Collective Graph Neural Nets for Multi-type Entity Alignment

Qi Zhu[1], Hao Wei[2], Bunyamin Sisman[2], Da Zheng[2], Christos Faloutsos[3], Xin Luna Dong[2], Jiawei Han[1]

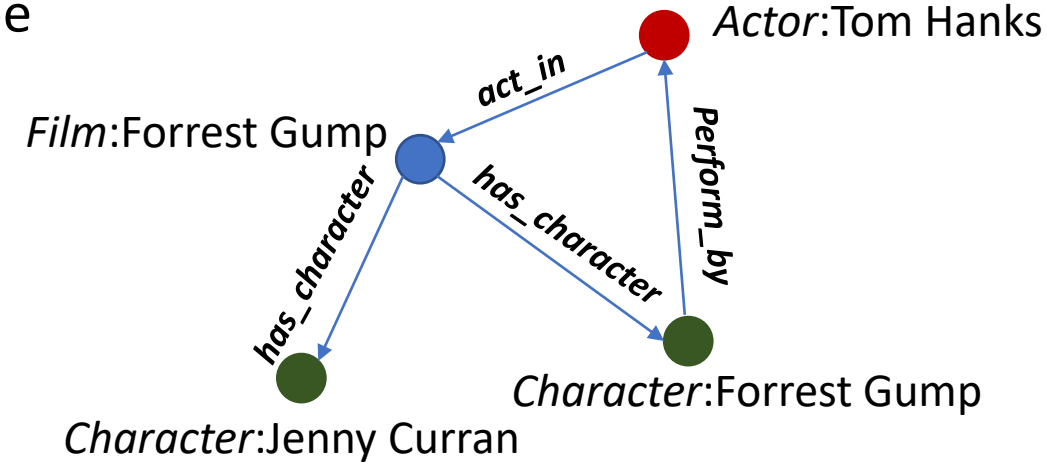
[1]University of Illinois at Urbana-Champaign
[2]Amazon, Inc.
[3]Carnegie Mellon University

Overview

- Introduction
  - Multi-type Entity Alignment
  - Data Representation
  - Graph Neural Networks
- Challenges
- Related work
- Proposed Framework
- Experiments
- Future work

Multi-type Entity Alignment

- Entity Alignment/Matching identifies entities from different data sources (i.e. 2) that refer to the same real-world entity.
  - Traditional industry system on single entity type
    - Blocking (candidate generation)
    - Feature Generation
    - Matching
- What & Why *multi-type* Alignment?
  - Purpose
    - end-to-end model for actor/film/character Alignment
  - Each entity could be multi-typed:
    - According to IMDB:
      - Tom Hanks is a actor/producer/writer and person
  - Decisions made on different types can affect each other
    - Movie acted by Tom Hanks will gain confidence if Tom Hanks are likely matched in Graph A and B



Example: IMDB graph

Data Representation

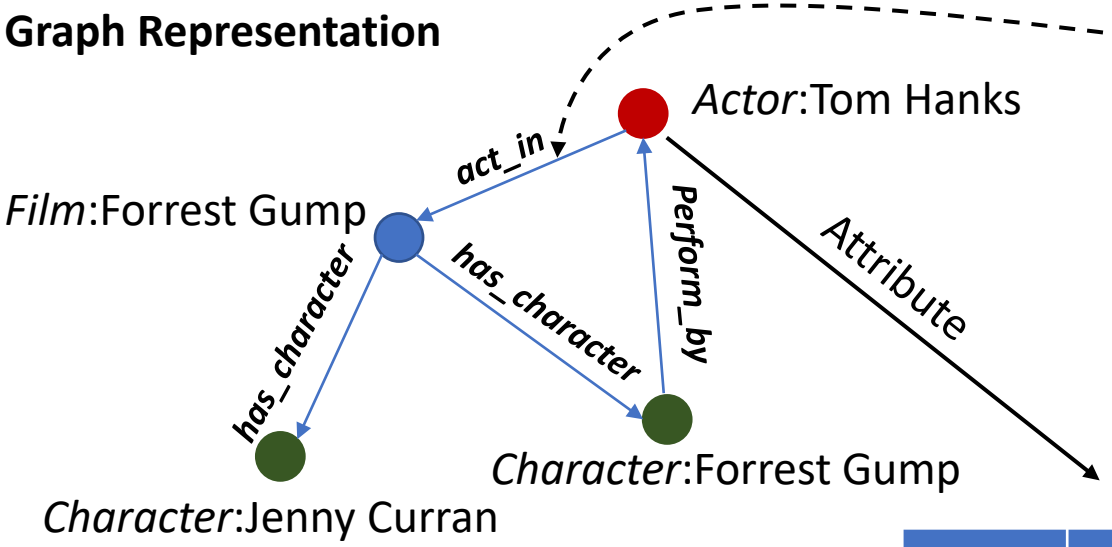
Relational Table Representation

Id	Name	PerformAs	ActIn	Age	Height
id/851nu	Tom Hanks	Forrect Gu…	Forrect Gu…	1956-07-09	1.83

Pros: Convenient, stand-alone data for every entities

Cons: limited representation power, expensive join computation for higher order information

Graph Representation

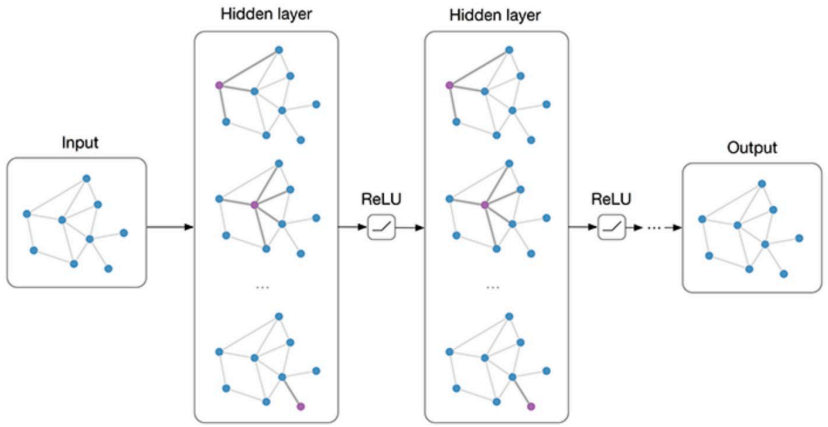


Relation predicate:
The object of the triple is also an entity
<Person, *act_in*, Film>
Attribute predicate:
The object of the triple is value
<Person, *has_name*, Chris>
<Person, *age*, 19>

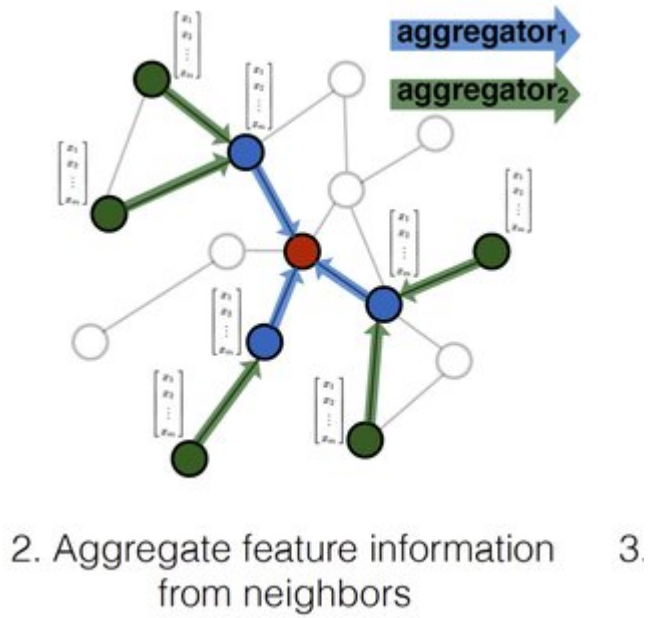
Name	Birth date	Death Date	Height	Place of Birth

Graph Neural Networks

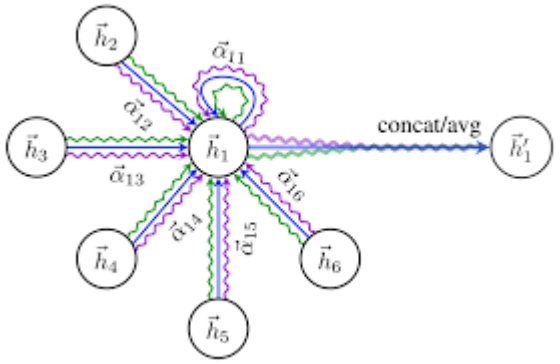
- Leverage the representation power of neural networks in Graph
- A suitable model to aggregate high order neighborhood information



Graph Convolutional Networks (Kipf, Thomas et al., 2016)



GraphSAGE (Hamilton et al., 2017)



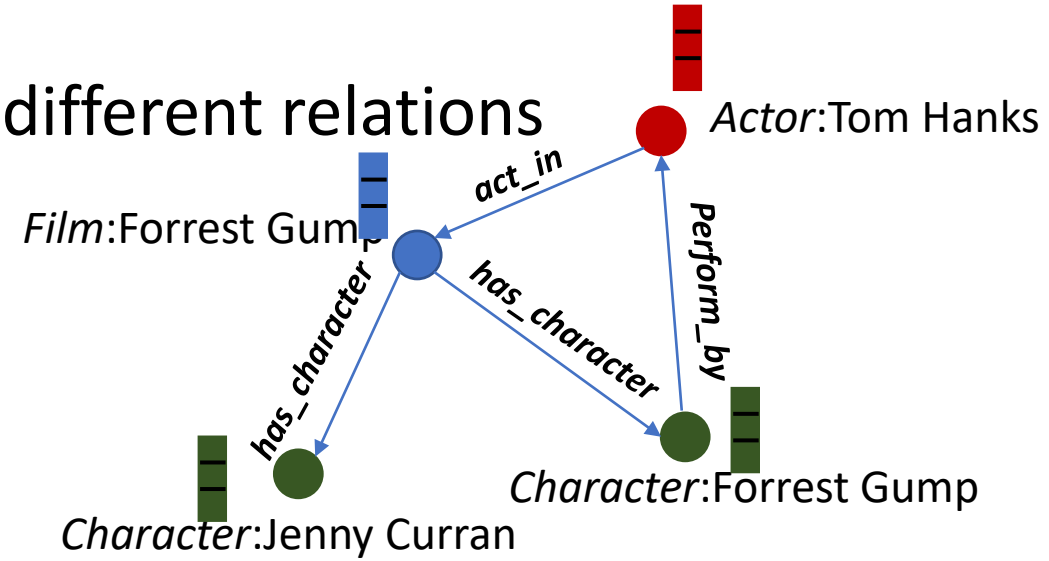
GAT (Veličković, Petar, et al., 2017)

Overview

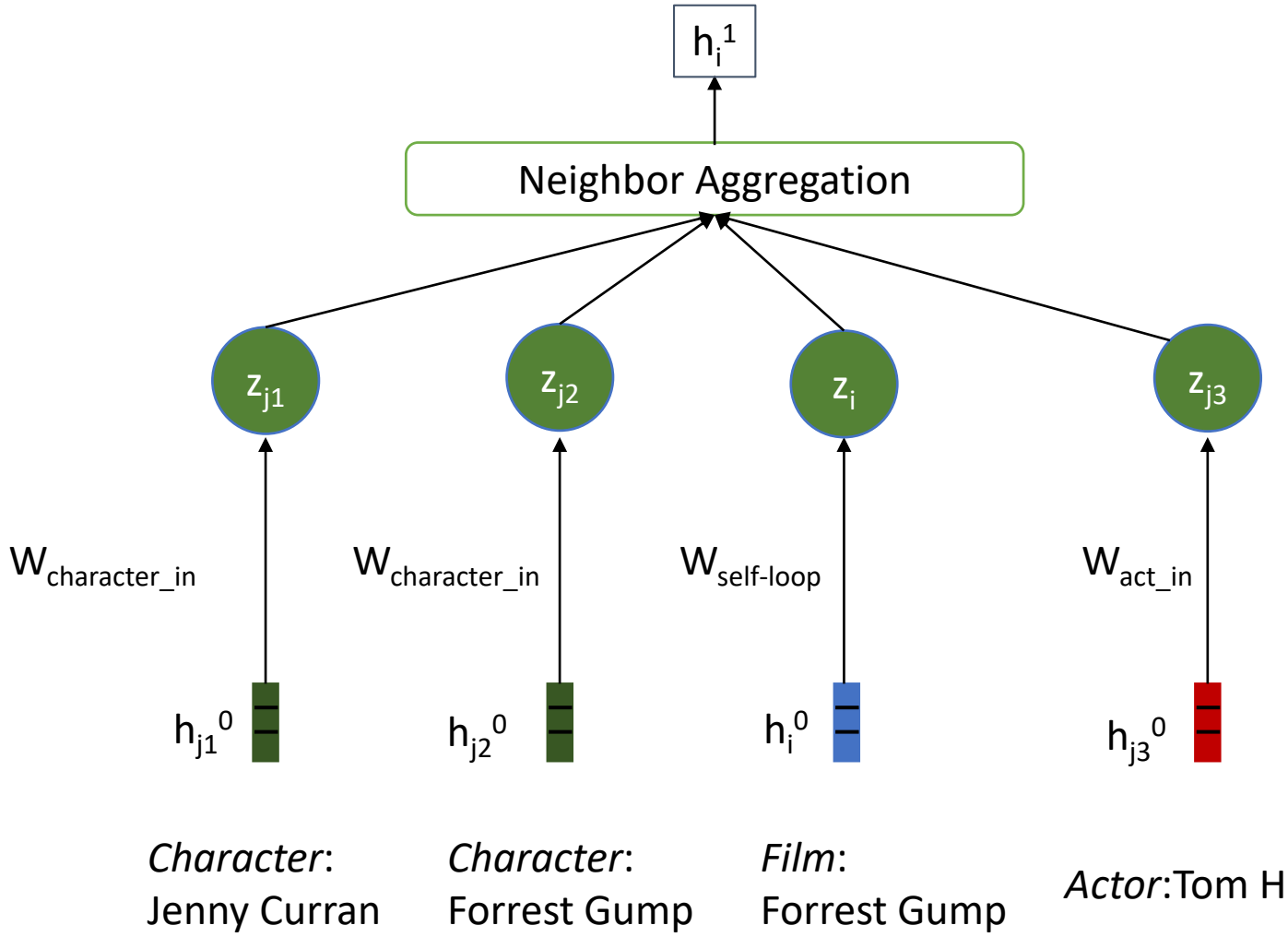
- Introduction
- Challenges
  - Attribute alignment for entities in different types
  - Aggregation function
  - Noisy neighborhood information
- Proposed Framework
- Experiments
- Related work
- Future work

Attribute encoding in multi-relational graph

- Different from previous application of GNN on knowledge graph embedding or knowledge graph alignment, which are attribute-light
- Attribute is important clue for the Entity Alignment in production
- Each entity can be subject or object for multiple relations
  - E.g. Film: Forrect Gump
- We use different matrices W_R for different relations
  - W_{act_in} X
  - $W_{has\_character}$ X
  - $W_{perform\_by}$ X
- Relational GNN



Relational GNN Layer



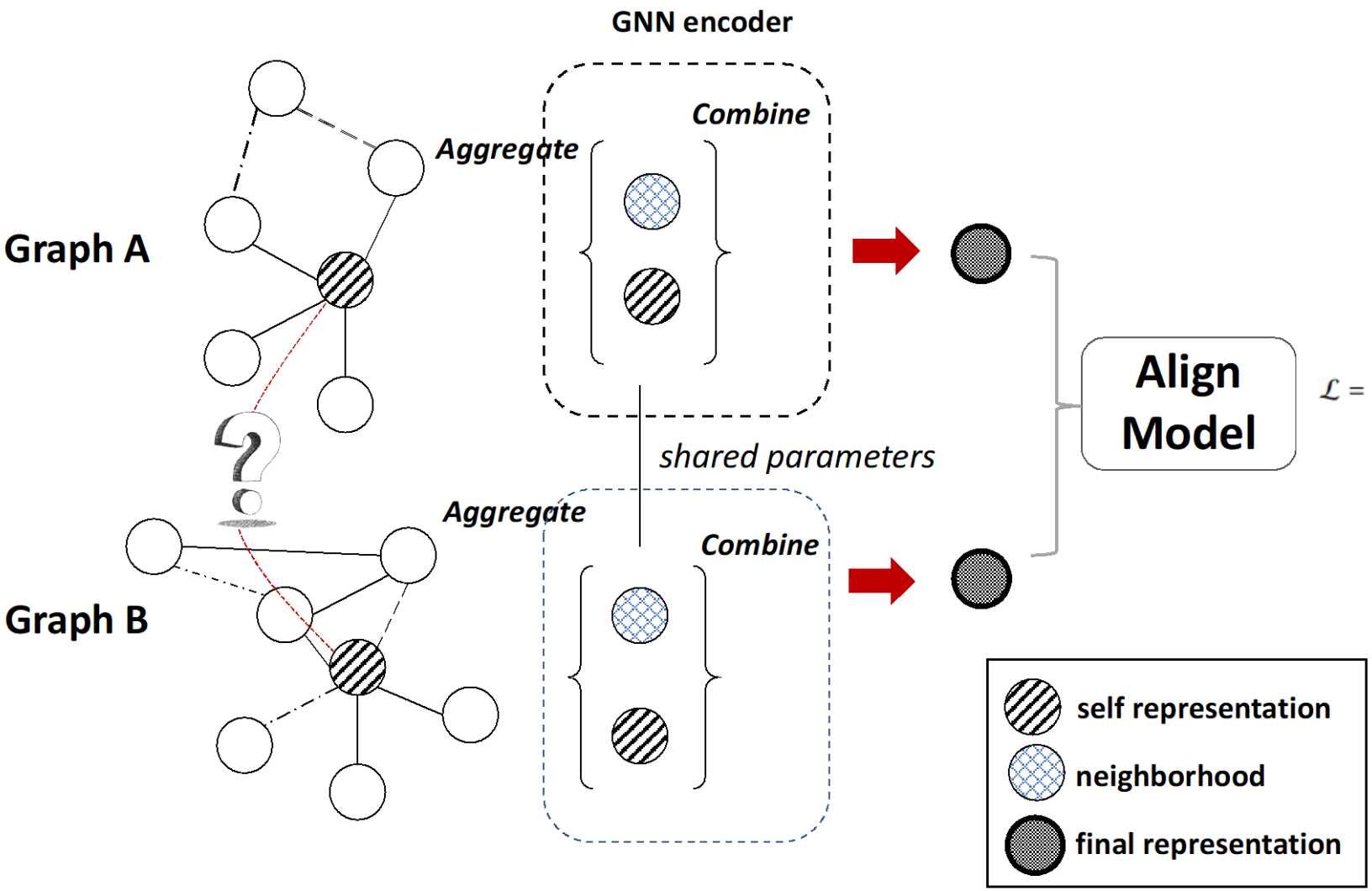
Hidden represention

Relational projection

Input feature

R-GCN (M Schlichtkrull et al., 2016)

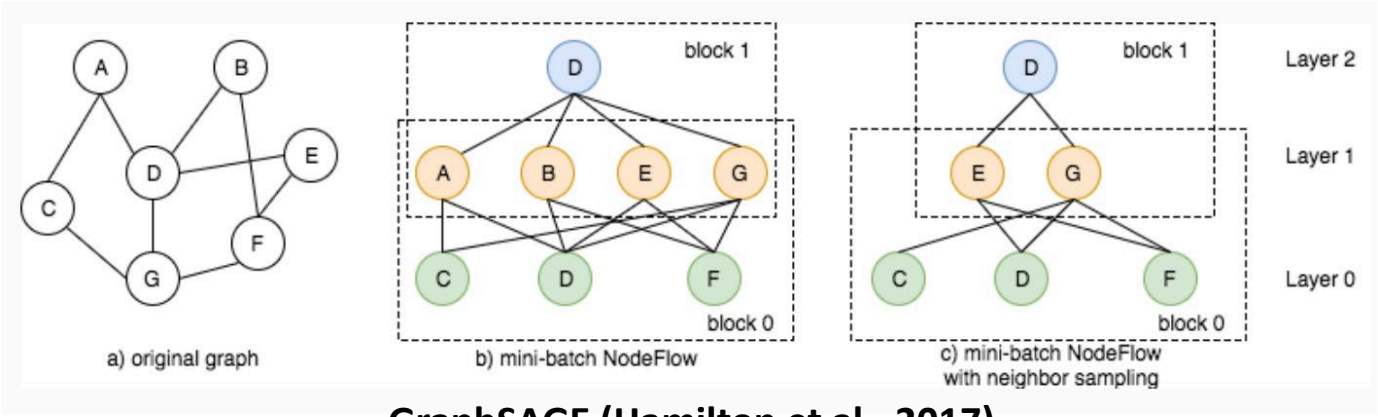
Entity Alignment Prediction with GNN



$$\mathcal{L} = \sum_{(i,i')} \sum_{(i-,i'-)} \max\left(0, d(h_i^K, h_{i'}^K) - d(h_{i-}^K, h_{i-'}^K) + \gamma\right)$$

Scalability

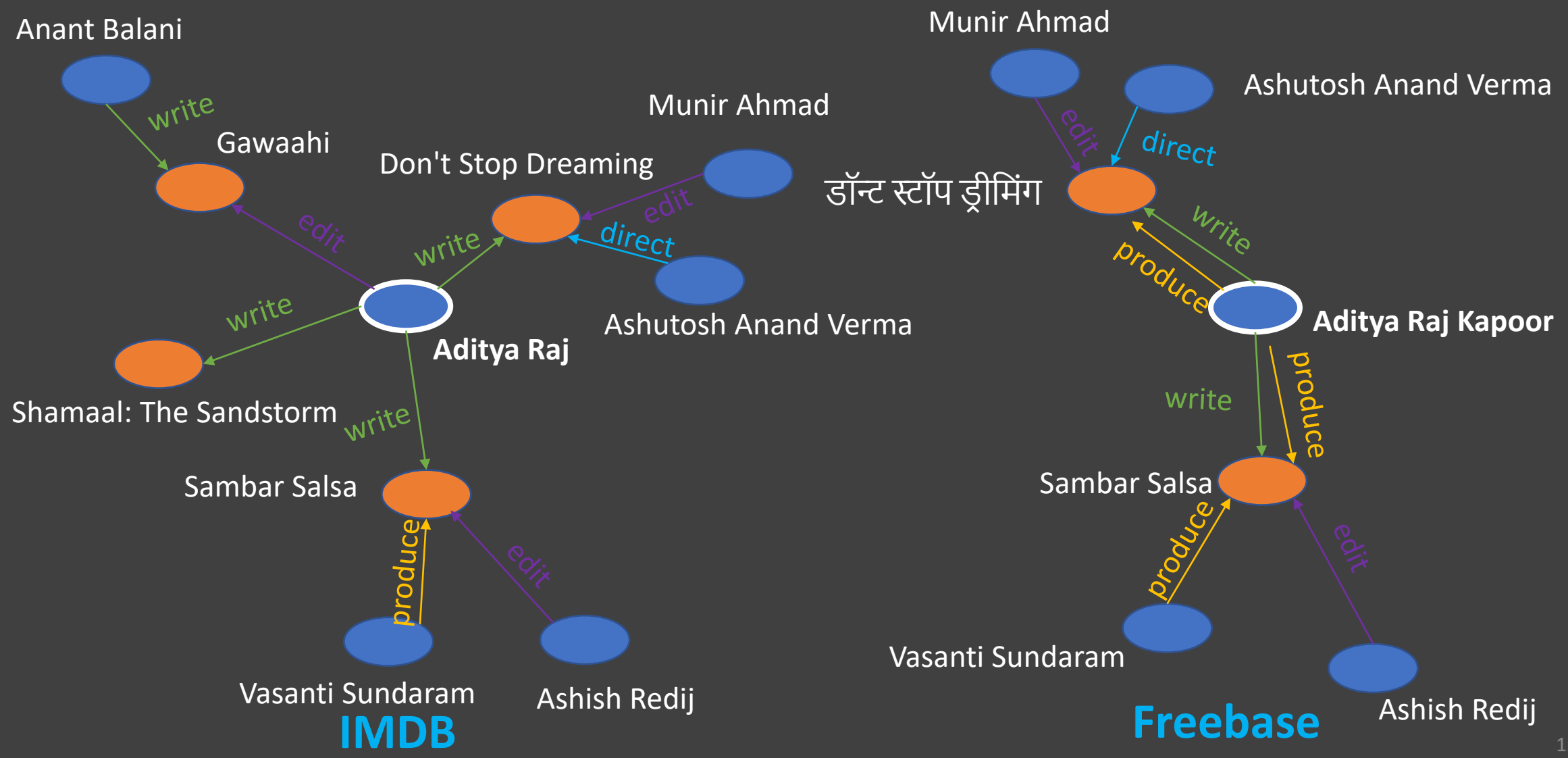
- The entity graph is huge
  - AMC(amazon music data), ~100 Million Records
  - Freebase-IMDB, >1M entities
- Traditional GNNs do not support mini-batch training
  - Sampler k-hop sub-graph and propagate the whole adjacency matrix
  - Not a generic solution for our needs(one producer will have thousands of songs, etc.



GraphSAGE (Hamilton et al., 2017)
Deep Graph Library (Wang, Minjie, et al., 2016)

So, are we done?

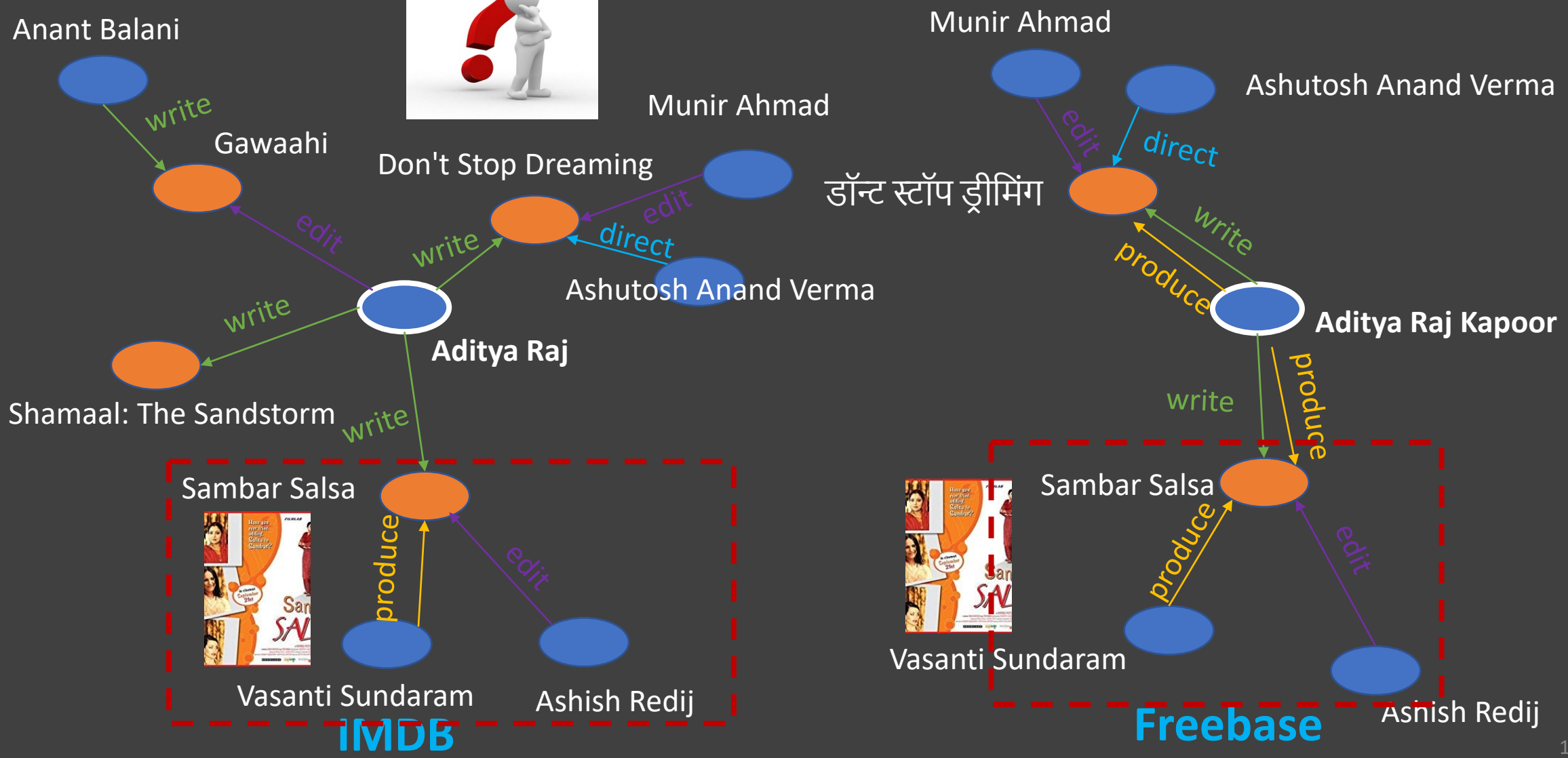
A Real Alignment Example



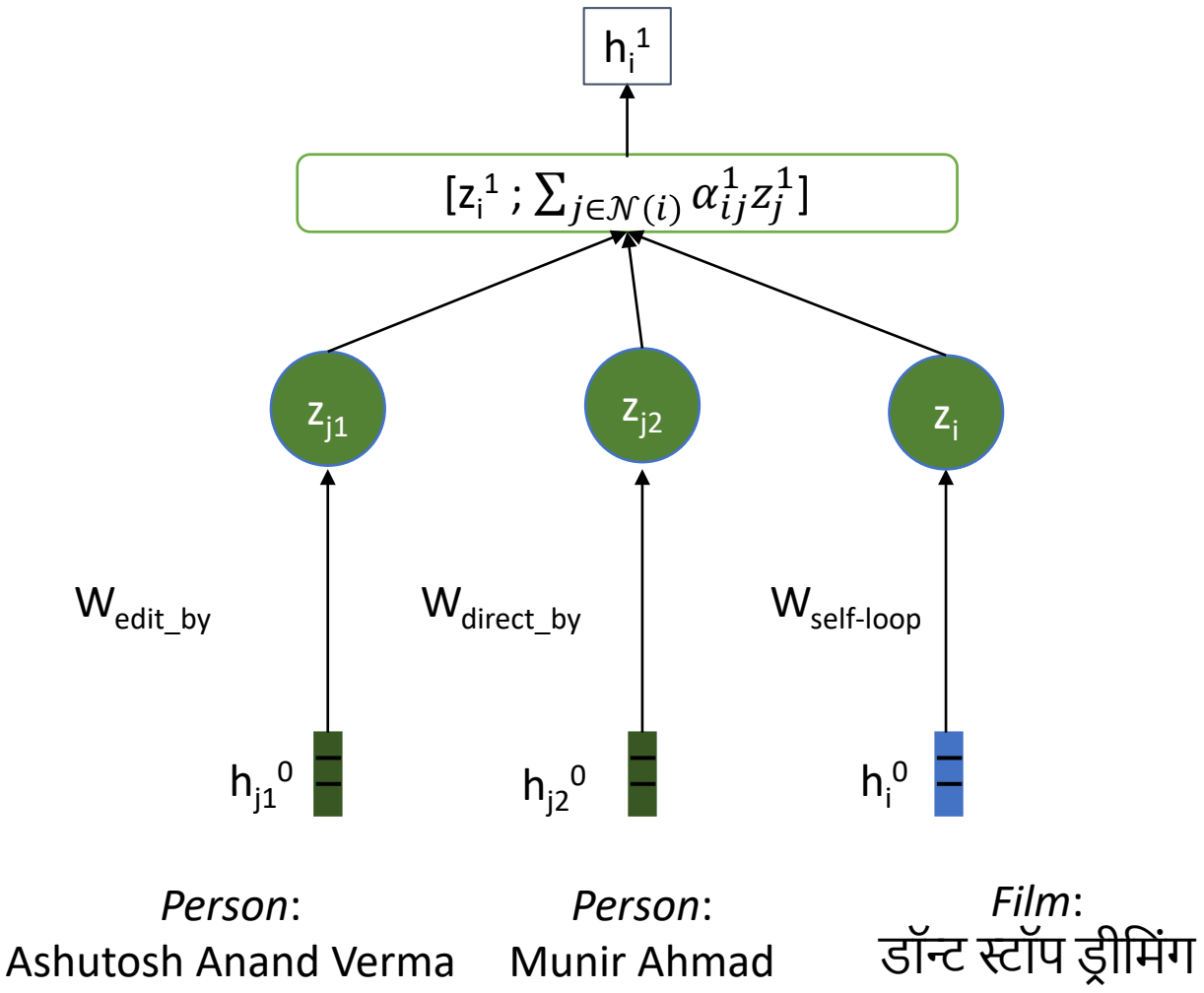
Overview

- Introduction
- Challenges
  - Attribute alignment for entities in different types
  - Aggregation function
  - Noisy neighborhood information
- Proposed Framework
- Related Work
- Experiments
- Next Steps

A Real Alignment Example



Idea I: ”concatenate”

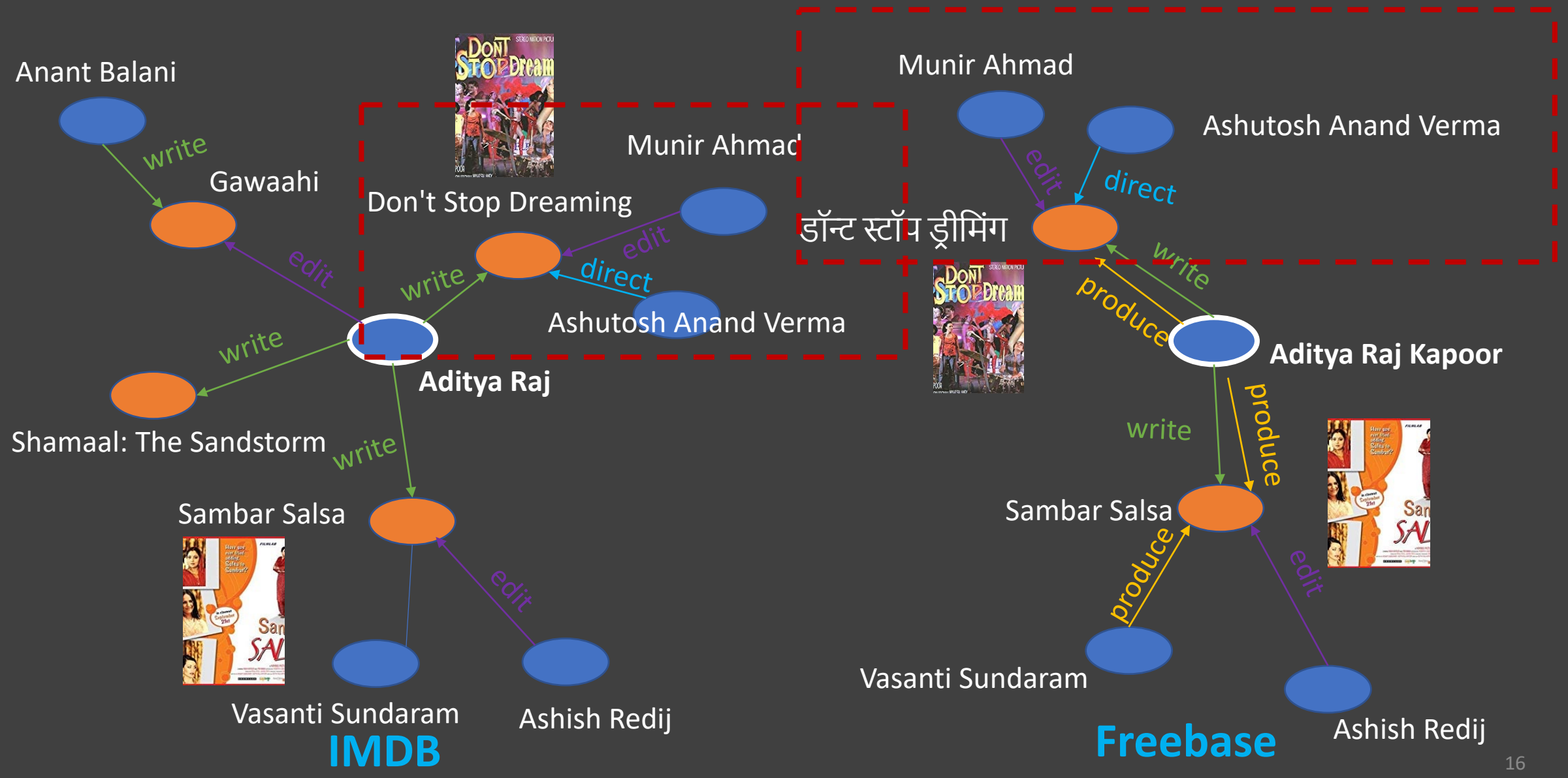


Layer 1:
Hidden represention

Relational projection

Layer 0:Input feature

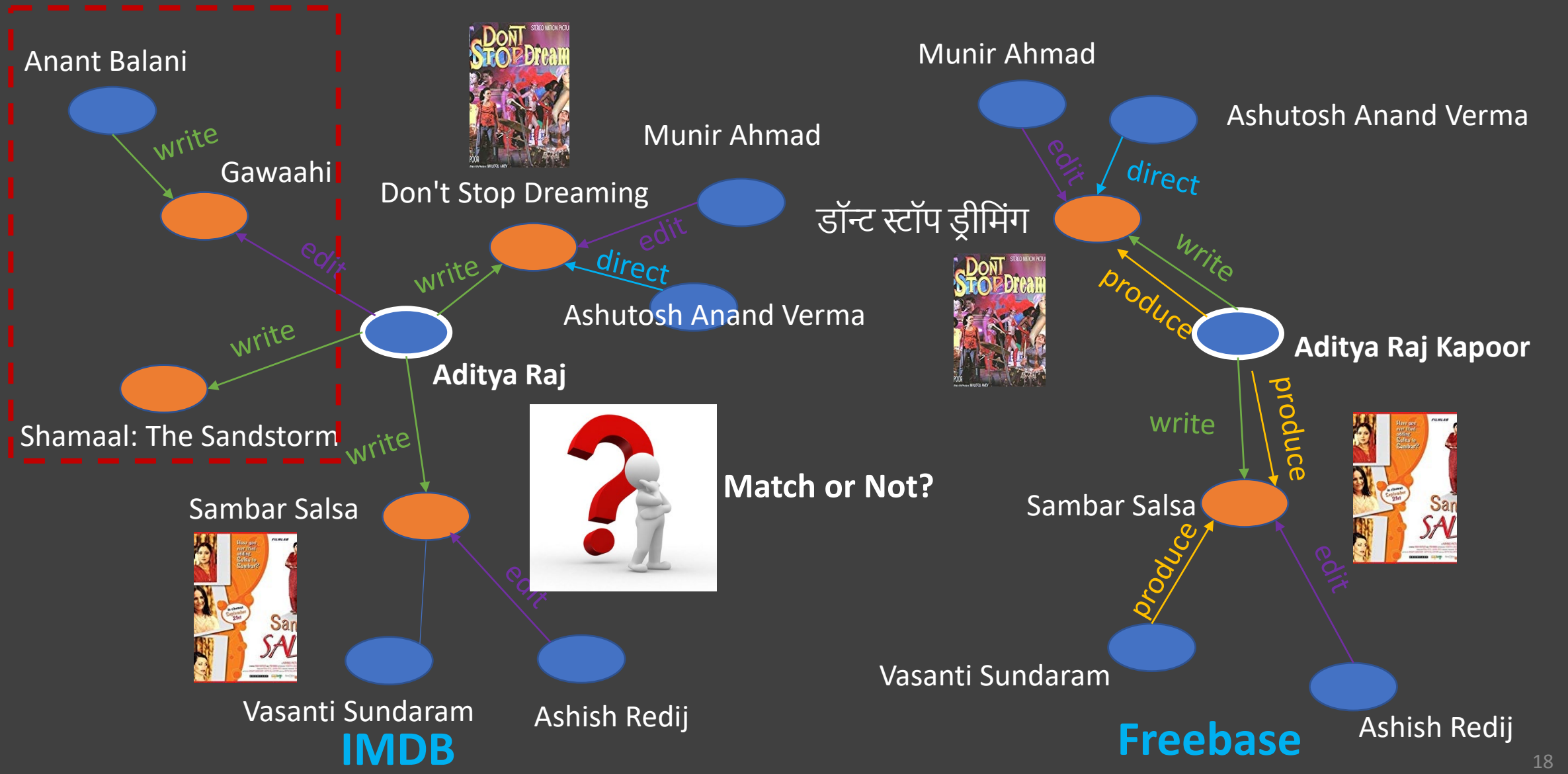
A Real Alignment Example



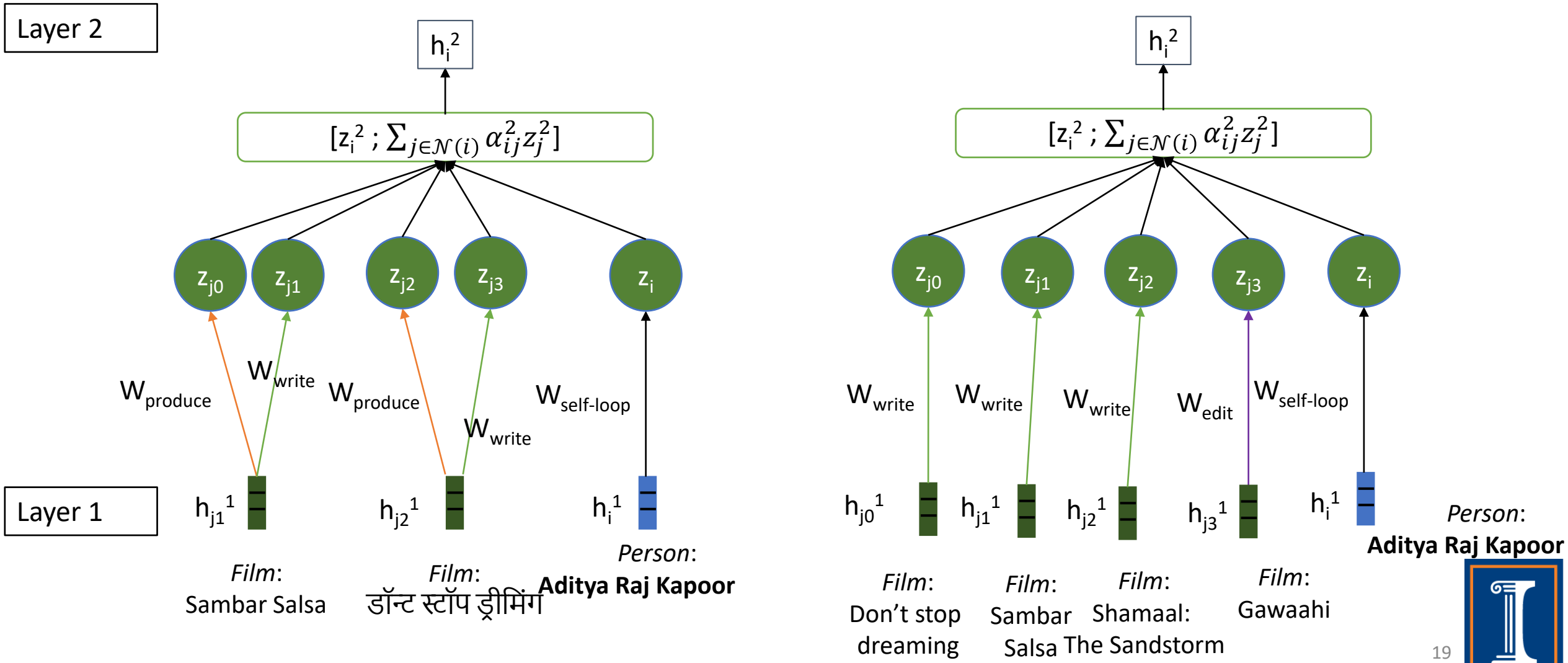
Overview

- Introduction
- Challenges
  - Attribute alignment for entities in different types
  - Aggregation function
  - Noisy neighborhood information
- Proposed Framework
- Related Work
- Experiments
- Next Steps

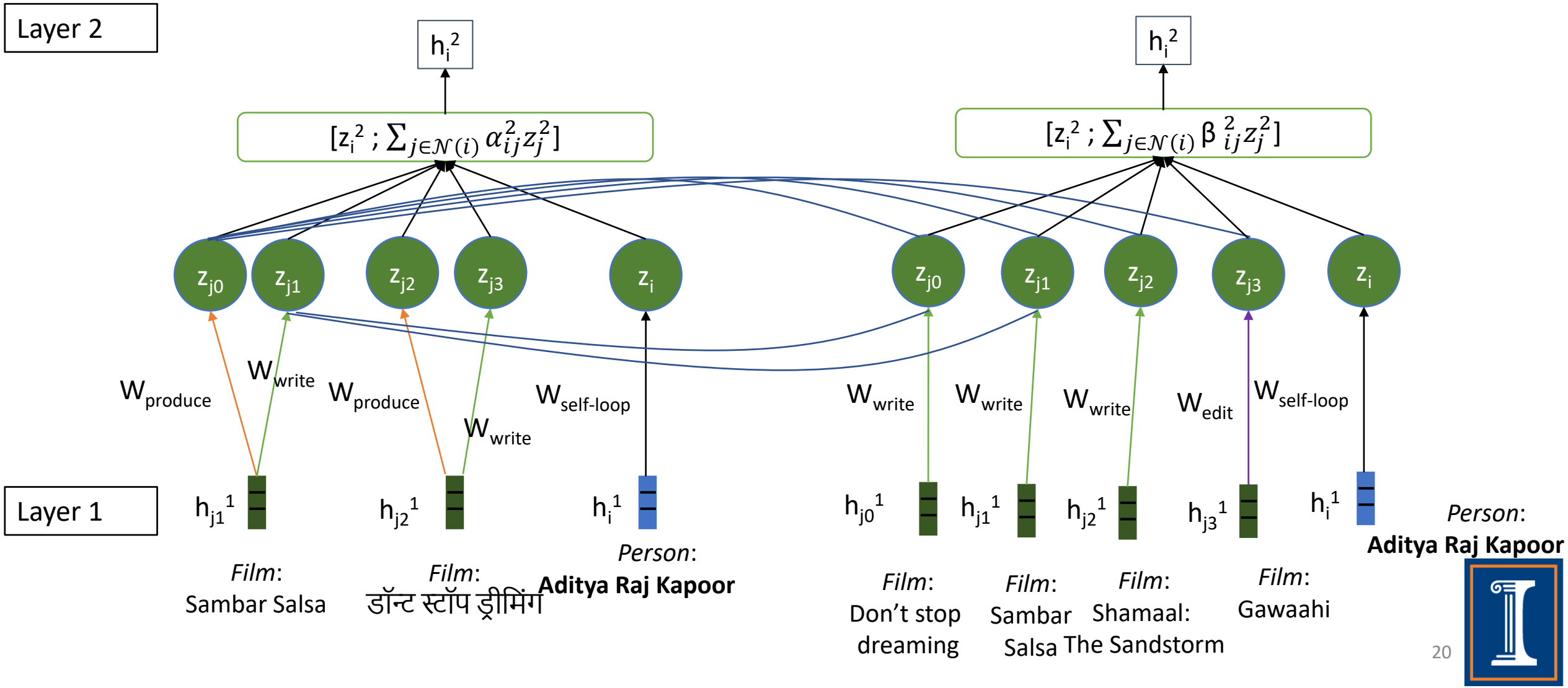
A Real Alignment Example



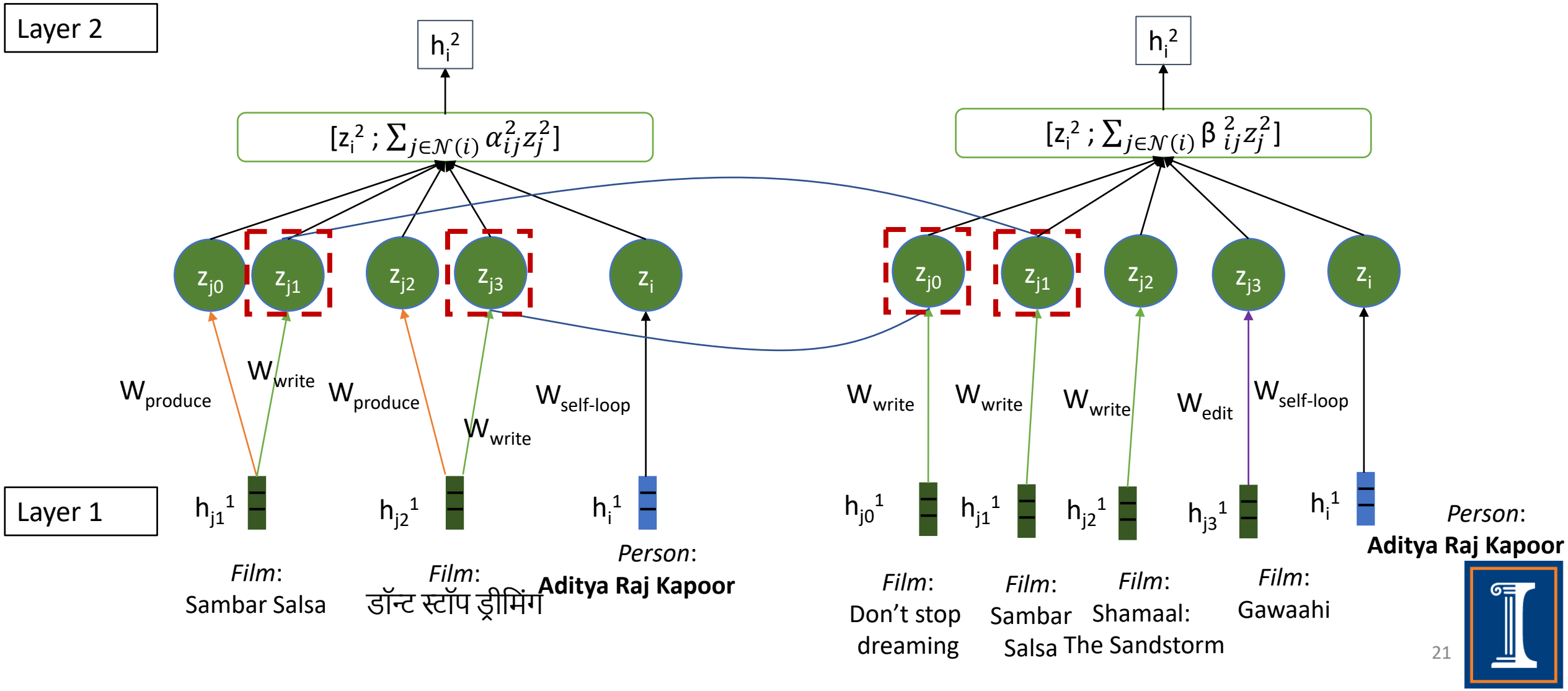
Idea II: “Cross-Attention”



Idea II: “Cross-Attention”

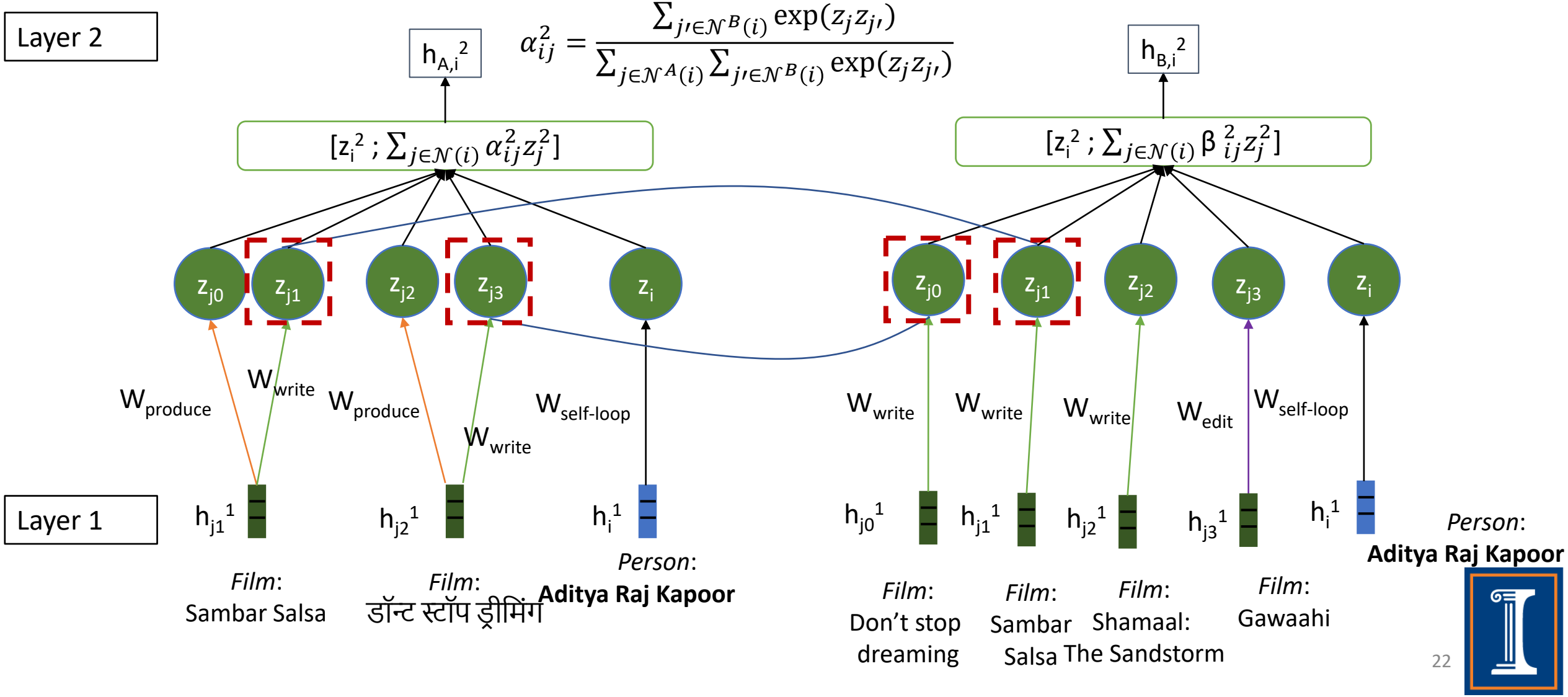


Idea II: “Cross-Attention”



Idea II: “Cross-Attention”

$$\alpha_{ij}^2 = \frac{\sum_{j' \in \mathcal{N}^B(i)} \exp(z_j z_{j'})}{\sum_{j \in \mathcal{N}^A(i)} \sum_{j' \in \mathcal{N}^B(i)} \exp(z_j z_{j'})}$$



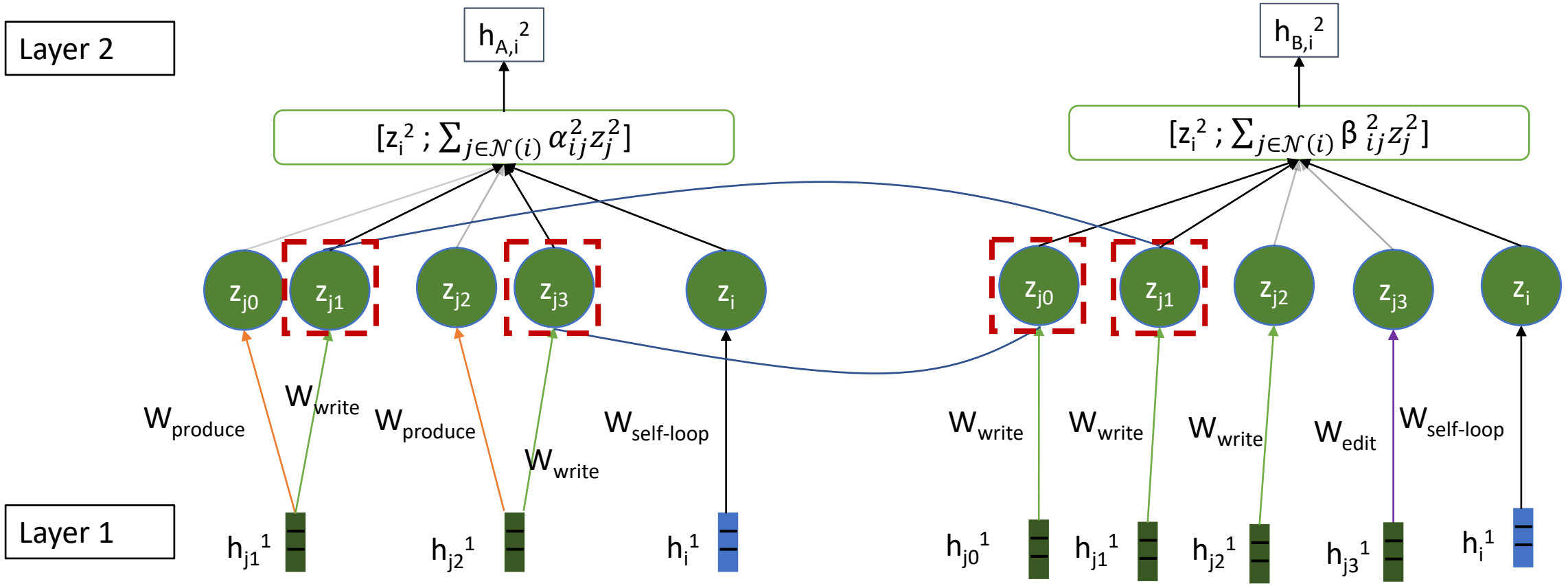
Overview

- Introduction
- Challenges
- Proposed Framework
- Related Work
- Experiments
- Next Steps

Collective GNN Layer – cross attention

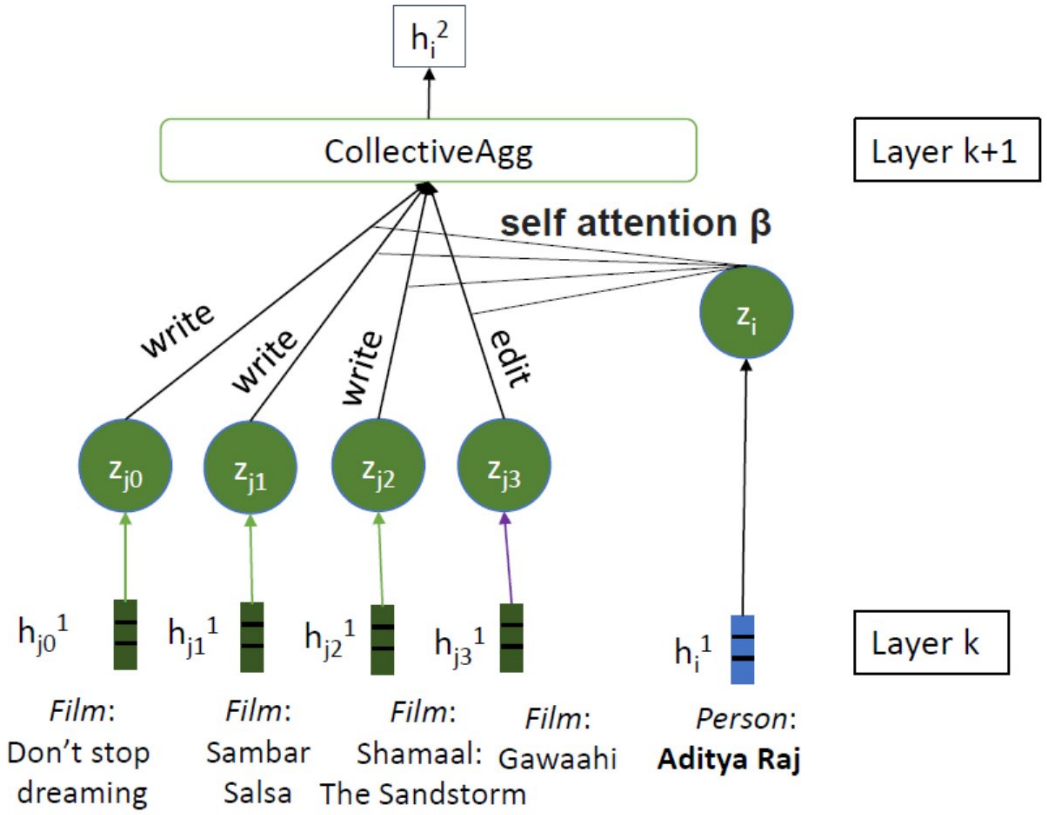
$$\alpha_{ij}^2 = \frac{\sum_{j' \in \mathcal{N}^B(i)} \exp(z_j z_{j'})}{\sum_{j \in \mathcal{N}^A(i)} \sum_{j' \in \mathcal{N}^B(i)} \exp(z_j z_{j'})}$$

$$\beta_{ij}^2 = \frac{\sum_{j' \in \mathcal{N}^A(i)} \exp(z_j z_{j'})}{\sum_{j \in \mathcal{N}^B(i)} \sum_{j' \in \mathcal{N}^A(i)} \exp(z_j z_{j'})}$$



Collective GNN Layer – self attention

$$\beta_{ij}^2 = \frac{\sum_{j' \in \mathcal{N}^A(i)} \exp(z_j z_{j'})}{\sum_{j \in \mathcal{N}^B(i)} \sum_{j' \in \mathcal{N}^A(i)} \exp(z_j z_{j'})}$$



(b) Relation-aware Self-Attention

Negative Evidence: self-attention

$$\beta_{ij} = \frac{\exp(\sigma(\vec{a}_r^T [z_i || z_j]))}{\sum_{k \in \mathcal{N}_i} \exp(\sigma(\vec{a}_r^T [z_i || z_k]))}$$

Table 2: Alignment Example for song Radioactive. Neighbor nodes are grouped by relations as described in Section 3.2. Bold font indicates the neighbor node with large cross-attention weights.

	Amazon Music	Wikipedia
Attributes		
Title Duration	Radioactive 2M19S	Radioactive 2M19S
Neighbors		
Song writer	**Wayne Sermon** **A. Grant** **Dan Reynolds** **Josh Mosser**	**Wayne Sermon** **Alexander Grant** **Dan Reynolds** **Josh Mosser**
Song producer	Alex Da Kid	
Album	Night Visions (Deluxe)	Night Visions
Main performer	**Imagine Dragons** Kendrick Lamar	**Imagine Dragons**

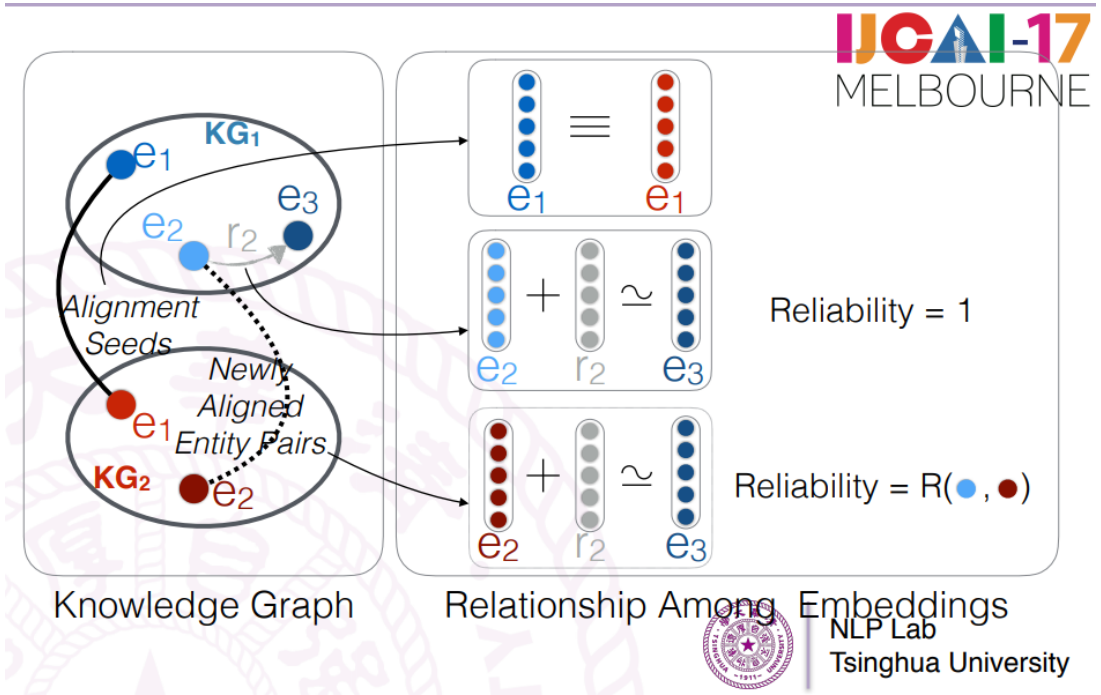
A Real Alignment Example

Overview

- Introduction
- Challenges
- Proposed Framework
- Related Work
- Experiments
- Next Steps

Related Work

- Embedding-based knowledge graph alignment
  - Transductive models that based on various knowledge graph embedding methods such as TransE, DistMult, TransH etc.
  - Large parameter space $O(V)$ when dealing with large scale knowledge graphs



[ITransE Zhu, Hao, et al. IJCAI 17']

Related Work

- Collective entity resolution
  - Decisions made on inter-connected entities are affected by each other.
  - PARIS [Suchanek, Fabian et al. VLDB 12’]

$$fun(r) = \frac{\#x : \exists y : r(x,y)}{\#x, y : r(x,y)}$$

    - Functionality = 1, indicates the right argument of the relation is unique, e.g. *isCitizenOf*
    - Inverse functionality, indicates the left argument of the relation is unique
    - Positive evidence:

$$\exists r, y : r(x,y) \wedge (\forall y' : r(x',y') \Rightarrow y \not\equiv y') \wedge fun(r) \text{ is high} \implies x \not\equiv x.$$

$$\Pr_1(x \equiv x') := 1 - \prod_{\substack{r(x,y) \\ r(x',y')}} \left(1 - fun^{-1}(r) \times \Pr(y \equiv y')\right)$$

    - Negative evidence

$$\exists r, y, y' : r(x,y) \wedge r(x',y') \wedge y \equiv y' \wedge fun^{-1}(r) \text{ is high} \implies x \equiv x'$$

$$\Pr_2(x \equiv x') := \prod_{r(x,y)} \left(1 - fun(r) \prod_{r(x',y')} \left(1 - \Pr(y \equiv y')\right)\right)$$

Related Work

- Graph Matching Networks [Li, Yujia, et al. ICML 19’]

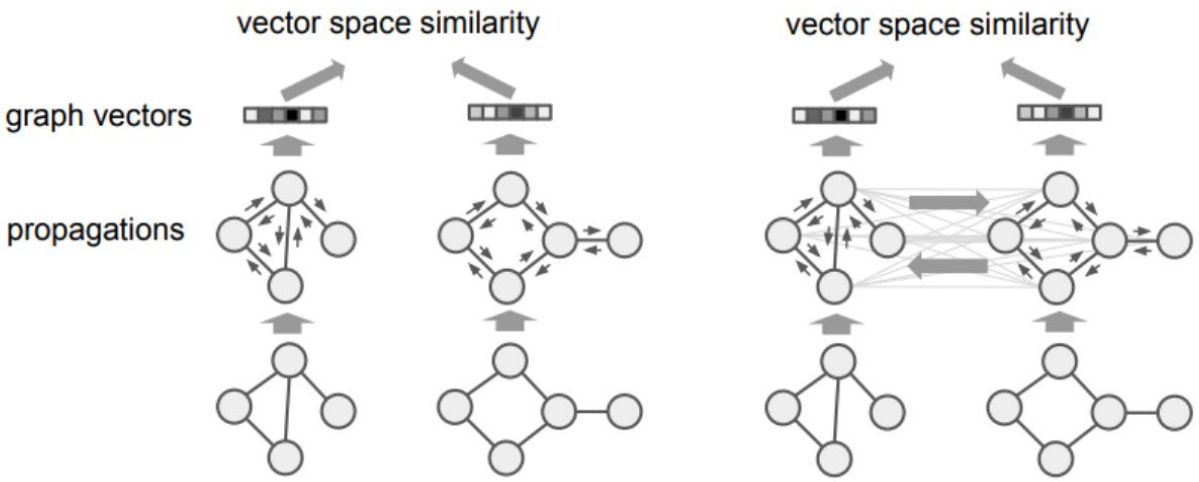


Figure 2. Illustration of the graph embedding (left) and matching models (right).

$$\mathbf{m}_{j\to i} = f_{\text{message}}(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}, \mathbf{e}_{ij}), \forall (i,j) \in E_1 \cup E_2$$

$$\boldsymbol{\mu}_{j\to i} = f_{\text{match}}(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}),$$

$$\forall i \in V_1, j \in V_2, \text{or } i \in V_2, j \in V_1$$

$$\mathbf{h}_i^{(t+1)} = f_{\text{node}}\left(\mathbf{h}_i^{(t)}, \sum_j \mathbf{m}_{j\to i}, \sum_{j'} \boldsymbol{\mu}_{j'\to i}\right)$$

$$\mathbf{h}_{G_1} = f_G(\{\mathbf{h}_i^{(T)}\}_{i\in V_1})$$

$$\mathbf{h}_{G_2} = f_G(\{\mathbf{h}_i^{(T)}\}_{i\in V_2})$$

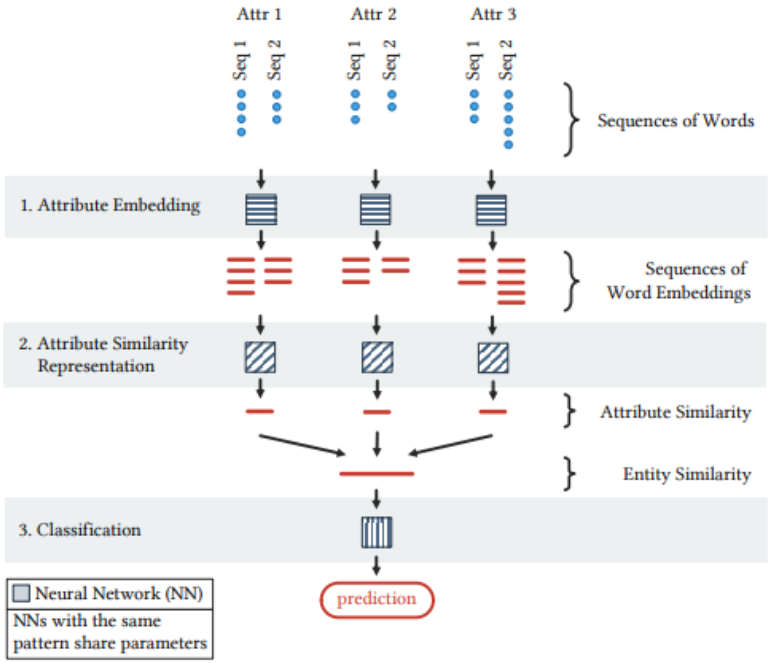
$$s = f_s(\mathbf{h}_{G_1}, \mathbf{h}_{G_2}).$$

Related Work

- Entity Matching in relational database

1. Load tables A and B into Magellan. Downsample if necessary.
2. Perform blocking on the tables to obtain a set of candidate tuple pairs C.
3. Take a random sample S from C and label pairs in S as matched / non-matched.
4. Create a set of features then convert S into a set of feature vectors H. Split H into a development set I and an evaluation set J.
5. Repeat until out of debugging ideas or out of time:
   (a) Perform cross validation on I to select the best matcher. Let this matcher be X.
   (b) Debug X using I. This may change the matcher X, the data, labels, and the set of features, thus changing I and J.
6. Let Y be the best matcher obtained in Step 5. Train Y on I, then apply to J and report the matching accuracy on J.

[Magellan Konda Pradap, et al. VLDB 16']



[DeepMatcher Mudgal, Sidharth, et al.SIGMOD 18']

Overview

- Introduction
- Challenges
- Proposed Framework
- Experiments
- Next Steps

Model Setting and Evaluation

- Recall@Precision=0.95, PRAUC, F1
- GNN model training: sample #negatives=20 for each positive pair
- Movie Vertical:
  - Use name edit distance as blocking key to create testing dataset

Table 4: Movie Dataset

Dataset	# Films	# People	# Characters	# Genres	# Train/Test
Freebase	273,526	314,869	859,289	599	53,405/53,405
IMDB	423,118	600,909	211,895	28	

- Music Vertical:
  - Sub-sample 1M music graph and a corresponding Wikipedia music graph
  - Training pairs are generated from existing system with high precision

Table 5: Music Dataset

Dataset	# Songs	# Albums	# Artists	# Train/Test
Wikipedia	104,179	188,602	71,409	57,062/23,485
Amazon-Music	999,900	200,911	201,550	

Data Schema

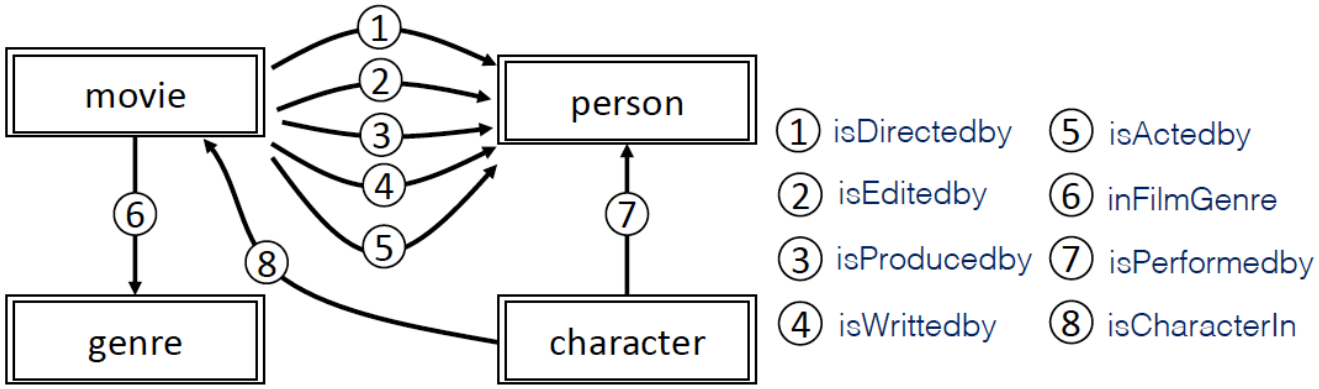


Figure 4: The schema of the Movie Graph

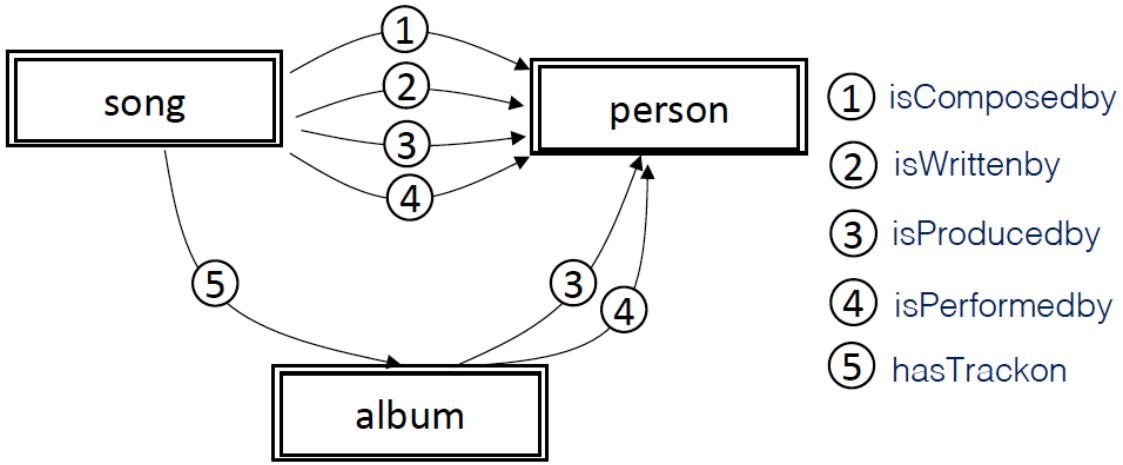


Figure 5: The schema of the Music Graph

Baselines

1. Entity feature + feed forward neural networks
2. GraphSage embedding + feed forward neural networks
3. Magellan
4. DeepMatcher
5. PARIS (collective)

Entity Matching

6. GCN + Alignment Loss
7. GraphSage + Alignment Loss
8. GAT + Alignment Loss
9. RGCN + Alignment Loss
10. R-GraphSage + Alignment Loss

GNN Variants

11. CG-MuAlign(proposed model)

Designed Experiments

- Entity alignment on labeled types
- Performance v.s. number of supervisions
- Entity alignment on unlabeled type with limited supervision

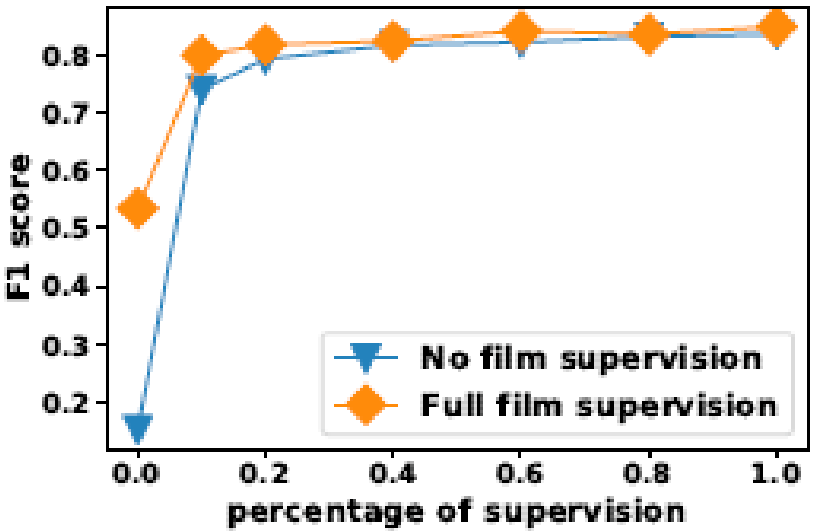
Entity alignment on labeled types

Table 6: Alignment Result on labeled types for inductive setting. For simplicity, transductive only methods are not included in this table. We report the standard deviation by 5-runs of each method except DeepMatcher, which takes long time for one run.

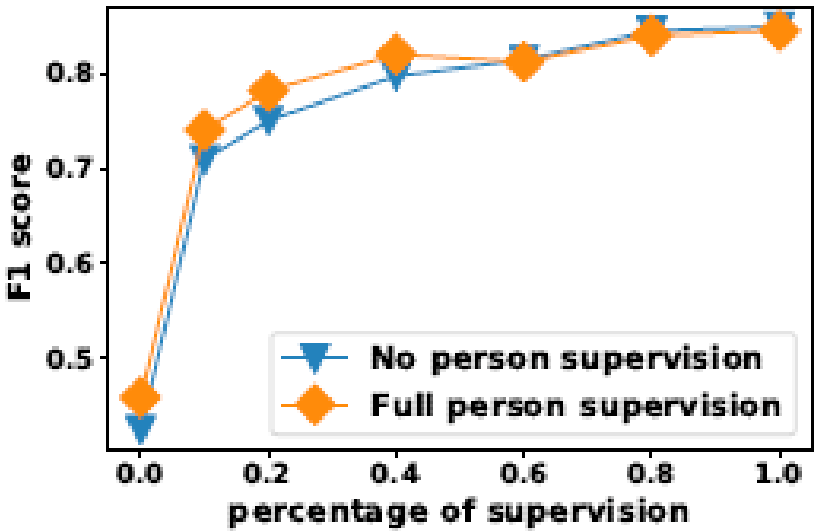
	Method	Movie Dataset				Music Dataset		
		Rec@Prec=.95	PRAUC	F1	hit@1	Rec@Prec=.95	PRAUC	F1
	Feature+NN	0	0.3672 ± 0.053	0.6380 ± 0.000	0.7197 ± 0.001	0.0025 ± 0.002	0.7251 ± 0.027	0.6795 ± 0.009
	GraphSage+NN	0.0155 ± 0.001	0.3229 ± 0.003	0.3557 ± 0.001	0.4503 ± 0.003	0.0002 ± 0.000	0.2468 ± 0.018	0.3134 ± 0.012
	Magellan	0.4387 ± 0.000	0.7067 ± 0.000	0.6945 ± 0.000	0.7974 ± 0.000	0.1071 ± 0.000	0.7461 ± 0.000	0.6760 ± 0.000
	DeepMatcher	0	0.5829 ± 0.000	0.7549 ± 0.000	0.8468 ± 0.000	0	0.1748 ± 0.000	0.3559 ± 0.000
	PARIS	0.5840 ± 0.000	0.7759 ± 0.000	0.7661 ± 0.000	0.7725 ± 0.000	0.2333	0.4175 ± 0.000	0.4640 ± 0.000
GNN variants	GCN	0.0098 ± 0.001	0.2831 ± 0.006	0.3313 ± 0.004	0.4896 ± 0.003	0.0020 ± 0.002	0.3829 ± 0.009	0.4190 ± 0.003
GNN variants	GraphSage	0.1900 ± 0.007	0.5589 ± 0.004	0.5251 ± 0.003	0.6605 ± 0.009	0.2868 ± 0.029	0.8252 ± 0.003	0.7637 ± 0.001
GNN variants	GAT	0.0147 ± 0.002	0.3448 ± 0.006	0.3793 ± 0.004	0.5483 ± 0.003	0.0004 ± 0.001	0.4485 ± 0.014	0.4819 ± 0.007
GNN variants	RGCN	0.0106 ± 0.002	0.4247 ± 0.003	0.4435 ± 0.001	0.5450 ± 0.002	0.0025 ± 0.004	0.4419 ± 0.024	0.4625 ± 0.020
GNN variants	R-GraphSage	0.2829 ± 0.009	0.6573 ± 0.003	0.6110 ± 0.004	0.7125 ± 0.003	0.4081 ± 0.029	0.8335 ± 0.004	0.7646 ± 0.003
	CG-MuAlign	**0.6010 ± 0.004**	**0.8548 ± 0.004**	**0.8050 ± 0.006**	**0.8869 ± 0.002**	**0.4437 ± 0.023**	**0.8400 ± 0.008**	**0.7762 ± 0.004**

Performance v.s. number of supervisions

Is the model sensitive to the amount of training data?



(a) Test on Person



(b) Test on Film

Entity alignment on unlabeled type with limited supervision

- Few-shot setting
  - model trained on type A and fine-tuned on type B with 2000 samples

Table 7: Alignment Result on unlabeled types for few-shot setting. We mark the best and second-best result. Column person stands for unlabeled type in the evaluation.

	Method	Person		Film	
		PRAUC	F1	PRAUC	F1
	Node features	0.8285	**0.8563**	0.4231	0.4780
GNN variants	GCN	0.3492	0.4659	0.2589	0.3223
	GraphSage	0.5495	0.6069	0.4269	0.4158
	GAT	0.3518	0.3791	0.4926	0.4818
	RGCN	0.3130	0.3518	0.4288	0.4369
	R-GraphSage	0.8065	0.7582	0.5008	0.4705
	Few-shot	0.8403	0.8033	0.8505	0.8136
	Fully-supervised	**0.8543**	0.8214	**0.9101**	**0.8794**

What's the trade-off between collective and scalability?

THEOREM 3.1. *If G and G' have the same number of nodes, i.e. $|\mathcal{V}_1| = |\mathcal{V}_2|$ and there exists a injective function $\mathcal{F} : \mathcal{V}_1 \to \mathcal{V}_2$. Let K denote the order of the neighborhood, $|\mathcal{E}|$ is the total number of edges in the underlying graph G_u, the expected Collective Power decays geometrically as K increases.*

$$\mathbb{E}_{(v,\mathcal{F}(v))\sim G_1} CP(K) \leq |\mathcal{E}| \cdot p^{\frac{K}{2}} q^{\frac{K}{2}}$$

PROOF. According to the definition of p and q. Let p_i and q_i be the actual observed ratio for node v_i and $\mathcal{F}(v_i)$ in graph G and G', we have,

$$p = \frac{\sum_{i=1}^{|\mathcal{V}_1|} |\mathcal{N}_i| \cdot p_i}{|\mathcal{E}|}, q = \frac{\sum_{i=1}^{|\mathcal{V}_2|} |\mathcal{N}_i| \cdot q_i}{|\mathcal{E}|}$$

For a specific node i, the expected number of same neighborhood from a uniform distribution in two graphs is $|\mathcal{N}_e| p_i q_i$. Thus, when $K = 1$,

$$\mathbb{E}_{(v,\mathcal{F}(v))\sim G_1} CP(1) = \sum_i |\mathcal{N}_e| p_i q_i \tag{6}$$

$$\leq \sqrt{\sum_i (\sqrt{\mathcal{N}_e} p_i)^2 \sum_i (\sqrt{\mathcal{N}_e} q_i)^2} \tag{7}$$

$$\leq \sqrt{\sum_i \mathcal{N}_e p_i \sum_i \mathcal{N}_e q_i} = |\mathcal{E}| \cdot \sqrt{pq} \tag{8}$$

Recursively, we repeat the same calculation on shared neighbor nodes in previous step, that is, $\mathbb{E}[CP(K+1)] = \mathbb{E}[CP(K)] \cdot \sqrt{pq}$ □

$$h_i^k = \begin{cases} \text{COLLECTIVEAGG}\left(\{h_j^{k-1}, j \in \mathcal{N}_i \cup \{i\}\}\right), & k = K-1 \\ \text{AVERAGEAGG}\left(\{h_j^{k-1}, j \in \mathcal{N}_i \cup \{i\}\}\right) & k < K-1 \end{cases} \tag{9}$$

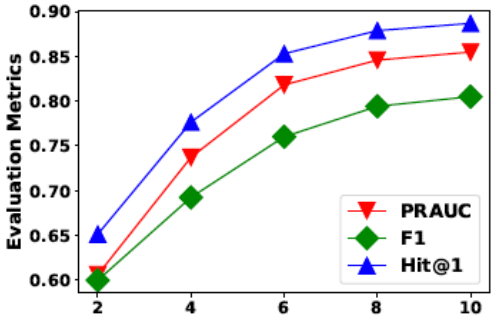
where the AVERAGEAGG replaces the $\alpha_{ij}\beta_{ij}$ in Equation 4 as $\frac{1}{|\mathcal{N}_i|}$.

We just need to have **one** collective layer to obtain the most collective power.

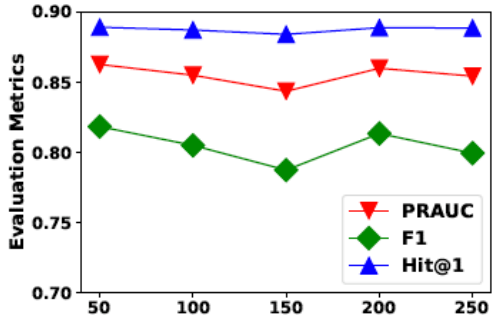
Space Complexity: $O(B \cdot N^2)$, N is the number of sampler neighborhoods.

Time Complexity: $O(S \cdot N^2)$, S is the # training pairs

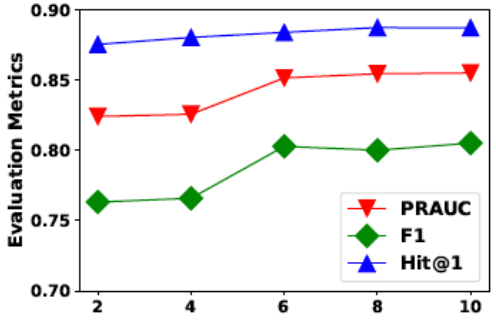
Parameter study & Running time



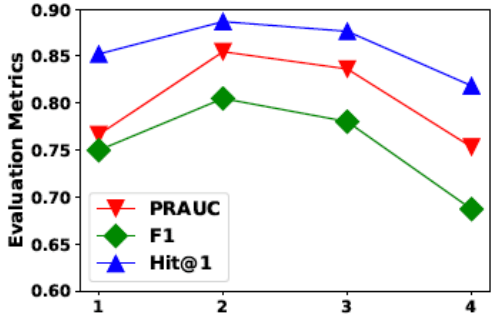
(a) Varying #sampled neighbors



(b) Varying #hidden dimensions



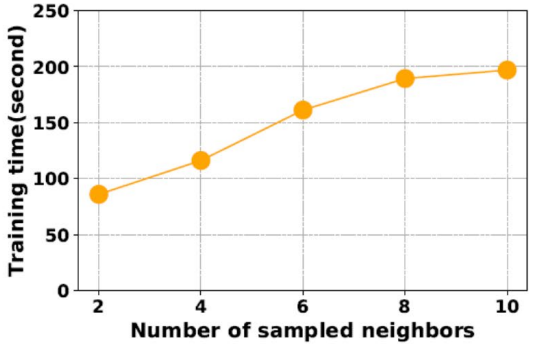
(c) Varying #negative samples



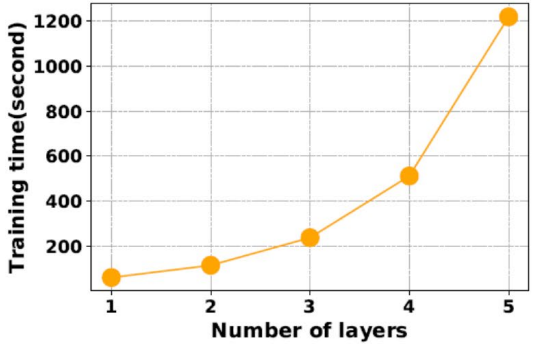
(d) Varying #layers

Table 9: Efficiency study of three different methods

Method	Training Time	# Parameters	Averaged-F1
Magellan	7m13s	9,300	0.6641
DeepMatcher	13h40m	17,757,810	0.6014
CG-MuAlign	30m47s	175,134	0.7925



(a) Varying #sampled neighbors



(b) Varying #layers

Future Work

- Multi-class collective learning on network structured data
  - Different node types can have different label space
  - Measure the cross-type label correlations as explainable knowledge
  - Relieve the label scarcity on multi-task setting
  - Scenario 1: different tasks share overlapped label space
  - Scenario 2: different tasks share disjoint label space
- Source code: https://github.com/GentleZhu/CG-MuAlign
- Paper: https://gentlezhu.github.io/files/CollectiveAlignment.pdf
- Slides:

Discussion & QA

Thank you!