Fast Entity Linking via Graph Embeddings

Alberto Parravicini
Rhicheek Patra
Davide Bartolini
Marco Santambrogio

2019-06-30, GRADES-NDA

POLITECNICO MILANO 1863

POLITECNICO MILANO 1863 NECST laboratory

ORACLE®

Entity Linking

Entity Linking (EL): connecting words of interest to unique identities (e.g. Wikipedia Page)

"The Indiana Pacers and Miami Heat [...] meet at Miami's American Airlines Arena"

en.wikipedia.org/wiki/Miami_Heat

.../wiki/American_Airlines_Arena

en.wikipedia.org/wiki/Indiana_Pacers

en.wikipedia.org/wiki/Miami

Use Cases

Component of applications that require high-level representations of text:

1. Search Engines, for semantic search
2. Recommender Systems, to retrieve documents similar to each other
3. Chat bots, to understand intents and entities

The EL Pipeline (1/2)

An EL system requires 2 steps:

1. Named Entity Recognition (NER): spot **mentions** (a.k.a. Named Entities)
   - High-accuracy in the state-of-the-art[1]

"Trump will answer Clinton's claims about the Wall"

Text:	**Trump**,	**Clinton**,	**Wall**,
Type:	*PER*,	*PER*,	*LOC*,
Offset:	*0*	*3*	*7*

[1] Huang, Zhiheng, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging."

The EL Pipeline (2/2)

An EL system requires 2 steps:

2. Entity Linking: connect mentions to entities

"Trump will answer Clinton's claims about the Wall"

	Trump	Clinton	Wall
Text:	**Trump**,	**Clinton**,	**Wall**,
Type:	*PER*	*PER*	*LOC*
	.../wiki/Donald_Trump	.../wiki/Hillary_Clinton	?

The EL Pipeline (2/2)

An EL system requires 2 steps:

2. Entity Linking: connect mentions to entities

"Trump will answer Clinton's claims about the Wall"

Text:	**Trump**,	**Clinton**,	**Wall**,
Type:	*PER*	*PER*	*LOC*

.../wiki/Donald_Trump

.../wiki/Hillary_Clinton

?

.../wiki/Defensive_wall
.../wiki/Berlin_wall
.../wiki/Mexico–United_States_barrier

The EL Pipeline (2/2)

An EL system requires 2 steps:

2. Entity Linking: connect mentions to entities

"Trump will answer Clinton's claims about the Wall"

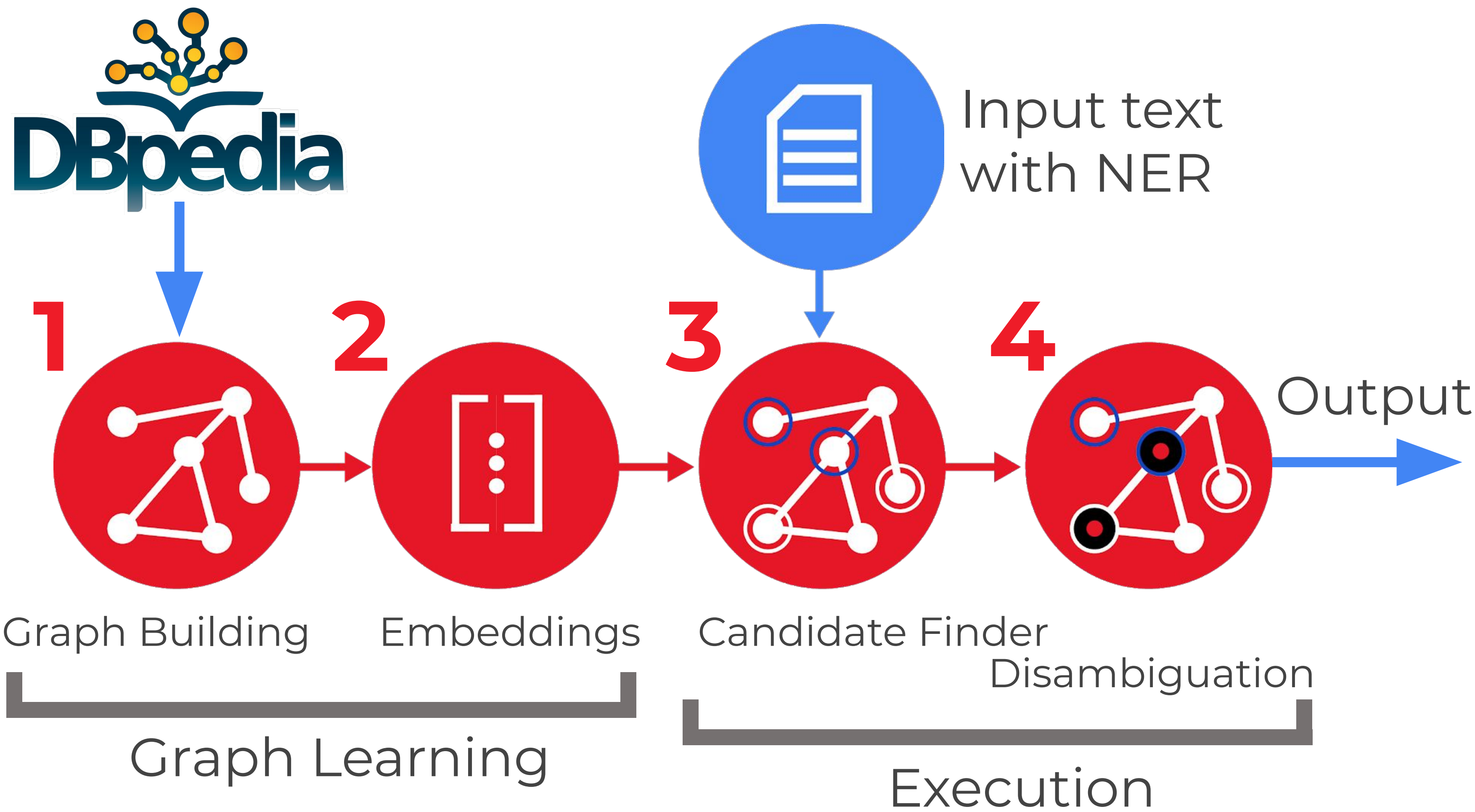
Text:	**Trump**,	**Clinton**,	**Wall**,
Type:	*PER*	*PER*	*LOC*
	.../wiki/Donald_Trump	.../wiki/Hillary_Clinton	!

.../wiki/Mexico–United_States_barrier

Our contributions

- Novel unsupervised framework for EL
  - No dependency on NLP
- First EL algorithm to use graph embeddings
  - Accuracy similar to supervised SoA techniques
- Highly scalable and real-time execution time
  - < 1 sec to process text with 30+ mentions

Our EL Pipeline

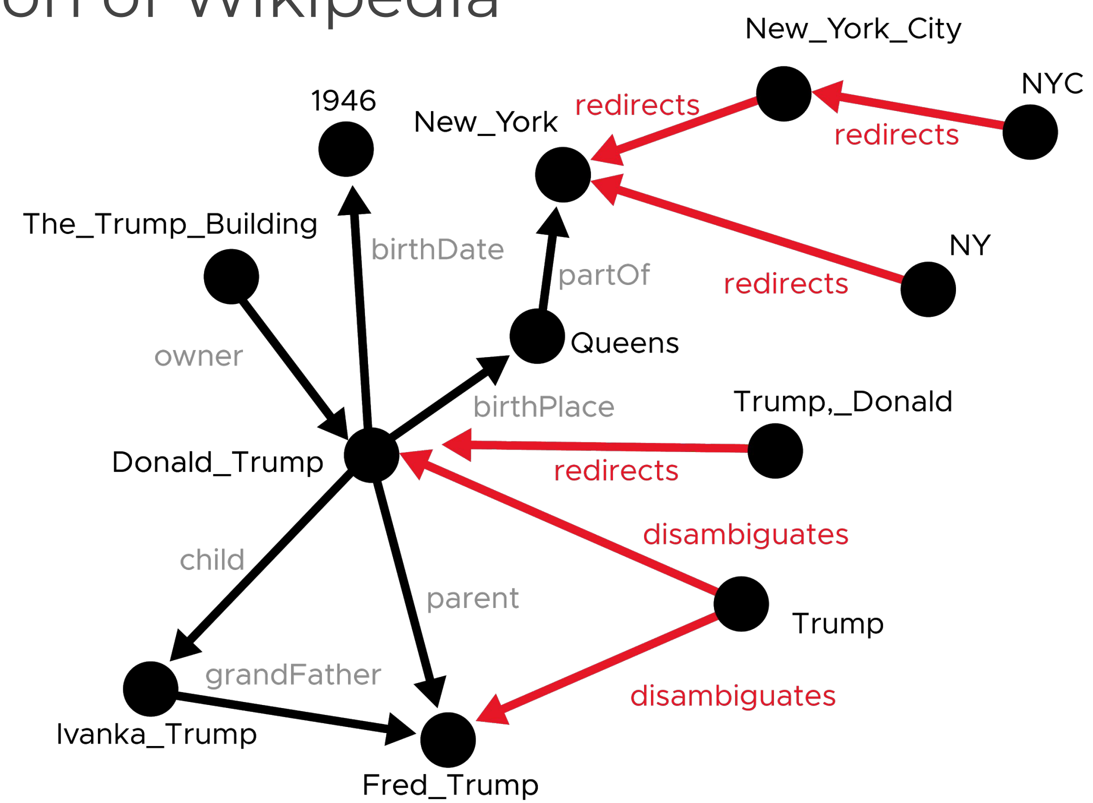


Step 1/4

Graph Creation

We obtain a large graph from DBpedia

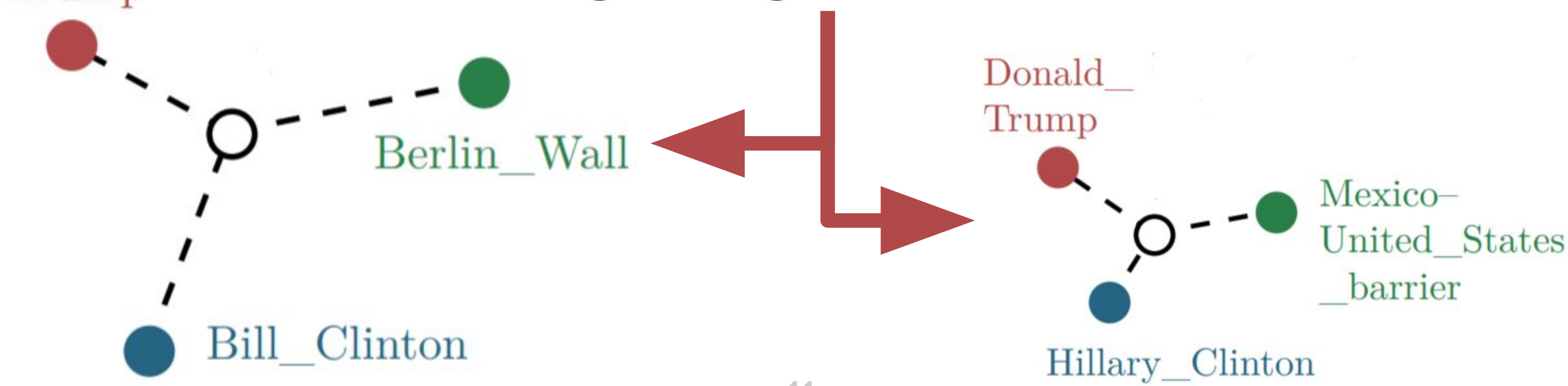
- All the information of Wikipedia
- Stored as triples
- 12M entities
- 170M links



Embeddings Creation (1/2)

- Graph embeddings encode vertices as vectors
  - “Similar” vertices have “similar” embeddings
- Idea: entities with the same context should have low embedding distance

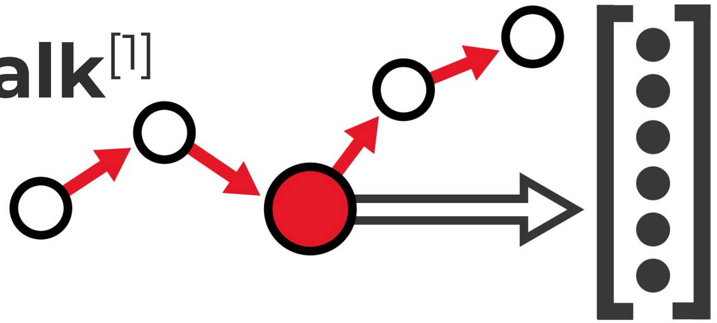
News: Trump will answer Clinton’s claims regarding the Wall.



Step 2/4

Embeddings Creation (2/2)

- In our work, we use **DeepWalk**[1]



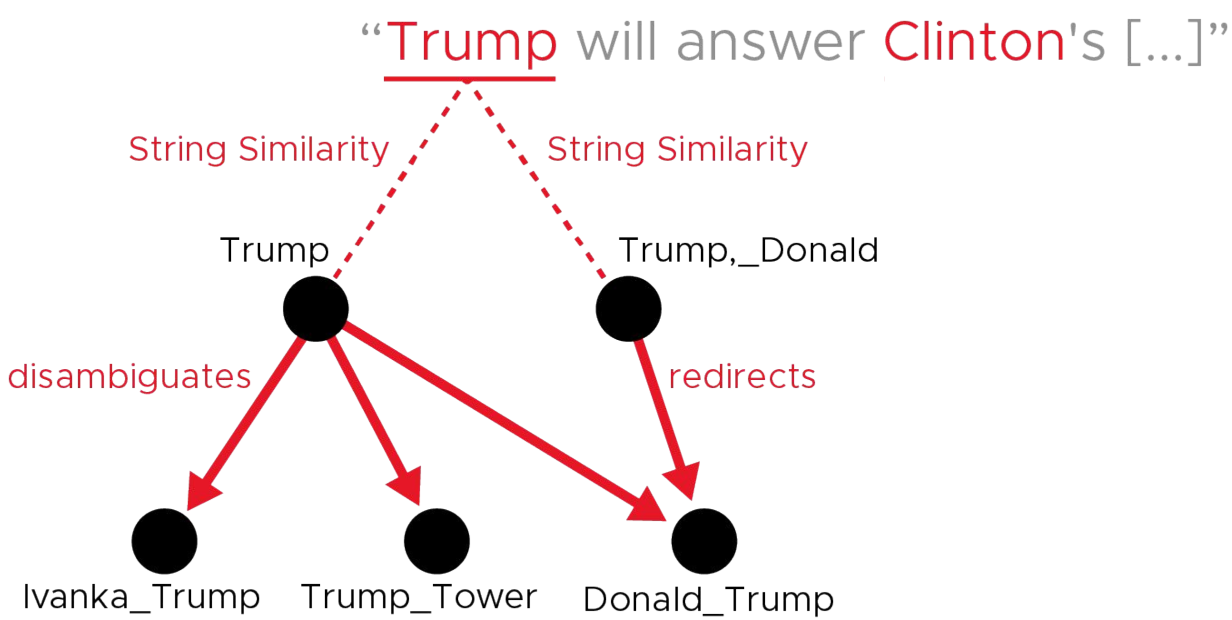
- Like **word2vec**[2], it leverages random walks (i.e. vertex sequences) to create embeddings
  - Embedding size 170, walk length 8
- DeepWalk uses only the graph topology
  - Simple baseline, we can use better algorithms and leverage graph features

[1] Bryan Perozzi et al.2014. Deepwalk
[2] Tomas Mikolov et al. 2013. Distributed representations of words and phrases and their compositionality

Candidate Finder

Idea: for each mention, select a few candidate vertices with index-based **string similarity**

- Solve ambiguity following **redirect** and **disambiguation** links

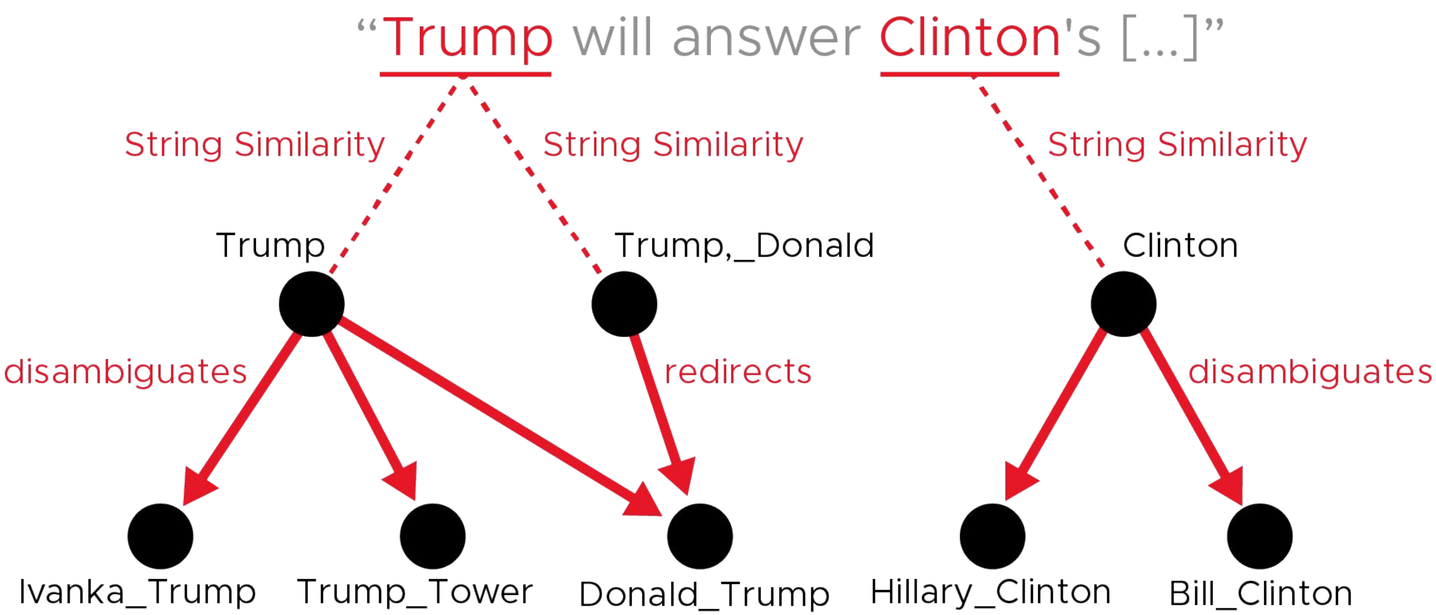


Step 3/4

Candidate Finder

Idea: for each mention, select a few candidate vertices with index-based **string similarity**

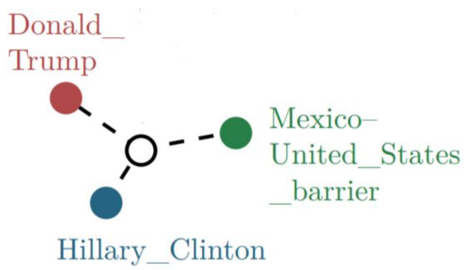
- Solve ambiguity following **redirect** and **disambiguation** links



Step 4/4

Disambiguation (1/3)

- We want to pick the “best” candidate for each mention
  - In a “good” solution, candidates are related to each other (e.g. *Donald Trump, Hillary Clinton*)
- Observation: a good tuple of candidates has embeddings close to each other



- Evaluating all combinations is infeasible
  - 10 mentions with 100 candidates → 100^{10}

Disambiguation (2/3)

- We use an heuristic state-space search algorithm to maximize:

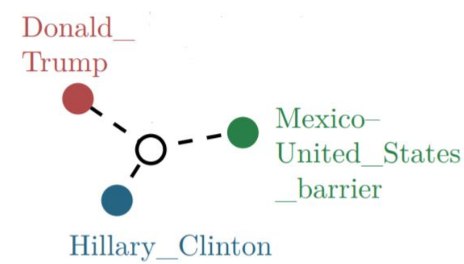
$$Best\ Tuple = \underset{T}{\operatorname{argmax}} \left(\sum_{t_i \in T} \boxed{Local(t_i)} + \boxed{Global(T)} \right)$$

Sum of string similarities

Sum of embedding cosine similarities w.r.t. tuple mean

$$\bar{\mathbf{e}}(T) = \frac{1}{|T|} \sum_{t_i \in T} \mathbf{e}(t_i)$$

$$Global(T) = \sum_{t_i \in T} \frac{\langle \mathbf{e}(t_i), \bar{\mathbf{e}}(T) \rangle}{\|\mathbf{e}(t_i)\|_2 \|\bar{\mathbf{e}}(T)\|_2}$$



Disambiguation (3/3)

- Iterative state-space heuristic

Function optimizer(*candidates*):
  T = find_initial_state(*candidates*)

  while *stop condition not met* **do**
    // Create *num_children* new tuples by modifying random elements of T.
    T = new_tuples(T, *num_children*)
    T = optimize_tuple(T)
    s = compute_score(T)
    if *curr_score* $\geq$ *best_score* **then**
      best_tuple = T
      best_score = s
    end
  end
  return best_tuple, best_score

Greedy iterative procedure

$$\underset{T}{\operatorname{argmax}} \left(\sum_{t_i \in T} Local(t_i) + Global(T) \right)$$

Results: accuracy

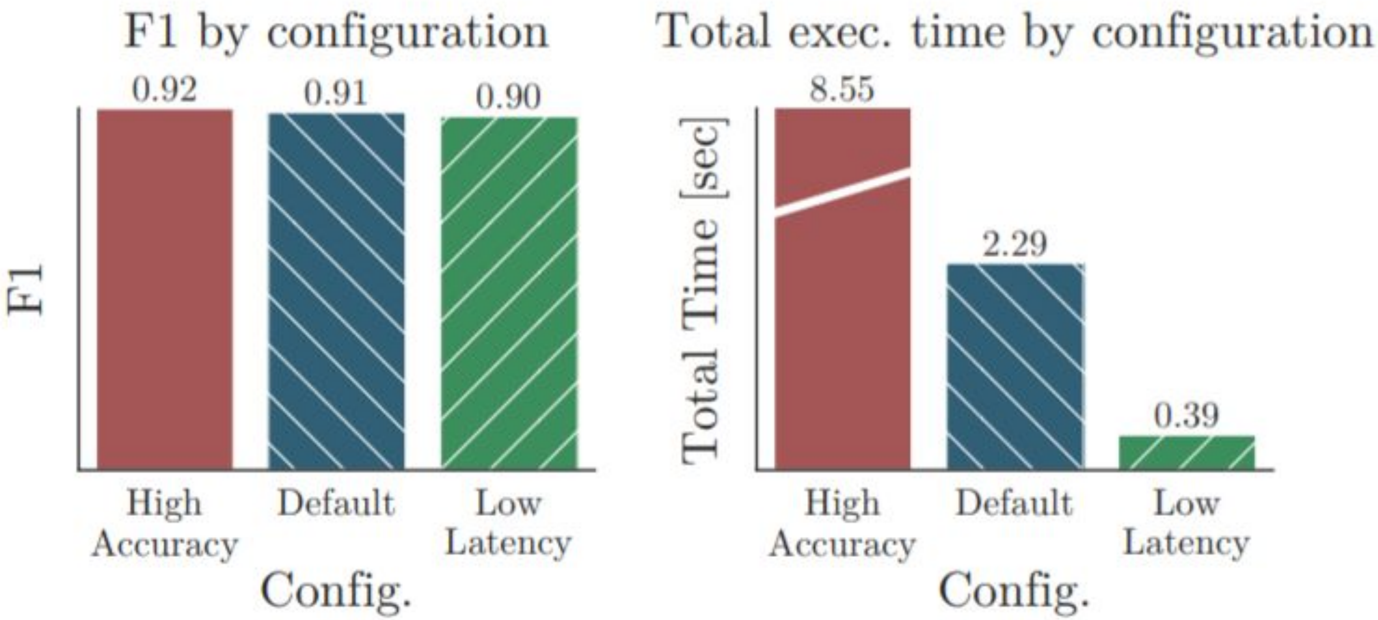
- We compared against 6 SoA EL algorithms, on 5 datasets
- Our Micro-averaged **F1** score is comparable with SoA supervised algorithms

Data Set	Ours	DoSeR	WK	AIDA	WAT	BB	SL
ACE2004	0.84	**0.90**	0.83	0.81	0.80	0.56	0.71
AQUAINT	**0.86**	0.84	**0.86**	0.53	0.77	0.65	0.71
MSNBC	**0.92**	0.91	0.85	0.78	0.78	0.60	0.51
N3-Reuters	0.82	**0.85**	0.70	0.60	0.64	0.53	0.58
N3-RSS-500	0.72	**0.75**	0.73	0.71	0.68	0.63	0.62

WK is Wikifier, **BB** is Babelfy, **SL** is Spotlight

Results: exec. time

- Different settings enable real-time EL, with minimal loss in accuracy
  - E.g. number of iterations, early stop



Conclusion & Future Work

- In short:
  - First EL algorithm to use graph embeddings
  - Accuracy similar to supervised SoA techniques
  - Real-time execution time
- Future works:
  - Better graph embeddings algorithms (use vertex/edge features)
  - Improve disambiguation algorithm for even faster exec. time

Thank you!

Fast Entity Linking via Graph Embeddings

- Novel unsupervised framework for EL
- First EL algorithm to use graph embeddings
  - Accuracy similar to supervised SoA techniques
- Real-time execution time

Alberto Parravicini, alberto.parravicini@polimi.it

Rhicheek Patra

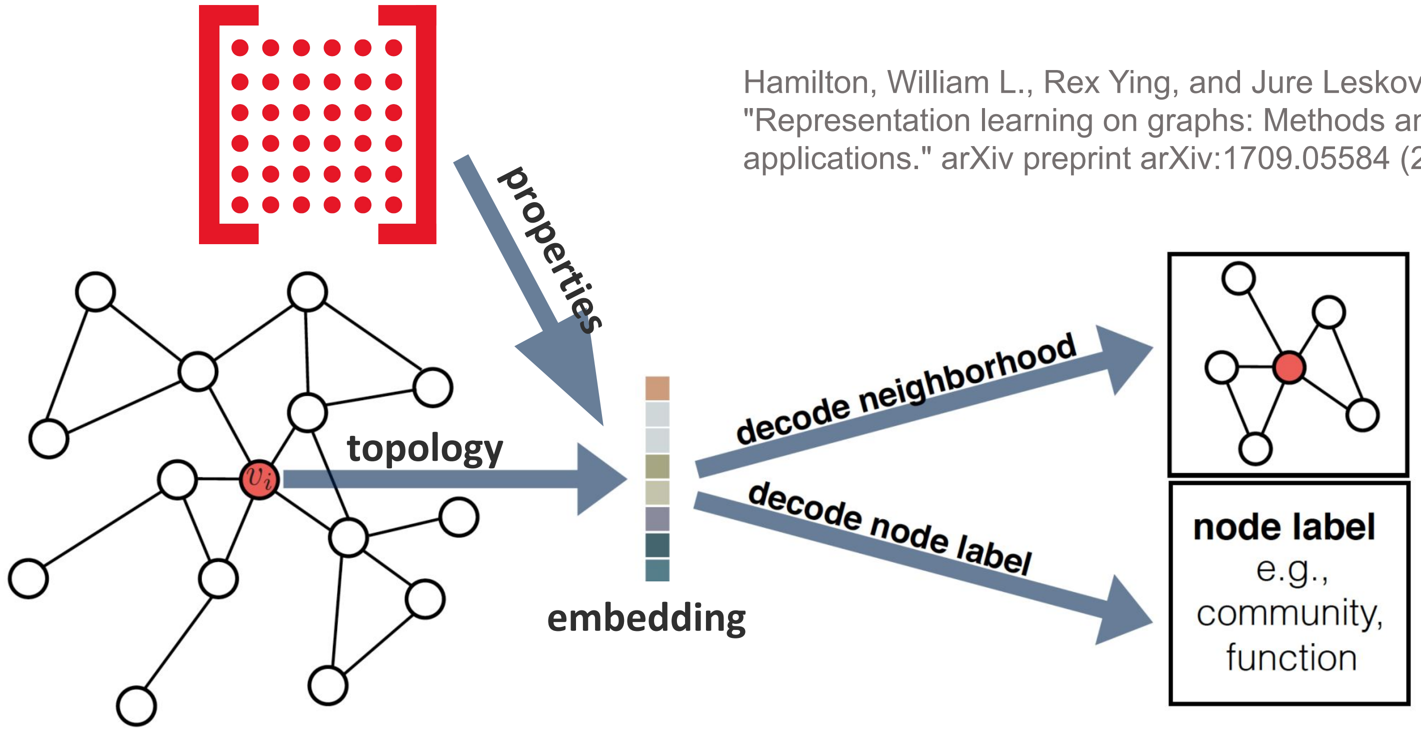
Davide Bartolini

Marco Santambrogio

2019-06-30, GRADES-NDA

Embeddings

- Turn topology and properties of each vertex into a vector
- “Similar” vertices have “similar” embeddings



Hamilton, William L., Rex Ying, and Jure Leskovec. "Representation learning on graphs: Methods and applications." arXiv preprint arXiv:1709.05584 (2017).

Graph Creation

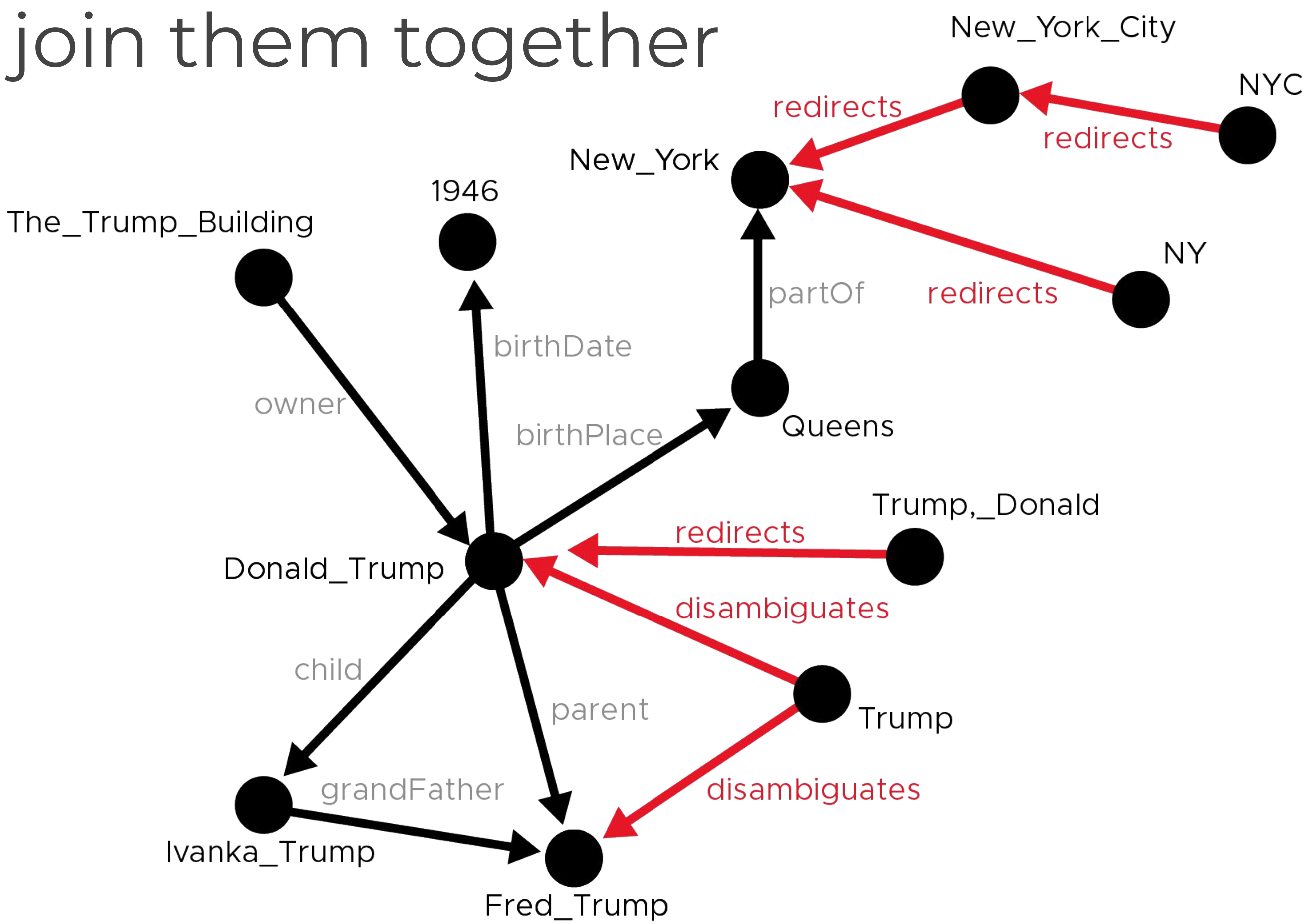
We obtain a large graph from DBpedia

- All the information of Wikipedia, stored as triples
- 12M entities, 170M links

Step 1/4

Graph Creation

... and join them together



Candidates Finder

Idea: for each mention, select a small number of candidate vertices with **string similarity**

- We use a simple index-based string search
- Fuzzy matching with 2-grams and 3-grams
- This provide a simple baseline (60-70% accuracy)

Results: exec. time of single steps

- Execution time is well divided between Candidate Finder and Disambiguation