Learning Structured Embeddings of Knowledge Bases

Antoine Bordes
CNRS - UTC - UdeM

Ronan Collobert
IDIAP

&

Jason Weston
Google

Yoshua Bengio
Université de Montréal

The Learning Workshop, April 13, 2011
Context

- Fundamental challenge for AI: organize and make intelligent use of the colossal amounts of information generated daily.

- Several Knowledge Bases (KBs) are available, built for different purposes:
  - **Cyc**: perform human-like reasoning (1984+, millions of facts).
  - **WordNet**: produce intuitively usable dictionary and thesaurus, and support automatic text analysis (1984+, 200k+ words).
  - **Freebase**: create a global resource which allows to access common information effectively (2007+, 360M facts).

- Far more data available as raw text.
Motivation

- Besides their design goals, highly-structured databases could be useful in many AI areas such as NLP or computer vision.

- WordNet has been used widely in NLP, but other KBs have been less used so far.

- It seems hard to insert KBs data in other systems because their underlying symbolic frameworks are not flexible enough to be fruitfully exported.

→ We propose a way of leveraging the structured data in KBs into statistical learning systems.

→ Can possibly extend to raw text as well.
Distributed Embeddings

- **Main idea**: represent elements of any KB into a relatively low-dimensional embedding vector space.

- Previous work has demonstrated that encoding data in distributed embeddings *induce gains in performance*:
  
  - in NLP via the framework of language models (Y. Bengio et al., 03), (Collobert & Weston, 08).
  
  - for matching text queries and images (Weston, S. Bengio, Usunier, 10).
  
  - for language understanding using a (very) small custom KB. (Bordes et al. 10)

  - We will mention the relation modeling work of (Paccanaro and Hinton, 01) and (Sutskever et al, 09) later.
Knowledge Bases

- Our work considers Knowledge Bases as graph models.
  - The data structure is defined by a set of nodes and edges.
  - Each node corresponds to an entity.
  - Each edge corresponds to a relation type (there are several kinds) that are usually directed.

- A relation is denoted by \((e^l, r, e^r)\), where \(e^l\) is the left entity, \(e^r\) the right one and \(r\) the type of relation between them.

- We worked on two KBs: WordNet and Freebase.
We considered all WordNet entities connected with the following relation types:

<table>
<thead>
<tr>
<th>Statistics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation types</td>
<td>11</td>
</tr>
<tr>
<td>Entities</td>
<td>55,166</td>
</tr>
<tr>
<td>Train. triples</td>
<td>164,467</td>
</tr>
<tr>
<td>Test. triples</td>
<td>4,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relation types</th>
</tr>
</thead>
<tbody>
<tr>
<td>_synset_domain_topic</td>
</tr>
<tr>
<td>_domain_region</td>
</tr>
<tr>
<td>_domain_topic</td>
</tr>
<tr>
<td>_has_part</td>
</tr>
<tr>
<td>_part_of</td>
</tr>
<tr>
<td>_type_of</td>
</tr>
<tr>
<td>_has_instance</td>
</tr>
<tr>
<td>_subordinate_instance_of</td>
</tr>
<tr>
<td>_similar_to</td>
</tr>
<tr>
<td>_member_holonym</td>
</tr>
<tr>
<td>_member_meronym</td>
</tr>
</tbody>
</table>

Examples:
- (_door_1, _has_part, _lock_2),
- (_brain_1, _type_of, _neural_structure_1),
- (_auto_1, _has_instance, _s_u_v_1).

Note: WordNet is composed of lexical concepts, here we have disambiguated the words and denoted entities = word + sense ID.
We only considered the sub-graph defined by all relations involving at least one entity of the Freebase type deceased people.

<table>
<thead>
<tr>
<th>Statistics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation types</td>
<td>13</td>
</tr>
<tr>
<td>Entities</td>
<td>81,061</td>
</tr>
<tr>
<td>Train. triples</td>
<td>356,517</td>
</tr>
<tr>
<td>Test. triples</td>
<td>4,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relation types</th>
</tr>
</thead>
<tbody>
<tr>
<td>_place_lived</td>
</tr>
<tr>
<td>_place_of_birth</td>
</tr>
<tr>
<td>_place_of_death</td>
</tr>
<tr>
<td>_profession</td>
</tr>
<tr>
<td>_spouse</td>
</tr>
<tr>
<td>_parents</td>
</tr>
<tr>
<td>_children</td>
</tr>
<tr>
<td>_religion</td>
</tr>
<tr>
<td>_ethnicity</td>
</tr>
<tr>
<td>_gender</td>
</tr>
<tr>
<td>_cause_of_death</td>
</tr>
<tr>
<td>_nationality</td>
</tr>
<tr>
<td>_education_institution</td>
</tr>
</tbody>
</table>

Examples:
- (_marylin_monroe, _profession, _actress),
- (_pablo_picasso, _place_of_birth, _màlaga),
- (_john_f_kennedy, _religion, _catholicism).
Structured Embeddings

Basic model:

1. **Entities** are modeled in a $d$-dimensional embedding space.
   
   → The $i^{th}$ entity is assigned a vector $E_i \in \mathbb{R}^d$.

2. For any given relation type, a specific similarity measure captures that relation between entities. For example, _part_of_ would use one measure of similarity, whereas _similar_to_ would use another.

   → The $k^{th}$ relation is assigned a pair $R_k = (R_{k}^{lhs}, R_{k}^{rhs})$, where $R_{j}^{lhs}$ and $R_{j}^{rhs}$ are both $d \times d$ matrices.

3. The similarity function for a triple $(i, r_k, j)$ is finally:

   $$S_k(E_i, E_j) = \| R_{k}^{lhs} E_i - R_{k}^{rhs} E_j \|_1.$$
NN Architecture

This can be parametrized via a **Neural network**:

\[
f(e^l_i, r_i, e^r_i) = \|R^\text{lhs}_{r_i} E v(e^l_i) - R^\text{rhs}_{r_i} E v(e^r_i)\|_1
\]

- \(R^\text{lhs}\) and \(R^\text{rhs}\) are both \(d \times d \times D_r\) tensors,
- \(E\) is the matrix of the entities embeddings,
- \(v(n)\) maps the entity index \(n\) into a sparse vector.
NN Architecture

This can be parametrized via a Neural network:

\[ f(e_i^l, r_i, e_i^r) = \| R_{r_i}^{lhs} E v(e_i^l) - R_{r_i}^{rhs} E v(e_i^r) \|_1 \]

- \(R^{lhs}\) and \(R^{rhs}\) are both \(d \times d \times D_r\) tensors,
- \(E\) is the matrix of the entities embeddings,
- \(v(n)\) maps the entity index \(n\) into a sparse vector.

Hence, to score a triple \((e_i^l, r_i, e_i^r)\):

\((\_\text{door\_1}, \_\text{has\_part}, \_\text{lock\_2})\)
NN Architecture

This can be parametrized via a Neural network:

\[ f(e^l_i, r_i, e^r_i) = \| R^{lhs}_{r_i} E v(e^l_i) - R^{rhs}_{r_i} E v(e^r_i) \|_1 \]

- \( R^{lhs} \) and \( R^{rhs} \) are both \( d \times d \times D_r \) tensors,
- \( E \) is the matrix of the entities embeddings,
- \( v(n) \) maps the entity index \( n \) into a sparse vector.

Hence, to score a triple \((e^l_i, r_i, e^r_i)\):
1. select the \((e^l_i)^{th}\) and \((e^r_i)^{th}\) columns of \( E \),
NN Architecture

This can be parametrized via a Neural network:

\[ f(e^l_i, r_i, e^r_i) = ||R^\text{lhs}_r Ev(e^l_i) - R^\text{rhs}_r Ev(e^r_i)||_1 \]

- \( R^\text{lhs} \) and \( R^\text{rhs} \) are both \( d \times d \times D_r \) tensors,
- \( E \) is the matrix of the entities embeddings,
- \( v(n) \) maps the entity index \( n \) into a sparse vector.

Hence, to score a triple \((e^l_i, r_i, e^r_i)\):
1. select the \((e^l_i)_\text{th}\) and \((e^r_i)_\text{th}\) columns of \( E \),
2. transform them by the \( d \times d \) left- and right-hand side matrices of \( r_i \),
NN Architecture

This can be parametrized via a Neural network:

\[ f(e^l_i, r_i, e^r_i) = \| R^{lhs}_{r_i} Ev(e^l_i) - R^{rhs}_{r_i} Ev(e^r_i) \|_1 \]

- \( R^{lhs} \) and \( R^{rhs} \) are both \( d \times d \times D_r \) tensors,
- \( E \) is the matrix of the entities embeddings,
- \( v(n) \) maps the entity index \( n \) into a sparse vector.

Hence, to score a triple \( (e^l_i, r_i, e^r_i) \):

1. select the \( (e^l_i)_{th} \) and \( (e^r_i)_{th} \) columns of \( E \),
2. transform them by the \( d \times d \) left- and right-hand side matrices of \( r_i \).
NN Architecture

This can be parametrized via a Neural network:

\[ f(e_i^l, r_i, e_i^r) = ||R_{r_i}^{lhs} Ev(e_i^l) - R_{r_i}^{rhs} Ev(e_i^r)||_1 \]

- \( R^{lhs} \) and \( R^{rhs} \) are both \( d \times d \times D_r \) tensors,
- \( E \) is the matrix of the entities embeddings,
- \( v(n) \) maps the entity index \( n \) into a sparse vector.

Hence, to score a triple \((e_i^l, r_i, e_i^r)\):
1. select the \((e_i^l)^{th}\) and \((e_i^r)^{th}\) columns of \( E \),
2. transform them by the \( d \times d \) left- and right-hand side matrices of \( r_i \),
3. measure the 1-norm distance in-between.
NN Architecture

This can be parametrized via a Neural network:

\[ f(e^l_i, r_i, e^r_i) = ||R_{\text{lhs}}^{\text{r}} Ev(e^l_i) - R_{\text{rhs}}^{\text{r}} Ev(e^r_i)||_1 \]

- \( R^{\text{lhs}} \) and \( R^{\text{rhs}} \) are both \( d \times d \times D \) tensors,
- \( E \) is the matrix of the entities embeddings,
- \( v(n) \) maps the entity index \( n \) into a sparse vector.

Hence, to score a triple \( (e^l_i, r_i, e^r_i) \):
1. select the \( (e^l_i) \)th and \( (e^r_i) \)th columns of \( E \),
2. transform them by the \( d \times d \) left- and right-hand side matrices of \( r_i \),
3. measure the 1-norm distance in-between.
**Intuition**: if the left- or right-hand side entities of a triplet were missing, we would like our model to predict it correctly. For example, this would allow us to answer questions like “what is part of a car?” or “where was Audrey Hepburn born?”.

Hence, for any training triplet $x_i = (e^l_i, r_i, e^r_i)$ we would like:

$$f(e^l_i, r_i, e^r_i) < f(e^l_j, r_i, e^r_j), \quad \forall j : (e^l_j, r_i, e^r_j) \notin x$$  \hspace{1cm} (1)

and

$$f(e^l_i, r_i, e^r_i) < f(e^l_i, r_j, e^r_j), \quad \forall j : (e^l_i, r_j, e^r_j) \notin x. \hspace{1cm} (2)$$

That is, the function $f$ is trained to rank all the training samples below all other triplets.
NN Training – Algorithm

To train the parameters $R_{lhs}$, $R_{rhs}$ and $E$ of our model we use stochastic gradient descent:

1. Randomly select a **positive training triplet** $x_i = (e^l_i, r_i, e^r_i)$.

2. Randomly select **either constraint (1) or (2)** and an entity $e^{neg}$:
   - **If constraint (1)**, construct the negative triplet $x^{neg} = (e^{neg}_i, r_i, e^r_i)$.
   - **Else if constraint (2)**, construct $x^{neg} = (e^l_i, r_i, e^{neg}_i)$ instead.

3. **If** $f(x_i) > f(x^{neg}) + 1$ **make a gradient step** to minimize:
   \[
   \max(0, 1 - f(x^{neg}) + f(x_i)).
   \]

4. Enforce the constraints that each column $\|E_i\| = 1$, $\forall i$.

Note: the normalization in step 4 helps remove scaling freedoms.
• **Problem:** the original symbolic framework asserts that all existing relations are true facts.

→ When the data is transferred in the embedding space, this is lost.

• **Remedy:** estimate the probability density at any point of the defined embedding space using **Kernel Density Estimation**.

→ KDE bases its estimation on the training points, so they get a high probability density.

• We define a **KDE estimator** $f_{kde}$ which allows to estimate the density for any triplet and can also be used for **prediction**.
Related Work

• *Linear Relational Embedding* (Paccanaro and Hinton, 01) is close to this except it uses a different loss and was applied to relatively small arithmetic tasks and a “family relation” problem.

• (Sutskever et. al, 09) proposed to learn a factorized representation of relations in a nonparametric Bayesian clustering framework for relational data. This work defines 2 embeddings per entity: “The disadvantage of using two vectors for each object is that the model cannot as easily capture the position-independent properties of the object”.

→ One of our main goal is to link the work above to the NLP embedding trend mentioned at the start of the talk.
Empirical Evaluation

• We assess the quality of our representations via a ranking task. For any triplet \((e^l, r, e^r)\):
  1. remove \(e^l\) (similar procedure is done with \(e^r\)),
  2. compute densities \(f_{kde}((e, r, e^r))\) for all \(e \in D_e\),
  3. sort values by decreasing order,
  4. record the rank of the correct entity \(e^l\).

• Our method, \(\text{Emb}_{MT+KDE}\), is compared against:
  - \(\text{Emb}_{MT}\): same embeddings but without KDE.
  - \(\text{Emb}\): embeddings have been learnt without multi-tasking (i.e. there is a different matrix \(E\) per relation type).
  - \(\text{Counts}\): no learning but ranks based on counting the appearance of pairs \((e^l, r)\) and \((r, e^r)\) in the training set.

• Hyperparameters: \(d = 50\), training during \(\approx 3\) days.
Empirical Evaluation – Ranking

Predicted ranks in test on WordNet and Freebase:

<table>
<thead>
<tr>
<th></th>
<th>WordNet</th>
<th>Freebase</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rank $e^f$</td>
<td>rank $e^r$</td>
</tr>
<tr>
<td>Counts</td>
<td>Train</td>
<td>662.7</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>6202.3</td>
</tr>
<tr>
<td>Emb</td>
<td>Train</td>
<td>16.2</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>3414.7</td>
</tr>
<tr>
<td>Emb$_{MT}$</td>
<td>Train</td>
<td>13.6</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>97.3</td>
</tr>
<tr>
<td>Emb$_{MT}$+KDE</td>
<td>Train</td>
<td><strong>11.8</strong></td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td><strong>87.8</strong></td>
</tr>
</tbody>
</table>

- Counts and Emb record information about train examples.
- But, both Emb$_{MT}$ and Emb$_{MT}$+KDE perform much better on test examples → generalization.
Empirical Evaluation - Generalization

- **Multi-tasking helps generalization:** information coming from different relations is encoded in the embeddings of entities.
- **Lists of** $e^r$ **and** $e^l$ **predicted using** $\text{Emb}_{MT+KDE}$ **after training on WordNet.** (All elements from the training set have been removed).

<table>
<thead>
<tr>
<th>$e^l$</th>
<th>_everest_1</th>
<th>_brain_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td><em>part_of</em></td>
<td>_has_part</td>
</tr>
<tr>
<td>$e^r$</td>
<td>_north_vietnam_1</td>
<td>_subthalaric_nucleus_1</td>
</tr>
<tr>
<td></td>
<td>_hindu_kush_1</td>
<td>_cladode_1</td>
</tr>
<tr>
<td></td>
<td>_karakoram_1</td>
<td>_subthalamus_1</td>
</tr>
<tr>
<td></td>
<td>_federal_2</td>
<td>_fluid_ounce_1</td>
</tr>
<tr>
<td></td>
<td>_burma_1</td>
<td>_sympathetic_nervous_system_1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$e^l$</th>
<th>_judgement_3</th>
<th>_thing_13</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td><em>type_of</em></td>
<td>_has_instance</td>
</tr>
<tr>
<td>$e^r$</td>
<td>_deciding_1</td>
<td>_language_1</td>
</tr>
</tbody>
</table>
Empirical Evaluation - Entities

Plot of embeddings of 650 WordNet entities projected using t-SNE.
Empirical Evaluation - Entities

Zoom 1: geographical entities.
Empirical Evaluation - Entities

Zoom 2: actions, people and objects of different domains
Empirical Evaluation - Entities

**Zoom 3:** mostly religion.
**Extension for Knowledge Extraction**

- Our model could be useful for **extracting knowledge from text**.

- **Illustration:** we conducted our own knowledge extraction:
  1. perform Semantic Role Labeling on 40,000 Wikipedia articles.
  2. keep only phrases following the scheme **subject-verb-direct object**.
  3. remove adjectives, adverbs and pronouns and stem the **verb**.
  4. create a dataset with triplets containing the 100 most frequents verbs.

  **Collected data:** 154,438 triplets, 100 relation types and 23,936 entities.

- **Example of lists of** $e^r$ **predicted for** $e^l = "people"$:

<table>
<thead>
<tr>
<th>$e^l$</th>
<th>people</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>build</td>
</tr>
<tr>
<td>$e^r$</td>
<td>livelihoods</td>
</tr>
<tr>
<td></td>
<td>homes</td>
</tr>
<tr>
<td></td>
<td>altars</td>
</tr>
<tr>
<td></td>
<td>houses</td>
</tr>
<tr>
<td></td>
<td>ramps</td>
</tr>
</tbody>
</table>
Conclusion

- We introduced a method to automatically learn structured distributed embeddings of KBs.
  - These new representations are compact and can be trained on large KBs.
  - Using KDE allows to estimate the probability density of any relation triple.
- Experiments show that our encoding preserves the knowledge of the data, and makes generalization possible.
- We can adapt our approach on raw text for knowledge extraction.