Three related models and A New Idea of Global Attention Graph Embedding Model for Out-of-Knowlege-Base Entities

YongTao Xia

LIC group

March 15, 2018

Abstract

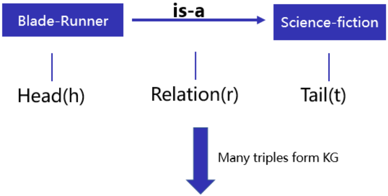
This report is divided into three parts.Firstly I will introduce some background knowledge about triple and knowledge graph.Secondly I will report three models about graph embedding.At last I will share the idea about my work. My purpose is to improve the performance of the first model and my motivations come from the next two models.

Overview

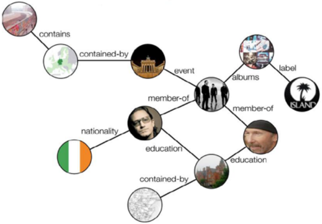
1. Background
2. Related Models
   - OOKB
   - SDNE
   - GAT
3. My idea

Triple and Knowledge Graph

Triple:



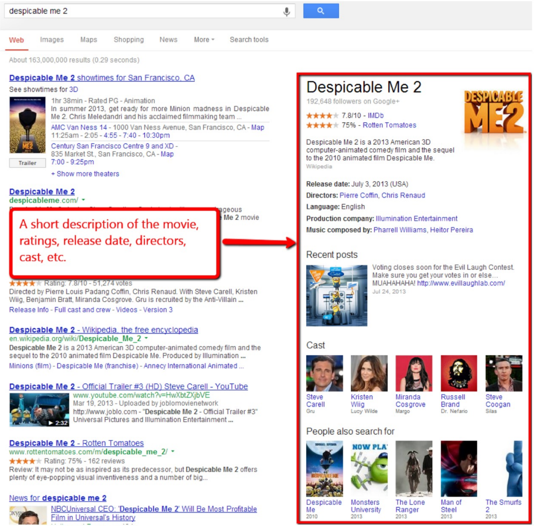
Knowledge Graph:



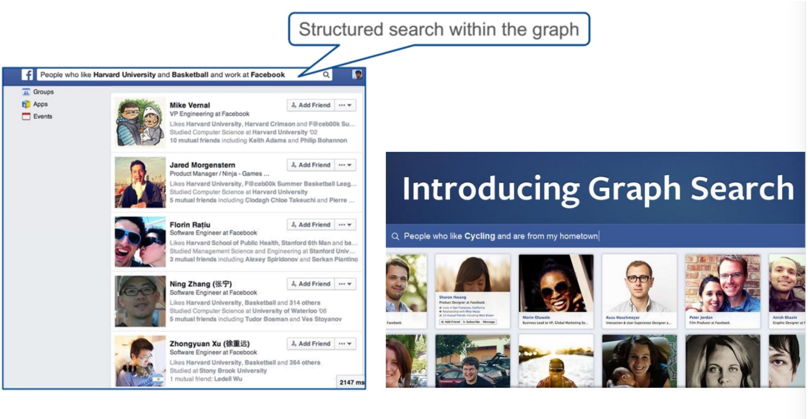
Knowledge Graph

Knowledge graph first proposed by google in 2012. It is developing very fast and has a lot of applications. Such as question answering, recommendation system and improve search capability. The most important application is improving search capability. From semantic network to knowledge graph, knowledge graph makes content things rather than strings.

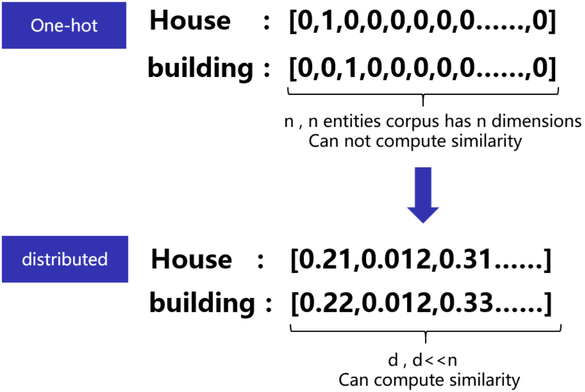
Application:Things not Strings



Application:Things not Strings



Distributed representation



Advantage and Application

Advantage of distributed representation

- **Increase computational efficiency**
- **Alleviate data sparseness**
- **Calculate the association of different entities**

Application

- **Named entity disambiguation**
- **Information extraction**
- **Knowledge graph completion**
- **Question answering**

Translational Distance Model

Introduction

Translational distance models exploit distance-based scoring functions. They measure the plausibility of a fact as the distance between the two entities.

Score function of TransE

$f_r(h, t) = ||V_h + V_r - V_t||$

Training process of TransE

1. Random initialization: V_h, V_r, V_t
2. Use Stochastic gradient descent to optimize V_h, V_r, V_t to satisfy the score function: $f_r(h, t) = ||V_h + V_r - V_t||$

Knowledge graph completion

Train

Use triples in the training set to optimize representation to satisfy the score function:

$$f_r(h, t)$$

Predict

Use score function to judge the new triple in test set is fact or not

The limitation of translational model

But the translational distance model has one fatal limitation. That is when a new entity was put into knowledge graph. This entity never showed up in training set. The translational distance model can't handle this. If don't retrain the model, it can't predict any new triple with this new entity.

Knowledge Transfer for Out-of-Knowledge-Base Entities: A Graph Neural Network Approach

Takuo Hamaguchi

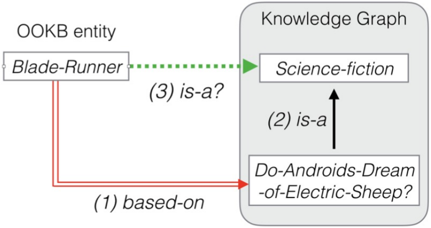
Publish:IJCAI Cited by 1

2017

Motivation

A problem of Translational Model

Unable to deal with a new entity without retraining



But in fact we can infer more facts from the knowledge we already have

Settings

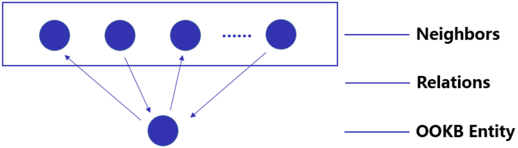
OOKB problem

- $\varepsilon(\mathcal{G}) = \{h|(h,r,t) \in \mathcal{G}\} \cup \{t|(h,r,t) \in \mathcal{G}\}$
- $\mathcal{R}(\mathcal{G}) = \{r|(h,r,t) \in \mathcal{G}\}$.
- $\mathcal{G}_{aux}$ are given at test time.
- $\mathcal{G}_{aux}$ contains new entities $\varepsilon_{OOKB} = \varepsilon(\mathcal{G}_{aux}) \backslash \varepsilon(\mathcal{G})$, but no new relations are involved.
- It is assumed that every triplet in $\mathcal{G}_{aux}$contains exactly one OOKB entity from ε_{OOKB} and one entity from $\varepsilon(\mathcal{G})$
- We want to design a model by which the information we already have in $\mathcal{G}$ can be transferred to OOKB entities ε_{OOKB}, with the help of the added knowledge $\mathcal{G}_{aux}$

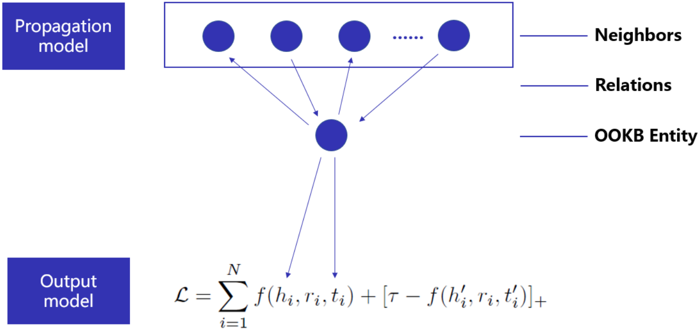
Settings

Neighbor

- $\mathcal{N}_{head}(e) = \{(h, r, e) | (h, r, e) \in \mathcal{G}\}$
- $\mathcal{N}_{tail}(e) = \{(e, r, t) | (e, r, t) \in \mathcal{G}\}$



OOKB model



$$\mathcal{L} = \sum_{i=1}^{N} f(h_i, r_i, t_i) + [\tau - f(h_i', r_i, t_i')]_+$$

Propagation model

propagation Function

- $S_{head}(e) = \{T_{head}(v_h; h, r, e) | (h, r, e) \in \mathcal{N}_h(e)\}$
- $S_{tail}(e) = \{T_{tail}(v_t; h, r, e) | (e, r, t) \in \mathcal{N}_t(e)\}$
- $v_e = P(S_{head}(e) \cup S_{tail}(e))$

Transition Function

- $T_{head}(v_h; h, r, e) = ReLU(BN(A_r^{head} v_h))$
- $T_{tail}(v_t; e, r, t) = ReLU(BN(A_r^{tail} v_t))$

Pooling Function

- sum pooling:$P(S) = \sum_{i=1}^{N} x_i$
- average pooling:$P(S) = \frac{1}{N} \sum_{i=1}^{N} x_i$
- max pooling:P(S)=$max(\{x_i\}_{i=1}^{N})$

Output model

Absolute-Margin Objective Function

$\mathcal{L} = \sum_{i=1}^{N} f(h_i, r_i, t_i) + [\tau - f(h'_i, r_i, t'_i)]_+$

- $f(h_i, r_i, t_i)$ is a positive triplet
- $f(h'_i, r_i, t'_i)$ is a negative triplet
- τ is a hyperparameter , called the margin.
- positive triplets will be optimized towards zero,whereas negative triplets are going to be at least τ

Datasets

	WordNet11	Freebase13
Relations	11	13
Entities	38,696	75,043
Training triplets	112,581	316,232
Validation triplets	5,218	11,816
Test triplets	21,088	47,466

Experiment: Standard Triplet Classification

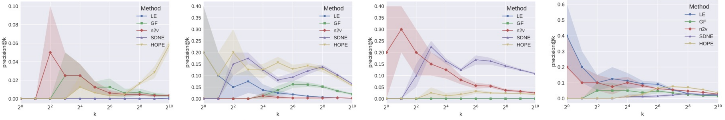
Method	WordNet11	Freebase13
NTN [Socher *et al.*, 2013]	70.4	87.1
TransE [Bordes *et al.*, 2013]	75.9	81.5
TransH [Wang *et al.*, 2014b]	78.8	83.3
TransR [Lin *et al.*, 2015]	85.9	82.5
TransD [Ji *et al.*, 2015]	86.4	**89.1**
TransE-COMP [Guu *et al.*, 2015]	80.3	87.6
TranSparse [Ji *et al.*, 2016]	86.8	88.2
ManifoldE [Xiao *et al.*, 2016a]	<u>87.5</u>	87.3
TransG [Xiao *et al.*, 2016b]	87.4	87.3
lppTransD [Yoon *et al.*, 2016]	86.2	<u>88.6</u>
NMM [Nguyen *et al.*, 2016]	86.8	<u>88.6</u>
Proposed method	**87.8**	81.6

OOKB Datasets

	Head			Tail			Both		
	1,000	3,000	5,000	1,000	3,000	5,000	1,000	3,000	5,000
Training triplets	108,197	99,963	92,309	96,968	78,763	67,774	93,364	71,097	57,601
Validation triplets	4,613	4,184	3,845	3,999	3,122	2,601	3,799	2,759	2,166
OOKB entities	348	1,034	1,744	942	2,627	4,011	1,238	3,319	4,963
Test triplets	994	2,969	4,919	986	2,880	4,603	960	2,708	4,196
Auxiliary entities	2,474	6,791	10,784	8,191	16,193	20,345	9,899	19,218	23,792
Auxiliary triplets	4,352	12,376	19,625	15,277	31,770	40,584	18,638	38,285	48,425

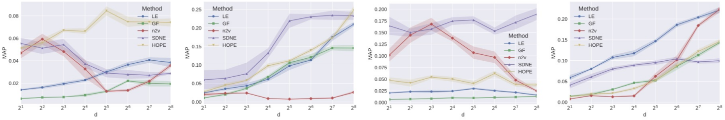
OOKB Classification Experiment

		Head			Tail			Both		
Method	Pooling	1,000	3,000	5,000	1,000	3,000	5,000	1,000	3,000	5,000
Baseline	sum	54.6	52.5	52.0	53.7	53.0	52.8	54.0	52.7	53.2
	max	58.1	56.3	56.4	55.2	54.2	55.3	56.8	56.8	56.4
	avg	63.0	60.2	61.1	63.8	63.9	63.0	65.3	63.9	64.8
Proposed	sum	70.2	62.6	59.6	64.6	56.5	55.0	59.5	55.2	54.2
	max	80.3	75.4	72.7	74.8	63.1	58.7	68.0	59.5	56.5
	avg	**87.3**	**84.3**	**83.3**	**84.0**	**75.2**	**69.2**	**83.0**	**73.3**	**68.2**



(a) PPI (b) AstroPh (c) BlogCatalog (d) Hep-th

Fig. 6. Precision@k of link prediction for different data sets (dimension of embedding is 128).



(a) PPI (b) AstroPh (c) BlogCatalog (d) Hep-th

Fig. 7. MAP of link prediction for different data sets with varying dimensions.

Structural Deep Network Embedding

Dai xin Wang

Publish:ACM Cited by 87

2016

Challenges of network representations

High non-linearity

how to design a model to capture the highly non-linear structure is rather difficult

Structure preserving

The similarity of vertexes is dependent on both the local and global network structure. Therefore, how to simultaneously preserve the local and global structure is a tough problem.

Sparsity

Many real-world networks are often so sparse that only utilizing the very limited observed links is not enough to reach a satisfactory performance

Problem Definition

Graph

- A graph is denoted as $G = (V, E), V = \{v_1, ..., v_n\}$represents n vertexes and $E = \{e_{i,j}\}_{i,j=1}^{n}$ represents the edges.
- Each edge $e_{i,j}$ is associated with a weight $s_{i,j} \geq 0$. For v_i and v_j not linked by an edge $s_{i,j} = 0$

Problem Definition

First-Order Proximity

The first-order proximity describes the pairwise proximity between vertexes.

Second-Order Proximity

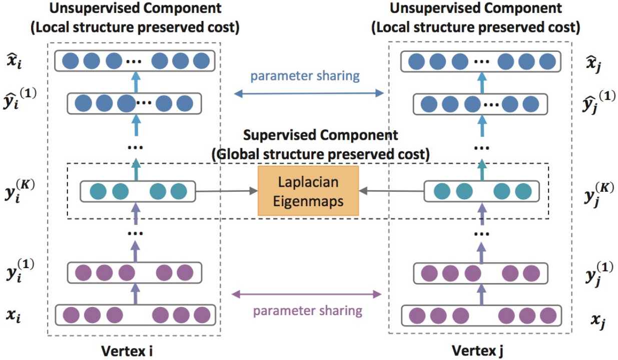
However, real-world datasets are often so sparse that the observed links only account for a small portion. There exist many vertexes which are similar with each other but not linked by any edges. Therefore, only capturing the first-order proximity is not sufficient.Intuitively, the second-order proximity assumes that if two vertexes share many common neighbors, they tend to be similar.

Problem Definition

Network Embedding

network embedding aims to learn a mapping function
$f : v_i \mapsto y_i \in \mathbb{R}^d$,where $d \ll |V|$

Model Frame



Unsupervised Component
(Local structure preserved cost)
Unsupervised Component
(Local structure preserved cost)
$\hat{x}_i$
$\hat{y}_i^{(1)}$
parameter sharing
$\hat{x}_j$
$\hat{y}_j^{(1)}$
Supervised Component
(Global structure preserved cost)
$y_i^{(K)}$
Laplacian
Eigenmaps
$y_j^{(K)}$
$y_i^{(1)}$
$y_j^{(1)}$
x_i
parameter sharing
x_j
Vertex i
Vertex j

The loss function of first-order proximity

First-order proximity

$\mathcal{L}_{1st} = \sum_{i,j=1}^{n} s_{i,j}||y_i - y_j||_2^2$

Borrows the idea of Laplacian Eigenmaps to make the vertexes linked by an edge be mapped near in the embedding space.

The loss function of second-order proximity

Second-order proximity

$\mathcal{L}_{2nd} = \sum_{i=1}^{n} ||(\hat{x}_i - x_i) \odot b_i||_2^2$

- $x_i = s_i$, S is the adjacency matrix.
- $\odot$ means the Hadamard product.If $s_{i,j} = 0$, $b_{i,j} = 1$ else $b_{i,j} = \beta > 1$.Impose more penalty to the reconstruction error of the non-zero elements than that of zero elements

Joint Objective Function

Mix

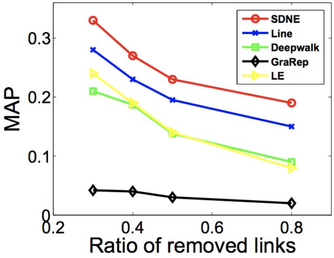
$$\mathcal{L}_{mix} = \mathcal{L}_{2nd} + \alpha\mathcal{L}_{1st} + \nu\mathcal{L}_{reg}$$

Experiment

Table 5: *precision@k* on ARXIV GR-QC for link prediction

Algorithm	P@2	P@10	P@100	P@200	P@300	P@500	P@800	P@1000	P@10000
SDNE	**1**	**1**	**1**	**1**	**1***	**0.99****	**0.97****	**0.91****	**0.257****
LINE	1	1	1	1	0.99	0.936	0.74	0.79	0.2196
DeepWalk	1	0.8	0.6	0.555	0.443	0.346	0.2988	0.293	0.1591
GraRep	1	0.2	0.04	0.035	0.033	0.038	0.035	0.035	0.019
Common Neighbor	1	1	1	0.96	0.9667	0.98	0.8775	0.798	0.192
LE	1	1	0.93	0.855	0.827	0.66	0.468	0.391	0.05

Significantly outperforms Line at the: ** 0.01 and * 0.05 level, paired t-test.



Attention Mechanism

Another latest paper about graph embedding is graph attention embedding.Attention mechanisms have become almost a standard in many sequence-based tasks. The motivation of attention mechanism is when people observe things, they pay attention to a certain part rather than the whole picture. One of the benefits of attention mechanisms is that they allow for dealing with variable sized inputs, focusing on the most relevant parts of the input to make decisions.

Graph Attention Networks

Petar / Bengio

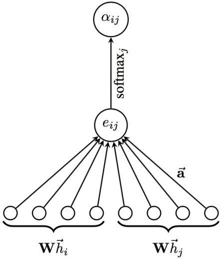
Publish:ICLR

2018

attention coefficients

$e_{i,j} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$

- $\vec{h}_i, \vec{h}_j$ are the vector of node i and node j.
- $\mathbf{W} \in \mathbb{R}^{F' \times F}$ a shared linear transformation($\vec{h}_i \in \mathbb{R}^F \Rightarrow \vec{h}'_i \in \mathbb{R}^{F'}$)
- a shared attentional mechanism $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$ there is a single-layer feedforward neural network.



GAT Architecture

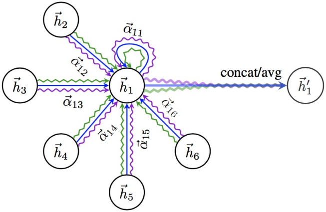
Normalize

$$\alpha_{i,j} = softmax_j(e_{i,j}) = \frac{exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} exp(e_{ik})}$$

GAT Architecture

Multi-head attention

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$



Advantages and Limitations

Advantages

- computationally efficent:the self-attentional layer can be parallelized across all edges, and the computation of output features can be parallelized across all nodes.
- allows for assigning different importances to nodes of a same neighborhood.
- applied in a shared manner to all edges in the graph, therefore it applicable to inductive learning problem.

Advantages and Limitations

Limitation

Practical nature: utilize softmax leads to $O(V^2)$ space complexity.This requirement may be reduced to $O(D_2)$(where D is the maximum node indegree in the graph).
One potential avenue for addressing this is utilizing pointwise activation functions (such as the logistic sigmoid) to compute the α_{ij} values.But this approach yielded significant drops in predictive power across all experiments.

Experiments

Transductive

Method	Cora	Citeseer
MLP	55.1%	46.5%
ManiReg (Belkin et al., 2006)	59.5%	60.1%
SemiEmb (Weston et al., 2012)	59.0%	59.6%
LP (Zhu et al., 2003)	68.0%	45.3%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%
ICA (Lu & Getoor, 2003)	75.1%	69.1%
Planetoid (Yang et al., 2016)	75.7%	64.7%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%
GCN (Kipf & Welling, 2017)	81.5%	70.3%
GAT (ours)	**83.3%**	**74.0%**
improvement w.r.t GCN	1.8%	3.7%

Inductive

Method	PPI
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GAT (ours)	**0.942**
improvement w.r.t GraphSAGE	33.0%

My idea

Contrast experiment

Method	WordNet11	Freebase13
NTN [Socher *et al.*, 2013]	70.4	87.1
TransE [Bordes *et al.*, 2013]	75.9	81.5
TransH [Wang *et al.*, 2014b]	78.8	83.3
TransR [Lin *et al.*, 2015]	85.9	82.5
TransD [Ji *et al.*, 2015]	86.4	**89.1**
TransE-COMP [Guu *et al.*, 2015]	80.3	87.6
TranSparse [Ji *et al.*, 2016]	86.8	88.2
ManifoldE [Xiao *et al.*, 2016a]	87.5	87.3
TransG [Xiao *et al.*, 2016b]	87.4	87.3
lppTransD [Yoon *et al.*, 2016]	86.2	88.6
NMM [Nguyen *et al.*, 2016]	86.8	88.6
Proposed method	**87.8**	81.6

		Head			Tail			Both		
Method	Pooling	1,000	3,000	5,000	1,000	3,000	5,000	1,000	3,000	5,000
Baseline	sum	54.6	52.5	52.0	53.7	53.0	52.8	54.0	52.7	53.2
	max	58.1	56.3	56.4	55.2	54.2	55.3	56.8	56.8	56.4
	avg	63.0	60.2	61.1	63.8	63.9	63.0	65.3	63.9	64.8
Proposed	sum	70.2	62.6	59.6	64.6	56.5	55.0	59.5	55.2	54.2
	max	80.3	75.4	72.7	74.8	63.1	58.7	68.0	59.5	56.5
	avg	**87.3**	**84.3**	**83.3**	**84.0**	**75.2**	**69.2**	**83.0**	**73.3**	**68.2**

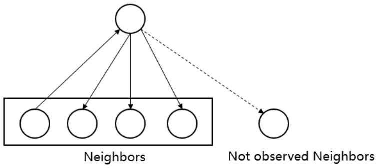
OOKB's First Problem

Ignore second-order proximity

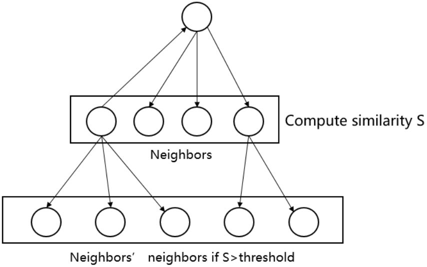
Many real-world networks are often so sparse that only utilizing the very limited observed links.

In SDNE:

- First-order proximity:$\mathcal{L}_{1st} = \sum_{i,j=1}^{n} s_{i,j} ||y_i - y_j||_2^2$
- Second-order proximity:$\mathcal{L}_{2nd} = \sum_{i=1}^{n} ||(\hat{x}_i - x_i) \odot b_i||_2^2$



My idea of second-order proximity



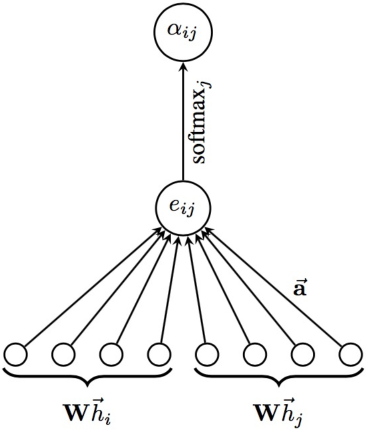
- utilize cosine similarity compute similarity S
- if S>threshold take neighbors' neighbors into consideration
- consider K layers , with decreasing function:
  - $w(t) = \alpha^t + \alpha^t(1-\alpha)(K-t)$
  - $w(t) = -t/K + (1 + 1/K)$
  - $w(t) = 1/t$

OOKB's Second Problem

Every neighbor has the same weigh

- $v_e = P(S_{head}(e) \cup S_{tail}(e))$
- average pooling: $P(S) = \frac{1}{N}\sum_{i=1}^{N} x_i$

My idea of introducing attention mechanism



The End