



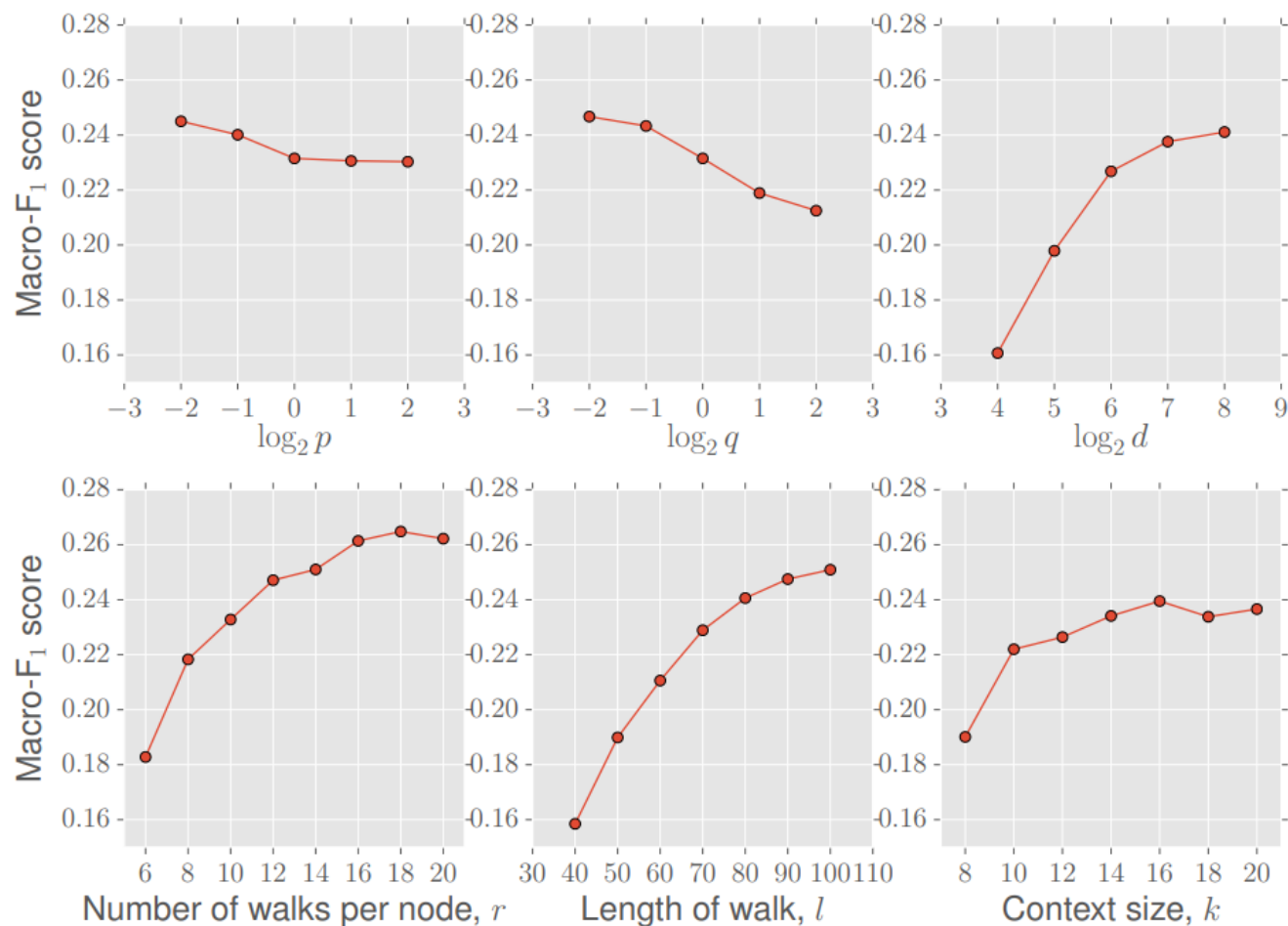
IIC-3641 Aprendizaje Automático basado en Grafos

<https://github.com/marcelomendoza/IIC3641>

- RECAPITULACIÓN -

En Node2vec existe una dependencia de los hiperparámetros (AF4)

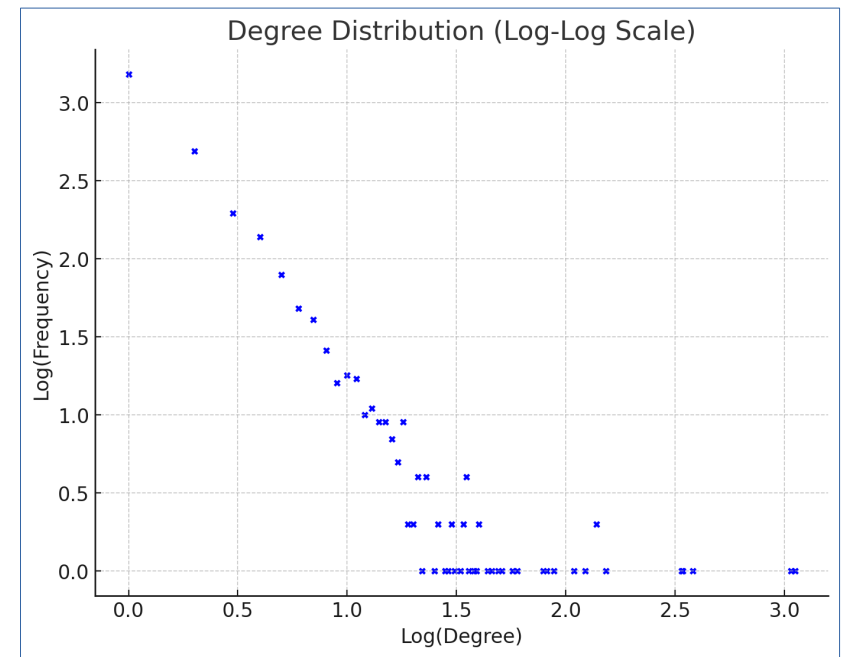
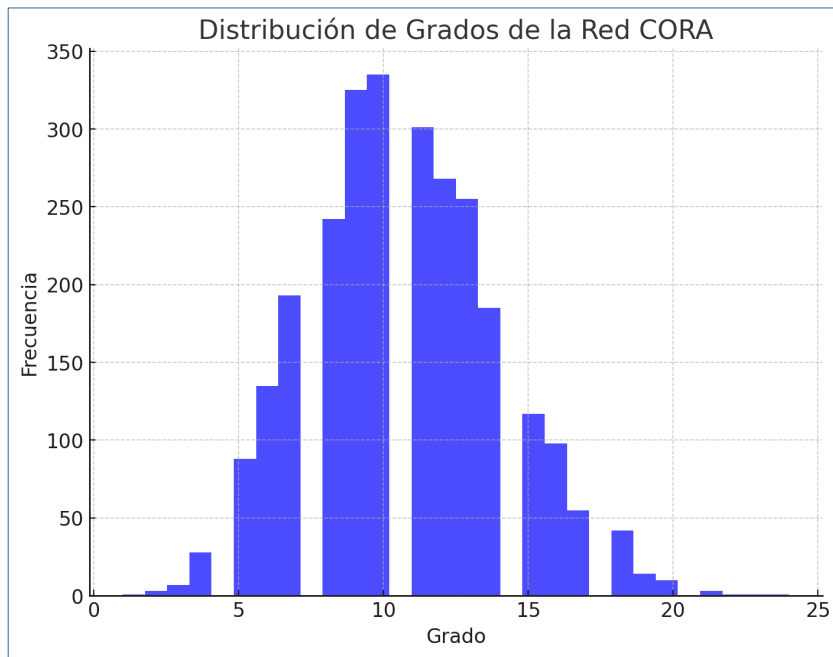
El clasificador que entrenamos sobre los nodos de CORA en la AF4 tenía dependencia de los hiperparámetros. En específico medimos dependencia de walk length y q , pero hay más hiperparámetros:



Importante: Los embeddings aprendidos desde la estructura de random walks son útiles para entrenar clasificadores a nivel de nodos.

¿Qué es CORA?

- Dirigida: Representa las citas entre los artículos. Si el artículo A cita al artículo B, entonces existe un enlace dirigido de A a B.
- No ponderada: Los enlaces generalmente no tienen pesos asociados, solo indican la existencia de una cita.
- Número de nodos: Alrededor de 2,708 artículos.
- Número de enlaces: Aproximadamente 5,278 citas (depende de la versión).



¿Tendrá la propiedad scale-free? Verifíquelo.

¿Qué es CORA?

CORA es un tipo de red heterogénea, ya que los nodos tienen atributos. Es decir, CORA tiene varios tipos de nodos, dependiendo de a cuál disciplina pertenece el paper que representa.



Case_Based
Genetic_Algorithms
Neural_Networks
Probabilistic_Methods
Reinforcement_Learning
Rule_Learning
Theory

Grafo heterogéneo: Un grafo heterogéneo G consiste de un set de nodos V y un conjunto de enlaces E , donde cada nodo y cada enlace tienen un tipo asociado. Sea T_n el conjunto de tipos de nodos y T_e el conjunto de tipos de enlaces. Un grafo heterogéneo podría tener hasta dos tipos de funciones, uno para nodos y otro para enlaces, de manera que cada función determina el tipo de nodo y enlace, respectivamente, es decir:

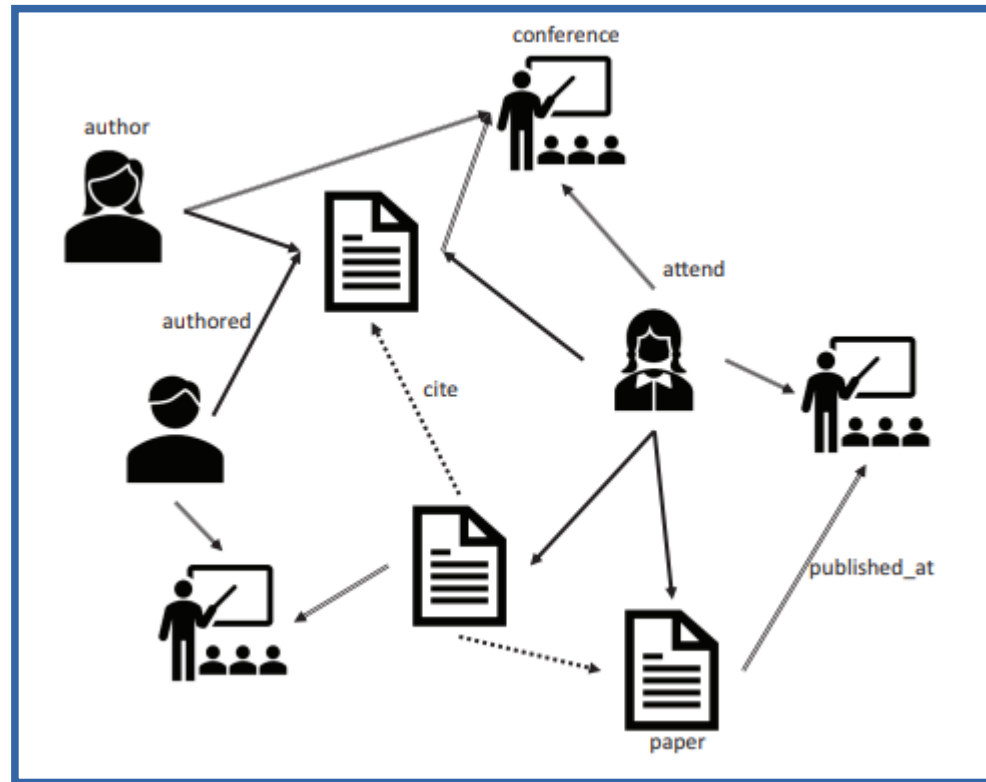
$$\phi_n : \mathcal{V} \rightarrow \mathcal{T}_n \qquad \phi_e : \mathcal{V} \rightarrow \mathcal{T}_e$$

Observe que el clasificador de la AF4 aproximó ϕ_n .

- VECTORIZACIÓN DE ENLACES -

Vectorización de enlaces

Son útiles para grafos heterogéneos, donde en particular tenemos varios tipos de enlaces.



En el grafo del ejemplo, tenemos distintos tipos de nodos y enlaces.

Un grafo que tiene varios tipos de enlaces pero un sólo tipo de nodo es un tipo especial de grafo heterogéneo conocido como **grafo multirelación**.


Vectorización de enlaces

Vamos a representar cada nodo y cada enlace con un vector de parámetros. Vampos a denotar a este vector por la variable θ .

Definimos una función de scoring:

$$f(\theta_s, \theta_r, \theta_d)$$

que produce un score para cada enlace de la forma:


$$(s, r, d) \in E$$

Origen Tipo de enlace Destino

Objetivo: Aprender una función de scoring que logre lo siguiente (contrastivo):

Maximizar $f(\theta_s, \theta_r, \theta_d)$ si $(s, r, d) \in E$

Minimizar $f(\theta_s, \theta_r, \theta_d)$ si $(s, r, d) \notin E$.

Vectorización de enlaces

Vamos a trabajar con una función de scoring basada en similitud, de manera que se cumpla la **hipótesis de clustering** en el espacio latente. Entonces:

$$f(\theta_s, \theta_r, \theta_d) = \text{sim} \left(g_{(s)}(\theta_s, \theta_r), g_{(d)}(\theta_d, \theta_r) \right)$$

Usualmente, *sim* se implementa en base del producto punto o de sim coseno.

Observemos que la función *sim* se define en base a dos factores, uno que depende del enlace y nodo origen y otro del enlace y nodo destino.

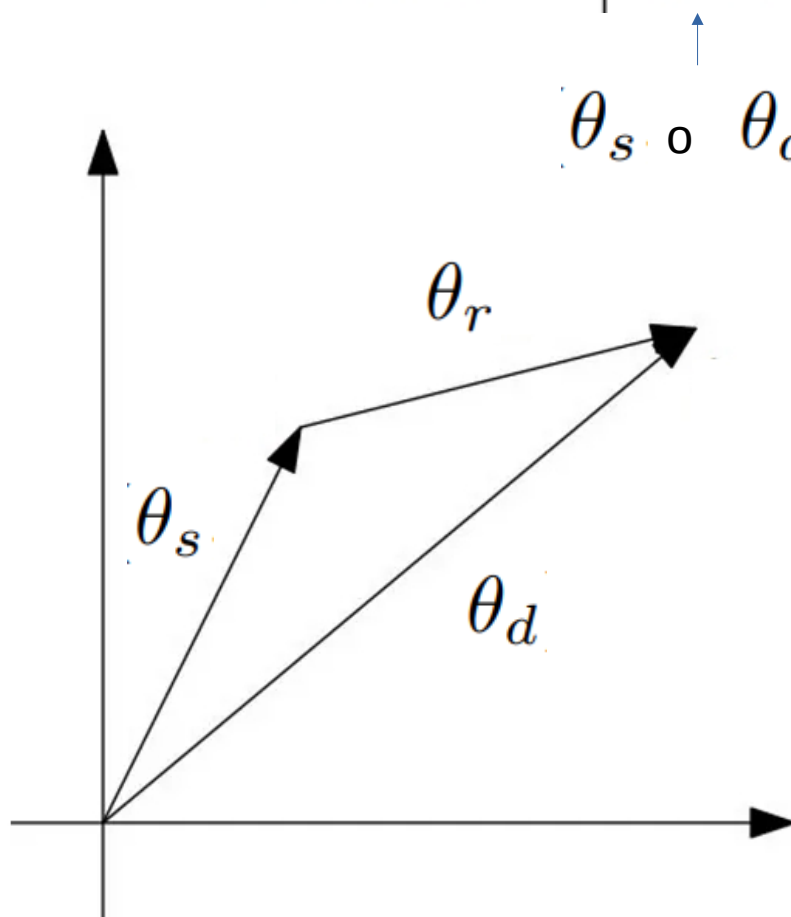
A su vez, tenemos una función *g*, la cual vincula el nodo con el enlace correspondiente. A esta función se le denomina operador relacional.

Observamos que tenemos varias alternativas para vectorizar los enlaces. De hecho, estamos vectorizando también los nodos.

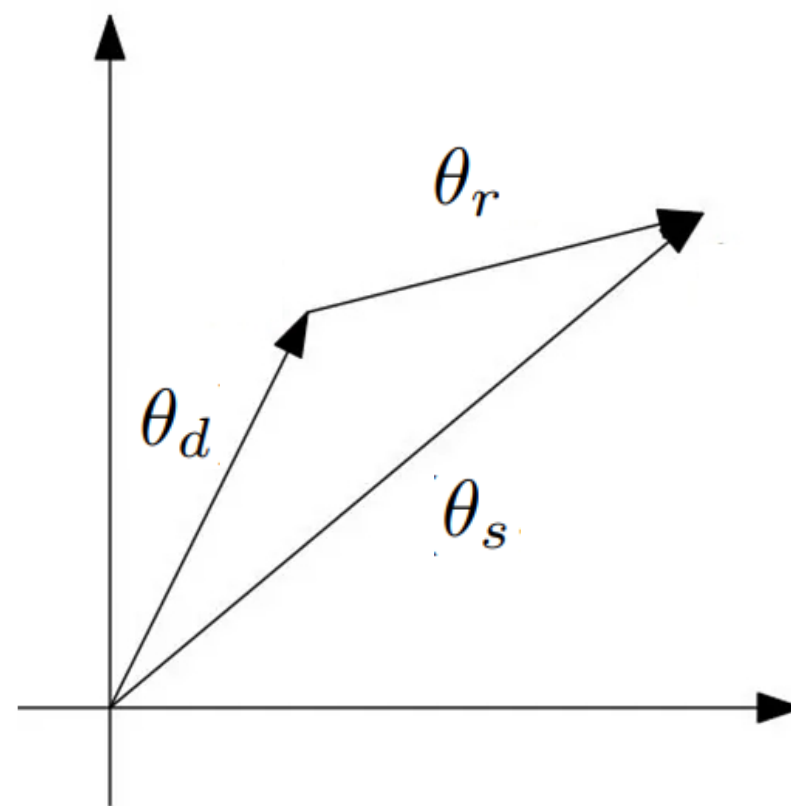
Importante: Este es un framework para grafos multirelacionales, es decir, un tipo de nodo y varios tipos de enlaces.

TransE

Model	$g(\mathbf{x}, \theta_r)$	$\text{sim}(\mathbf{a}, \mathbf{b})$
TransE	$x + \theta_r$	$\cos(a, b)$



TransE



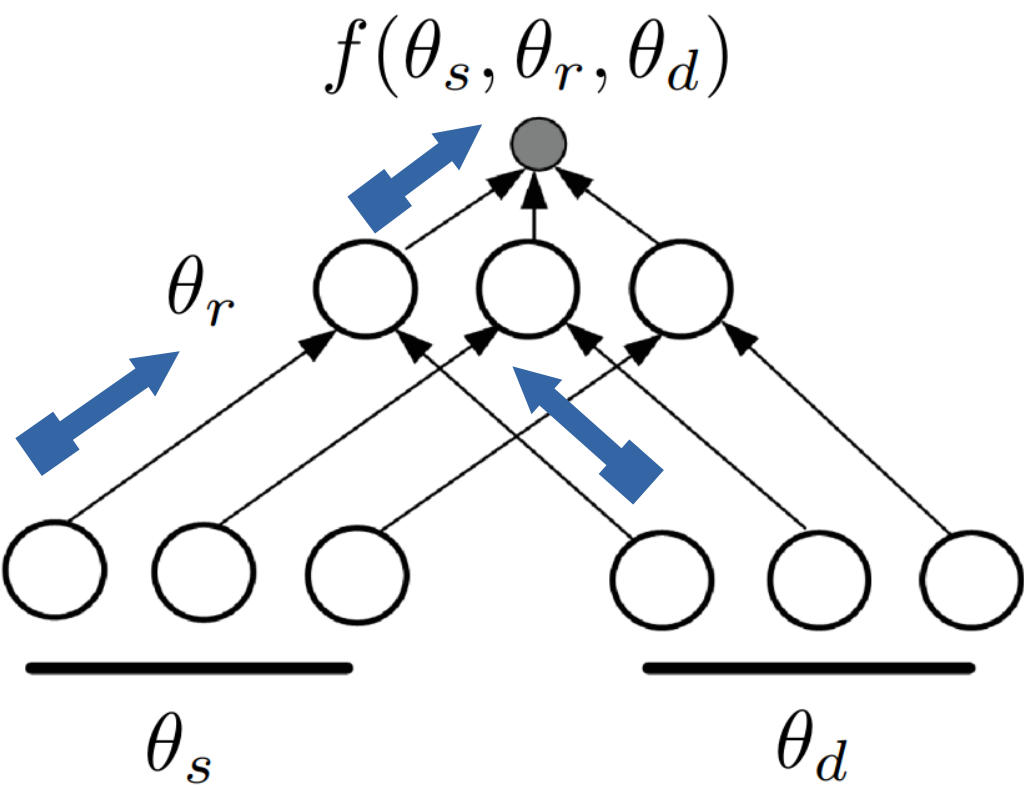
TransE

$$f(\theta_s, \theta_r, \theta_d) = \cos(\theta_s + \theta_r, \theta_d + \theta_r)$$

DistMult

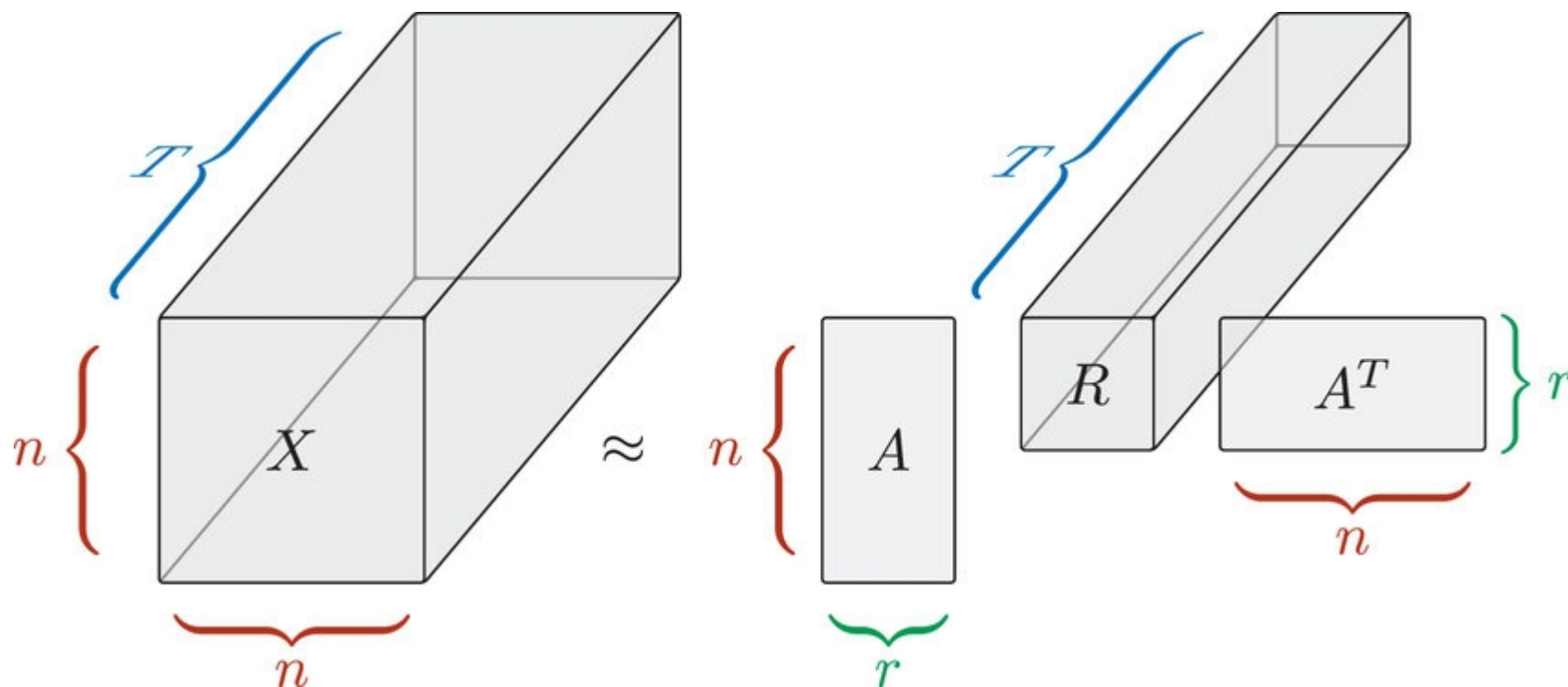
Model	$g(\mathbf{x}, \theta_r)$	$\text{sim}(\mathbf{a}, \mathbf{b})$
DistMult	$x \odot \theta_r$	$\langle \mathbf{a}, \mathbf{b} \rangle$

$$f(\theta_s, \theta_r, \theta_d) = \sum_{i=0}^{d-1} [\theta_r]_i \cdot [\theta_s]_i \cdot [\theta_d]_i$$



RESCAL

Model	$g(\mathbf{x}, \theta_r)$	$\text{sim}(\mathbf{a}, \mathbf{b})$
RESCAL	$A_r x$	$\langle \mathbf{a}, \mathbf{b} \rangle$



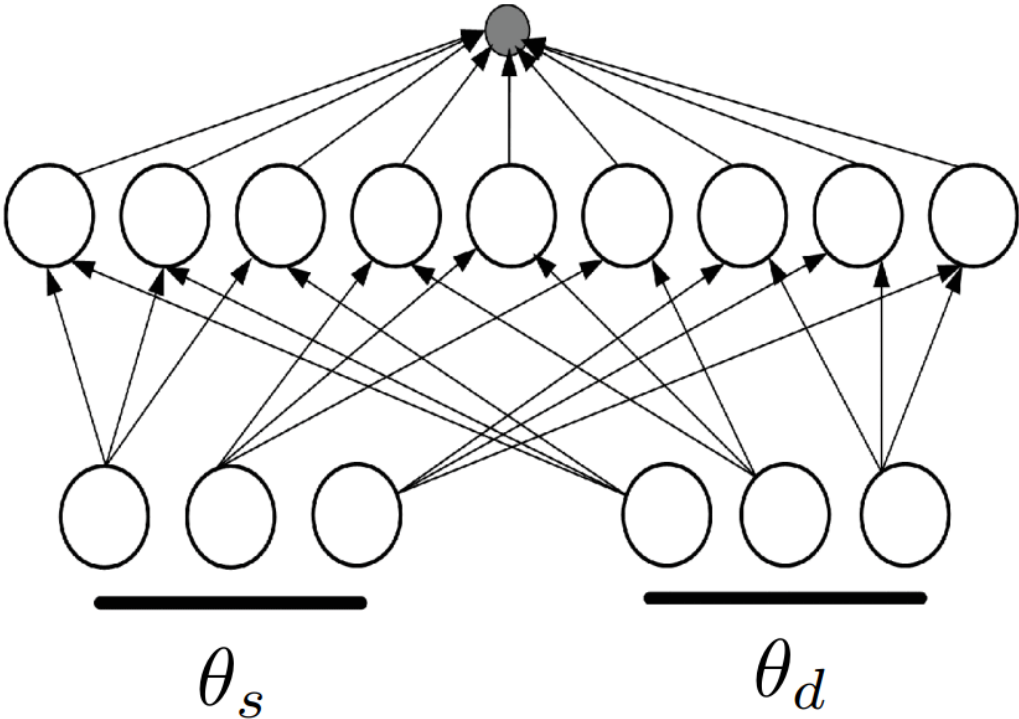
La relación queda expresada por la matriz A_r .

RESCAL


Model	$\mathbf{g}(\mathbf{x}, \theta_{\mathbf{r}})$	$\text{sim}(\mathbf{a}, \mathbf{b})$
RESCAL	$A_r x$	$\langle a, b \rangle$

$$f(\theta_s, A_r, \theta_d) = \sum_{i=0}^{d-1} \sum_{j=0}^{r-1} [A_r]_{ij} \cdot [\theta_s]_i \cdot [\theta_d]_j$$

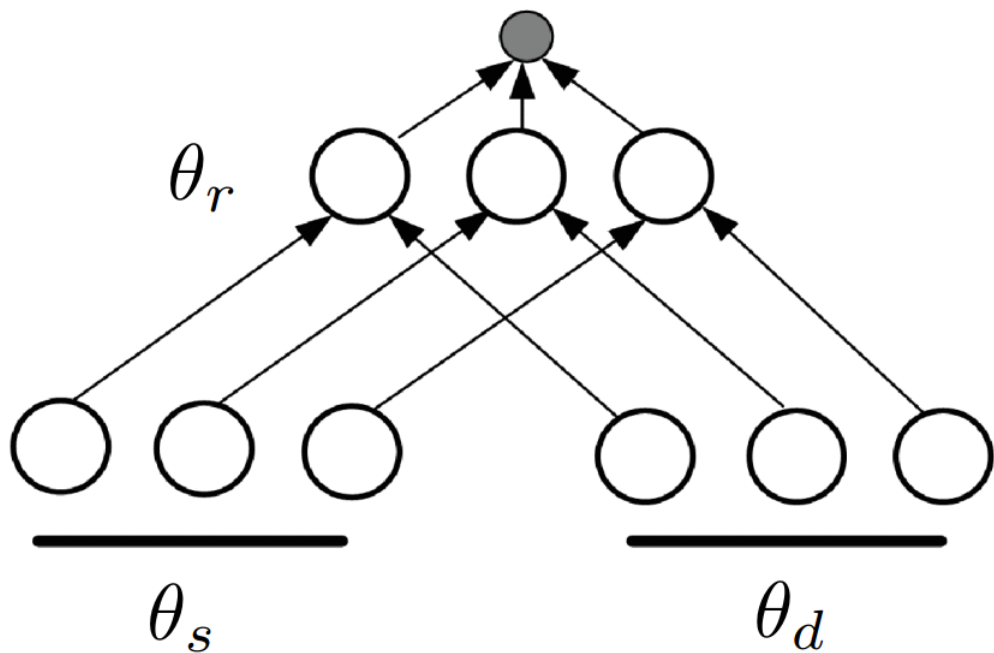
$$f(\theta_s, A_r, \theta_d)$$



RESCAL y DistMult

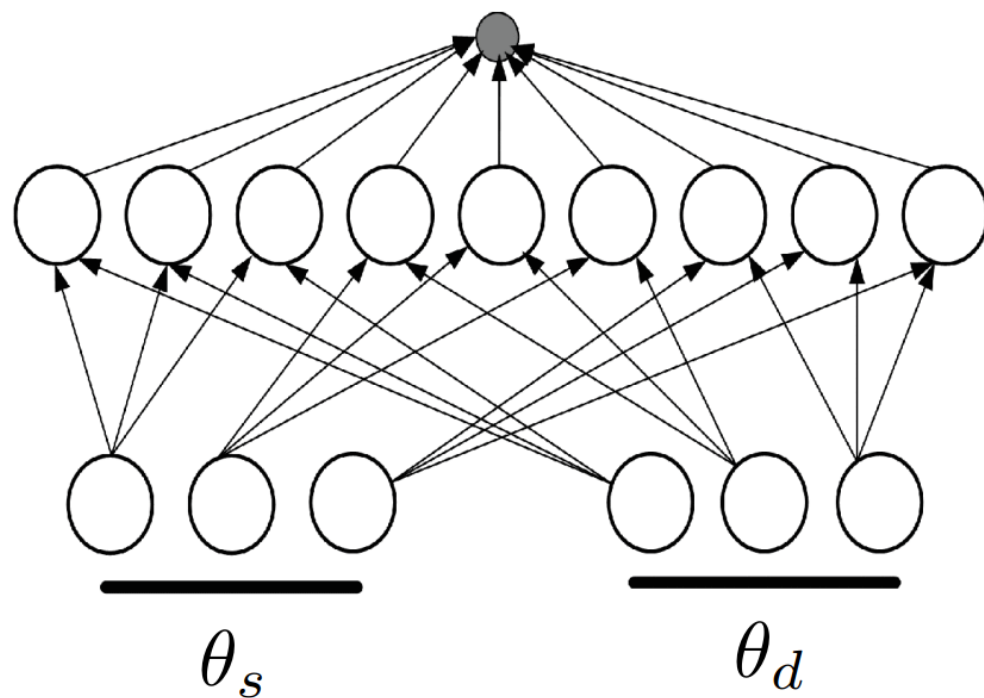
 $diag(A_r)$

$$f(\theta_s, \theta_r, \theta_d)$$



DistMult

$$f(\theta_s, A_r, \theta_d)$$



RESCAL

Vectorización de enlaces

En resumen, como existen varias alternativas, denominamos a lo anterior un framework. Es decir, en base a cada alternativa tenemos un modelo específico de vectorización:

Model	$g(\mathbf{x}, \theta_r)$	$\text{sim}(\mathbf{a}, \mathbf{b})$
RESCAL	$A_r x$	$\langle a, b \rangle$
TransE	$x + \theta_r$	$\cos(a, b)$
DistMult	$x \odot \theta_r$	$\langle a, b \rangle$

Estrategia de entrenamiento:

hiperparámetro
↓

$$\mathcal{L}_{hinge} = \sum_{e \in G} \sum_{e' \in S'_e} \max(0, \lambda - (f(e) - f(e')))$$

Negative sampling (enlace corroido): $S'_e = \{(s', r, d) | s' \in V\} \cup \{(s, r, d') | d' \in V\}$