GMatching

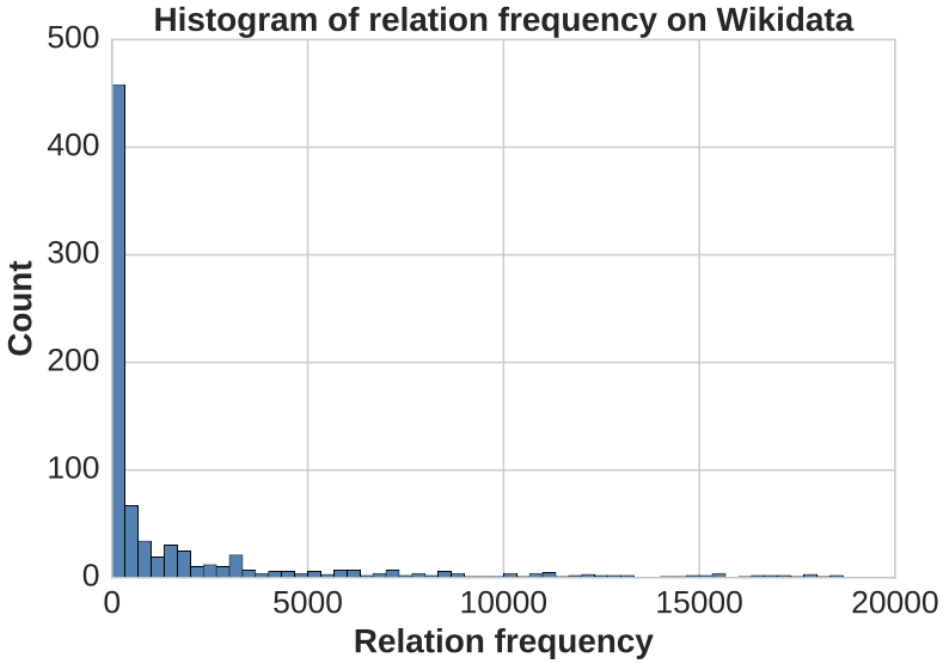
One-Shot Relational Learning for Knowledge Graphs

Introduction & related work

- Introduction
- Contributions
- Related work

Introduction

- Problem:
  - datasets (e.g. FB15k, WN18) used by previous models mostly only cover common relations in KGs
- key properties of KG completion models
  - a large portion of KG relations are actually sparse long-tail
  - Realworld KGs are often dynamic and evolving at any given moment



Contributions

- the first to consider the long-tail relations in the link prediction task and formulate the problem as few-shot relational learning
- propose an effective one-shot learning framework for relational data, which achieves better performance than various embedding-based methods .
- present two newly constructed datasets for the task of one-shot knowledge graph completion.

Related work——Embedding Models

- RESCAL(2011)
  - models the relationship using tensor operations
- DistMult(2014)、 ConvE (2017)
  - assume enough training instances for all relations and entities
  - do not pay attention to those sparse symbols
- ConMASK (2018)
  - handle unseen entities by leveraging text descriptions

Related work——Few-Shot Learning

- metric based approaches
  - learn generalizable metrics and corresponding matching functions from a set of training tasks
  - adopt the general matching framework proposed in deep siamese network
  - multi-metric based approach for text classification (language domain 2018)
- meta-learner based approaches
  - learn the optimization of model parameters
    - outputting the parameter updates
    - directly predicting the model parameters
  - LSTM- based meta-learner (2017)
    - learns the step size for each dimension of the stochastic gradients
- Bayesian Program Induction (2015)
  - represents concepts as simple programs that best explain observed examples under a Bayesian criterion

Algorithm

- Problem Formulation
- One-Shot Learning Settings
- Model
- Loss Function and Training

Problem Formulation

- Knowledge graph :

$$\{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E},$$

- knowledge graph completion
  - predict unseen relations r between two existing entities (h, ?, t)
  - predict the tail entity t given the head entity and the query relation (h, r, ?)
- goal:
  - rank the true tail entity t_{true} higher than other candidate entities $t \in C_{h,r}$, given only an example triple
- predict new facts of the relation r, only consider a closed set of entities
  - Eg: no unseen entities during testing
- future work
  - For open-world settings where new entities might appear during testing, external information such as text descriptions about these entities are usually required

One-Shot Learning Settings

- Meta-training set $\mathbb{T}_{meta-train}$
  - Every KG relations r has its own training/testing triples $\{D_r^{train}, D_r^{test}\}$
- Each D_r^{train} has only one triple (h_0, r, t_0)
- $D_r^{test} = \{(h_i, r, t_i, \mathcal{C}_{h_i,r})\}$
  - Ground-truth tail entities t_i for each query (h_i,r)
  - corresponding tail entity candidates $\mathcal{C}_{h_i,r} = \{t_{ij}\}$
- Loss function:

  $\ell_\theta(h_i, r, t_i | \mathcal{C}_{h_i,r}, D_r^{train})$
- Meta-training objective

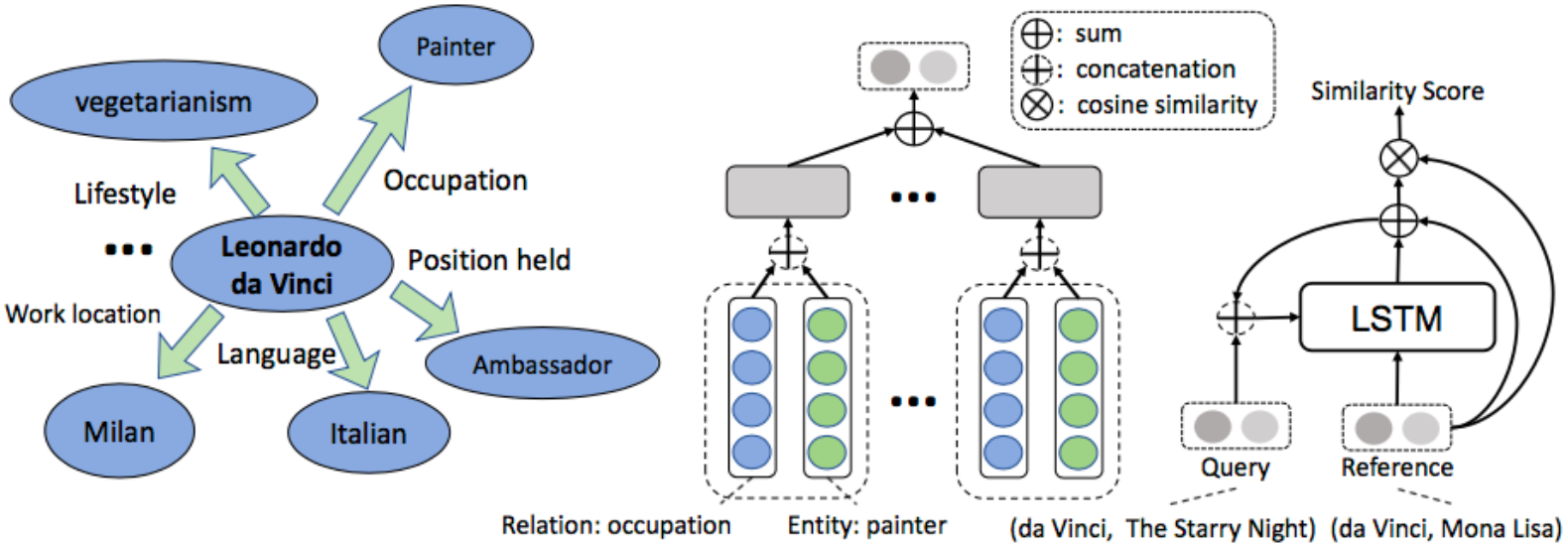
$$\min_\theta \mathbb{E}_{T_r} \left[\sum_{(h_i,r,t_i,\mathcal{C}_{h_i,r}) \in D_r^{test}} \frac{\ell_\theta(h_i, r, t_i | \mathcal{C}_{h_i,r}, D_r^{train})}{|D_r^{test}|} \right]$$

One-Shot Learning Settings

- Meta-testing set $\mathbb{T}_{meta-test}$
  - meta-testing relations are unseen from meta-training
  - $D_{r'}^{train}$ and $D_{r'}^{test}$ defined in the same way as in meta-training
- Meta-validation set $\mathbb{T}_{meta-validation}$
  - Because the meta-testing relations do not have validation sets like in the traditional machine learning setting
- background knowledge graph G′ :
  - subset of G with all the relations removed

Model

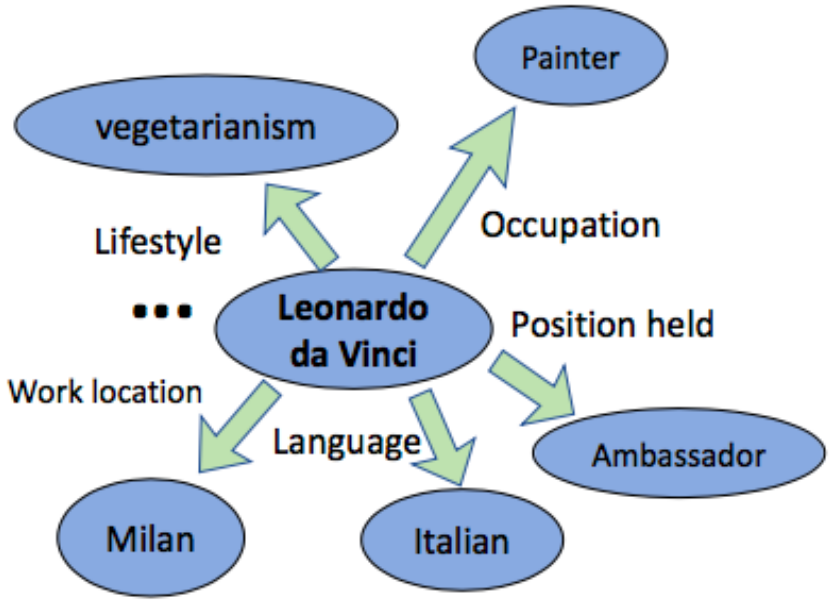
- Similarity function: $\mathcal{M}((h,t),(h',t')|\mathcal{G}')$
  - One known fact (h_0, r, t_0)
  - predict the likelihood of testing triples based on the matching score
- sub-problems
  - the representations of entity pairs;
  - the comparison function between two entity-pair representations.



Neighbor encoder

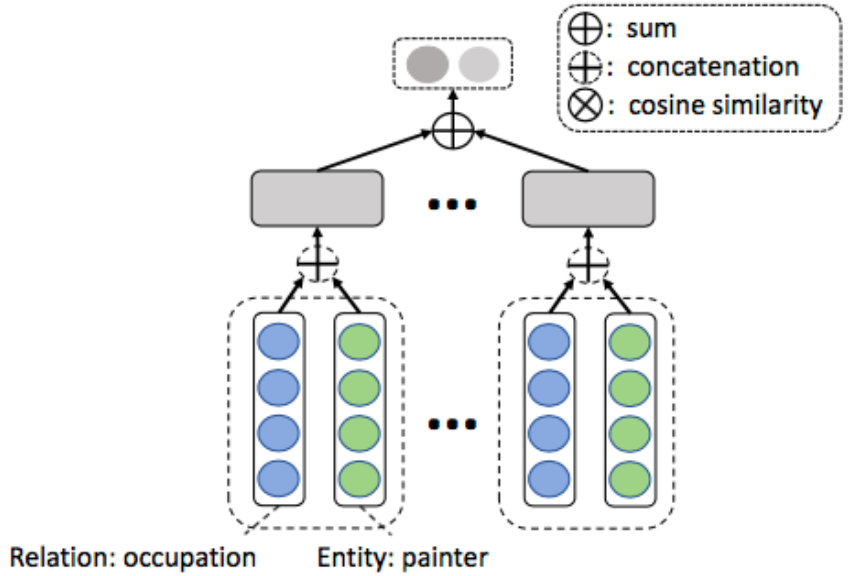
- Modeling the structural patterns:
  - use a neighbor encoder to incorporate graph structures into metric-learning model
  - Only considers entities' local connections, *i.e.* the one-hop neighbors (efficiency)
- Neighbor set: $\mathcal{N}_e = \{(r_k, e_k) | (e, r_k, e_k) \in \mathcal{G}'\}$
  - encode N_e and output a vector as the latent representation of e.
  - **Need**: invariant to permutations & insensitive to the size of the neighbor set .
- Function f: (Zaheer et al., 2017)

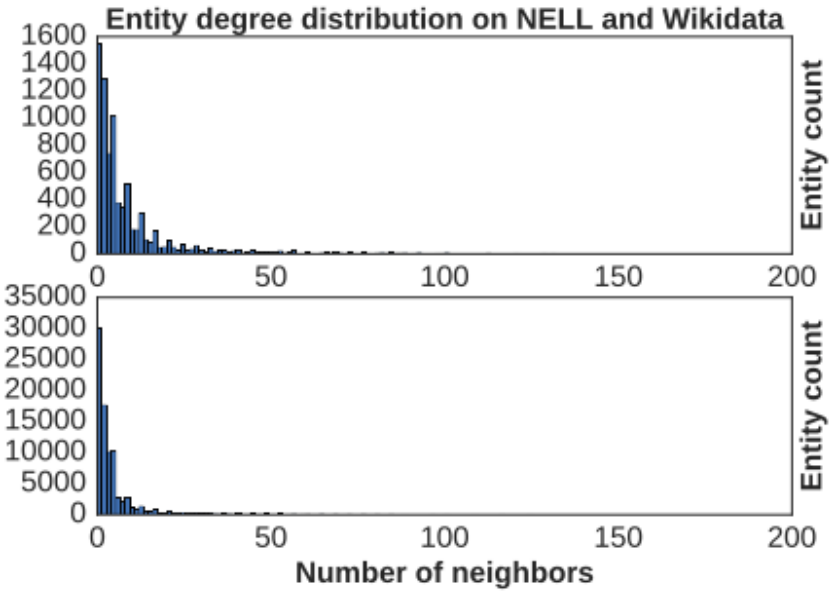
$$f(\mathcal{N}_e) = \sigma\left(\frac{1}{|\mathcal{N}_e|} \sum_{(r_k, e_k) \in \mathcal{N}_e} C_{r_k, e_k}\right).$$



Neighbor encoder

- Embedding layer **emb** with dimension d:
  - can be pre-trained using existing embedding-based models
  - $v_{r_k} = \mathbf{emb}(r_k), v_{e_k} = \mathbf{emb}(e_k)$
- Dropout & feed-forward layer :
  - $C_{r_k,e_k} = W_c(v_{r_k} \oplus v_{e_k}) + b_c$
- enable batching during training
  - specific the maximum number of neighbors
  - use all-zero vectors as “dummy” neighbors
  - degree distribution is usually very concentrated
- different from RGCN(2017)
  - only encode the local graphs
  - perform one-step propagation
  - operate on pre-trained graph embeddings





Matching processor

- process:
  - applying f(N_e) to the reference entity pair (h_0,t_0) and any query entity pair (h_i, t_{ij})
  - Get neighbor vectors for each $[f(\mathcal{N}_{h_0}); f(\mathcal{N}_{t_0})]$ $[f(\mathcal{N}_{h_i}); f(\mathcal{N}_{t_{ij}})]$
- LSTM-based recurrent “processing” block (2015,2016)
  - For every query (h_i, r, ?), by comparing (h_i, t_{ij}) with (h_0, t_0), we can get the ranking scores for every t_{ij}

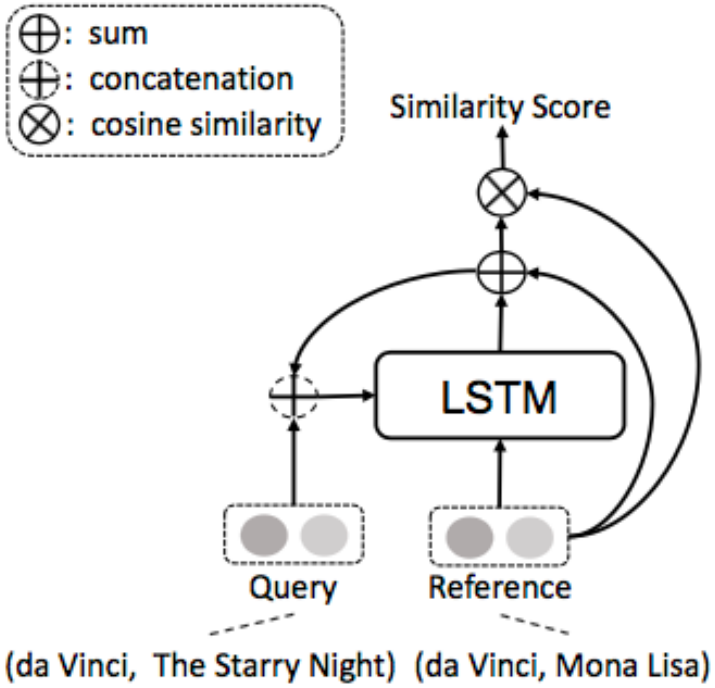
$$h'_{k+1}, c_{k+1} = LSTM(q, [h_k \oplus s, c_k])$$

$$h_{k+1} = h'_{k+1} + q$$

$$score_{k+1} = \frac{h_{k+1} \odot s}{\|h_{k+1}\| \, \|s\|},$$

$$s = f(\mathcal{N}_{h_0}) \oplus f(\mathcal{N}_{t_0})$$

$$q = f(\mathcal{N}_{h_i}) \oplus f(\mathcal{N}_{t_{ij}})$$



Loss Function and Training

- Loss function:
  - positive (true) query triples $\{(h_i, r, t_i^+)|(h_i, r, t_i^+) \in \mathcal{G}\}$
  - group negative (false) query triples by polluting the tail entities. $\{(h_i, r, t_i^-)|(h_i, r, t_i^-) \notin \mathcal{G}\}$
  - Hinge loss function $\ell_\theta = max(0, \gamma + score_\theta^- - score_\theta^+)$
- every training episode
  - sample one task/relation T_r from the meta-training set
  - from all the known triples in T_r ,sample one triple as the reference/training triple D_r^{train} and a batch of other triples as the positive query/test triples D_r^{test}.

Algorithm

Algorithm 1 One-shot Training

Input:
a) Meta-training task set $\mathbb{T}_{meta-training}$;
b) Pre-trained KG embeddings (excluding relation in $\mathbb{T}_{meta-training}$);
c) Initial parameters θ of the metric model;
for epoch = 0:M-1 **do**
    Shuffle the tasks in $\mathcal{T}_{meta-learning}$
    for T_r in $\mathcal{T}_{meta-learning}$ **do**
        Sample one triple as the reference
        Sample a batch B^+ of query triples
        Pollute the tail entity of query triples to get B^-
        Calculate the matching scores for triple in B^+ and B^-
        Calculate the batch loss $\mathcal{L} = \sum_B \ell$
        Update θ using gradient $g \propto \nabla \mathcal{L}$
    end for
end for

PART 3

Experiment

- Datasets
- Link prediction
- Analysis on Neighbor-Encoder & Ablation Studies
- Performance on Different Relations

Datasets

- rest of the relations
  - as background relations, their triples provide important background knowledge to match entity pairs
- datasets
  - select the relations with less than 500 but more than 50 triples as one-shot tasks
- training/validation/testing
  - NELL-One: 51/5/11 task relations
  - Wiki-One: 133:16:34 division ratio

Dataset	# Ent.	# R.	# Triples	# Tasks
NELL-One	68,545	358	181,109	67
Wiki-One	4,838,244	822	5,859,240	183

Table 1: Statistics of the Datasets. # Ent. denotes the number of unique entities and # R. denotes the number of all relations. # Tasks denotes the number of relations we use as one-shot tasks.

Link Prediction

	NELL-One				Wiki-One			
Model	MRR	Hits@10	Hits@5	Hits@1	MRR	Hits@10	Hits@5	Hits@1
RESCAL	.071/.140	.100/.229	.082/.186	.048/.089	.119/.072	.167/.082	.132/.062	.093/.051
TransE	.082/.093	.177/.192	.126/.141	.032/.043	.023/.035	.036/.052	.029/.043	.015/.025
DistMult	.075/.102	.128/.177	.093/.126	.045/.066	.042/.048	.086/.101	.055/.070	.017/.019
ComplEx	.072/.131	.128/.223	.041/.086	.041/.086	.079/.069	.148/.121	.106/.092	.046/.040
GMatching (RESCAL)	.144/**.188**	.277/.305	.216/.243	.087/**.133**	.113/.139	.330/.305	.180/.228	.033/.061
GMatching (TransE)	**.168**/.171	.293/.255	**.239**/.210	**.103**/.122	.167/.219	.349/.328	.289/.269	.083/.163
GMatching (DistMult)	.119/.171	.238/.301	.183/.221	.054/.114	.190/**.222**	**.384/.340**	**.291**/.271	.114/**.164**
GMatching (ComplEx)	.132/.185	**.308/.313**	.232/**.260**	.049/.119	**.201**/.200	.350/.336	.231/**.272**	**.141**/.120
GMatching (Random)	.083/.151	.211/.252	.135/.186	.024/.103	.174/.198	.309/.299	.222/.260	.121/.133

Table 2: Link prediction results on validation/test relations. KG embeddings baselines are shown at the top of the table and our one-shot learning (GMatching) results are shown at the bottom. **Bold** numbers denote the best results on meta-validation/meta-test. Underline numbers denote the model selection results from all KG embeddings baselines, or from all one-shot methods, i.e. selecting the method with the best validation score and reporting the corresponding test score.

Analysis on Neighbor-Encoder & Ablation Studies

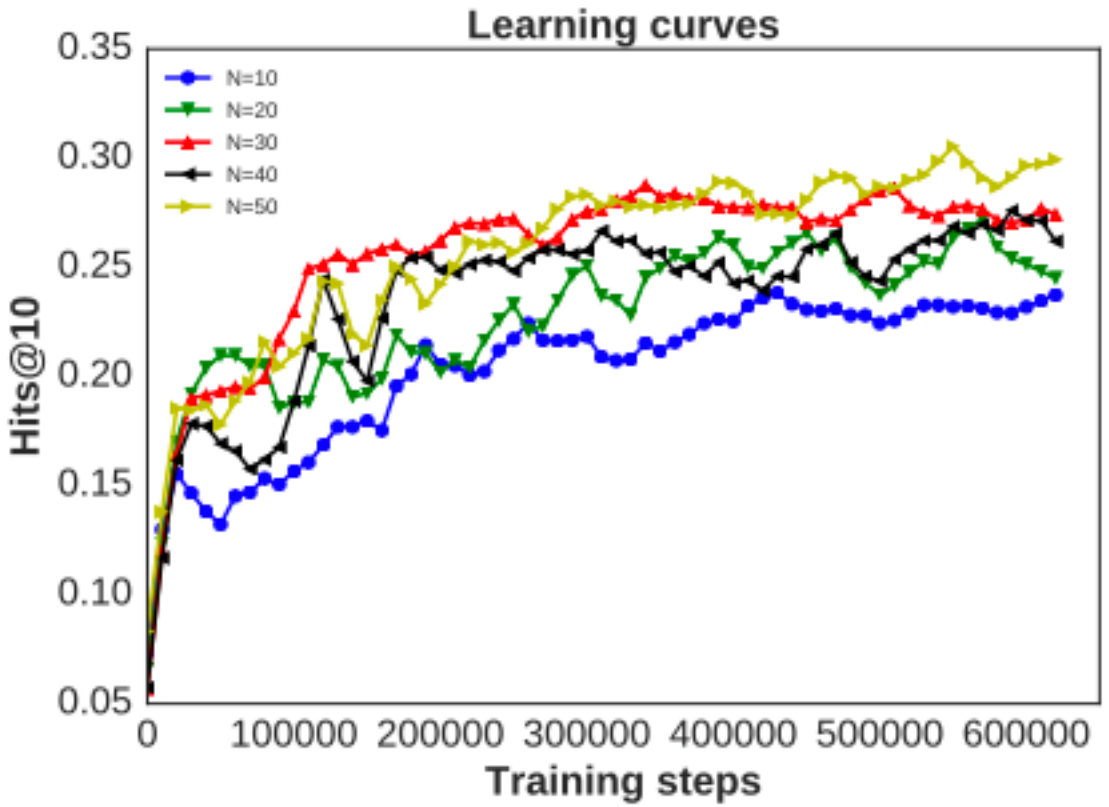


Figure 4: The learning curves on NELL-one. Every run uses different number of neighbors. The y-axis is Hits@10 calculated on all the validation relations.

Configuration	Hits@10
Full Model with ComplEx	**.308/.313**
w/o Matching Processor	.266/.269
w/o Neighbor Encoder	.248/.296
w/o Scaling Factor	.229/.219

Table 4: Ablation on different components.

Performance on Different Relations

Relations	# Candidates	MRR		Hits@10	
		GMatching	ComplEx	GMatching	ComplEx
sportsGameSport	123	0.424	**0.466**(0.139*)	**1.000**	0.479(0.200*)
athleteInjuredHisBodypart	299	0.025	0.026(**0.330***)	0.015	0.059(**0.444***)
animalSuchAsInvertebrate	786	0.447	0.333(**0.555***)	0.626	0.587(**0.783***)
automobilemakerDealersInCountry	1084	**0.453**	0.245(0.396*)	**0.821**	0.453(0.500*)
sportSchoolIncountry	2100	**0.534**	0.324(0.294*)	**0.745**	0.529(0.571*)
politicianEndorsesPolitician	2160	0.093	0.026(**0.194***)	0.226	0.047(**0.357***)
agriculturalProductFromCountry	2222	**0.120**	0.029(0.042*)	**0.288**	0.058(0.086*)
producedBy	3174	0.085	0.040(**0.165***)	0.179	0.075(**0.241***)
automobilemakerDealersInCity	5716	0.026	0.024(**0.041***)	0.040	0.051(**0.174***)
teamCoach	10569	0.017	0.065(**0.376***)	0.024	0.079(**0.547***)
geopoliticalLocationOfPerson	11618	0.028	0.016(**0.284***)	0.035	0.035(**0.447***)

Table 3: Results decomposed over different relations. "*" denotes the results with standard training settings and "# Candidates" denotes the size of candidate entity set.

END

THANK YOU