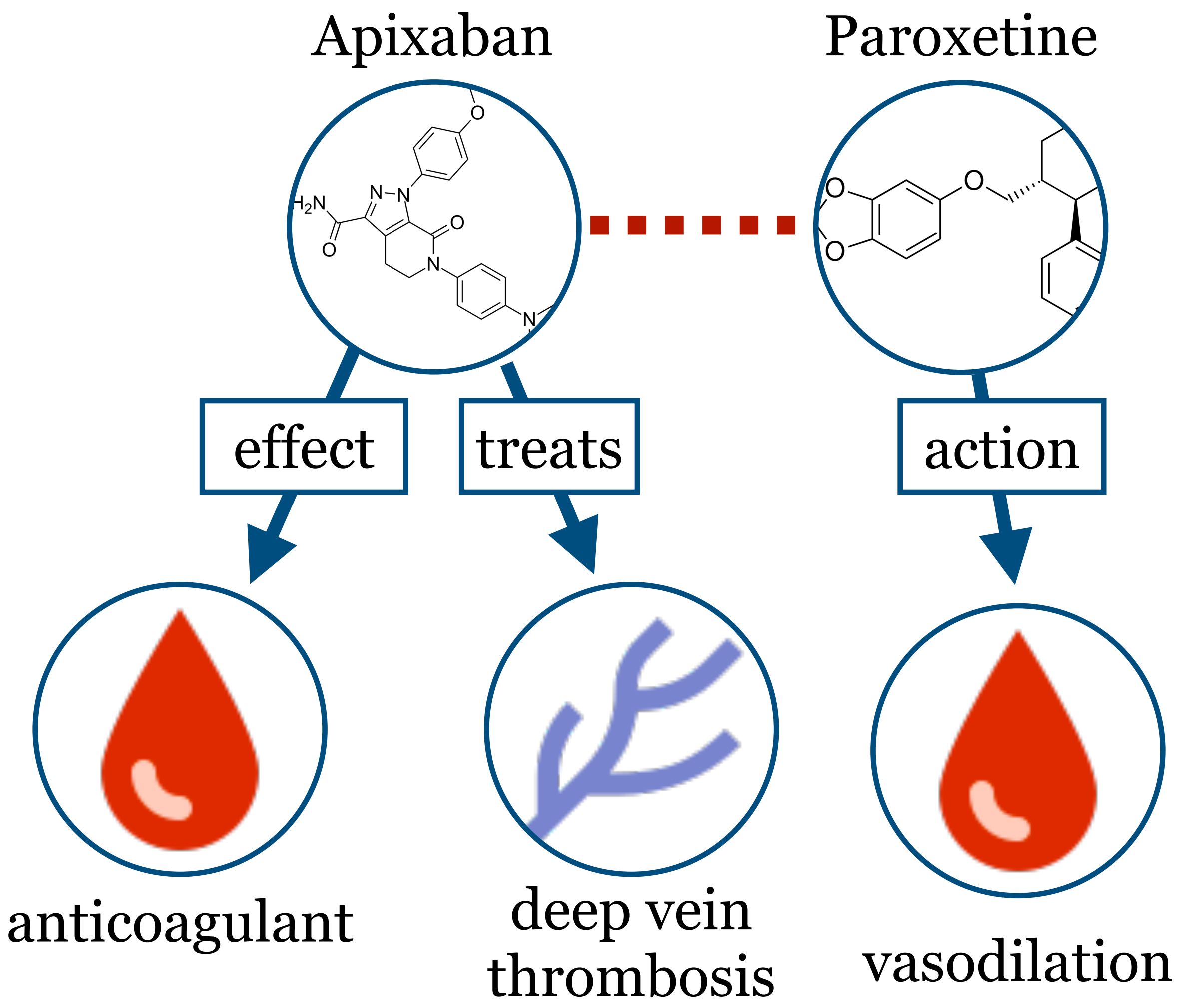
Complex Query Answering

Pasquale Minervini
p.minervini@ed.ac.uk

Main References

1. Approximate Answering of Graph Queries — https://arxiv.org/abs/2308.06585
2. Reasoning Beyond Triples: Recent Advances in Knowledge Graph Embeddings (CIKM 2023 Tutorial) — https://kg-beyond-triple.github.io/
3. Neural Graph Reasoning: Complex Logical Query Answering Meets Graph Databases — https://arxiv.org/abs/2303.14617
4. Complex Query Answering with Neural Link Predictors — https://arxiv.org/abs/2011.03459

Neural Link Predictors



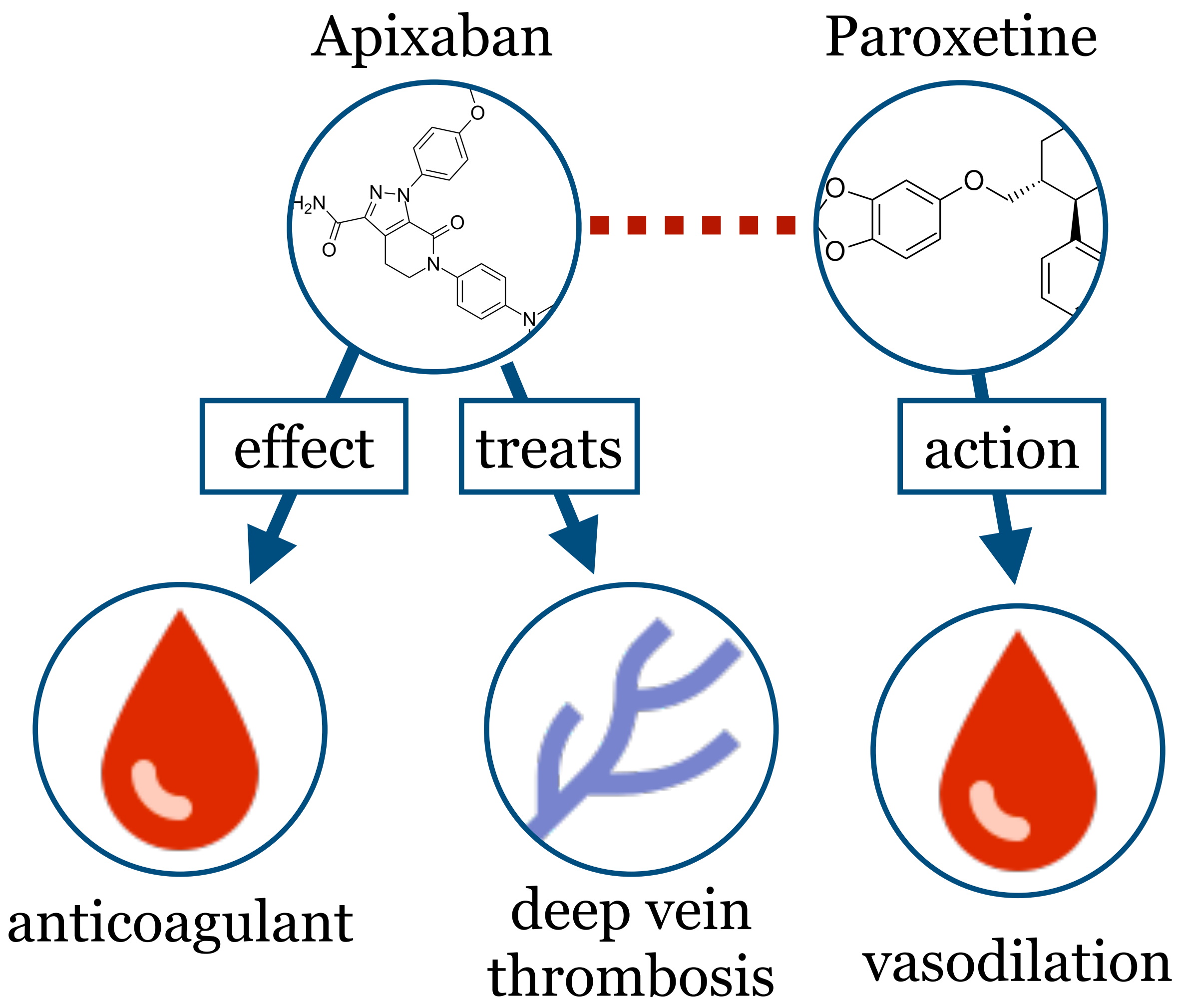
They map entities and relation types to *embedding* vectors.

$$\mathbf{e}_{\text{Apixaban}} \in \mathbb{R}^{d} \quad \mathbf{r}_{\text{treats}} \in \mathbb{R}^{d'};$$

They train embeddings so that links in the graph will have an higher score than links not in the graph;

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

Neural Link Predictors

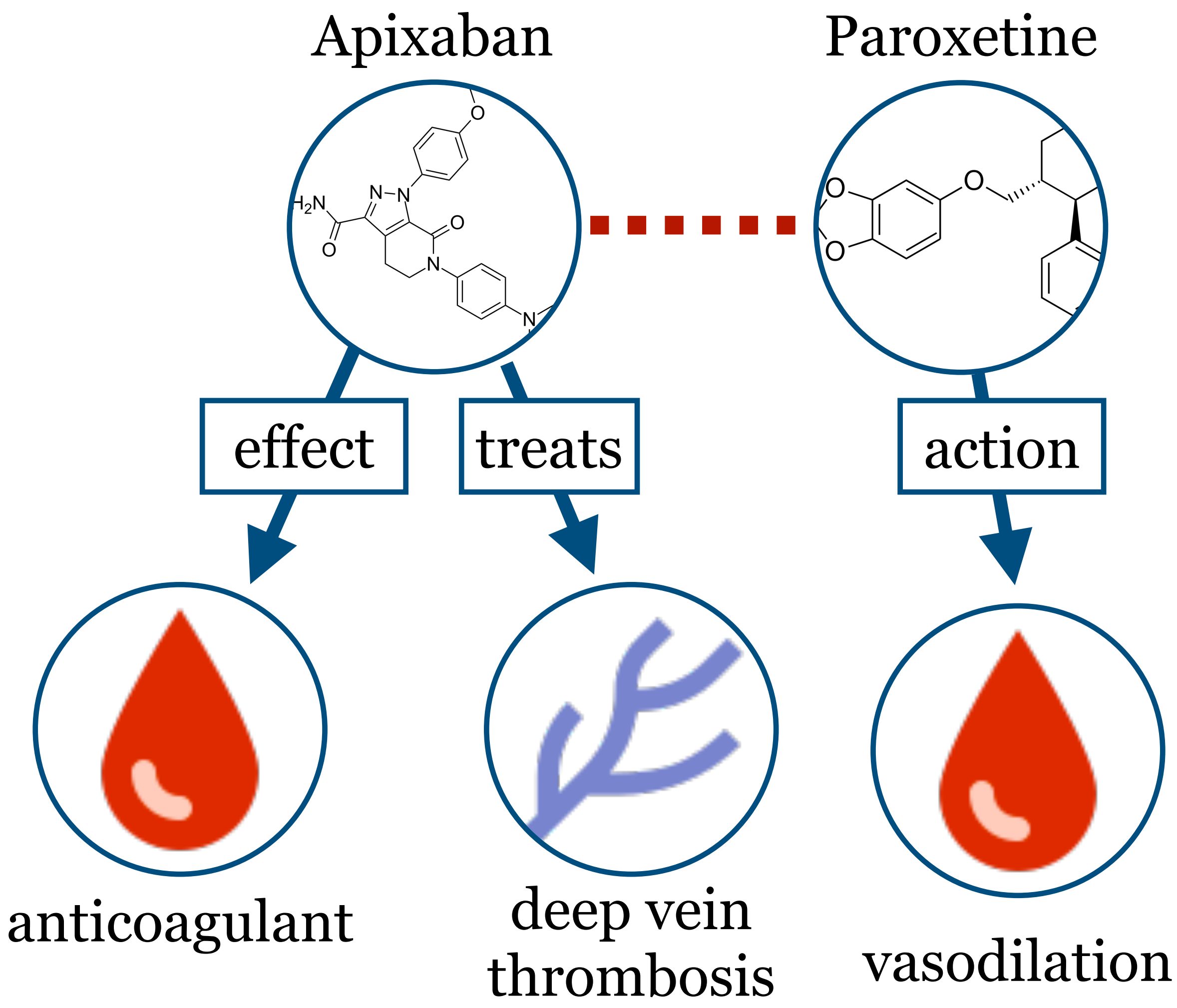


We can answer queries like "*what are the drugs that interact with Apixaban?*"

$$?D : \text{interactsWith}(\text{Apixaban}, D)$$

These are atomic queries, i.e., queries that do not contain any logical connectives (like AND, OR, NOT).

Complex Queries?



However, what if we want to answer more complex queries? For example:

$$?D : \text{interactsWith}(\text{Apixaban}, D) \wedge \text{action}(D, \text{vasodilation})$$

$$?D : \exists D' . \text{interactsWith}(\text{Apixaban}, D') \wedge \text{interactsWith}(D', D)$$

Complex Query Answering

Two main approaches:

Neural Query Embedding

Embeds the query and candidate answers, and matches them

Query Decomposition

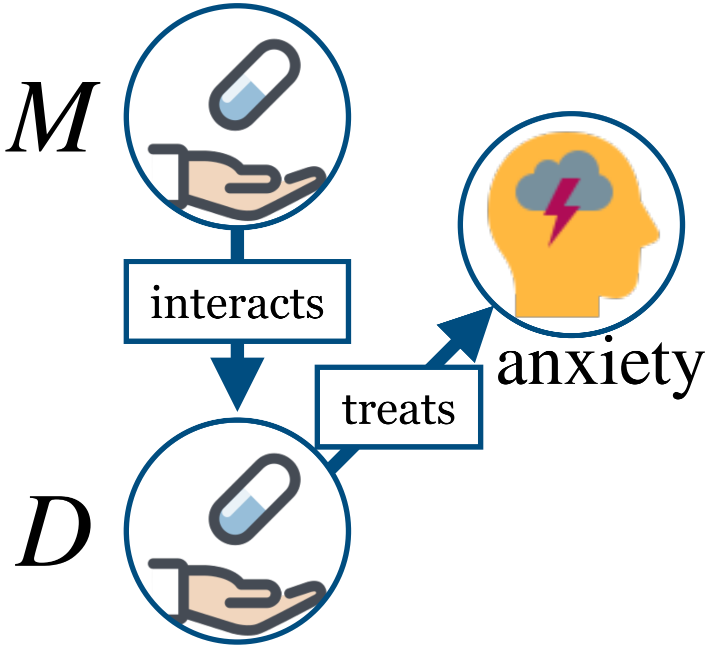
Decomposes the query into atomic sub-queries

Neural Query Embedding

Neural Query Embedding

Query: Which medications have side-effects when taken with drugs for treating Anxiety?

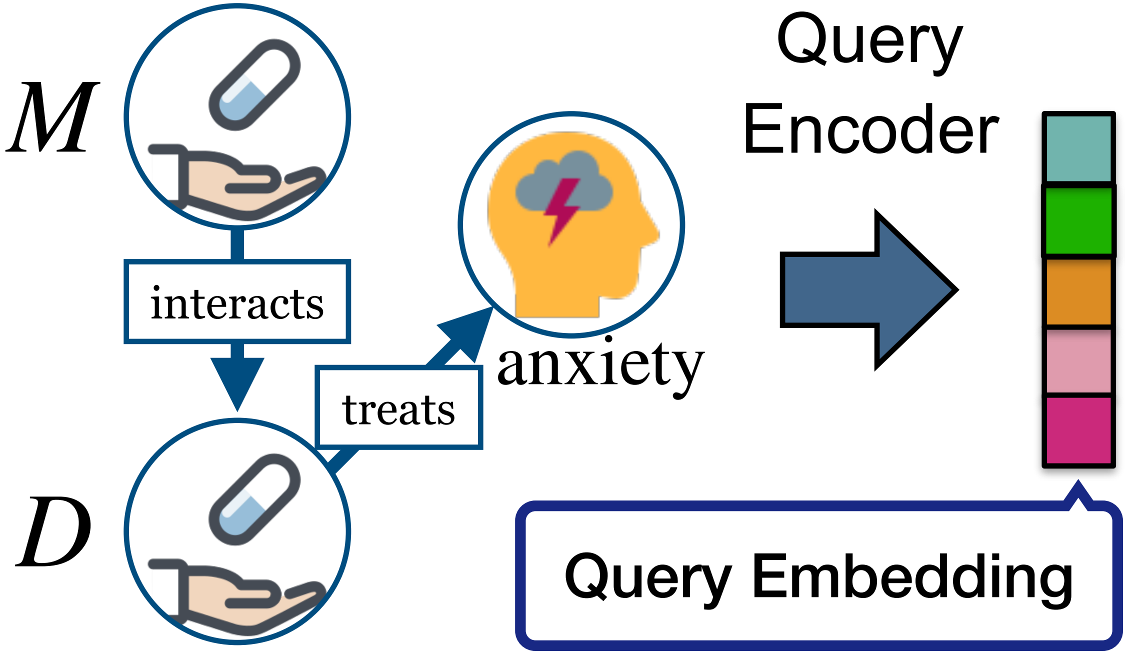
$$?M : \exists D \,.\, \text{interacts}\,(M, D) \wedge \text{treats}\,\big(D, \text{anxiety}\big)$$



Neural Query Embedding

Query: Which medications have side-effects when taken with drugs for treating Anxiety?

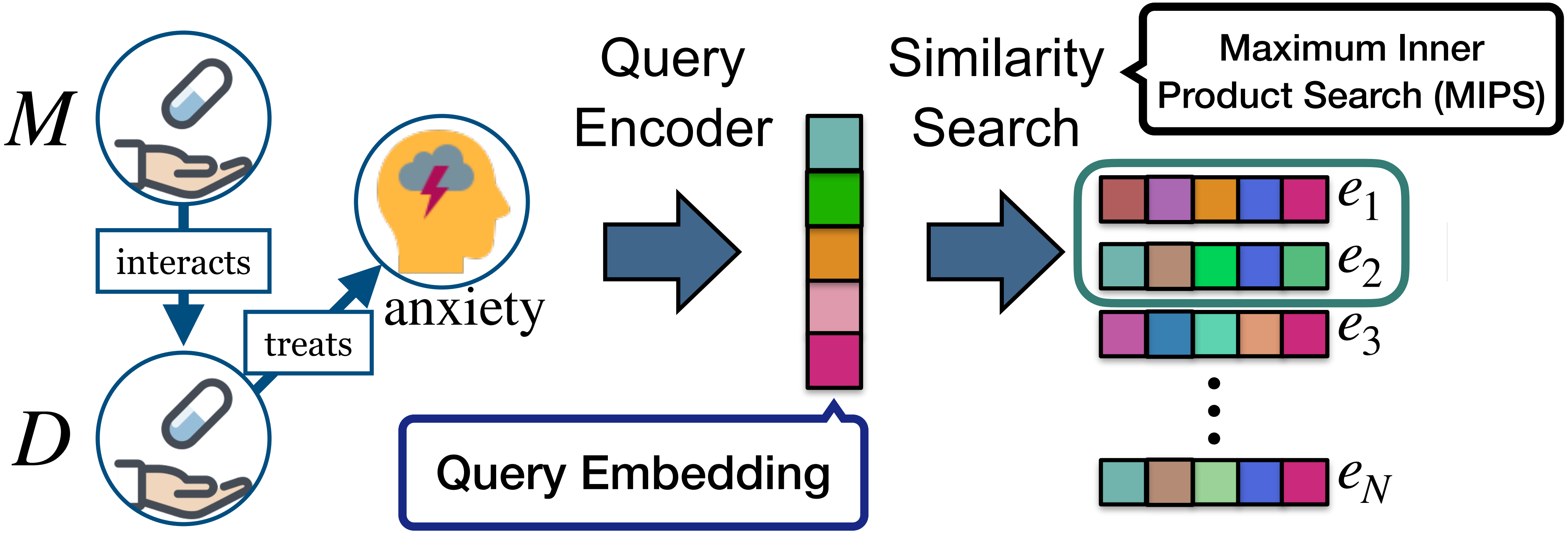
$$?M : \exists D \,.\, \text{interacts}\,(M, D) \wedge \text{treats}\,\big(D, \text{anxiety}\big)$$



Neural Query Embedding

Query: Which medications have side-effects when taken with drugs for treating Anxiety?

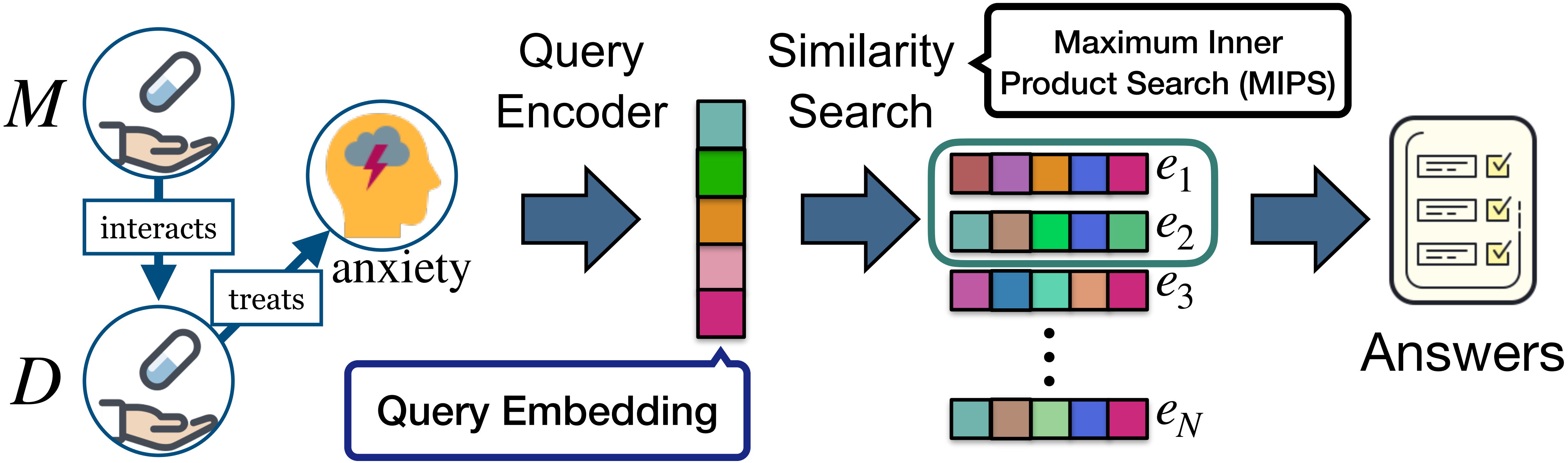
$$?M : \exists D \,.\, \text{interacts}\,(M, D) \wedge \text{treats}\,\big(D, \text{anxiety}\big)$$



Neural Query Embedding

Query: Which medications have side-effects when taken with drugs for treating Anxiety?

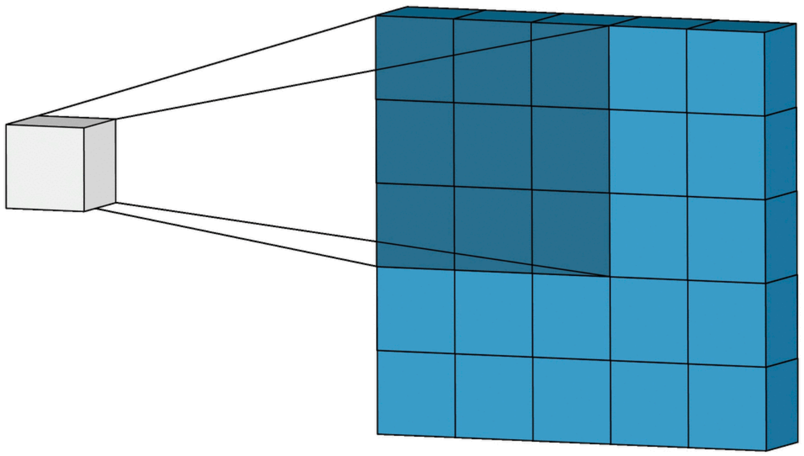
$$?M : \exists D \,.\, \text{interacts}\,(M, D) \wedge \text{treats}\,\big(D, \text{anxiety}\big)$$



The Query (Graph) Encoder

Graphs can be seen as a strict generalisation of **images**:

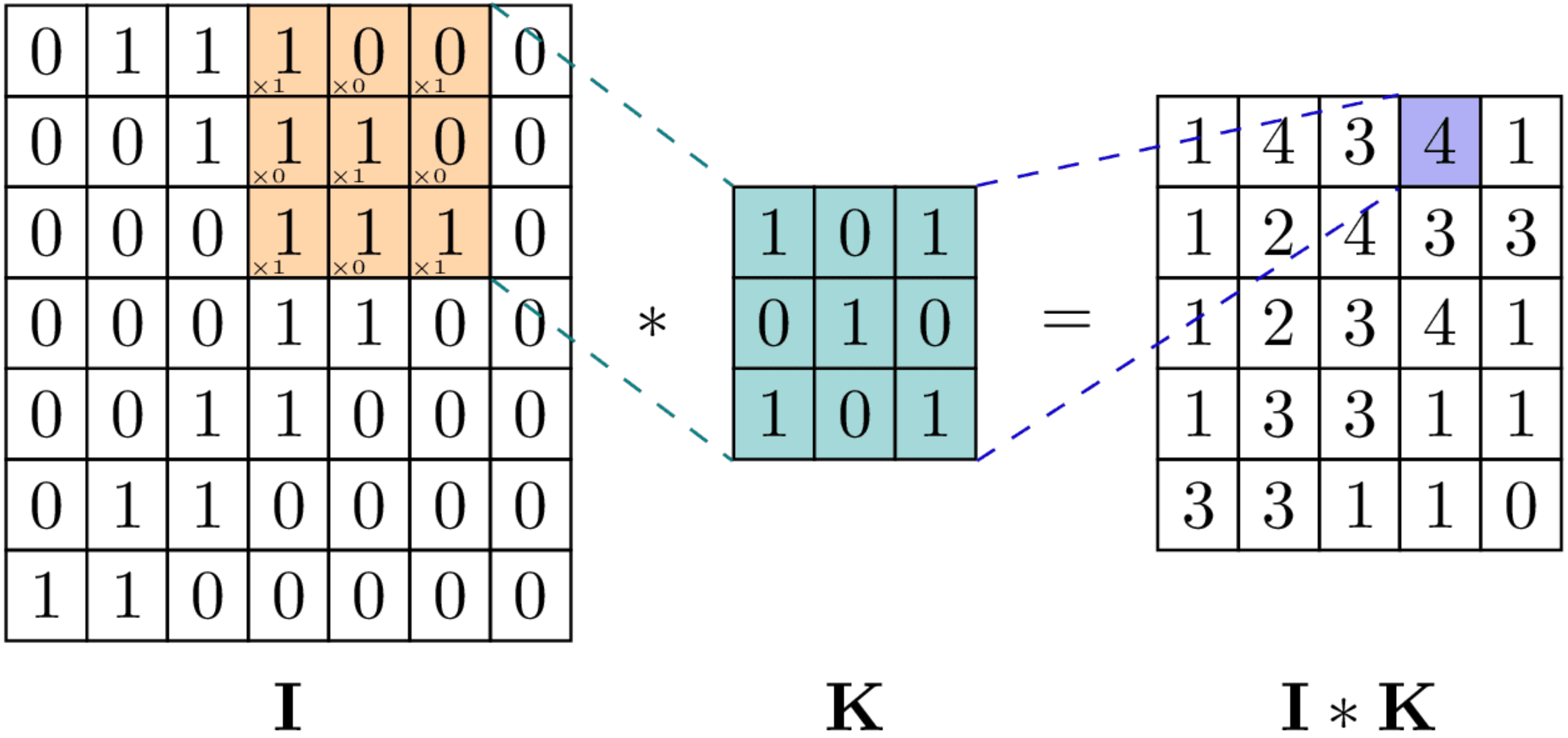
- An image can be seen as a “grid graph”,
- Each node corresponds to a pixel, adjacent to its four neighbours.



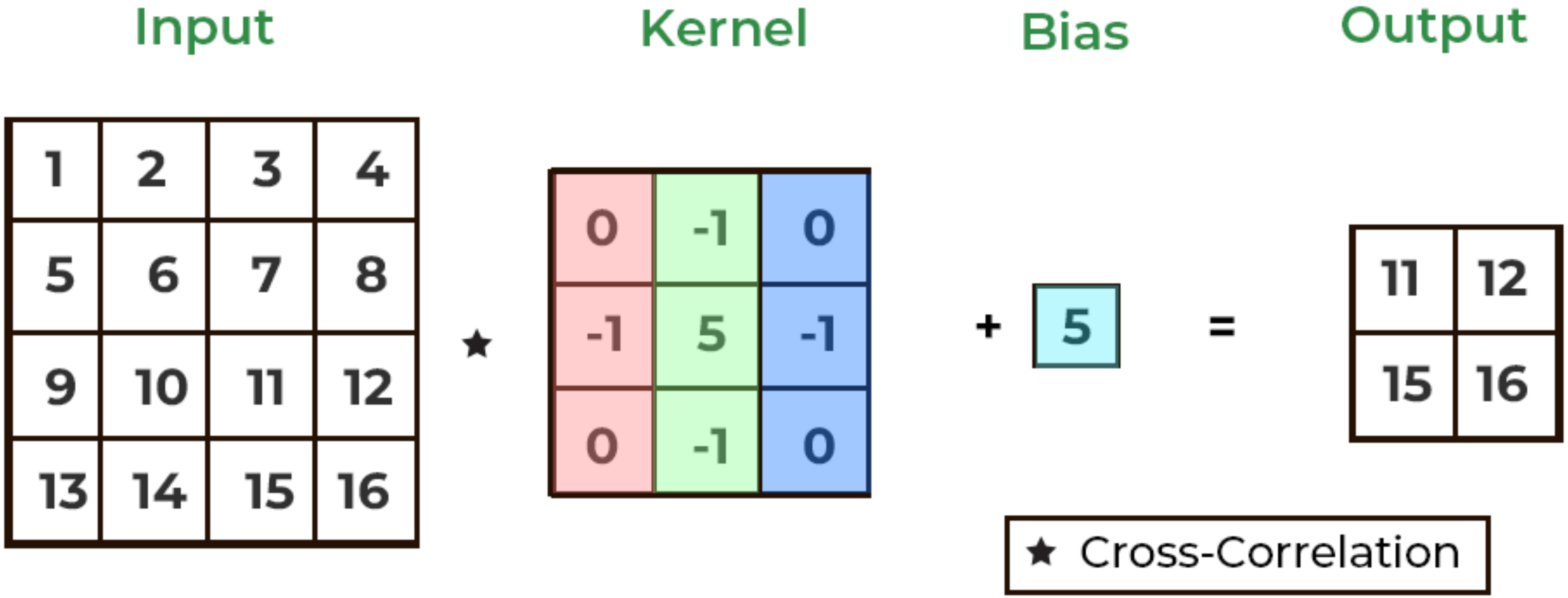
Convolutional Neural Networks leverage the **convolutional operator** to extract the spatial regularity in images

Can we generalise this operator to operate on **arbitrary graphs?**

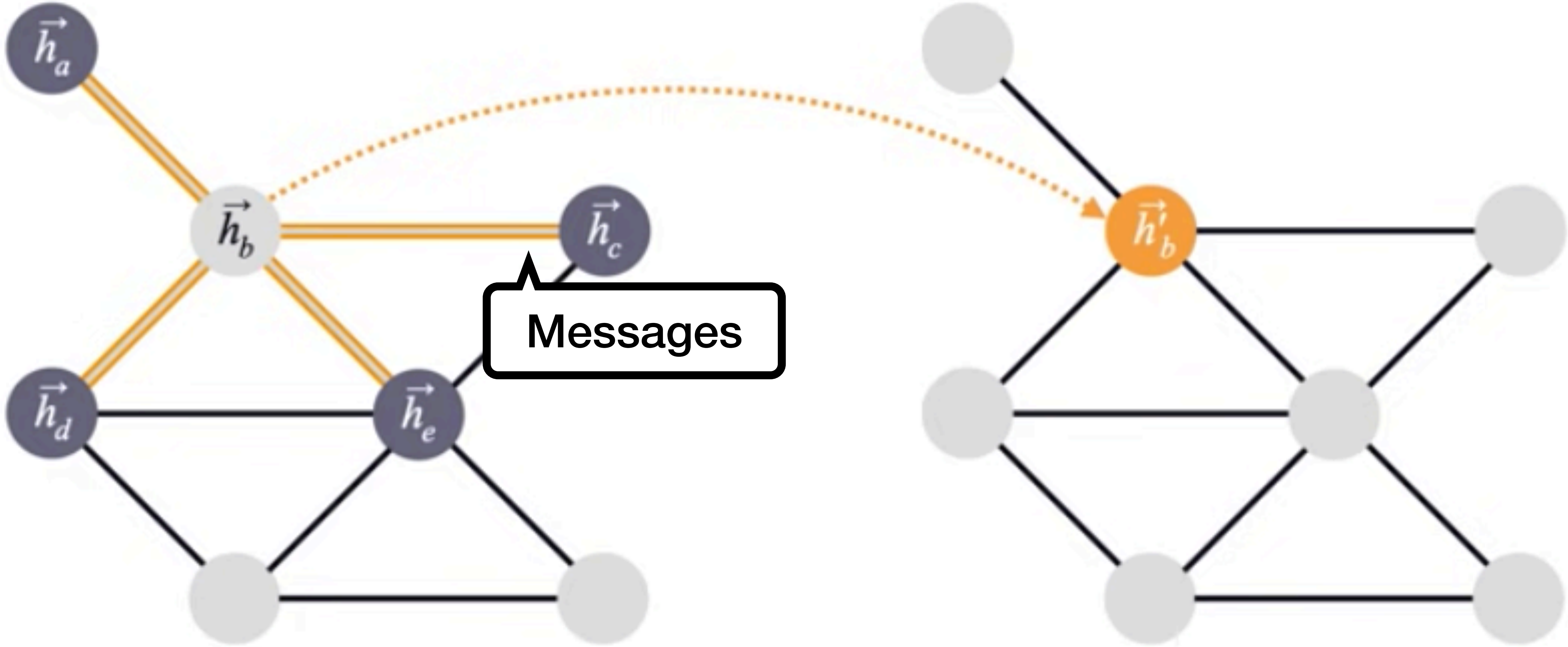
(2D) Convolution on Images



(2D) Convolution on Images

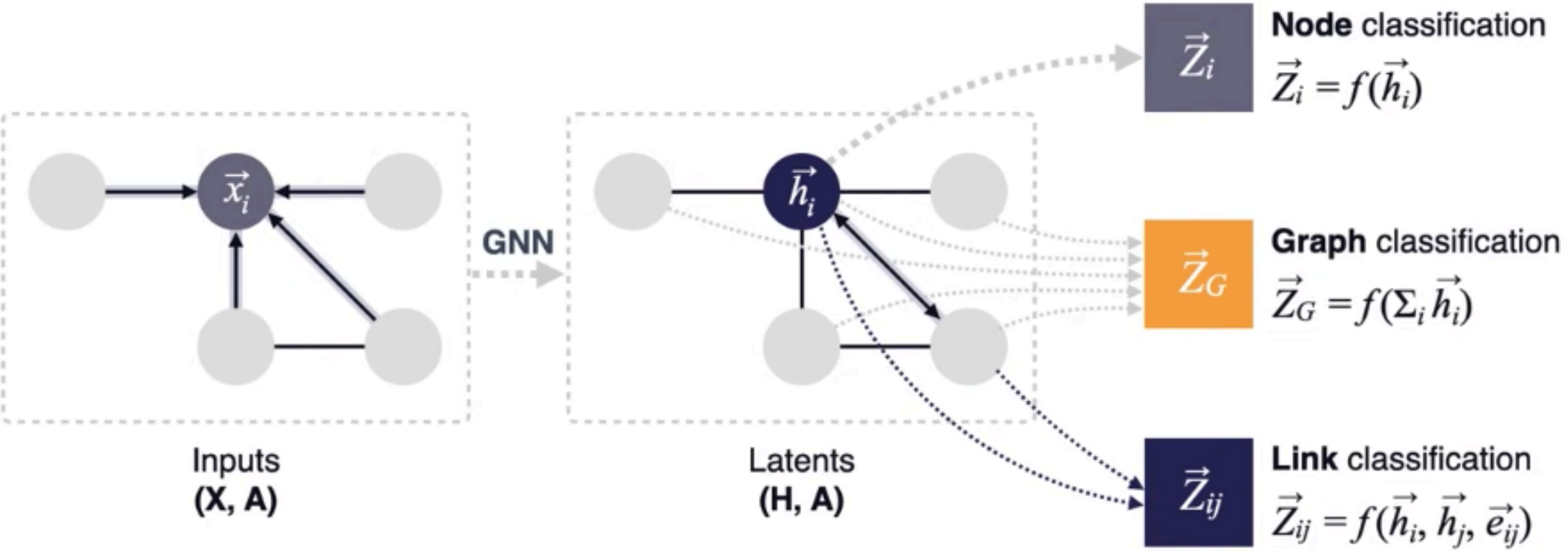


Convolution on Graphs



$\vec{h}'_i = g(\vec{h}_a, \vec{h}_b, \vec{h}_c, \ldots)$ $\quad (a, b, c, \ldots \in N_i)$

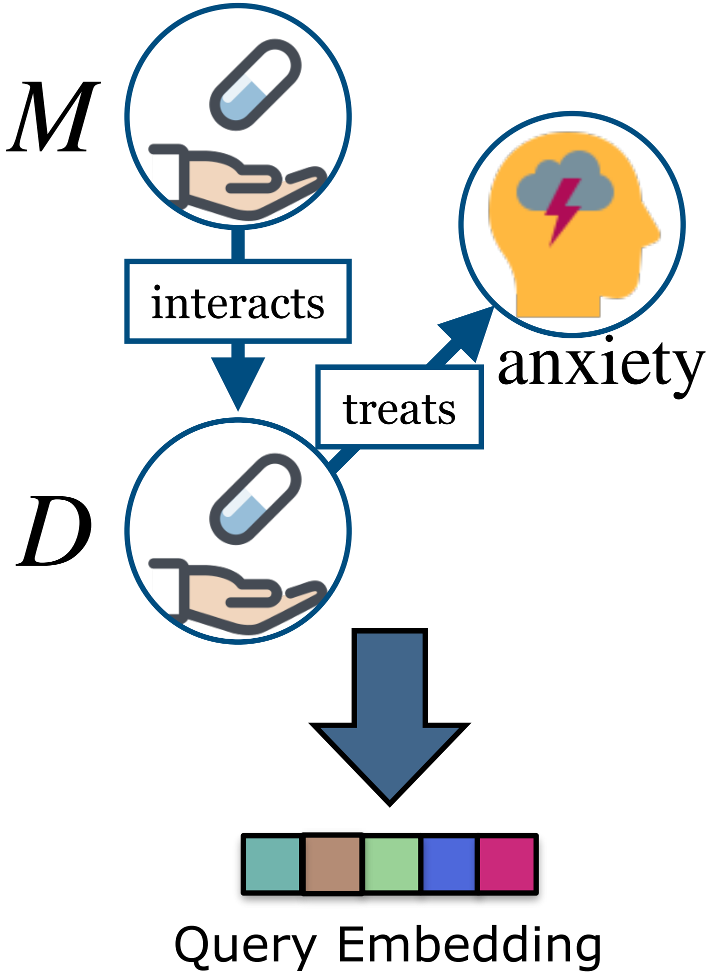
Convolution on Graphs



The Query (Graph) Encoder

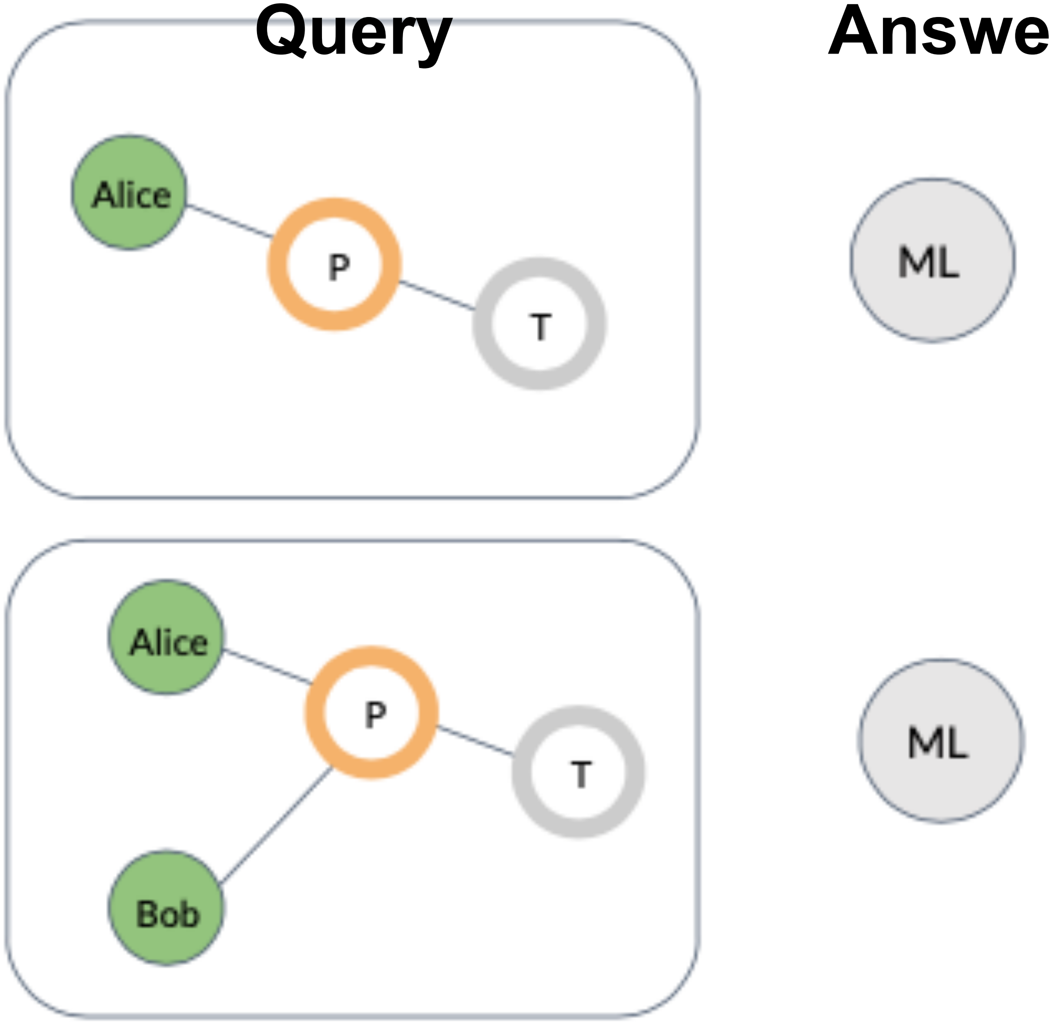
Graph Encoder models operate on graphs, by applying message passing:

- Learnable parameters include **entity** and **variable** node embeddings;
- Messages propagate across the query graph;
- After k steps of message passing, map all node representations to a single query embedding.



Training

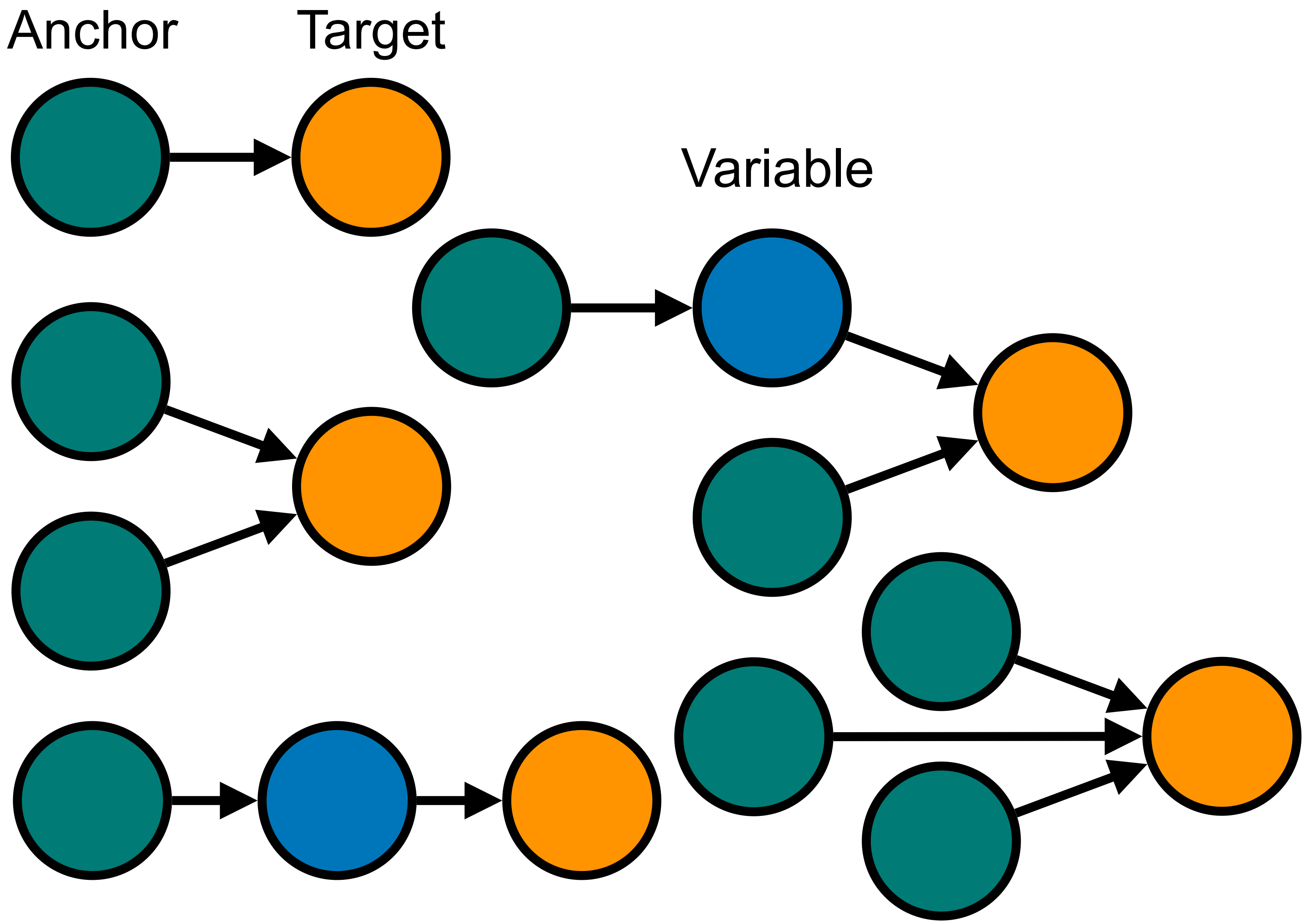
Query Answer



Trained on automatically generated complex query-answer pairs:

- Sample sub-graphs
- Replace some entities by variables
- Train the model parameters (entity and relation embeddings, query encoder, etc.) via gradient-based optimisation methods, by maximising the likelihood of the training answers.

Evaluation



Models often evaluated on a set of **~14 query structures**.

Given a set of query-answer pairs, evaluate using:

- Mean Reciprocal Rank (MRR)
- Hits@k

Neural Query Embedding Methods

Pros

- Each query is embedded in a single vector, and query answering has complexity $\mathcal{O}(n)$
- We can encode arbitrary query types
- Scales well, since only small graphs are encoded each time

Cons

- The query encoder is a **black box**
- Requires training on a potentially very large amount of query-answer pairs

Query Decomposition

Announcing ICLR 2021 Outstanding Paper Awards

ICLR · Follow

2 min read · Mar 31, 2021

- Beyond Fully-Connected Layers with Quaternions: Parameterization of Hypercomplex Multiplications with 1/n Parameters by Aston Zhang, Yi Tay, Shuai Zhang, Alvin Chan, Anh Tuan Luu, Siu Hui, and Jie Fu
- Complex Query Answering with Neural Link Predictors by Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez
- EigenGame: PCA as a Nash Equilibrium by Ian Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel

Complex Query Decomposition

- Train a neural link predictor (e.g., **ComplEx**)

$$?M : \exists D \,.\, \text{interacts}\,(M, D) \wedge \text{treats}\left(D, \text{anxiety}\right)$$

Arakelyan et al. Complex Query Answering with Neural Link Predictors. ICLR 2021

Complex Query Decomposition

- Train a neural link predictor (e.g., **ComplEx**)
- Answer complex queries by **decomposing** them in **atomic sub-queries**, and **answering them using the neural link predictor**

$$?M : \exists D \,.\, \text{interacts}\,(M, D) \wedge \text{treats}\,\left(D, \text{anxiety}\right)$$

Link Predictor

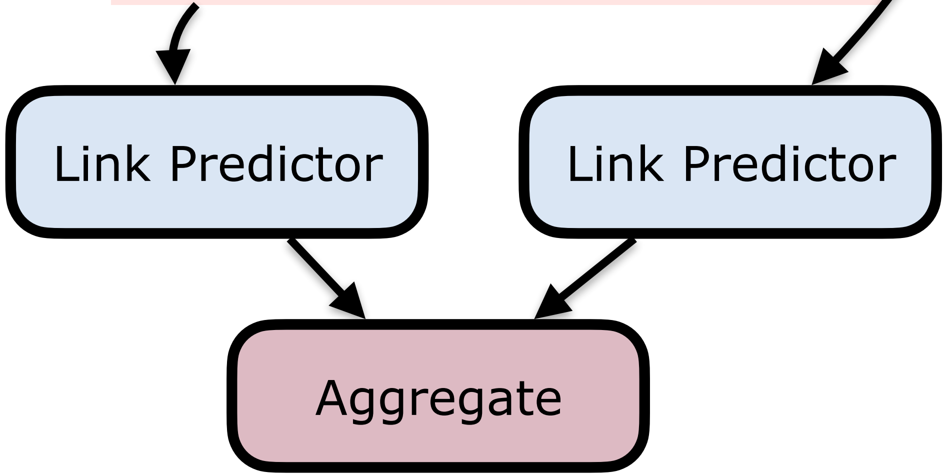
Link Predictor

Arakelyan et al. Complex Query Answering with Neural Link Predictors. ICLR 2021

Complex Query Decomposition

- Train a neural link predictor (e.g., **ComplEx**)
- Answer complex queries by **decomposing** them in **atomic sub-queries**, and **answering them using the neural link predictor**
- **Aggregate the answers** to the sub-queries

$$?M : \exists D \,.\, \text{interacts}(M, D) \wedge \text{treats}\left(D, \text{anxiety}\right)$$

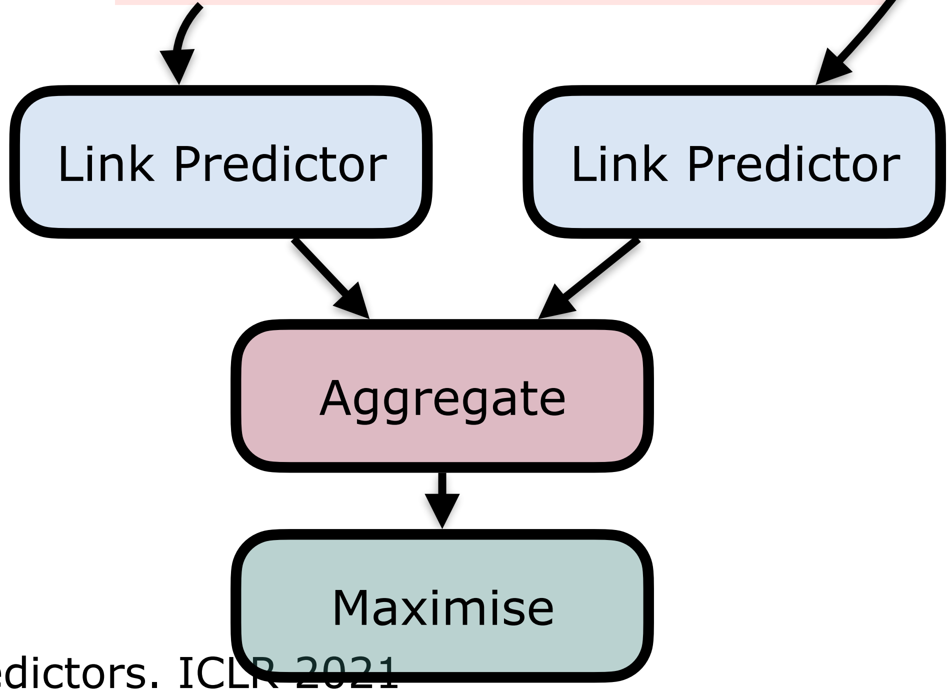


Arakelyan et al. Complex Query Answering with Neural Link Predictors. ICLR 2021

Complex Query Decomposition

- Train a neural link predictor (e.g., **ComplEx**)
- Answer complex queries by **decomposing** them in **atomic sub-queries**, and **answering them using the neural link predictor**
- **Aggregate the answers** to the sub-queries
- Identify the variable assignments that maximise the score of the query

$$?M : \exists D \,.\, \text{interacts}\,(M, D) \wedge \text{treats}\,\left(D, \text{anxiety}\right)$$



Arakelyan et al. Complex Query Answering with Neural Link Predictors. ICLR 2021

Complex Query Decomposition

Query: Which medications have side-effects when taken with drugs for treating Anxiety?

$$?M : \exists D \,.\, \text{interacts}\,(M, D) \wedge \text{treats}\,\big(D, \text{anxiety}\big)$$

Complex Query Decomposition

Proposed solution: train a neural model ϕ for answering atomic (simple) queries (e.g. “which drugs treat Anxiety?”), and cast the query answering task as an *optimisation problem:*

$$?M : \exists D \,.\, \text{interacts}\,(M, D) \land \text{treats}\,\big(D, \text{anxiety}\big)$$

Complex Query Decomposition

Proposed solution: train a neural model ϕ for answering atomic (simple) queries (e.g. "which drugs treat Anxiety?"), and cast the query answering task as an *optimisation problem:*

$$?M : \exists D \,.\, \text{interacts}\,(M, D) \wedge \text{treats}\,\big(D, \text{anxiety}\big)$$

Optimisation problem

$$\arg\max_{M,D \in \mathcal{E}} \Big[\qquad \Big]$$

Complex Query Decomposition

Proposed solution: train a neural model ϕ for answering atomic (simple) queries (e.g. “which drugs treat Anxiety?”), and cast the query answering task as an *optimisation problem:*

$$?M : \exists D \,.\, \text{interacts}\,(M, D) \wedge \text{treats}\,\left(D, \text{anxiety}\right)$$

Optimisation problem

$$\underset{M,D \in \mathscr{E}}{\arg\max} \left[\phi_{\text{interacts}}\left(\mathbf{e}_M, \mathbf{e}_D\right) \quad \phi_{\text{treats}}\left(\mathbf{e}_D, \mathbf{e}_{\text{anxiety}}\right) \right]$$

Likelihood that M interacts with D

Likelihood that D treats anxiety

Complex Query Decomposition

Proposed solution: train a neural model ϕ for answering atomic (simple) queries (e.g. "which drugs treat Anxiety?"), and cast the query answering task as an *optimisation problem:*

$$?M : \exists D \,.\, \text{interacts}\,(M, D) \wedge \text{treats}\left(D, \text{anxiety}\right)$$

Optimisation problem

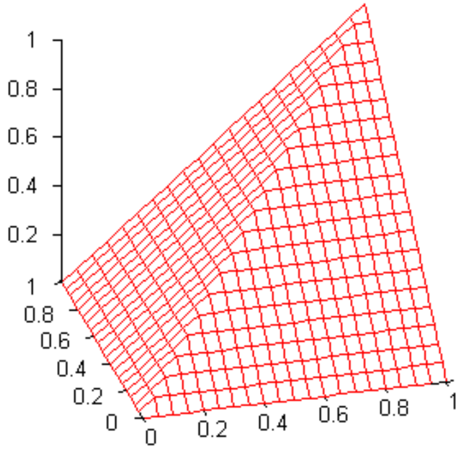
Continuous relaxation of the logical AND (**t-norm**)

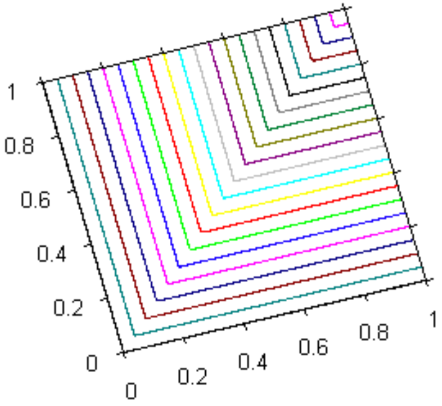
$$\arg\max_{M,D \in \mathcal{E}} \left[\phi_{\text{interacts}}\left(\mathbf{e}_M, \mathbf{e}_D\right) \top \phi_{\text{treats}}\left(\mathbf{e}_D, \mathbf{e}_{\text{anxiety}}\right) \right]$$

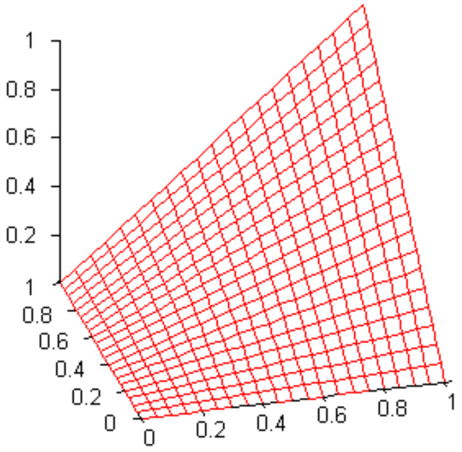
Likelihood that M interacts with D

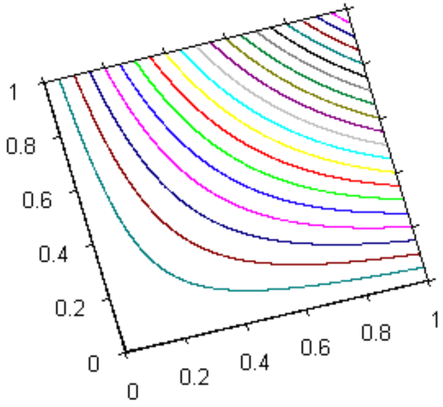
Likelihood that D treats anxiety

T-Norms and T-Conorms









Minimum (or Gödel) t-norm:

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

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

Product t-norm:

$$A \wedge B \mapsto \top_{\text{product}}(a,b) = ab$$

$$A \vee B \mapsto \perp_{\text{sum}}(a,b) = a + b - ab$$

Complementary t-conorm:

$$\perp(a,b) = 1 - \top(1-a, 1-b)$$

Complex Query Answering as Search

$$\arg\max_{M,D\in\mathscr{E}} \left[\phi_{\text{interacts}}\left(\mathbf{e}_M, \mathbf{e}_D\right) \top \phi_{\text{treats}}\left(\mathbf{e}_D, \mathbf{e}_{\text{anxiety}}\right)\right]$$

Greedy Search

Complex Query Answering as Search

$$\arg\max_{M,D\in\mathscr{E}} \left[\phi_{\text{interacts}}\left(\mathbf{e}_M, \mathbf{e}_D\right) \top \phi_{\text{treats}}\left(\mathbf{e}_D, \mathbf{e}_{\text{anxiety}}\right)\right]$$

Greedy Search

- Identify the k most likely values for D

Complex Query Answering as Search

$$\arg\max_{M,D \in \mathcal{E}} \left[\phi_{\text{interacts}}\left(\mathbf{e}_M, \mathbf{e}_D\right) \top \phi_{\text{treats}}\left(\mathbf{e}_D, \mathbf{e}_{\text{anxiety}}\right) \right]$$

Greedy Search

- Identify the k most likely values for D
- For each value of D:

Complex Query Answering as Search

$$\arg\max_{M,D\in\mathcal{E}} \left[\phi_{\text{interacts}}\left(\mathbf{e}_M, \mathbf{e}_D\right) \top \phi_{\text{treats}}\left(\mathbf{e}_D, \mathbf{e}_{\text{anxiety}}\right)\right]$$

Greedy Search

- Identify the k most likely values for D
- For each value of D:
  - Identify the k most likely values for M

Complex Query Answering as Search

$$\arg\max_{M,D\in\mathcal{E}} \left[\phi_{\text{interacts}}\left(\mathbf{e}_M, \mathbf{e}_D\right) \top \phi_{\text{treats}}\left(\mathbf{e}_D, \mathbf{e}_{\text{anxiety}}\right)\right]$$

Greedy Search

- Identify the k most likely values for D
- For each value of D:
  - Identify the k most likely values for M
- Compute the query score for all (M, D) combinations

Complex Query Answering as Search

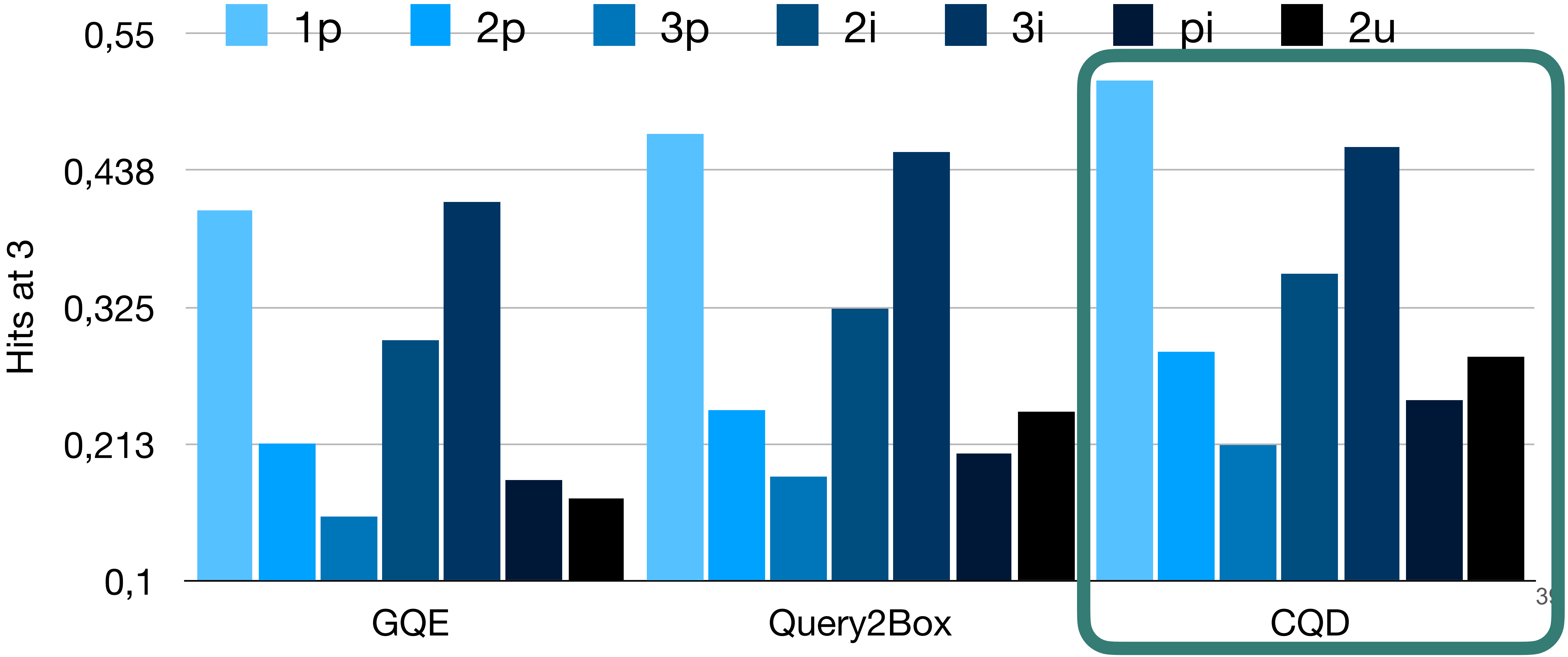
$$\arg\max_{M,D\in\mathcal{E}} \left[\phi_{\text{interacts}}\left(\mathbf{e}_M, \mathbf{e}_D\right) \top \phi_{\text{treats}}\left(\mathbf{e}_D, \mathbf{e}_{\text{anxiety}}\right)\right]$$

Greedy Search

- Identify the k most likely values for D
- For each value of D:
  - Identify the k most likely values for M
- Compute the query score for all (M, D) combinations
- Return the most likely value for (M, D)

Experiments

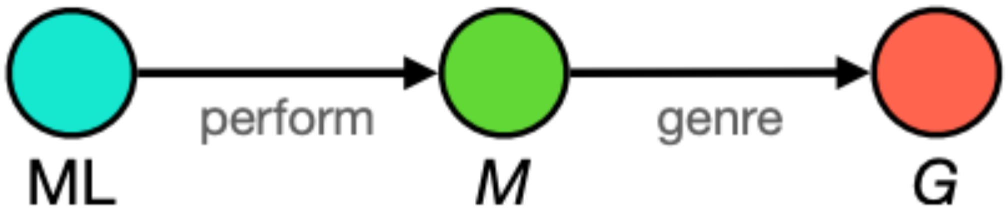
Dataset: FB15k-237



Inspecting Generated Answers

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

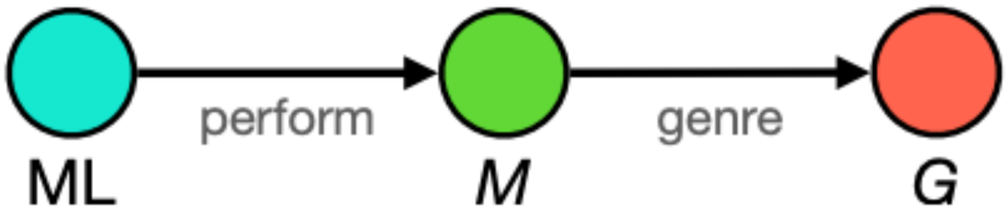
$$?G : \exists M \,.\, \text{perform}(\text{ML}, M) \wedge \text{genre}(M, G)$$



Inspecting Generated Answers

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

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

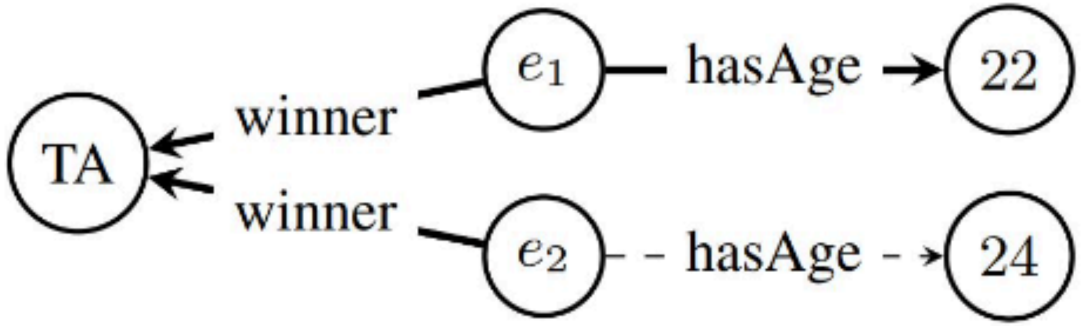


Query: $?G : \exists M.\mathrm{perform}(\mathrm{ML}, M) \wedge \mathrm{genre}(M, G)$

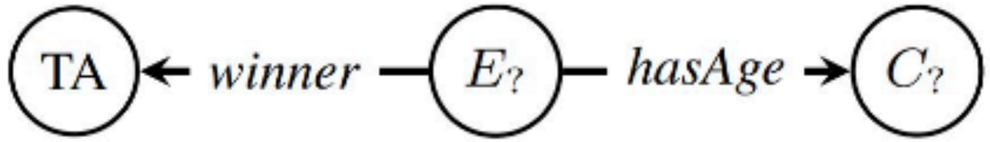
M	G	**Rank**	**Correctness**
Do the Right Thing	Drama	1	✓
	Comedy	4	✓
	Crime Fiction	7	✓
National Security	Action	2	✓
	Thriller	3	✓
	Crime Fiction	5	✓
The Nutty Professor	Comedy	6	✓
	Romantic Com.	8	✗
	Romance Film	9	✗

Complex Queries with Literals

Knowledge graph with literals



Complex query with literals



Demir et al. LitCQD: Multi-Hop Reasoning in Incomplete Knowledge Graphs with Numeric 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

Demir et al. LitCQD: Multi-Hop Reasoning in Incomplete Knowledge Graphs with Numeric Literals”