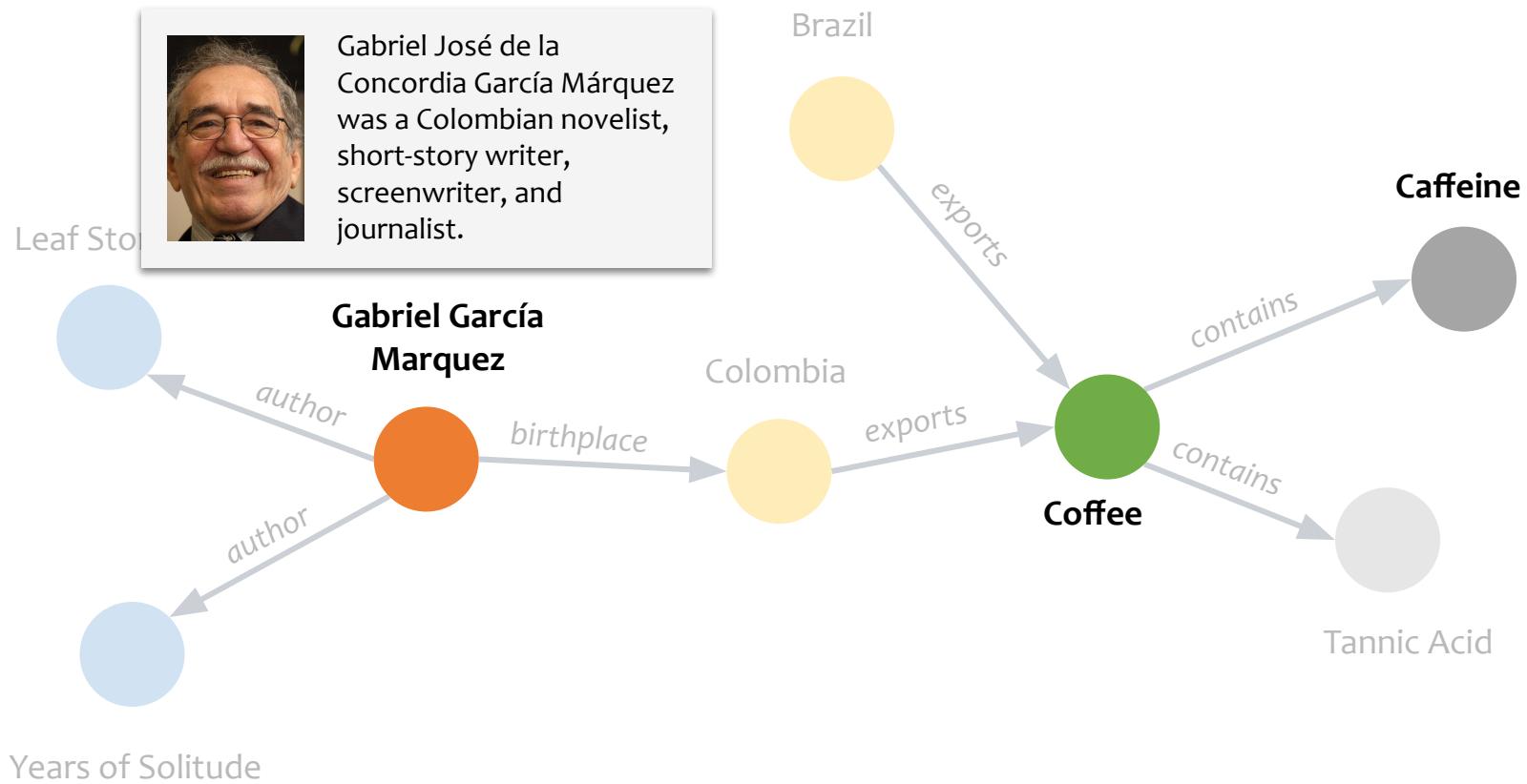
<https://blog.google/products/search/introducing-knowledge-graph-things-not/>

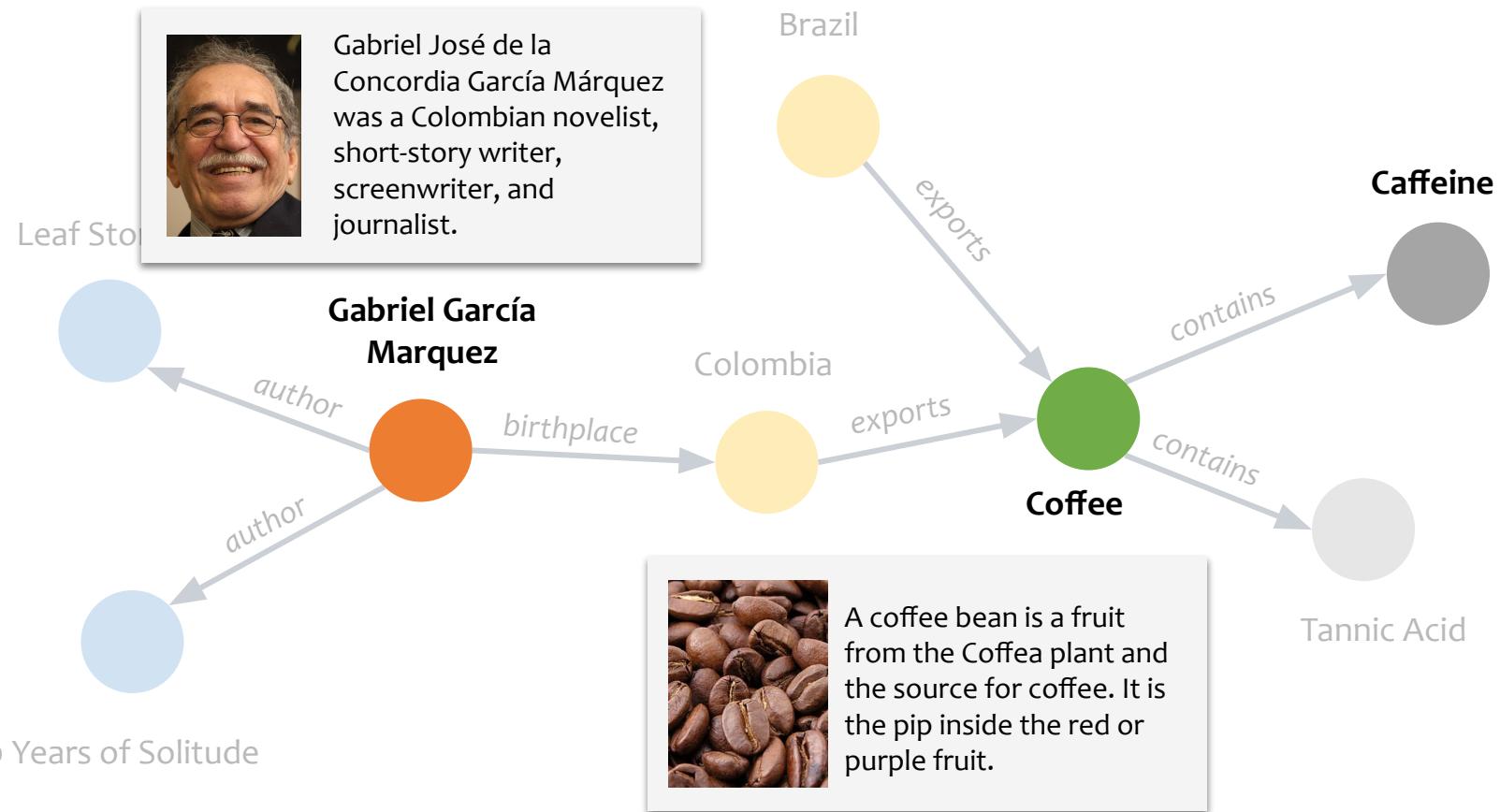


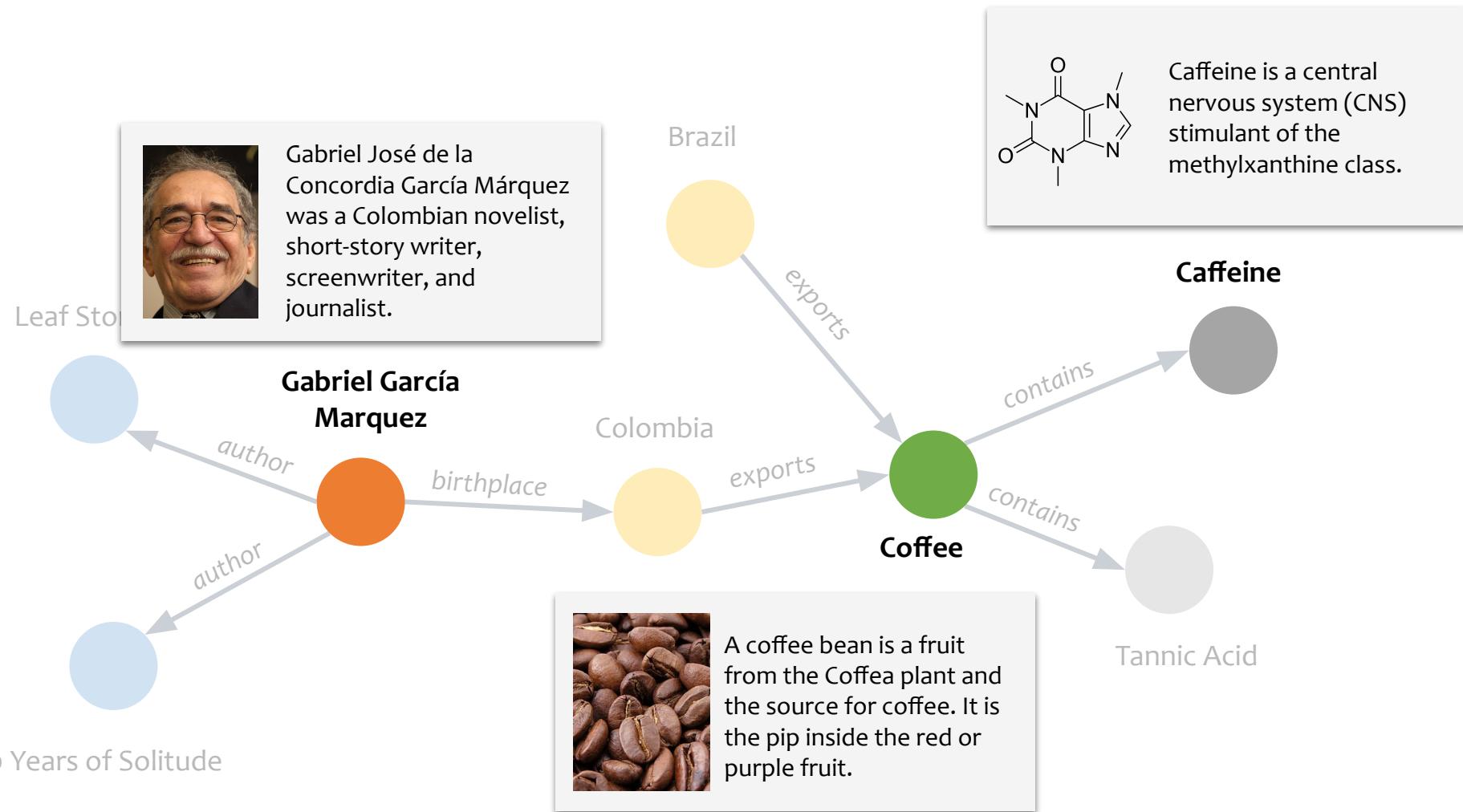












Knowledge Graphs with attribute data

- Textual descriptions
- Images
- Molecular structures
- Protein sequences



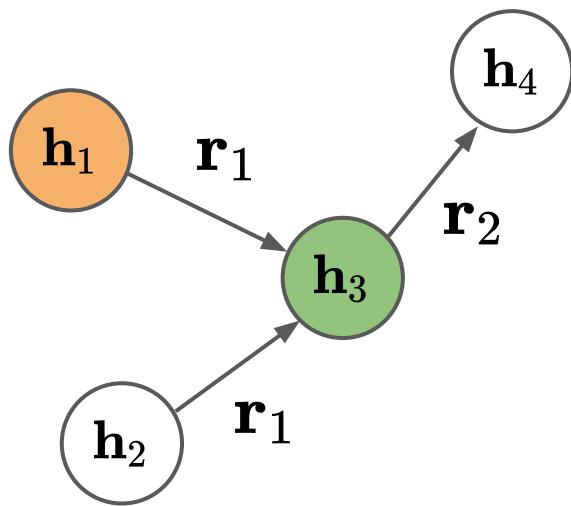
Knowledge Graphs with attribute data

- Textual descriptions
- Images
- Molecular structures
- Protein sequences



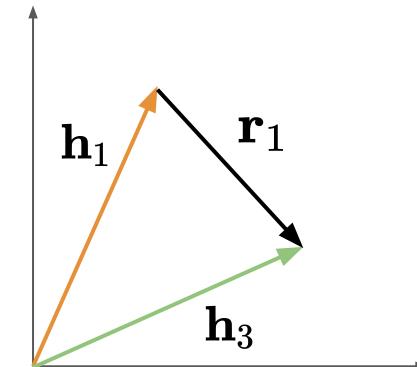
Can we take advantage of such a wealth of data to learn better representations for knowledge graphs?

Incorporating textual data



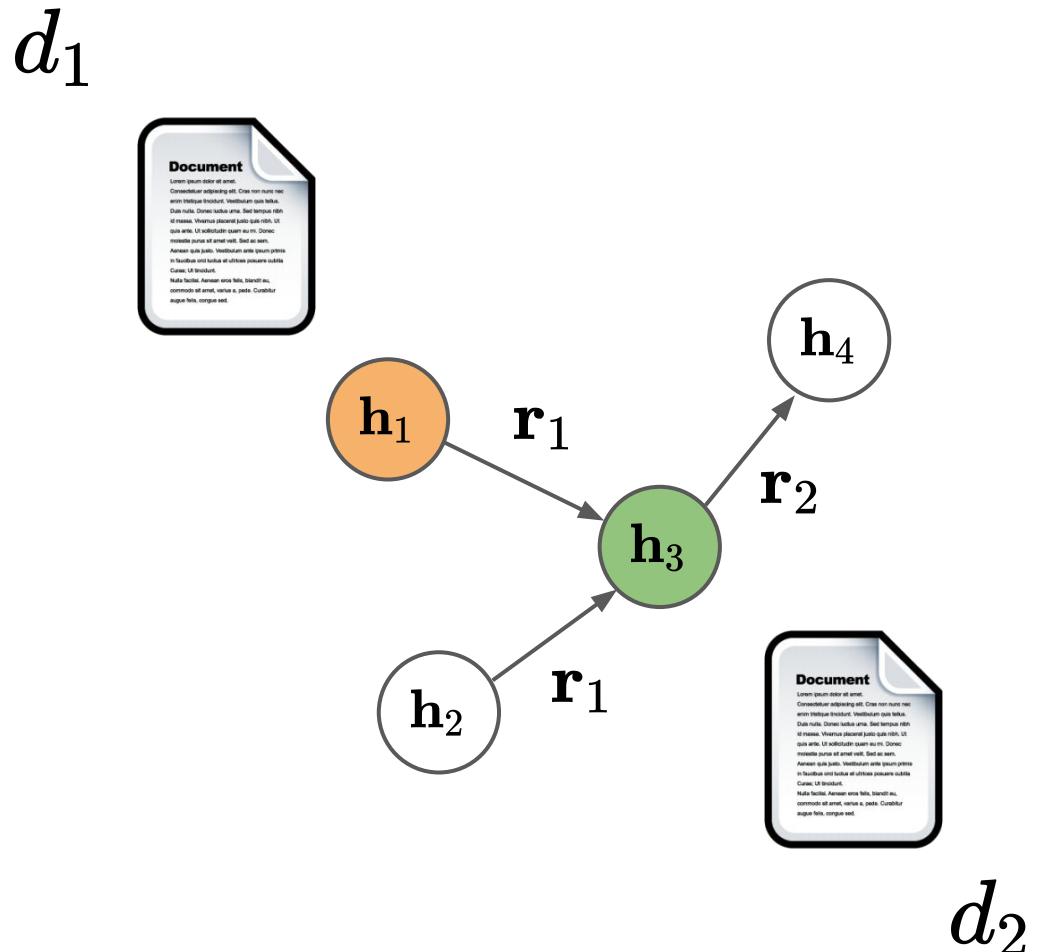
TransE embeddings optimized to minimize

$$\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$$



Xie et al. (2016), “Representation Learning of Knowledge Graphs with Entity Descriptions”.

Incorporating textual data



TransE embeddings optimized to minimize

$$\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$$

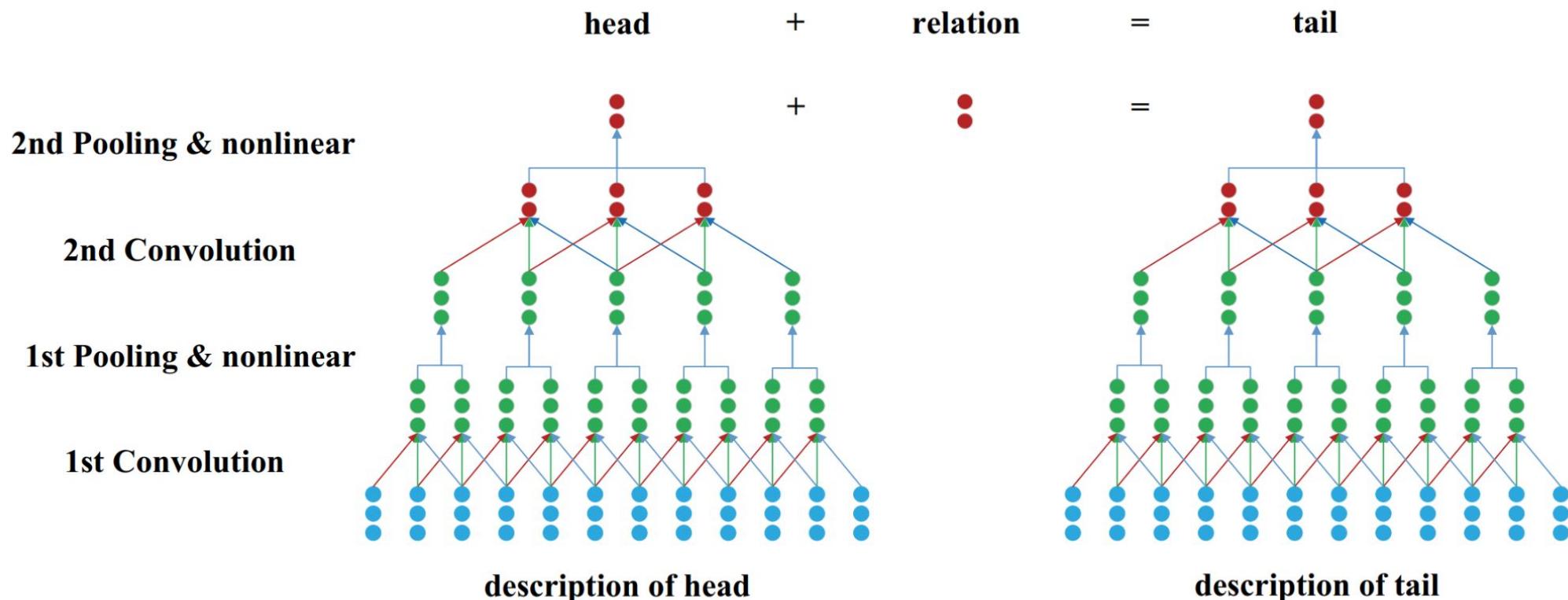
and

$$\|f(d_1) + \mathbf{r} + f(d_2)\|$$

where f is a **description encoder**.

Description encoders

- A function mapping text to a fixed-size vector.
- DKRL relies on **word embeddings** and a **CNN-based encoder**.



Xie et al. (2016), “Representation Learning of Knowledge Graphs with Entity Descriptions”.

Embeddings from textual descriptions

Link prediction

Metric	Mean Rank		Hits@10(%)	
	Raw	Filter	Raw	Filter
TransE	210	119	48.5	66.1
DKRL(CBOW)	236	151	38.3	51.8
DKRL(CNN)	200	113	44.3	57.6
DKRL(CNN)+TransE	181	91	49.6	67.4

- Performance comparable with regular TransE.
- Open questions:
 - Other embedding models (e.g. ComplEx)?
 - More expressive textual encoders?

Entity classification

Metric	FB15K	FB20K
TransE	87.9	-
BOW	86.3	57.5
DKRL(CBOW)	89.3	52.0
DKRL(CNN)	90.1	61.9

Xie et al. (2016), “Representation Learning of Knowledge Graphs with Entity Descriptions”.

Beyond DKRL

- Other KG embedding models

Model	Function
TransE	$-\ \mathbf{e}_i + \mathbf{r}_j - \mathbf{e}_k\ _p$
DistMult	$\langle \mathbf{e}_i, \mathbf{r}_j, \mathbf{e}_k \rangle$
ComplEx	$\text{Re}(\langle \mathbf{e}_i, \mathbf{r}_j, \bar{\mathbf{e}}_k \rangle)$
SimplE	$\frac{1}{2} (\langle \mathbf{e}_{i1}, \mathbf{r}_{j1}, \mathbf{e}_{k1} \rangle + \langle \mathbf{e}_{i2}, \mathbf{r}_{j2}, \mathbf{e}_{k2} \rangle)$

- Pretrained language models: BERT, RoBERTa, etc.

Beyond DKRL

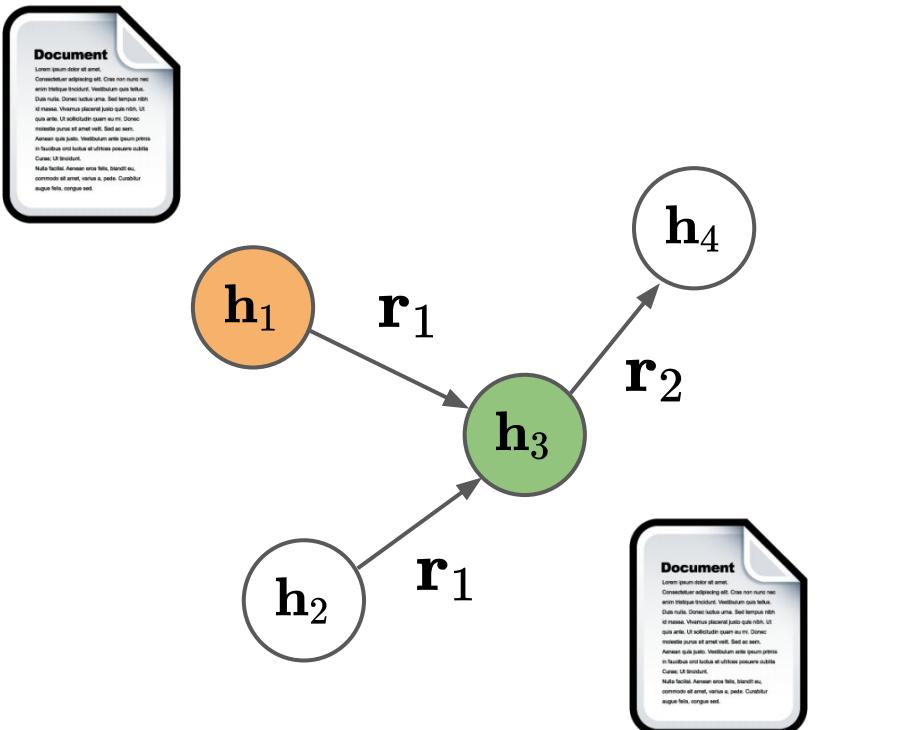
- Other KG embedding models

Model	Function
TransE	$-\ \mathbf{e}_i + \mathbf{r}_j - \mathbf{e}_k\ _p$
DistMult	$\langle \mathbf{e}_i, \mathbf{r}_j, \mathbf{e}_k \rangle$
ComplEx	$\text{Re}(\langle \mathbf{e}_i, \mathbf{r}_j, \bar{\mathbf{e}}_k \rangle)$
SimplE	$\frac{1}{2} (\langle \mathbf{e}_{i1}, \mathbf{r}_{j1}, \mathbf{e}_{k1} \rangle + \langle \mathbf{e}_{i2}, \mathbf{r}_{j2}, \mathbf{e}_{k2} \rangle)$

- Pretrained language models: BERT, RoBERTa, etc.
- Implications on **inductive predictions** on knowledge graphs!

Incorporating textual data

d_1



TransE embeddings optimized to minimize

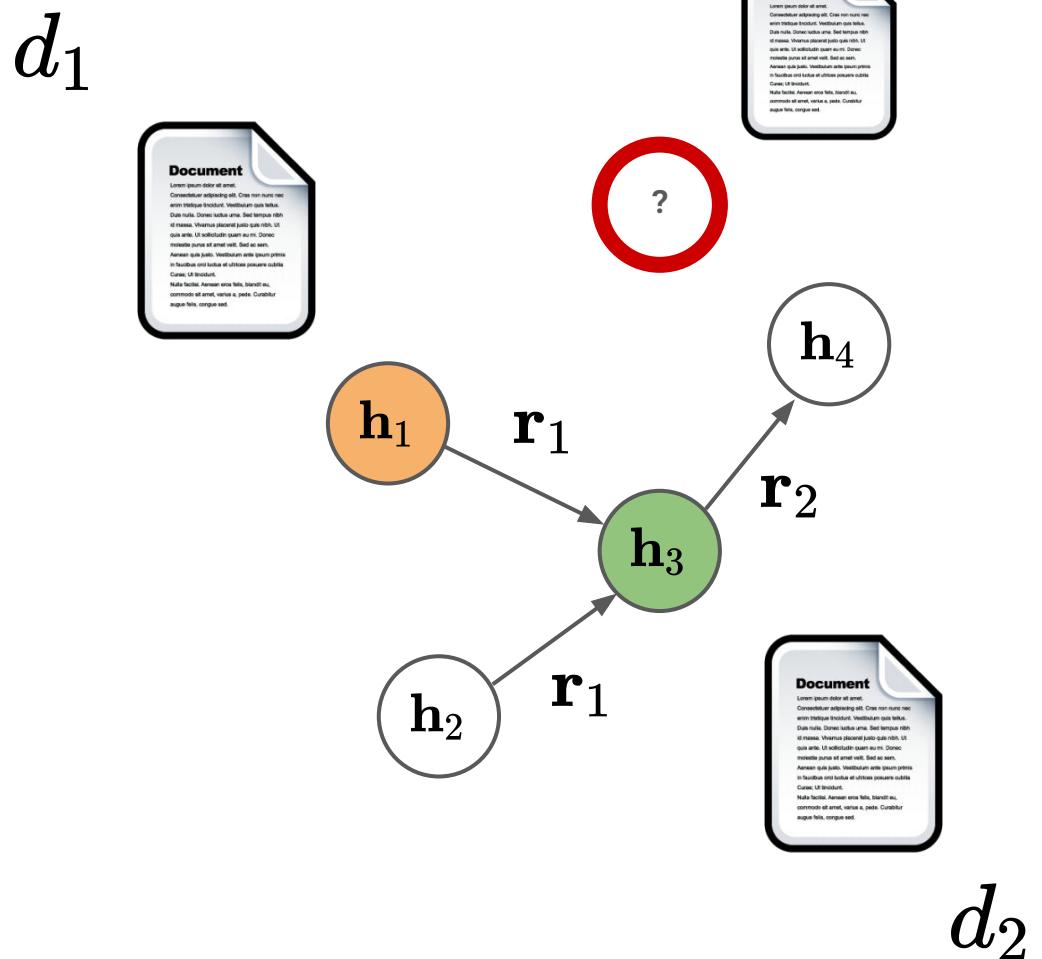
$$\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$$

and

$$\|f(d_1) + \mathbf{r} + f(d_2)\|$$

where f is a **description encoder**.

Incorporating textual data



TransE embeddings optimized to minimize

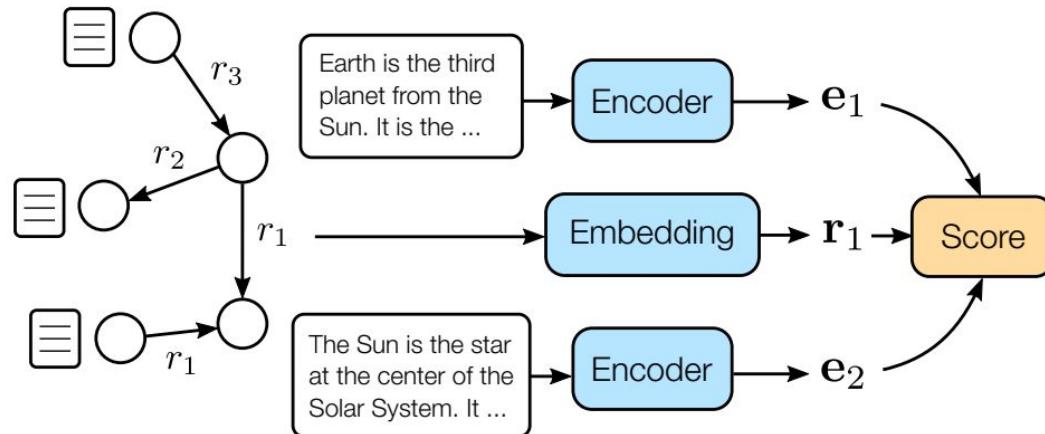
$$\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$$

and

$$\|f(d_1) + \mathbf{r} + f(d_2)\|$$

where f is a **description encoder**.

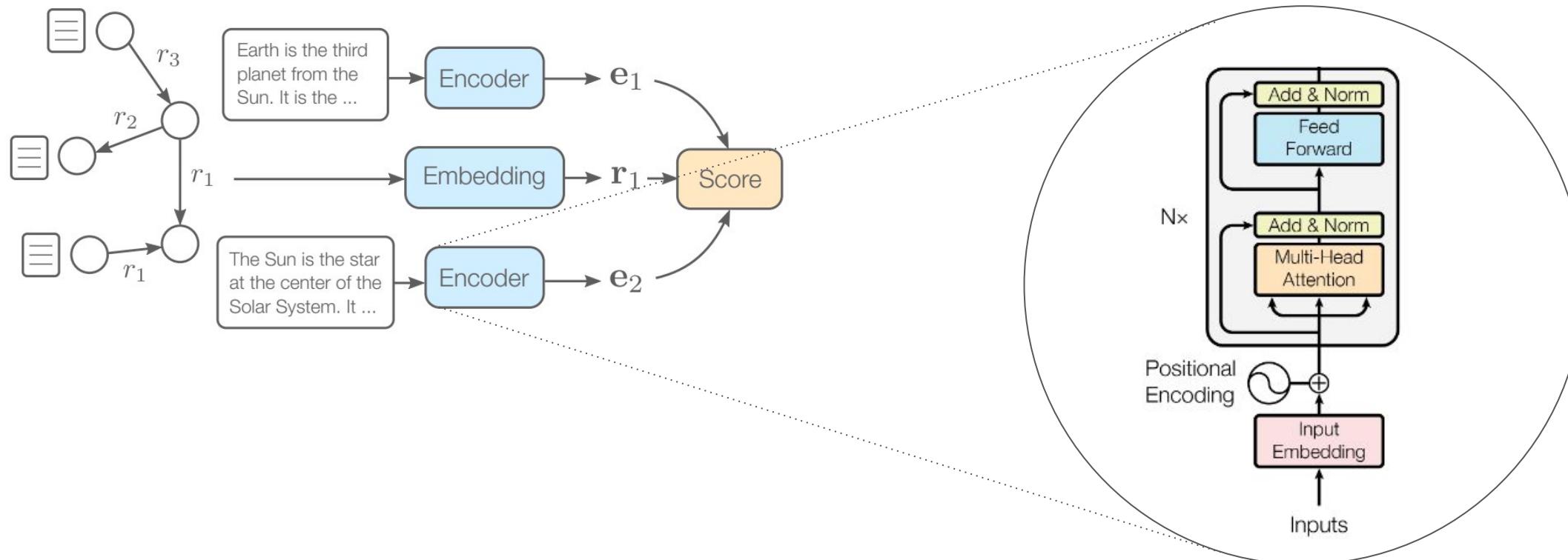
BERT for Link Prediction (BLP)



Model	Score
TransE	$-\ \mathbf{e}_i + \mathbf{r}_j - \mathbf{e}_k\ _p$
DistMult	$\langle \mathbf{e}_i, \mathbf{r}_j, \mathbf{e}_k \rangle$
ComplEx	$\text{Re}(\langle \mathbf{e}_i, \mathbf{r}_j, \bar{\mathbf{e}}_k \rangle)$
SimplE	$\frac{1}{2} (\langle \mathbf{e}_{i1}, \mathbf{r}_{j1}, \mathbf{e}_{k1} \rangle + \langle \mathbf{e}_{i2}, \mathbf{r}_{j2}, \mathbf{e}_{k2} \rangle)$

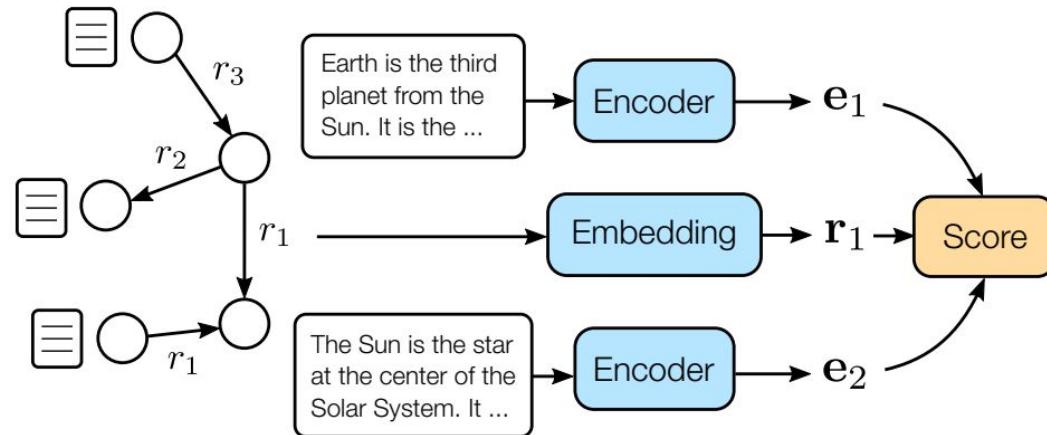
Daza, D., Cochez, M., & Groth, P. (2021). "Inductive Entity Representations from Text via Link Prediction"

BERT for Link Prediction (BLP)



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

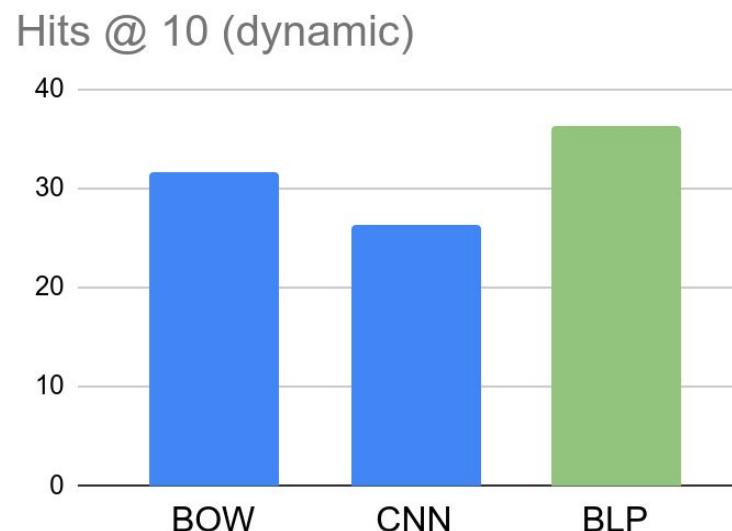
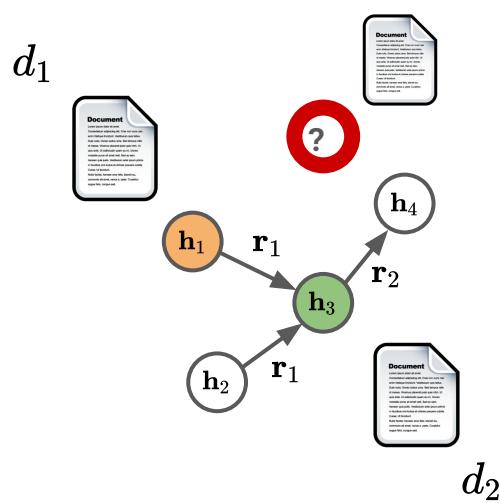
BERT for Link Prediction (BLP)



Testing:

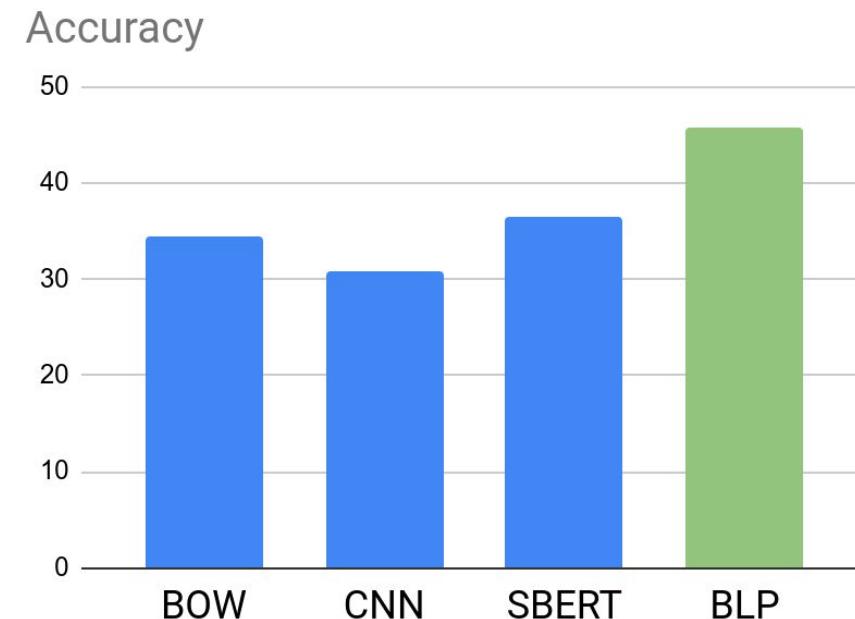
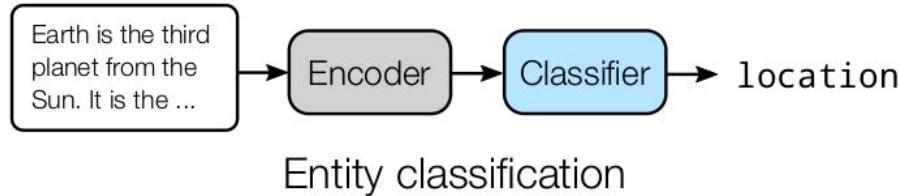
- Predictions with new entities added to the graph.
- Predictions over a disconnected subset of the graph.

BERT for Link Prediction (BLP)



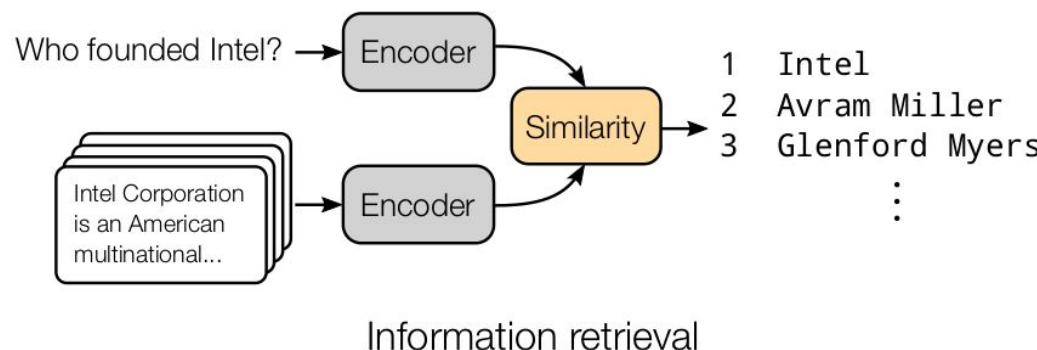
Daza, D., Cochez, M., & Groth, P. (2021). "Inductive Entity Representations from Text via Link Prediction"

BLP for entity classification



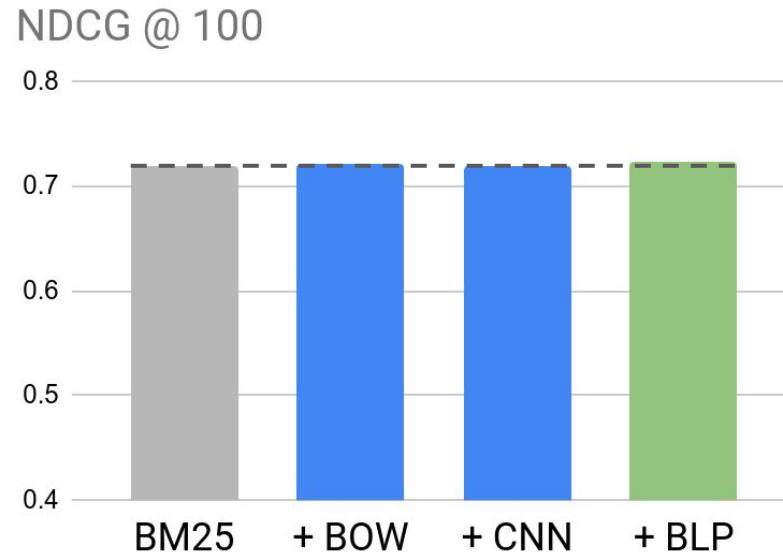
Daza, D., Cochez, M., & Groth, P. (2021). "Inductive Entity Representations from Text via Link Prediction"

BLP for information retrieval

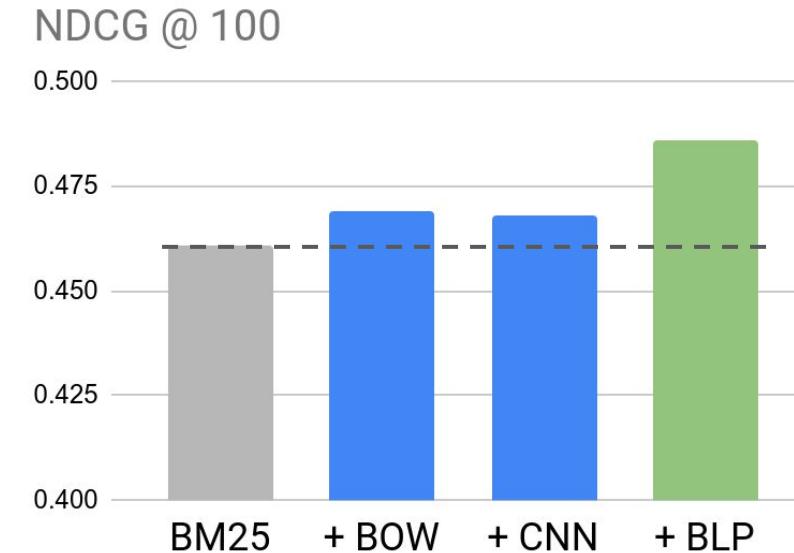


- Queries on entity-oriented search:
 - Airports in Germany
 - What is the longest river?
 - bicycle holiday nature
- Approach: **reranking** BM25 results

BLP for information retrieval



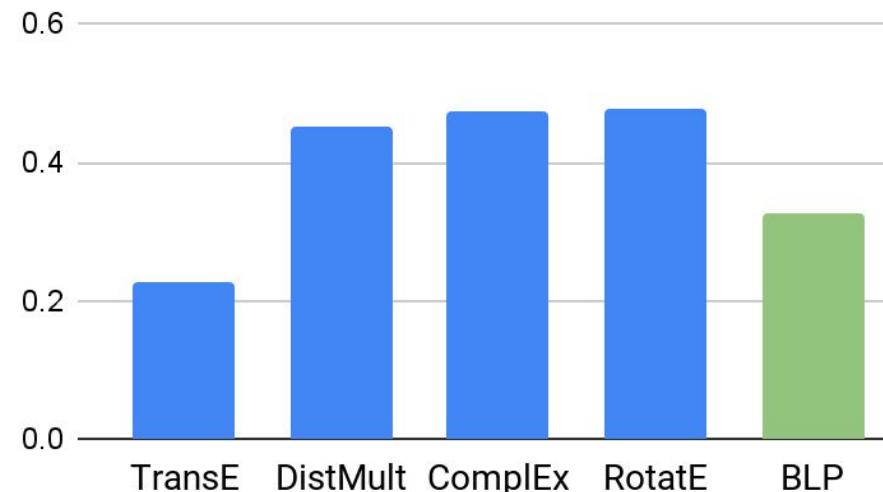
“bicycle holiday nature”



“What is the longest river?”

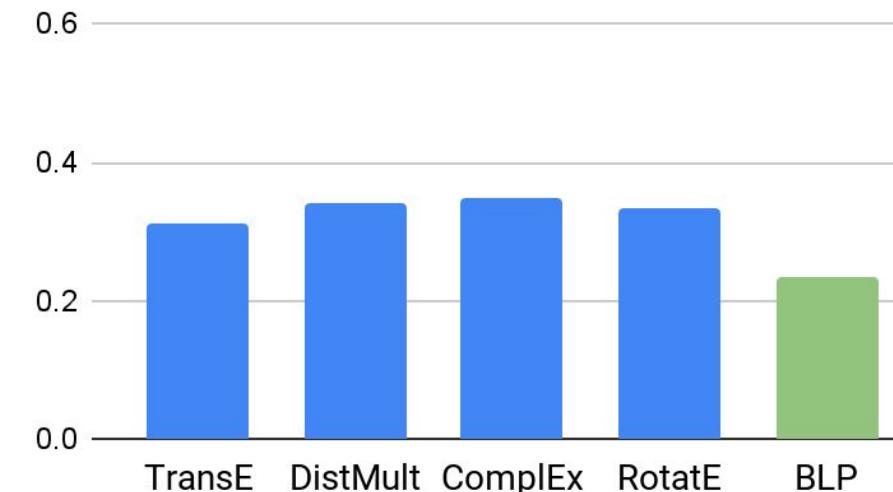
BLP for transductive link prediction

MRR



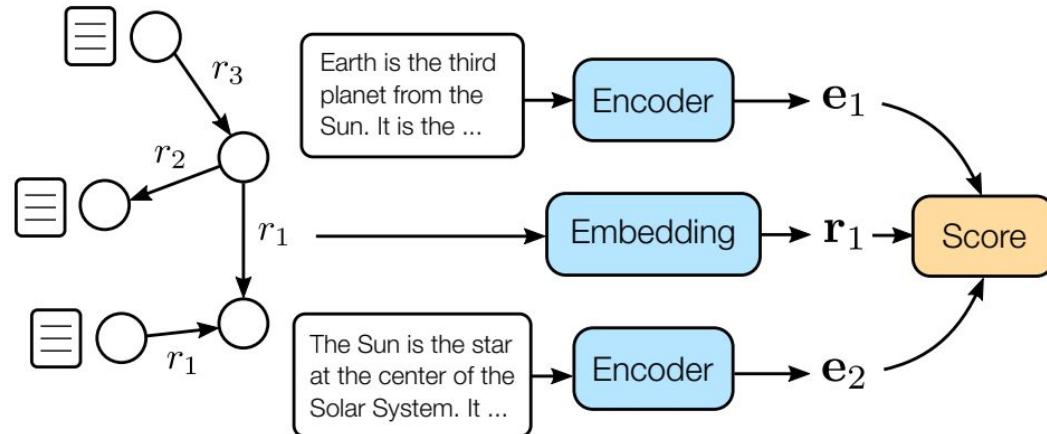
WN18RR

MRR



FB15k-237

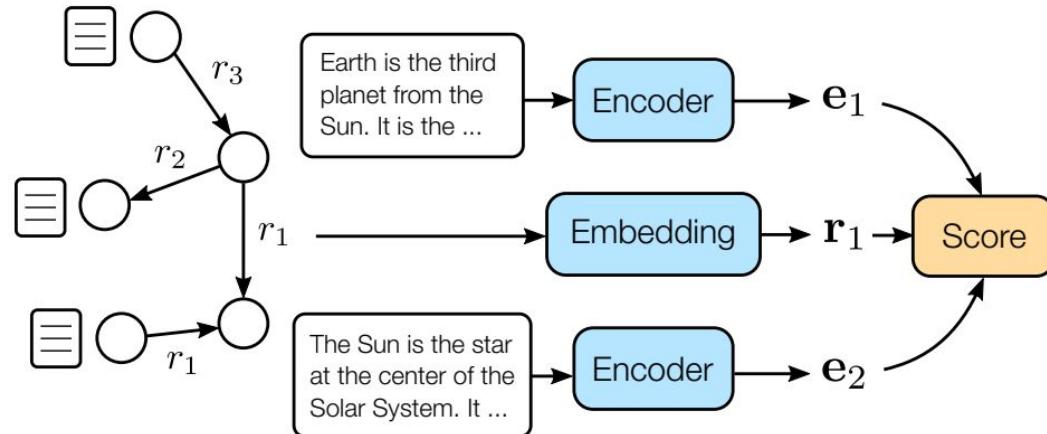
KG embeddings from textual descriptions



- **Pros:**
 - Pretrained language models improve generalization.
 - Relying on textual descriptions enables inductive predictions.
- **Cons:**
 - Encoder performance can be improved.
 - Increased computational costs.

Daza, D., Cochez, M., & Groth, P. (2021). "Inductive Entity Representations from Text via Link Prediction"

KG embeddings from textual descriptions

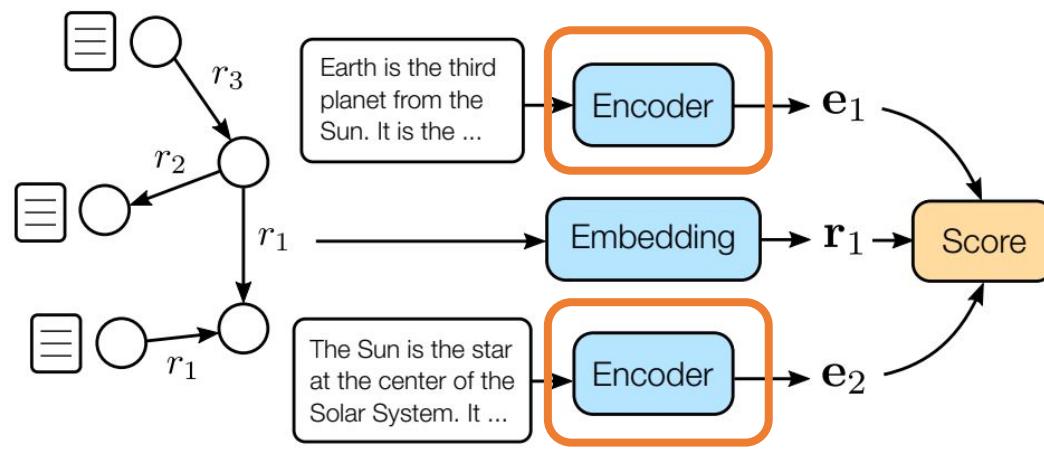


- **Pros:**
 - Pretrained language models improve generalization.
 - Relying on textual descriptions enables inductive predictions.
- **Cons:**
 - **Encoder performance can be improved.**
 - Increased computational costs.

Daza, D., Cochez, M., & Groth, P. (2021). "Inductive Entity Representations from Text via Link Prediction"

Improving encoder performance: SimKGC

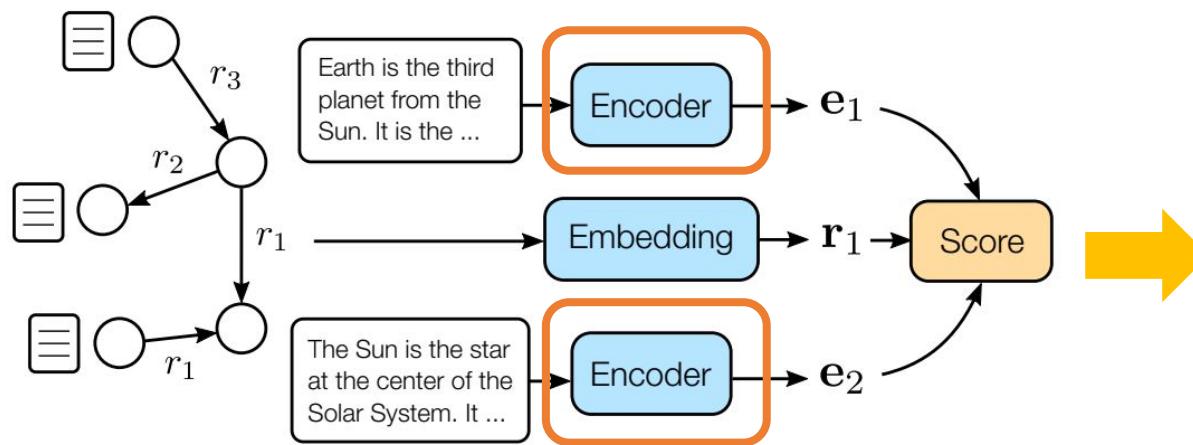
- Solution 1 - **SimKGC**: separate into a cross-encoder for the **head** and **relation**, and an encoder for the **tail**.



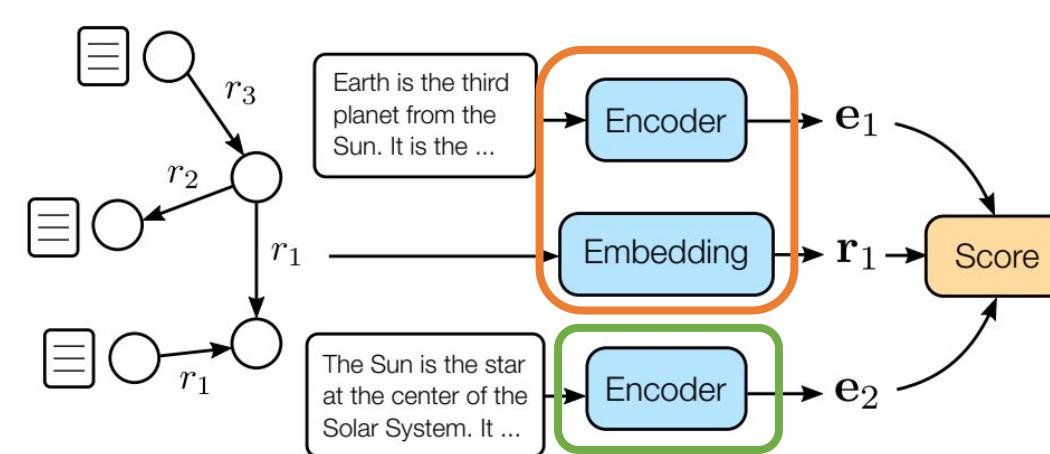
Encoder shared for head and tail entities

Improving encoder performance: SimKGC

- Solution 1 - **SimKGC**: separate into a cross-encoder for the **head** and **relation**, and an encoder for the **tail**.



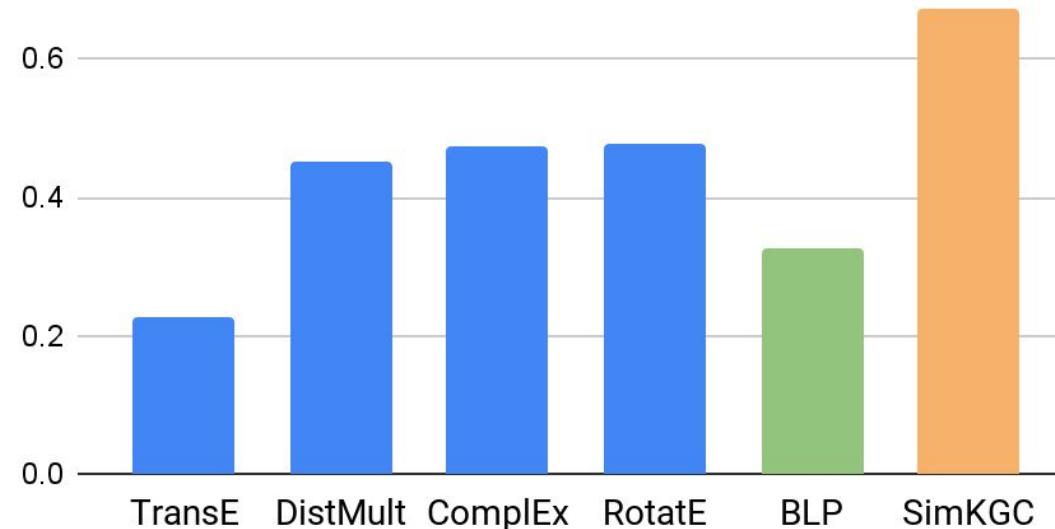
Encoder shared for head and tail entities



Encoder for head + relation, and separate encoder for tail entities
~2x the number of parameters

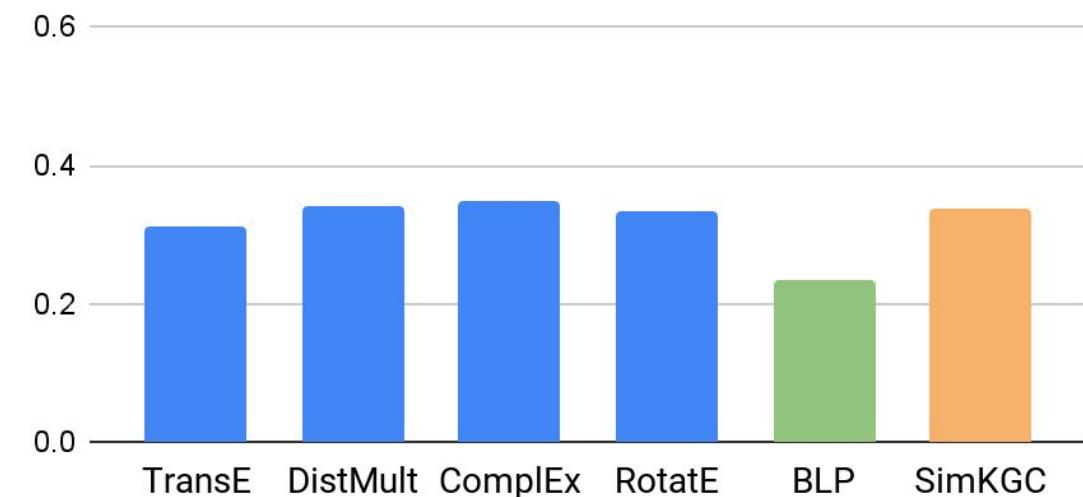
Improving encoder performance: SimKGC

MRR



WN18RR

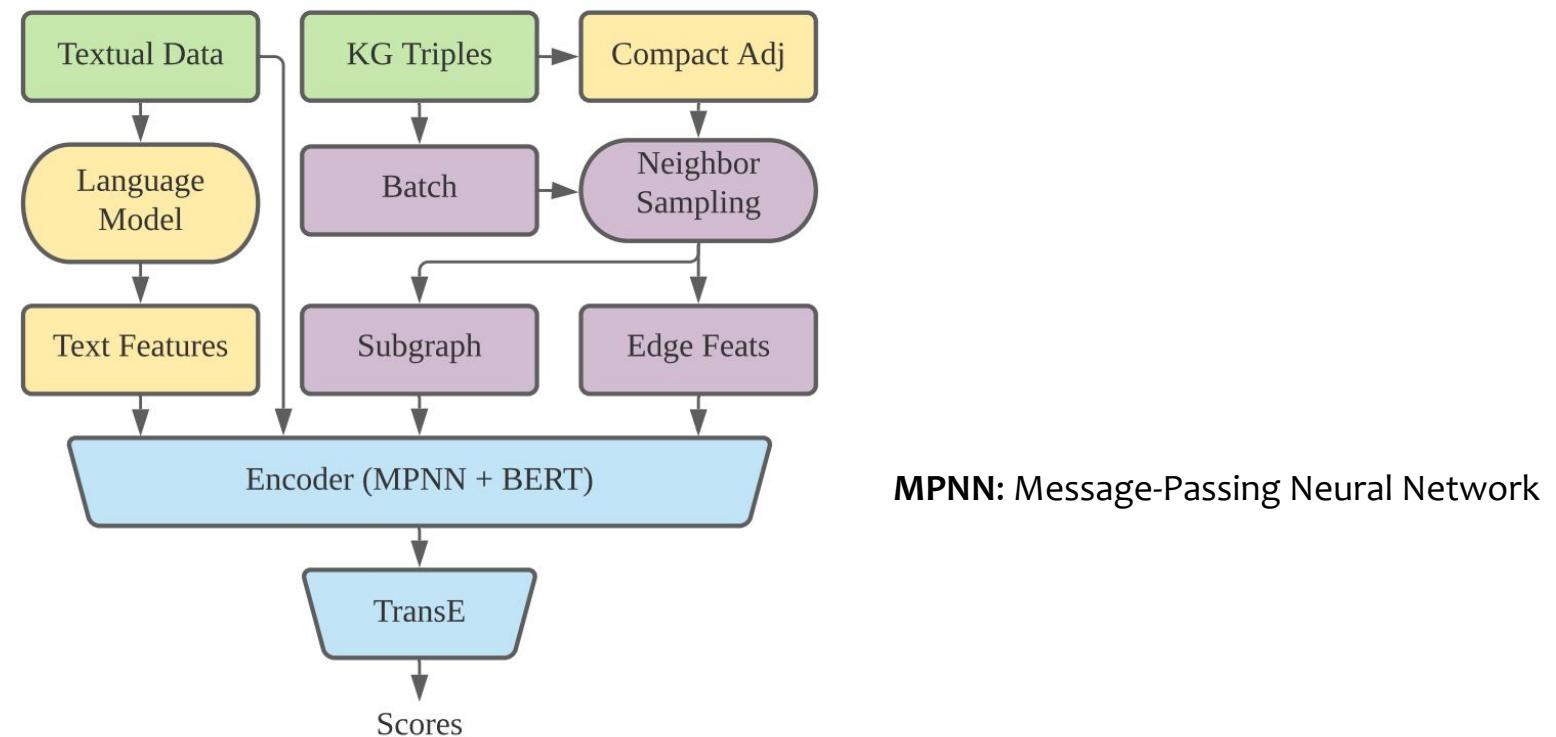
MRR



FB15k-237

Improving encoder performance: StATIK

- Solution 2 - **StATIK**: combine textual encoders with graph neural networks.



Markowitz et al. (2022), "StATIK: Structure and Text for Inductive Knowledge Graph Completion"

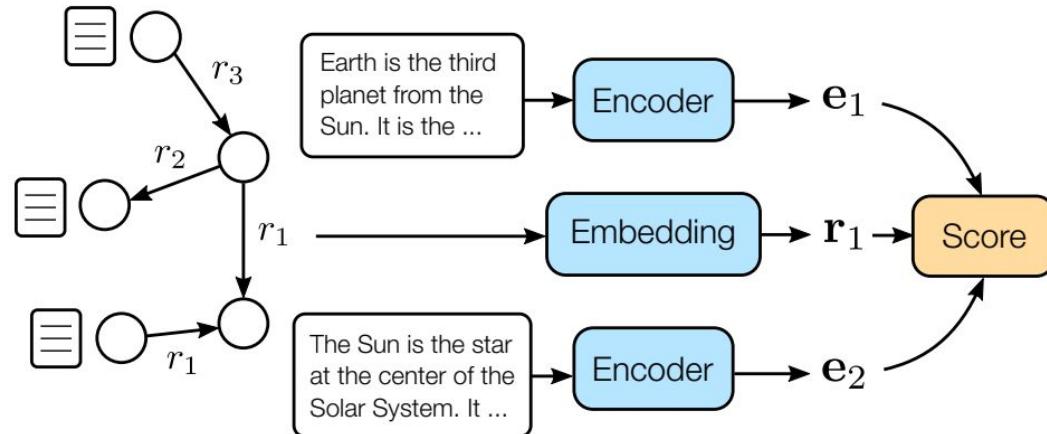
Improving encoder performance: StATIK

- Solution 2 - **StATIK**: combine textual encoders with graph neural networks.

Model	WN18RR				FB15k-237			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Commonsense-KBC**	0.01	0.0055	0.009	0.019	0.00028	0.00001	0.000028	0.000056
GloVe-BOW*	0.170	0.055	0.215	0.405	0.172	0.099	0.188	0.316
BE-BOW*	0.180	0.045	0.244	0.450	0.173	0.103	0.184	0.316
GloVe-DKRL*	0.115	0.031	0.141	0.282	0.112	0.062	0.111	0.211
BE-DKRL*	0.139	0.048	0.169	0.320	0.144	0.084	0.151	0.263
BLP-TransE†	0.285	0.135	0.361	0.580	0.195	0.113	0.213	0.363
BLP-DistMult†	0.248	0.135	0.288	0.481	0.146	0.076	0.156	0.286
BLP-ComplEx†	0.261	0.156	0.297	0.472	0.148	0.081	0.154	0.283
BLP-SimplE†	0.239	0.144	0.265	0.435	0.144	0.077	0.152	0.274
StAR‡	0.321	0.192	0.381	0.576	0.163	0.092	0.176	0.309
StATIK	0.516	0.425	0.558	0.690	0.224	0.143	0.248	0.381
Improvement	60.7%	121.3%	54.6%	19.8%	14.9%	26.5%	16.4%	5.0%

Markowitz et al. (2022), “StATIK: Structure and Text for Inductive Knowledge Graph Completion”

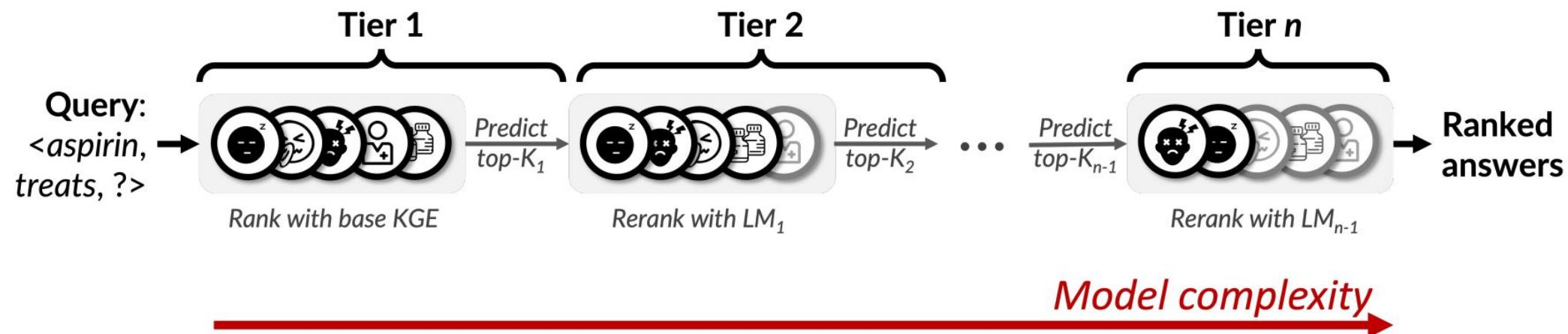
KG embeddings from textual descriptions



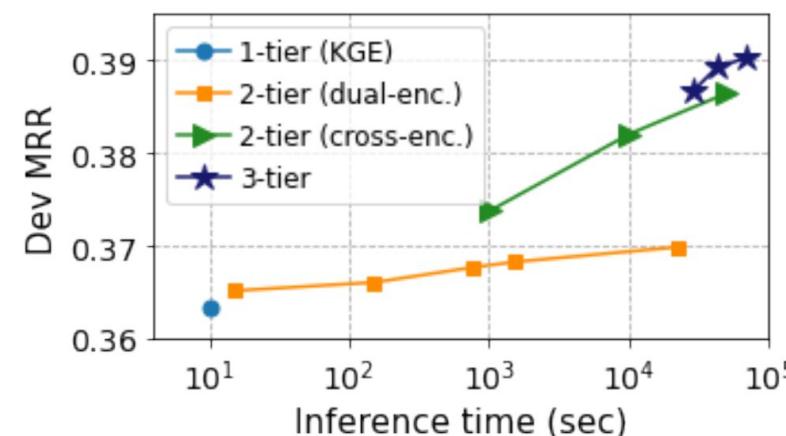
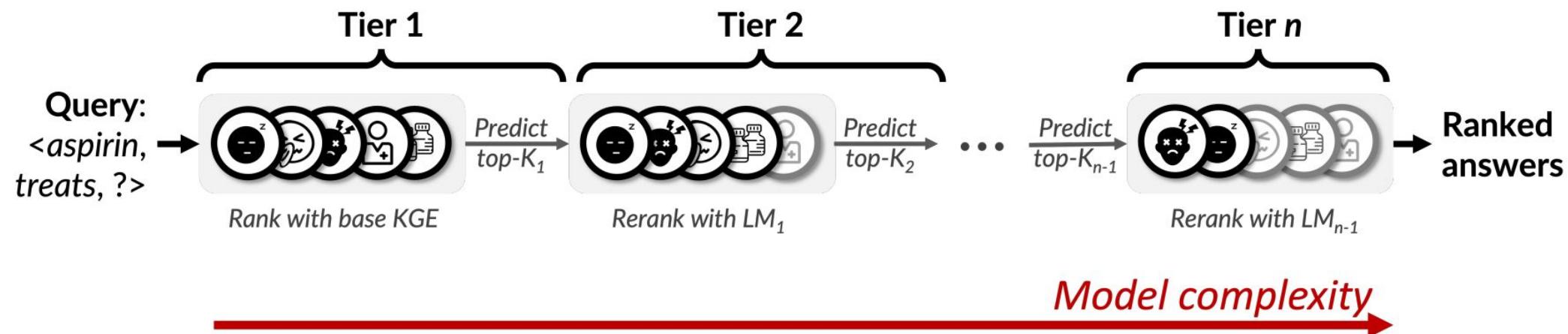
- **Pros:**
 - Pretrained language models improve generalization.
 - Relying on textual descriptions enables inductive predictions.
- **Cons:**
 - Encoder performance can be improved.
 - **Increased computational costs.**

Daza, D., Cochez, M., & Groth, P. (2021). "Inductive Entity Representations from Text via Link Prediction"

Reducing computational cost: CascaDER



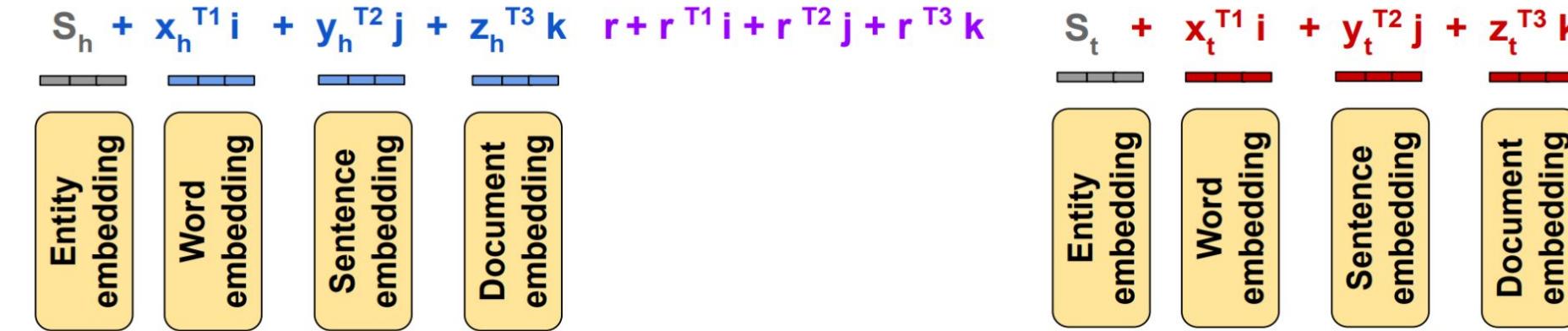
Reducing computational cost: CascaDER



Safavi et al. (2022), “CascaDER: Cross-Modal Cascading for Knowledge Graph Link Prediction”

Reducing computational cost: Dihedron embeddings

- Combine pretrained features into common space



Danny Pena (born June 17, 1968 in Inglewood, California) is a retired U.S. soccer defensive midfielder. He spent most of his career, both indoors and outdoors, with teams in western U.S. ...

Q5220733 (Danny Pena)

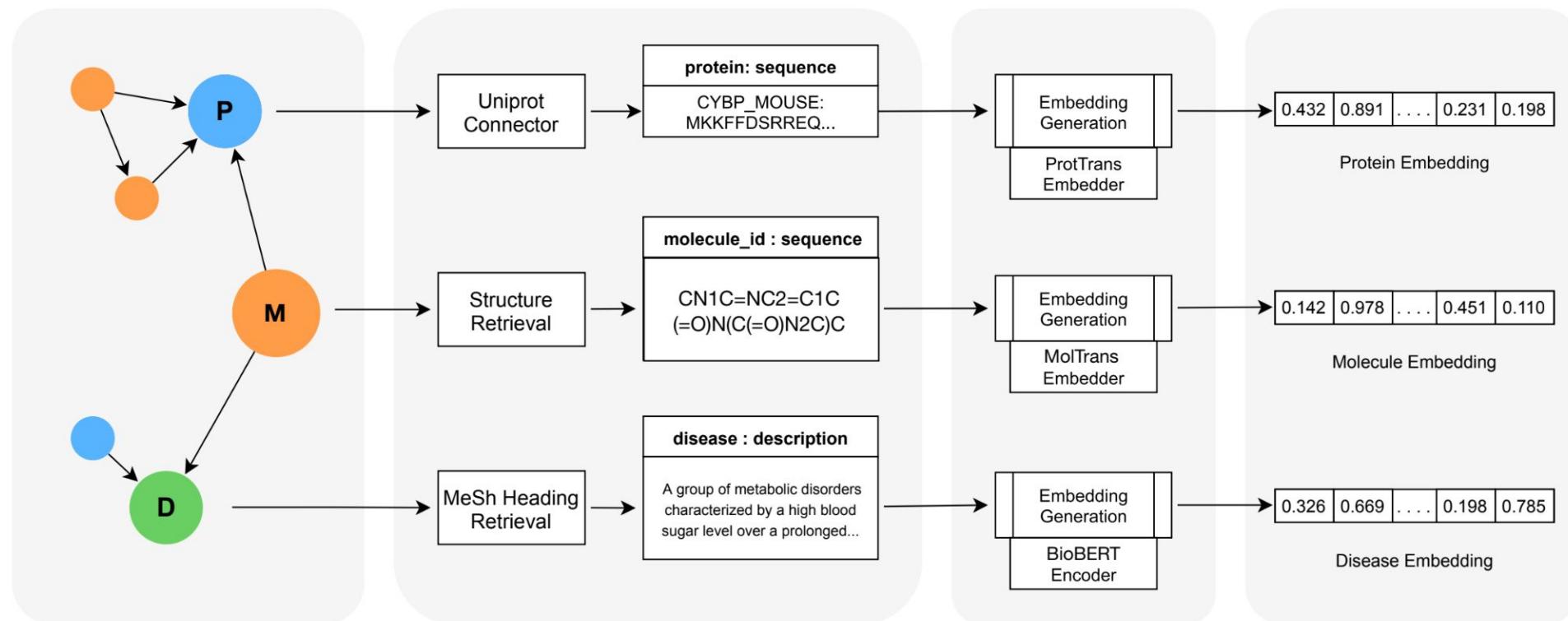
P19
(place of birth)

Inglewood is a city in southwestern Los Angeles County, California, southwest of downtown Los Angeles. As of the 2010 U.S. Census, the city had a population of 109,673. It was incorporated ...

Q621549 (Inglewood)

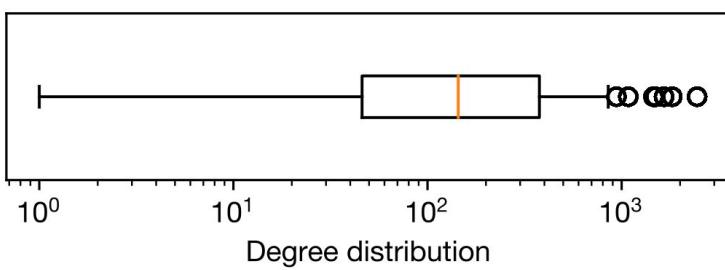
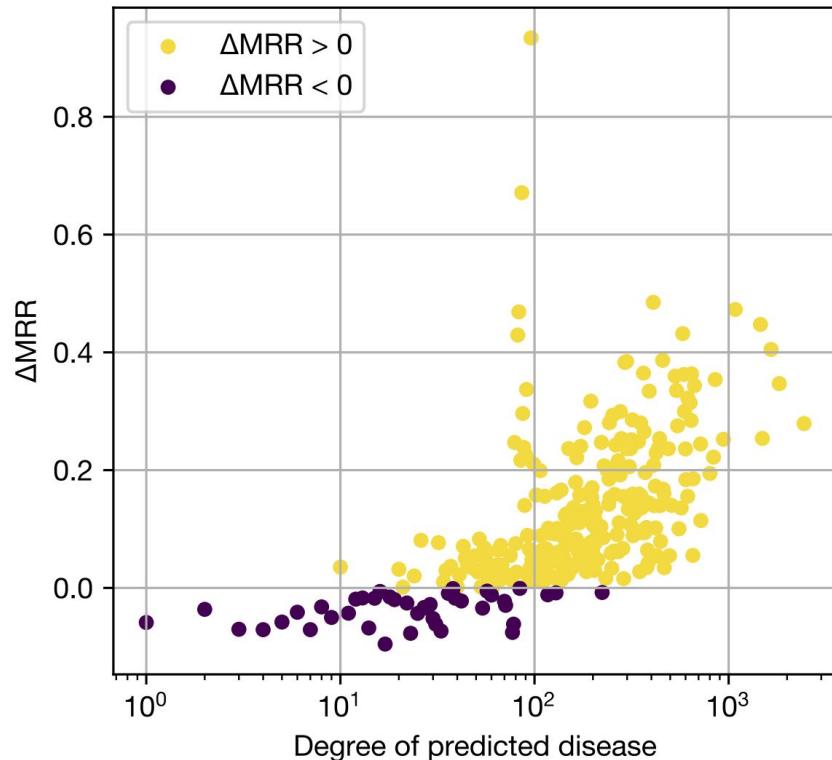
Further applications: BioBLP

- The previous ideas can be extended to domains such as biology and medicine.



Daza et al. (2023), "BioBLP: A Modular Framework for Learning on Multimodal Biomedical Knowledge Graphs"

Further applications: BioBLP



- The previous ideas can be extended to domains such as biology and medicine.
- Optimization is challenging, but promising towards **low degree entities**.

Daza et al. (2023), “BioBLP: A Modular Framework for Learning on Multimodal Biomedical Knowledge Graphs”

Knowledge Graphs with attribute data

- Textual descriptions
- Images
- Molecular structures
- Protein sequences



Can we take advantage of such a wealth of data to learn better representations for knowledge graphs?

Yes!

Knowledge Graphs with attribute data

- There is more to KGs than entity identifiers.
- Attribute encoders open up new possibilities, by
 - Incorporating background knowledge in their design
 - Sharing parameters over entities
 - Domain-specific pretraining
 - Inductive predictions over KGs

Knowledge Graphs with attribute data

- There is more to KGs than entity identifiers.
- Attribute encoders open up new possibilities, by
 - Incorporating background knowledge in their design
 - Sharing parameters over entities
 - Domain-specific pretraining
 - Inductive predictions over KGs
- Ongoing work on
 - Improving representations learned by encoders
 - Reducing computational cost

Part 4. KG embeddings with literal and text (Daniel, 14:00 – 15:00)

4.1 Incorporating entity descriptions

- Early approaches: DKRL
- KGloVe with literals
- BLP / BioBLP

4.2 Incorporate numeric literals as features

- Incorporating other modalities, BioBLP
- multi-modal GNN (Xander)
- Other multi-modal approaches, e.g.
<https://aclanthology.org/D18-1359/>

4.3 Incorporating relation descriptions

- Delft question answering [link](#) ?
- KG-BERT, SimKGC, Statik

10 minutes Q/A

(slides mehwish)

<https://arxiv.org/abs/1910.12507>

Part 5. Advanced Topic (Michael, 15:00-15:30 & 16:00 – 16:30)

5.1a Complex query embeddings (Daniel)

- Neural query encoders
 - GQE, MPQE, StarQE, GNN-QE
- Decomposition approaches
 - CQD, CQD-A
 - LitCQD

Coffee Break (15:30 – 16:00)

16:00-16:30

5.1b NGDB? 10 minutes or so

- evaluation + framework we are designing
- query expressivity + gaps

5.2 Inductive learning settings

- NodePiece + extensions ??? Not beyond triples.
- Inductive query answering
- Summarization
- mention BLP/BioBLP/ relation description

5.3 Incorporating ontology, LLM, etc.

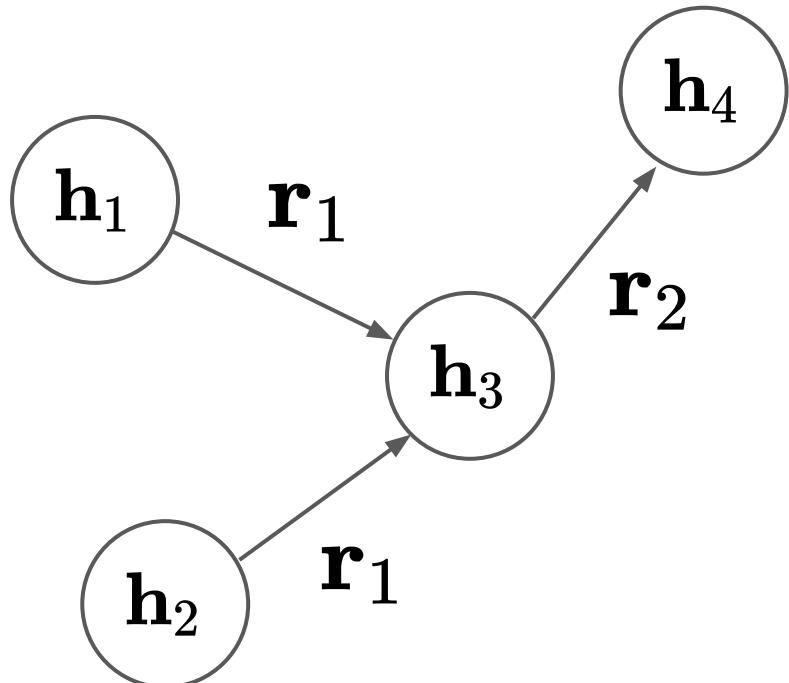
- Faithful embeddings for EL++
- Paper Pascal hitzler
- Paper D'Amato <https://openreview.net/pdf?id=C9g-pwemYxd>



Part 5. Advanced Topics

Complex Query Answering

From simple link prediction...



- Map entities and edges to vectors
 - “Embeddings”:

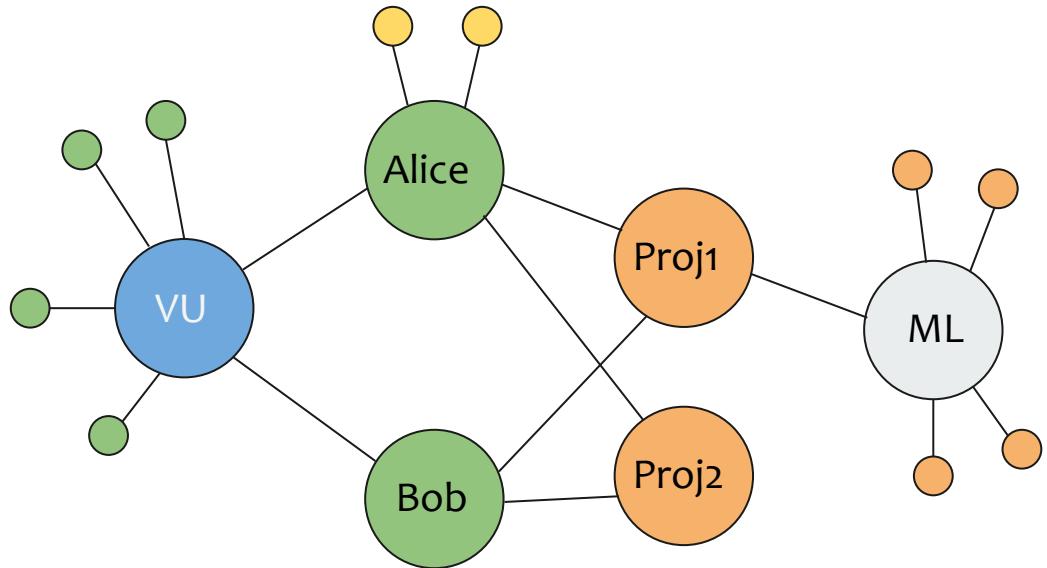
$$\mathbf{r}_i \in \mathbb{R}^d \quad \mathbf{h}_i \in \mathbb{R}^d$$

- Train embeddings to model relations as operations in embedding space:

$$\mathbf{h}_1 + \mathbf{r}_1 \approx \mathbf{h}_3$$

- We can use these to answer **simple** (1-hop) link prediction queries.

To complex queries



- Example:
 - Select all Projects, related to ML, on which Alice works
 - Answer: **Proj1**

Complex Query Answering

Neural Query
Embedding

Query
Decomposition

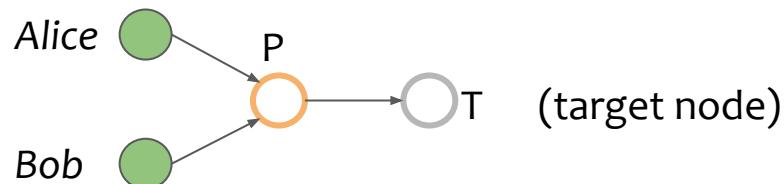
Queries as Graphs

Example:

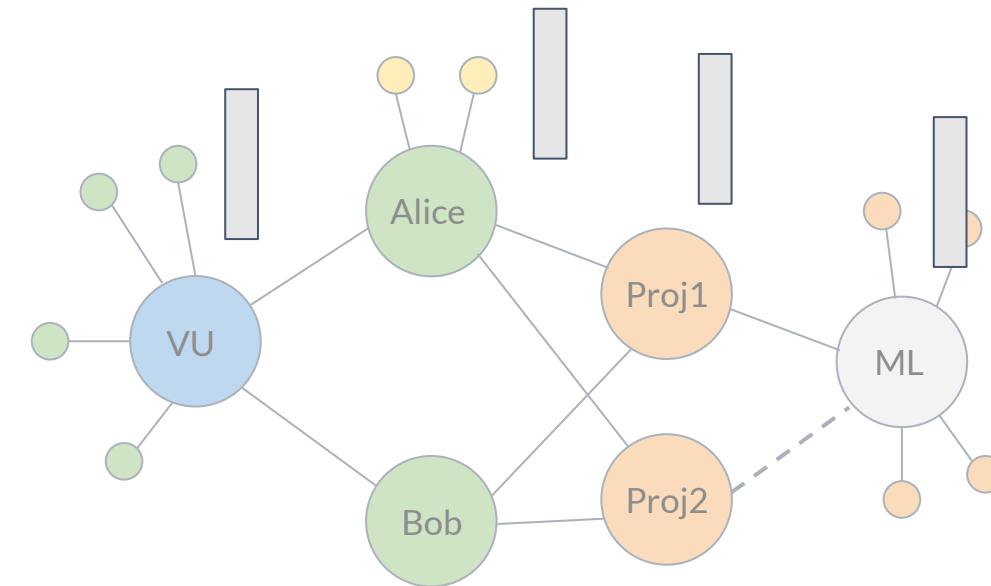
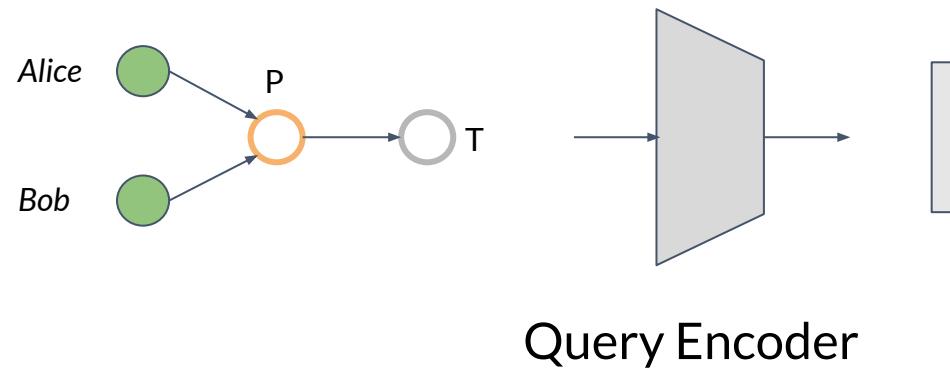
Select all topics **T** where

T is related to topic **P**

Alice works on **P** and **Bob** works on **P**

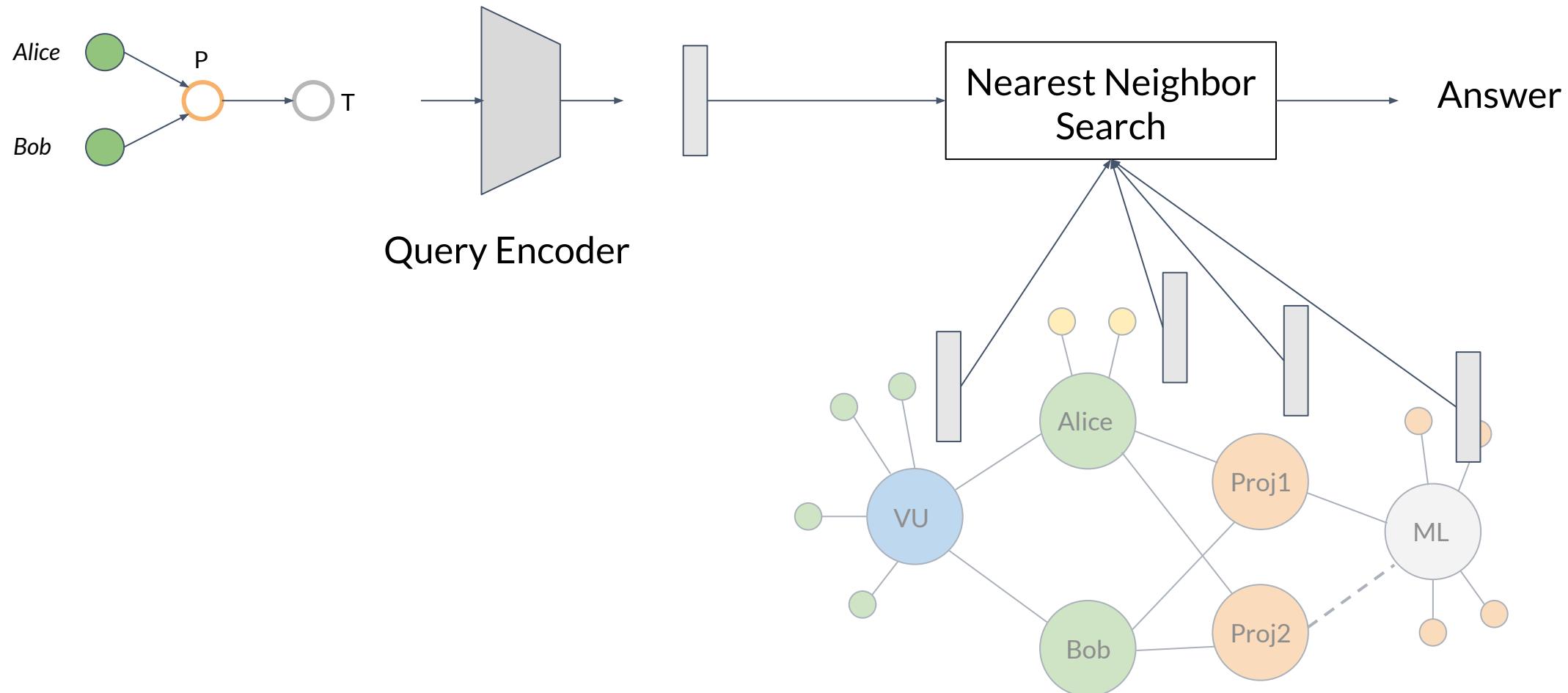


Embedding Queries



Daza and Cochez (2020), “Message Passing Query Embedding”

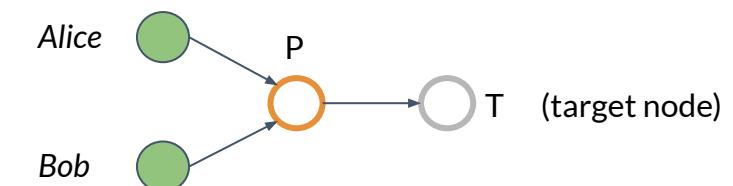
Embedding Queries



Daza and Cochez (2020), “Message Passing Query Embedding”

The query encoder

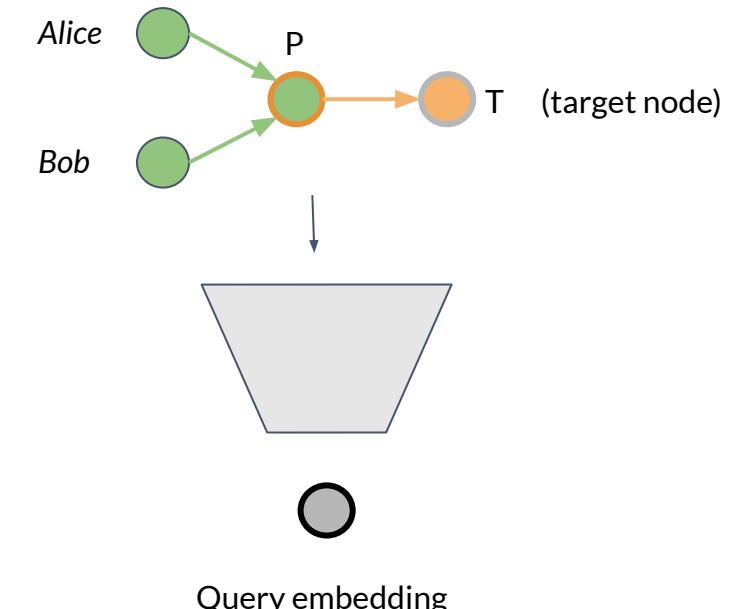
- Graph Convolutional Networks operate on graphs, by applying message passing:
 - **Messages are vectors**
- **Message-Passing Query Embedding:**
 - Learnable parameters include both **entity** and **variable** node embeddings
 - Propagate messages across the query graph



Daza and Cochez (2020), “Message Passing Query Embedding”

The query encoder

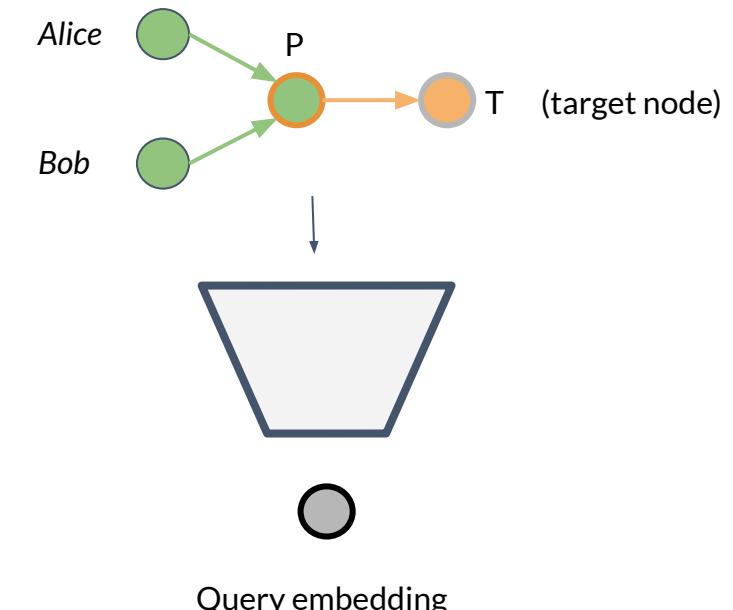
- Graph Convolutional Networks operate on graphs, by applying message passing:
 - **Messages are vectors**
- **Message-Passing Query Embedding:**
 - Learnable parameters include both **entity** and **variable** node embeddings
 - Propagate messages across the query graph
 - After k steps of MP, map all node messages to a single query embedding



Daza and Cochez (2020), “Message Passing Query Embedding”

Aggregation functions

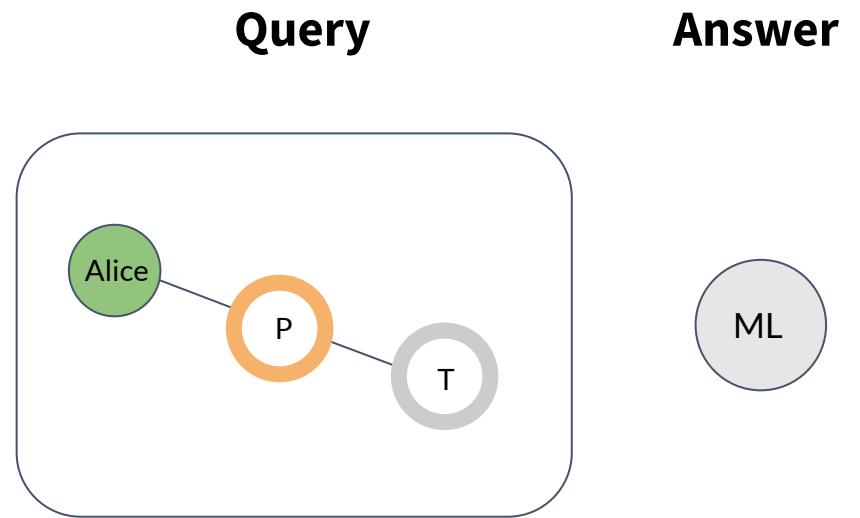
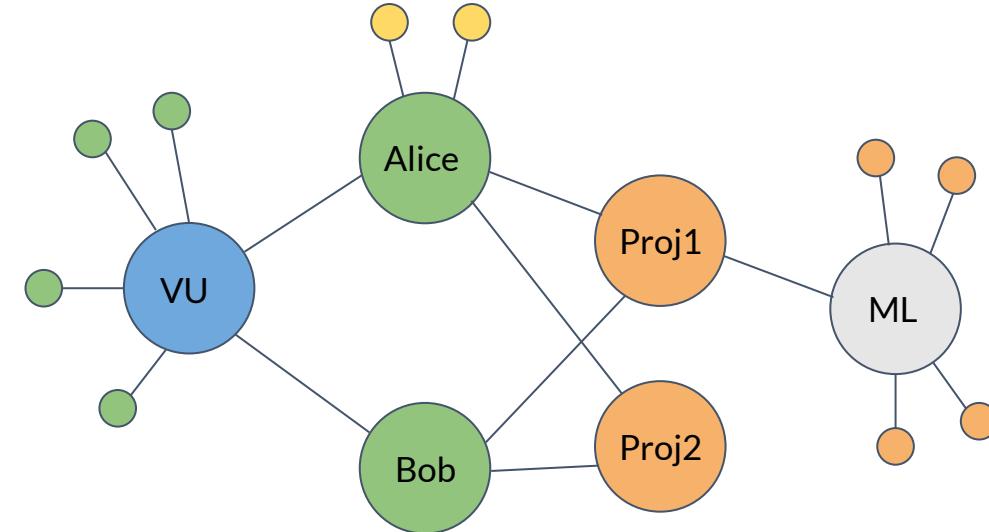
- Map node messages to query embedding
- Ideally permutation invariant
- Can contain learnable parameters for increased flexibility
- Simplest form: message at the target node



Daza and Cochez (2020), “Message Passing Query Embedding”

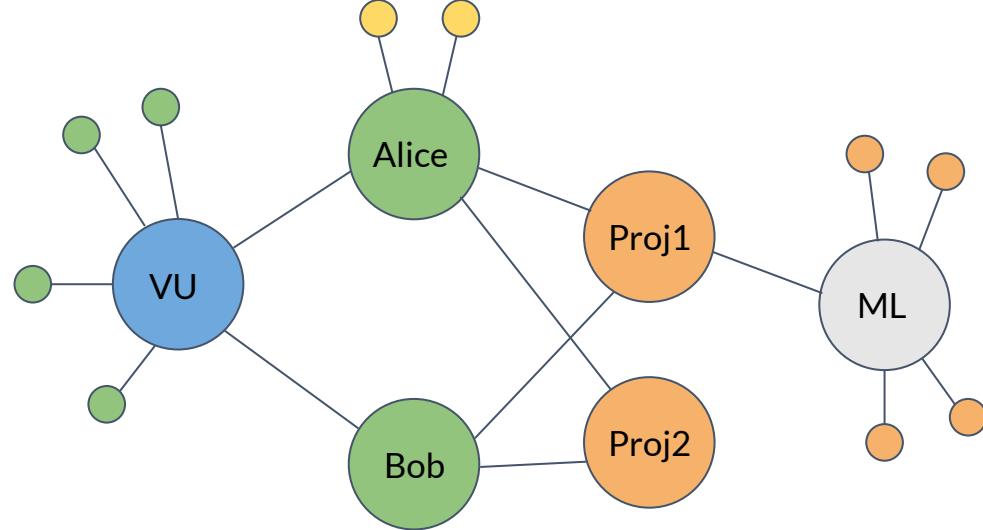
Training

- Queries obtained from KG:
 - Sample subgraphs
 - Replace some entities by variables

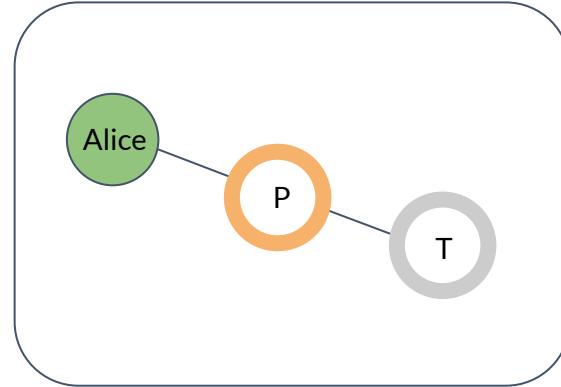


Training

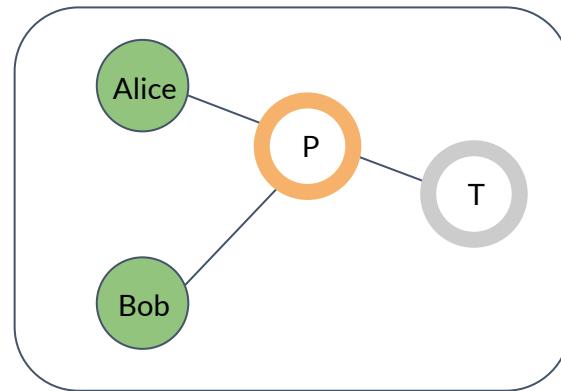
- Queries obtained from KG:
 - Sample subgraphs
 - Replace some entities by variables



Query

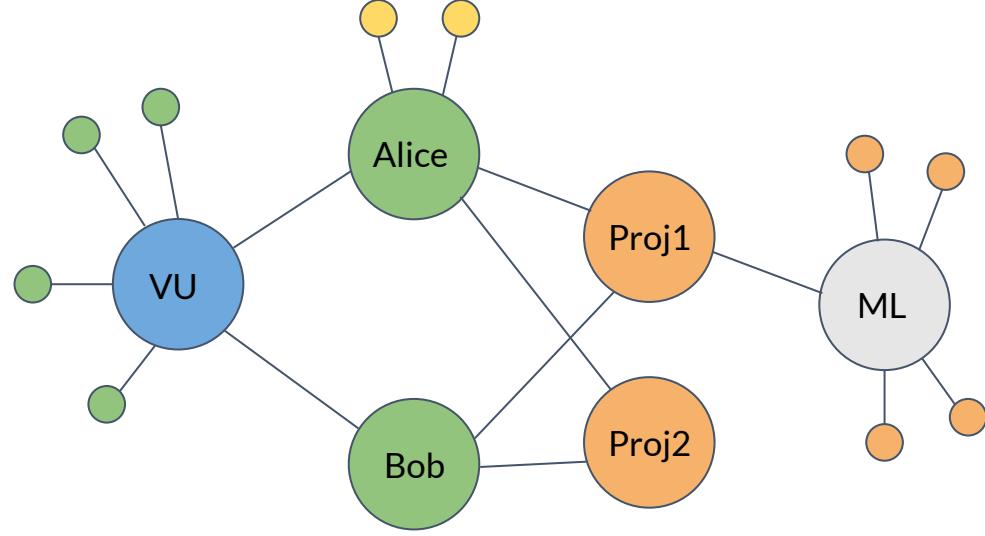


Answer

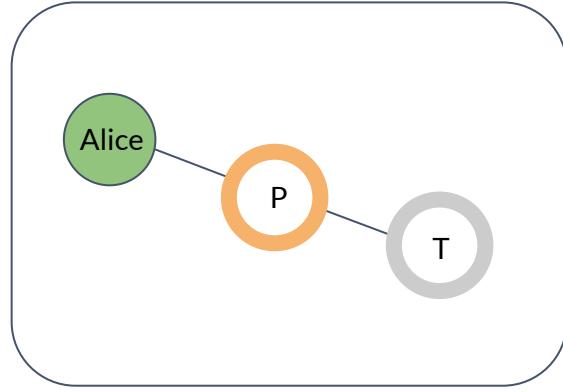


Training

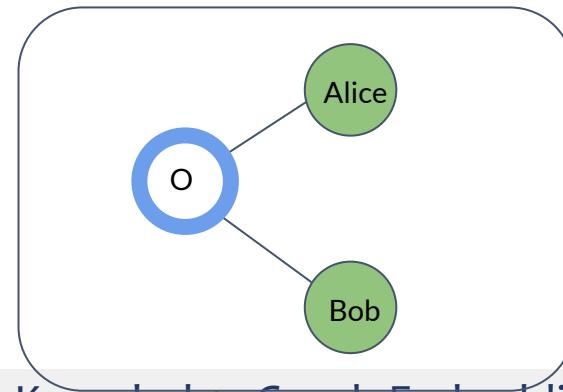
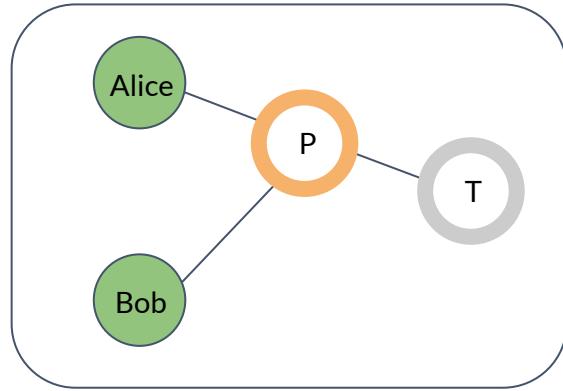
- Queries obtained from KG:
 - Sample subgraphs
 - Replace some entities by variables



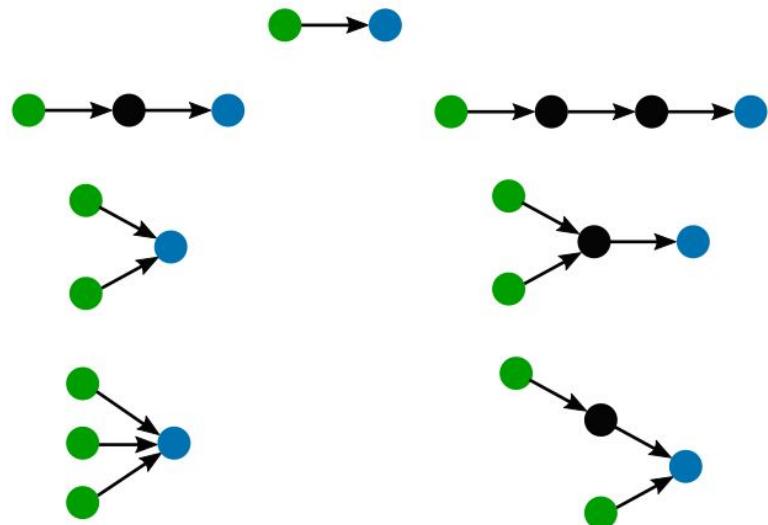
Query



Answer



Evaluation



Models often evaluated on **7 query structures**

Given query answer-pairs, evaluate using

- Mean reciprocal rank
- Hits @ k

Neural Query Embedding

Pros

- Query is embedded into a single vector, query answering becomes $O(n)$
- An arbitrary set of queries can be encoded
- Scales well since only small graphs are encoded at a time

Cons

- Query encoder is a black box
- Requires training with a large amount of query-answer pairs

Complex Query Answering

Neural Query
Embedding

Query
Decomposition

Complex Query Decomposition

- Train a simple link predictor (e.g. ComplEx)

Complex Query Decomposition

- Train a simple link predictor (e.g. ComplEx)

$$D : \exists P. \text{interacts}(D, P) \wedge \text{assoc}(P, t)$$

Complex Query Decomposition

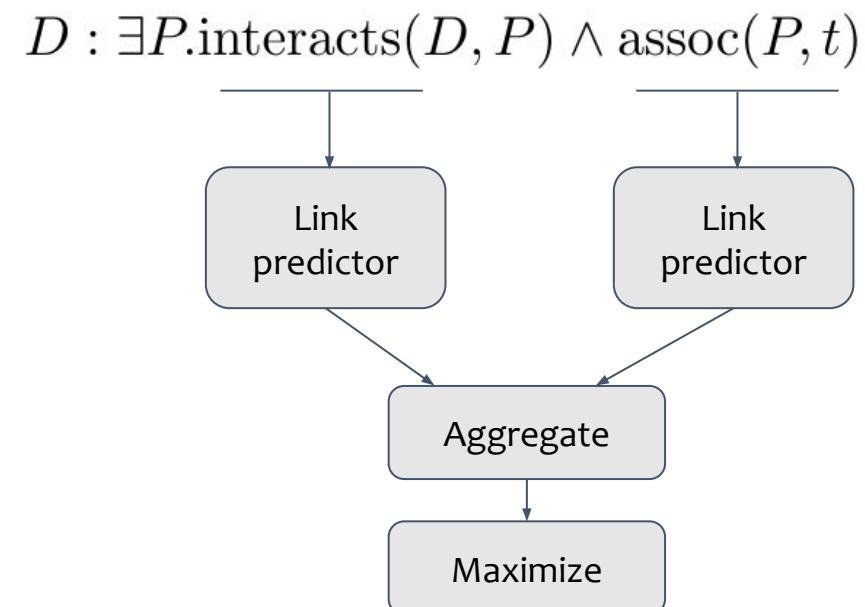
- Train a simple link predictor (e.g. **ComplEx**)
- Answer complex queries by breaking them down into simple ones, and **reusing** the link predictor

$$D : \exists P. \text{interacts}(D, P) \wedge \text{assoc}(P, t)$$



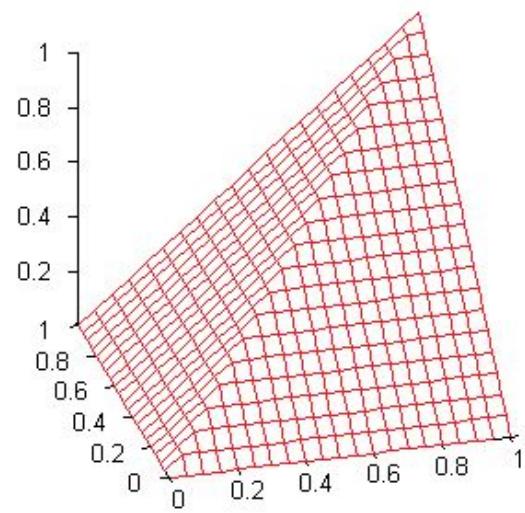
Complex Query Decomposition

- Train a simple link predictor (e.g. **ComplEx**)
- Answer complex queries by breaking them down into simple ones, and **reusing** the link predictor
- Aggregate individual results
- Maximize aggregated score



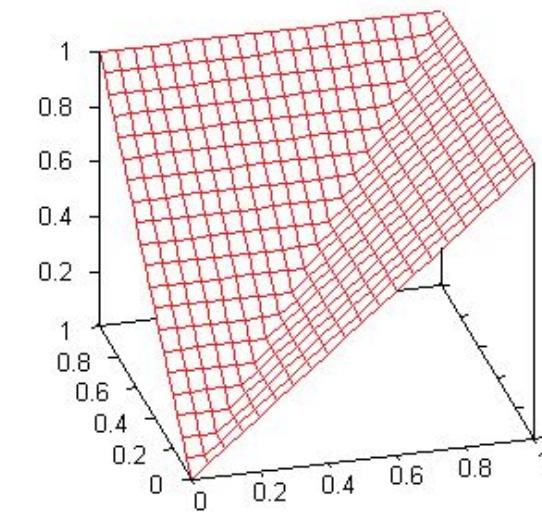
T-norms and T-conorms

Min t-norm



$$A \wedge B \rightsquigarrow \top_{\min}(a, b) = \min(a, b)$$

Max t-conorm



$$A \vee B \rightsquigarrow \perp_{\max}(a, b) = \max(a, b)$$

Query answering as optimization

$$D : \exists P. \text{interacts}(D, P) \wedge \text{assoc}(P, t) \rightsquigarrow \arg \max_{\mathbf{e}_D, \mathbf{e}_P} \phi_{\text{assoc}}(\mathbf{e}_P, \mathbf{e}_t) \top \phi_{\text{inter}}(\mathbf{e}_D, \mathbf{e}_P)$$

Query answering as optimization

$$D : \exists P.\text{interacts}(D, P) \wedge \text{assoc}(P, t) \rightsquigarrow \arg \max_{\mathbf{e}_D, \mathbf{e}_P} \phi_{\text{assoc}}(\mathbf{e}_P, \mathbf{e}_t) \top \phi_{\text{inter}}(\mathbf{e}_D, \mathbf{e}_P)$$

- **Continuous:** Initialize $\mathbf{e}_D, \mathbf{e}_P$ randomly, maximize with gradient descent
- **Combinatorial:** For each predicate, pick top-k entity embeddings that maximize score

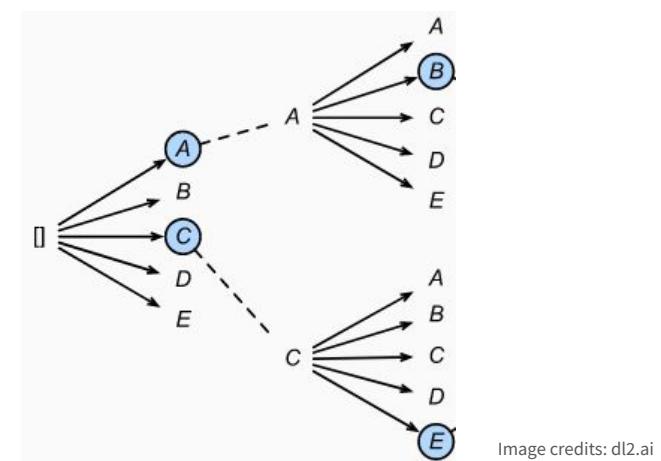
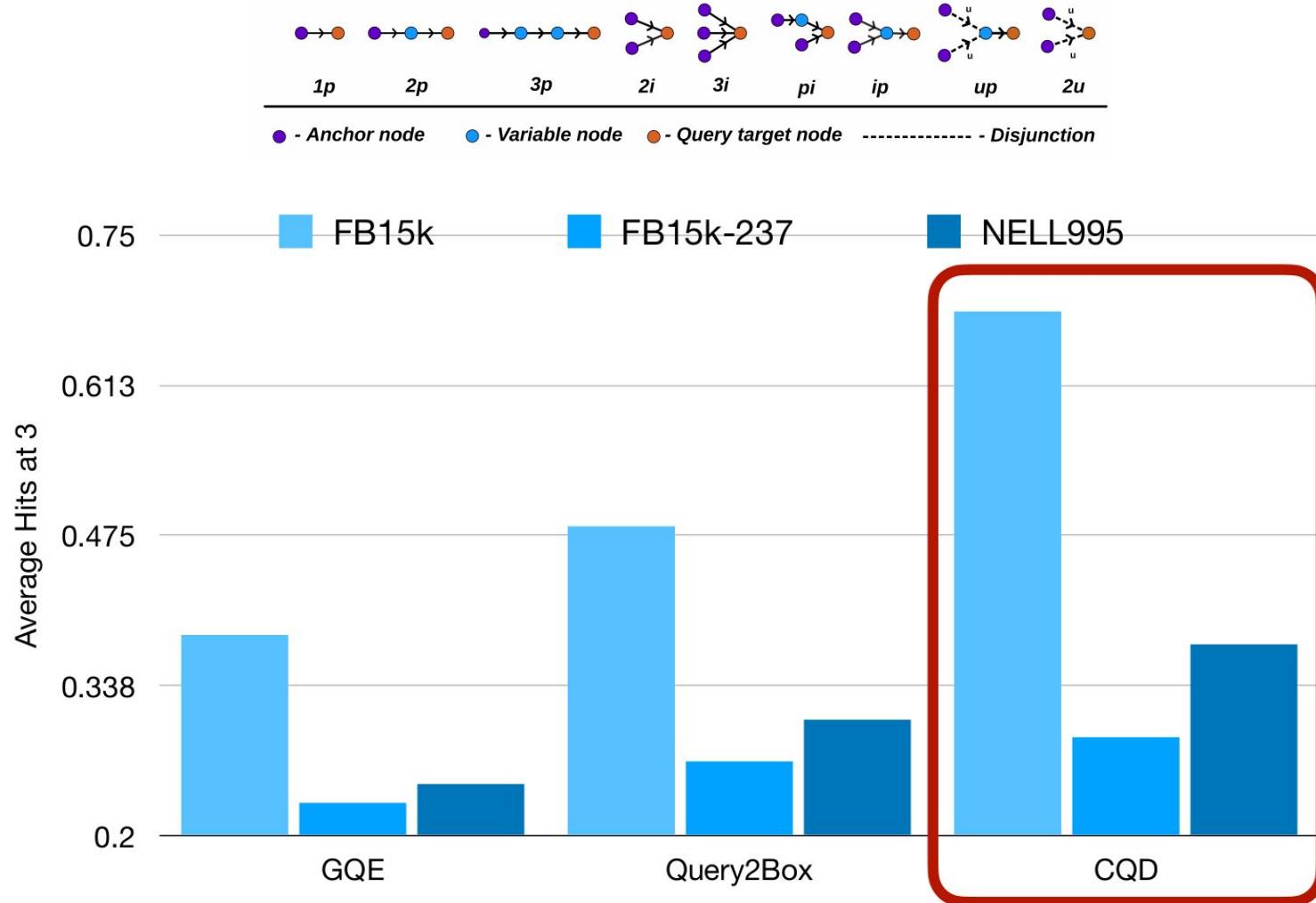


Image credits: dl2.ai

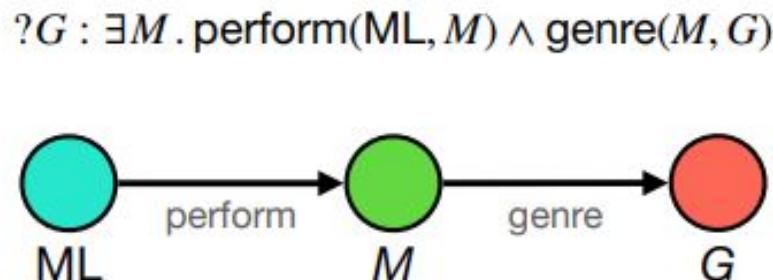
Results



CQD^A adds the ability to deal with negation in the queries.
<https://arxiv.org/abs/2301.12313v3>

Inspecting generated answers

"In what genres of movies did Martin Lawrence appear?"

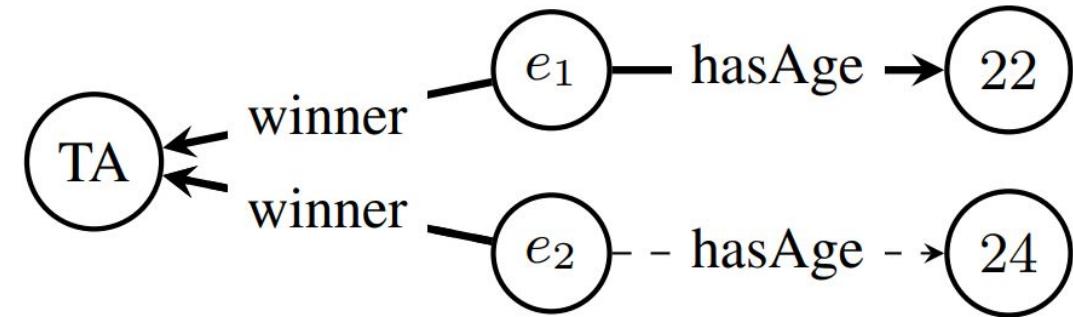


Query: $\text{?}G : \exists M . \text{perform(ML, } M) \wedge \text{genre}(M, } G)$

M	G	Rank	Correctness
Do the Right Thing	Drama	1	✓
	Comedy	4	✓
	Crime Fiction	7	✓
National Security	Action	2	✓
	Thriller	3	✓
	Crime Fiction	5	✓
The Nutty Professor	Comedy	6	✓
	Romantic Com.	8	✗
	Romance Film	9	✗

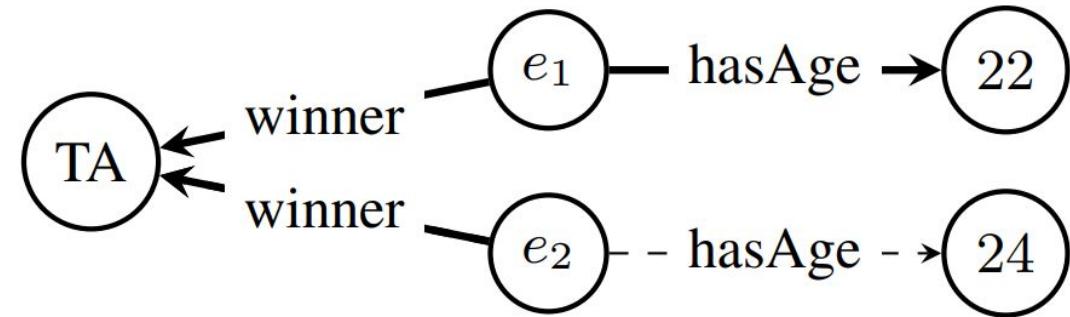
Complex Queries with literals

Knowledge graph with literals

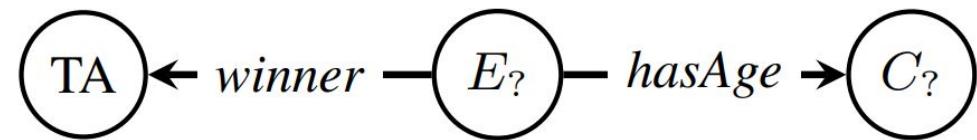


Complex Queries with literals

Knowledge graph with literals



Complex query with literals

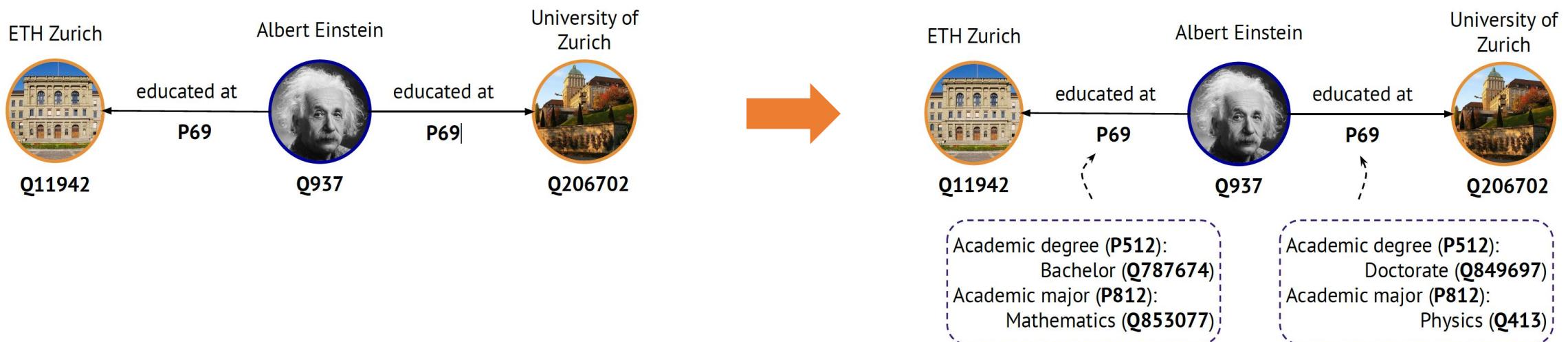


Complex Queries with literals

Method	Average	1p	2p	3p	2i	3i	ip	pi	2u	up
MRR										
Query2Box	0.213	0.403	0.198	0.134	0.238	0.332	0.107	0.158	0.195	0.153
CQD	0.295	0.454	0.275	0.197	0.339	0.457	0.188	0.267	0.261	0.214
LitCQD (ours)	0.301	0.457	0.285	0.202	0.350	0.466	0.193	0.274	0.266	0.215

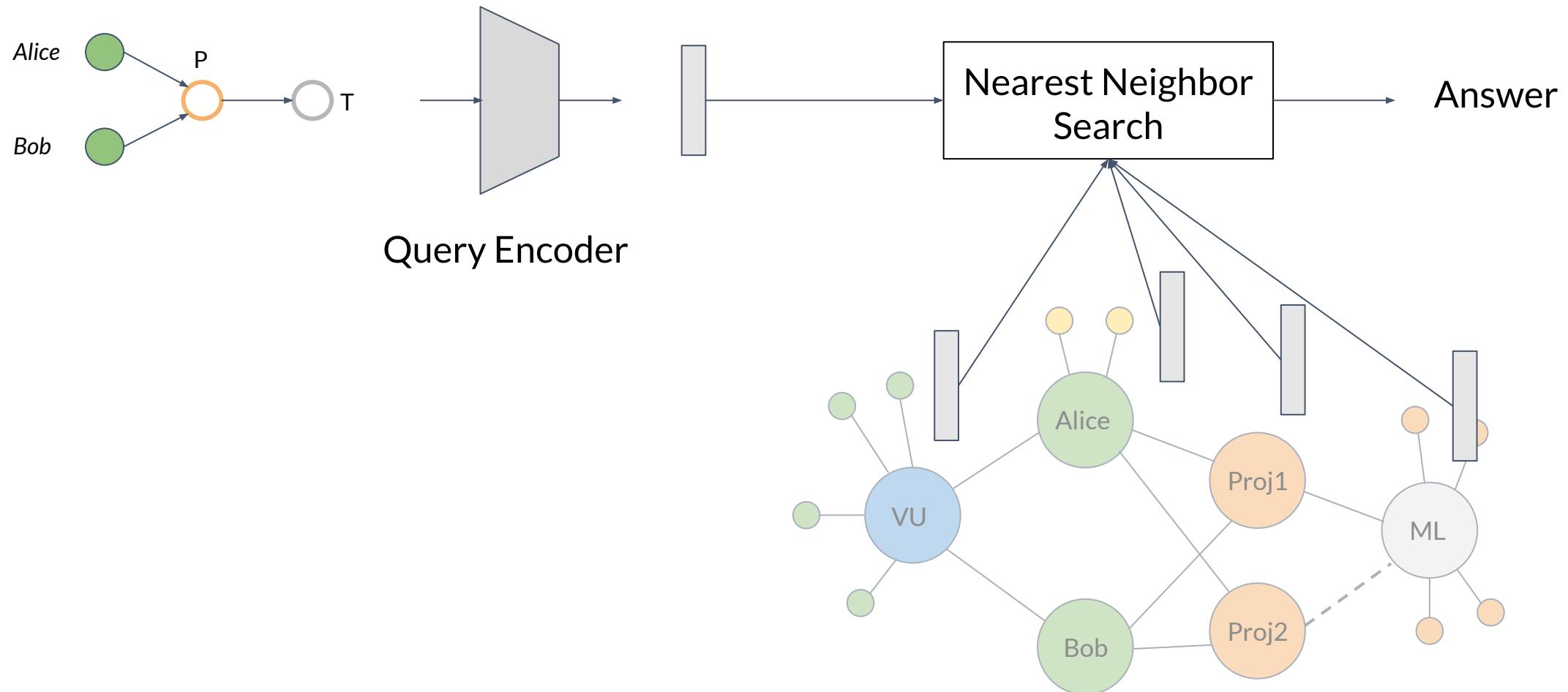
- Training with literals beneficial for queries without them
- Bonus: we can **predict literal values** in complex queries

Complex Queries with Qualifiers



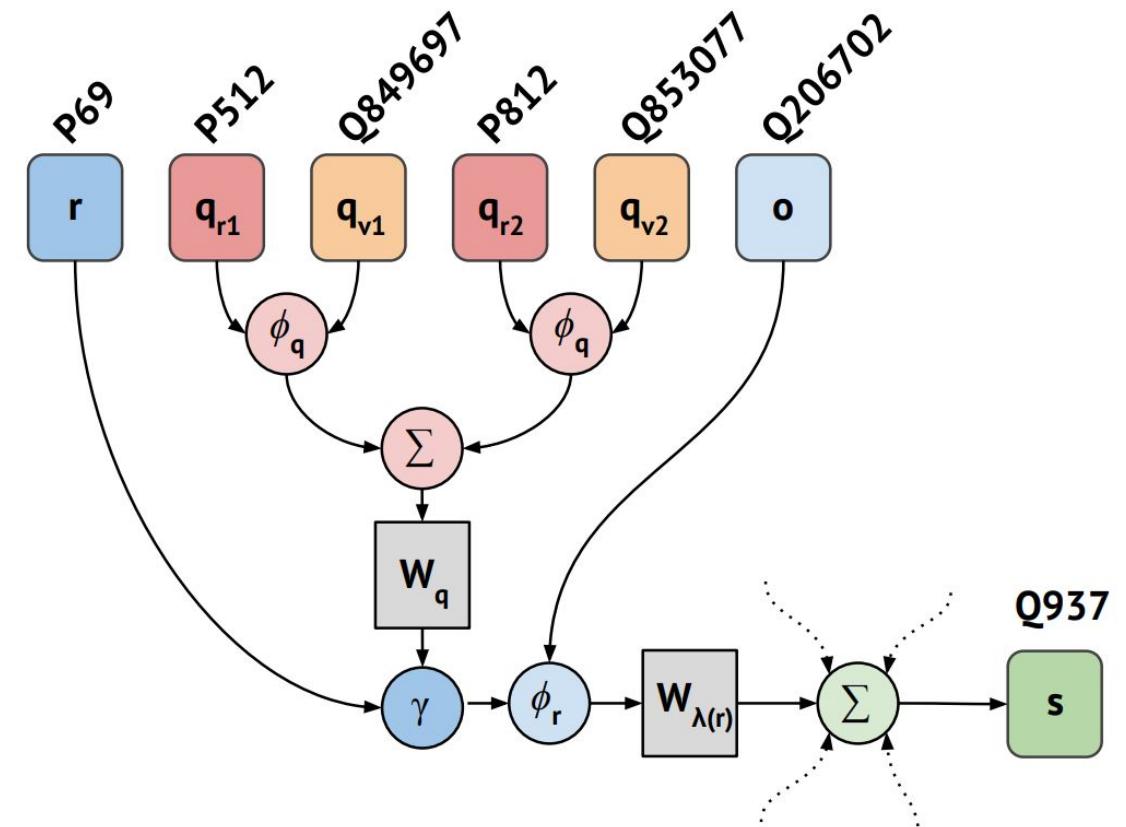
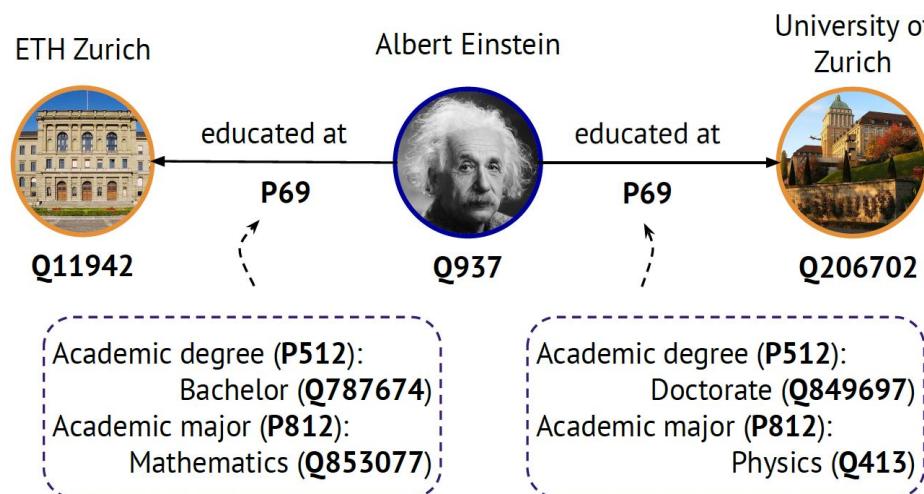
Galkin et al. (2020), “Message Passing for Hyper-Relational Knowledge Graphs”

Embedding Queries



Daza and Cochez (2020), "Message Passing Query Embedding"

Encoder for queries with qualifiers



Conclusion

- Increasing interest in transition from **simple to complex queries** in KGs.
- Two main approaches:
 - Neural query encoders
 - Decomposition methods
- These approaches extend to KGs beyond triples:
 - KGs with literals
 - Hyper-relational KGs

Conclusion

- Increasing interest in transition from **simple to complex queries** in KGs.
- Two main approaches:
 - Neural query encoders
 - Decomposition methods
- These approaches extend to KGs beyond triples:
 - KGs with literals
 - Hyper-relational KGs
- Limitations
 - Synthetic vs. real world queries
 - Evaluation protocol



Part 5. Advanced Topics

Neural Graph Databases

Agenda

Part 1. Introduction (Bo & Mojtaba, 9:00 - 9:30)

- 1.1 Tutorial Overview (Bo)
- 1.2 Introduction to KG Embeddings
- 1.3 Motivation to go beyond triple-based KGs

Part 2. Temporal KG Embeddings (Mojtaba, 9:30 – 10:30)

- 2.1 Functional models
 - 2.2 Factorization models
 - 2.3 Neural embedding models
- Coffee Break (10:30 – 11:00)*

Part 3. Hyper-relational & N-ary KG embeddings (Bo, 11:00 – 12:30)

- 3.1 Introduction to hyper-relational & N-Ary KGs
 - 3.2 Functional embedding models
 - 3.2 Graph neural networks models
- Lunch Break (12:30 – 13:30)*

Part 4. KG embeddings with literal and text (Daniel, 14:00 – 15:00)

- 4.1 Incorporating entity descriptions
- 4.2 Incorporate numeric literals as features

Part 5. Advanced Topic (Michael, 15:00-15:30 & 16:00 – 16:30)

- 5.1 Complex query embeddings (Daniel)
- Coffee Break (15:30 – 16:00)*
- 5.2 Neural Graph Databases
- 5.3 Inductive learning settings
- 5.4 Incorporating ontologies and LLMs

Part 6. Conclusion & Future Works (Michael, 16:30 – 17:00)

- 6.1 Conclusion
- 6.2 Future Works

Neural Graph Reasoning: Complex Logical Query Answering Meets Graph Databases

Hongyu Ren^{*4}Mikhail Galkin^{*1}Michael Cochez⁵Zhaocheng Zhu^{2,3}Jure Leskovec⁴

¹Intel AI Lab, ²Mila, ³Université de Montréal, ⁴Stanford University, ⁵VU Amsterdam

<https://arxiv.org/abs/2303.14617>

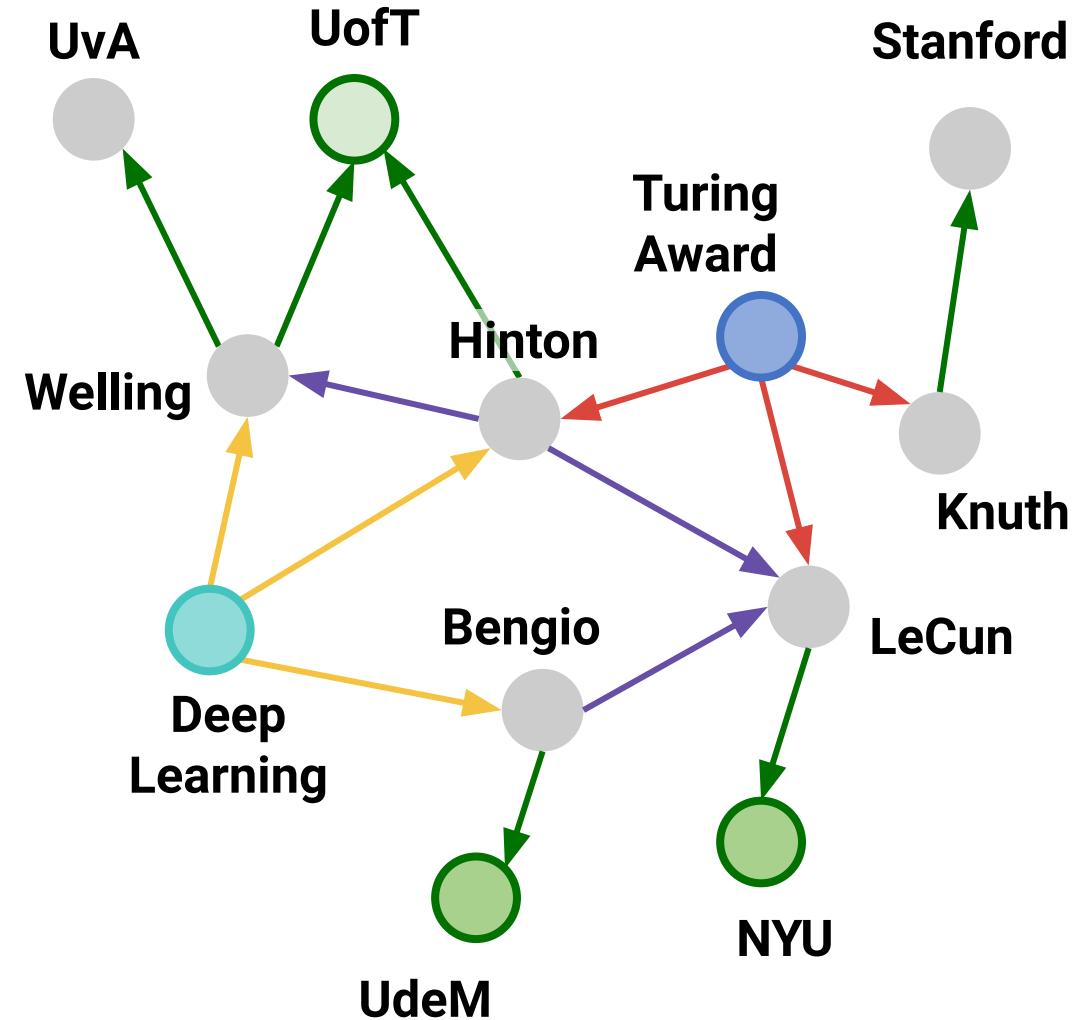
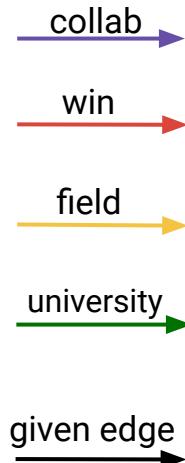
Our Graph

Multi-relational graphs

Modeled as:

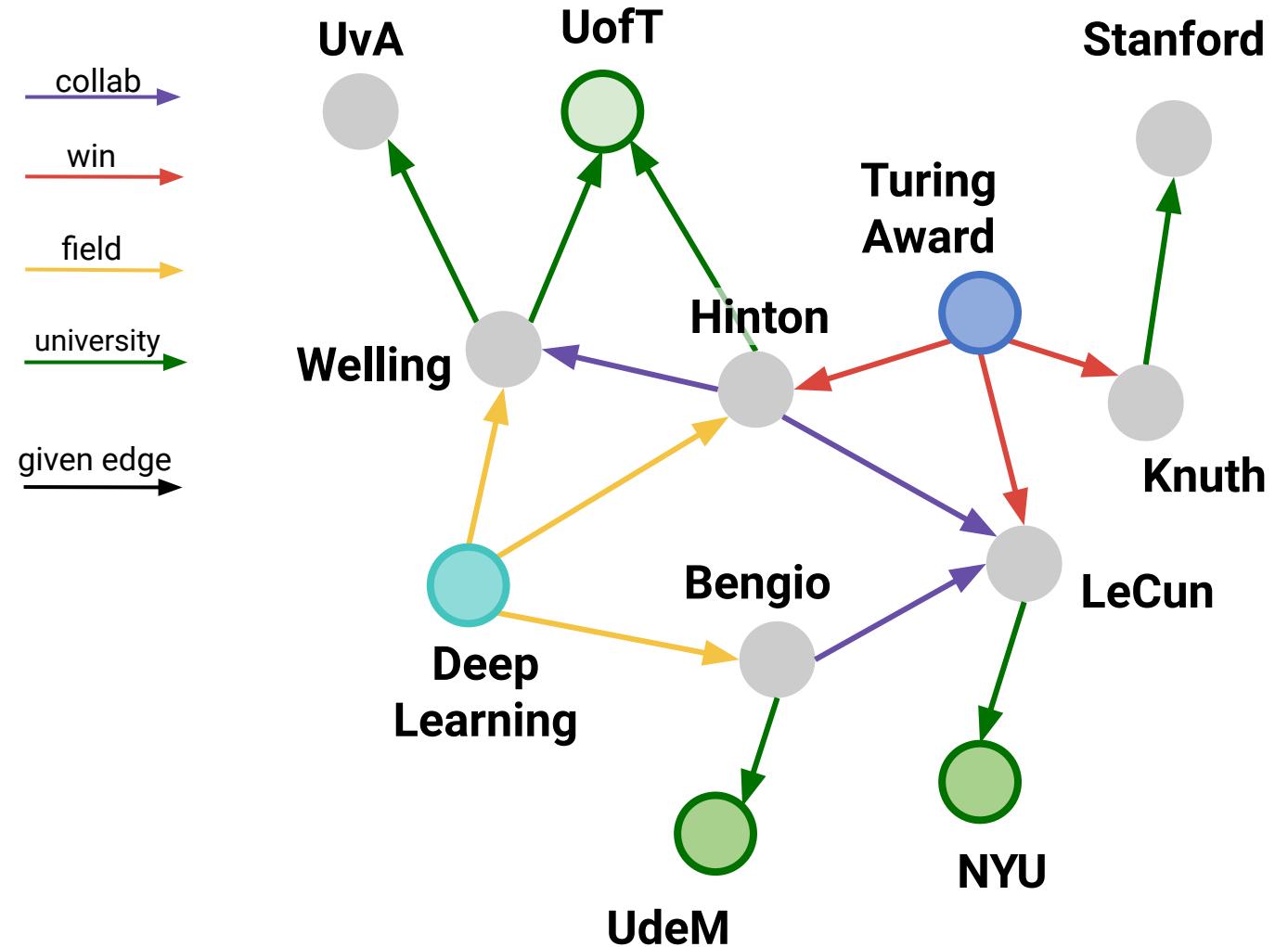
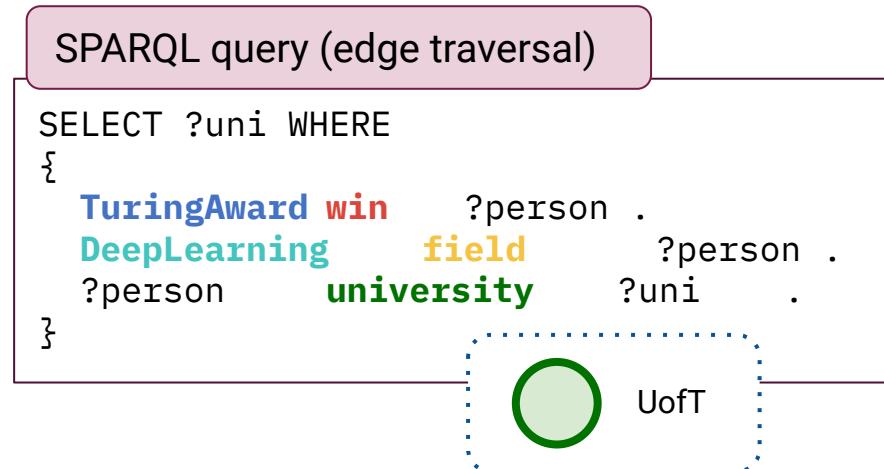
- RDF graphs
- Labeled Property graphs (LPG)
- RDF* (RDF-star)

Stored in Graph DBs, queried with:
SPARQL / Cypher / Gremlin / etc

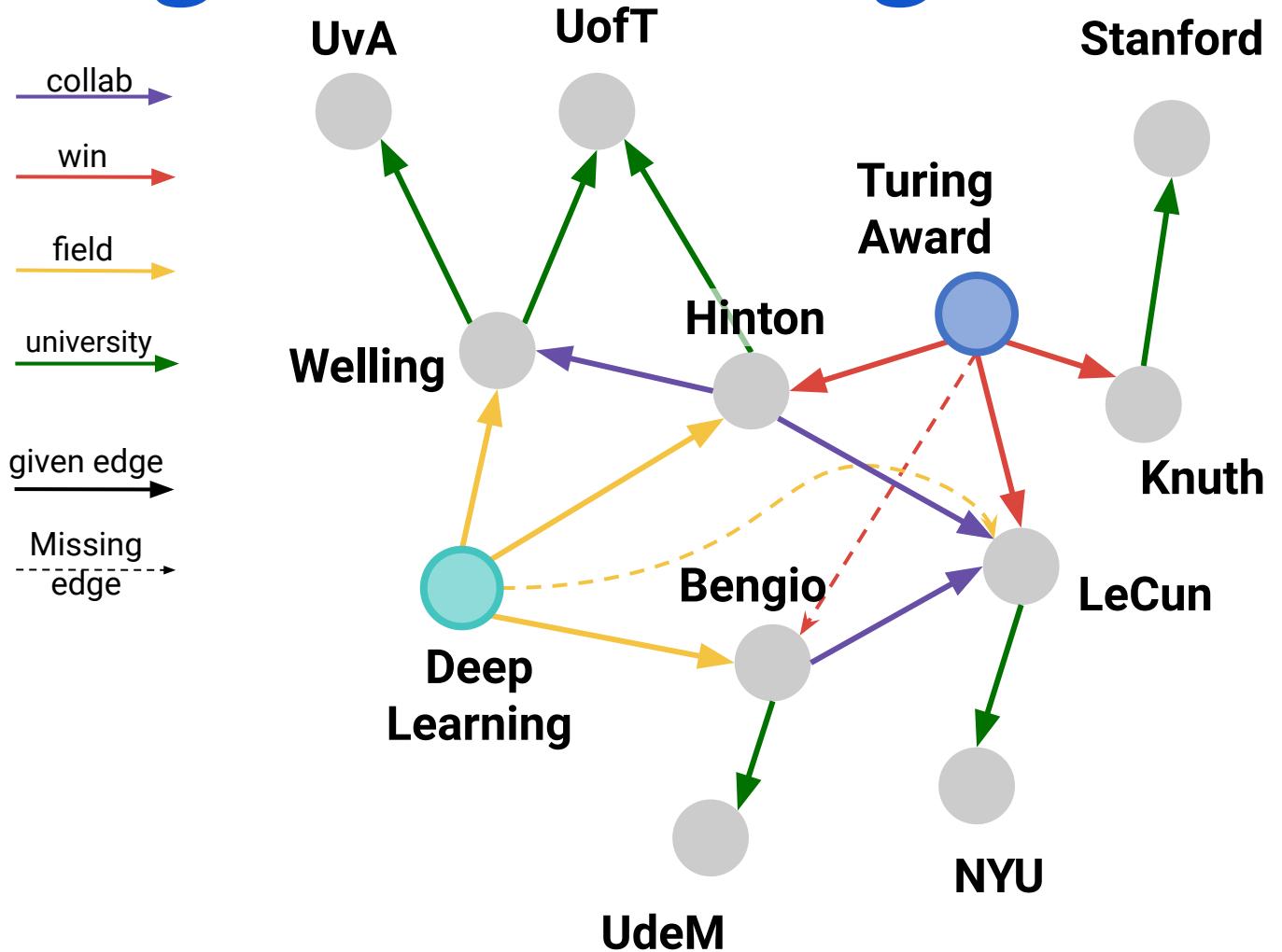


At what universities do the Turing Award winners in the field of Deep Learning work?

$q = U_? . \exists V : \text{win}(\text{TuringAward}, V) \wedge \text{field}(\text{DeepLearning}, V) \wedge \text{university}(V, U_?)$

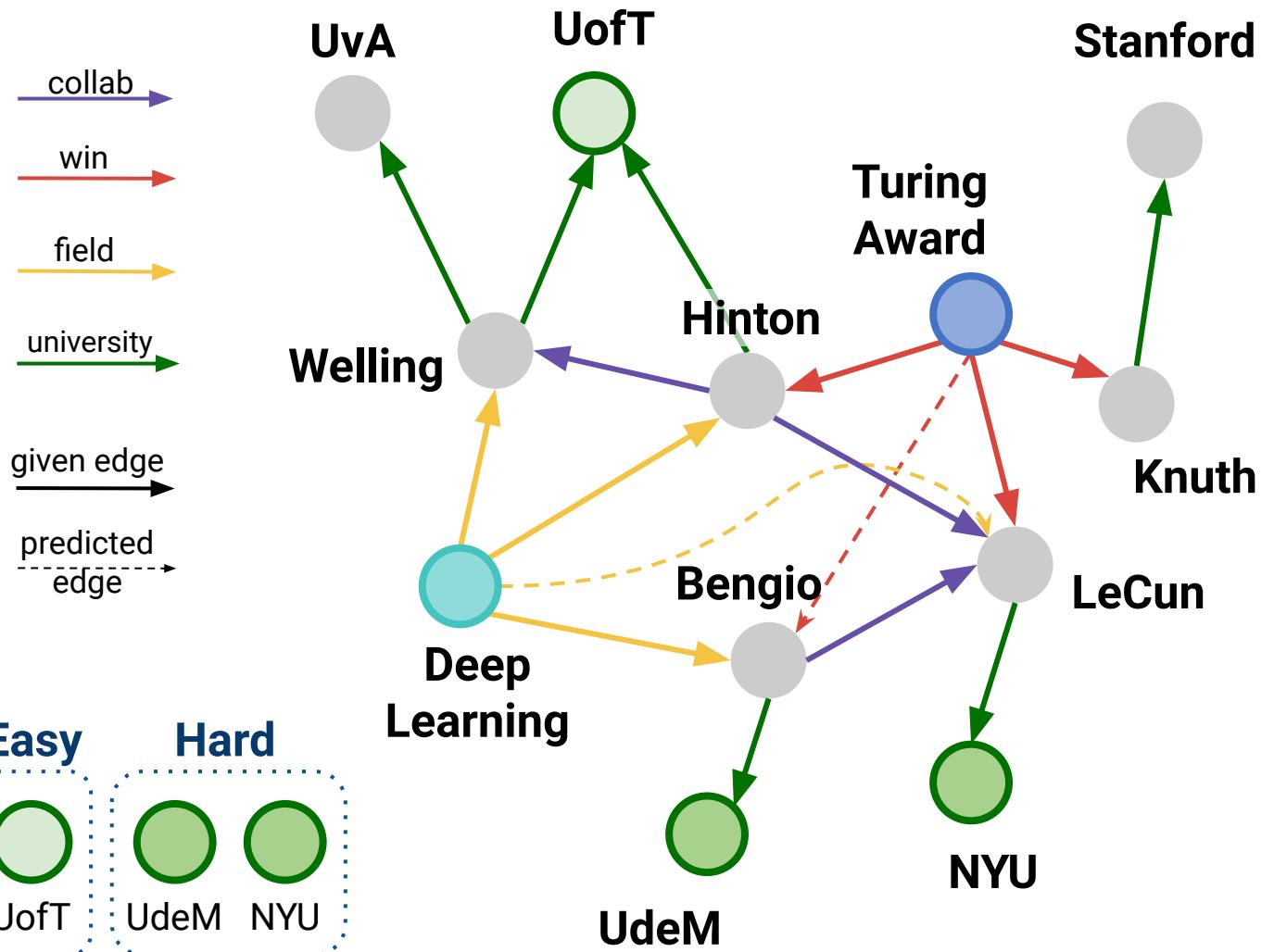
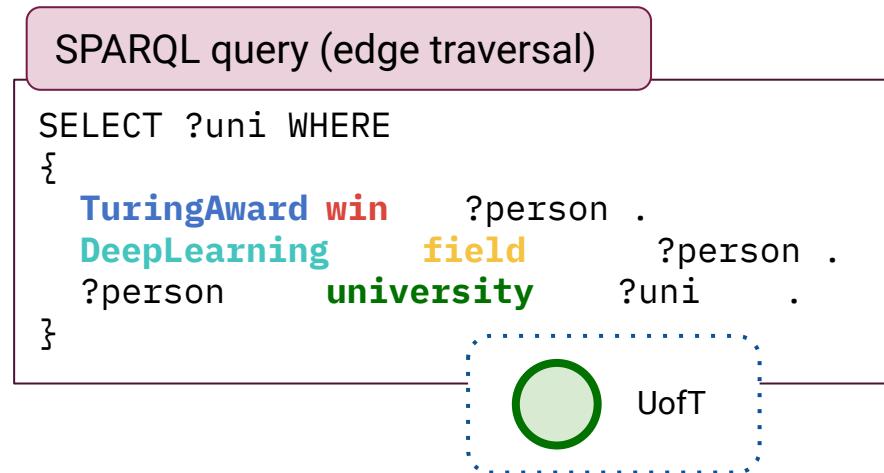


Problem: some edges are missing

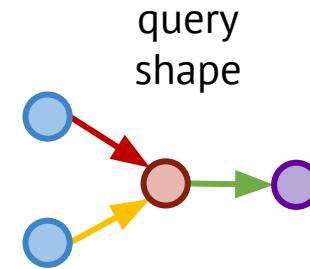


At what universities do the Turing Award winners in the field of Deep Learning work?

$q = U_? . \exists V : \text{win}(\text{TuringAward}, V) \wedge \text{field}(\text{DeepLearning}, V) \wedge \text{university}(V, U_?)$



Query Answering in KGs


$$q = \underline{U_?} . \exists \underline{V} : \underline{\text{win}(\text{TuringAward}, V)} \wedge \underline{\text{field}(\text{DeepLearning}, V)} \wedge \underline{\text{university}(V, U?)}$$

Variables

Constants

Projections R(a,b)

Logical Operators \wedge, \vee, \neg

U, V

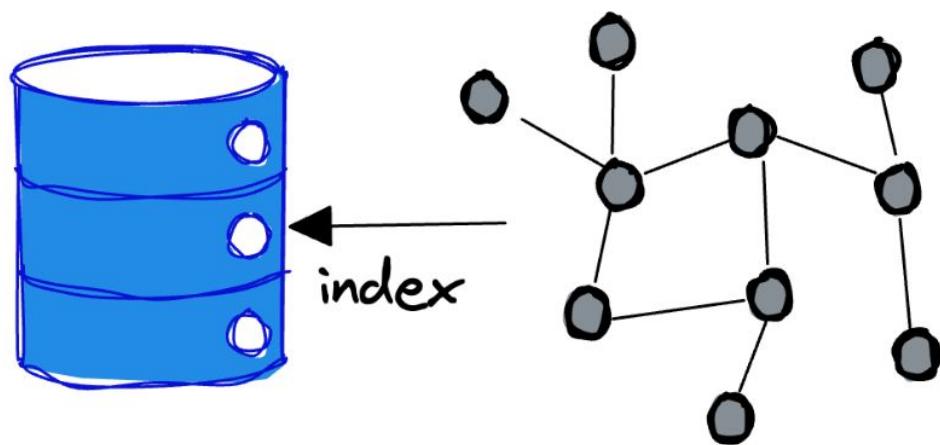
TuringAward
DeepLearning

win(a, b)
field(a, b)
university(a, b)

\wedge conjunctive queries
 \wedge, \vee EPFO
 \wedge, \vee, \neg EFO

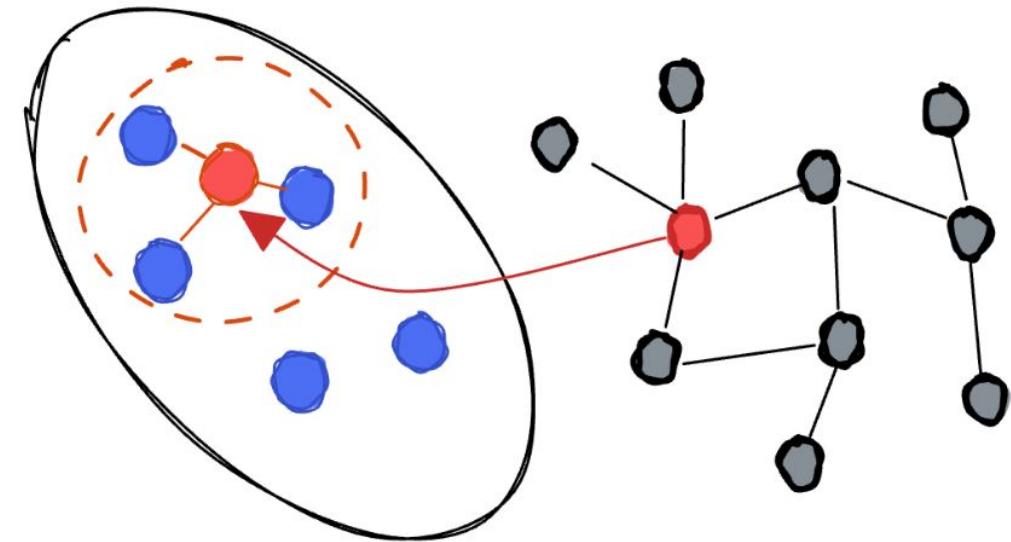
Can be parsed to a structured query format (eg, SPARQL)

Symbolic Graph Databases



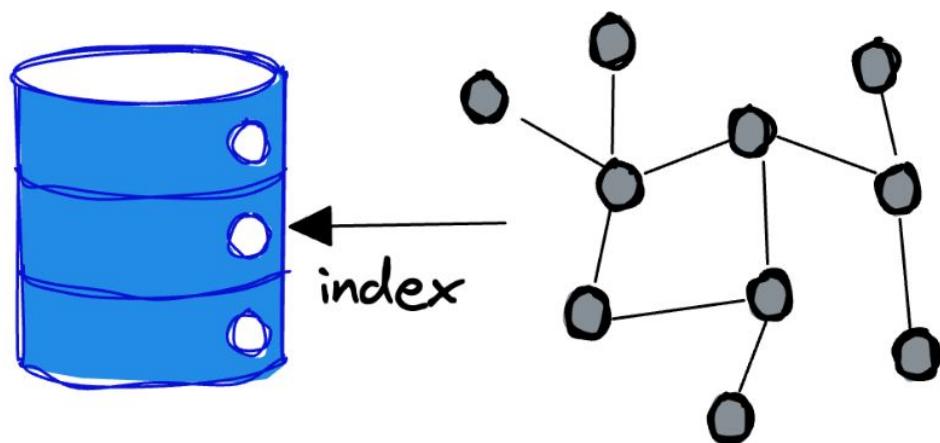
- What is there? **(traversal)**

Neural Graph Databases



- What is there? **(traversal)**
- What is missing? **(inference)**

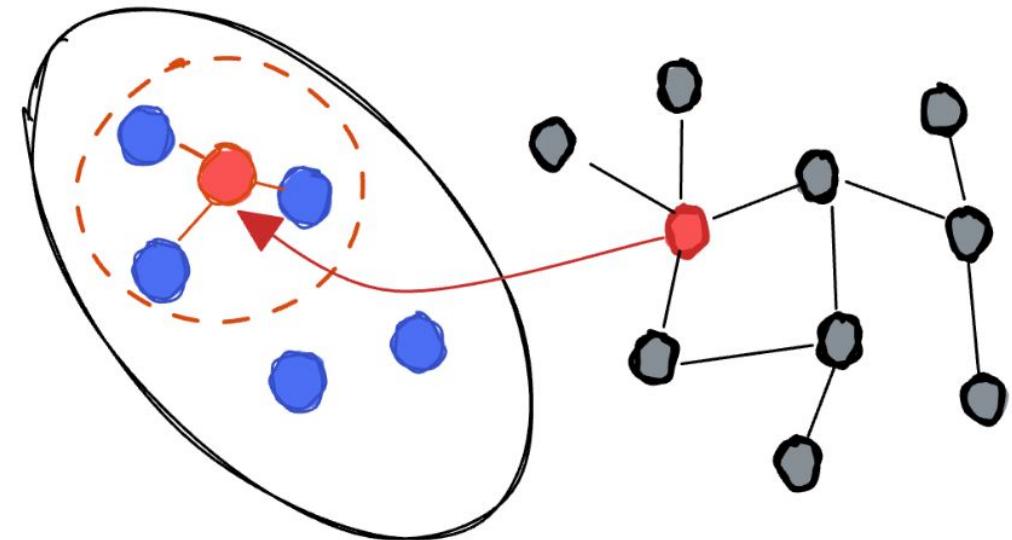
Symbolic Graph Databases



- What is there? **(traversal)**

Potentially infinite-time symbolic inference (SPARQL entailment regimes)

Neural Graph Databases



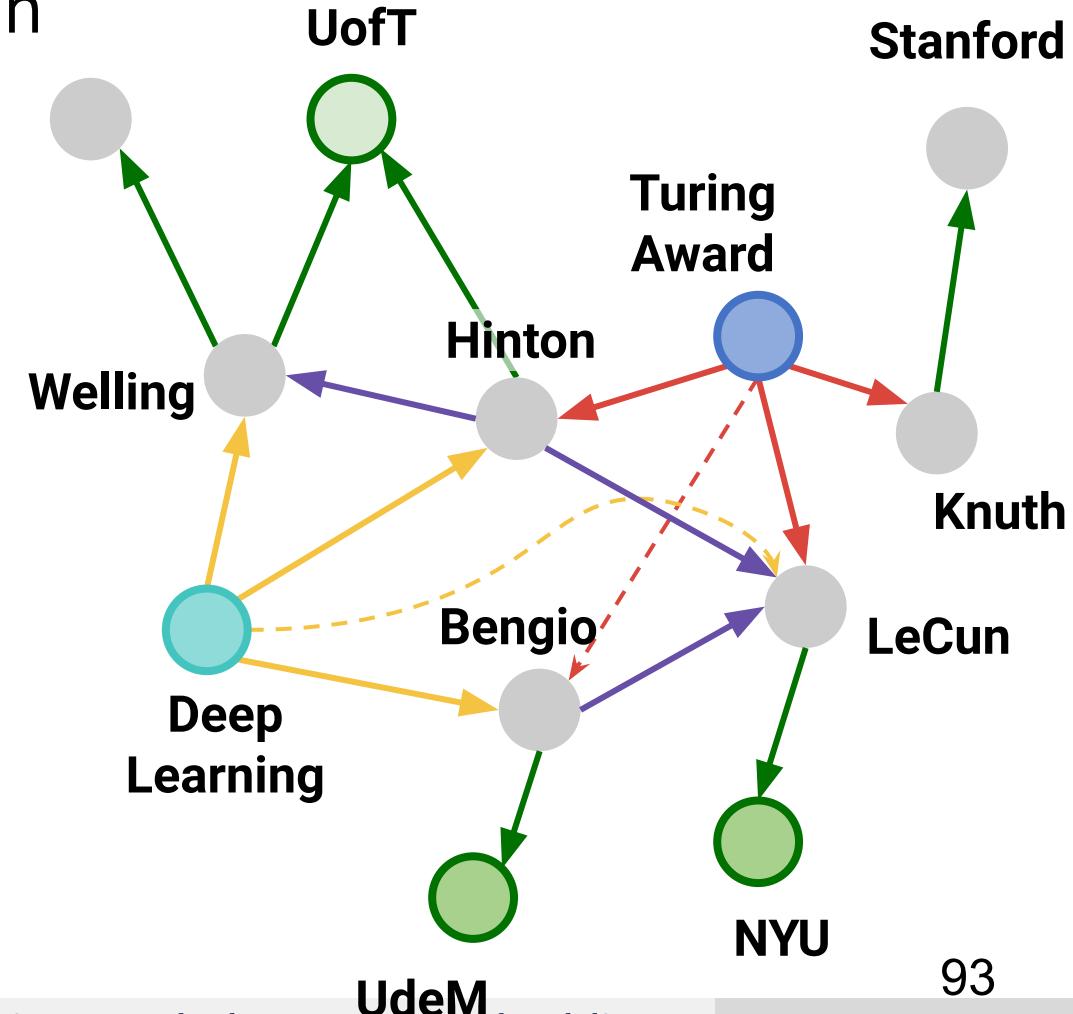
- What is there? **(traversal)**
- What is missing? **(inference)**

Graph Representation Learning!

Neural Graph DB Design Principles

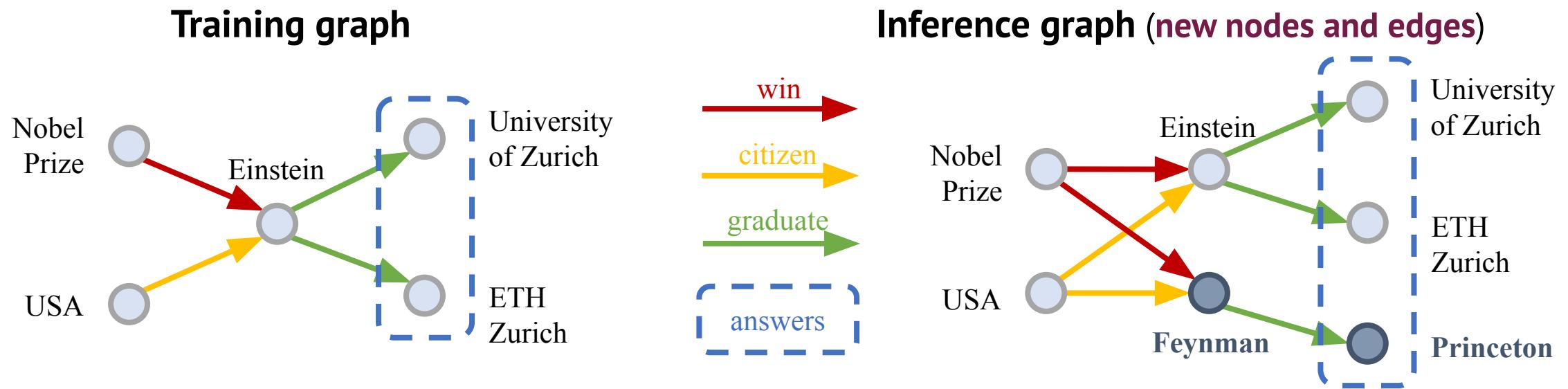
1. The data incompleteness assumption

Infer missing links on the fly
and enrich the answer set



Neural Graph DB Design Principles

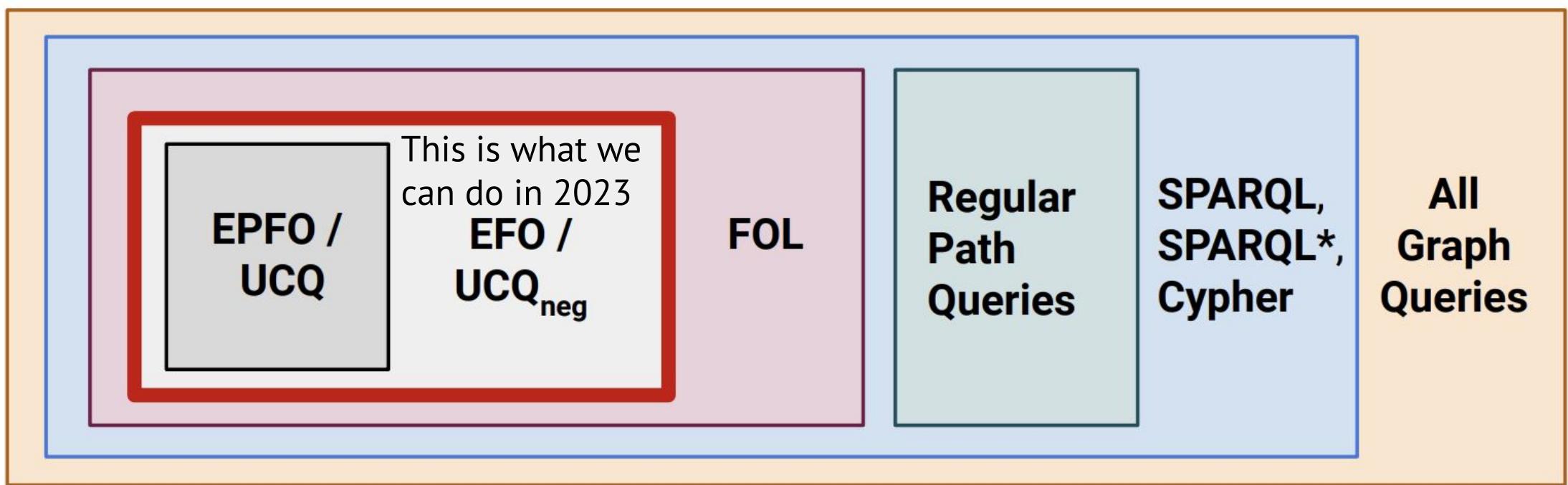
1. The data incompleteness assumption
2. Inductiveness and updatability



**Answer queries after updating the graph w/o retraining
graph representations**

Neural Graph DB Design Principles

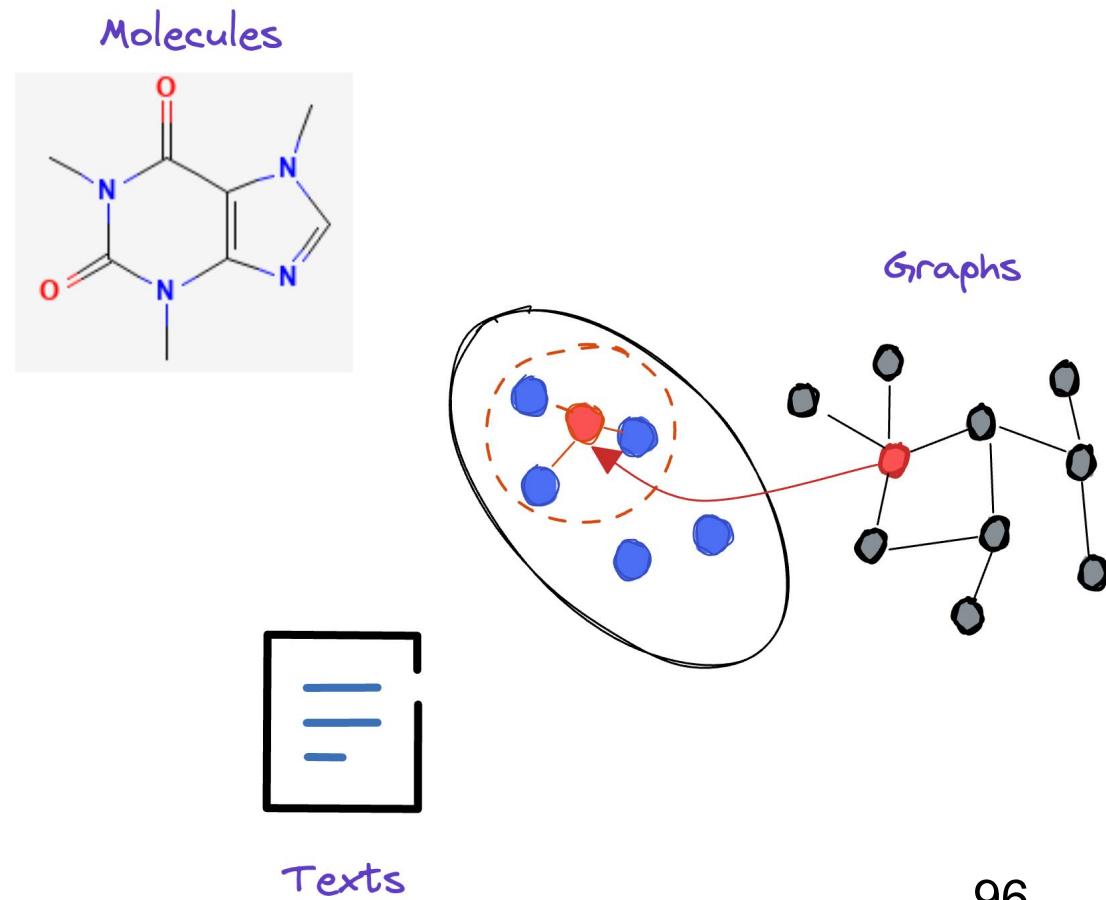
1. The data incompleteness assumption
2. Inductiveness and updatability
3. Expressiveness, at least match SPARQL



Neural Graph DB Design Principles

1. The data incompleteness assumption
2. Inductiveness and updatability
3. Expressiveness
4. Multimodality

Support different modalities to answer queries against



Neural Graph DB Design Principles

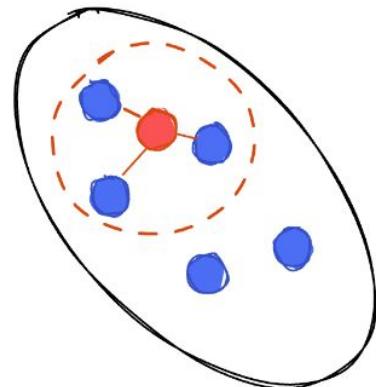
1. The data incompleteness assumption
2. Inductiveness and updatability
3. Expressiveness
4. Multimodality

The key methods:

- Vector representation as atomic element
- Neural query execution in the latent space

Neural Databases: Overview

Vector Databases



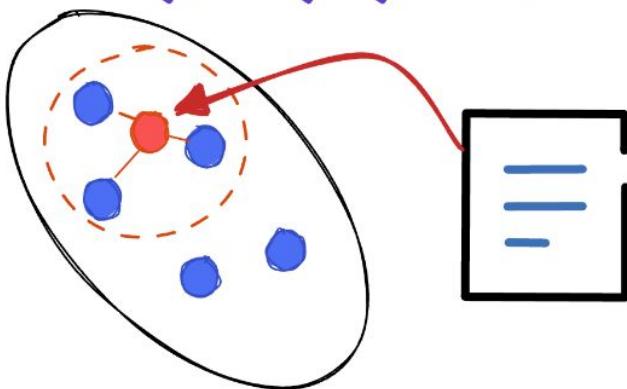
Content:

embeddings from
any neural net

Query:

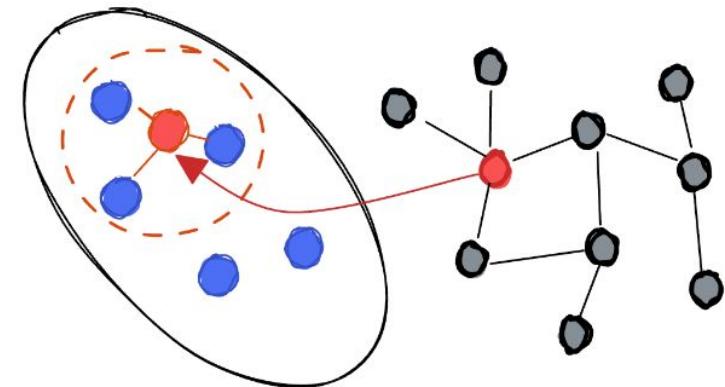
vector similarity
distance-based (MIPS, L1, L2)

Natural Language Databases
(Large Language Models)



document embeddings from LLM

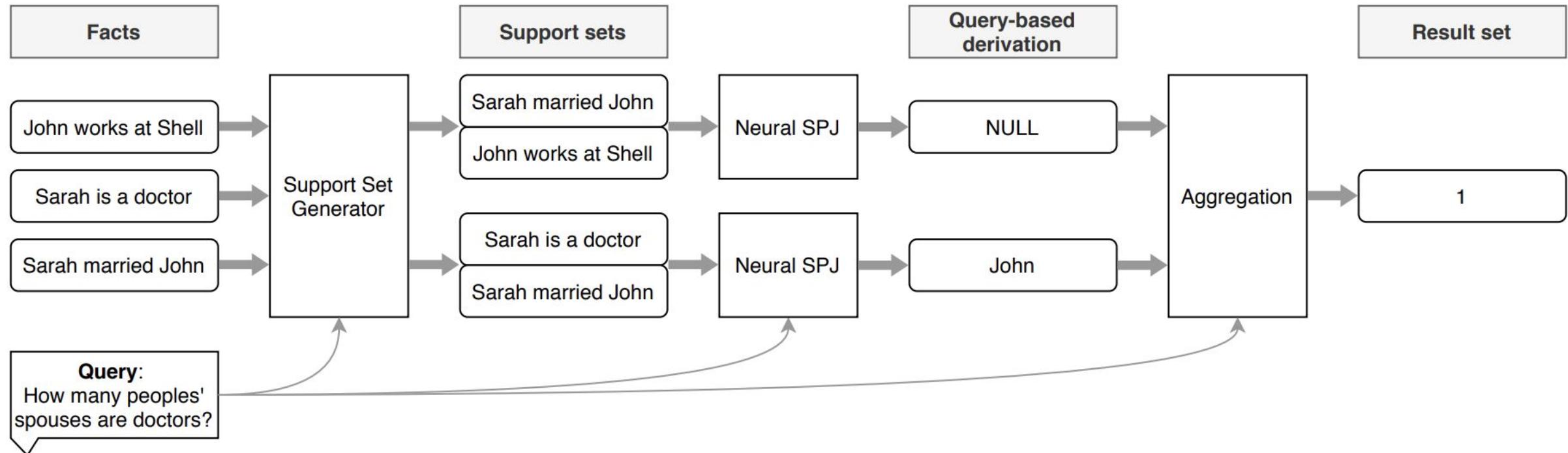
Neural Graph Databases



graph latents
node- / link- / subgraph- embeddings

logical queries
graph-shape queries

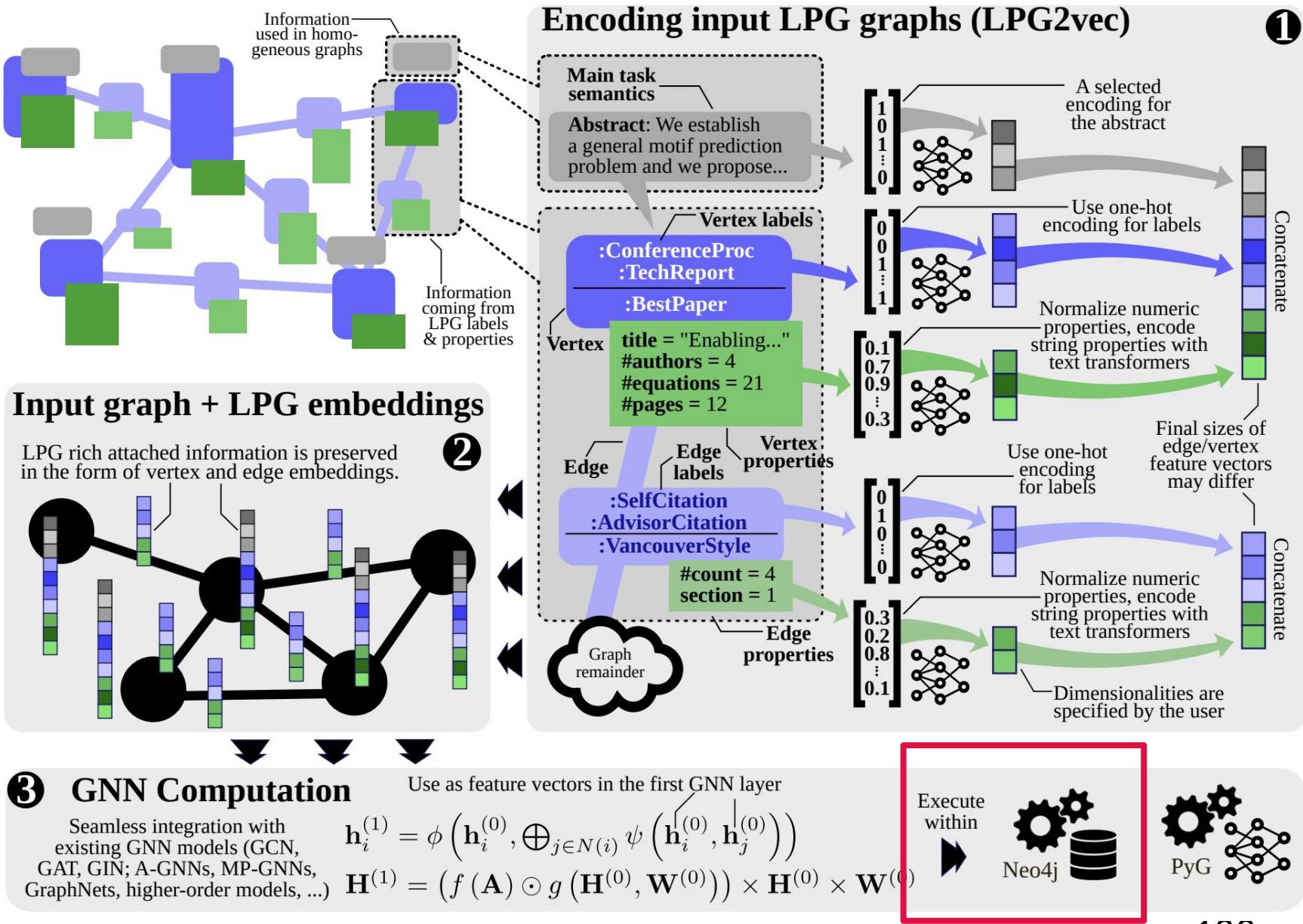
Natural Language Databases



Retriever-Reader paradigm

LPG 2 Vec

Query
execution
is still in
Neo4j

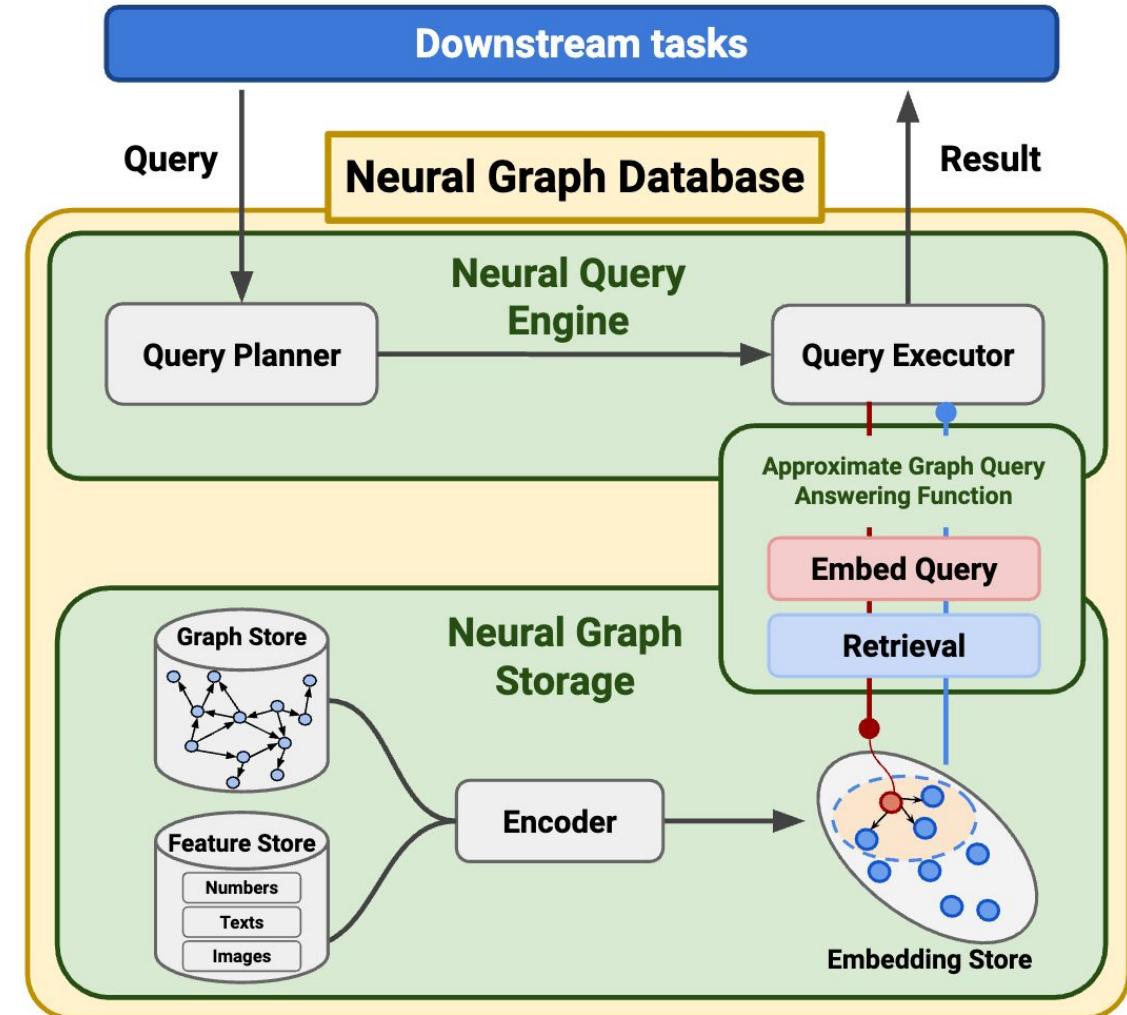


NGDB: Architecture

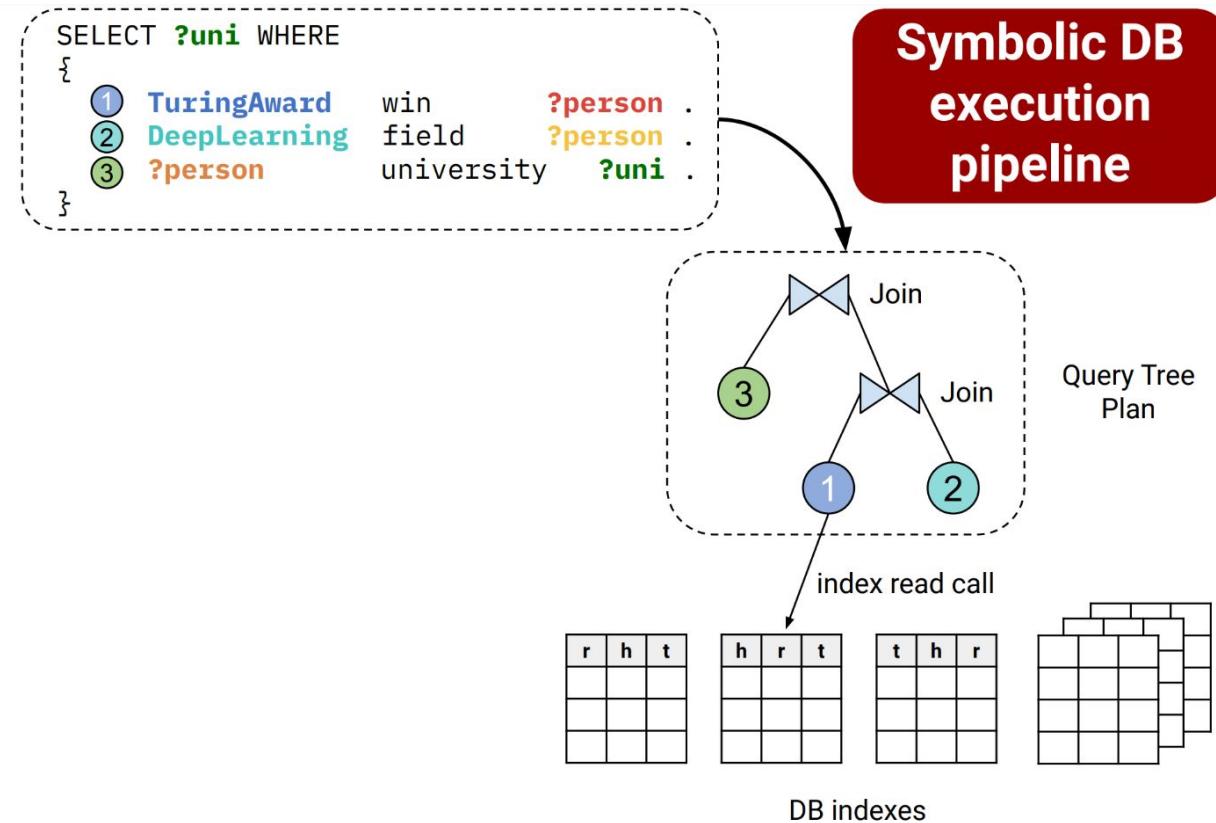
Two main components:

- Neural Graph Storage
 - Graph / Feature Store
 - Encoder
 - Embedding Store
- Neural Query Engine
 - Query Planner
 - Query Executor

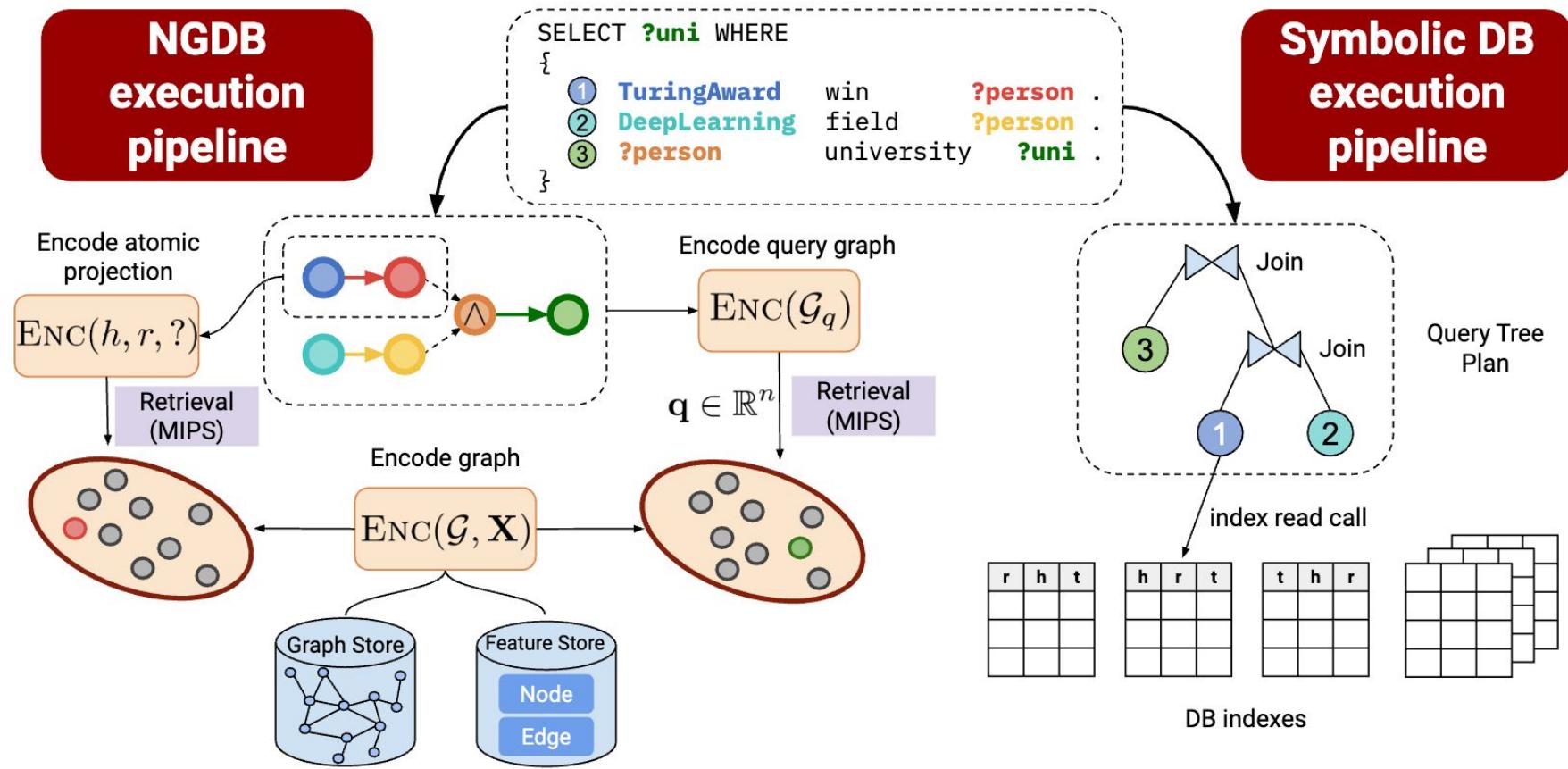
Interacting via
Query Encoding + Retrieval



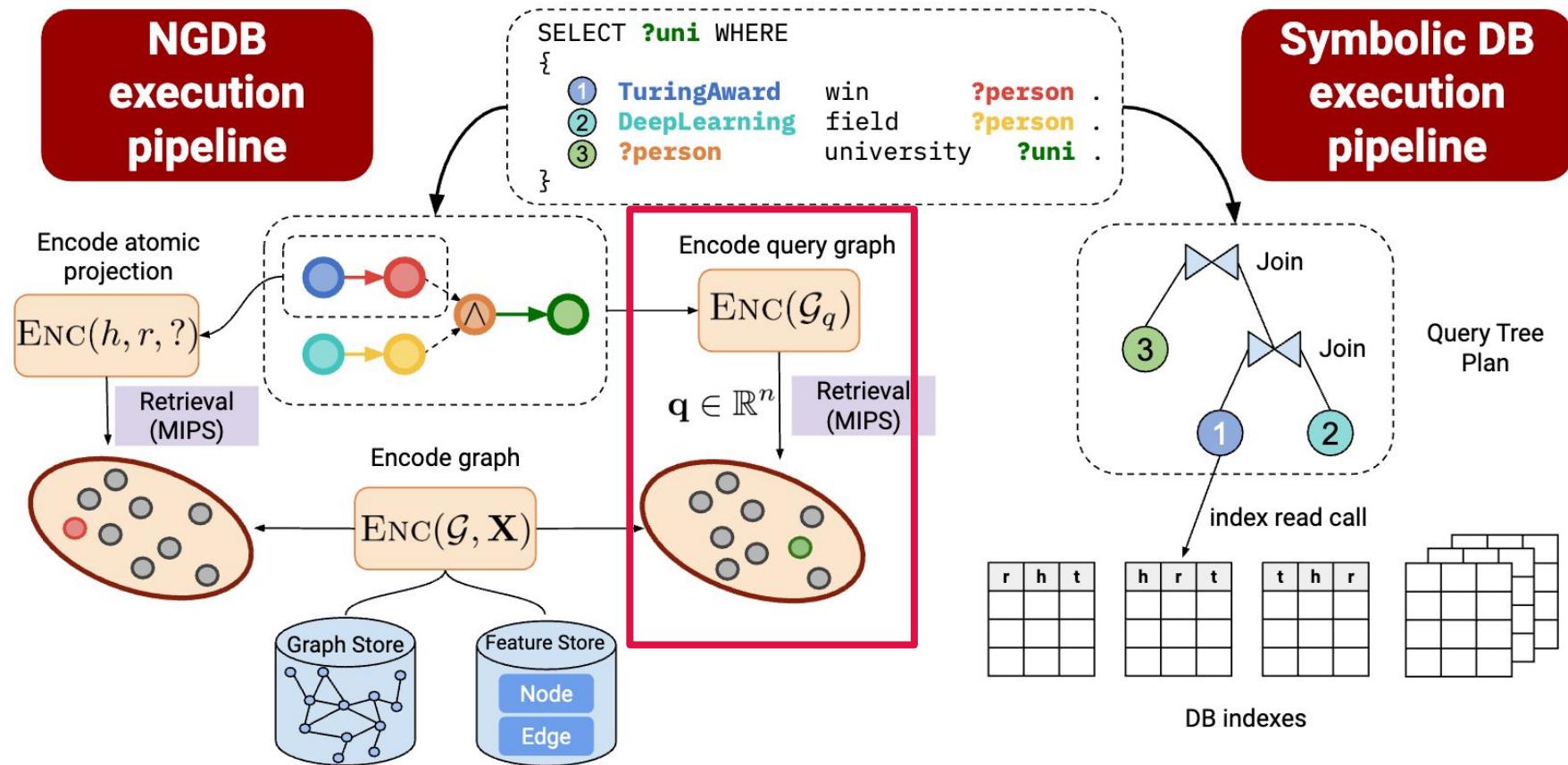
Neural Graph Storage



Neural Graph Storage

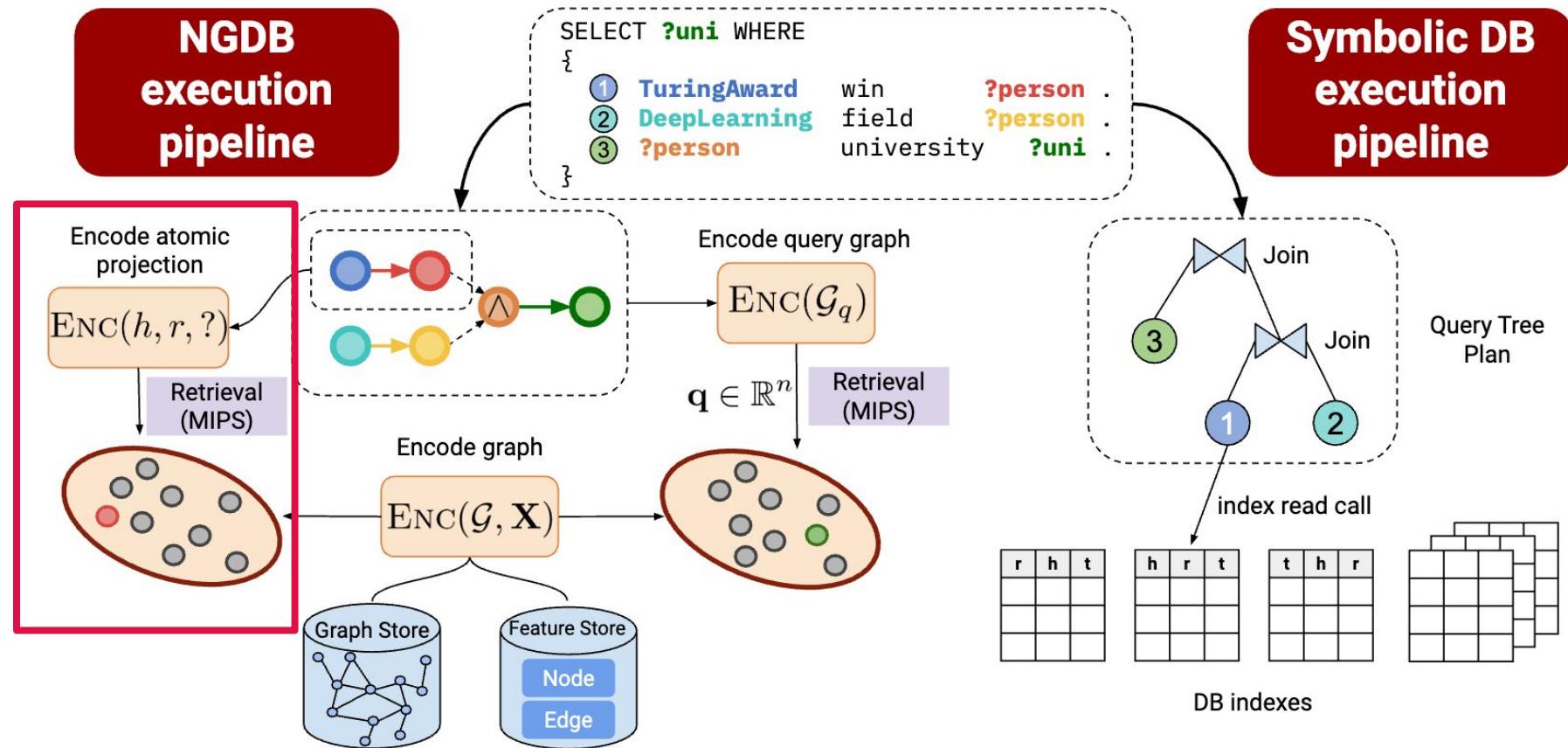


Neural Graph Storage

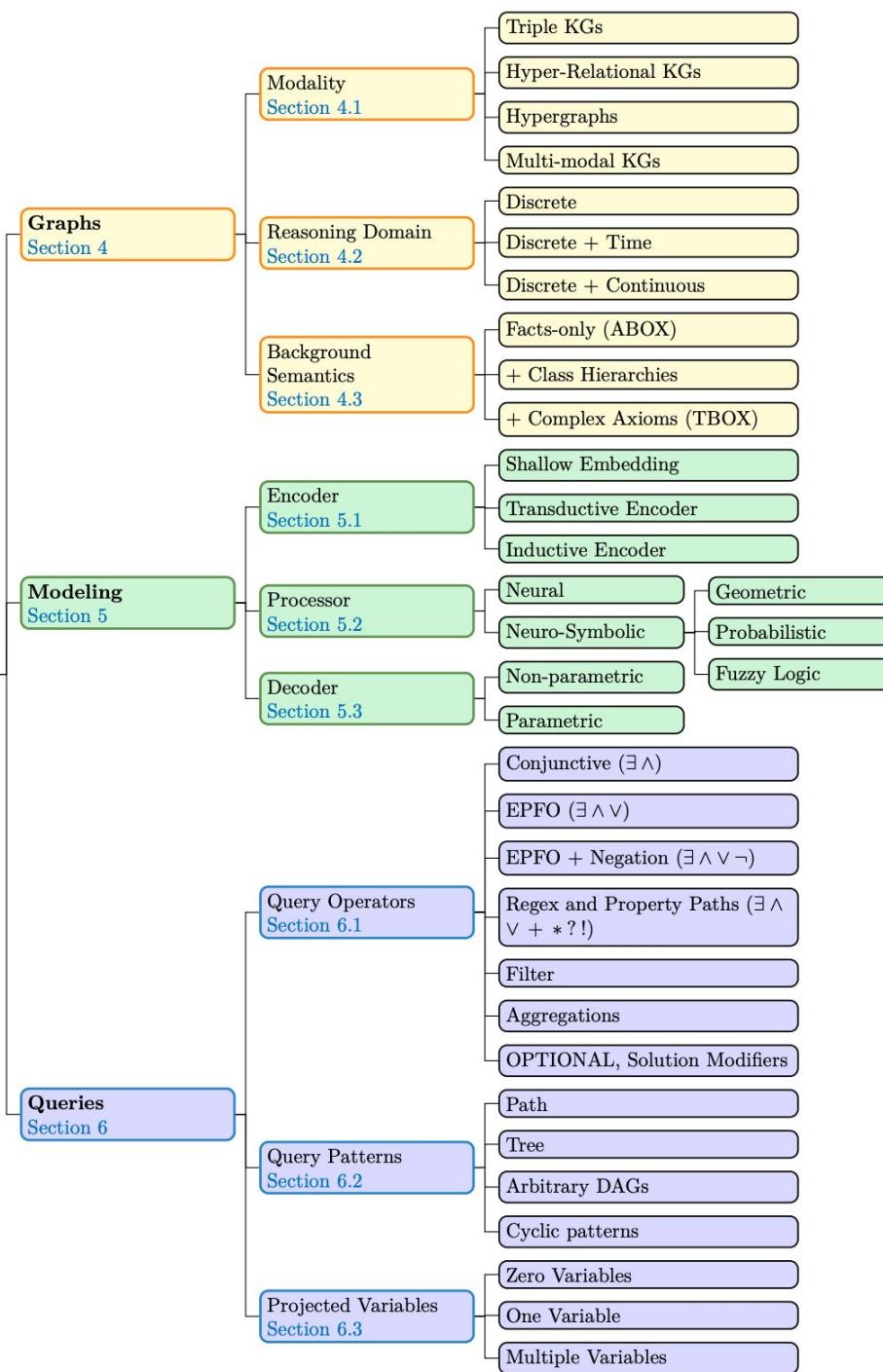


Option 1: Embed the whole query and do Retrieval (MPQE, BiQE)

Neural Graph Storage



Option 2: Embed parts of the query, execute sequentially (Query2Box, CQD)



Mega Taxonomy





Ren, Galkin, Cochez, Zhu, and Leskovec

Neural Graph Reasoning: Complex Logical Query Answering Meets Graph Databases, 2023

Code & Data



<https://github.com/neuralgraphdatabases/awesome-logical-query>
<https://www.ngdb.org/>

Blog



[Medium post](#)

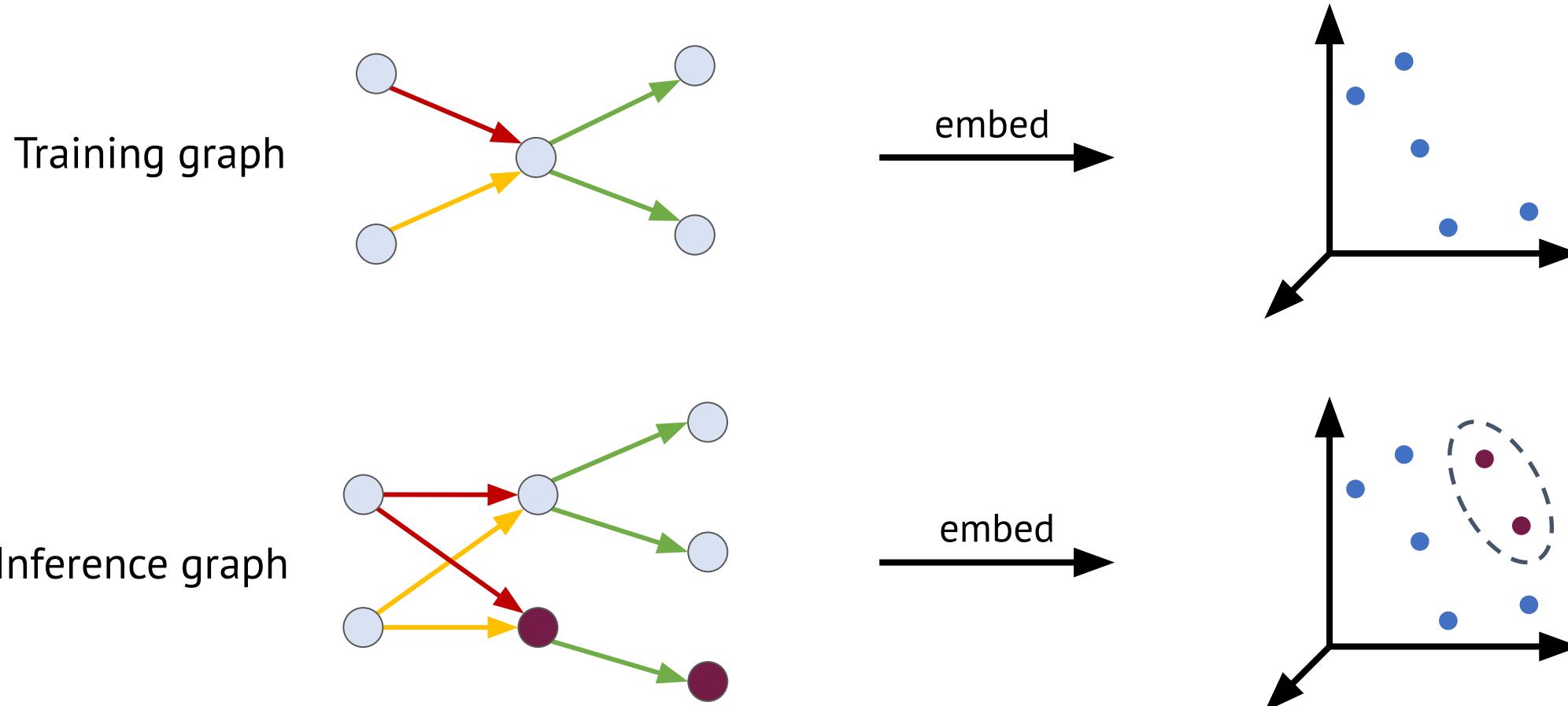


Part 5. Advanced Topics

Inductive learning settings

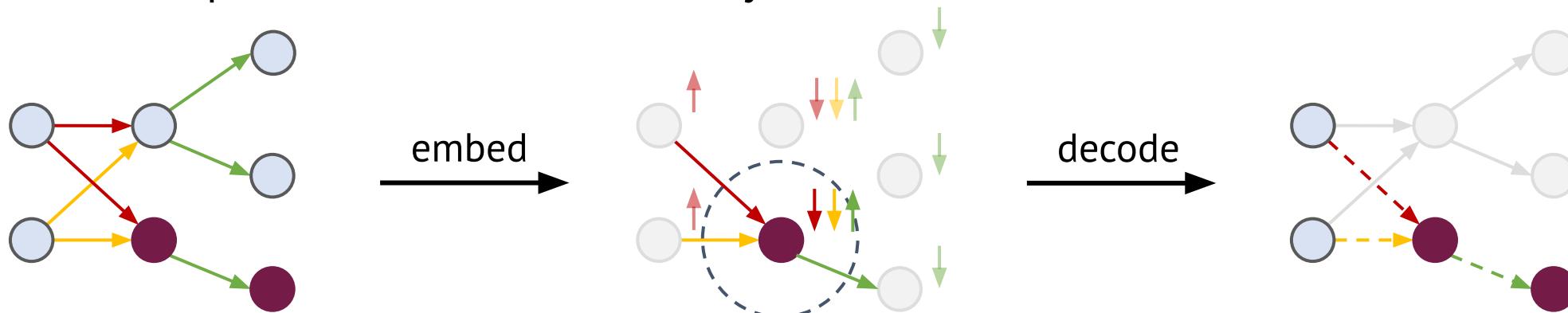
Transductive Embeddings

e.g. GQE, Query2Box, BetaE, FuzzQE, ConE, ...



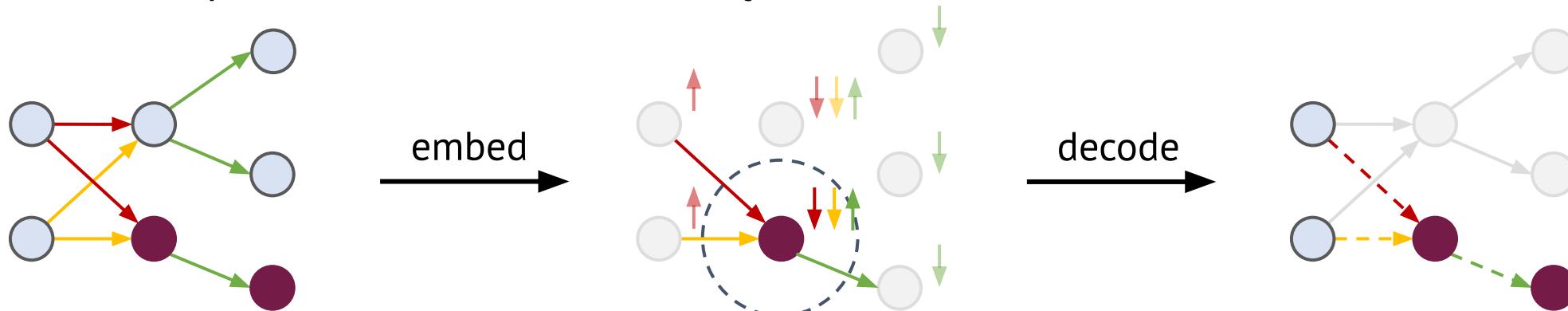
The Essence of Inductiveness

1. Inductive representations of each **entity**

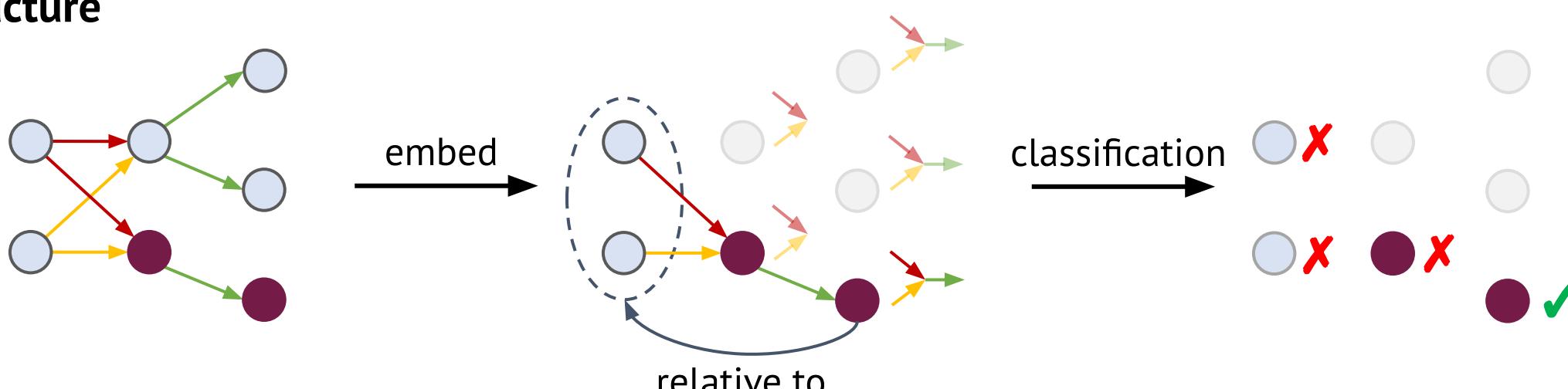


The Essence of Inductiveness

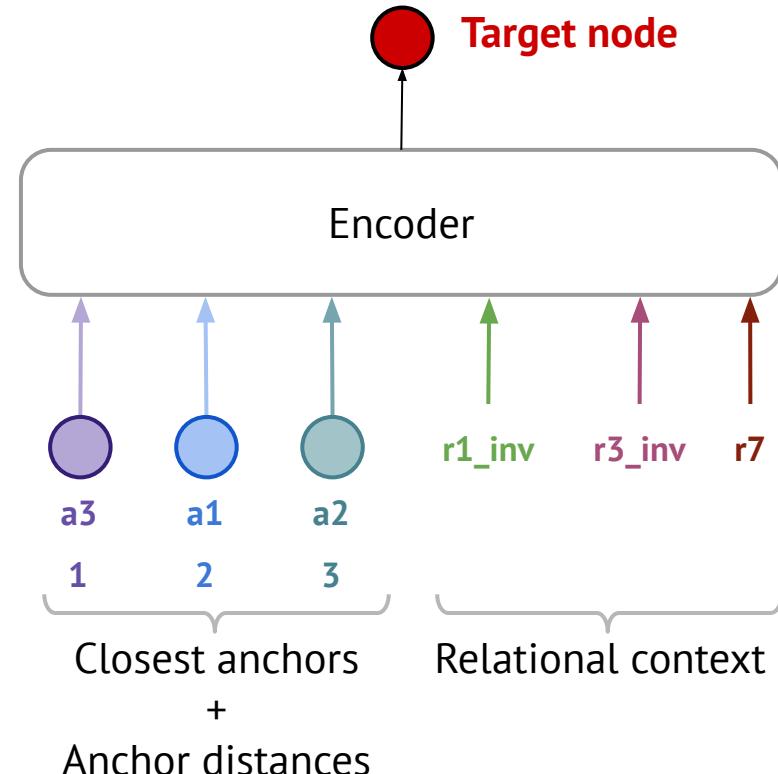
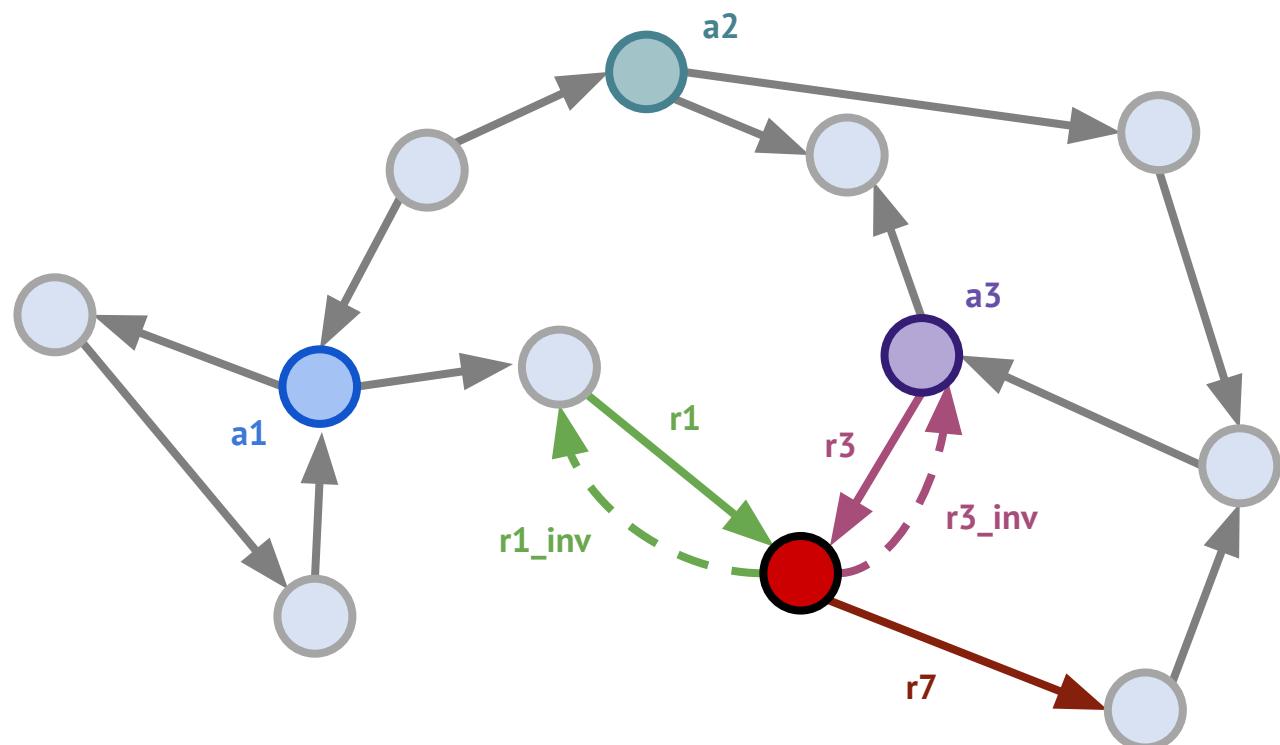
1. Inductive representations of each **entity**



2. Inductive representations of **the relative relational structure**



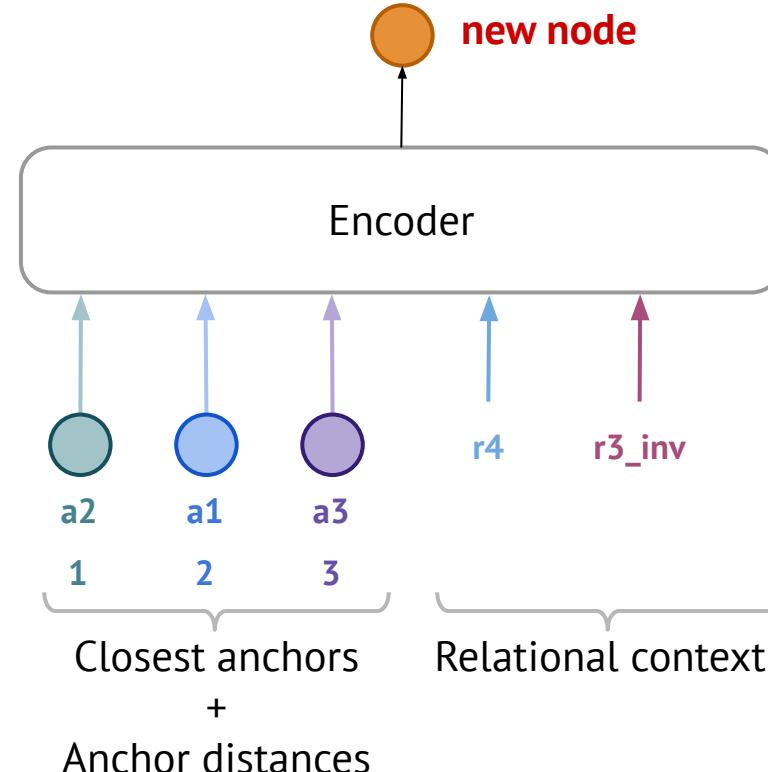
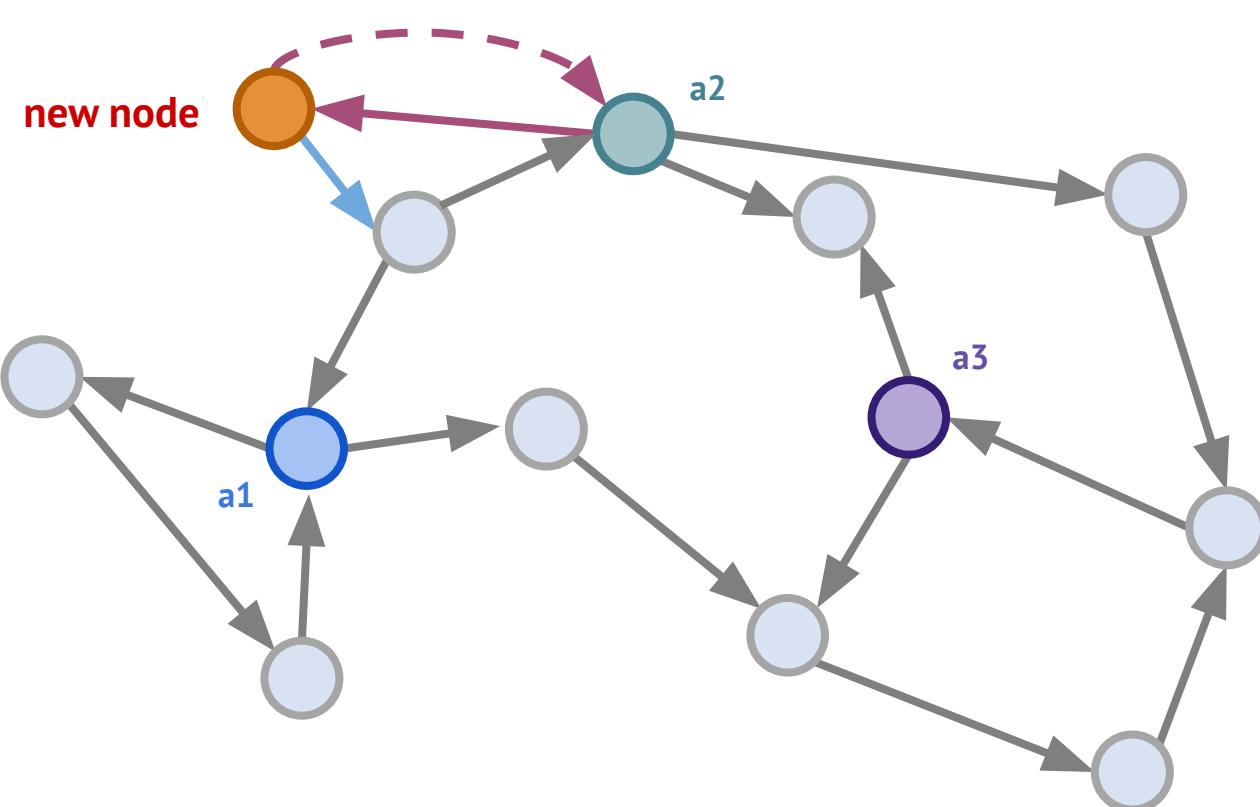
NodePiece - “subword units” for KGs



Vocabulary = Anchors + Relation types

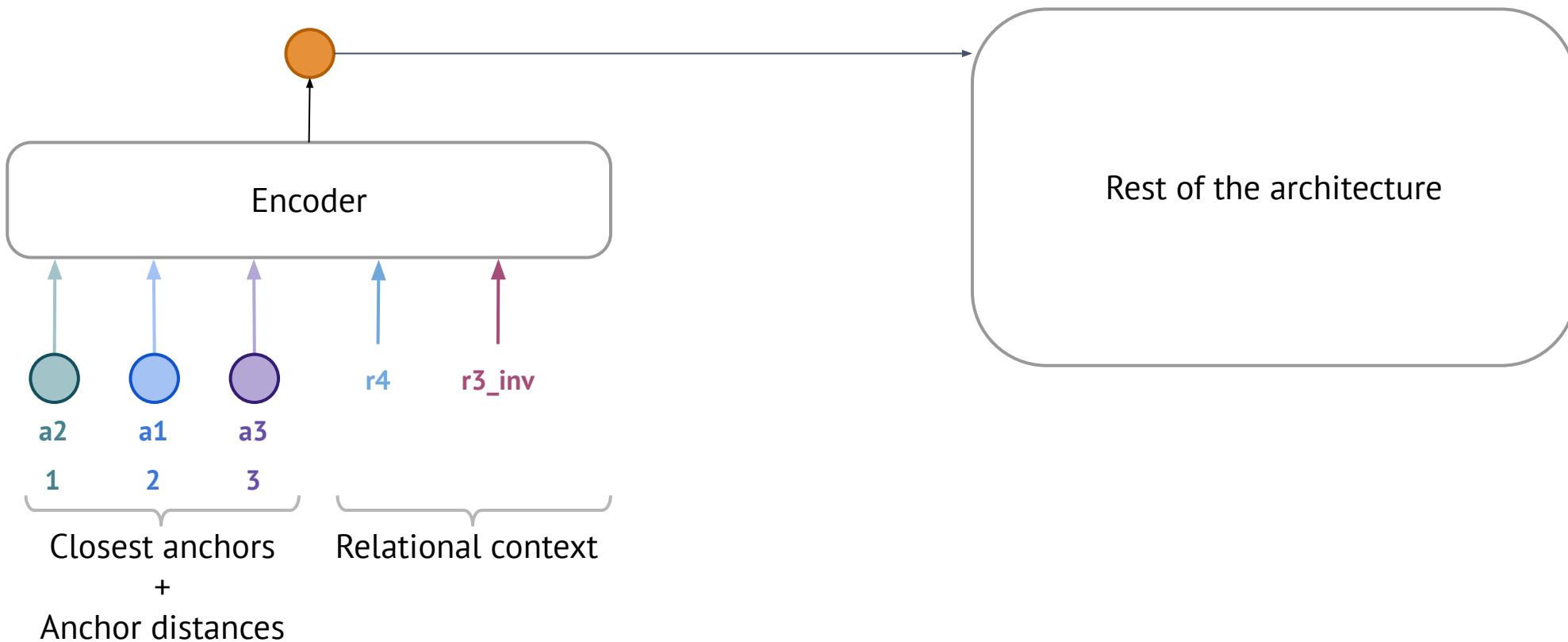
Inductive out-of-the-box: unseen nodes are “tokenized” with the same Vocab

Inductive Node Tokenization



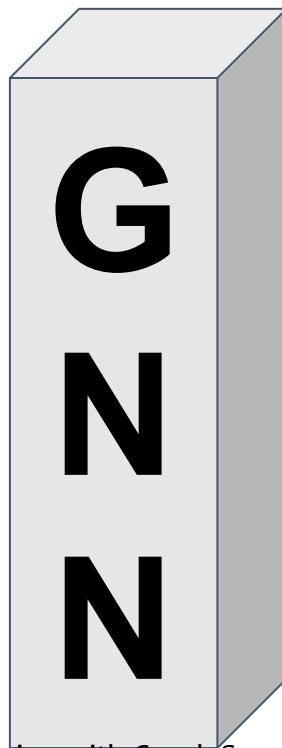
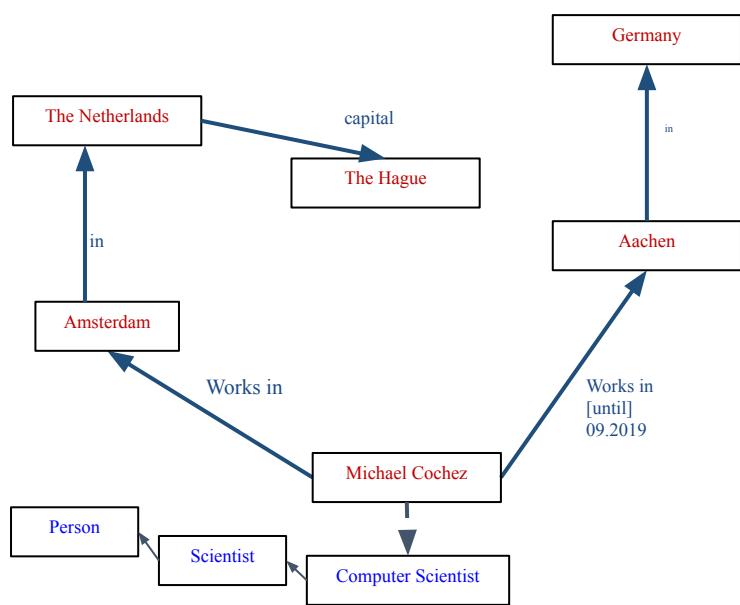
Using NodePiece for other tasks

- NodePiece is a building building block!
 - You can use this is part of a larger system



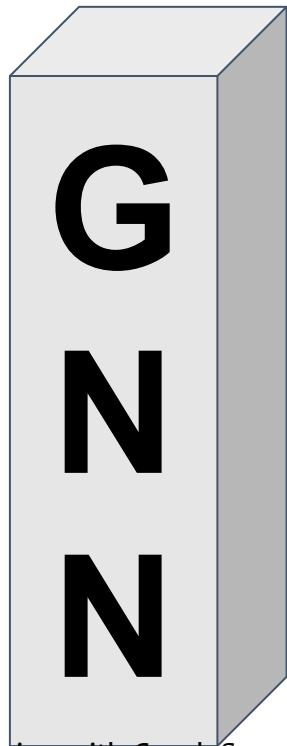
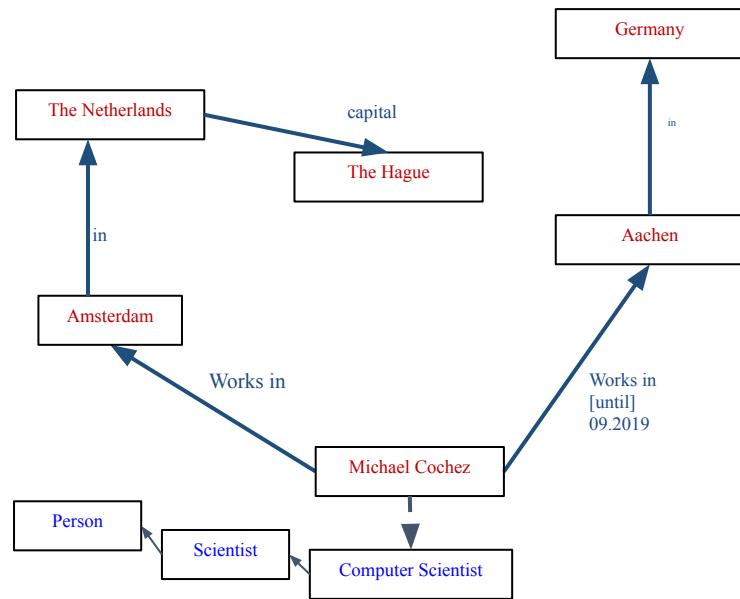
Graph summarization for inductiveness

Graph Neural Networks

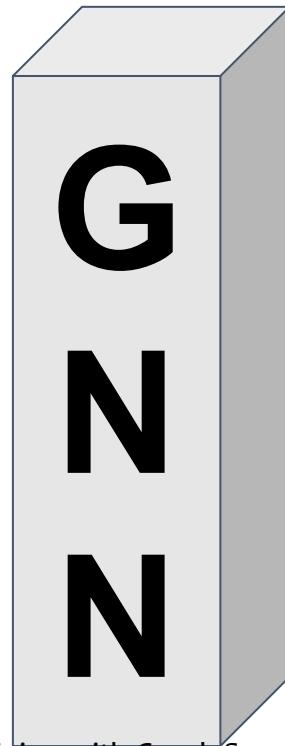
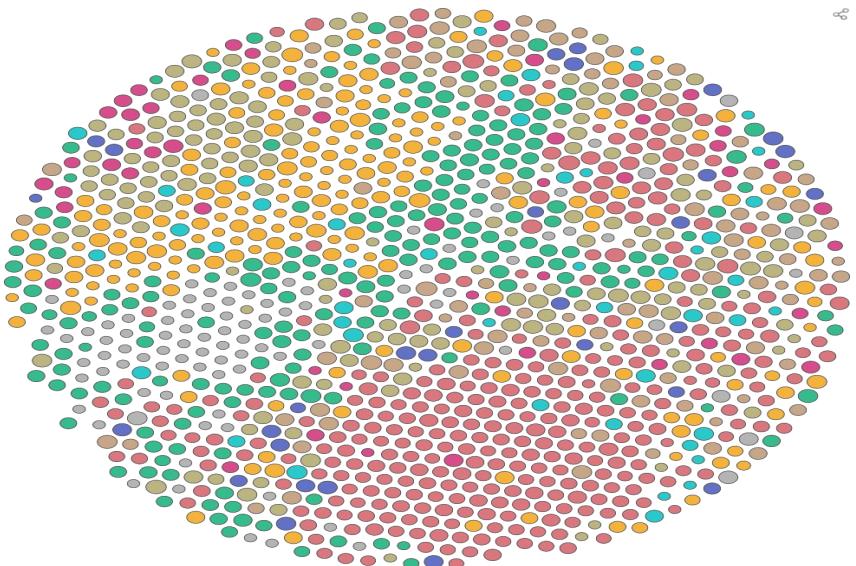


Classification
Regression
Etc.

Graph Neural Networks - small graphs

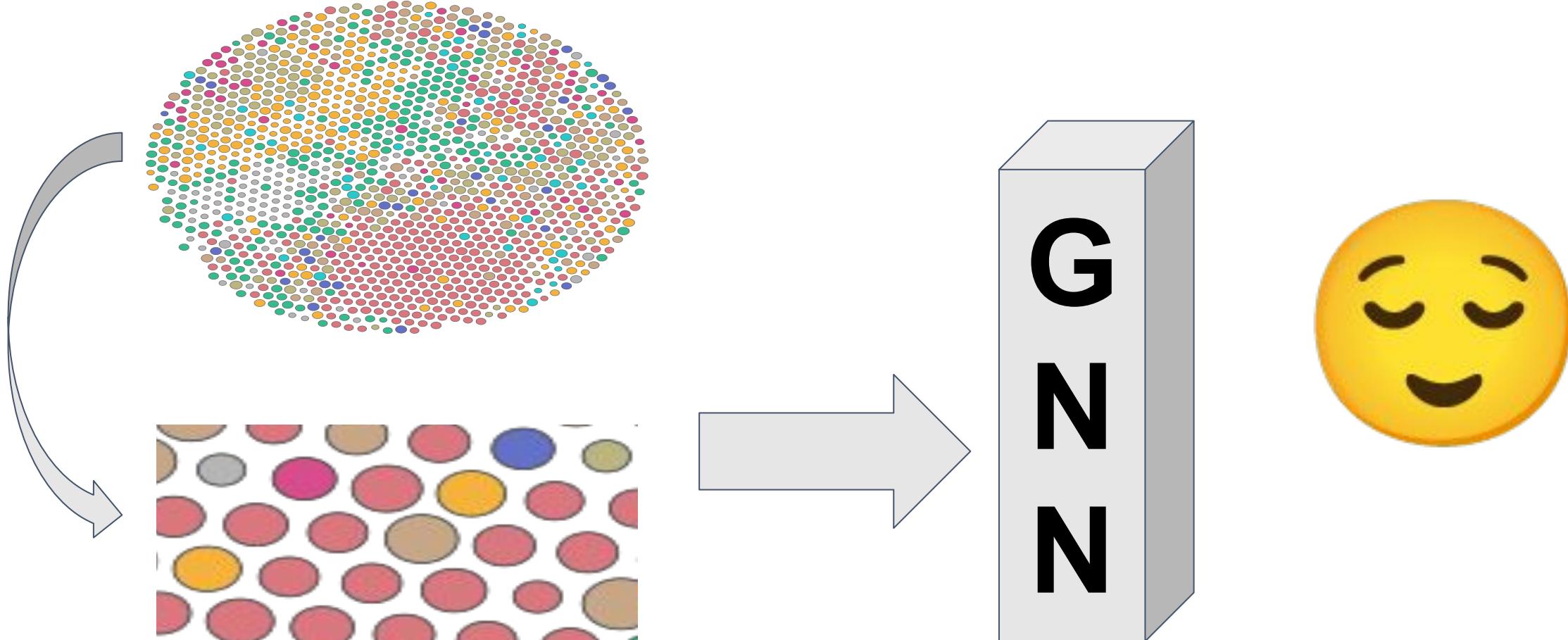


Graph Neural Networks - large graphs



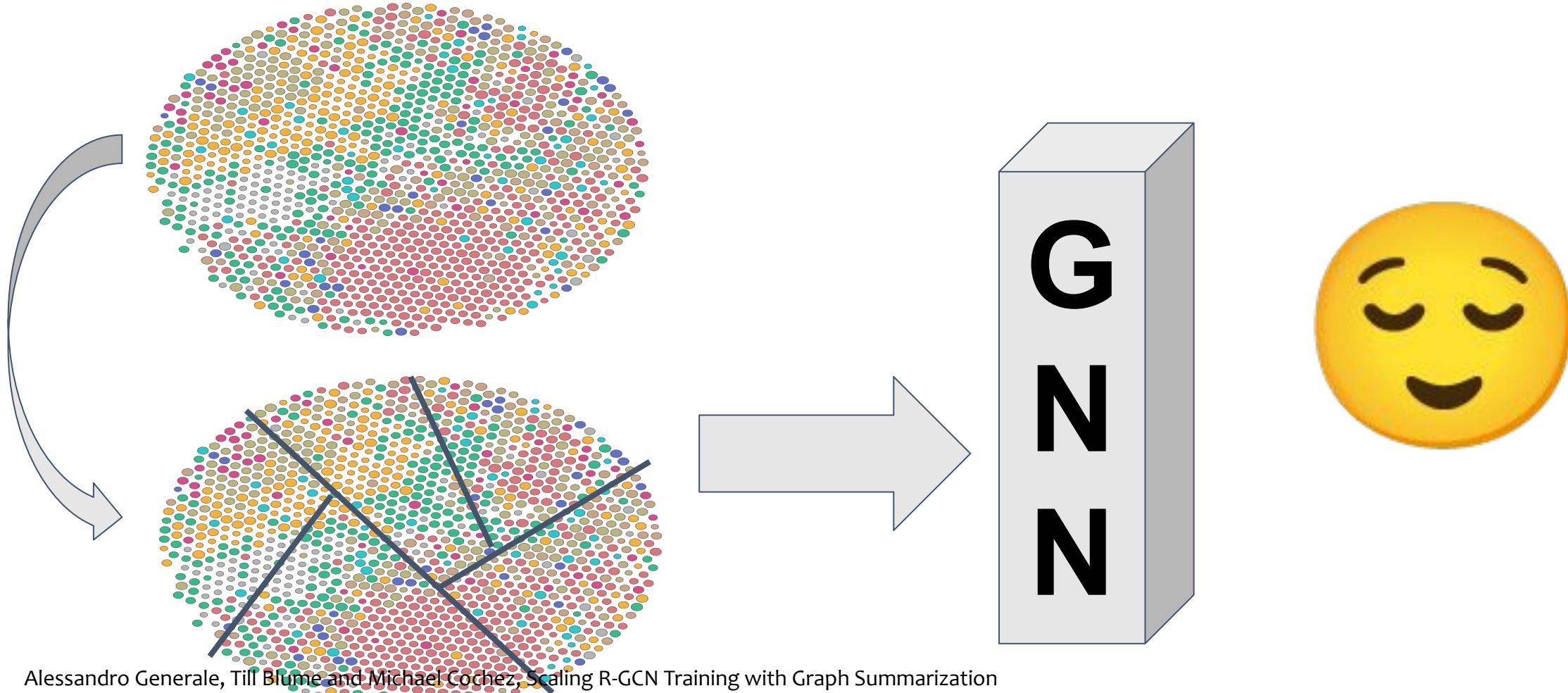
Alessandro Genovese, Till Blume and Michael Cochez, Scaling R-GCN Training with Graph Summarization

Graph Neural Networks - sampling



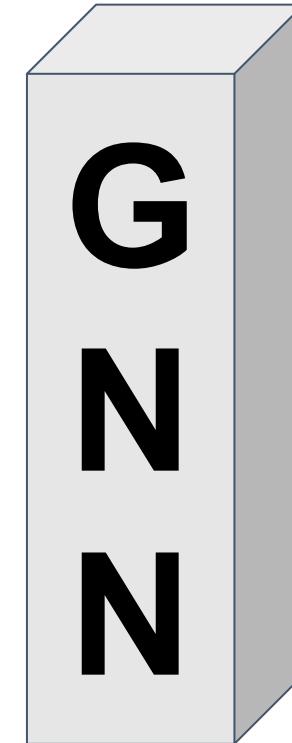
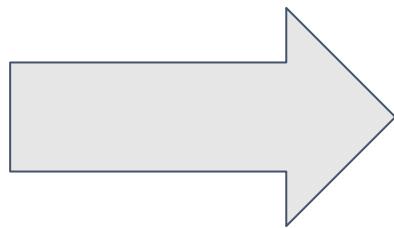
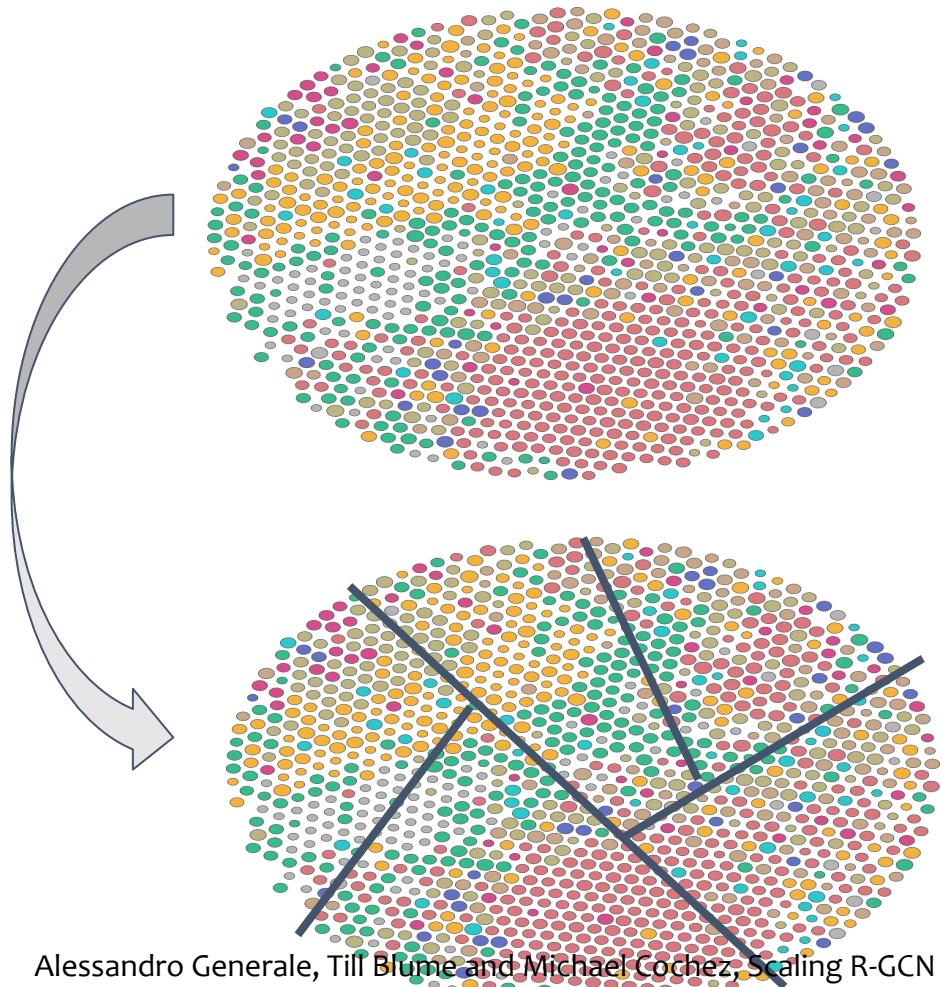
Alessandro Genovese, Till Blume and Michael Cochez, Scaling R-GCN Training with Graph Summarization

Graph Neural Networks - partitioning

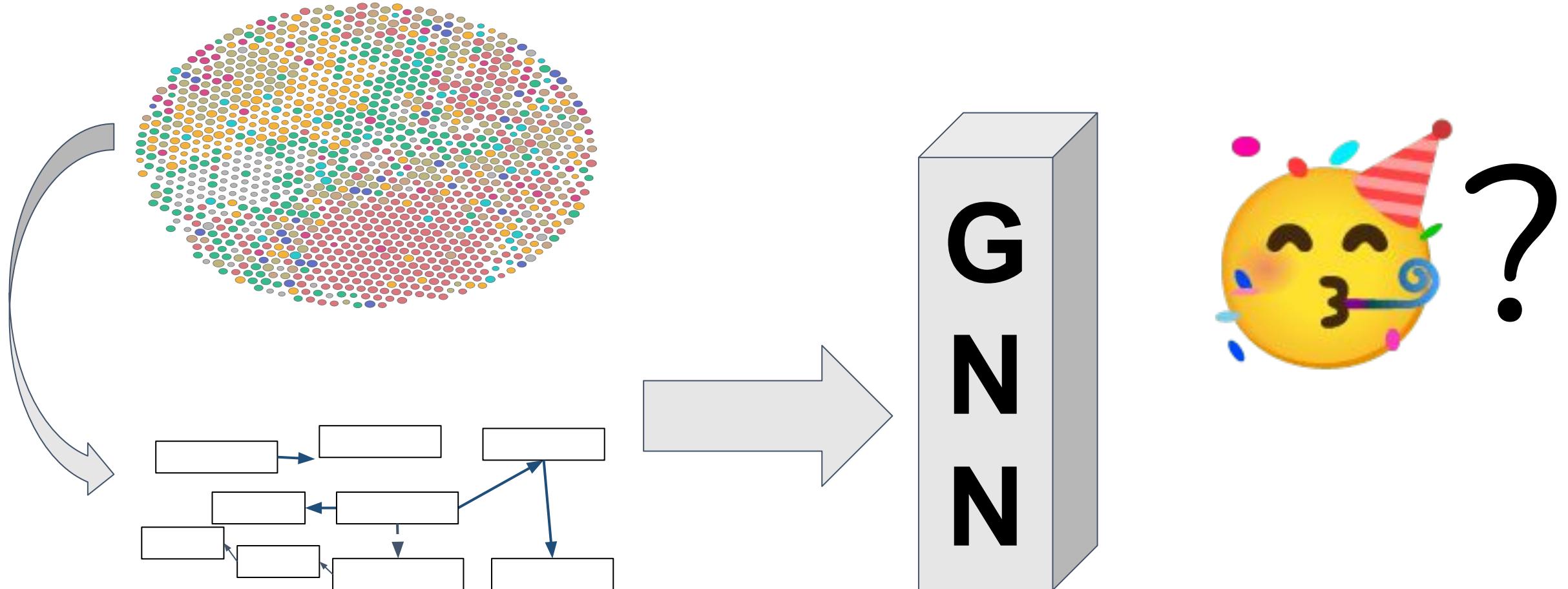


Alessandro Generale, Till Blume and Michael Cochez, Scaling R-GCN Training with Graph Summarization

Graph Neural Networks - partitioning

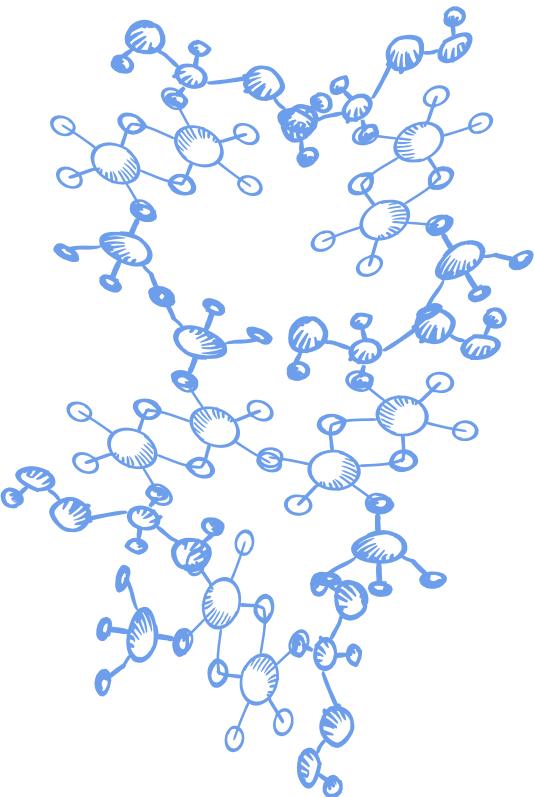


Graph Neural Networks - graph summary



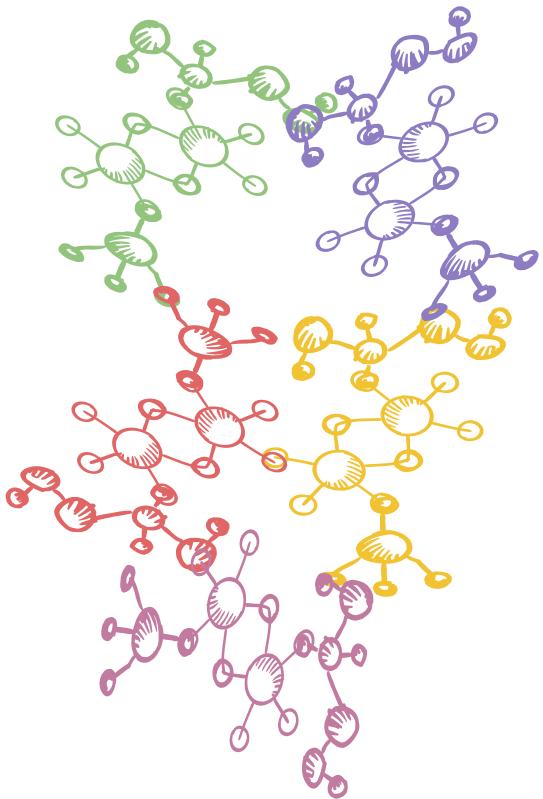
Alessandro Genovese, Till Blume and Michael Cochez, Scaling R-GCN Training with Graph Summarization

Graph Summaries for scalability



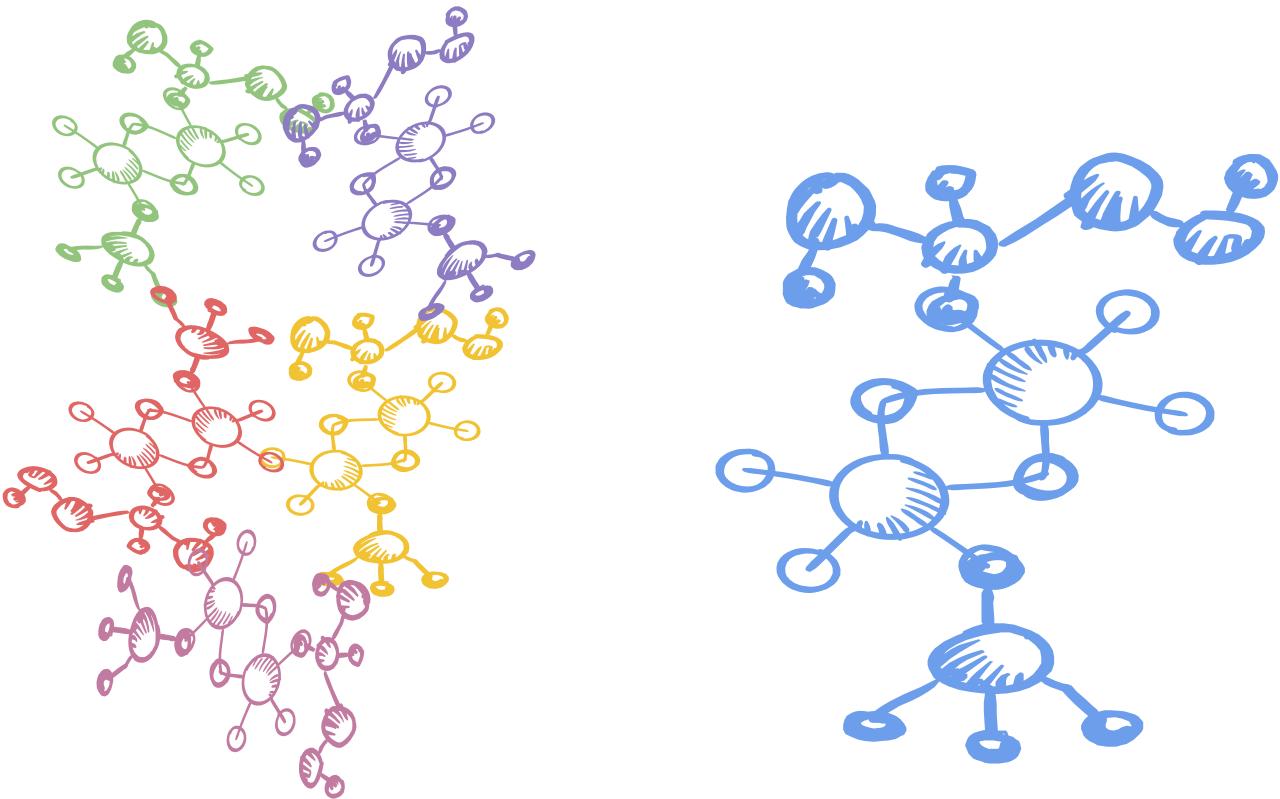
Alessandro Generale, Till Blume and Michael Cochez, Scaling R-GCN Training with Graph Summarization

Graph Summaries for scalability

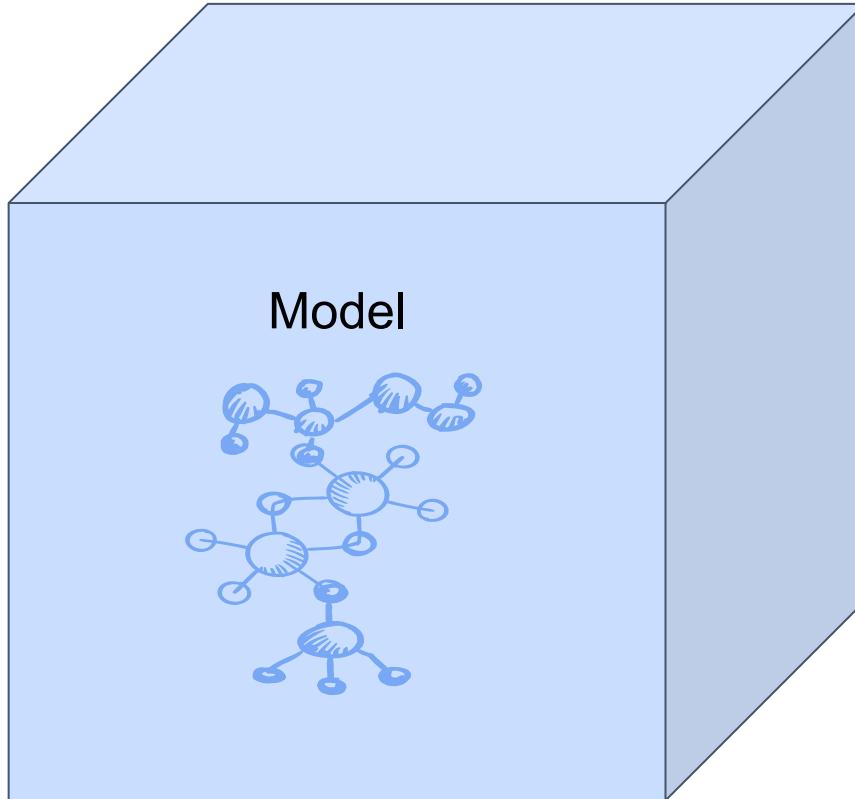
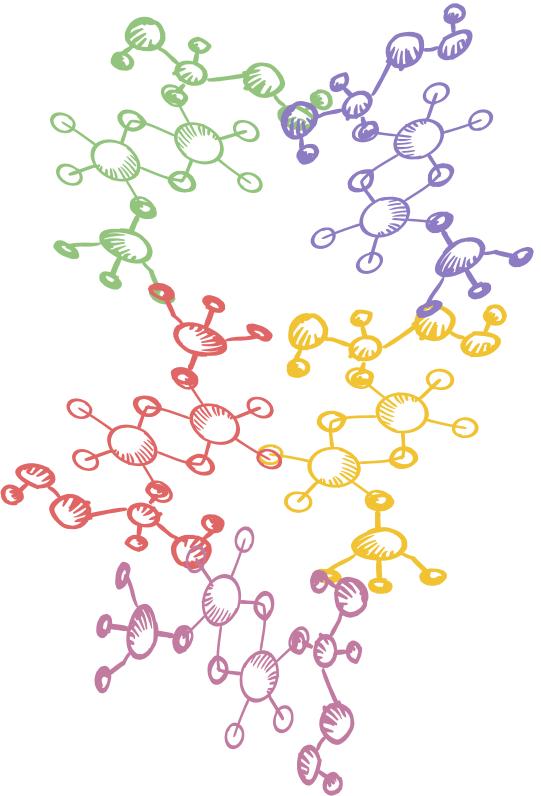


Alessandro Generale, Till Blume and Michael Cochez, Scaling R-GCN Training with Graph Summarization

Graph Summaries for scalability

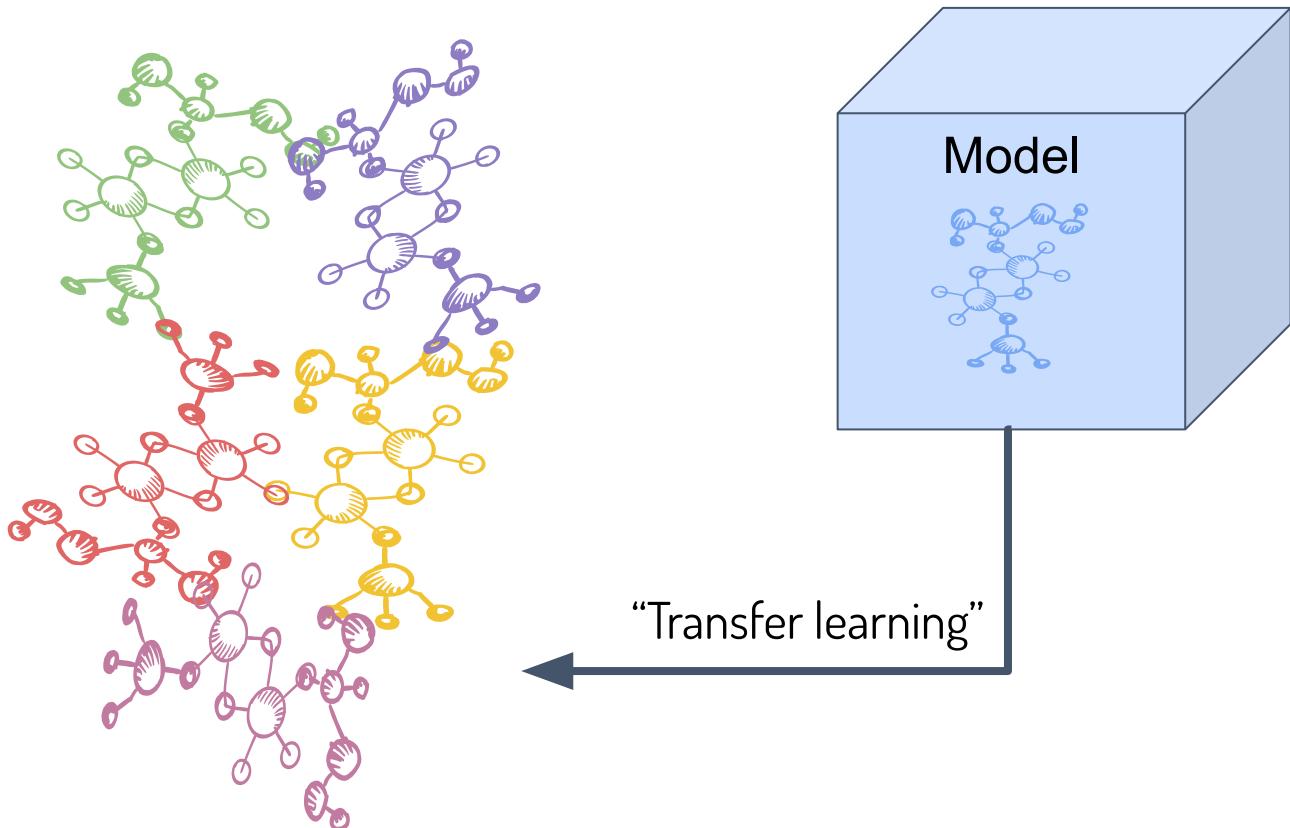


Graph Summaries for scalability



Graph Summaries for scalability

- The final model can be trained further
- The summarizing cost must be taken into account!



Using graph summarization for Inductiveness

- When the graph changes, but the summary does not
 - We are inductive!!
- What if it does change?
 - We can work like we do for NodePiece

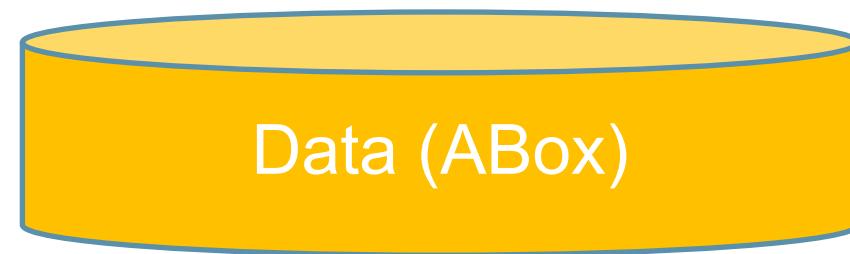


Part 5. Advanced Topics

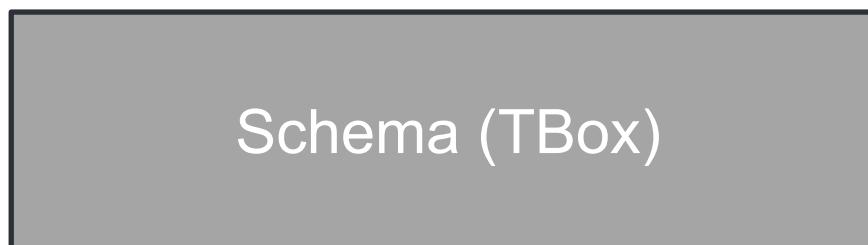
Incorporating formal background knowledge

Data vs Conceptual Knowledge

- A **knowledge base (KB)** can be divided into
 - ABox: instance information (**data level**)
 - TBox: class information (**concept level**)
- Knowledge is described via **statements expressed in Description Logic / Ontology**



- $(alice, \text{type}, \text{DataScientist})$
- $(bob, \text{type}, \text{SoftwareEngineer})$
- $(alice, \text{has_employer}, \text{ibm})$
- $(bob, \text{has_employer}, \text{ibm})$



- $\text{DataScientist} \sqsubseteq \text{Employee}$
- $\text{SoftwareEngineer} \sqsubseteq \text{Employee}$
- $\text{Employee} \equiv \exists \text{has_employer}. \text{Employer}$
- $\text{TechCompany} \sqsubseteq \text{Company}$

We aim at embedding KBs with conceptual knowledge

Small selection of methods

- Change the loss function to include (soft) semantic constraints
- Change the space in which we embed to reflect semantics

TransOWL / TransROWL

From TransE

$$L = \sum_{\substack{\langle h, r, t \rangle \in \Delta \\ \langle h', r, t' \rangle \in \Delta'}} [\gamma + f_r(h, t) - f_r(h', t')]$$

TransOWL / TransROWL

From TransE ... to TransOWL

$$\begin{aligned} L = & \sum_{\substack{\langle h, r, t \rangle \in \Delta \\ \langle h', r, t' \rangle \in \Delta'}} [\gamma + f_r(h, t) - f_r(h', t')]_+ + \sum_{\substack{\langle t, q, h \rangle \in \Delta_{\text{inverseOf}} \\ \langle t', q, h' \rangle \in \Delta'_{\text{inverseOf}}}} [\gamma + f_q(t, h) - f_q(t', h')]_+ \\ & + \sum_{\substack{\langle h, s, t \rangle \in \Delta_{\text{equivProperty}} \\ \langle h', s, t' \rangle \in \Delta'_{\text{equivProperty}}}} [\gamma + f_s(h, t) - f_s(h', t')]_+ + \sum_{\substack{\langle h, \text{typeOf}, l \rangle \in \Delta \cup \Delta' \\ \langle h', \text{typeOf}, l' \rangle \in \Delta' \cup \Delta'_{\text{equivClass}}}} [\gamma + f_{\text{typeOf}}(h, l) - f_{\text{typeOf}}(h', l')]_+ \\ & + \sum_{\substack{\langle h, \text{subClassOf}, p \rangle \in \Delta_{\text{subClass}} \\ \langle h', \text{subClassOf}, p' \rangle \in \Delta'_{\text{subClass}}}} [(\gamma - \beta) + f(p, h) - f(p', h')]_+ \end{aligned} \tag{5}$$

TransOWL / TransROWL

From TransE ... to TransOWL

$$L = \sum_{\substack{\langle h, r, t \rangle \in \Delta \\ \langle h', r, t' \rangle \in \Delta'}} [\gamma + f_r(h, t) - f_r(h', t')]_+ + \sum_{\substack{\langle h, s, t \rangle \in \Delta_{\text{equivProperty}} \\ \langle h', s, t' \rangle \in \Delta'_{\text{equivProperty}}}} [\gamma + f_s(h, t) - f_s(h', t')]_+ + \sum_{\substack{\langle h, \text{subClassOf}, p \rangle \in \Delta_{\text{subClass}} \\ \langle h', \text{subClassOf}, p' \rangle \in \Delta'_{\text{subClass}}}} [(\gamma + f_p(h, p) - f_p(h', p')]_+$$

Each of these extra pieces adds more triples which are obtained by a formal reasoner on

- inverseOf
 - A relation r' is the inverse of relation r
 - $(a, r', b) \Rightarrow (b, r, a)$
- equivProperty
 - Relation r is equivalent to relation q
 - $(a, r, b) \Rightarrow (a, q, b)$
- equivClass
 - A class C is equivalent with class D
 - $(a, \text{type } C) \Rightarrow (a, \text{type } D)$
- subClass
 - A class C is subclass of S
 - $(a, \text{type } C) \text{ and } (C, \text{subClassOf}, D) \Rightarrow (a, \text{type } D)$

$$t', h')]_+ - f_{\text{typeOf}}(h', l')]_+ \quad (5)$$

EL++ Knowledge Bases

- EL++ is a lightweight description logic that
 - balances **expressive power** and **reasoning complexity (polynomial)**
 - is applied for large-scale ontologies, e.g. Gene Ontology



ABox contains:

1. concept assertion: $C(a)$
2. role assertion: $r(a,b)$

TBox statements can be normalized into:

1. concept subsumption: $C \sqsubseteq D$,
2. concept intersection: $C_1 \sqcap C_2 \sqsubseteq D$,
3. right existential: $\exists r. C_1 \sqsubseteq D$,
4. left existential: $C_1 \sqsubseteq \exists r. C_2$,

Ball EL Embedding

- **Geometric construction:** EL embeddings [Kulmanov et al. 2019]

- concepts/entities as **balls**, relations as **translation** (like TransE)
- do not capture all **intersectional closure** and **complex relational mappings**
- no distinction between entities/concepts



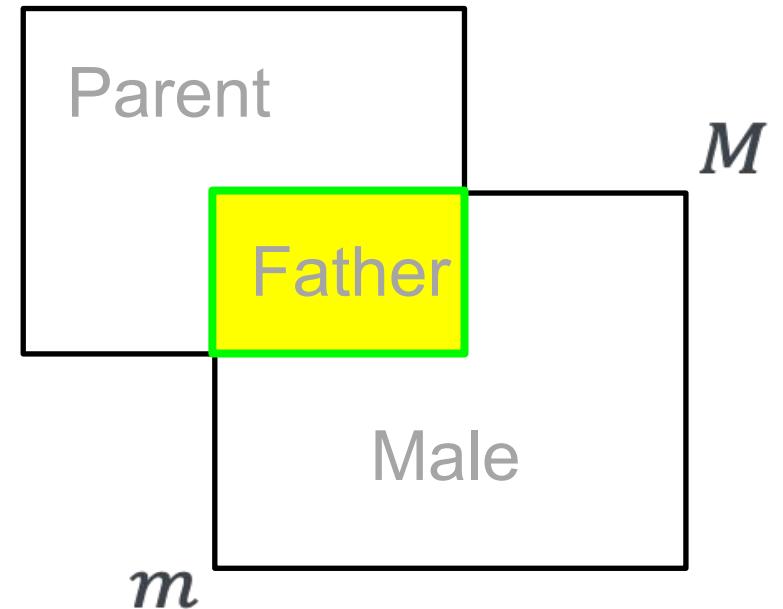
Box EL++ Embedding

- Concepts are represented by boxes

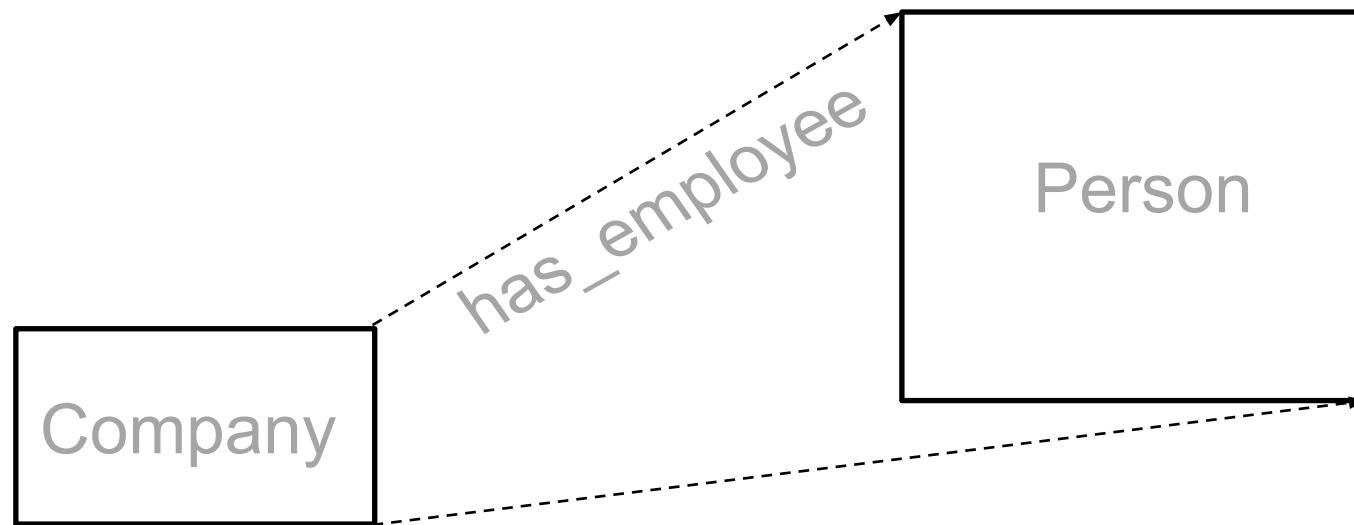
- capture **intersectional closure**
- parameterized by
a **lower-left corner** + an **upper-right corner**

- Entities are represented by **points**

- special cases of boxes where $m=M$ (i.e., boxes with minimal volume)
- entities are most specific concepts



- Relations are represented by **affine transformations**
 - parameterized by a **diagonal matrix** + a **translation vector**
 - modeling relational semantics between **concepts with varying size**



Idea: mapping **logical constraints** to **geometric (soft) constraints**

- designing one loss term for each logical statement
- such that the KB is **satisfiable** when the loss is 0 (a.k.a. **soundness**)
- satisfiability implies that there is a **geometric interpretation** satisfying all logical statements

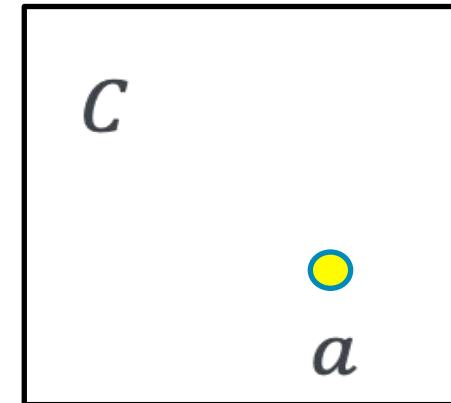
Solution: finding a geometric interpretation for each statement

Geometric Interpretations of ABox

Concept assertion $C(a)$



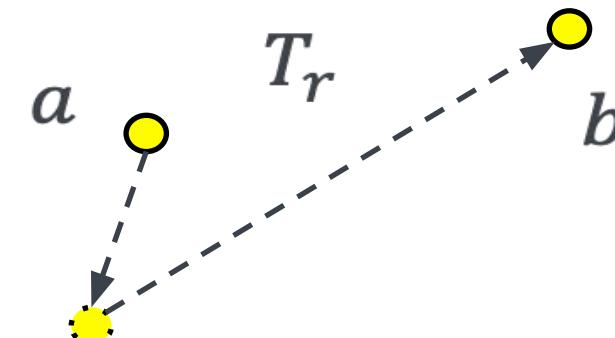
Geometric membership



Role assertion $r(a, b)$



Affine transformation
between two points

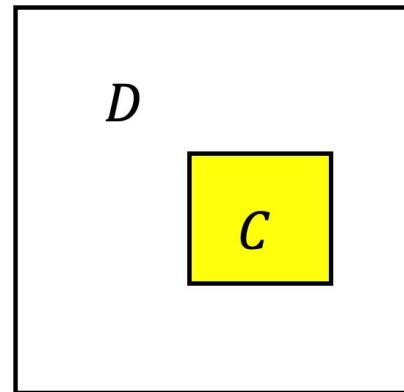


Geometric Interpretations of TBox

$$C \sqsubseteq D, D \neq \perp$$



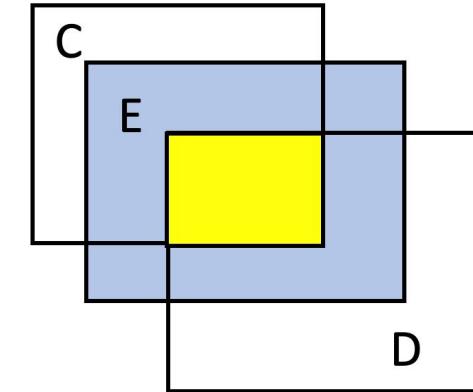
Box entailment



$$C \sqcap D \sqsubseteq E, E \neq \perp$$



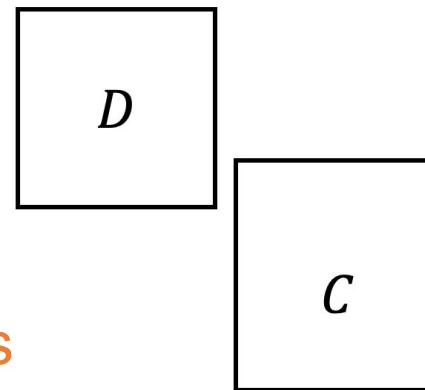
Box intersection



$$C \sqcap D \sqsubseteq \perp$$



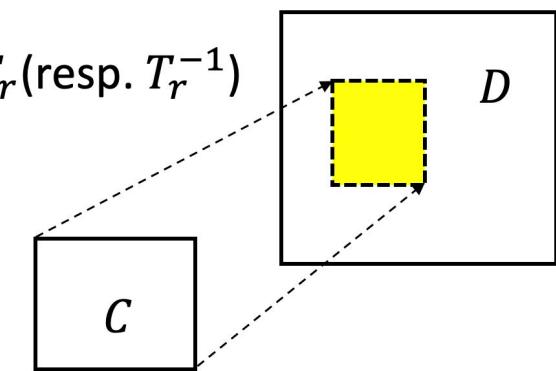
Box disjointedness



$$C \sqsubseteq \exists r. D$$

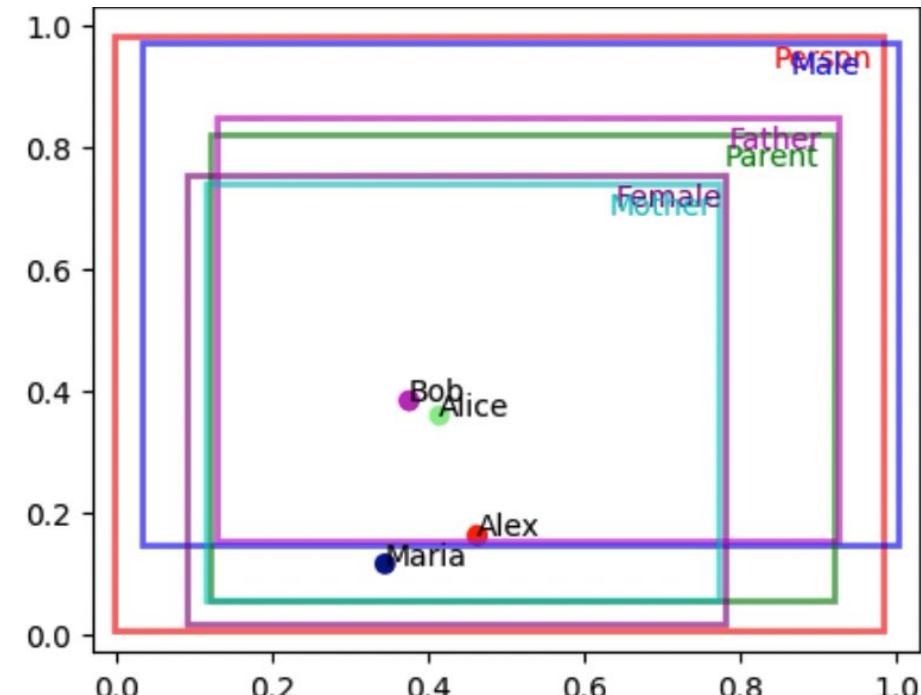


Box affine
transformation



Toy Family Example

Male ⊑ Person	Female ⊑ Person
Father ⊑ Male	Mother ⊑ Female
Father ⊑ Parent	Mother ⊑ Parent
Female ∩ Male ⊑ ⊥	Female ∩ Parent ⊑ Mother
Male ∩ Parent ⊑ Father	$\exists \text{hasChild}.\text{Person} \sqsubseteq \text{Parent}$
Parent ⊑ Person	Parent ⊑ $\exists \text{hasChild}.\text{Person}$
Father(Alex)	Father(Bob)
Mother(Marie)	Mother(Alice)



Subsumption Reasoning

$$P(C \sqsubseteq D) = \frac{\text{MVol}(\text{Box}(C) \cap \text{Box}(D))}{\text{MVol}(\text{Box}(C))}$$

Table 4: The ranking based measures of embedding models for subsumption reasoning on the testing set. * denotes the results from [20].

Dataset	Metric	TransE*	TransH*	DistMult*	ELEM	EmEL ⁺⁺	BoxEL
GO	Hits@10	0.00	0.00	0.00	0.09	0.10	0.03
	Hits@100	0.00	0.00	0.00	0.16	0.22	0.08
	AUC	0.53	0.44	0.50	0.70	0.76	0.81
	Mean Rank	-	-		13719	11050	8980
GALEN	Hits@10	0.00	0.00	0.00	0.07	0.10	0.02
	Hits@100	0.00	0.00	0.00	0.14	0.17	0.03
	AUC	0.54	0.48	0.51	0.64	0.65	0.85
	Mean Rank	-	-		8321	8407	3584
ANATOMY	Hits@10	0.00	0.00	0.00	0.18	0.18	0.03
	Hits@100	0.01	0.00	0.00	0.38	0.40	0.04
	AUC	0.53	0.44	0.49	0.73	0.76	0.91
	Mean Rank	-	-		28564	24421	10266

$$P(\text{interacts}(P_1, P_2)) = \left\| T_w^{\text{interacts}}(m_w(P_1)) - m_w(P_2) \right\|_2$$

Table 6: Prediction performance on protein-protein interaction (human).

Method	Raw	Filtered	Raw	Filtered	Raw	Filtered	Raw	Filtered
	Hits@10	Hits@10	Hits@100	Hits@100	Mean Rank	Mean Rank	AUC	AUC
TransE	0.05	0.11	0.24	0.29	3960	3891	0.78	0.79
BoxE	0.05	0.10	0.26	0.32	2121	2091	0.87	0.87
SimResnik	0.05	0.09	0.25	0.30	1934	1864	0.88	0.89
SimLin	0.04	0.08	0.20	0.23	2288	2219	0.86	0.87
ELEm	0.01	0.02	0.22	0.26	1680	1638	0.90	0.90
EmEL ⁺⁺	0.01	0.03	0.23	0.26	1671	1638	0.90	0.91
Onto2Vec	0.05	0.08	0.24	0.31	2435	2391	0.77	0.77
OPA2Vec	0.03	0.07	0.23	0.26	1810	1768	0.86	0.88
BoxEL (Ours)	0.07	0.10	0.42	0.63	1574	1530	0.93	0.93

Conclusion

- We propose a Box EL++ KB embedding
 - representing ABox + TBox knowledge
 - better encoding concept intersection and role semantics
 - theoretical guarantee (e.g., soundness) for preserving logical structure
 - empirical improvements on biomedical KBs
- Future Work
 - negation, completeness, and uncertainty in description logic
 - Injecting background knowledge into ML tasks

Other Approaches

- **Annotation based:** Onto2Vec & OPA2Vec [Smaili et al. 2019]
 - do not model logical structure explicitly
- **Capture what reasoning means**
 - Reasoning over RDF Knowledge Bases using Deep Learning [Ebrahimi et al. 2019]
 - On the Generalization Capability of Memory Networks for Reasoning [Ebrahimi et al. 2019]
- **Box2EL: Concept and Role Box Embeddings for the Description Logic EL++** [M Jackermeier et al. 2023]
- ETC...

Open questions

- How to encode full first-order logic (conjunction, disjunction, negation)?
- How to encode relation over regions? especially one-to-N relations?
 - Translation? affine transformation?
- Any other thoughts for logic embeddings?
- What are the potential applications?

Using LLMs

One example for Link Prediction / Knowledge Graph Completion

Prompting as Probing: Using Language Models for Knowledge Base Construction – GPT-3 Prompting



Triple

(The Netherlands, CountryBordersCountry, ?)

Prompt

Few-shot Examples

Which countries neighbour North Korea?
['South Korea', 'China', 'Russia']

Which countries neighbour Serbia?
['Montenegro', 'Kosovo', 'Bosnia and Herzegovina', 'Hungary',
'Croatia', 'Bulgaria', 'Macedonia', 'Albania', 'Romania']

Which countries neighbour Fiji?
['None']

Question

Which countries neighbour The Netherlands?

Answer

United Kingdom, Germany, Belgium

Natural Language vs. Triples

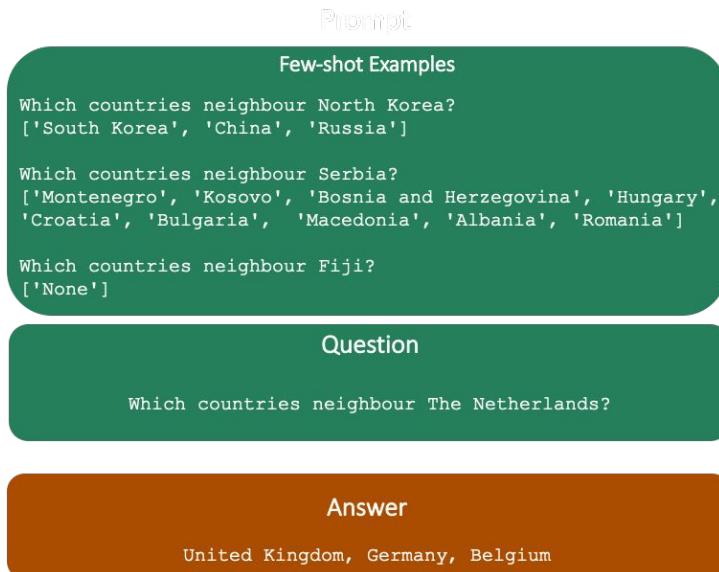
Natural Language Question

Which countries neighbour North Korea?
['South Korea', 'China', 'Russia']

Triple Question

North Korea **CountryBordersCountry**
['South Korea', 'China', 'Russia']

Factual Probing



Our Result

67.6% F1

Factual Probing

The Netherlands borders Belgium.



True



False

F1-Scores per Relation

Relation	Triple	Natural Language
ChemicalCompoundElement	0.94	0.88
CompanyParentOrganization	0.59	0.39
CountryBordersWithCountry	0.77	0.79
CountryOfficialLanguage	0.83	0.79
PersonCauseOfDeath	0.55	0.50
PersonEmployer	0.23	0.26
PersonInstrument	0.50	0.45
PersonLanguage	0.83	0.79
PersonPlaceOfDeath	0.82	0.84
PersonProfession	0.56	0.58
RiverBasinsCountry	0.83	0.82
StateSharesBorderState	0.47	0.52
Average	0.66	0.63

Alivanistos, Báez Santamaría, Cochez, Kalo, van Krieken, Thanapalasingam, Prompting as Probing: Using Language Models for Knowledge Base Construction

Shortcomings of the Model

- **Timeliness:** Language models often are outdated
 - Current employers of persons were often missing
 - Updating language models consistently is difficult
- Using the model is **expensive**
 - We paid around 5\$ dollars per run!
- Entities are just **strings**
 - Predictions might be correct, but are using the wrong entity name

Shortcomings of the Dataset

- **Missing aliases**
 - Ground Truth: ["national aeronautics and space administration"]
 - Our prediction: ["NASA"]
- **Incompleteness of the dataset**
 - e.g. PersonEmployer is incomplete
 - Guido van Rossum: only current employer Microsoft is mentioned
 - Dick Costolo: mentions two previous employers Twitter and Google

Improving results with Wikidata



Conclusions

- GPT-3 knows a lot!
- Large models perform better than smaller models
 - But they are very slow and expensive to work with
- Small changes in the prompt have large impacts on the results
 - Triple prompts mostly perform better
- Knowledge Graphs can be used to improve results

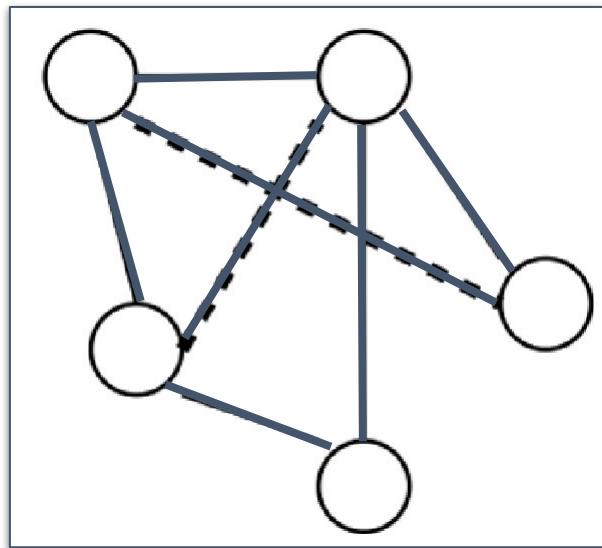
GPT–3 performs pretty good for KBC, but is far from perfect.



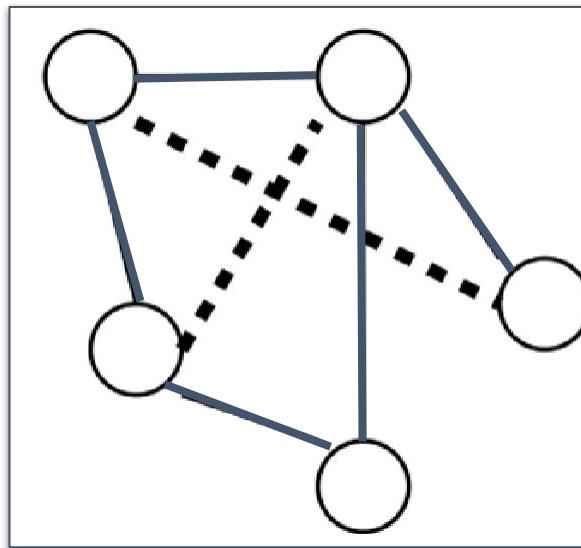
Part 6. The Future

How are we evaluating?

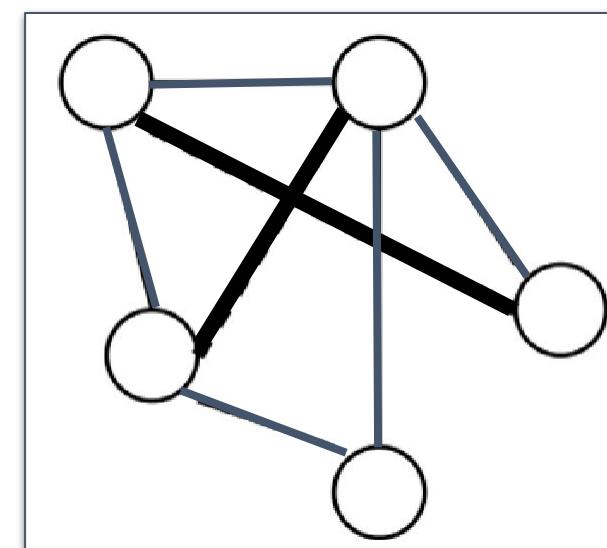
- We usually train and test on the same datasets!
 - Both for normal link prediction and nearly all domains we discussed today



Original graph



Training graph



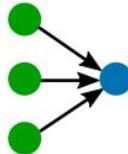
Test graph

Is this problematic?

- Yes, does what we do correspond to reality?
 - Randomly removed edges ≠ Edges which are missing
 - Missing edges are missing in a more structured way
 - All professions are missing for students
 - All birth places are missing for people who applied for a position, but are not hired.
 - Edges could be missing because they are not true yet, but will be true in the future.
 - Time based splits vs. a specific time
- So, this might overestimate performance

Inspired by an internally shared opinion piece by Frank van Harmelen, with help from many people

Is this problematic?

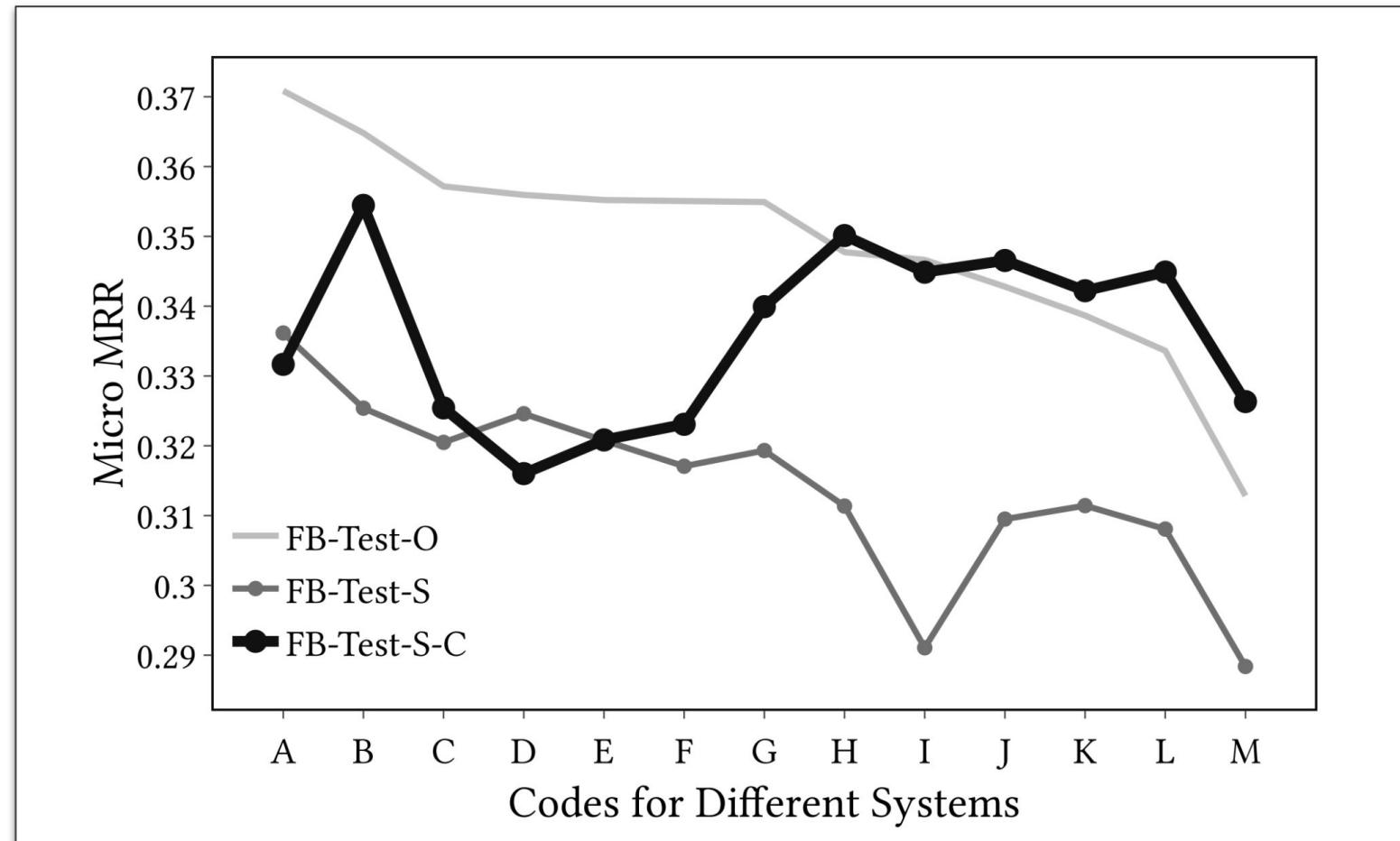
- Yes, our metrics are biased by high degree nodes?
 - For link prediction, a high in-degree node is a more common answer
 - When creating derived datasets, this can become extreme
 - For 3-i queries
 - For a node with in-degree n , creating all queries with this pattern gives $\binom{n}{3}$ different queries with this node as its answer.
 - A node of degree 3 leads to 1 query
 - A node of degree 5 leads to 10 queries
 - A node of degree 10 leads to 120 queries
 - A node of degree 4,424 leads to 14,421,138,424 queries
 - The total number of 3-i queries in the dataset was 38,011,148,464
 - 38% of queries had this high degree node as their answer!
 - Always returning the 10 nodes with highest degree as an answer would give a hits@10 of 0.93
 - Without any learning at all!

Is this problematic?

- Yes! Our metrics do not necessarily reflect performance on TRUE triples which are unknown
 - It is provable that a model exists which performs excellent on the usual metrics, but still does very bad on triples not in the test set.
 - Open question: Find a bound on the guarantees a specific model could provide on the unknown triple set.

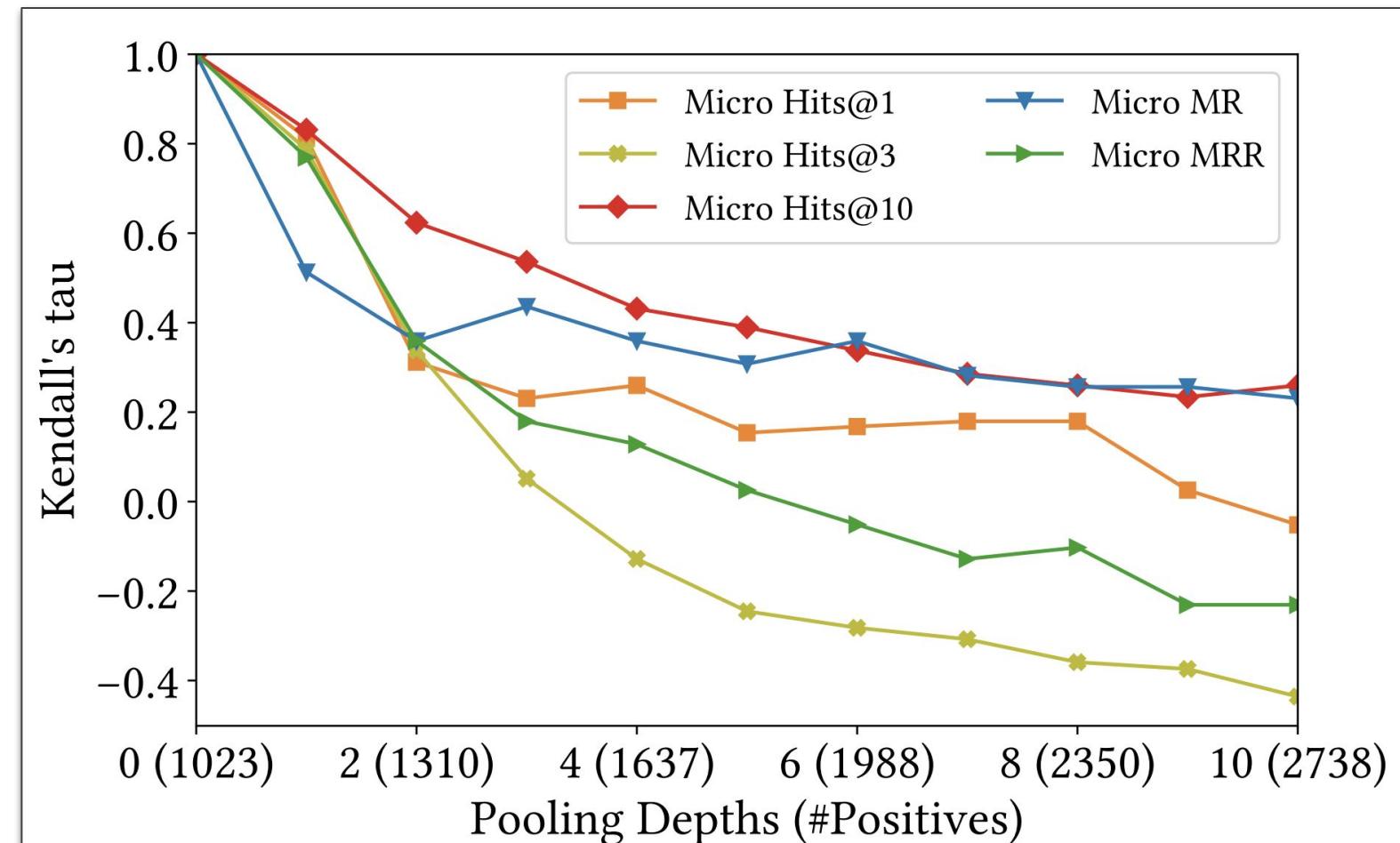
Is this problematic?

- Yes! Our datasets are not necessarily good!
 - Correcting edges in datasets does lead to different results!



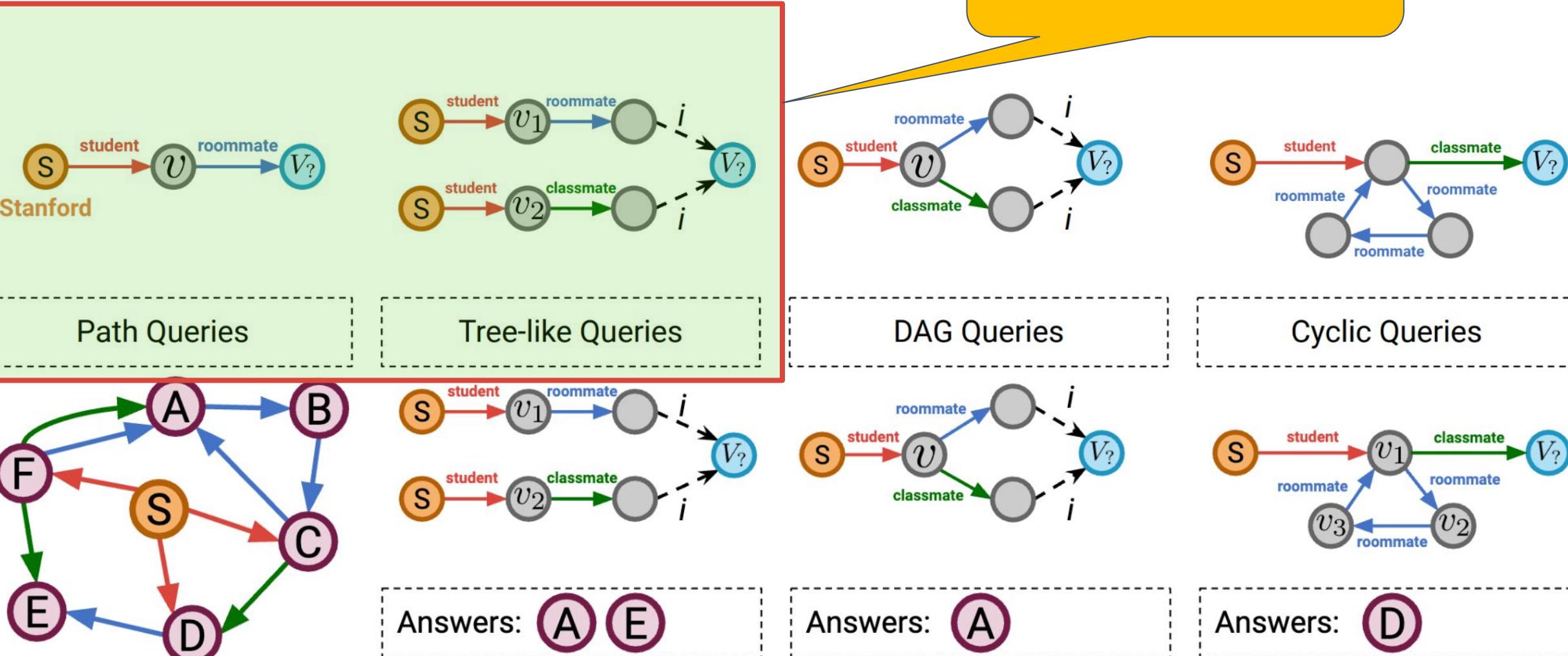
Is this problematic?

- Yes! Our datasets are not necessarily good!
 - Correcting edges in datasets does lead to different results!
 - And different ranking of how well systems perform!

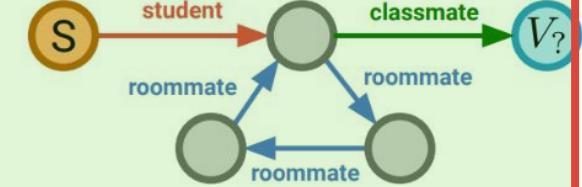
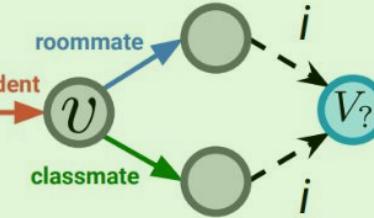
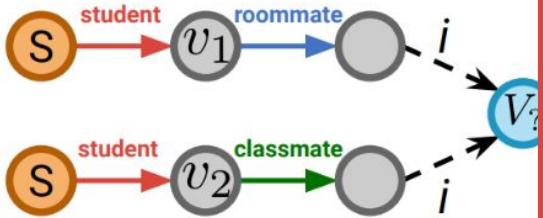
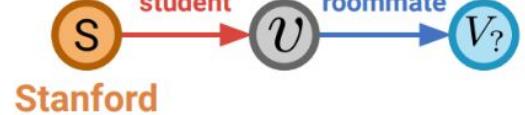


Query Patterns: Easy \rightarrow Hard

Most works focus on these



Query Patterns: Easy -> Hard

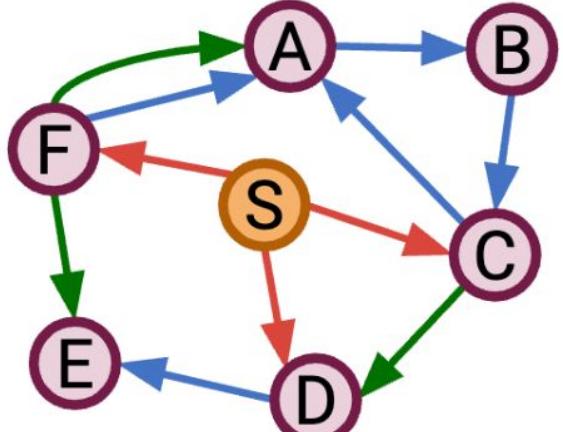


Path Queries

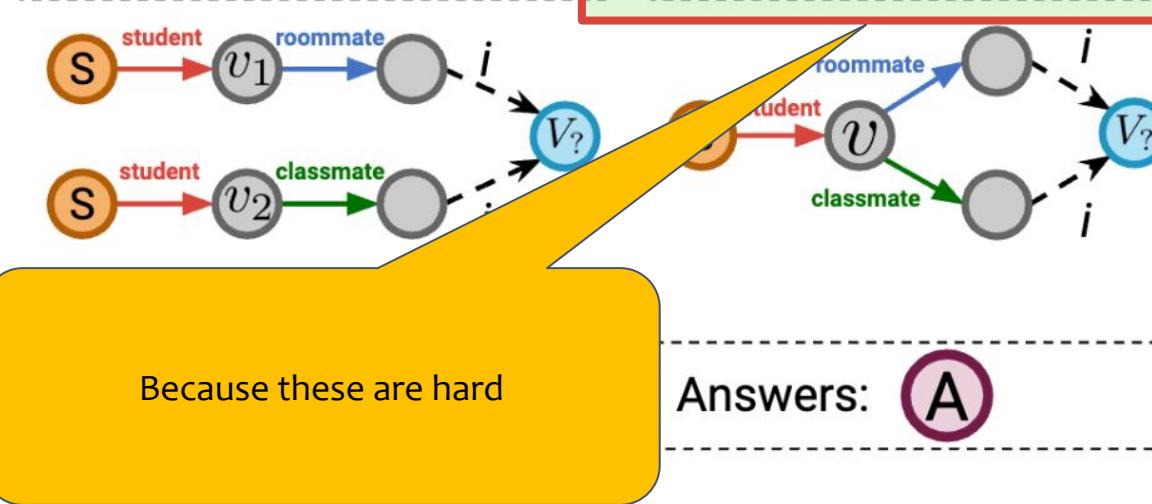
Tree-like Queries

DAG Queries

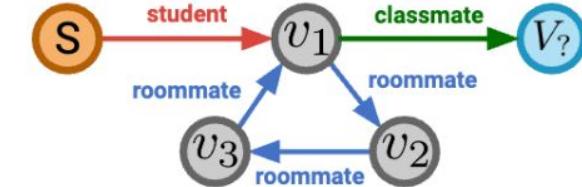
Cyclic Queries



Because these are hard



Answers: A



Answers: D

We predict Rankings

- Argued that this is because of incomplete graphs
- But, this also means that any downstream application has to do the cut-off
 - Only useful for information retrieval tasks
 - Rankings are **not** calibrated
 - So applications cannot even put a fixed threshold!
- We need to start evaluating with binary classification!

Most works focus on vertices/nodes only

- Real graphs contain all sorts of modalities
 - Numeric
 - Textual
 - Images
 - Video
 - Live measurements
- Relations in real graphs are more complex
 - Hyper-relational
 - Weights / uncertainty
 - Signed graphs
 - Mix of directed and undirected

Most works do not take semantics into account

- The meaning of things is important!
 - And can be exploited
 - For example, transitivity
- If we do not take this into account
 - We get an inconsistent result

Many methods do not think about scalability!

- Many benchmark datasets are toy graphs
 - A real world KG starts at 10M nodes
- Tasks get much harder
 - More candidates for link prediction
- Methods do not scale
 - Memory is a limited resource
 - Methods needed more than $O(n^2)$ will not work

What to do with LLMs?



- Often results in good performance
- Flexible integrations
 - Natural language as an interface between different systems



- Unclear whether they generalize
- Unclear bounds
- Hardly any theory

What to do?

- More investigations of what our datasets really contain
- More focus on methods, less focus on specific datasets
- If we do experiments, more datasets, more realism in the benchmarks, larger graphs

What to do?

- Work on graphs which have more complex relations
- Work on real graphs, not toy graphs, think about scalability
- More experiments with downstream tasks to show that the added information is meaningful - more scrutiny with reasoning, logic or other symbolic methods to determine whether the new links are really new
- Work towards understanding, more theory on top of experimental evaluation, stop SoTa hunting (small differences are not informative)



Reasoning Beyond Triples: Recent Advances in Knowledge Graph Embeddings



Bo Xiong ¹



Mojtaba Nayyeri ¹



Daniel Daza ^{2,3}



Michael Cochez ²



¹ University of Stuttgart ² Vrije Universiteit Amsterdam ³ University of Amsterdam

Saturday 21 October, 09:00 - 17:30 (GMT+1) @ University of Birmingham - Birmingham, UK



Thank you