Random Walk Inference Over Knowledge Bases and Text

11-805 class presentation
Matt Gardner
work done by: Ni Lao, Matt Gardner, and collaborators
What is a knowledge base?
Why knowledge bases?
Why knowledge bases?

“Flights overhead now”

Here is what I found:

<table>
<thead>
<tr>
<th>Input interpretation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>flights seen from current location</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>altitude</th>
<th>angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Southwest Airlines</td>
<td>35000</td>
<td>45°  up</td>
</tr>
<tr>
<td>flight 4504</td>
<td>feet</td>
<td></td>
</tr>
<tr>
<td>Southwest Airlines</td>
<td>34700</td>
<td>43°  up</td>
</tr>
<tr>
<td>flight 987</td>
<td>feet</td>
<td></td>
</tr>
<tr>
<td>Delta Air</td>
<td>33000</td>
<td>42°  up</td>
</tr>
<tr>
<td></td>
<td>feet</td>
<td></td>
</tr>
</tbody>
</table>
Why knowledge bases?

"Flights overhead now"

Here is what I found:

<table>
<thead>
<tr>
<th>Input interpretation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flights seen from current location</td>
<td>altitude</td>
</tr>
<tr>
<td>Southwest Airlines flight 4504</td>
<td>35,000 feet</td>
</tr>
<tr>
<td>Southwest Airlines flight 987</td>
<td>34,700 feet</td>
</tr>
<tr>
<td>Delta Air</td>
<td>33,000 feet</td>
</tr>
</tbody>
</table>
But ...
But ...
But ...

Google search:
- What does Matt Gardner work on?

Matt Gardner profiles | LinkedIn
- www.linkedin.com/pub/dir/?first=matt&last=GARDNER
- View the profiles of professionals named Matt Gardner on LinkedIn. ... how we will be able to add value to your search and work in partnership with you from ...

Matt Gardner - State Farm Agent Jobs | Indeed.com
- www.indeed.com/forums/company/mattgardner
- Do you work at Matt Gardner - State Farm Agent? How did you find the job? How did you get that first interview? Any advice for someone trying to... Host ...

Matthew Gardner - the Gardner Law Firm, PC
- www.gardnerlawpc.com/attorney-profile
- While my profession is an attorney, my most important job is that of a father to my young daughters. With those values that I place on my own family, the work I do ...
So we do inference
So we do inference

- Predict missing facts given what we know
So we do inference

- Predict missing facts given what we know

- Lots of ways to do this, today we’ll talk about random walk inference, or the Path Ranking Algorithm (PRA)
So we do inference

- Predict missing facts given what we know

- Lots of ways to do this, today we’ll talk about random walk inference, or the Path Ranking Algorithm (PRA)

- [Lao, Mitchell, Cohen, EMNLP 2011]
Inference

If: $x_1$ competes with $(x_1, x_2)$

Then: economic sector $(x_1, x_3)$
CityLocatedInCountry - Selecting path features

[Path type] [Count]

[Lao et al, EMNLP 2011]
CityLocatedInCountry - Selecting path features

Path type                       Count
CityInState, StateInCountry     1

[EMNLP 2011]
CityLocatedInCountry - Selecting path features

<table>
<thead>
<tr>
<th>Path type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>CityInState, StateInCountry</td>
<td>1</td>
</tr>
<tr>
<td>CityInState, CityInState(^{-1}), CityLocatedInCountry</td>
<td>1</td>
</tr>
</tbody>
</table>

[Laos et al, EMNLP 2011]
CityLocatedInCountry - Selecting path features

Path type                     Count
CityInState, StateInCountry  1
CityInState, CityInState⁻¹, CityLocatedInCountry  2

[Note: Diagram and table data]
CityLocatedInCountry - Selecting path features

Path type                                                                                                         Count
CityInState, StateInCountry                                                                                         1
CityInState, CityInState\(^{-1}\), CityLocatedInCountry                                                              2
AtLocation\(^{-1}\), AtLocation, CityLocatedInCountry                                                                1

[19]

[4, EMNLP 2011]
CityLocatedInCountry - Selecting path features

Path type                                                                                                 Count
CityInState, StateInCountry                                                                             2
CityInState, CityInState$^{-1}$, CityLocatedInCountry                                                   24
AtLocation$^{-1}$, AtLocation, CityLocatedInCountry                                                     10
CityLiesOnRiver, CityLiesOnRiver$^{-1}$, CityLocatedInCountry                                           1

[Laor et al, EMNLP 2011]
CityLocatedInCountry - Selecting path features

Path type                                                                                      Count
CityInState, StateInCountry                                                              3,892
CityInState, CityInState⁻¹, CityLocatedInCountry                                        234
AtLocation⁻¹, AtLocation, CityLocatedInCountry                                          1,543
CityLiesOnRiver, CityLiesOnRiver⁻¹, CityLocatedInCountry                                  123
…                                                                                           …
CityLocatedInCountry - Selecting path features

Select the most frequent path types, and keep them as features in the model
CityLocatedInCountry - Computing feature values for (Pittsburgh, USA)

Feature Value

Pittsburgh

Feature = Typed Path
CityInState, CityInState⁻¹, CityLocatedInCountry

Feature Value

[Lao et al, EMNLP 2011]
CityLocatedInCountry - Computing feature values for (Pittsburgh, USA)

Feature = Typed Path
CityInState, CityInState$^{-1}$, CityLocatedInCountry

Feature Value

[CityInState, CityInState$^{-1}$, CityLocatedInCountry]
CityLocatedInCountry - Computing feature values for (Pittsburgh, USA)

Feature = Typed Path
CityInState, CityInState⁻¹, CityLocatedInCountry

Feature Value

[Laò et al, EMNLP 2011]
CityLocatedInCountry - Computing feature values for (Pittsburgh, USA)

Feature = Typed Path
CityInState, CityInState⁻¹, CityLocatedInCountry

Feature Value
CityLocatedInCountry - Computing feature values for (Pittsburgh, USA)

[26]
CityLocatedInCountry - Computing feature values for (Pittsburgh, USA)

Feature = Typed Path
CityInState, CityInState$^{-1}$, CityLocatedInCountry

Feature Value
1.0

Pr(U.S. | Pittsburgh, TypedPath)

[Lao et al, EMNLP 2011]
CityLocatedInCountry - Computing feature values for (Pittsburgh, USA)

Feature = Typed Path
CityInState, CityInState$^{-1}$, CityLocatedInCountry
AtLocation$^{-1}$, AtLocation, CityLocatedInCountry

Feature Value
1.0

[La et al, EMNLP 2011]
CityLocatedInCountry - Computing feature values for (Pittsburgh, USA)

Feature = Typed Path
CityInState, CityInState^{-1}, CityLocatedInCountry
AtLocation^{-1}, AtLocation, CityLocatedInCountry

Feature Value
1.0

[Lao et al, EMNLP 2011]
CityLocatedInCountry - Computing feature values for (Pittsburgh, USA)

Feature = Typed Path
CityInState, CityInState⁻¹, CityLocatedInCountry
AtLocation⁻¹, AtLocation, CityLocatedInCountry

Feature Value
1.0

[Laurent et al, EMNLP 2011]
CityLocatedInCountry - Computing feature values for (Pittsburgh, USA)

**Feature = Typed Path**

CityInState, CityInState\(^{-1}\), CityLocatedInCountry

AtLocation\(^{-1}\), AtLocation, CityLocatedInCountry

**Feature Value**

1.0

0.6

[Lao et al, EMNLP 2011]
This is a row in a feature matrix! Use standard logistic regression
# PRA Feature Matrix

CityLocatedInCountry

<table>
<thead>
<tr>
<th>CityLocatedInCountry</th>
<th>AtLocation₁</th>
<th>CityLocatedInCountry</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Pittsburgh, USA)</td>
<td>1.0</td>
<td>(Tokyo, USA)</td>
</tr>
<tr>
<td>(Pittsburgh, Japan)</td>
<td>0.0</td>
<td>(Tokyo, Japan)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
PRA Feature Matrix

CityLocatedInCountry

(Pittsburgh, USA) 1.0 0.6 0.0 0.4 ...
(Pittsburgh, Japan) 0.0 0.2 0.2 0.1 ...
...
(Tokyo, USA) ...
(Tokyo, Japan) ...
...

Large data sets?
PRA Feature Matrix

CityLocatedInCountry

<table>
<thead>
<tr>
<th></th>
<th>Pittsburgh, USA</th>
<th>Pittsburgh, Japan</th>
<th>Tokyo, USA</th>
<th>Tokyo, Japan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pittsburgh, USA</td>
<td>1.0 0.6 0.0 0.4</td>
<td>0.0 0.2 0.2 0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pittsburgh, Japan</td>
<td>0.0 0.2 0.2 0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tokyo, USA</td>
<td>0.0 0.2 0.2 0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tokyo, Japan</td>
<td>0.0 0.2 0.2 0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Large data sets?

\[ O(n^2) \]
### PRA Feature Matrix

<table>
<thead>
<tr>
<th>CityLocatedInCountry</th>
<th>1.0</th>
<th>0.6</th>
<th>0.0</th>
<th>0.4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Pittsburgh, USA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Pittsburgh, Japan)</td>
<td>0.0</td>
<td>0.2</td>
<td>0.2</td>
<td>0.1</td>
<td>...</td>
</tr>
<tr>
<td>(Tokyo, USA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Tokyo, Japan)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ O(n^2) \]

\[ O(r^l) \]

Large data sets?
PRA Feature Matrix

\[ O\left(\frac{e}{n}\right)^l \]

CityLocatedInCountry

\begin{array}{cccccc}
(Pittsburgh, USA) & 1.0 & 0.6 & 0.0 & 0.4 & \ldots \\
(Pittsburgh, Japan) & 0.0 & 0.2 & 0.2 & 0.1 & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
(Tokyo, USA) & \ldots & \ldots & \ldots & \ldots & \ldots \\
(Tokyo, Japan) & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\end{array}

Large data sets?

\[ O\left(n^2\right) \]

\[ O(r^l) \]
Implementation
Implementation

- Small graph: in memory
Implementation

- Small graph: in memory
- Larger graph: GraphChi
Implementation

- Small graph: in memory
- Larger graph: GraphChi
  - Vertex-centric computation. Go through the graph sequentially, processing each walk at each vertex and sending it to the next stop.
Implementation

- Small graph: in memory
- Larger graph: GraphChi
  - Vertex-centric computation. Go through the graph sequentially, processing each walk at each vertex and sending it to the next stop.
- On a cluster: have a graph server
So...
So...

- KB inference is all well and good, but…
So...

- KB inference is all well and good, but...

- What about text?
Inference over KB plus text
Inference over KB plus text

- Augment NELL or Freebase with information automatically extracted from text
Inference over KB plus text

- Augment NELL or Freebase with information automatically extracted from text
- From the combined graph, predict new NELL (or Freebase) relations
Inference over KB plus text

- Augment NELL or Freebase with information automatically extracted from text
- From the combined graph, predict new NELL (or Freebase) relations
- Basically “aggregate” relation extraction
Simple relation embeddings

\[
\begin{align*}
KBB & \rightarrow (e, e) \\
\text{PCA} & \rightarrow R
\end{align*}
\]
“Steel City overlooks the Allegheny”
“Pittsburgh lies on the Mon”
“Pittsburgh sits on the Monongahela”
“Steel City overlooks the Allegheny”
“Pittsburgh lies on the Mon”
“Pittsburgh sits on the Monongahela”
“Steel City overlooks the Allegheny”
“Pittsburgh lies on the Mon”
“Pittsburgh sits on the Monongahela”
“Steel City overlooks the Allegheny”
“Pittsburgh lies on the Mon”
“Pittsburgh sits on the Monongahela”
“Steel City overlooks the Allegheny”
“Pittsburgh lies on the Mon”
“Pittsburgh sits on the Monongahela”
Relation: cityLiesOnRiver
Path: canReferTo⁻¹, “lies on”, canReferTo
Relation: cityLiesOnRiver
Path: canReferTo⁻¹, “lies on”, canReferTo

“Steel City” canReferTo “overlooks” “the Allegheny” canReferTo Allegheny River
canReferTo “the Mon” canReferTo Monongahela River
canReferTo “the Monongahela”
canReferTo “sits on” “Pittsburgh”

“lies on” “the Mon”
canReferTo “lies on” “Pittsburgh”

“Pittsburgh”

KB + Text
Relation: cityLiesOnRiver

Path: canReferTo⁻¹, “lies on”, canReferTo

“Steel City” \(\xrightarrow{\text{canReferTo}}\) “overlooks” \(\xrightarrow{\text{canReferTo}}\) “the Allegheny”

Pittsburgh \(\xrightarrow{\text{canReferTo}}\) “lies on” \(\xrightarrow{\text{canReferTo}}\) “the Mon”

“Pittsburgh” \(\xrightarrow{\text{canReferTo}}\) “sits on” \(\xrightarrow{\text{canReferTo}}\) “the Monongahela”

Allegheny River

Monongahela River
Relation: cityLiesOnRiver

Path: canReferTo^{-1}, “lies on”, canReferTo


Pittsburgh

“Pittsburgh” – “lies on” – “sits on”

“the Mon” – “the Monongahela”

Monongahela River

- Large data problem: verb forms are sparse!
- Can clustering help? [Gardner et al., EMNLP 2013]
- “lies on” -> C1
- “sits on” -> C1
- “overlooks” -> C2
Relation: cityLiesOnRiver
Path: canReferTo^{-1}, “C1”, canReferTo
Relation: cityLiesOnRiver
Path: canReferTo⁻¹, “C1”, canReferTo

“Steel City” canReferTo “C2” canReferTo “the Allegheny” canReferTo Allegheny River

“Pittsburgh” canReferTo “C1” canReferTo “the Mon” canReferTo Monongahela River

“Pittsburgh” canReferTo “C1” canReferTo “the Monongahela” canReferTo Monongahela River
Relation: cityLiesOnRiver
Path: canReferTo\(-1\), “C1”, canReferTo

“Steel City” canReferTo “C2” canReferTo Allegheny River

Pittsburgh canReferTo “the Allegheny”

“Pittsburgh” canReferTo “the Mon” canReferTo Monongahela River

“C1” canReferTo “the Monongahela”

“C1” canReferTo “C1” canReferTo “C1”
Relation: cityLiesOnRiver
Path: canReferTo⁻¹, “C1”, canReferTo

“Steel City” → “C2” → “the Allegheny”

“Pittsburgh” → “C1” → “the Mon” → “the Monongahela”
Relation: \textit{cityLiesOnRiver}

Path: \textit{canReferTo}^{-1}, \textit{“C1”}, \textit{canReferTo}, \textit{“C1”}

- "Steel City"
- "C1"
- Pittsburgh
- "the Allegheny"
- "the Mon"
- "the Monongahela"
- "Pittsburgh"
- "Monongahela River"
- "C1"
- "C1"

- Much better, but still can be sparse
- Use vector space similarity directly
  \cite{Gardner et al., EMNLP 2014}

- "lies on" $\to$ [.4, .3, -.1, .5]
- "sits on" $\to$ [.35, .3, -.2, .4]
- "overlooks" $\to$ [.2, .4, -.05, .2]
KB + Text Vectors

Relation: cityLiesOnRiver

Path: canReferTo^{-1}, “lies on”: [.4, .3, -.1, .5], canReferTo
Relation: cityLiesOnRiver
Path: canReferTo⁻¹, “lies on”: [.4, .3, -.1, .5], canReferTo

“Steel City” ➔ “the Allegheny”
“overlooks”: [.2, .4, -.05, .2]

Pittsburgh ➔ “the Mon”
“lies on”: [.4, 3, -.1, .5]

“Pittsburgh” ➔ “the Monongahela”
“sits on”: [.35, .3, -.2, .4]

Allegheny River ➔ “Pittsburgh”
Monongahela River ➔ “Pittsburgh”

Relation: cityLiesOnRiver

Path: canReferTo⁻¹, “lies on”: [.4, .3, -.1, .5], canReferTo

“Steel City” \(\xrightarrow{\text{canReferTo}}\) “the Allegheny” \(\xrightarrow{\text{canReferTo}}\) Allegheny River

“Pittsburgh” \(\xrightarrow{\text{canReferTo}}\) “Pittsburgh” \(\xrightarrow{\text{canReferTo}}\) “the Mon” \(\xrightarrow{\text{canReferTo}}\) Monongahela River

“the Monongahela” \(\xrightarrow{\text{canReferTo}}\) “the Mon” \(\xrightarrow{\text{canReferTo}}\) “Pittsburgh” \(\xrightarrow{\text{canReferTo}}\) “Steel City”
KB + Text Vectors

Relation: cityLiesOnRiver

Path: canReferTo⁻¹, "lies on": [.4, .3, -.1, .5], canReferTo

"Steel City" canReferTo

"Pittsburgh" canReferTo "overlooks": [.2, .4, -.05, .2]

"the Allegheny" canReferTo

Allegheny River

"lies on": [.4, .3, -.1, .5]

"sits on": [.35, .3, -.2, .4]

"the Mon" canReferTo

Monongahela River

"the Monongahela"

"Pittsburgh" canReferTo

"Steel City" canReferTo
Relation: cityLiesOnRiver

Path: canReferTo⁻¹, “lies on”: [.4, .3, -.1, .5], canReferTo

“Steel City” — “overlooks”: [.2, .4, -.05, .2] — “the Allegheny”

“Pittsburgh” — “lies on”: [.4, .3, -.1, .5] — “the Mon”

“Pittsburgh” — “sits on”: [.35, .3, -.2, .4] — “the Monongahela”

Allegheny River — canReferTo

Monongahela River — canReferTo

KB + Text Vectors
Empirical Results

**NELL MAP**

- KB
- +SVO
- +Clustered SVO
- +Vector SVO

**Freebase MAP**

- KB
- +SVO
- +Clustered SVO
- +Vector SVO

**NELL MRR**

- KB
- +SVO
- +Clustered SVO
- +Vector SVO

**Freebase MRR**

- KB
- +SVO
- +Clustered SVO
- +Vector SVO
Questions?