Complex Query Answering

Pasquale Minervini
p.minervini@ed.ac.uk
Main References


2. Reasoning Beyond Triples: Recent Advances in Knowledge Graph Embeddings (CIKM 2023 Tutorial) — [https://kg-beyond-triple.github.io/](https://kg-beyond-triple.github.io/)


Neural Link Predictors

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

\[
e_{\text{Apixaban}} \in \mathbb{R}^d \quad r_{\text{treats}} \in \mathbb{R}^{d'};
\]

They train embeddings so that links in the graph will have a 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 : interactsWith(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:

\[ D : \text{interactsWith(Apixaban, } D) \land \text{action}(D, \text{vasodilation}) \]

\[ D : \exists D'. \text{interactsWith(Apixaban, } D') \land \text{interactsWith}(D', D) \]
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?

\[ M : \exists D . \text{interacts} (M, D) \land \text{treats} (D, \text{anxiety}) \]
Neural Query Embedding

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

\[ ?M : \exists D . \text{interacts} (M, D) \land \text{treats} (D, \text{anxiety}) \]
Neural Query Embedding

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

\[ \exists D . \text{interacts} (M, D) \land \text{treats} (D, \text{anxiety}) \]
Neural Query Embedding

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

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

\[
\begin{array}{cccccc}
0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccc}
1 & 4 & 3 & 4 & 1 \\
1 & 2 & 4 & 3 & 3 \\
1 & 2 & 3 & 4 & 1 \\
1 & 3 & 3 & 1 & 1 \\
3 & 3 & 1 & 1 & 0 \\
\end{array}
\]

\[
I \ast K
\]
(2D) Convolution on Images

<table>
<thead>
<tr>
<th>Input</th>
<th>Kernel</th>
<th>Bias</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4</td>
<td>0 -1 0</td>
<td>0</td>
<td>11 12</td>
</tr>
<tr>
<td>5 6 7 8</td>
<td>-1 5 -1</td>
<td>5</td>
<td>15 16</td>
</tr>
<tr>
<td>9 10 11 12</td>
<td>0 -1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 14 15 16</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

★ Cross-Correlation
Convolution on Graphs

\[ h_i' = g(h_a, h_b, h_c, \ldots) \]

\[ (a, b, c, \ldots \in N_i) \]
Convolution on Graphs

Inputs $(X, A)$

Latents $(H, A)$

Node classification
$$\vec{Z}_i = f(\vec{h}_i)$$

Graph classification
$$\vec{Z}_G = f(\Sigma_i \vec{h}_i)$$

Link classification
$$\vec{Z}_{ij} = f(\vec{h}_i, \vec{h}_j, \vec{e}_{ij})$$
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

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

- **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)

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

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

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

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

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

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

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

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?

\[ \exists M : \exists D \text{. interacts } (M, D) \land \text{treats } (D, \text{anxiety}) \]
Complex Query Decomposition

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

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

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

$$\arg\max_{M,D \in \mathcal{E}} \exists D. \text{interacts } (M,D) \land \text{treats } (D, \text{anxiety})$$
Complex Query Decomposition

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

$$\arg\max_{M,D \in \mathcal{E}} \phi_{\text{interacts}}(e_M, e_D) \land \phi_{\text{treats}}(e_D, e_{\text{anxiety}})$$

- **Optimisation problem**
- **Likelihood that M interacts with D**
- **Likelihood that D treats anxiety**
Complex Query Decomposition

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

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

Optimisation problem

$\arg\max_{M, D \in \mathcal{E}} [\phi_{\text{interacts}}(e_M, e_D) \land \phi_{\text{treats}}(e_D, e_{\text{anxiety}})]$

Continuous relaxation of the logical AND (t-norm)

- Likelihood that $M$ interacts with $D$
- Likelihood that $D$ treats anxiety

$\mathcal{E}$
T-Norms and T-Conorms

Minimum (or Gödel) t-norm:
\[ A \land B \mapsto \top_{\text{min}} (a, b) = \min(a, b) \]
\[ A \lor B \mapsto \bot_{\text{max}} (a, b) = \max(a, b) \]

Product t-norm:
\[ A \land B \mapsto \top_{\text{product}} (a, b) = ab \]
\[ A \lor B \mapsto \bot_{\text{sum}} (a, b) = a + b - ab \]

Complementary t-conorm:
\[ \bot (a, b) = 1 - \top (1 - a, 1 - b) \]
Complex Query Answering as Search

\[
\arg\max_{M,D\in\mathcal{E}} \left[ \phi_{\text{interacts}} (e_M, e_D) \top \phi_{\text{treats}} (e_D, e_{\text{anxiety}}) \right]
\]

Greedy Search
Complex Query Answering as Search

$$\arg \max_{M,D\in\mathcal{E}} \left[ \phi_{\text{interacts}} \left( e_M, e_D \right) \top \phi_{\text{treats}} \left( e_D, e_{\text{anxiety}} \right) \right]$$

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 $D$: 
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

$$\arg \max_{M,D \in \mathcal{E}} \left[ \phi_{\text{interacts}} (e_M, e_D) \lor \phi_{\text{treats}} (e_D, e_{\text{anxiety}}) \right]$$
Complex Query Answering as Search

\[ \arg\ max_{M,D \in \mathcal{E}} \left[ \phi_{\text{interacts}} \left( e_M, e_D \right) \top \phi_{\text{treats}} \left( e_D, e_{\text{anxiety}} \right) \right] \]

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

\[ \arg \max_{M,D \in \mathcal{E}} \left[ \phi_{\text{interacts}}(e_M, e_D) \top \phi_{\text{treats}}(e_D, e_{\text{anxiety}}) \right] \]

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

Hits at 3

GQE
Query2Box
CQD
Inspecting Generated Answers

“In what genres of movies did Martin Lawrence appear?”

\[ ?G : \exists M . \text{perform}(ML, M) \land \text{genre}(M, G) \]
Inspecting Generated Answers

“In what genres of movies did Martin Lawrence appear?”

\[ ?G : \exists M . \text{perform}(ML, M) \land \text{genre}(M, G) \]

<table>
<thead>
<tr>
<th>Query: ( ?G : \exists M . \text{perform}(ML, M) \land \text{genre}(M, G) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M )</td>
</tr>
<tr>
<td>Do the Right Thing</td>
</tr>
<tr>
<td>Do the Right Thing</td>
</tr>
<tr>
<td>Do the Right Thing</td>
</tr>
<tr>
<td>National Security</td>
</tr>
<tr>
<td>National Security</td>
</tr>
<tr>
<td>National Security</td>
</tr>
<tr>
<td>The Nutty Professor</td>
</tr>
<tr>
<td>The Nutty Professor</td>
</tr>
<tr>
<td>The Nutty Professor</td>
</tr>
</tbody>
</table>
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

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

- 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”