Graph Convolutional Neural Networks for Web-Scale Recommender Systems

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai,
William L. Hamilton & Jure Leskovec
KDD 2018 [https://arxiv.org/abs/1806.01973]

Presented by Cătălina Cangea

Recommender systems

- Graph-structured data essential for recommendation applications (can exploit *user-to-item relations* and *social graphs*)
- Item embeddings learned with deep models can be re-used across multiple tasks (e.g. *item* recommendation and *collection* recommendation - playlists, news feed)
- GCN-based methods successful on recommender system benchmarks

Theory → scale?

- Challenge: apply GCN-based training and inference to graphs with *billions* of nodes and *tens of billions* of edges
- Recommender systems of this kind perform operations using the full graph Laplacian during training, which is problematic if:
  - There are billions of nodes in the graph
  - The structure of the graph is *constantly evolving*

PinSage

- Used for web-scale recommendation at Pinterest
- GCN-based algorithm which leverages random walks to generate node embeddings that incorporate features and graph structure
- Largest application of deep graph embeddings:
  - 3BN nodes (“pins” and “boards”), 18BN edges
  - (about 10,000x larger than typical GCN applications)

Key insights

- **Localized convolutions:**
  - Sampling node neighborhoods through short random walks (also gives importance scores)
  - Convolutional modules share parameters across nodes
- **Importance pooling:** use scores to weight node features (+46%)
- **Curriculum training:** increase difficulty of examples (+12%)
- **Efficiency:** producer-consumer minibatches, MapReduce

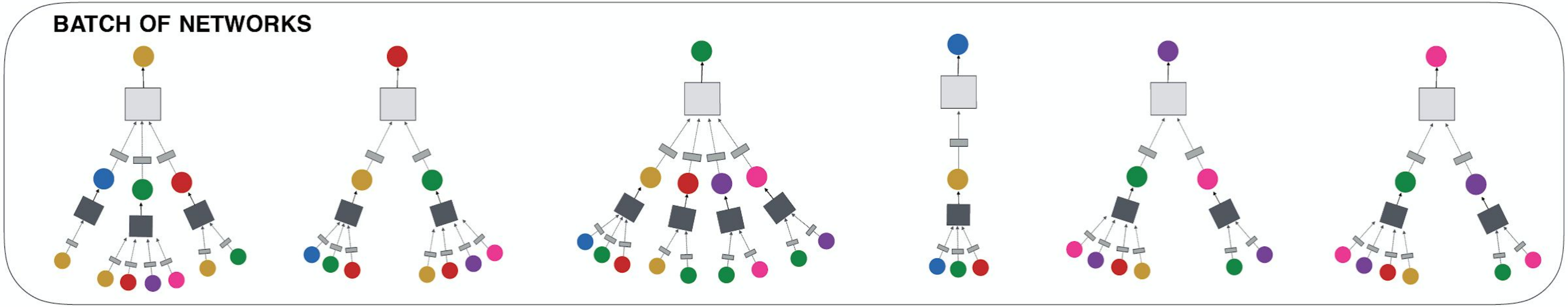


Figure 1. Embeddings for each node are computed by a different network, but parameters are shared among boxes with same shading.

Graph problem setup

- Pinterest: content discovery application
  - *Pins* (visual links to online content) - 2BN
  - *Boards* (collections of thematically related pins) - 1BN
- Model as bipartite graph ($\boldsymbol{V} = \boldsymbol{I} \cup \boldsymbol{C}$):
  - $\boldsymbol{I}$ - pins, $\boldsymbol{C}$ - boards
  - 18BN edges (pin-board)
- A pin $\boldsymbol{u}$ has real-valued attributes $\boldsymbol{x}_{\boldsymbol{u}}$ (text and image features)

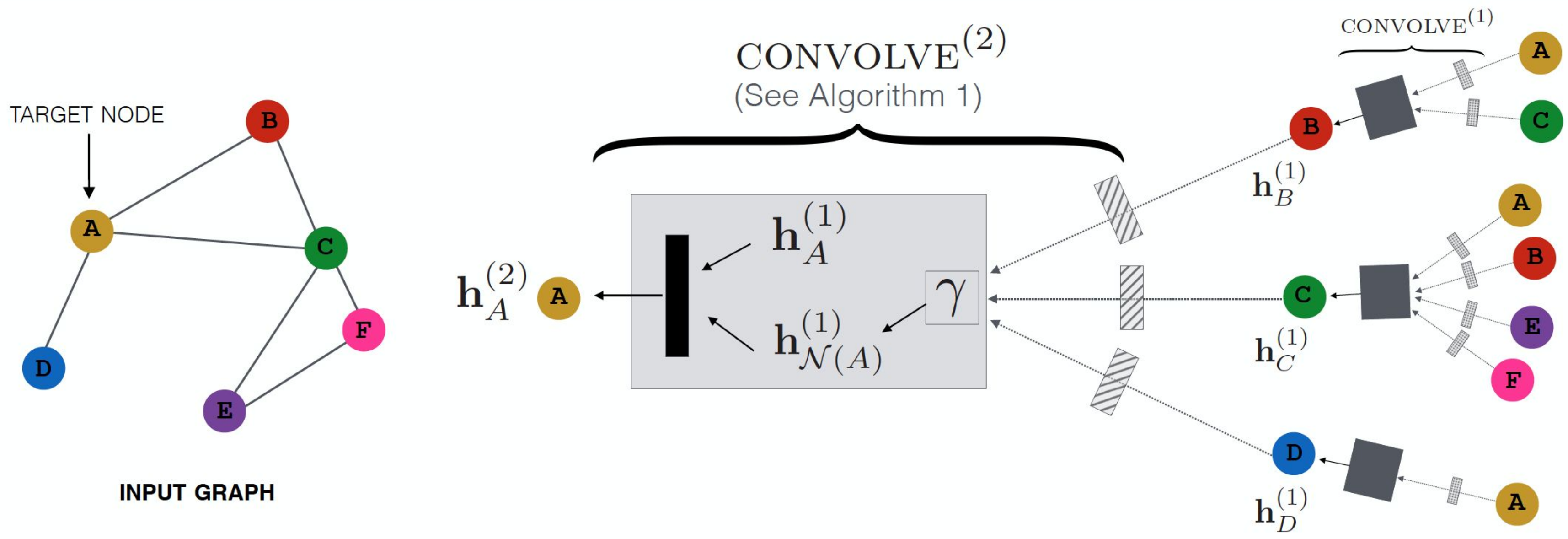


Figure 2. An input graph (*left*) and the 2-level network used to compute the embedding of node **A** (*right*).

Importance-based neighbor sampling

- Previous approaches: k-hop graph neighborhoods
- PinSage:
  - Start random walk from u
  - Compute L1-normalized visit count of nodes
  - $\boldsymbol{N(u)}$ = T most "influential" neighbors of node u (having the highest visit counts) → set of weights α

Algorithm 1: CONVOLVE

Input : Current embedding $\mathbf{z}_u$ for node u; set of neighbor embeddings $\{\mathbf{z}_v | v \in \mathcal{N}(u)\}$, set of neighbor weights $\boldsymbol{\alpha}$; symmetric vector function $\gamma(\cdot)$ *weighted sum*

Output : New embedding $\mathbf{z}_u^{\text{NEW}}$ for node u

1 $\mathbf{n}_u \leftarrow \gamma\left(\{\text{ReLU}\left(\mathbf{Q}\mathbf{h}_v + \mathbf{q}\right) \mid v \in \mathcal{N}(u)\}, \boldsymbol{\alpha}\right);$ *weights*

2 $\mathbf{z}_u^{\text{NEW}} \leftarrow \text{ReLU}\left(\mathbf{W} \cdot \text{CONCAT}(\mathbf{z}_u, \mathbf{n}_u) + \mathbf{w}\right);$

3 $\mathbf{z}_u^{\text{NEW}} \leftarrow \mathbf{z}_u^{\text{NEW}} / \|\mathbf{z}_u^{\text{NEW}}\|_2$

Algorithm 2: MINIBATCH

Input : Set of nodes $\mathcal{M} \subset \mathcal{V}$; depth parameter K;
neighborhood function $\mathcal{N} : \mathcal{V} \to 2^{\mathcal{V}}$

Output: Embeddings $\mathbf{z}_u, \forall u \in \mathcal{M}$

/* Sampling neighborhoods of minibatch nodes. */

$\mathcal{S}^{(K)} \leftarrow \mathcal{M}$;

for $k = K, ..., 1$ **do**

    $\mathcal{S}^{(k-1)} \leftarrow \mathcal{S}^{(k)}$;

    for $u \in \mathcal{S}^{(k)}$ **do**

        $\mathcal{S}^{(k-1)} \leftarrow \mathcal{S}^{(k-1)} \cup \mathcal{N}(u)$;

    end

end

/* Generating embeddings */

$\mathbf{h}_u^{(0)} \leftarrow \mathbf{x}_u, \forall u \in \mathcal{S}^{(0)}$;

for $k = 1, ..., K$ **do**

    for $u \in \mathcal{S}^{(k)}$ **do**

        $\mathcal{H} \leftarrow \left\{\mathbf{h}_v^{(k-1)}, \forall v \in \mathcal{N}(u)\right\}$;

        $\mathbf{h}_u^{(k)} \leftarrow \textsc{convolve}^{(k)}\left(\mathbf{h}_u^{(k-1)}, \mathcal{H}\right)$

    end

end

for $u \in \mathcal{M}$ **do**

    $\mathbf{z}_u \leftarrow \mathbf{G}_2 \cdot \mathrm{ReLU}\left(\mathbf{G}_1 \mathbf{h}_u^{(K)} + \mathbf{g}\right)$

end

Training

- Labelled pairs of items: $\boldsymbol{L} = \{(q, i) \mid \text{item } i \text{ is a good recommendation candidate for query } q\}$
- Goal: output embeddings of q and i are close to each other

Loss function

- Maximize inner product of positive examples (q is related to i)
- Make inner product of negative examples (q is *unrelated* to n_k) smaller than the one of the positive example by Δ
- For a pair of embeddings $(\boldsymbol{z}_q, \boldsymbol{z}_i) : (q, i) \in \boldsymbol{L}$, the loss function is:

$$J_{\mathcal{G}}(\mathbf{z}_q \mathbf{z}_i) = \mathbb{E}_{n_k \sim P_n(q)} \max\{0, \mathbf{z}_q \cdot \mathbf{z}_{n_k} - \mathbf{z}_q \cdot \mathbf{z}_i + \Delta\}$$

Negative sampling

- Approximate the normalization factor of edge likelihood
- Sample 500 negative items *shared* across all training examples in each minibatch
- Include “hard” negative examples:
  - Somewhat relevant to q, but not as related as i
  - Randomly sample items with Personalized PageRank $score \in [2000, 5000]$

Curriculum learning

- Using negative items requires 2x epochs for convergence
- First epoch: no negative items used → find area in parameter space with small loss
- Gradually add negative items, focusing model on learning to distinguish between highly related and somewhat related items
  - At epoch n, have $n - 1$ hard negative items for each item

Experimental setup

- Pairs of pins (*q*, *i*): a user interacted with pin *i* immediately after interacting with pin *q*
- 1.2BN pairs of positive examples (+500 negative per minibatch, 6 hard negative per pin) → 7.5BN for training
- Only train on subset of Pinterest graph, generate embeddings for entire graph using a MapReduce pipeline

Features used

- Each pin contains an image and text annotations
- Concatenate:
  - Visual embeddings (4096-D, 6th layer of VGG-16)
  - Textual embeddings (256-D, Word2Vec)
  - Log-degree of pin in the graph

Algorithm 1: CONVOLVE

Input : Current embedding $\mathbf{z}_u$ for node u; set of neighbor embeddings $\{\mathbf{z}_v | v \in \mathcal{N}(u)\}$, set of neighbor weights $\boldsymbol{\alpha}$; symmetric vector function $\gamma(\cdot)$

Output: New embedding $\mathbf{z}_u^{\text{NEW}}$ for node u

$\mathbf{n}_u \leftarrow \gamma\left(\{\text{ReLU}\left(\mathbf{Q}\mathbf{h}_v + \mathbf{q}\right) \mid v \in \mathcal{N}(u)\}, \boldsymbol{\alpha}\right)$;

$\mathbf{z}_u^{\text{NEW}} \leftarrow \text{ReLU}\left(\mathbf{W} \cdot \text{CONCAT}(\mathbf{z}_u, \mathbf{n}_u) + \mathbf{w}\right)$;

$\mathbf{z}_u^{\text{NEW}} \leftarrow \mathbf{z}_u^{\text{NEW}} / \|\mathbf{z}_u^{\text{NEW}}\|_2$

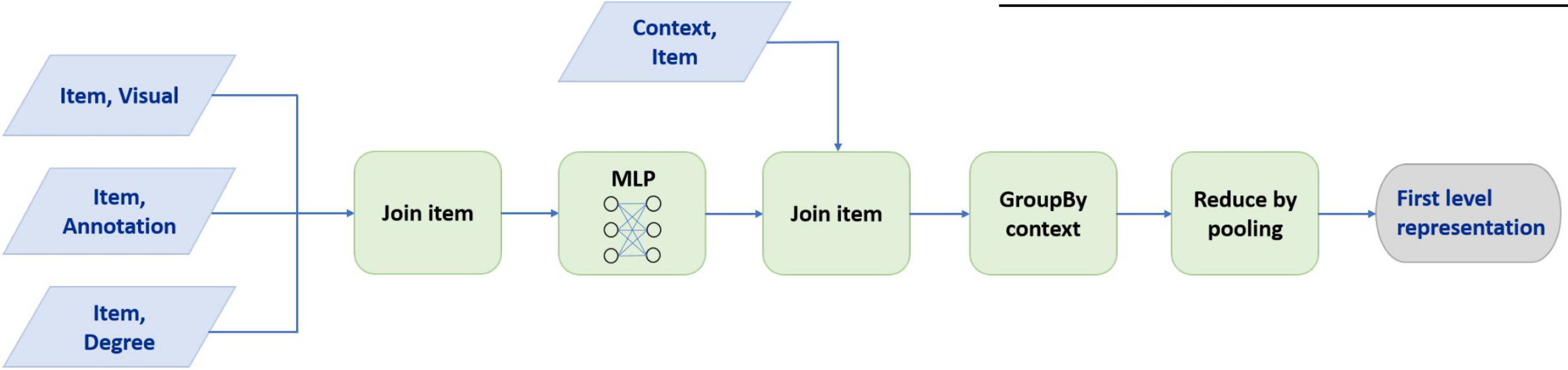


Figure 3. MapReduce-based node embedding computation; similar for higher layers (inputs are representations from previous layers).

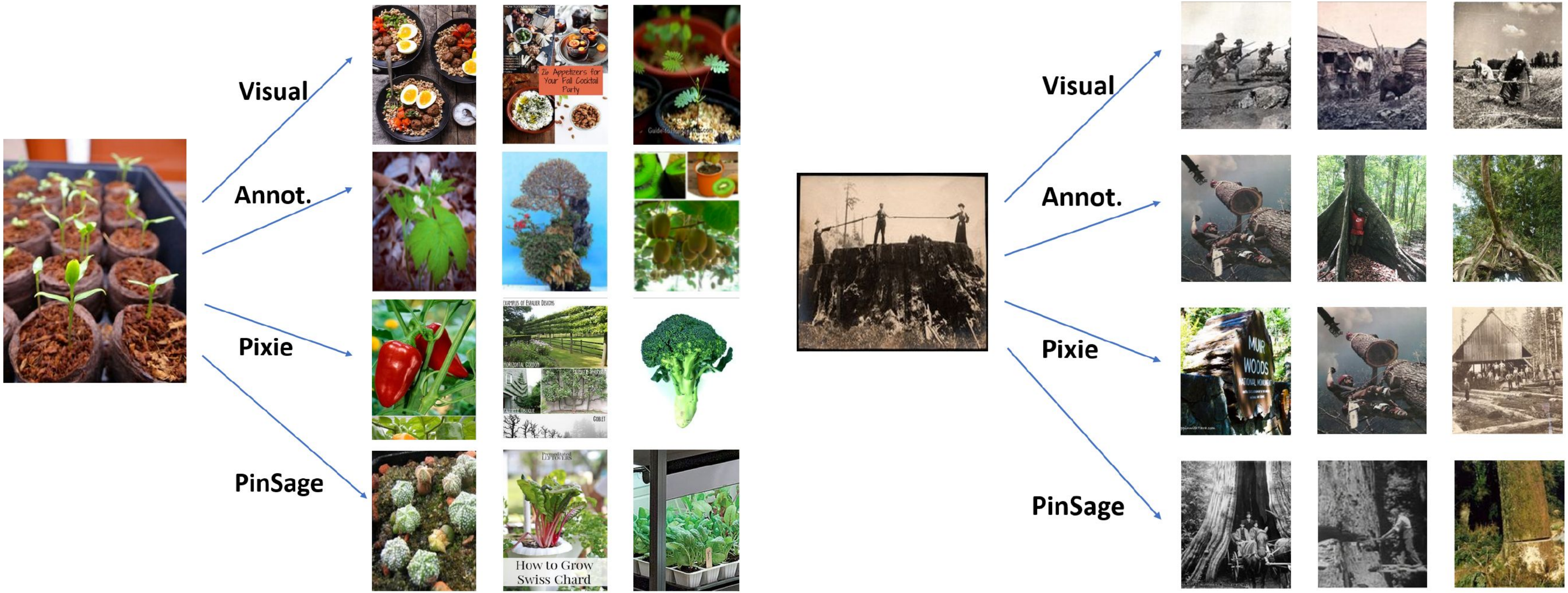
Baselines

- **Visual**: use nearest-neighbors of deep visual embeddings to make recommendations
- **Annotation**: use annotation embeddings
- **Combined**: concatenate visual and annotation embeddings, pass through 2-layer MLP
- **Pixie**: biased random walks from q, recommend items with top K ranking scores (Pinterest SOTA for some recommendation tasks)

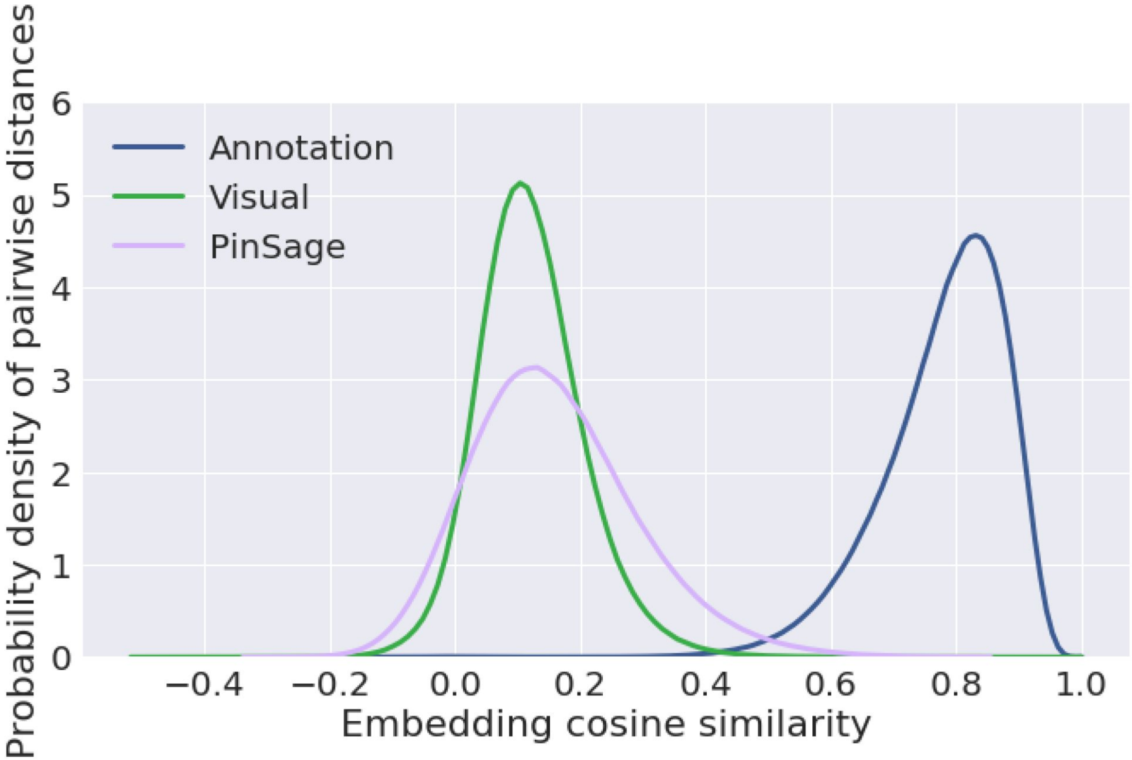
User studies

Methods	Win	Lose	Draw	Fraction of wins
PinSage vs. Visual	28.4%	21.9%	49.7%	56.5%
PinSage vs. Annot.	36.9%	14.0%	49.1%	72.5%
PinSage vs. Combined	22.6%	15.1%	57.5%	60.0%
PinSage vs. Pixie	32.5%	19.6%	46.4%	62.4%

Table 2: Head-to-head comparison of which image is more relevant to the recommended query image.


Visual
Annot.
Pixie
PinSage
26 Appetizers for Your Fall Cocktail Party
How to Grow Swiss Chard
Visual
Annot.
Pixie
MUIR WOODS
PinSage

Spread of pairwise distance distributions



Ablation studies

$$\text{MRR} = \frac{1}{n} \sum_{(q,i)\in\mathcal{L}} \frac{1}{\lceil R_{i,q}/100 \rceil}$$

% of times where $i \in NN_q$

Method	Hit-rate	MRR
Visual	17%	0.23
Annotation	14%	0.19
Combined	27%	0.37
max-pooling	39%	0.37
mean-pooling	41%	0.51
mean-pooling-xent	29%	0.35
mean-pooling-hard	46%	0.56
PinSage	**67%**	**0.59**

Figure 6: t-SNE plot of item embeddings in 2 dimensions.

Future directions?

- The whole training process is based on (q, i) pairs, so would be interesting to improve informativeness of this kind of link
- Relate boards as well, not only pins
- Weight relationship by:
  - Frequency of user's interaction with other pins that are close in the t-SNE representation
  - Some function of user statistics

Thank you!

Questions?

Catalina.Cangea@cst.cam.ac.uk

www.cst.cam.ac.uk/~ccc53/