Contextual Graph Attention for Answering Logical Queries over Incomplete Knowledge Graphs

K-CAP 2019, Nov. 2019

Gengchen Mai[1] Krzysztof Janowicz[1] Bo Yan[1]
Rui Zhu[1] Ling Cai[1] Ni Lao[2]

[1] STKO Lab, University of California Santa Barbara
[2]SayMosaic Inc., Palo Alto, CA, USA

Introduction

- **Knowledge Graph** (KG): a **data repository** that describes **entities** and their **relationships** across domains according to some **schema**
- Problem: **Incompleteness**, **Sparsity**, and **Noise**

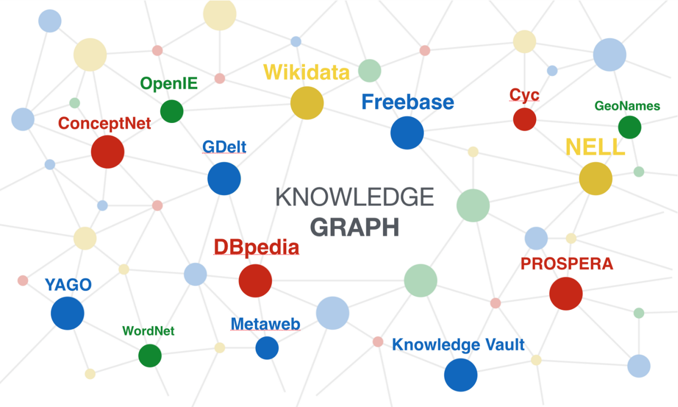
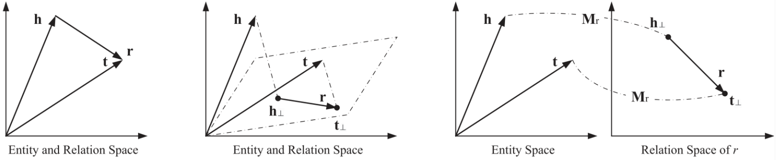


Figure From https://medium.com/@sderymail/challenges-of-knowledge-graph-part-1-d9ffe9e35214

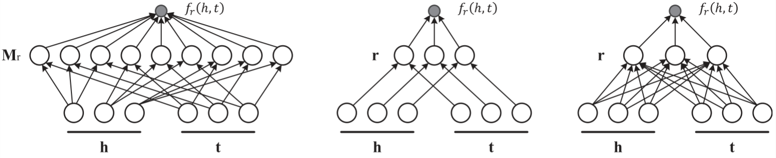
Introduction

- **Knowledge Graph Embedding** for **Knowledge Graph Completion**
- The major **KG Embedding** models can be classified as two categories (Wang et al. 2017):
  - **Translation-based models** (e.g. TransE, TransH, and TransR)



(a) TransE. (b) TransH. (c) TransR.

  - **Semantic matching models** (e.g. RESCAL, DisMult, and HolE)



(a) RESCAL. (b) DistMult. (c) HolE.

Introduction

Training a KG embedding model over a knowledge graph (KG) $\mathcal{G} = (\mathcal{V}, \mathcal{R})$

- **Task**: **link prediction** and **entity classification**
- The **model complexity** is linear with respect to $|\mathcal{V}|$
- Dealing with **more complex tasks**?

Gengchen Mai[1], Krzysztof Janowicz[1], Bo Yan[1], Rui Zhu[1], Ling Cai[1], Ni Lao[2]

Conjunctive Graph Query (CGQ)

Using KG Embeddings for **Conjunctive Graph Query (CGQ)**
A query $q \in Q(\mathcal{G})$ that can be written as follows:

$$q = V_?.\exists V_1, V_2, .., V_m : b_1 \wedge b_2 \wedge ... \wedge b_n$$

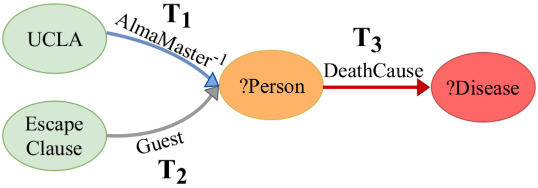
$$\text{where} \quad b_i = r(e_k, V_l), V_l \in \{V_?, V_1, V_2, .., V_m\}, e_k \in \mathcal{V}, r \in \mathcal{R}$$

$$\text{or} \quad b_i = r(V_k, V_l), V_k, V_l \in \{V_?, V_1, V_2, .., V_m\}, k \neq l, r \in \mathcal{R}$$

Requirements:

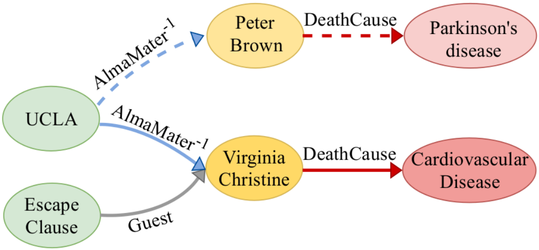
- Require **one variable** as the answer denotation: **Target Node**
- **No variable in the predicate position**
- Only consider the **conjunction** of graph patterns
- The dependence graph of q must be a **directed acyclic graph (DAG)**

Conjunctive Graph Query (CGQ)

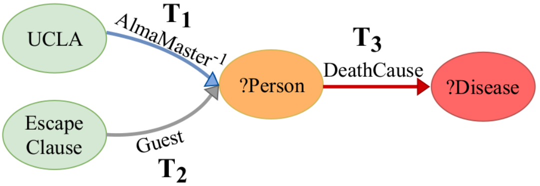


$$
\begin{aligned}
?Disease.\exists ?Person : \ & AlmaMater^{-1}(UCLA, ?Person) \wedge \\
& Guest(EscapeClause, ?Person) \wedge \\
& DeathCause(?Person, ?Disease)
\end{aligned}
$$

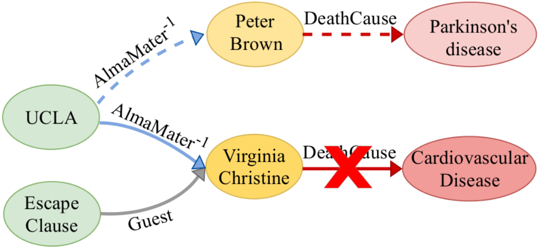
SPARQL?



Conjunctive Graph Query (CGQ)



$$
\begin{aligned}
?Disease.\exists ?Person : \; & AlmaMater^{-1}(UCLA, ?Person) \wedge \\
& Guest(EscapeClause, ?Person) \wedge \\
& DeathCause(?Person, ?Disease)
\end{aligned}
$$



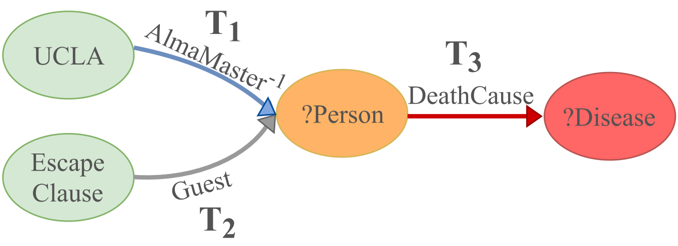
KG Embedding

Gengchen Mai[1], Krzysztof Janowicz[1], Bo Yan[1], Rui Zhu[1], Ling Cai[1], Ni Lao[2]

Conjunctive Graph Query (CGQ)

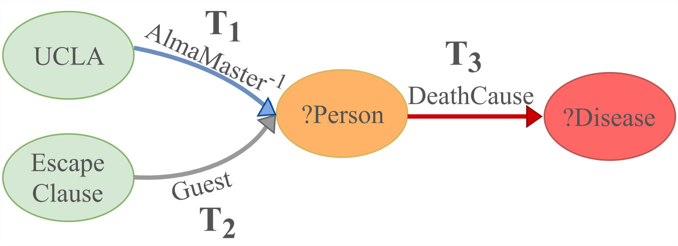
Using **KG Embedding** to predict the answer to a **CGQ**:

- **Projection Operation**: **Translate** from the corresponding entity nodes via different relation embeddings through different paths (triple T_1 and T_2).
- **Intersection Operation**: **Integrate** predicted embeddings for the same variable (**?Person**) from different paths (triple T_1 and T_2).
- Recursively use these two operators until we get the **embedding for the target variable q**.
- **Nearest neighbor search** for answer entities with **q** by **cosine similarity**.



Related Work

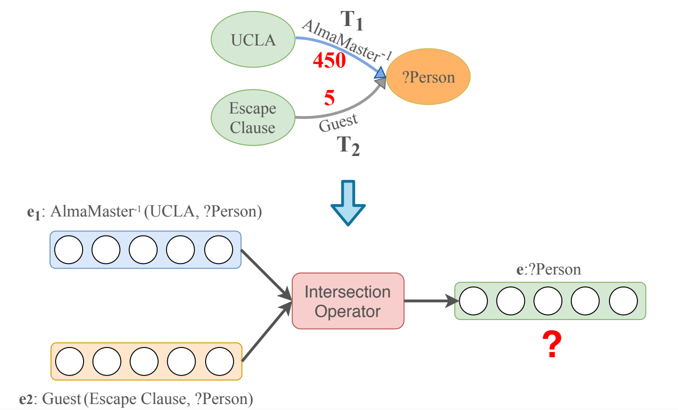
- **Wang et al. (2018)**: Pretrain KG embeddings and use it for CGQ
  - **lacks flexibility**: deterministic weighting approach for path embedding integration
  - **No end-to-end**: does not directly optimized on the QA objective
- **Hamilton et al. (2018)**: An end-to-end model for logic query answering with an **elementwise-mean intersection operator** which treats query path **equally**
  - Fail to consider **unequal contribution from different paths**
  - Do not consider the **original KG structure**
- **Attention**?



Attention Mechanism

Consider **unequal contribution from different triple paths**:

- Problem for **Attention mechanism**: The **center node embedding/query embedding** is a **prerequisite** for attention score computing which is **unknown** in this case



Entity Embedding

Entity embedding lookup:

$$\mathbf{e}_i = \frac{\mathbf{Z}_\gamma \mathbf{x}_i}{\| \mathbf{Z}_\gamma \mathbf{x}_i \|_{L2}} \tag{1}$$

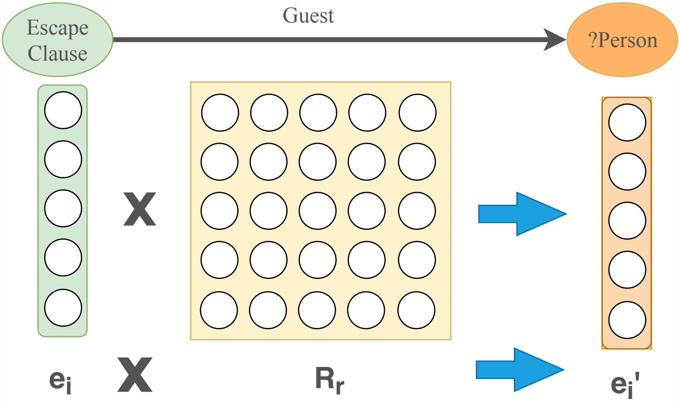
- $\mathbf{Z}_\gamma \in \mathbb{R}^{d \times m_\gamma}$ is the type-specific embedding matrices for all entities with type $\gamma = \Gamma(e_i)$.

Gengchen Mai[1], Krzysztof Janowicz[1], Bo Yan[1], Rui Zhu[1], Ling Cai[1], Ni Lao[2]

Projection Operation

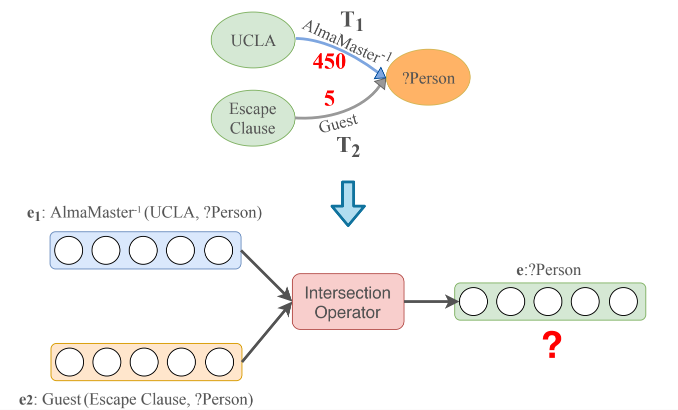
$$\mathbf{e}_i' = \mathcal{P}(e_i, r) = \mathbf{R}_r \mathbf{e}_i \qquad (2)$$

- $\mathbf{R}_r \in \mathbb{R}^{d \times d}$ is a trainable and relation-specific matrix for relation type r.



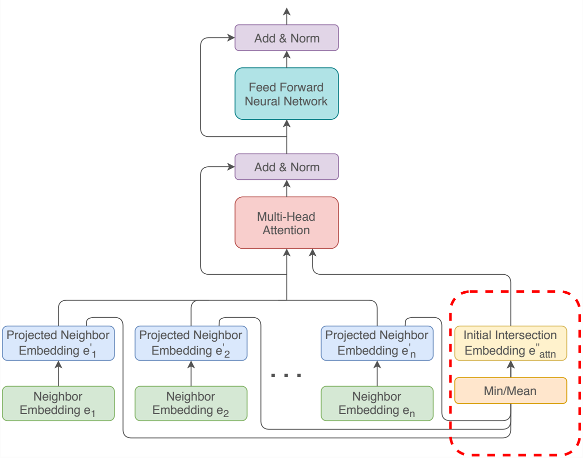
Intersection Operation

$$\mathbf{e}'' = \mathcal{I}_{CGA}(\{\mathbf{e}'_1, \mathbf{e}'_2, ..., \mathbf{e}'_i, ..., \mathbf{e}'_n\}) \quad (3)$$



Intersection Operation

- Multi-head attention inspired by Transformer (Vaswani et al. 2017).
- **An initial intersection embedding layer** (red) is used so that center variable embedding is no longer a prerequisite.



Gengchen Mai[1], Krzysztof Janowicz[1], Bo Yan[1], Rui Zhu[1], Ling Cai[1], Ni Lao[2]

Model Training

- **Original KG Training Phase**:

$$\mathcal{L}_{KG} = \sum_{e_i \in \mathcal{V}} \sum_{e_i^- \in Neg(e_i)} max(0, \Delta - \Phi(\mathbf{H}_{KG}(e_i), \mathbf{e}_i) + \Phi(\mathbf{H}_{KG}(e_i), \mathbf{e}_i^-)) \tag{4}$$

  - Φ: cosine similarity function.
  - $\mathbf{H}_{KG}(e_i)$ indicates a new embedding $\mathbf{e}_i''$ for entity e_i given its 1-degree neighborhood $N(e_i)$.
  - $e_i^- \in Neg(e_i)$ is a negative sample.
- **Logical Query-Answer Pair Training Phase**:

$$\mathcal{L}_{QA} = \sum_{(q_i, a_i) \in S} \sum_{a_i^- \in Neg(q_i, a_i)} max(0, \Delta - \Phi(\mathbf{q}_i, \mathbf{a}_i) + \Phi(\mathbf{q}_i, \mathbf{a}_i^-)) \tag{5}$$

  - q_i: query embedding.
  - $\mathbf{a}_i$, $\mathbf{a}_i^-$: the embedding for the correct answer entity & negative answers.
- **The whole loss function**:

$$\mathcal{L} = \mathcal{L}_{KG} + \mathcal{L}_{QA} \tag{6}$$

Datasets

- The original **Bio** dataset (Hamilton et al. 2018)
- We constructed two datasets from publicly available *DBpedia* and *Wikidata*: **DB18**, **WikiGeo19**
- **Two metrics**: **ROC AUC score** and **average percentile rank** (APR)

Table 1: Statistics for Bio, *DB18* and *WikiGeo19* (Section 4.1). "NUM/QT" indicates the number of QA pairs per query type.

	Bio			DB18			WikiGeo19		
	Training	Validation	Testing	Training	Validation	Testing	Training	Validation	Testing
# of Triples	3,258,473	20,114	181,028	122,243	1,358	12,224	170,409	1,893	17,041
# of Entities	162,622	-	-	21,953	-	-	18,782	-	-
# of Relations	46	-	-	175	-	-	192	-	-
# of Sampled 2-edge QA Pairs	1M	1k/QT	10k/QT	1M	1k/QT	10k/QT	1M	1k/QT	10k/QT
# of Sampled 3-edge QA Pairs	1M	1k/QT	10k/QT	1M	1k/QT	10k/QT	1M	1k/QT	10k/QT

Evaluation Results

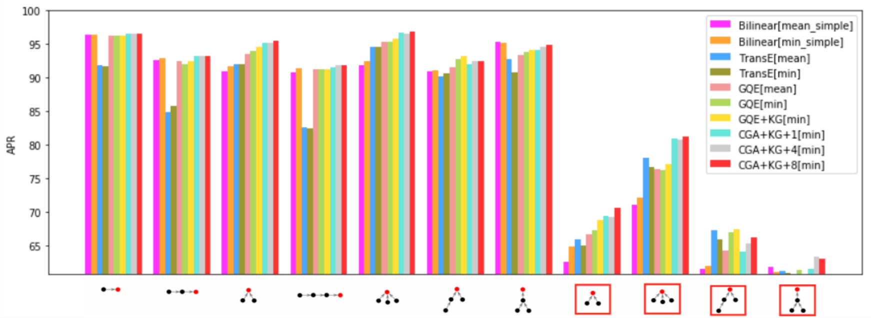
- Adding the **original KG training phase** in the model training process improves the model performance.
- Adding the **attention mechanism** further improves the model performance.
- CGA has **less learnable parameters** with better performance.
- CGA shows strong advantages over baseline models especially on query types with **hard negative sampling**.

Table 2: Macro-average AUC and APR over test queries with different DAG structures are used to evaluate the performance. *All* and *H-Neg*. denote macro-averaged across all query types and query types with hard negative sampling (see Section 3.2.3).

Dataset	Bio				DB18				WikiGeo19			
Metric	AUC		APR		AUC		APR		AUC		APR	
	All	H-Neg	All	H-Neg	All	H-Neg	All	H-Neg	All	H-Neg	All	H-Neg
Billinear[mean_simple]	81.65	67.26	82.39	70.07	82.85	64.44	85.57	71.72	81.82	60.64	82.35	64.22
Billinear[min_simple]	82.52	69.06	83.65	72.7	82.96	64.66	86.22	73.19	82.08	61.25	82.84	64.99
TransE[mean]	80.64	73.75	81.37	76.09	82.76	65.74	85.45	72.11	80.56	65.21	81.98	68.12
TransE[min]	80.26	72.71	80.97	75.03	81.77	63.95	84.42	70.06	80.22	64.57	81.51	67.14
GQE[mean]	83.4	71.76	83.82	73.41	83.38	65.82	85.63	71.77	83.1	63.51	83.81	66.98
GQE[min]	83.12	70.88	83.59	73.38	83.47	66.25	86.09	73.19	83.26	63.8	84.3	67.95
GQE+KG[min]	83.69	72.23	84.07	74.3	84.23	68.06	86.32	73.49	83.66	64.48	84.73	68.51
CGA+KG+1[min]	84.57	74.87	85.18	77.11	84.31	67.72	87.06	74.94	83.91	64.83	85.03	69
CGA+KG+4[min]	**85.13**	**76.12**	85.46	**77.8**	84.46	67.88	87.05	74.66	83.96	64.96	85.36	69.64
CGA+KG+8[min]	85.04	76.05	**85.5**	77.76	**84.67**	**68.56**	**87.29**	**75.23**	**84.15**	**65.23**	**85.69**	**70.28**
Relative Δ over GQE	2.31	**7.29**	2.28	**5.97**	1.44	**3.49**	1.39	**2.79**	1.07	**2.24**	1.65	**3.43**

Evaluation Results

- CGA **outperforms** the baseline models in almost all query types.



APR for *WikiGeo19*

Conclusion

- We propose an **end-to-end attention-based** logical query answering model, **contextual graph attention model (CGA)**.
- The **multi-head attention mechanism** is utilized in the intersection operator to automatically learn **different weights for different query paths**.
- Our models **outperform** the baseline models on three dataset (Bio, *DB18*, and *WikiGeo19*) despite using **less parameters**.

Future Work

- Explore ways to use our model in an **inductive learning setup**
- Consider **disjunction, negation, and filters** in query answering
- Consider **variables in the predicate position**