Reasoning over Knowledge Graphs

CS224W: Machine Learning with Graphs
Jure Leskovec, Hongyu Ren, Stanford University
http://cs224w.stanford.edu
Introduction to Knowledge Graphs

Knowledge Graph completion

Path Queries

Conjunctive Queries

Query2Box: Reasoning with Box Embeddings
Knowledge Graphs

- Knowledge in graph form
  - Capture entities, types, and relationships
- Nodes are **entities**
- Nodes are labeled with their **types**
- Edges between two nodes capture **relationships** between entities
Example: Bibliographic networks

- **Node types**: paper, title, author, conference, year
- **Relation types**: pubWhere, pubYear, hasTitle, hasAuthor, cite
Example: Social networks

- **Node types**: account, song, post, food, channel
- **Relation types**: friend, like, cook, watch, listen
Example: Google Knowledge Graph

[paintedBy] Da Vinci

Mona Lisa

Date of birth: April 15, 1452
Date of death: May 2, 1519
(age 67 years)

Michelangelo

Italy
Knowledge Graphs in Practice

- Google Knowledge Graph
- Amazon Product Graph
- Facebook Graph API
- IBM Watson
- Microsoft Satori
- Project Hanover/Literome
- LinkedIn Knowledge Graph
- Yandex Object Answer
Applications of Knowledge Graphs

- Serving information
Applications of Knowledge Graphs

- Question answering and conversation agents

I want to travel to NY 2 days before Thanksgiving, staying for a week

Okay, booking a flight to JFK from November 20 to November 27. Where will you be flying from?

From San Francisco, and also non-stop in first class

Got it, I've found some flights for you ...

How about leaving in the afternoon

travel  Thanksgiving
NY    week

NY    JFK airport

NY    Nov 22
Thanksgiving    Flight Search
San Francisco    Result Set
non-stop    Filter Set
first class
leaving    SFO airport
afternoon
Outline

1. Introduction to Knowledge Graphs
2. Knowledge Graph completion
3. Path Queries
4. Conjunctive Queries
5. Query2Box: Reasoning with Box Embeddings
Knowledge Graph Datasets

- Publicly available KGs:
  - FreeBase, Wikidata, Dbpedia, YAGO, NELL, etc.

- Common characteristics:
  - Massive: millions of nodes and edges
  - Incomplete: many true edges are missing

Given a massive KG, enumerating all the possible facts is intractable!

Can we predict plausible BUT missing links?
Freebase

- ~50 million entities
- ~38K relation types
- ~3 billion facts/triples
- 93.8% of persons from Freebase have no place of birth and 78.5% have no nationality!

FB15k/FB15k-237

- A complete subset of Freebase, used by researchers to learn KG models

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Entities</th>
<th>Relations</th>
<th>Total Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB15k</td>
<td>14,951</td>
<td>1,345</td>
<td>592,213</td>
</tr>
<tr>
<td>FB15k-237</td>
<td>14,505</td>
<td>237</td>
<td>310,079</td>
</tr>
</tbody>
</table>

Given an enormous KG, can we complete the KG / predict missing relations?

- links + type

missing relation: genre
Edges in KG are represented as **triples** \((h, r, t)\)
- head \((h)\) has relation \((r)\) with tail \((t)\).

**Key Idea:**
- Model entities and relations in the embedding/vector space \(\mathbb{R}^d\).
- Given a true triple \((h, r, t)\), the goal is that the embedding of \((h, r)\) **should be close** to the embedding of \(t\).
  - How to embed \((h, r)\)?
  - How to define closeness?
Relation Patterns

- **Symmetric Relations:**
  \[ r(h, t) \Rightarrow r(t, h) \quad \forall h, t \]
  - **Example:** Family, Roommate

- **Composition Relations:**
  \[ r_1(x, y) \land r_2(y, z) \Rightarrow r_3(x, z) \quad \forall x, y, z \]
  - **Example:** My mother’s husband is my father.

- **1-to-N, N-to-1 relations:**
  \[ r(h, t_1), r(h, t_2), \ldots, r(h, t_n) \] are all True.
  - **Example:** \( r \) is “StudentsOf”
**Translation Intuition:**

For a triple \((h, r, t)\), \(h, r, t \in \mathbb{R}^d\),

\[h + r = t\]

**Score function:** \(f_r(h, t) = ||h + r - t||\)

Translation Intuition: for a triple \((h, r, t)\), \(h + r = t\)

Max margin loss:

\[
\mathcal{L} = \sum_{(h,r,t) \in G, (h,r,t') \notin G} [\gamma + f_r(h, t) - f_r(h, t')]_+
\]

where \(\gamma\) is the margin, i.e., the smallest distance tolerated by the model between a valid triple and a corrupted one.

NOTE: check lecture 7 for a more in-depth discussion of TransE!
- Who has won the Turing award?

- Who is a Canadian citizen?
Composition in TransE

- **Composition Relations:**
  \[ r_1(x, y) \land r_2(y, z) \Rightarrow r_3(x, z) \quad \forall x, y, z \]

- **Example:** My mother’s husband is my father.

- In TransE:
  \[ r_3 = r_1 + r_2 \]
Limitation: Symmetric Relations

- **Symmetric Relations:**
  \[ r(h, t) \Rightarrow r(t, h) \quad \forall h, t \]

- **Example:** Family, Roommate

- **In TransE:**
  \[ r = 0, \quad h = t \times \]

If we want TransE to handle symmetric relations \( r \), for all \( h, t \) that satisfy \( r(h, t) \), \( r(t, h) \) is also True, which means \( \|h + r - t\| = 0 \) and \( \|t + r - h\| = 0 \). Then \( r = 0 \) and \( h = t \), however \( h \) and \( t \) are two different entities and should be mapped to different locations.
Limitation: N-ary Relations

- 1-to-N, N-to-1, N-to-N relations.
- Example: \((h, r, t_1)\) and \((h, r, t_2)\) both exist in the knowledge graph, e.g., \(r\) is “StudentsOf”

With TransE, \(t_1\) and \(t_2\) will map to the same vector, although they are different entities.

- \(t_1 = h + r = t_2\)
- \(t_1 \neq t_2\)  contradictory!
TransR

- TransR: model entities as vectors in the entity space $\mathbb{R}^d$ and **model each relation as vector** $r$ **in relation space** $\mathbb{R}^k$ with $M_r \in \mathbb{R}^{k \times d}$ as the projection matrix.

- $h_\perp = M_r h$, $t_\perp = M_r t$
- $f_r(h, t) = ||h_\perp + r - t_\perp||$

Symmetric Relations in TransR

- **Symmetric Relations:**
  \[ r(h, t) \Rightarrow r(t, h) \ \forall h, t \]

- **Example:** Family, Roommate

\[ r = 0, \ h_\perp = M_r h = M_r t = t_\perp \checkmark \]

For TransR, we can map \( h \) and \( t \) to the same location on the space of relation \( r \).
N-ary Relations in TransR

- **1-to-N, N-to-1, N-to-N relations**
- **Example**: If \((h, r, t_1)\) and \((h, r, t_2)\) exist in the knowledge graph.

We can learn \(M_r\) so that \(t_\perp = M_r t_1 = M_r t_2\), note that \(t_1\) does not need to be equal to \(t_2\)!
Limitation: Composition in TransR

- Composition Relations:
  \[ r_1(x, y) \land r_2(y, z) \Rightarrow r_3(x, z) \quad \forall x, y, z \]
- Example: My mother’s husband is my father.

Each relation has different space.
It is not naturally compositional for multiple relations! ✗
## Translation-Based Embedding

### Embedding

<table>
<thead>
<tr>
<th>Embedding</th>
<th>Entity</th>
<th>Relation</th>
<th>$f_r(h, t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE</td>
<td>$h, t \in \mathbb{R}^d$</td>
<td>$r \in \mathbb{R}^d$</td>
<td>$</td>
</tr>
<tr>
<td>TransR</td>
<td>$h, t \in \mathbb{R}^d$</td>
<td>$r \in \mathbb{R}^k, M_r \in \mathbb{R}^{k \times d}$</td>
<td>$</td>
</tr>
</tbody>
</table>

### Symmetry, Composition, One-to-many

<table>
<thead>
<tr>
<th>Embedding</th>
<th>Symmetry</th>
<th>Composition</th>
<th>One-to-many</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>TransR</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
1. Introduction to Knowledge Graphs
2. Knowledge Graph completion
3. Path Queries
4. Conjunctive Queries
5. Query2Box: Reasoning with Box Embeddings
Can we do multi-hop reasoning, i.e., **answer complex queries efficiently** on an **incomplete, massive KG**?

<table>
<thead>
<tr>
<th>Query Types</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-hop Queries</td>
<td>Where did Hinton graduate?</td>
</tr>
<tr>
<td>Path Queries</td>
<td>Where did Turing Award winners graduate?</td>
</tr>
<tr>
<td>Conjunctive Queries</td>
<td>Where did Canadians with Turing Award graduate?</td>
</tr>
<tr>
<td>EPFO Queries</td>
<td>Where did Canadians with Turing Award or Nobel graduate?</td>
</tr>
</tbody>
</table>
We can formulate link prediction problems as answering one-hop queries.

- **Link prediction**: Is link \((h, r, t)\) True?
- **One-hop query**: Is \(t\) an answer to query \((h, r)\)?
Path Queries

- Generalize one-hop queries to path queries by adding more relations on the path.
- Path queries can be represented by $q = (v_a, r_1, \ldots, r_n)$
  - $v_a$ is a constant node, answers are denoted by $[q]$.

Computation graph of $q$:

Computation graph of path queries is a chain.
“Where did Turing Award winners graduate?”

- $\nu_a$ is “Turing Award”.
- $(r_1, r_2)$ is (“win”, “graduate”).

Given a KG, **how to answer the query?**
Answer path queries by traversing the KG.

“Where did Turing Award winners graduate?”

The anchor node is Turing Award.
Answer path queries by traversing the KG. 
“Where did Turing Award winners graduate?”

Start from the anchor node “Turing Award” and traverse the KG by the relation “Win”, we reach entities {“Pearl”, “Hinton”, “Bengio”}. 
Answer path queries by traversing the KG. “Where did Turing Award winners graduate?”

Start from nodes {“Pearl”, “Hinton”, “Bengio”} and traverse the KG by the relation “Graduate”, we reach entities {“NYU”, “Edinburgh”, “Cambridge”, “McGill”}. These are the answers to the query!
Answer path queries by traversing the KG. “Where did Turing Award winners graduate?”

What if KG is incomplete?
Can we first do link prediction and then traverse the completed (probabilistic) KG?

\textbf{No!} The completed KG is a \textbf{dense graph}!

Time complexity of traversing a dense KG with $|V|$ entities to answer $(v_a, r_1, \ldots, r_n)$ of length $n$ is $\mathcal{O}(|V|^n)$. 
Traversing KG in Vector Space

- **Key idea: embed queries!**
  - Generalize TransE to multi-hop reasoning.

Given a path query \( q = (v_a, r_1, \ldots, r_n) \),

\[
q = v_a + r_1 + \cdots + r_n
\]

- **Is \( v \) an answer to \( q \)?**
  - Do a nearest neighbor search for all \( v \) based on \( f_q(v) = ||q - v|| \), time complexity is \( \mathcal{O}(V) \).

embed path queries in vector space.

“Where did Turing Award winners graduate?”

Follow the computation graph:

Computation Graph

Embedding Space

Turing Award
Embed path queries in vector space. “Where did Turing Award winners graduate?”
Follow the computation graph:

Computation Graph

Embedding Space

Projection
Embed path queries in vector space.

“Where did Turing Award winners graduate?”

Follow the computation graph:
Outline of Today’s Lecture

1. Introduction to Knowledge Graphs
2. Link Prediction
3. Path Queries
4. Conjunctive Queries
5. Query2Box: Reasoning with Box Embeddings
Can we answer more complex queries?
What if we start from multiple anchor nodes?

“Where did Canadian citizens with Turing Award graduate?”

Computation graph of $q$:
Can we answer even more complex queries?
“Where did Canadian citizens with Turing Award graduate?”

Two anchor nodes: Canada and Turing Award.

Start from the first anchor node “Turing Award”, and traverse by relation “Win”, we reach {“Pearl”, “Hinton”, “Bengio”}.
Can we answer even more complex queries?

“Where did Canadian citizens with Turing Award graduate?”

Two anchor nodes: Canada and Turing Award.

Can we answer even more complex queries?

“Where did Canadian citizens with Turing Award graduate?”

Two anchor nodes: Canada and Turing Award.

Then, we take intersection of the two sets and achieve {'Hinton', 'Bengio'}
Can we answer even more complex queries? “Where did Canadian citizens with Turing Award graduate?”

Two anchor nodes: Canada and Turing Award.

We do another traverse and arrive at the answers!
Key Idea: embed queries in vector space

“Where did Canadian citizens with Turing Award graduate?”

Follow the computation graph:
Key Idea: embed queries in vector space
“Where did Canadian citizens with Turing Award graduate?”

Follow the computation graph:
Neural Intersection Operator

- How do we take intersection of several vectors in the embedding space?

- Design a neural intersection operator $I$
  - Input: current query embeddings $q_1, \ldots, q_m$
  - Output: intersection query embedding $q$
  - $I$ should be permutation invariant:

$$ I(q_1, \ldots, q_m) = I(q_{p(1)}, \ldots, q_{p(m)}) $$

$[p(1), \ldots, p(m)]$ is any permutation of $[1, \ldots, m]$
DeepSets architecture

Vector embeddings of the input queries

Features of the input queries

Permutation Invariant

Vector embedding of the intersection query
Traversing KG in Vector Space

- Key Idea: embed queries in vector space
  “Where did Canadian citizens with Turing Award graduate?”

Follow the computation graph:

Computation Graph

Embedding Space

\[ q = j(q_1, q_2) \]
Given an entity embedding $v$ and a query embedding $q$, the distance is $f_q(v) = ||q - v||$.

**Trainable parameters:**
- entity embeddings: $d|V|$
- relation embeddings: $d|R|$
- intersection operator $\phi, \beta$: number of parameters does not depend on graph size

**Same training strategy as TransE**
Whole Process

- **Training:**
  1. Sample a query $q$, answer $v$, negative sample $v'$.
  2. Embed the query $q$.
  3. Calculate the distance $f_q(v)$ and $f_q(v')$.
  4. Optimize the loss $\mathcal{L}$.

- **Query evaluation:**
  1. Given a test query $q$, embed the query $q$.
  2. For all $v$ in KG, calculate $f_q(v)$.
  3. Sort the distance and rank all $v$. 
Limitations

- Taking the intersection between two vectors is an operation that does not follow intuition.

- When we traverse the KG to achieve the answers, each step produces a set of reachable entities. How can we better model these sets?

- Can we define a more expressive geometry to embed the queries?
Outline

1. Introduction to Knowledge Graphs
2. Knowledge Graph completion
3. Path Queries
4. Conjunctive Queries
5. Query2Box: Reasoning with Box Embeddings
Box Embeddings

- Embed queries with hyper-rectangles (boxes)
  \[ q = (\text{Center}(q), \text{Offset}(q)) \]
Addressing Limitations

- Taking intersection between two vectors is an operation that does **not follow intuition**.
  - Intersection of boxes is well-defined!
- When we traverse the KG to achieve the answers, each step produces a set of reachable entities. How can we better model these sets?
  - Boxes are a **powerful abstraction**, as we can project the center and control the offset to model the set of entities enclosed in the box.
Parameters:

- **entity embeddings**: $d|V|$
  - entities are seen as zero-volume boxes
- **relation embeddings**: $2d|R|$
  - augment each relation with an offset
- intersection operator $\phi, \beta$: number of parameters does not depend on graph size
  - New operator, inputs are boxes and output is a box
Embed queries in vector space

“Where did Canadian citizens with Turing Award graduate?”

Note that computation graph stays the same!

Follow the computation graph:

**Computation Graph**

- Turing Award
- Canada

**Embedding Space**

- Turing Award
- Canada
Embed queries in vector space

“Where did Canadian citizens with Turing Award graduate?”

Note that computation graph stays the same!

Follow the computation graph:

**Computation Graph**

**Embedding Process**
Geometric Projection Operator $\mathcal{P}$

$\mathcal{P} : \text{Box} \times \text{Relation} \rightarrow \text{Box}$

\[
\begin{align*}
\text{Cen}(q') &= \text{Cen}(q) + \text{Cen}(r) \\
\text{Off}(q') &= \text{Off}(q) + \text{Off}(r)
\end{align*}
\]
Embed queries in vector space

“Where did Canadian citizens with Turing Award graduate?”

Note that computation graph stays the same!

Follow the computation graph:
Embed queries in vector space

“Where did Canadian citizens with Turing Award graduate?”

Note that computation graph stays the same!

Follow the computation graph:
Geometric Intersection Operator $\mathcal{I}$

$\mathcal{I} : \text{Box} \times \cdots \times \text{Box} \rightarrow \text{Box}$

- The new center is a weighted average.
- The new offset shrinks.
Geometric Intersection Operator $I$

$I : \text{Box} \times \cdots \times \text{Box} \rightarrow \text{Box}$

$\text{Cen}(q_{\text{inter}}) = \sum_i w_i \odot \text{Cen}(q_i)$

dimension-wise product

$\text{Off}(q_{\text{inter}}) = \min(\text{Off}(q_1), \ldots, \text{Off}(q_n))$

$\odot \sigma(\text{Deepsets}(q_1, \ldots, q_n))$

Sigmoid function: squashes output in (0,1)

weight

guarantees shrinking
Embed queries in vector space

“Where did Canadian citizens with Turing Award graduate?”

Note that computation graph stays the same!

Follow the computation graph:
Embed queries in vector space

“Where did Canadian citizens with Turing Award graduate?”

Note that computation graph stays the same!

Follow the computation graph:
Given a query box \( q \) and entity vector \( v \),

\[
d_{\text{box}}(q, v) = d_{\text{out}}(q, v) + \alpha \cdot d_{\text{in}}(q, v)
\]

where \( 0 < \alpha < 1 \).
Given a set of queries and answers,

\[ \mathcal{L} = -\log \sigma(\gamma - d_{box}(q, v)) - \log \sigma(d_{box}(q, v_i') - \gamma) \]

\(-\log \sigma(\gamma - d_{box}(q, v))\) minimize loss \(\rightarrow\) minimize \(d_{box}(q, v)\)

\(-\log \sigma(d_{box}(q, v') - \gamma)\) minimize loss \(\rightarrow\) maximize \(d_{box}(q, v')\)
Relation Patterns

- Can query2box handle different relation patterns?

<table>
<thead>
<tr>
<th>Embedding</th>
<th>Symmetry</th>
<th>Composition</th>
<th>One-to-many</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>TransH</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Query2Box</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

For details please check the paper [https://openreview.net/forum?id=BJgr4kSFDS](https://openreview.net/forum?id=BJgr4kSFDS)
N-ary Relations in query2box

- **1-to-N, N-to-1, N-to-N relations.**
- **Example:** Both \((h, r, t_1)\) and \((h, r, t_2)\) exist.

- Box Embedding can handle since \(t_1\) and \(t_2\) will be mapped to different locations in the box of \((h, r)\). ✓
Symmetric Relations in query2box

- Symmetric Relations:
  \[ r(h, t) \Rightarrow r(t, h) \quad \forall h, t \]
- **Example**: Family, Roommate
- Box Embedding
  \[ Cen(r) = 0 \checkmark \]

For symmetric relations \( r \), we could assign \( Cen(r) = 0 \). In this case, as long as \( t \) is in the box of \( (h, r) \), it is guaranteed that \( h \) is in the box of \( (t, r) \). So we have \( r(h, t) \Rightarrow r(t, h) \).
Composition Relations:

\[ r_1(x, y) \land r_2(y, z) \Rightarrow r_3(x, z) \quad \forall x, y, z \]

**Example:** My mother’s husband is my father.

**Box Embedding**

\[ r_3 = r_1 + r_2 \]

For composition relations, if \( y \) is in the box of \((x, r_1)\) and \( z \) is in the box of \((y, r_2)\), it is guaranteed that \( z \) is in the box of \((x, r_1 + r_2)\).
Can we embed even more complex queries? “Where did Canadians with Turing Award or Nobel graduate?”

Conjunctive queries + disjunction is called Existential Positive First-order (EPFO) queries.

Can we also design a disjunction operator and embed EPFO queries in low-dimensional vector space? YES!

For details please check the paper https://openreview.net/forum?id=BJgr4kSFDS
Datasets: FB15K, FB15K-237

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Entities</th>
<th>Relations</th>
<th>Training Edges</th>
<th>Validation Edges</th>
<th>Test Edges</th>
<th>Total Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB15k</td>
<td>14,951</td>
<td>1,345</td>
<td>483,142</td>
<td>50,000</td>
<td>59,071</td>
<td>592,213</td>
</tr>
<tr>
<td>FB15k-237</td>
<td>14,505</td>
<td>237</td>
<td>272,115</td>
<td>17,526</td>
<td>20,438</td>
<td>310,079</td>
</tr>
</tbody>
</table>

Goal: can the model discover true answers that cannot be achieved by traversing the KG?
- **Training KG**: Training Edges
- **Validation KG**: Training Edges + Validation Edges
- **Test KG**: Training Edges + Validation Edges + Test Edges

Queries:

- Training Conjunctive Queries
- Unseen Conjunctive Queries
- Union Queries
Query Generation

- Given a query structure, use pre-order traversal (traverse from root to leaves) to assign an entity/relation for every node/edge.

- We explicitly rule out degenerated queries.
After instantiation, run post-order traversal (traverse from leaves $v_1, v_2$ to root) to achieve all answers.

For test queries, we guarantee that they cannot be fully answered on training/validation KG.
Query Statistics

Training Conjunctive Queries

Unseen Conjunctive Queries

Union Queries

<table>
<thead>
<tr>
<th>Queries</th>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td>1p</td>
<td>others</td>
<td>1p</td>
</tr>
<tr>
<td></td>
<td>273,710</td>
<td>273,710</td>
<td>59,097</td>
</tr>
<tr>
<td>FB15k</td>
<td>273,710</td>
<td>59,097</td>
<td>67,016</td>
</tr>
<tr>
<td>FB15k-237</td>
<td>149,689</td>
<td>20,101</td>
<td>22,812</td>
</tr>
</tbody>
</table>

11/21/19
What does query2box actually learn?

Example: “List male instrumentalists who play string instruments”

- We use T-SNE to reduce the embedding space to a 2-dimensional space, in order to visualize the query results.
“List male instrumentalists who play string instruments”
“List male instrumentalists who play string instruments”
“List male instrumentalists who play string instruments”

TPR: 100%
FPR: 0%

# of string instruments: 10
Embedding Space

“List male instrumentalists who play string instruments”

# of instrumentalists: 472

TPR: 98.4%
FPR: 0.01%

String Instrument

Projection

Projection
“List male instrumentalists who play string instruments”
“List male instrumentalists who play string instruments”
“List male instrumentalists who play string instruments”

- TP: True Positive
- FN: False Negative
- FP: False Positive
- TN: True Negative

# of answers: 396

TPR: 99.4%
FPR: 0.01%