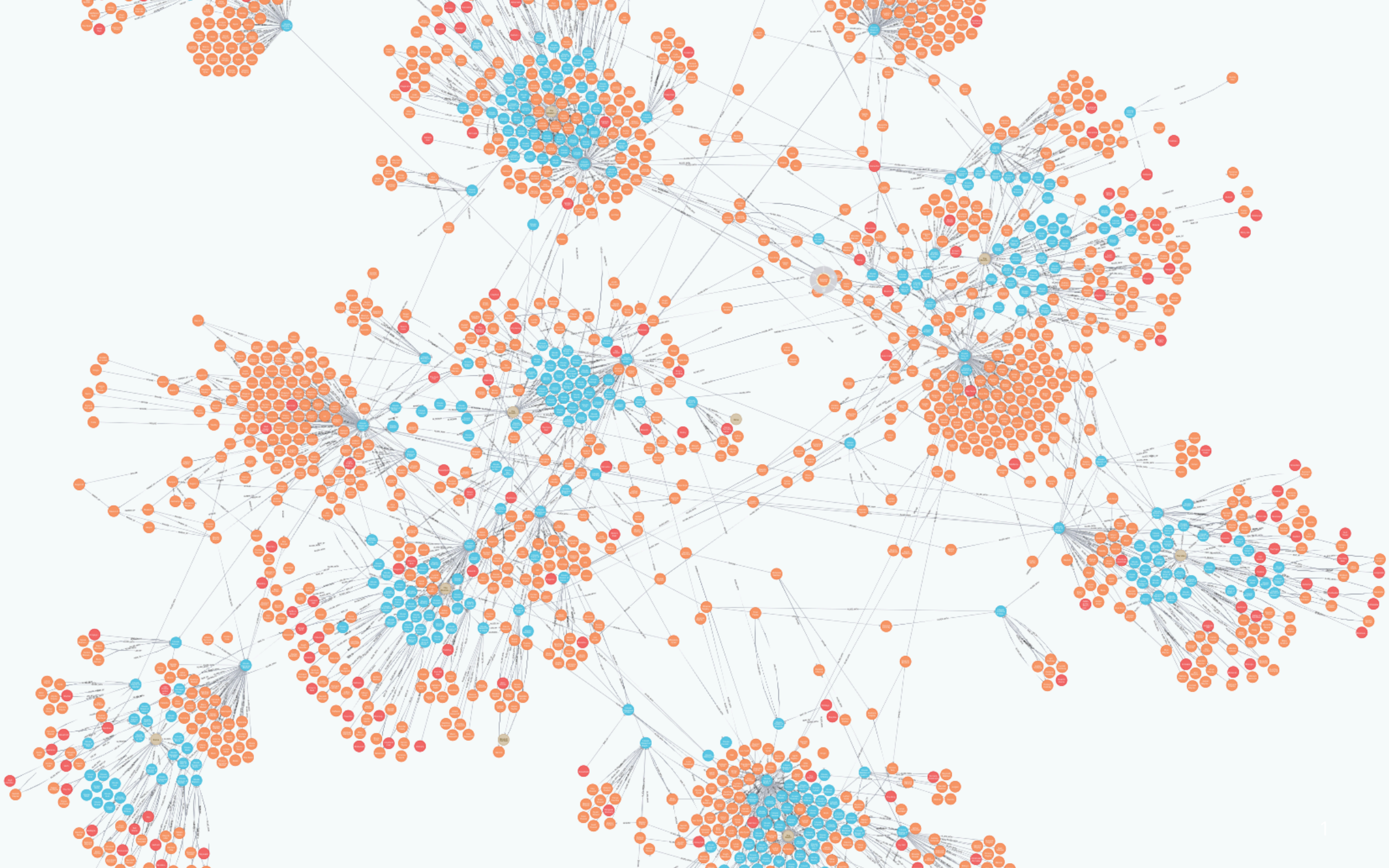


*(2122) House(444) Person(1442) Seat(224) Region(12)

*(3233) ALLIED_WITH(1567) PARENT_OF(61) SPOUSE(199) HEIR_TO(60) FOUNDED_BY(44) LED_BY(180) SEAT_OF(244) BRANCH_OF(43) SWORN_TO(397) IN_REGION(438)

github.com/neo4j-examples/game-of-thrones

PARENT_OF ?

PARENT_OF ?

IN_REGION?

Luca Costabello
Sumit Pai
Chan Le Van
Rory McGrath
Nicholas McCarthy

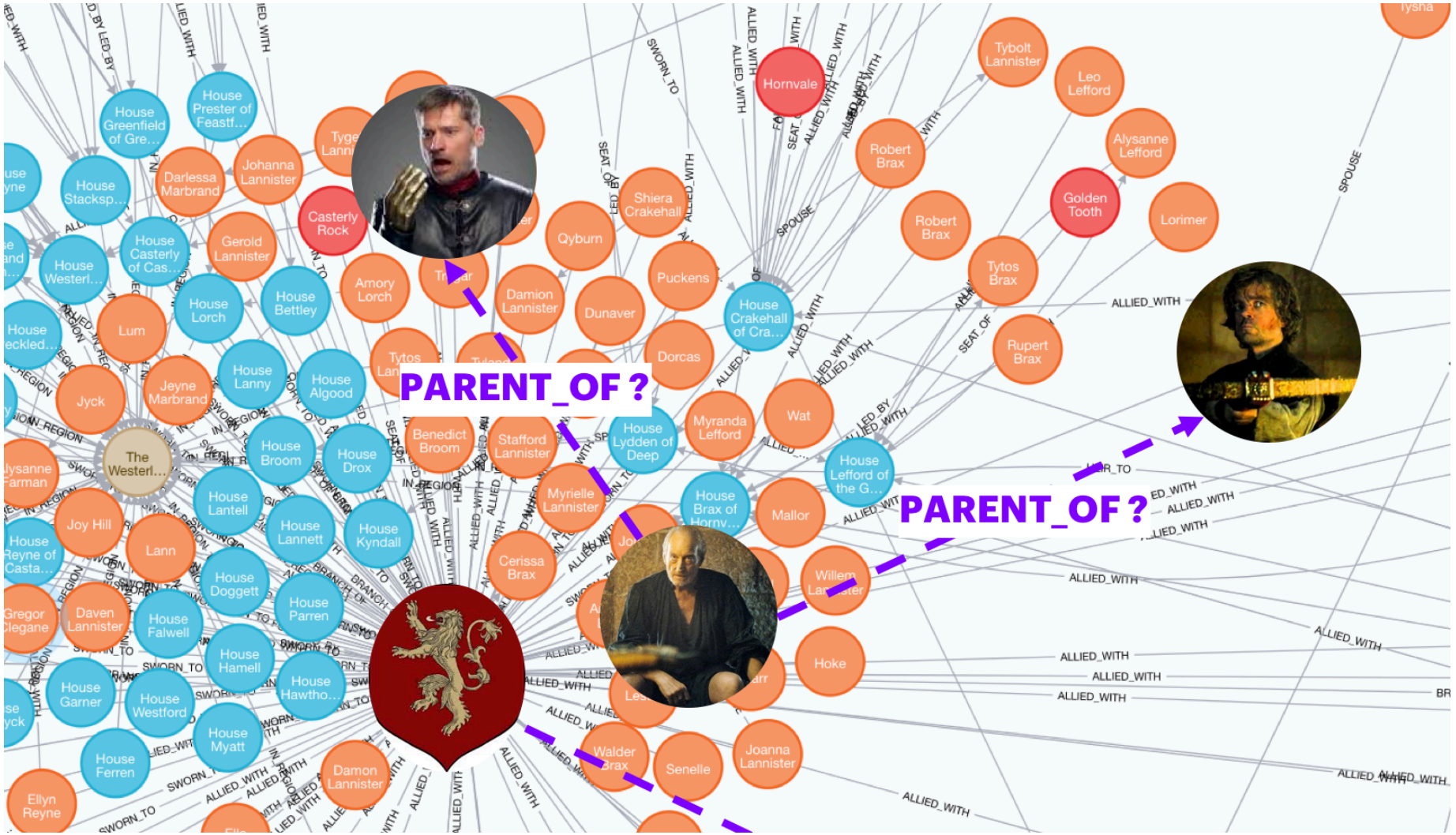
AmpliGraph

A Library for Graph Representation Learning

ampligraph.org

Open source Python library that predicts links between concepts in a knowledge graph.

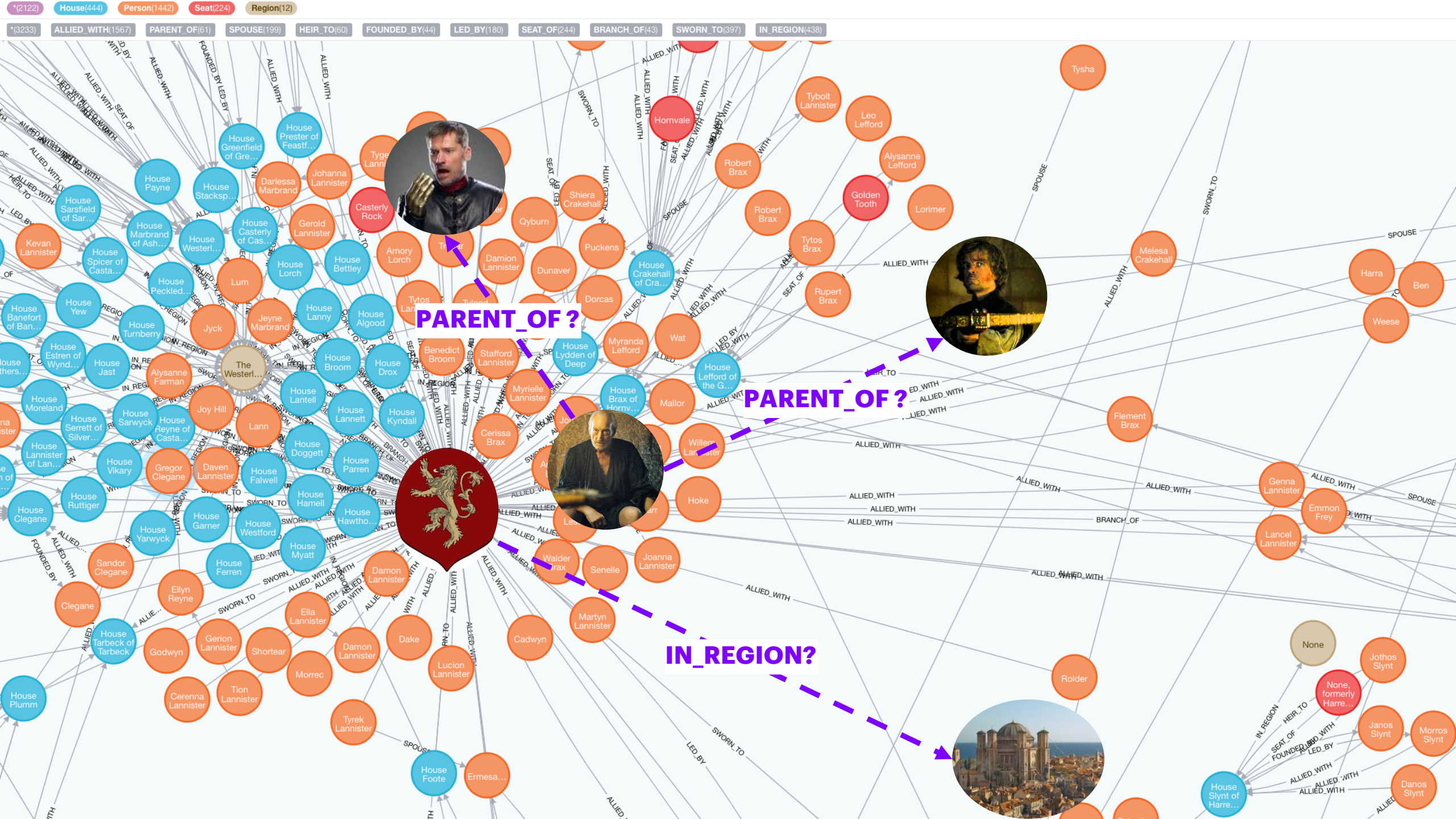
AmpliGraph is a suite of neural machine learning models for relational Learning, a branch of machine learning that deals with supervised learning on knowledge graphs.

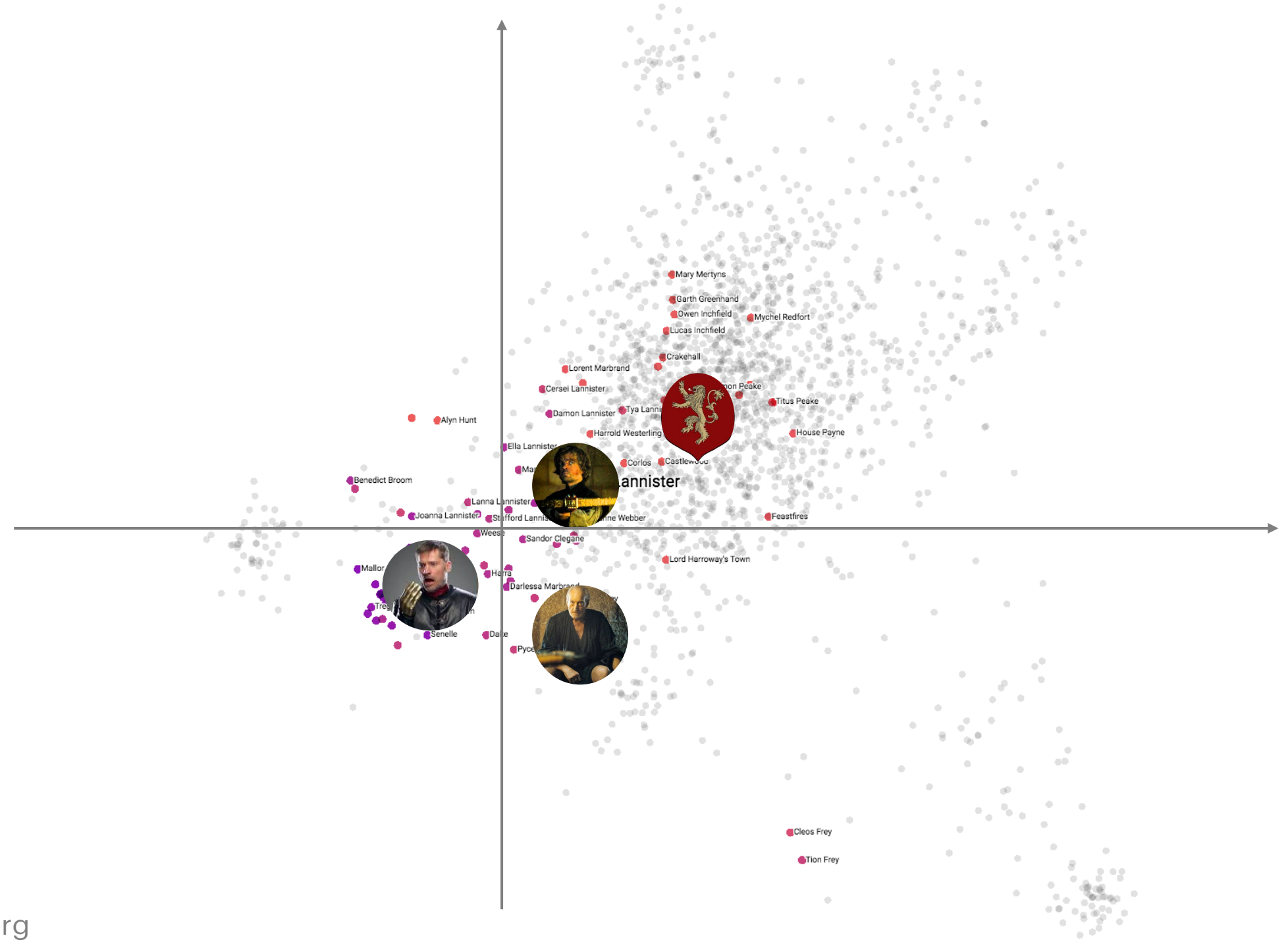


Use AmpliGraph if you need to:

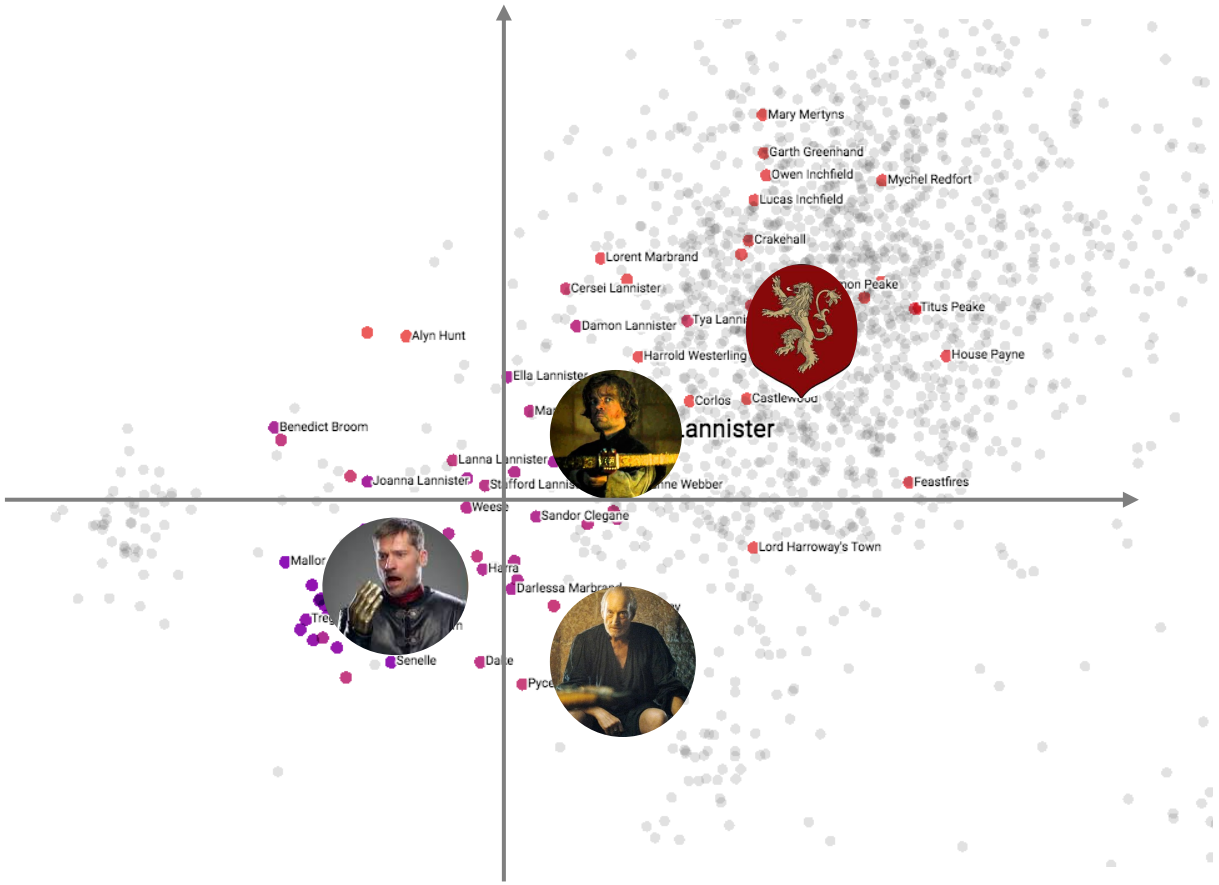
- Discover new knowledge from an existing graph
- Complete a large and incomplete knowledge graph
- Cluster similar concepts according to their relations
- Identify near-duplicate concepts
- Use graph embeddings in downstream tasks

AmpliGraph's machine learning models generate **knowledge graph embeddings**, vector representations of concepts in a metric space:

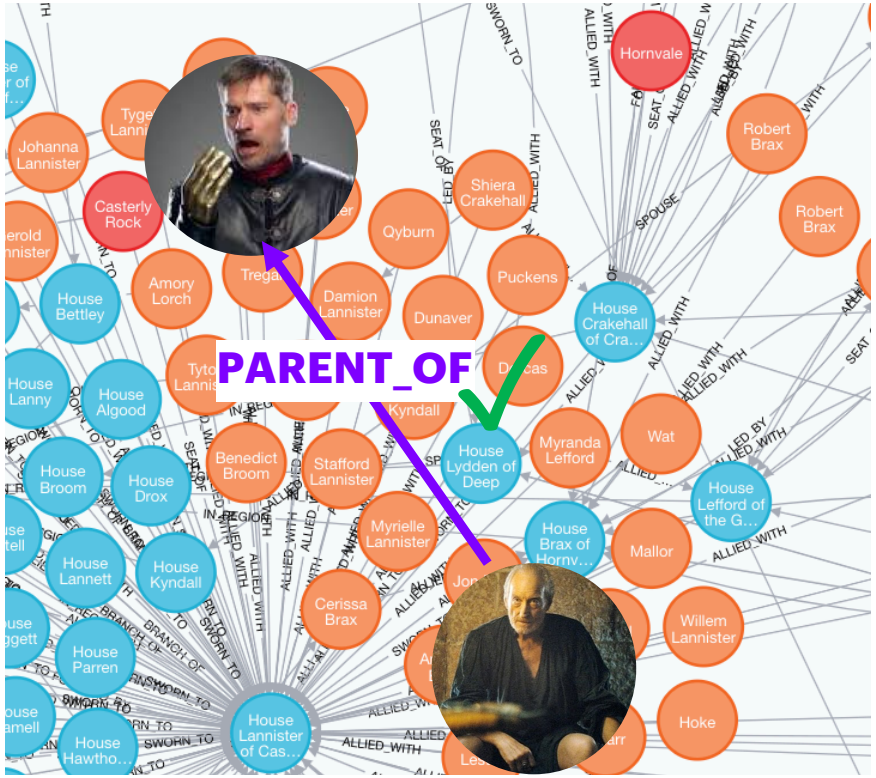

*(2122)
House(444)
Person(1442)
Seat(224)
Region(12)
*(3233)
ALLIED_WITH(1567)
PARENT_OF(61)
SPOUSE(199)
HEIR_TO(60)
FOUNDED_BY(44)
LED_BY(180)
SEAT_OF(244)
BRANCH_OF(43)
SWORN_TO(397)
IN_REGION(438)
PARENT_OF ?
PARENT_OF ?
IN_REGION?
The Westerl…
House Lannister of Lan…
House Clegane
House Plumm
House Tarbeck of Tarbeck
House Ruttiger
House Vikary
House Serrett of Silver…
House Estren of Wynd…
House Banefort of Ban…
House Yew
House Jast
House Moreland
House Sarwyck
House Yarwyck
House Reyne of Casta…
House Spicer of Casta…
House Sarsfield of Sar…
House Payne
House Marbrand of Ash…
House Peckled…
House Turnberry
House Westerl…
House Stacksp…
House Greenfield of Gre…
House Prester of Feastf…
House Casterly of Cas…
House Lorch
House Bettley
House Lanny
House Algood
House Broom
House Drox
House Lantell
House Lannett
House Kyndall
House Doggett
House Parren
House Falwell
House Hamell
House Hawtho…
House Myatt
House Westford
House Garner
House Ferren
House Foote
Ermesa…
Kevan Lannister
Gregor Clegane
Daven Lannister
Sandor Clegane
Clegane
Ellyn Reyne
Godwyn
Gerion Lannister
Shortear
Morrec
Cerenna Lannister
Tion Lannister
Tyrek Lannister
Damon Lannister
Dake
Lucion Lannister
Ella Lannister
Cadwyn
Martyn Lannister
Senelle
Joanna Lannister
Walder Frax
Damion Lannister
Dunaver
Puckens
Dorcas
Amory Lorch
Qyburn
Shiera Crakehall
Casterly Rock
Johanna Lannister
Darlessa Marbrand
Gerold Lannister
Jeyne Marbrand
Jyck
Lum
Alysanne Farman
Joy Hill
Lann
Benedict Broom
Stafford Lannister
Myrielle Lannister
Cerissa Brax
House Lydden of Deep
House Crakehall of Cra…
Myranda Lefford
Wat
House Brax of Hornv…
Mallor
House Lefford of the G…
Willem Lannister
Hoke
Rupert Brax
Robert Brax
Tytos Brax
Golden Tooth
Lorimer
Alysanne Lefford
Leo Lefford
Tybolt Lannister
Hornvale
Tysha
Melesa Crakehall
Harra
Ben
Weese
Flement Brax
Genna Lannister
Emmon Frey
Lancel Lannister
Rolder
None
None, formerly Harre…
Jothos Slynt
Janos Slynt
Morros Slynt
Danos Slynt
House Slynt of Harre…



It then combines embeddings with model-specific scoring functions to predict unseen and novel links:



f( ,PARENT_OF, )= 0.85



Anatomy of a KG Embedding Model

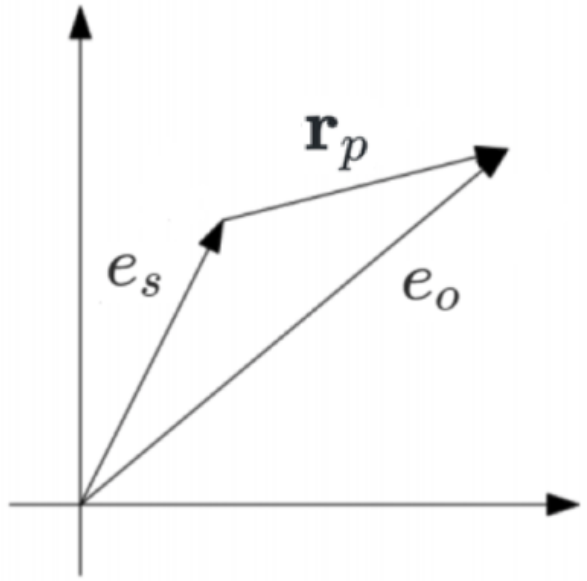
- Knowledge Graph (KG) $\mathcal{G}$
- Scoring function for a triple $f(t)$
- Loss function $\mathcal{L}$
- Optimization algorithm
- Negatives generation strategy

Scoring function f

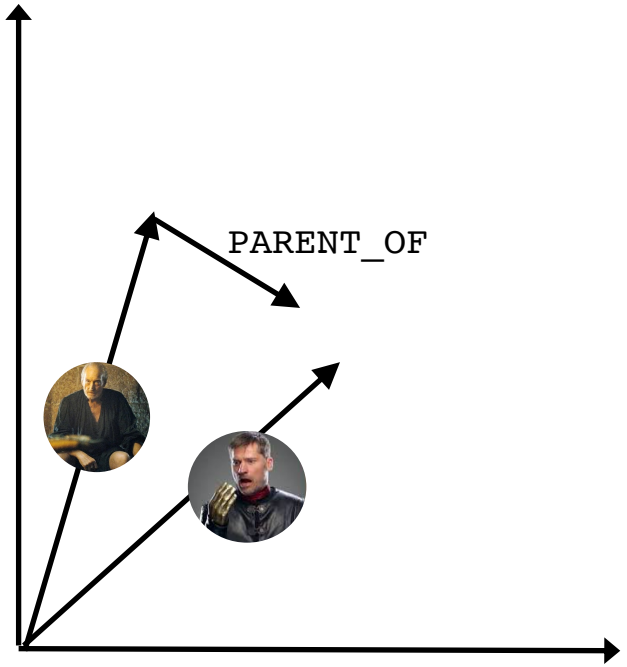
f assigns a score to a triple (s, p, o):

- **TransE**: Translating Embeddings [Bordes13]

$$f_{TransE} = -||(\mathbf{e}_s + \mathbf{r}_p) - \mathbf{e}_o||_n$$



( ,PARENT_OF, )



Loss function $\mathcal{L}$

- Pairwise margin-based:

$$\mathcal{L}(\Theta) = \sum_{t^+ \in \mathcal{G}} \sum_{t^- \in \mathcal{C}} max(0, [\gamma + f(t^-; \Theta) - f(t^+; \Theta)])$$

- Absolute Margin
- Negative log-likelihood
- Self-adversarial
- ...

Optimization Procedure

- Estimate the optimal parameters Θ with (same variant of) SGD:

$$\min_{\Theta} \mathcal{L}(\Theta)$$

- On-the-fly negatives generation with the negatives strategy

Negatives generation strategies

- **Uniform sampling**: Generate and uniformly sample corruptions $\mathcal{C}$:

$$\mathcal{C}((s,p,o)) = \{(\hat{s},p,o)|\hat{s} \in \mathcal{E}\} \cup \{(s,p,\hat{o})|\hat{o} \in \mathcal{E}\}$$

- Based on **generative adversarial network** [Cai17]
- **Self-adversarial negative sampling** [Sun19]

Negatives Generation with Uniform Sampling

$$\mathcal{E} = \{Tyrion, Jamie, Tywin, House\_Lannister\}$$
$$\mathcal{R} = \{PARENT\_OF\}$$

$t_i \in \mathcal{G} =$ `(Tywin PARENT_OF Tyrion)`

$\mathcal{C}_{t_i} =$

```
Tywin                PARENT_OF   House_Lannister
Tywin                PARENT_OF   Jamie
Jamie                PARENT_OF   Tyrion
House_Lannister      PARENT_OF   Tyrion
```

Evaluation Metrics

Learning to Rank metrics

How well are positive triples ranked against their corruptions?

$$Hits@N = \frac{1}{|Q|} \sum_{i=1}^{|Q|} 1 \text{ if } rank_{(s,p,o)_i} \leq N$$

$$MR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} rank_{(s,p,o)_i}$$ **[Mean Rank]**

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_{(s,p,o)_i}}$$ **[Mean Reciprocal Rank]**

Example

s	p	o	score	rank	
Tywin	PARENT_OF	House_lannister	0.789	1	
Tywin	PARENT_OF	Tyrion	0.753	2	*
Tywin	PARENT_OF	House_Lannister	0.695	3	
Sansa	PARENT_OF	Tyrion	0.695	3	
Cersei	PARENT_OF	Tyrion	0.234	5	

s	p	o	score	rank	
Tywin	PARENT_OF	Jamie	0.852	1	*
Sansa	PARENT_OF	Jamie	0.345	2	
Cersei	PARENT_OF	Jamie	0.293	3	
Tywin	PARENT_OF	Sansa	0.201	4	
Tywin	PARENT_OF	Cersei	0.156	5	

$$Hits@3 = \frac{1}{2}\sum_{i=1}^{2} 1 \text{ if } rank_{(s,p,o)_i} \leq 3 = 1$$

$$Hits@1 = \frac{1}{2}\sum_{i=1}^{2} 1 \text{ if } rank_{(s,p,o)_i} \leq 1 = 0.5$$

Import a dataset and split into train/test:

```
X = load_from_csv("datasets/", "GoT_graph.csv")
X_train, X_test = train_test_split_no_unseen(X, test_size=10)
```

Initialize a graph embedding model:

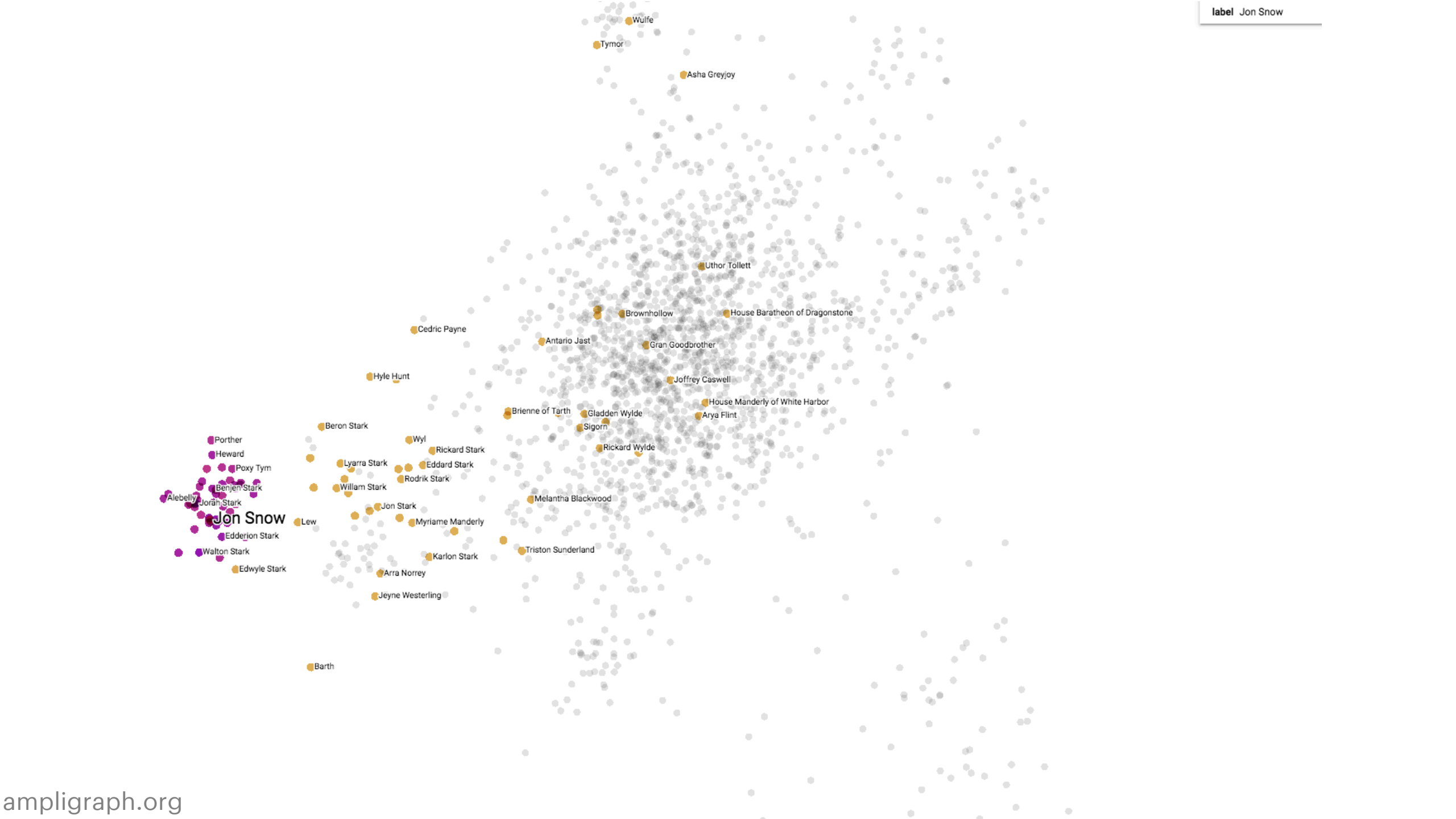
```
model = ComplEx(k=150, epochs=500, eta=5, loss='nll')
```

Generate the embeddings:

```
model.fit(X_train)
```

Visualize the embeddings:

```
create_tensorboard_projector_files(model, 'GoT_embeddings.tsv')
```


label Jon Snow
Wulfe
Tymor
Asha Greyjoy
Uthor Tollett
Brownhollow
House Baratheon of Dragonstone
Cedric Payne
Antario Jast
Gran Goodbrother
Hyle Hunt
Joffrey Caswell
House Manderly of White Harbor
Brienne of Tarth
Gladden Wylde
Arya Flint
Sigorn
Beron Stark
Wyl
Porther
Heward
Rickard Stark
Rickard Wylde
Lyarra Stark
Eddard Stark
Poxy Tym
Rodrik Stark
Benjen Stark
Willam Stark
Melantha Blackwood
Alebelly
Jorah Stark
Jon Stark
Jon Snow
Lew
Myriame Manderly
Edderion Stark
Triston Sunderland
Walton Stark
Karlon Stark
Edwyle Stark
Arra Norrey
Jeyne Westerling
Barth
ampligraph.org

Import a dataset and split into train/test:

```
X = load_from_csv("datasets/", "GoT_graph.csv")
X_train, X_test = train_test_split_no_unseen(X, test_size=10)
```

Initialize a graph embedding model:

```
model = ComplEx(k=150, epochs=500, eta=5, loss='nll')
```

Generate the embeddings:

```
model.fit(X_train)
```

Visualize the embeddings:

```
create_tensorboard_projector_files(model, 'GoT_embeddings.tsv')
```

Assess predicting power:

```
ranks = evaluate_performance(X_test, model=model)
mrr_score(ranks)
hits_at_n_score(ranks, n=10)
```

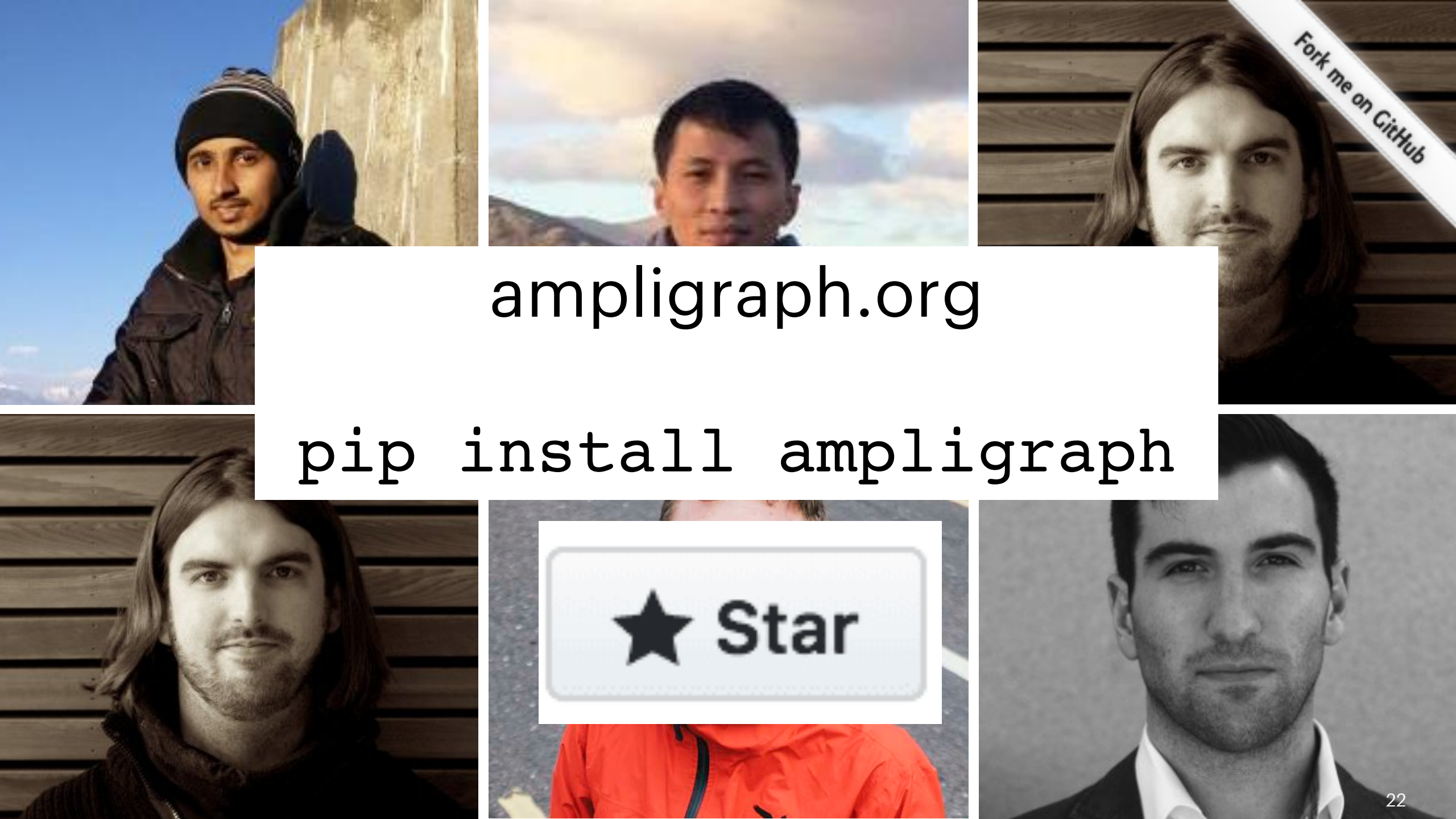
```
MRR: 0.393462
Hits@10: 0.495000
```

```
evaluate_performance(np.array([['Daenerys Targaryen', 'SPOUSE', 'Jon Snow']]), model=model)
```

```
1844/6350
```

```
evaluate_performance(np.array([["King's Landing", 'SEAT_OF', 'Brandon Stark']]), model=model)
```

```
385/6350
```



ampligraph.org

```
pip install ampligraph
```