# 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.

# KB 1 - WordNet

We considered all WordNet entities connected with the following relation types:

| Statistics | |
|---|---|
| Relation types | 11 |
| Entities | 55,166 |
| Train. triples | 164,467 |
| Test. triples | 4,000 |

| Relation types |
|---|
| *_synset_domain_topic* |
| *_domain_region* |
| *_domain_topic* |
| *_has_part* |
| *_part_of* |
| *_type_of* |
| *_has_instance* |
| *_subordinate_instance_of* |
| *_similar_to* |
| *_member_holonym* |
| *_member_meronym* |

**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.

# KB 2 - Freebase

We only considered the sub-graph defined by all relations involving at least one entity of the Freebase type *deceased people*.

| Statistics | |
|---|---|
| Relation types | 13 |
| Entities | 81,061 |
| Train. triples | 356,517 |
| Test. triples | 4,000 |

| Relation types |
|---|
| *_place_lived* |
| *_place_of_birth* |
| *_place_of_death* |
| *_profession* |
| *_spouse* |
| *_parents* |
| *_children* |
| *_religion* |
| *_ethnicity* |
| *_gender* |
| *_cause_of_death* |
| *_nationality* |
| *_education_institution* |

**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:

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

- $R^{lhs}$ 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.

( _door_1, _has_part, _lock_2 )

# 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 $(e_i^l, r_i, e_i^r)$:

( _door_1, _has_part, _lock_2 )

# 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 $(e_i^l, r_i, e_i^r)$:

1. select the $(e_i^l)^{th}$ and $(e_i^r)^{th}$ columns of $E$,

# NN Architecture

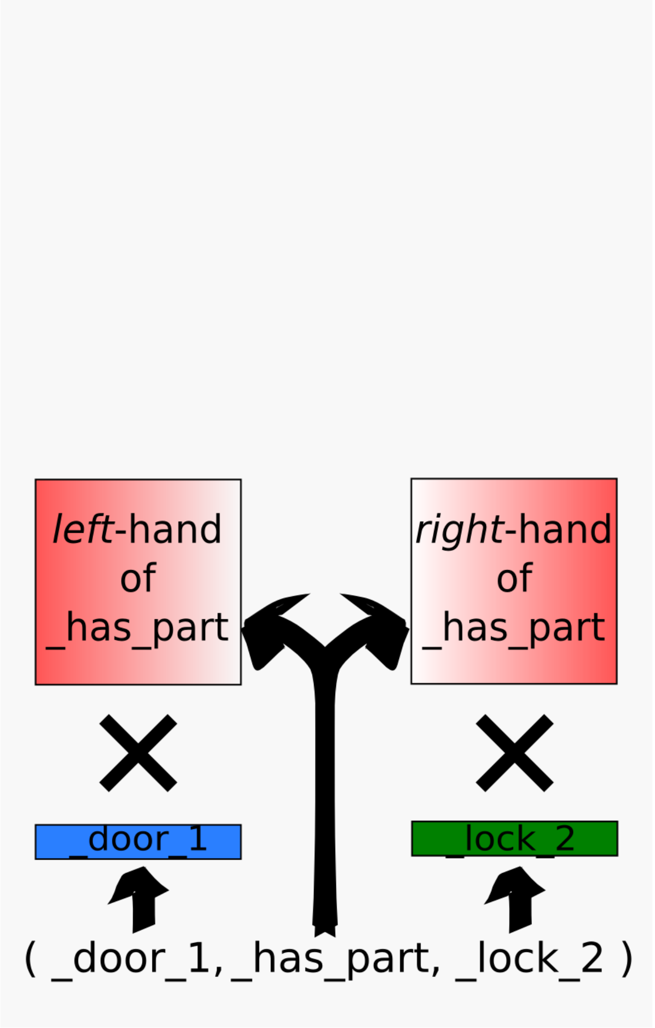
This can be parametrized via a Neural network:

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

- $R^{lhs}$ 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 $r_i$,



# NN Architecture

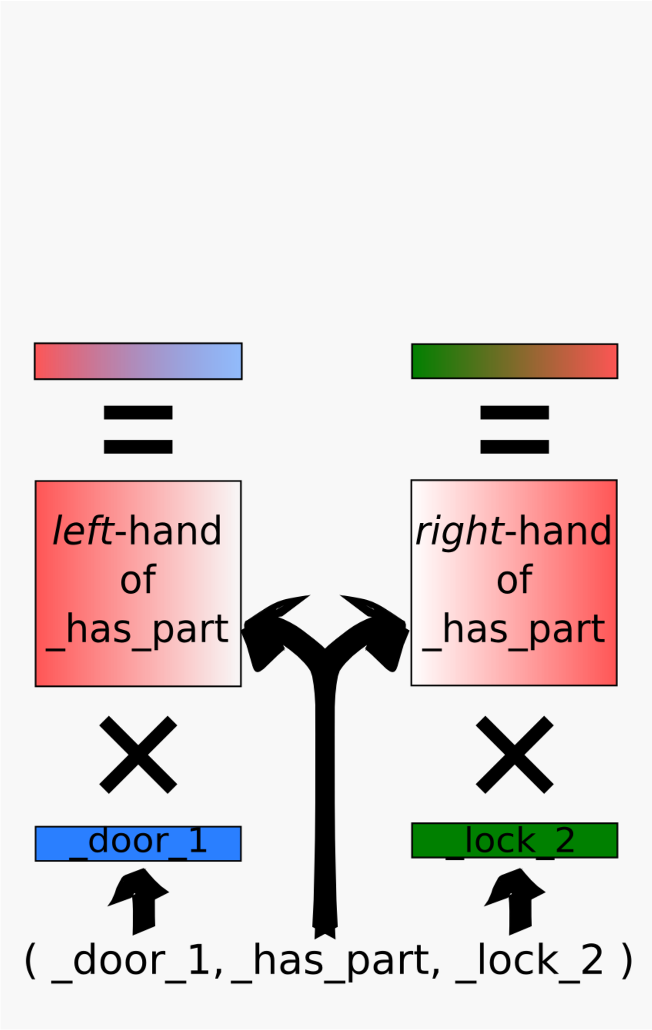
This can be parametrized via a Neural network:

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

- $R^{lhs}$ 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 $(e_i^l, r_i, e_i^r)$:

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 $r_i$,



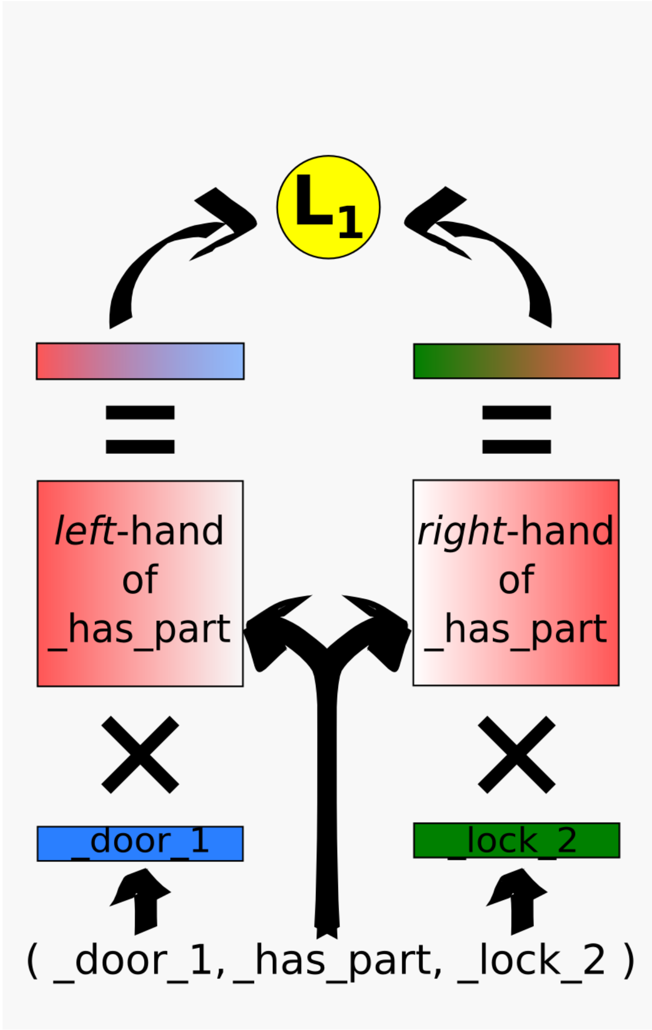
# NN Architecture

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 $R^{lhs}$ 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.



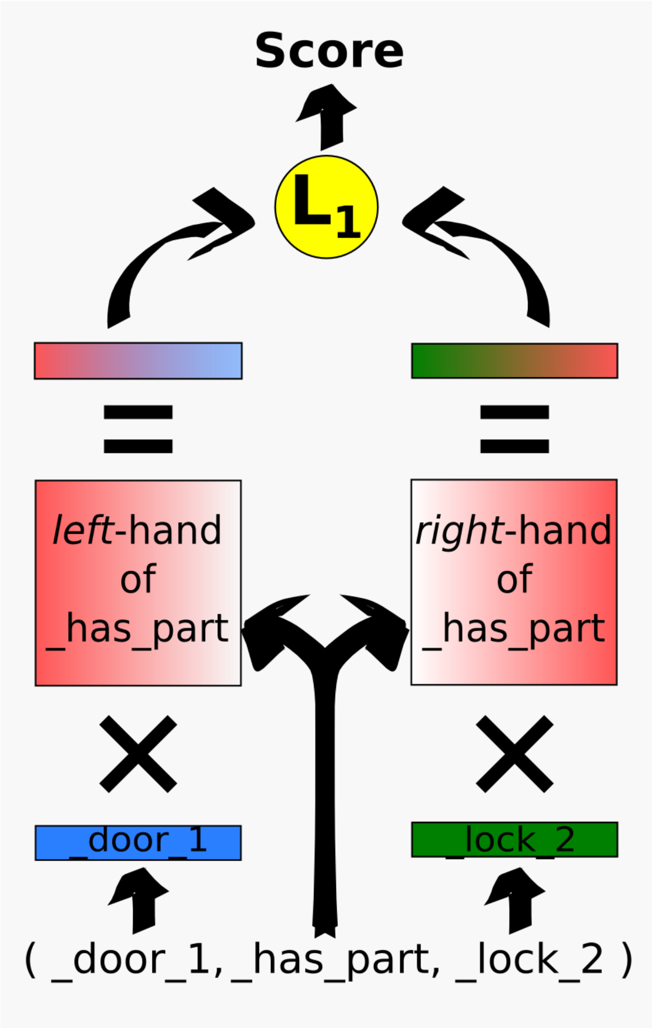
# NN Architecture

This can be parametrized via a Neural network:

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

- $R^{lhs}$ and $R^{lhs}$ 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 Training - Constraints

**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 $x_i = (e_i^l, r_i, e_i^r)$ we would like:

$$f(e_i^l, r_i, e_i^r) < f(e_j^l, r_i, e_i^r), \quad \forall j : (e_j^l, r_i, e_i^r) \notin x \tag{1}$$

and

$$f(e_i^l, r_i, e_i^r) < f(e_i^l, r_i, e_j^r), \quad \forall j : (e_i^l, r_i, e_j^r) \notin x. \tag{2}$$

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

# Probability Landscape Estimation

- **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, Emb$_{MT}$+KDE, is compared against:

  \- Emb$_{MT}$: same embeddings but without KDE.

  \- Emb: embeddings have been learnt without multi-tasking (i.e. there is a different matrix $E$ per relation type).

  \- 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:

| | | WordNet | | Freebase |
|---|---|---|---|---|
| | | rank $e^l$ | rank $e^r$ | rank $e^r$ |
| Counts | *Train* | 662.7 | 804.1 | 541.8 |
| | *Test* | 6202.3 | 5894.2 | 804.9 |
| Emb | *Train* | 16.2 | 23.3 | – |
| | *Test* | 3414.7 | 3380.8 | – |
| Emb$_{MT}$ | *Train* | 13.6 | 20.9 | 2.9 |
| | *Test* | 97.3 | 223.0 | 317.2 |
| Emb$_{MT}$+KDE | *Train* | **11.8** | **19.9** | **1.6** |
| | *Test* | **87.8** | **192.5** | **314.5** |

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

# Empirical Evaluation - Generalization

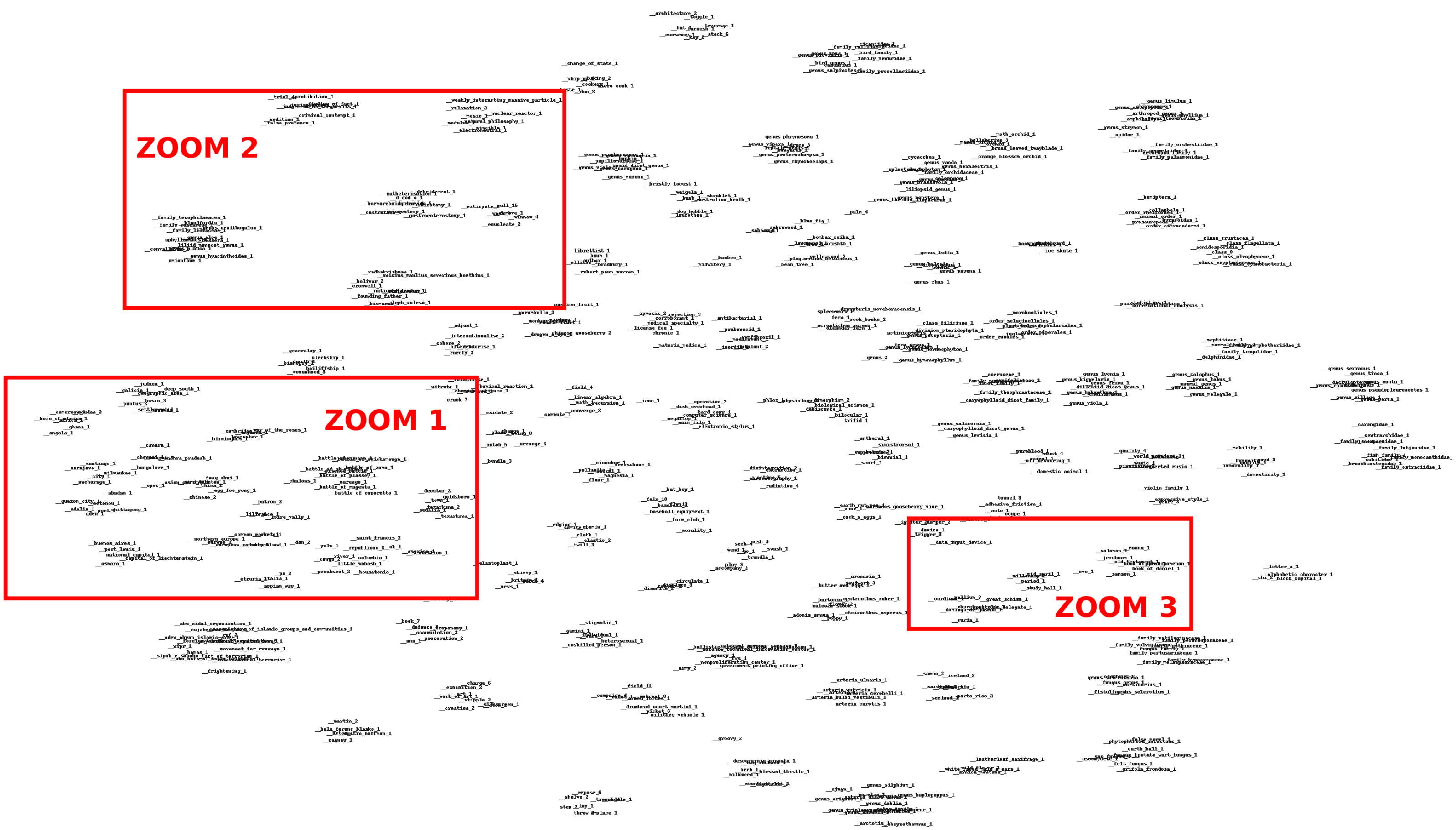
- Multi-tasking helps generalization: information coming from different relations is encoded in the embeddings of entities.
- Lists of $e^r$ and $e^l$ predicted using Emb$_{MT}$+KDE after training on WordNet. (All elements from the training set have been removed).

| | | |
|---|---|---|
| $e^l$ | _everest_1 | _brain_1 |
| $r$ | **_part_of** | **_has_part** |
| $e^r$ | *_north_vietnam_1*<br>*_hindu_kush_1*<br>*_karakoram_1*<br>*_federal_2*<br>*_burma_1* | *_subthalamic_nucleus_1*<br>*_cladode_1*<br>*_subthalamus_1*<br>*_fluid_ounce_1*<br>*_sympathetic_nervous_system_1* |

| | | |
|---|---|---|
| $e^l$ | *_judgement_3*<br>*_delayed_action_1*<br>*_experience_5*<br>*_bawl_out_1*<br>*_carry_over_1* | *_thing_13*<br>*_transfer_5*<br>*_situation_1*<br>*_illness_1*<br>*_cognition_1* |
| $r$ | **_type_of** | **_has_instance** |
| $e^r$ | _deciding_1 | _language_1 |

# Empirical Evaluation - Entities

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



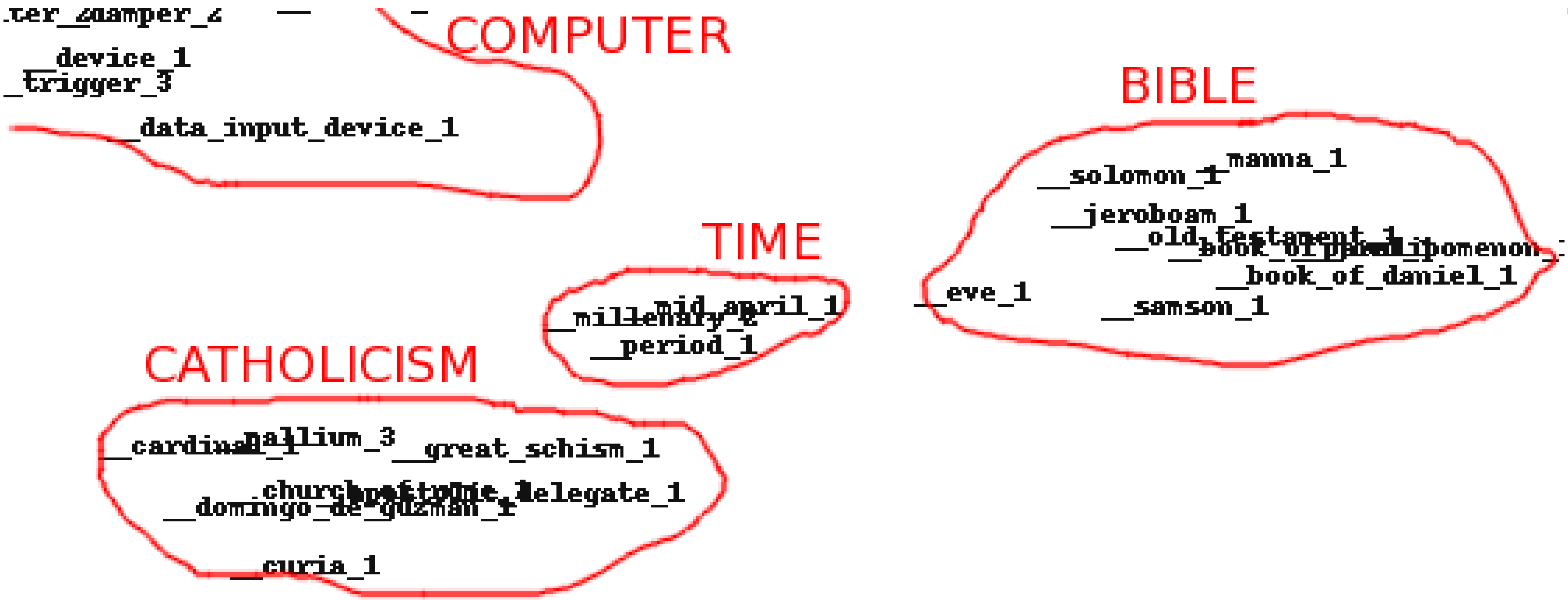
# Empirical Evaluation - Entities


**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*":

| $e^l$ | people | | | | |
|---|---|---|---|---|---|
| $r$ | **build** | **destroy** | **won** | **suffer** | **control** |
| $e^r$ | *livelihoods* | *icons* | *emmy* | *sores* | *rocket* |
| | *homes* | *virtue* | *award* | *agitation* | *stores* |
| | *altars* | *donkeys* | *everything* | *treatise* | *emotions* |
| | *houses* | *cowboy* | *standings* | *eczema* | *spending* |
| | *ramps* | *chimpanzees* | *pounds* | *copd* | *fertility* |

# 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.