Knowledge Graph Completion using Tensor Factorization Approaches

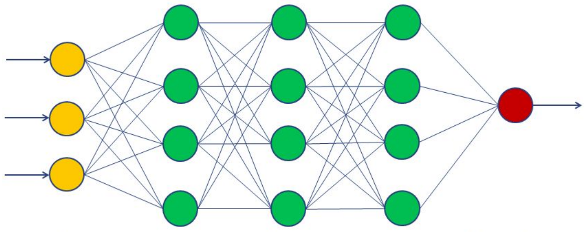
Bahare Fatemi

bfatemi@cs.ubc.ca

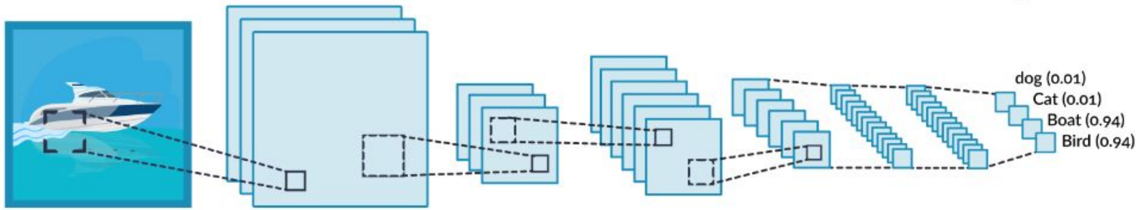
@BahareFatemi

Why structured data?

Vectors



Grids



Sequences

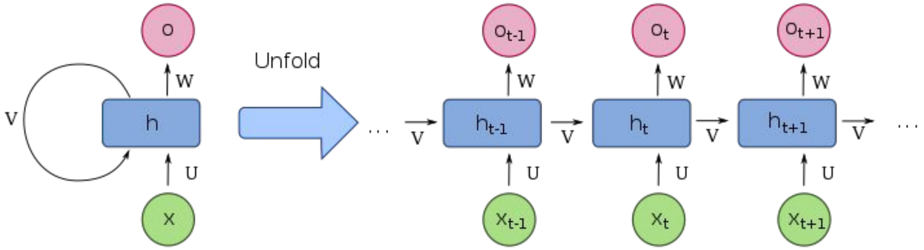
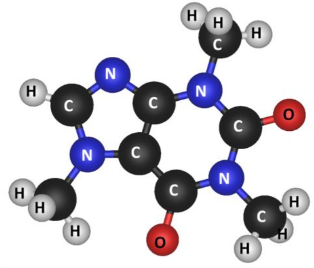
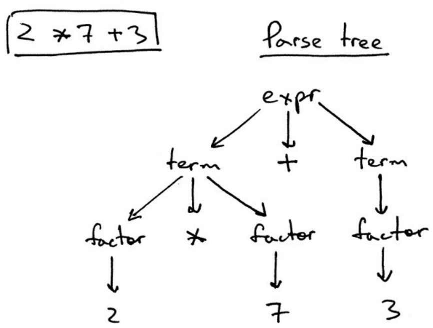
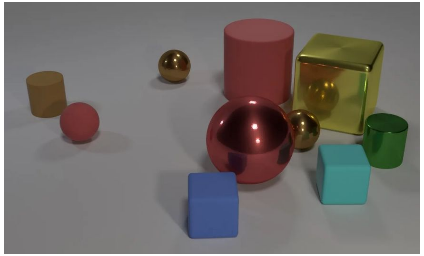


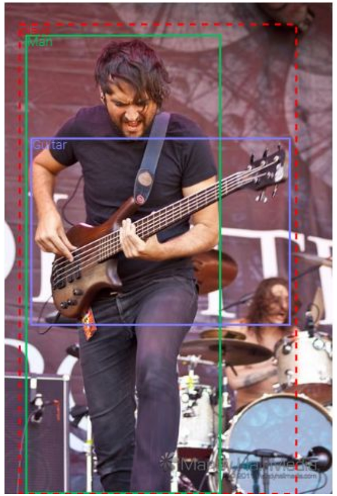
Image Credits https://www.datacamp.com/community/tutorials/neural-network-models-r, https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-tutorial-basic-advanced/, https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg

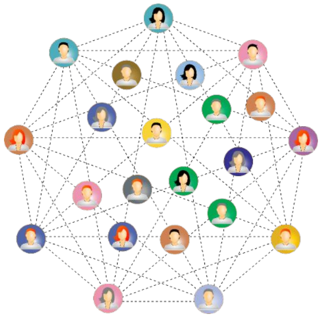
Many complex systems are structured











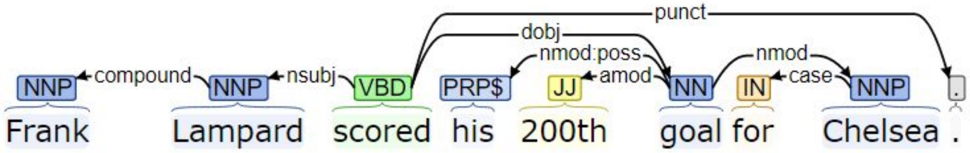


Image Credits: https://storage.googleapis.com/openimages, https://deepmind.com/blog/article/neural-approach-relational-reasoning, www.corenlp.run, https://courses.lumenlearning.com/introchem/chapter/molecules/

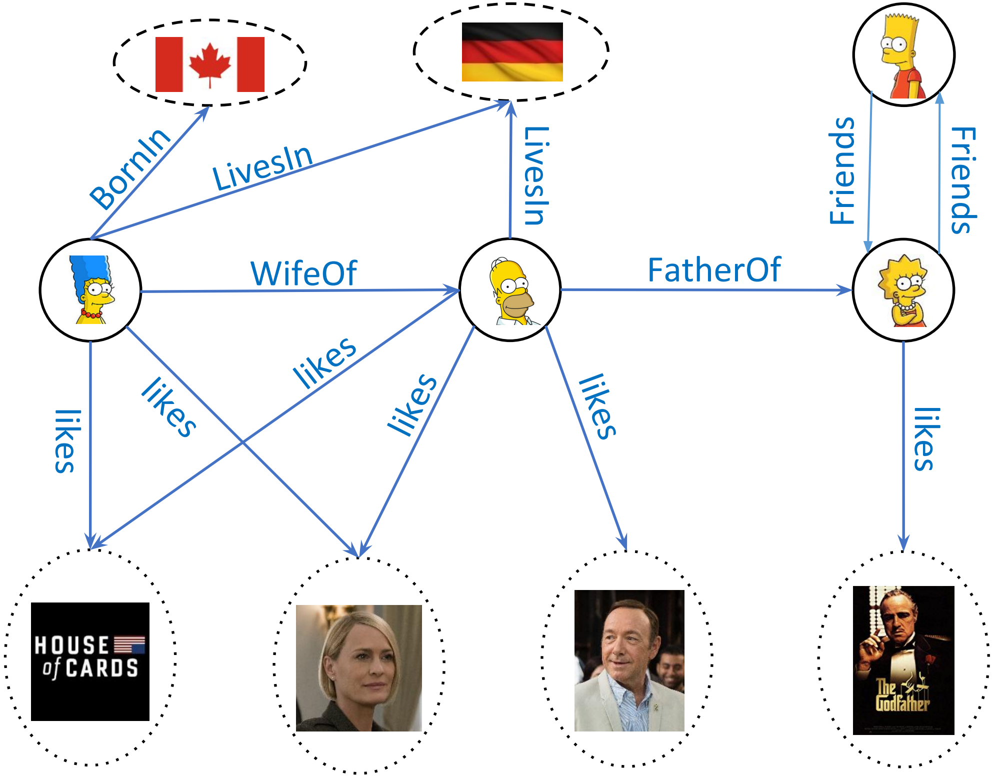
Knowledge Graphs

- Knowledge in the form of a graph!
  - Nodes represent entities.
  - Labelled edges represent relationships between entities.
- Can be also represented as a set of (subject, relation, object) triples:

( , WifeOf, )

( , Likes, )

...



Knowledge Graph Applications

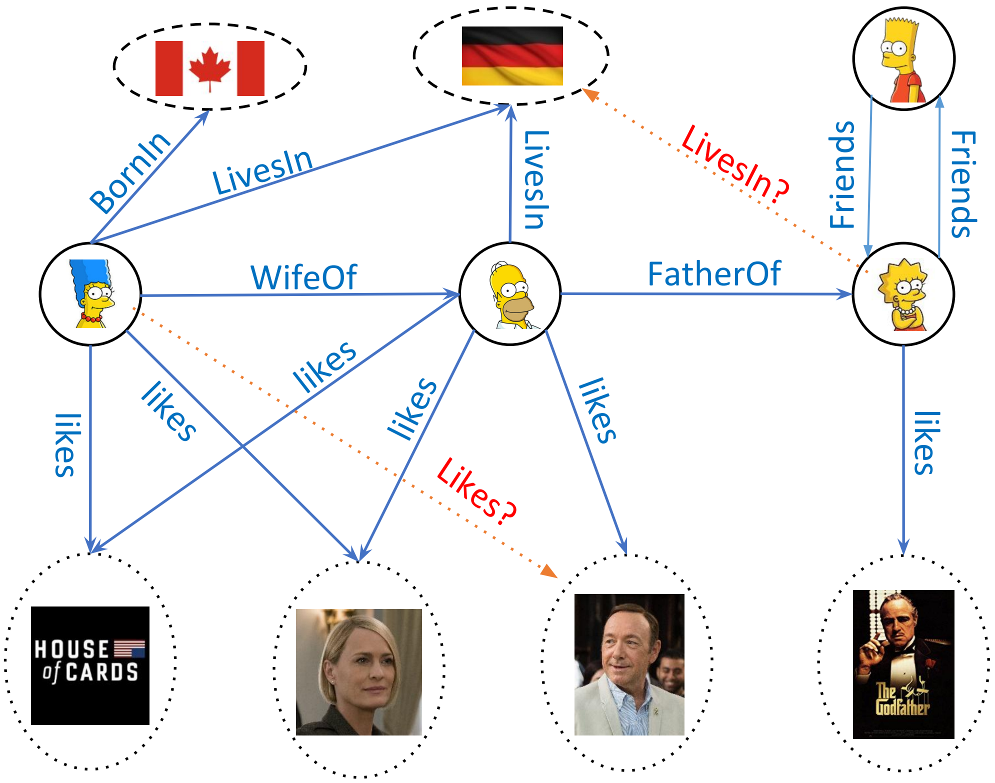
Question Answering

Predicting the Election

Problem Definition & Previous Work

Knowledge Graph (KG) Completion

- **Formally:**
  - $\mathcal{E}$: A set of entities
  - $\mathcal{R}$: A set of relations
  - ζ: Set of all triples involving entities from $\mathcal{E}$ and relations from $\mathcal{R}$ that are facts
  - Knowledge graph: $\mathcal{G} \subset \zeta$
  - KG completion: Inferring ζ from $\mathcal{G}$



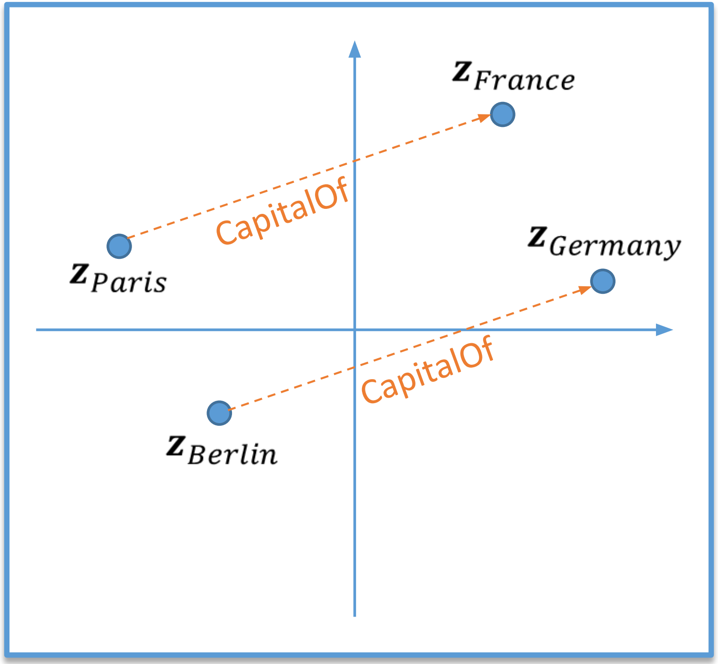
Translational Models: Inspiration

- Observation from word embeddings:

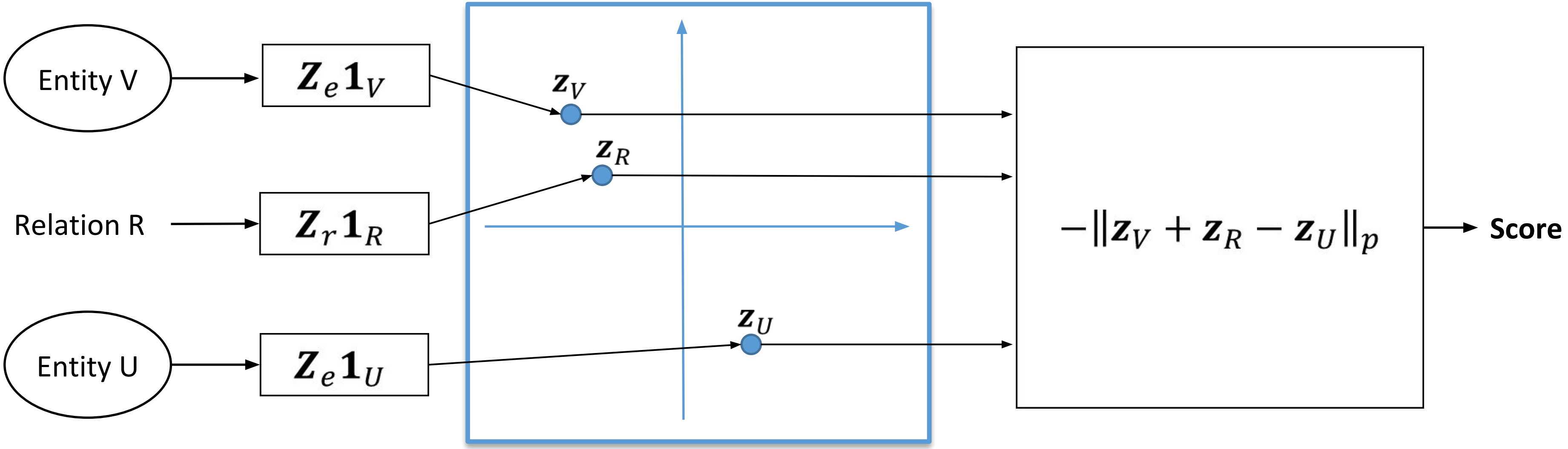
$$\mathbf{z}_{France} - \mathbf{z}_{Paris} \approx \mathbf{z}_{Germany} - \mathbf{z}_{Berlin}$$

- Idea:
  - Model relations as translations from subject entities to object entities.

Word Embeddings

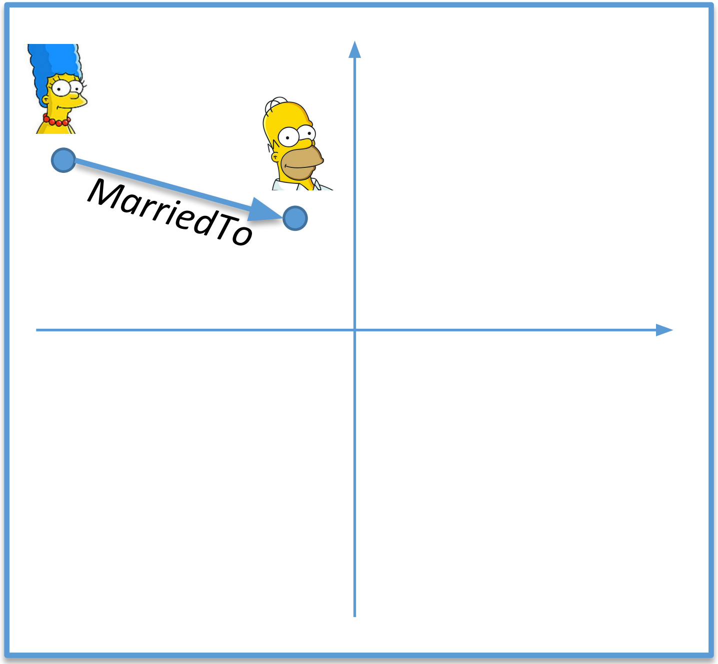


TransE

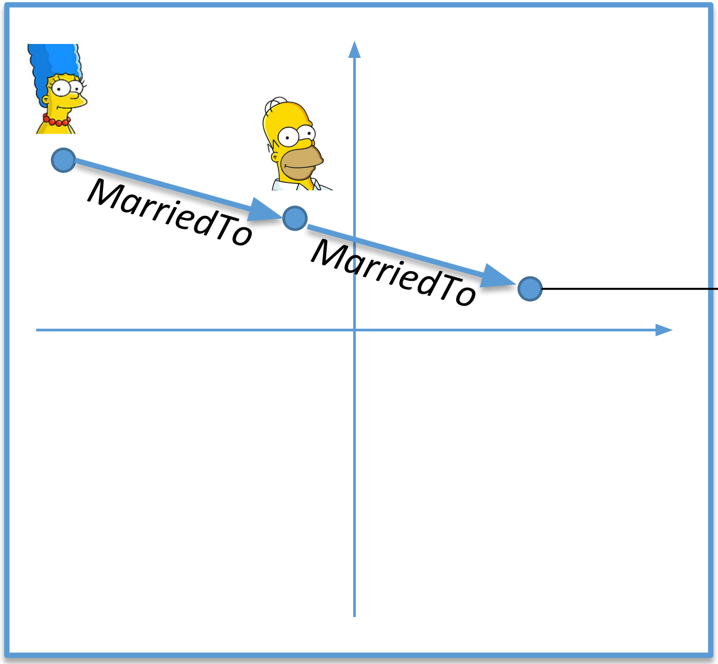


- $\boldsymbol{Z}_e$ and $\boldsymbol{Z}_r$ are matrices with learnable parameters.
- $\mathbf{1}_V$ and $\mathbf{1}_R$ are one hot encodings of node V and relation R.

TransE: Example



TransE and Symmetric Relations



An entity who is married to Homer should be mapped to a point close to this point.

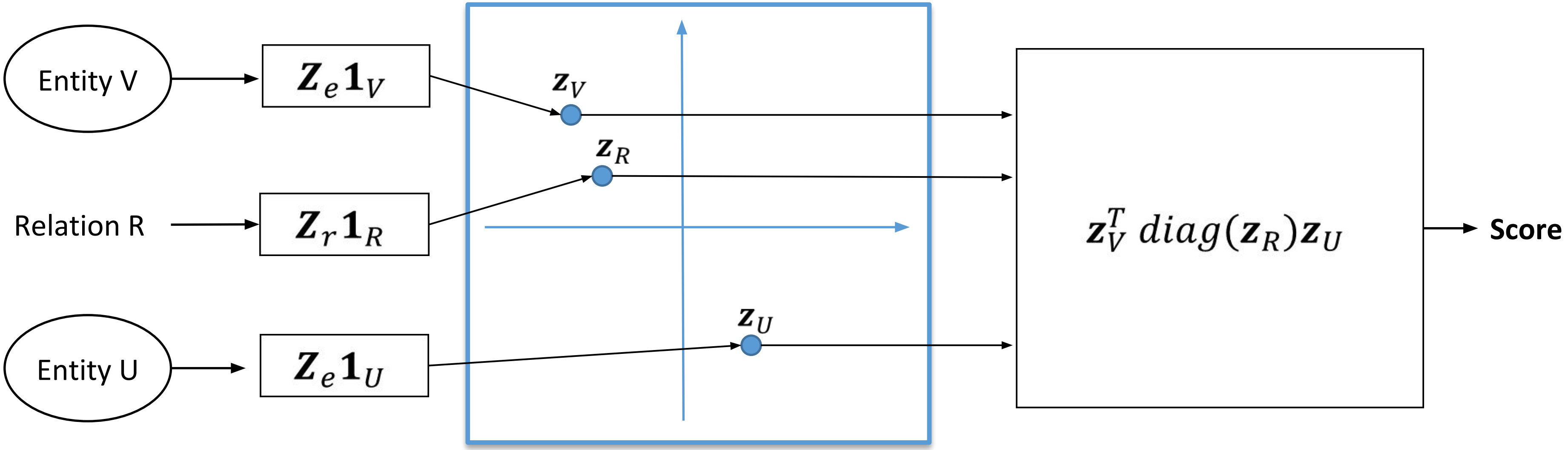
Other issues with translational models

- **FSTransE:** A translational model that subsumed existing translational models at the time
  - Score function: $\phi(V, R, U) = -\min_{\alpha} \|\boldsymbol{P}_R \boldsymbol{z}_V + \boldsymbol{z}_R - \alpha \boldsymbol{Q}_R \boldsymbol{z}_U\|_p$

Theorem: FSTransE has the following restrictions on the types of relations it can model:

- If a relation R is reflexive on $\Delta \subset \mathcal{E}$, R must also be symmetric on Δ.
- If a relation R is reflexive on $\Delta \subset \mathcal{E}$, R must also be transitive on Δ.
- If entity V_1 has relation R with every entity in $\Delta \subset \mathcal{E}$ and entity V_2 has relation R with a single entity in Δ, then V_2 must have relation R with every other entity in Δ.

DistMult

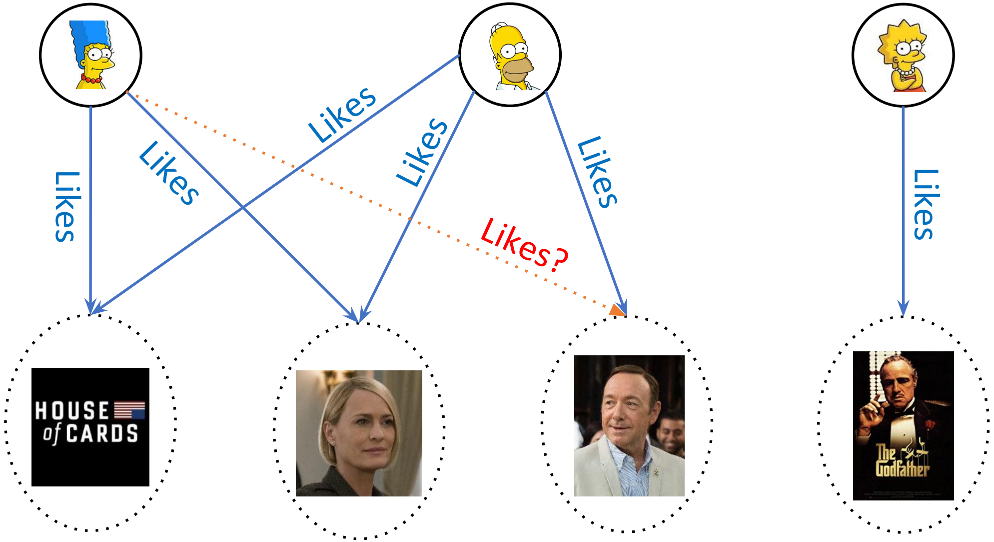


- $\boldsymbol{Z}_e$ and $\boldsymbol{Z}_r$ are matrices with learnable parameters.
- $\mathbf{1}_V$ and $\mathbf{1}_R$ are one hot encodings of node V and relation R.

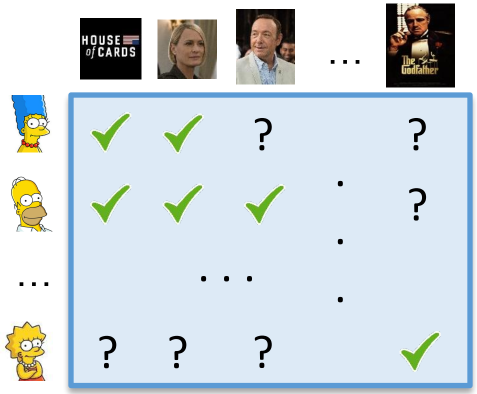
SimplE

Inspiration from matrix factorization

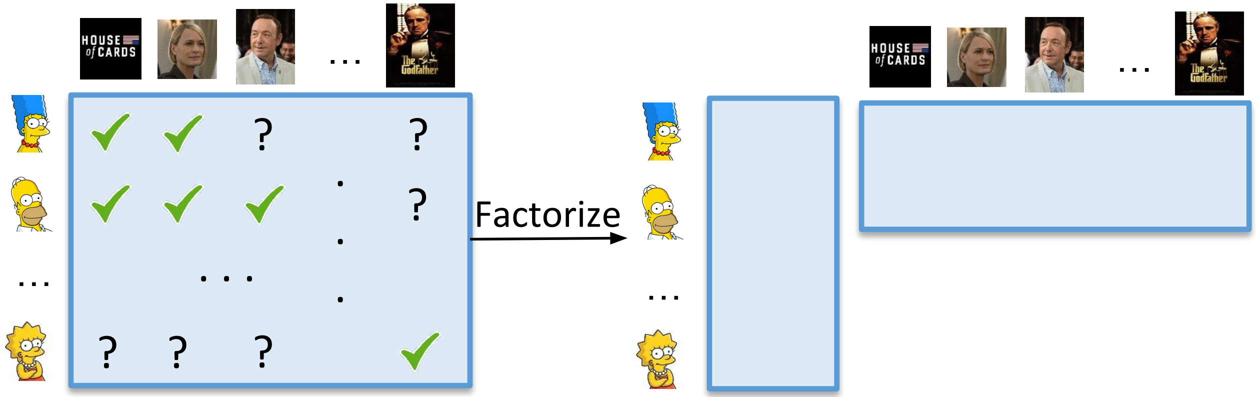
Let's start with a simple case where there is only one type of relationship



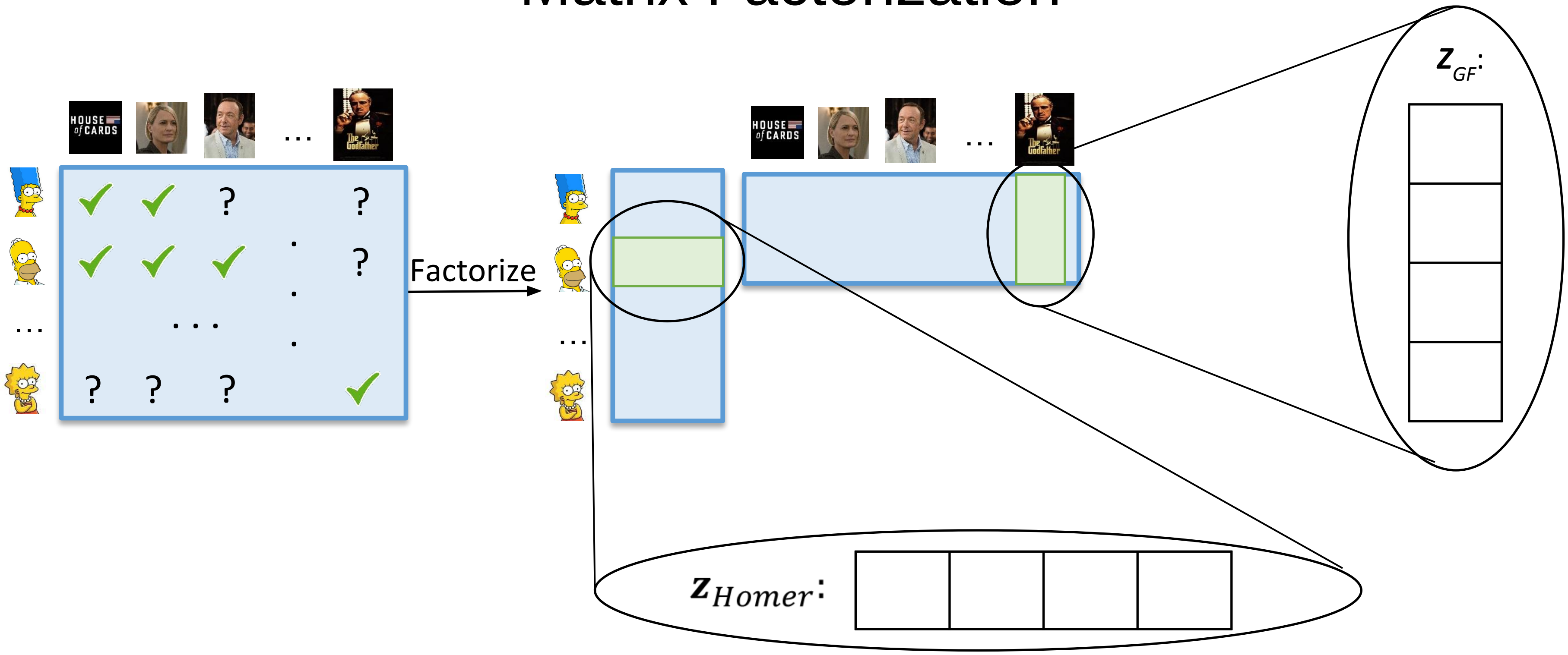
Adjacency Matrix



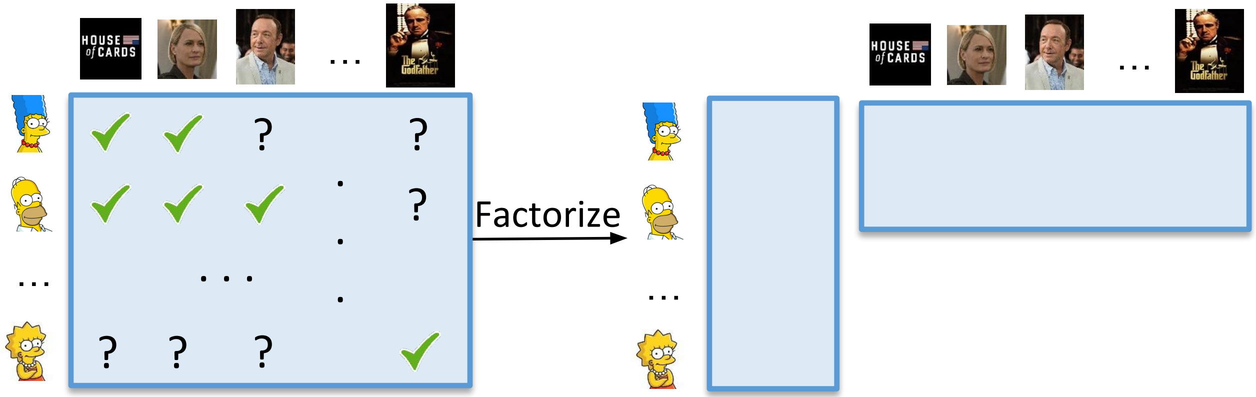
Matrix Factorization



Matrix Factorization



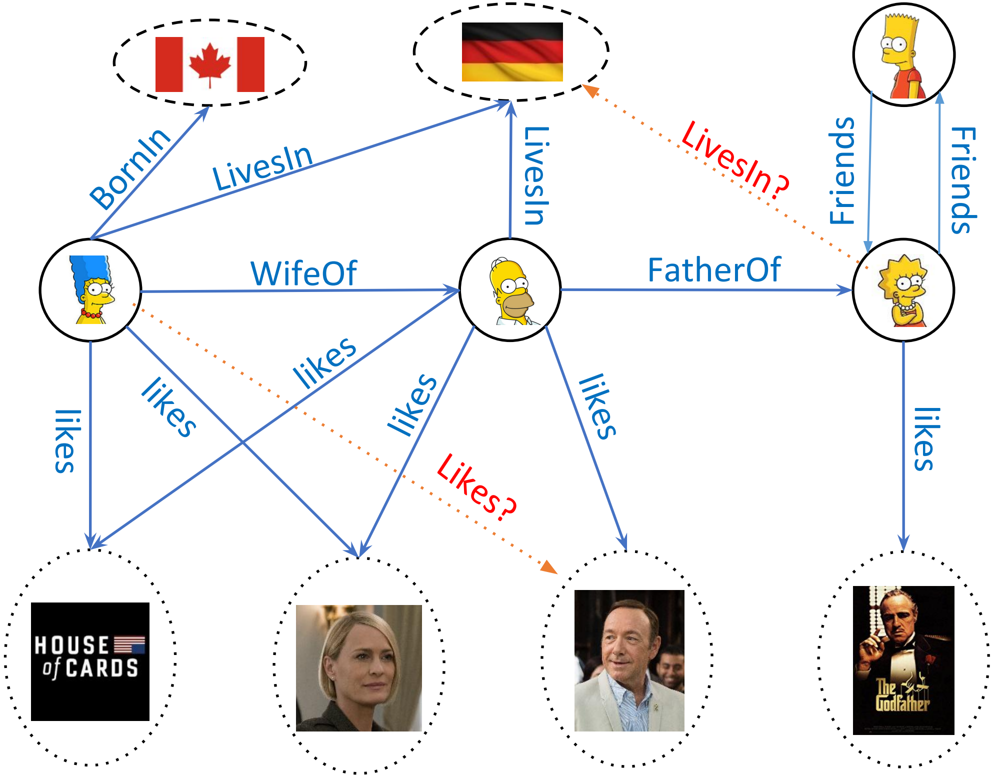
Matrix Factorization



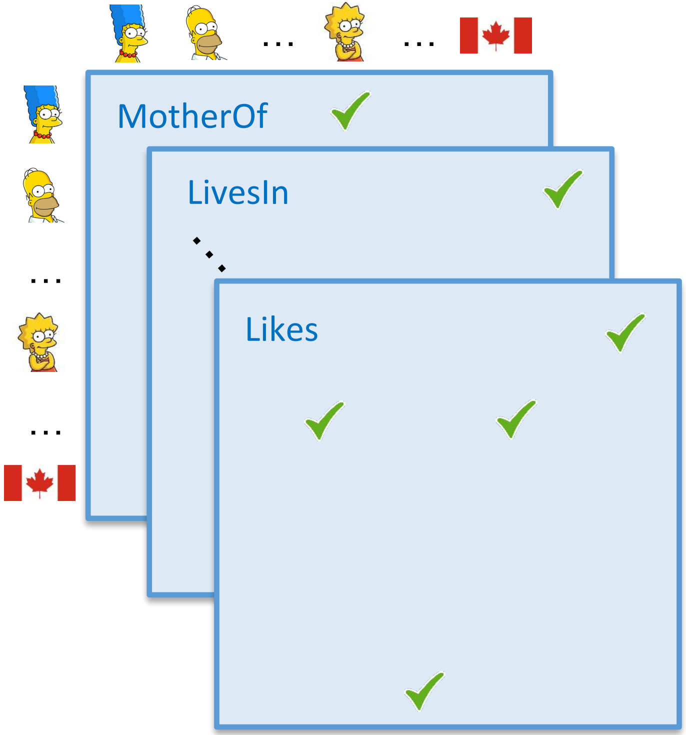
Training: Learn embedding parameters by minimizing the following loss function:

$$\mathcal{L}(\theta) = \sum_{(U,P)\in Train} \left(\mathcal{L}^{+}\big(\phi_\theta(U,P)\big) + \sum_{(U',P')\in Neg(U,P)} \mathcal{L}^{-}\big(\phi_\theta(U',P')\big) \right)$$

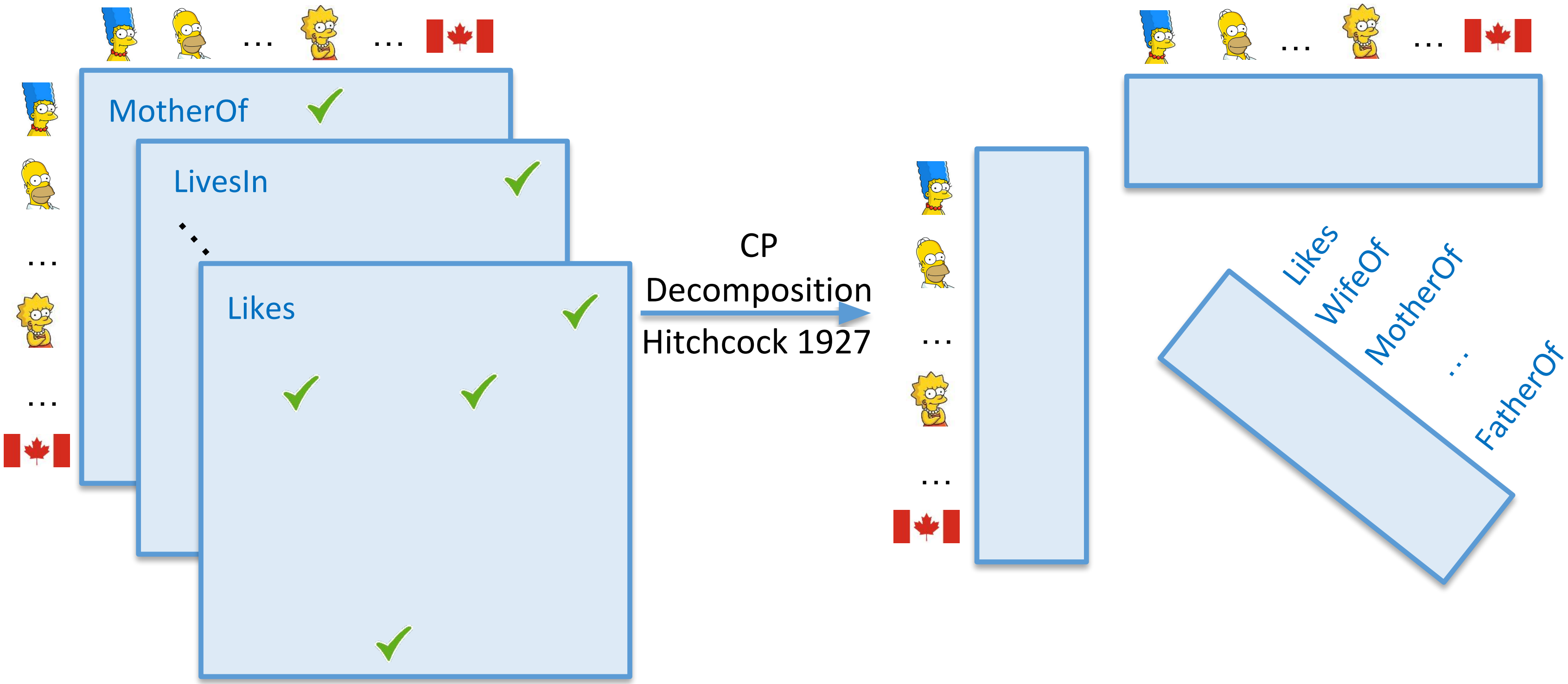
Knowledge Graph Completion



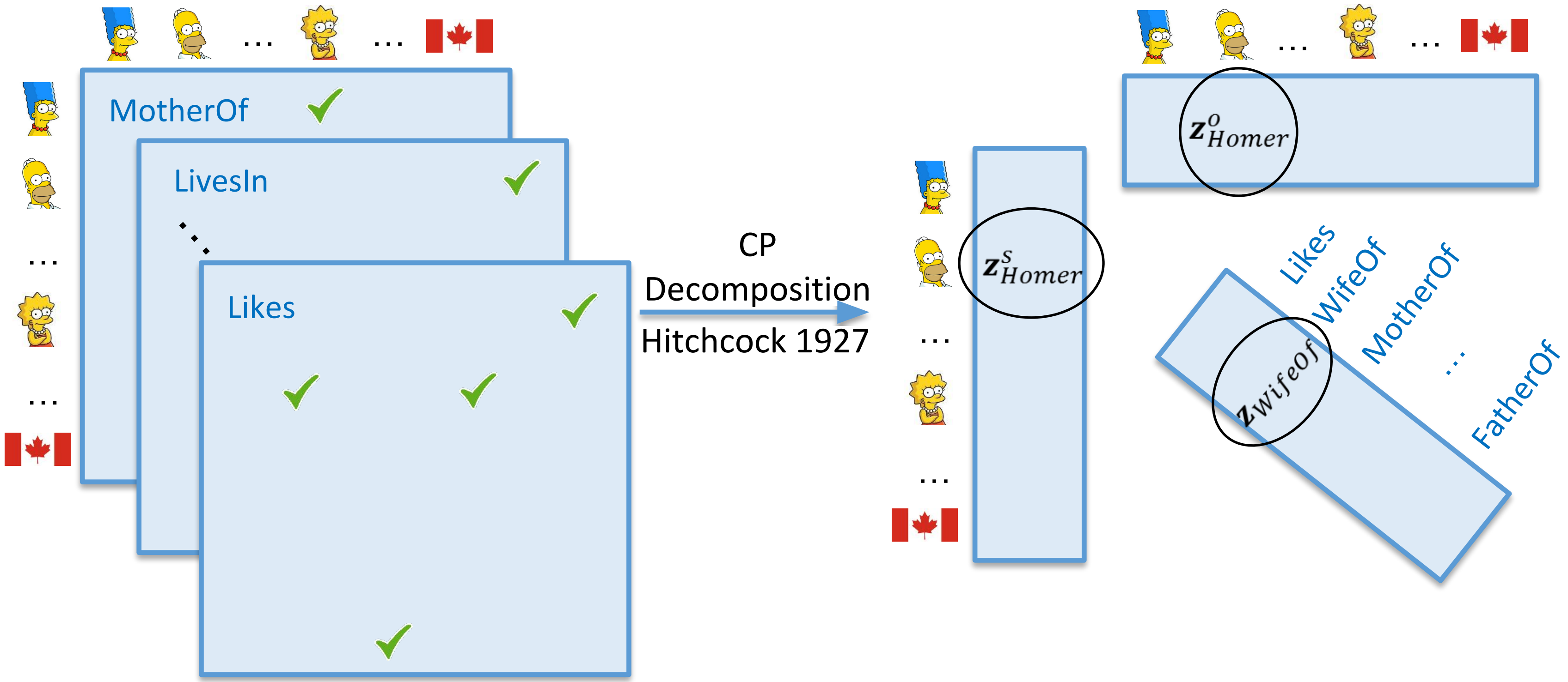
Adjacency Tensor



Tensor Factorization



Tensor Factorization



Tensor Factorization

- **Train:** Learn the embedding parameters by minimizing the following loss function:

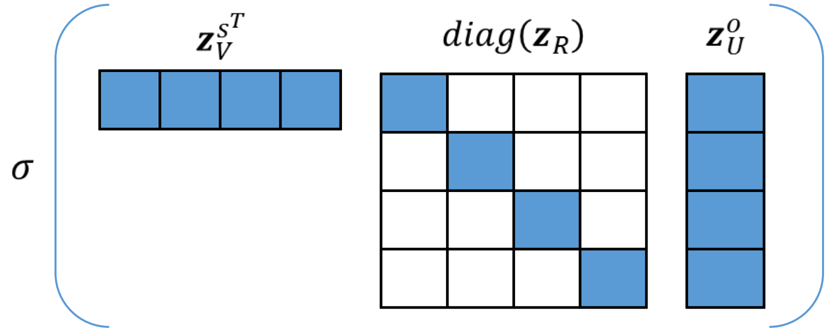
$$\mathcal{L}(\theta) = \sum_{(V,R,U)\in Train} \left(\mathcal{L}^{+}\big(\phi_{\theta}(V,R,U)\big) + \sum_{(V',R',U')\in Neg(V,R,U)} \mathcal{L}^{-}\big(\phi_{\theta}(V',R',U')\big) \right)$$

Tensor Factorization

- **Train:** Learn the embedding parameters by minimizing the following loss function:

$$\mathcal{L}(\theta) = \sum_{(V,R,U)\in Train} \left(-\log\big(\phi_\theta(V,R,U)\big) + \sum_{(V',R',U')\in Neg(V,R,U)} -log\big(1-\phi_\theta(V',R',U')\big) \right)$$

$$\phi_\theta(V,R,U) = \sigma(\mathbf{z}_V^{s^T} diag(\mathbf{z}_R)\, \mathbf{z}_U^o)$$

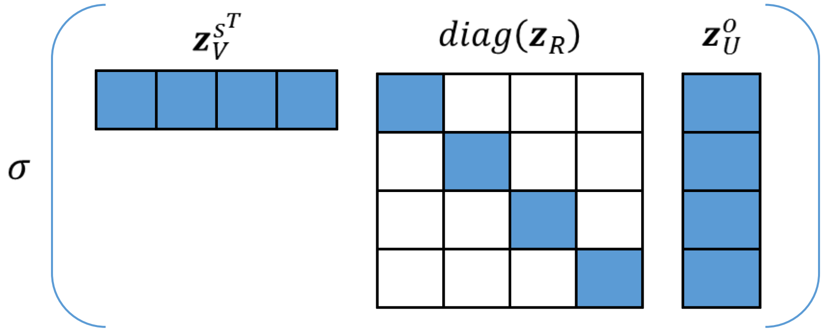


Tensor Factorization

- **Train:** Learn the embedding parameters by minimizing the following loss function:

$$\mathcal{L}(\theta) = \sum_{(V,R,U) \in Train} \left(-\log\big(\phi_\theta(V,R,U)\big) + \sum_{(V',R',U') \in Neg(V,R,U)} -log\big(1 - \phi_\theta(V',R',U')\big) \right)$$

$$\phi_\theta(V,R,U) = \sigma(\mathbf{z}_V^{s\,T}\, diag(\mathbf{z}_R)\, \mathbf{z}_U^{o})$$



➢ **Negative sampling based on local closed world assumption:**
  - Corrupting the subject: $(V,R,U) \rightarrow (V',R,U)$
  - Corrupting the object: $(V,R,U) \rightarrow (V,R,U')$

Negative Example Generator

Train Data

(*Michelle Obama*, Studied, *Princeton*)
(*Kevin Spacy*, PlayedInMovie, *House of Cards*)
...

Negative Example Generator

+ (*Michelle Obama*, Studied, *Princeton*)
- (*Michelle Obama*, Studied, *UBC*)
- (*Melania Trump*, Studied, *Princeton*)

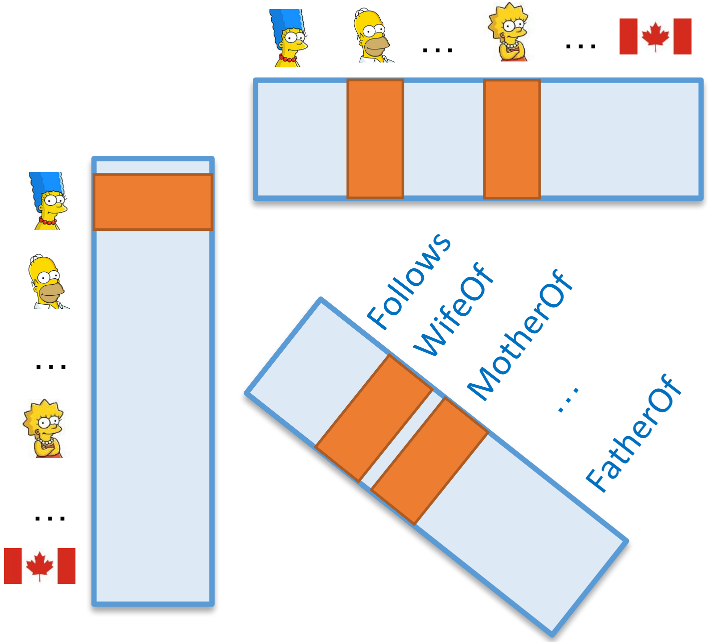
...

Softmax Loss

$$\mathcal{L}(\{\mathbf{r}\}, \{\mathbf{e}\}) = \sum_{x' \in \tau'_{train}} -log\left(\frac{e^{\phi(x')}}{\sum_{x \in T_{neg}(x')} e^{\phi(x)} + e^{\phi(x')}}\right)$$

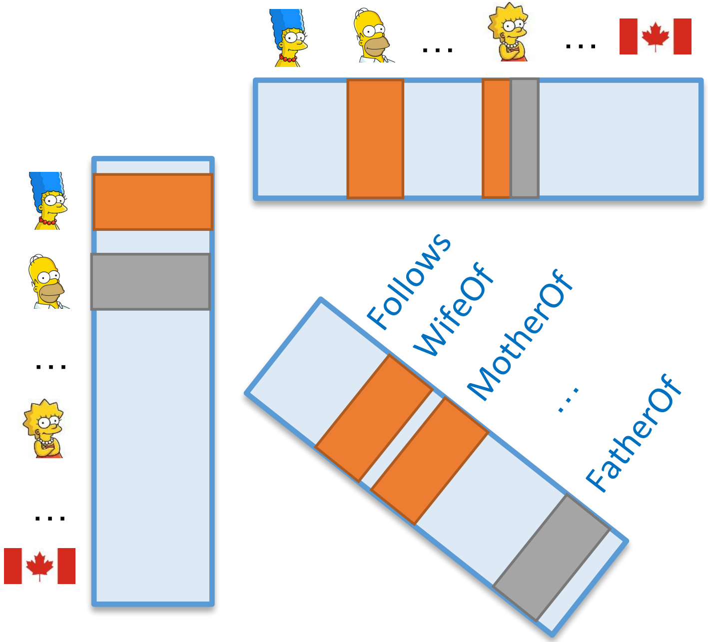
Analysis of CP Decomposition

- Observations (train set):
  - ( , WifeOf , )
  - ( , MotherOf, )



Analysis of CP Decomposition

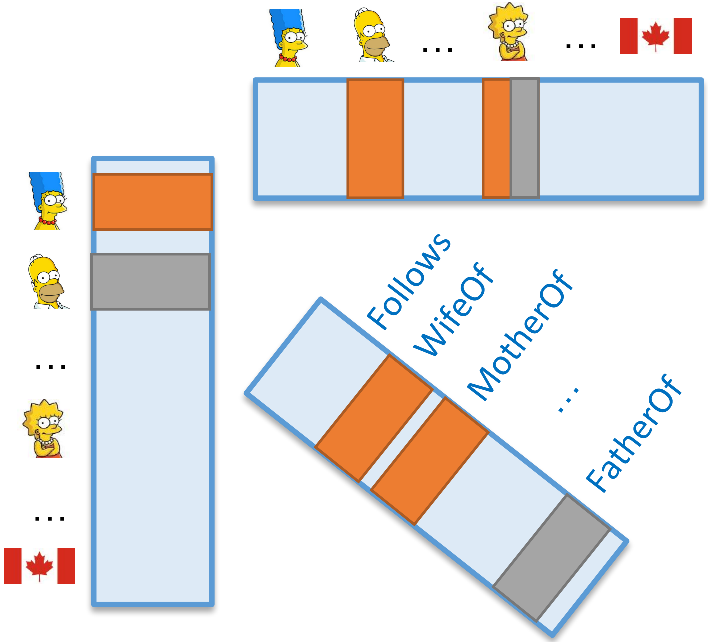
- Observations (train set):
  - ( , WifeOf , )
  - ( , MotherOf, )
- Query (test set):
  - ( , FatherOf, )



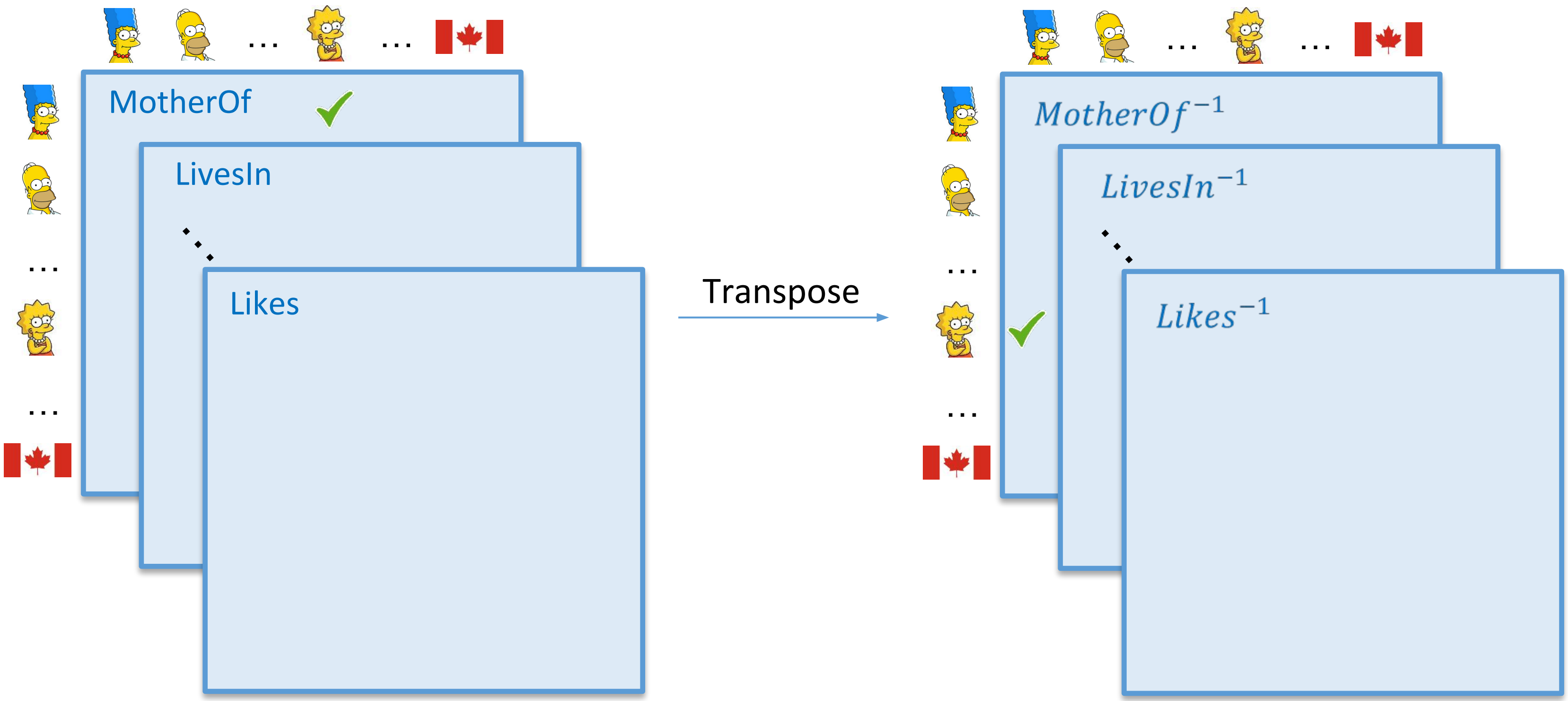
Analysis of CP Decomposition

- Observations (train set):
  - ( , WifeOf , )
  - ( , MotherOf, )
- Query (test set):
  - ( , FatherOf, )

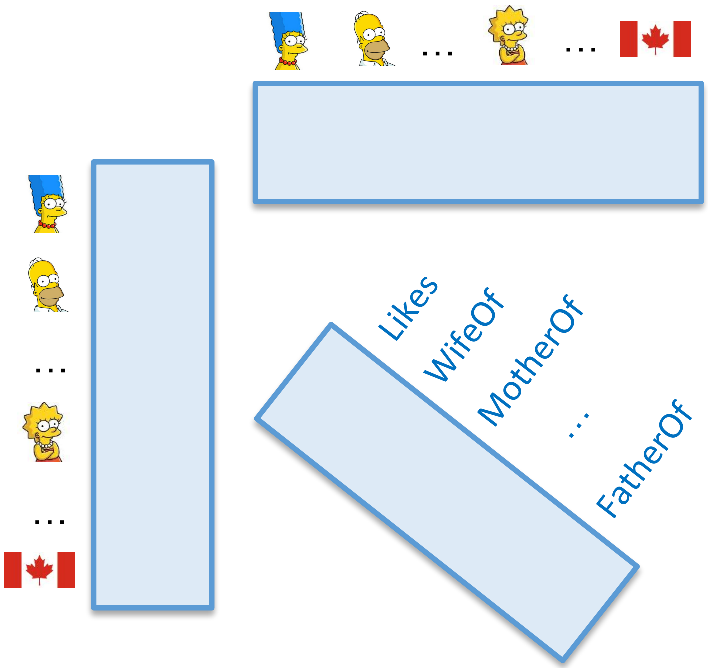
The information does not flow well between the two entity embeddings

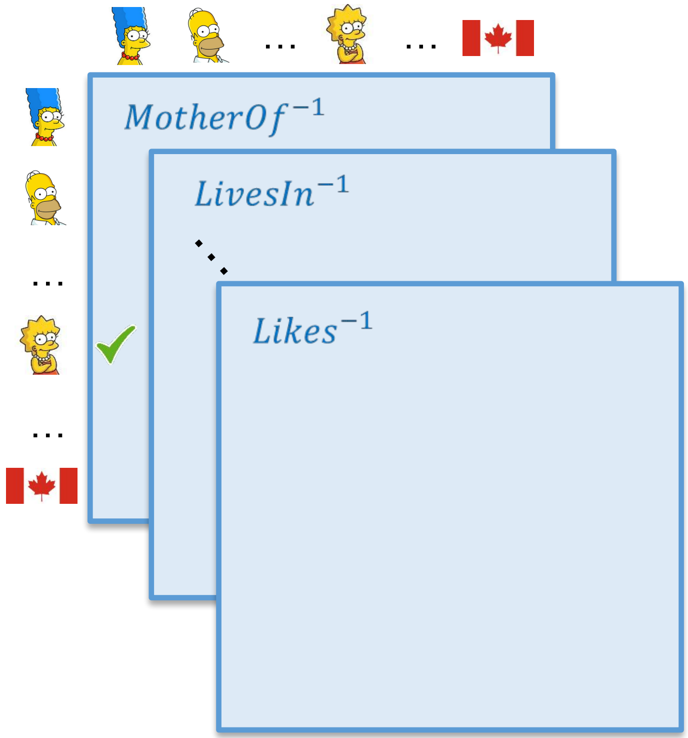


Solving the Information Flow Problem

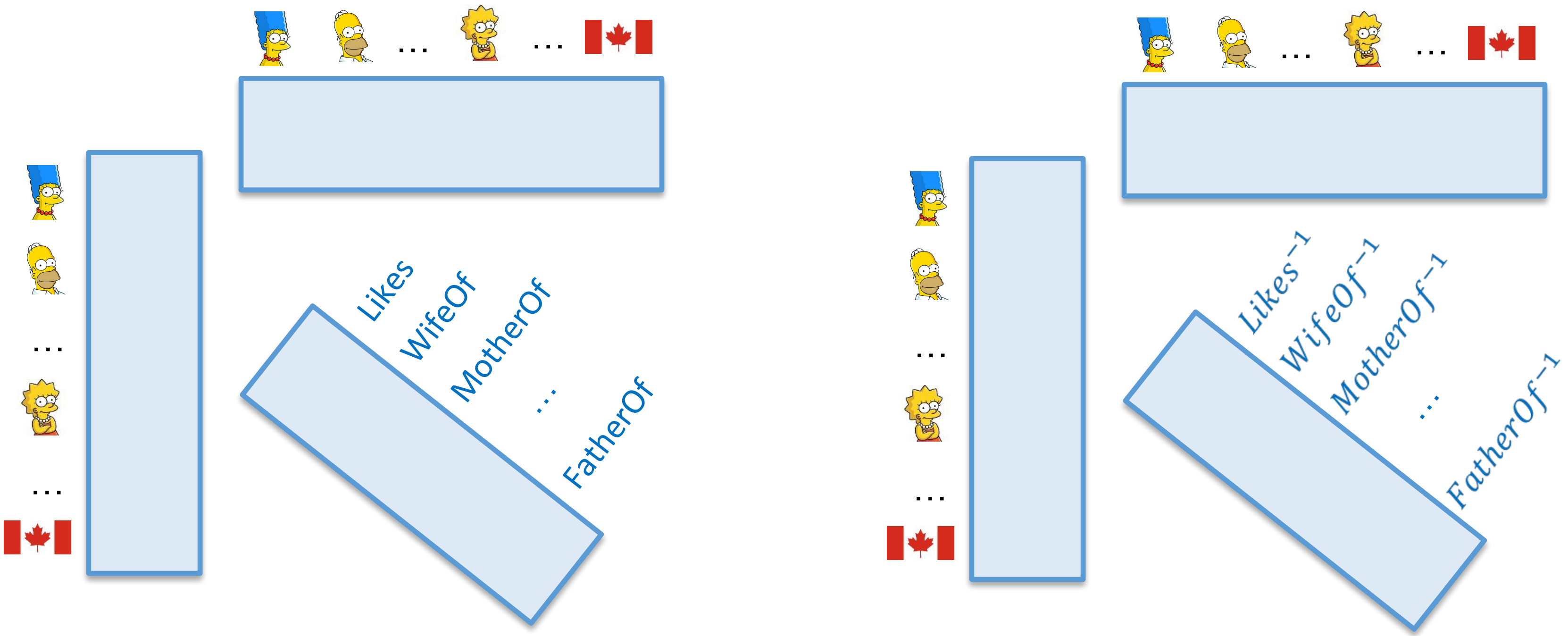


Solving the Information Flow Problem

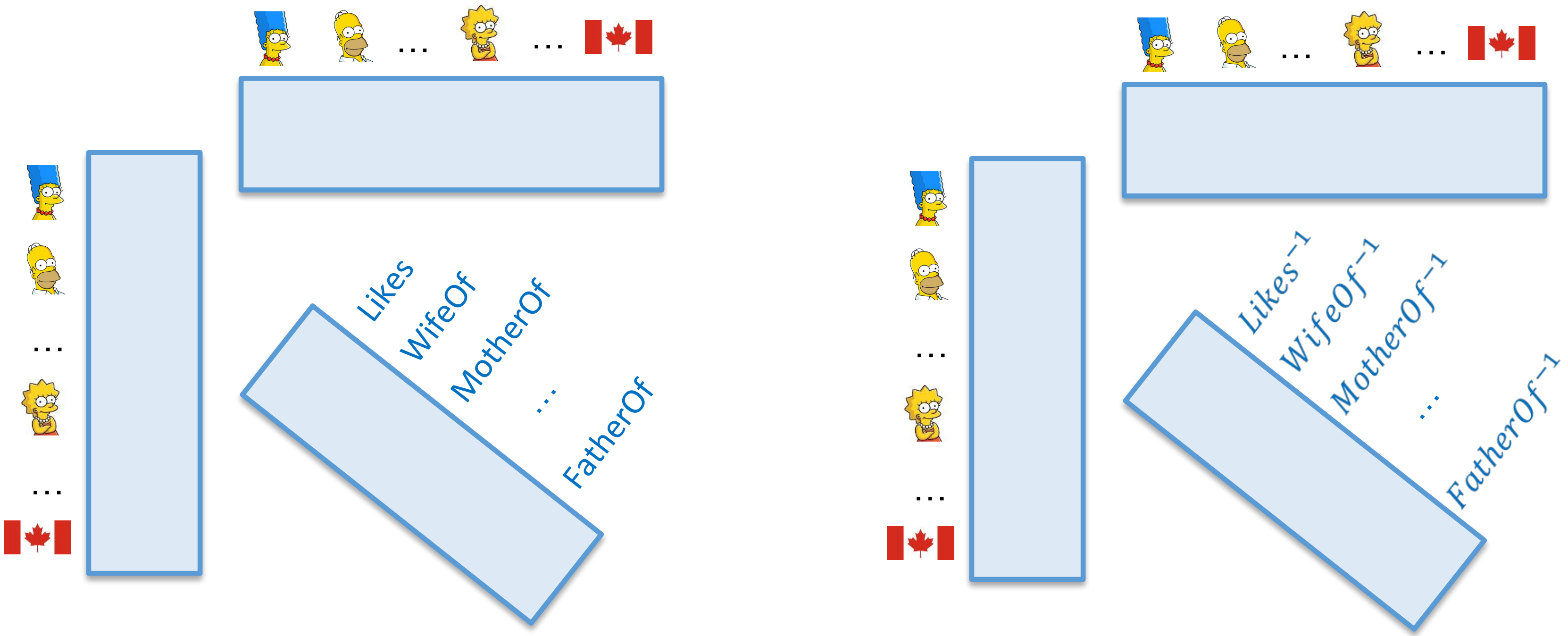




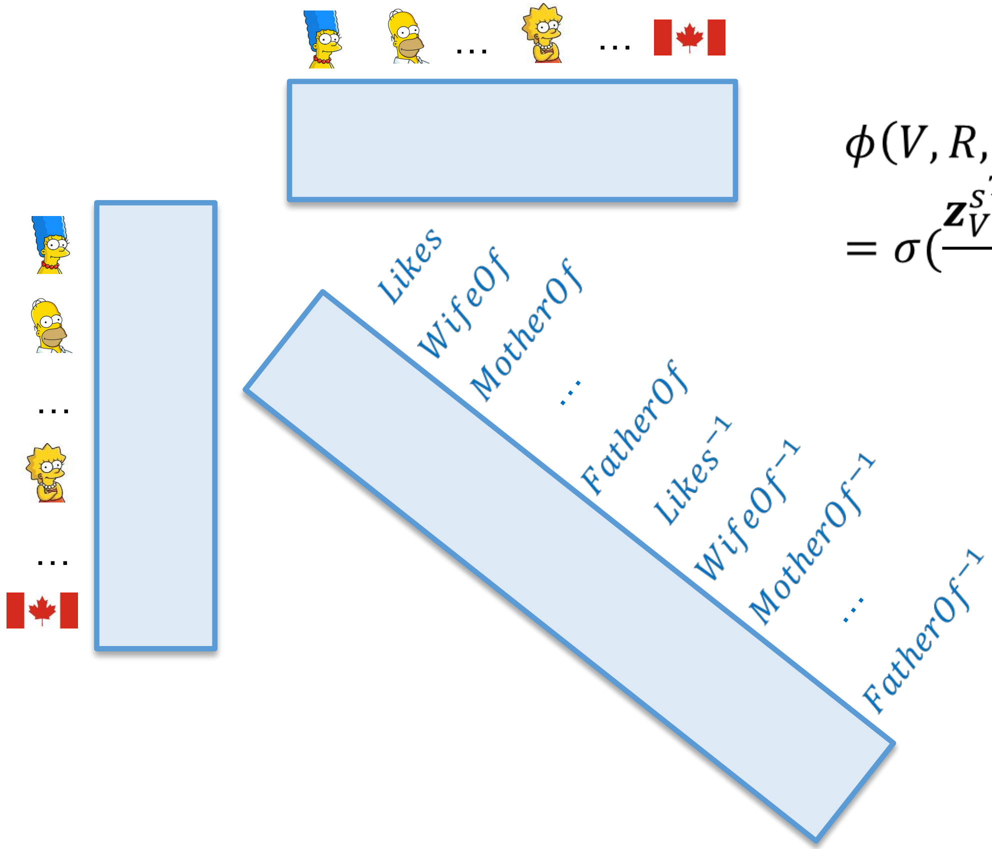
Solving the Information Flow Problem



Decompose both tensors
Use shared entity embeddings
Take the average of the two scores



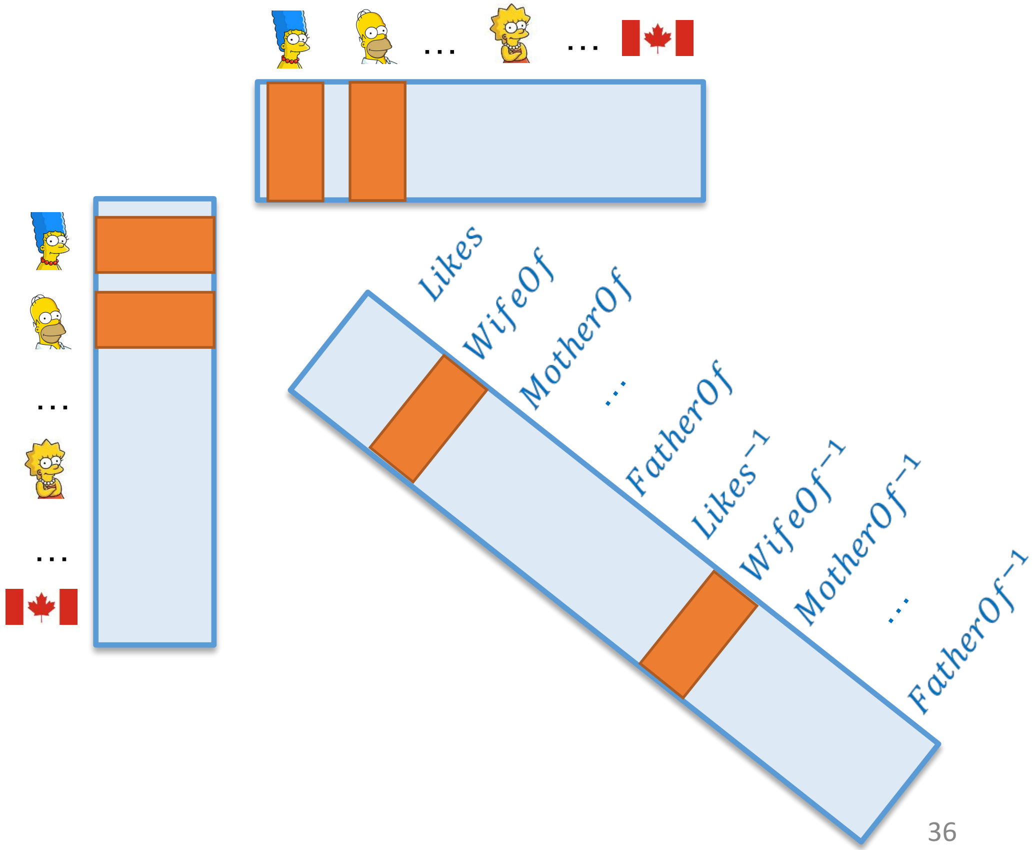
SimplE (Simple Embedding)



$$\phi(V,R,U) = \sigma\left(\frac{\mathbf{z}_V^{s\,T}\, diag(\mathbf{z}_R)\mathbf{z}_U^o + \mathbf{z}_U^{s\,T}\, diag(\mathbf{z}_{R^{-1}})\mathbf{z}_V^o}{2}\right)$$

Previous Example Revisited

- Observations (train set):
  - ( , $WifeOf$, )
  - ( , $WifeOf^{-1}$, )



Empirical & Theoretical Results

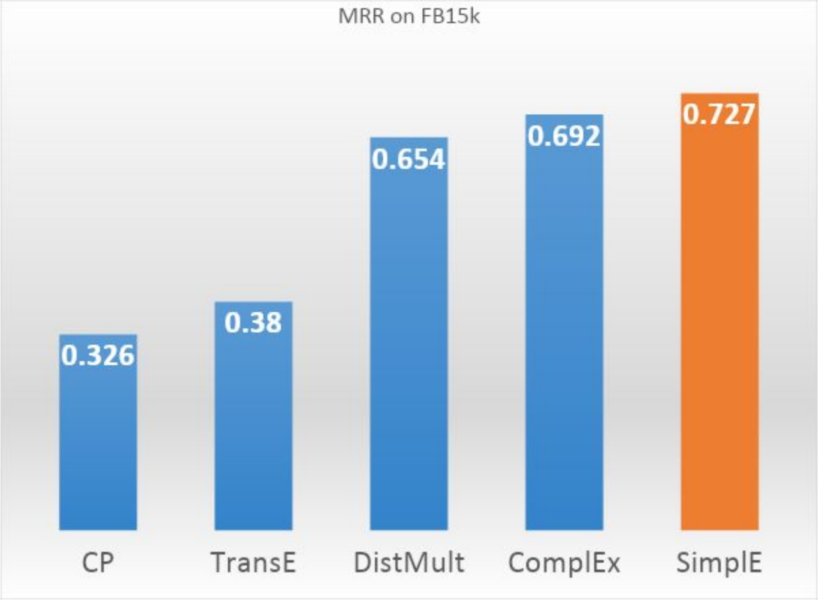
Evaluation Protocol

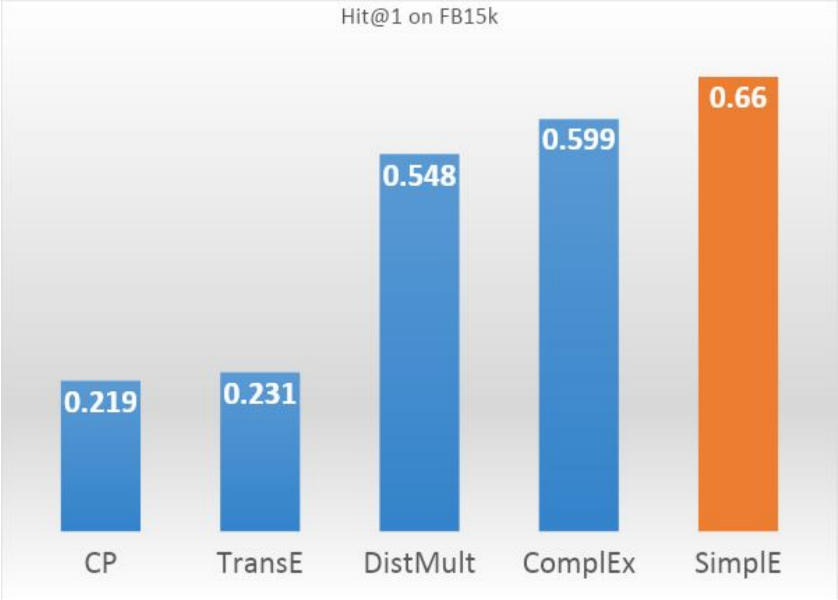
Test Triple	Query (Q)	Sorted Scores	Rank ($rank_Q$)
(Paris, CapitalOf, France)	(Paris, CapitalOf, ?)	(Paris, CapitalOf, Germany): 0.9	2
		(Paris, CapitalOf, France): 0.8	
		(Paris, CapitalOf, Canada): 0.1	
		…	

$$Mean\ Reciprocal\ Rank\ (MRR) = \frac{1}{|Q|}\sum_{Q}\frac{1}{rank_Q}$$

$$Hit@k = \frac{1}{|Q|}\sum_{Q}\mathbb{1}_{rank_Q \leq k}$$

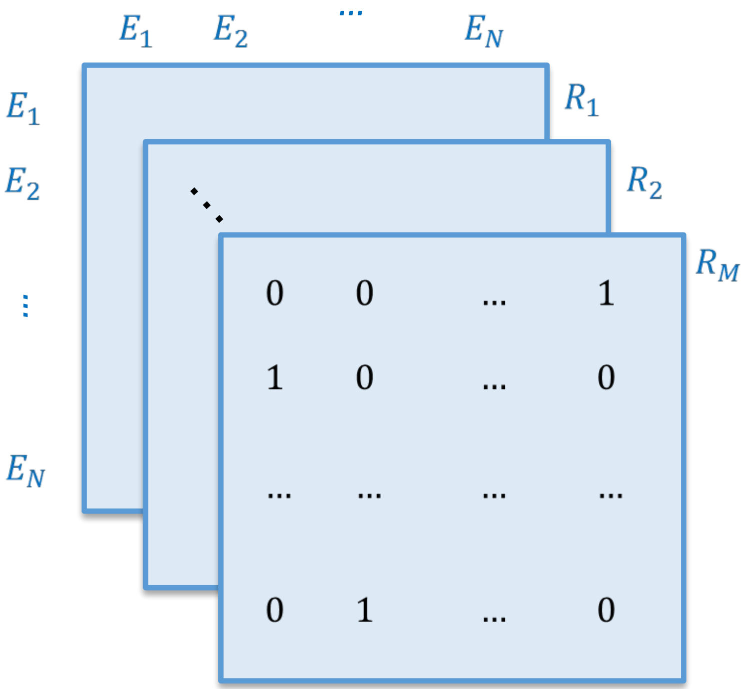
SimplE Results on FB15k

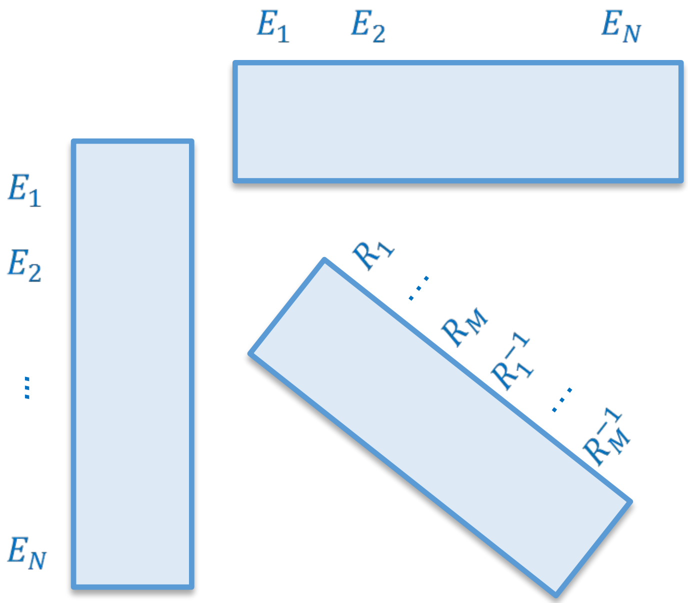




Theorem: SimplE is Fully Expressive

Given any ground truth adjacency tensor, there exists an instantiation of SimplE that correctly separates the 0s and 1s of the tensor.





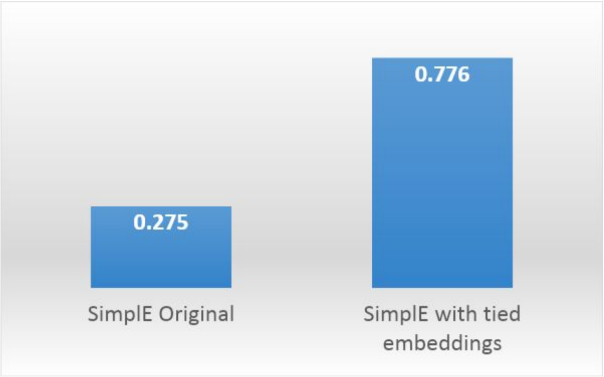
Incorporating background knowledge into the embeddings

- If R_i is known to be symmetric:
  - Tie $\boldsymbol{z}_{R_i}$ to $\boldsymbol{z}_{R_i^{-1}}$
- If R_i is known to be anti-symmetric:
  - Tie $\boldsymbol{z}_{R_i}$ to $-\boldsymbol{z}_{R_i^{-1}}$
- If R_i is known to be the inverse of R_j:
  - Tie $\boldsymbol{z}_{R_i}$ to $\boldsymbol{z}_{R_j^{-1}}$
  - Tie $\boldsymbol{z}_{R_j}$ to $\boldsymbol{z}_{R_i^{-1}}$

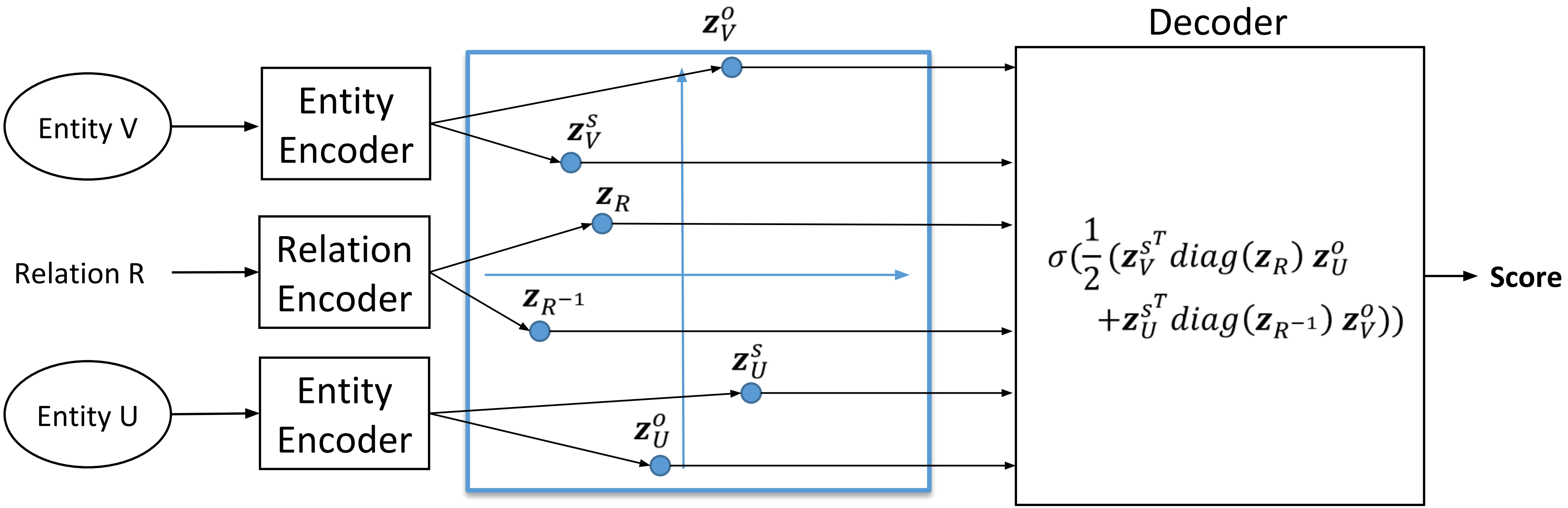
Experiment

Dataset: WN18

Setting: Remove any triple from the train set if it can be inferred from the background knowledge and the other triples in the train set



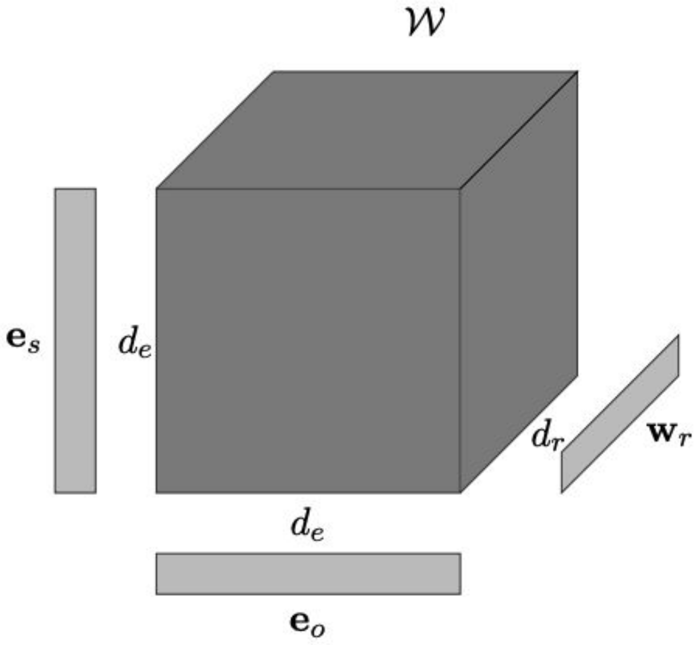
SimplE from an Encoder-Decoder Point-of-View



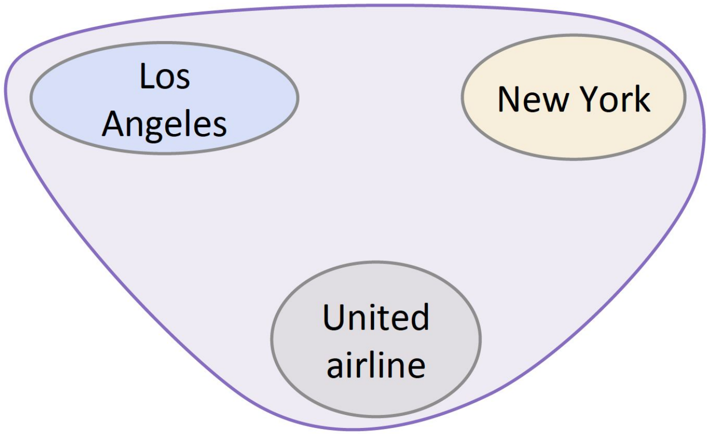
Other Applications of Tensors in Knowledge Graphs

TuckER: Tensor Factorization for Knowledge Graph Completion using TuckER Decomposition

Intuition: Rather than learning distinct relation specific matrices, learning a core tensor W containing a shared pool of “prototype” relation matrices.

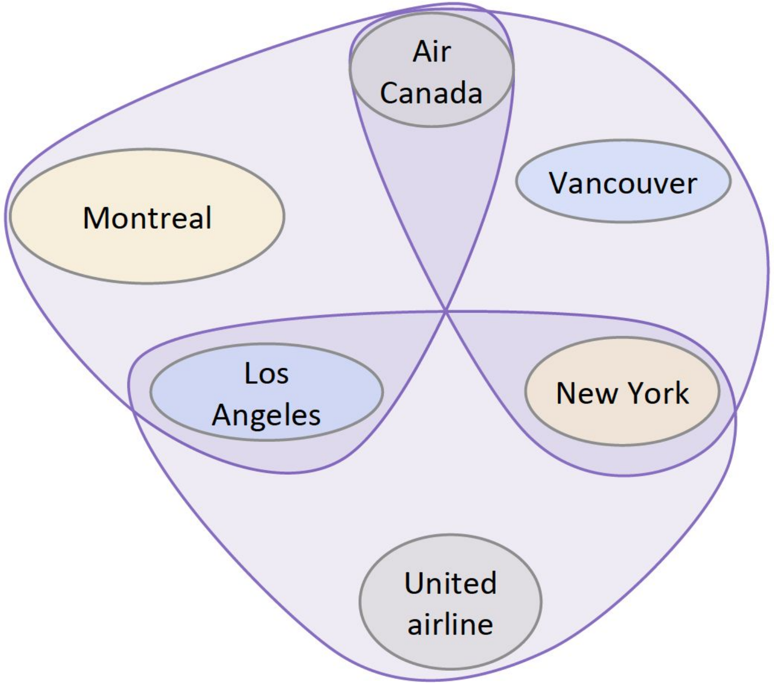


Binary vs. Beyond Binary Relations



Flies(Airline, Departure city, Arrival city)

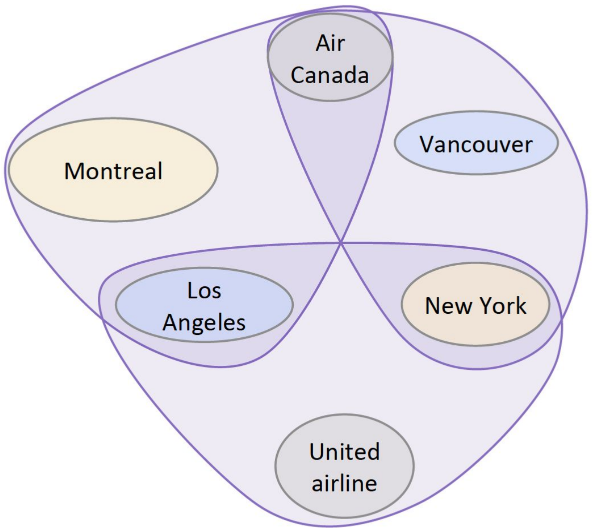
Knowledge Hypergraph as Tuples



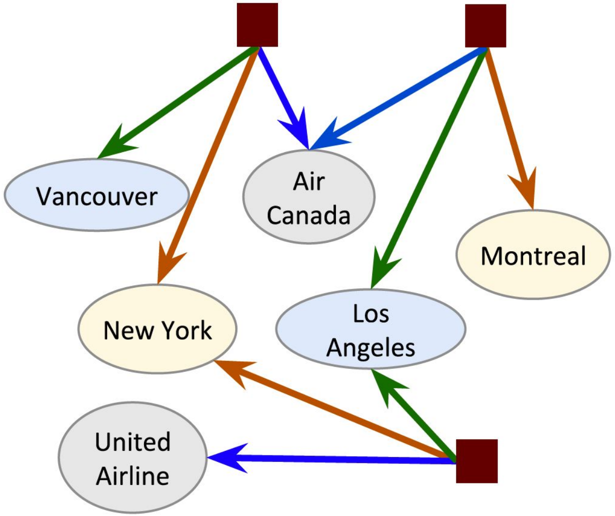
Flies(*Air Canada, Montreal, Los Angeles*)
Flies(*Air Canada, New York, Vancouver*)
Flies(*United airline, New York, Los Angeles*)

Relation(*entity 0, entity 1, …, entity n*)

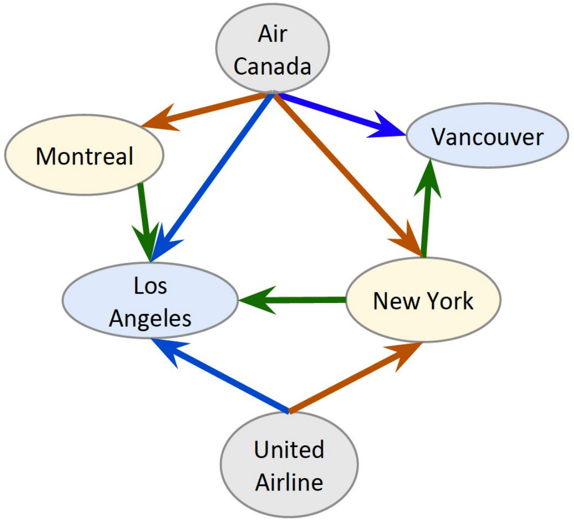
Knowledge Hypergraph



Hypergraph



Reification



Star to Clique

Question?