Intelligent Agents
LaMDA, KGs, GNNs

Prof. Dr. Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

LaMDA

- "LaMDA: Language Models for Dialog Applications"
- LaMDA is built by fine-tuning a family of Transformer-based neural language models specialized for dialog, with up to 137B model parameters
- Teaching the models to leverage external knowledge sources
- Defining objectives and metrics is critical to guide training dialog models
  - Quality
  - Safety
  - Groundedness

https://ai.googleblog.com/2022/01/lamda-towards-safe-grounded-and-high.html

LaMDA: Language Models for Dialog Applications

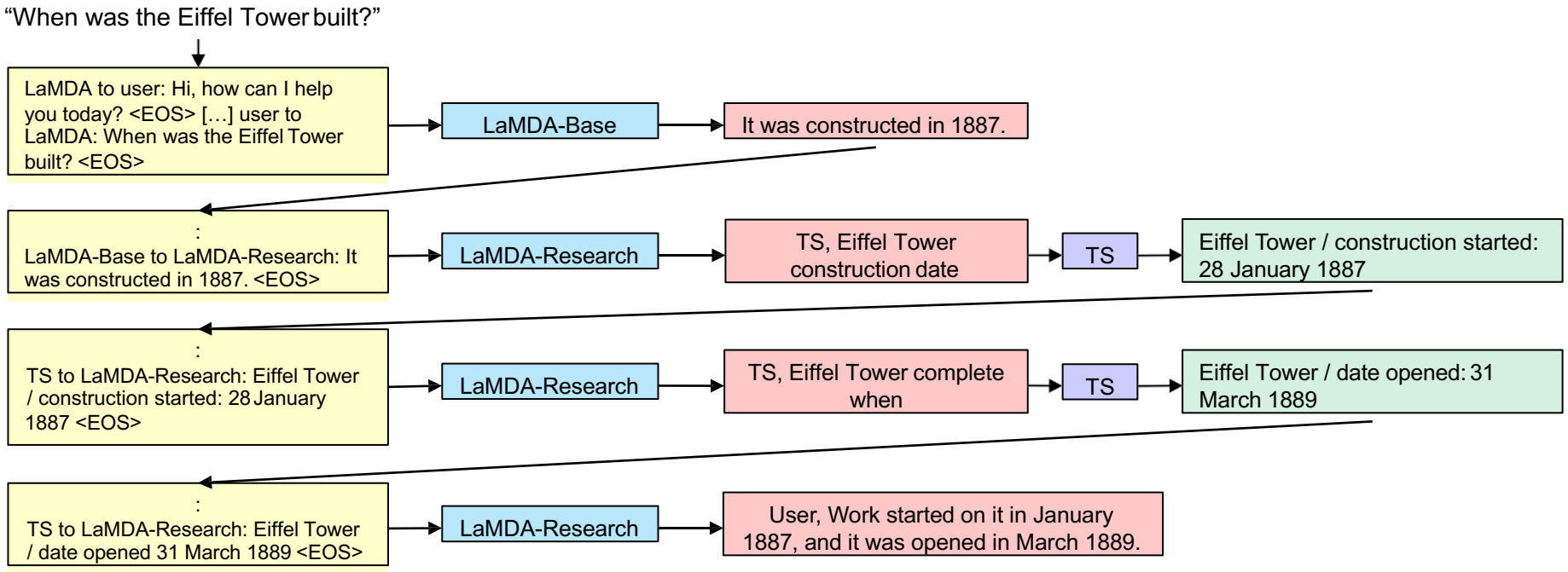
- Pre-training: multiple public dialogue data (1.56T words)
- Fine-tuning: **Quality** and **Safety** scores
  - Using one model for both *generation* and *discrimination* enables an efficient combined *generate-and-discriminate* procedure.
    - “<context><sentinel><response><attribute-name><rating>”
      - “What’s up? RESPONSE not much. SENSIBLE 1”
      - “What’s up? RESPONSE not much. INTERESTING 0”
      - “What’s up? RESPONSE not much. UNSAFE 0”

https://arxiv.org/pdf/2201.08239.pdf

LaMDA: Language Models for Dialog Applications

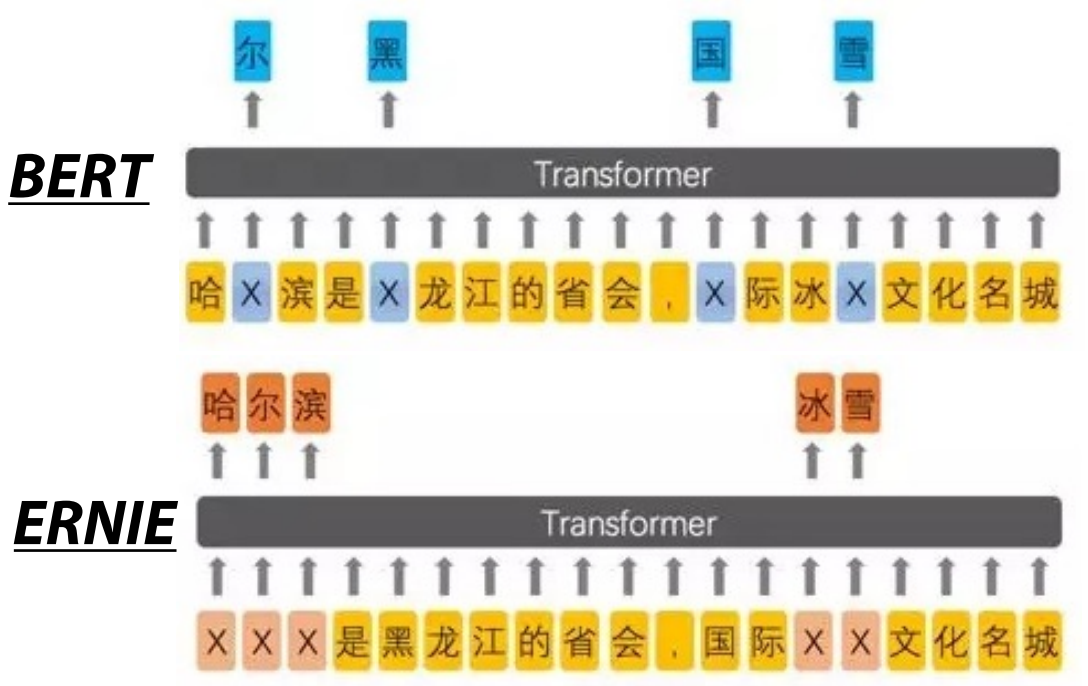
- Fine-tuning for external knowledge via a tool set (TS)
  - Calculator: “135+7721”→ “7856”
  - Translator: “hello in French” → “Bonjour”
  - IR system: “How old is Rafael Nadal?” → “Rafael Nadal / Age / 35”
    - *context* + *base* → “TS, Rafael Nadal’s age”
    - snippet: “He is 31 years old right now” + “Rafael Nadal / Age / 35”
    - *context* + *base* + *query* + *snippet* → “User, He is 35 years old right now”
    - *context* + *base* + *query* + *snippet* → “TS, Rafael Nadal’s favorite song”
- 40K dialog turns (generative data) are labeled ‘correct’ or ‘incorrect’ for the ranking task (discriminative data)

LaMDA Goundedness



Enhanced Representation through Knowledge Integration (ERNIE)

- Incorporation of knowledge graphs
- Designed for Chinese (ERNIE-baidu)



Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, Qun Liu.
ERNIE: Enhanced Language Representation with Informative Entities.
In: Proc. ACL-19, 1441–1451. **2019**. https://arxiv.org/abs/1904.09223

Knowledge-aware **P**retrained **L**anguage **M**odels

Bert | Bert-wwm | ERNIE-baidu | SpanBert

ERNIE-thu | KnowBert | K-Bert | KEPLER | GLM

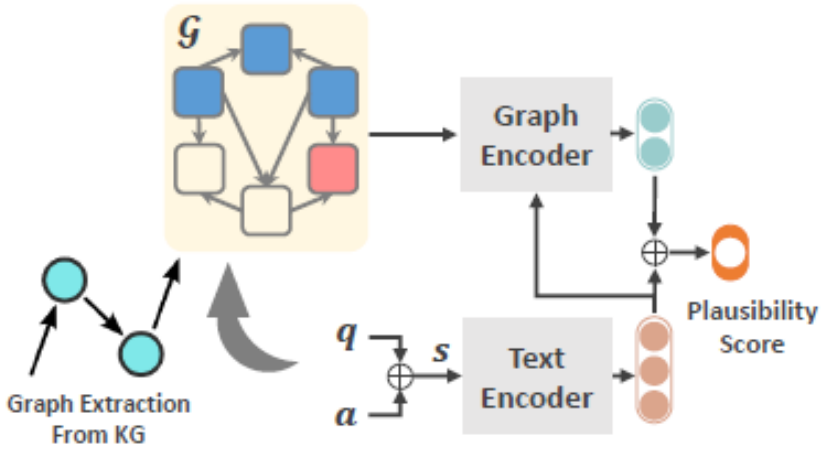
Knowledge-aware PLMs guided by KG

Knowledge-aware KG-enhanced QA

KagNet | CSQA | PG | MHGRN

BERT: Pre-training of deep bidirectional transformers for language understanding (NAACL 19)
Bert-wwm: Pre-Training with Whole Word Masking for Chinese BERT (Arxiv 19)
SpanBERT: Improving Pre-training by Representing and Predicting Spans (TACL 20)
ERNIE-baidu: Enhanced representation through knowledge integration (Arxiv 19)
ERNIE 2.0: A Continual Pre-Training Framework for Language Understanding (AAAI 20)
ERNIE-thu: Enhanced Language Representation with Informative Entities (ACL 19)
K-BERT: Enabling Language Representation with Knowledge Graph (AAAI 20)
KnowBert: Knowledge enhanced contextual word representations (EMNLP 19)
KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation (Arxiv 19)
GLM: Exploiting Structured Knowledge in Text via Graph-Guided Representation Learning (Arxiv 20)
KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning (EMNLP 19)
CSQA: Graph-Based Reasoning over Heterogeneous External Knowledge for Common sense Question Answering (AAAI 20)
PG: Connecting the Dots: A Knowledgeable Path Generator for Commonsense Question Answering (EMNLP 2020 finding)
MHGRN: Scalable Multi-Hop Relational Reasoning for Knowledge-Aware Question Answering (EMNLP 2020)

Unified Architecture in KG-enhanced QA



Graph Encoder: GNN, Relational Network…

Text Encoder: Bert, XLNet…

Next topic:
Graph encoding

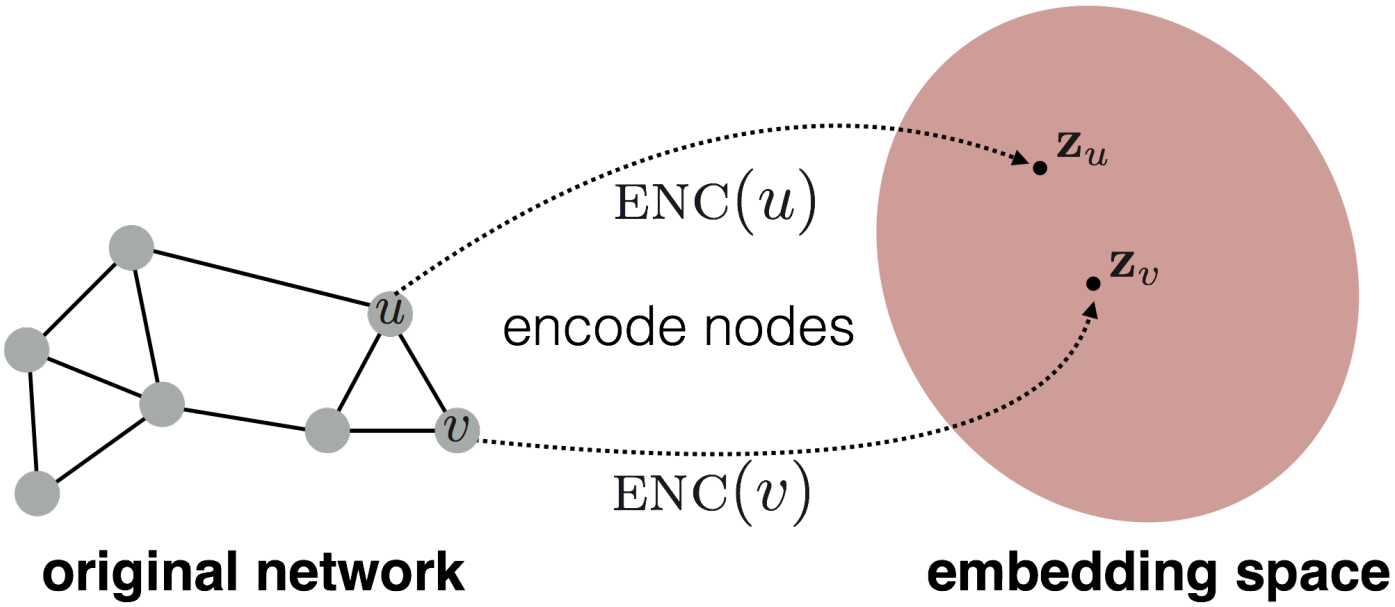
Acknowledgements

- Slides for this presentation are taken from
  - Representation Learning on Networks
    snap.stanford.edu/proj/embeddings-www, WWW 2018
  - Efficient Probabilistic Logic Reasoning with Graph Neural Networks
    Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi & Le Song (slides taken from a presentation by Hengda Shi, Gaohong Liu and Jian Weng)
  - Probabilistic Logic Neural Network for Reasoning, Meng Qu, Jian Tang (slides taken from a presentation by Zijie Huang, Roshni Iyer, Alex Wang)

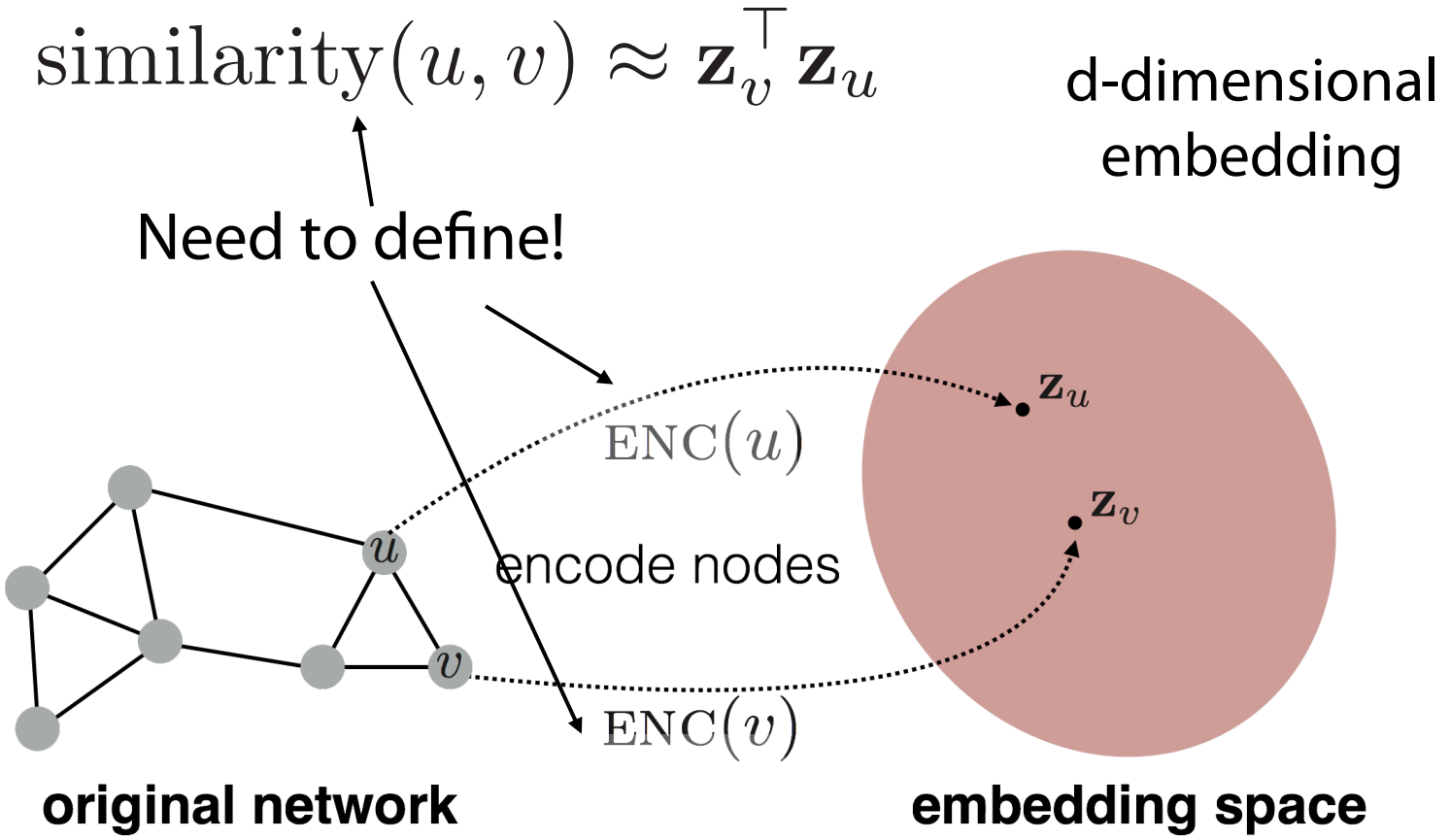
- Slides have been adapted (all faults are mine)

Embedding Nodes of a Graph

- Encode nodes so that …
  - similarity in the embedding space approximates …
  - similarity in the original network



Embedding Nodes of a Graph

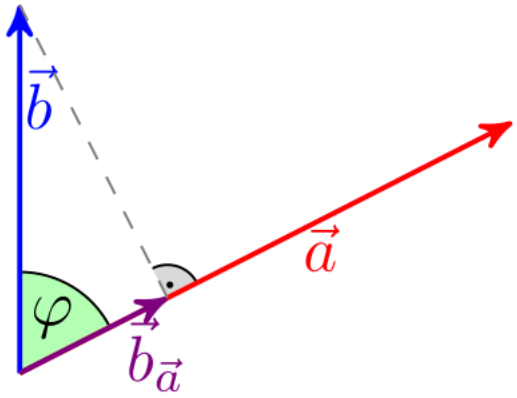


Recap: Dot Product

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \ \|b_a\|$$

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos \ \varphi$$

$$\cos\,(\varphi) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$



Orthogonale Projektion $\vec{b}_{\vec{a}}$ des Vektors $\vec{b}$ auf die durch $\vec{a}$ bestimmte Richtung

Wikipedia

Simple ("Shallow") Embedding Approaches

Vectors with d components (with d being a hyperparameter)

Solve optimization problem

- Select embedding vectors for nodes such that "similar" nodes have similar vectors

Various ways to specify similarity of nodes

- Adjacency-based embedding
- Multi-hop embedding
- Random walk approaches

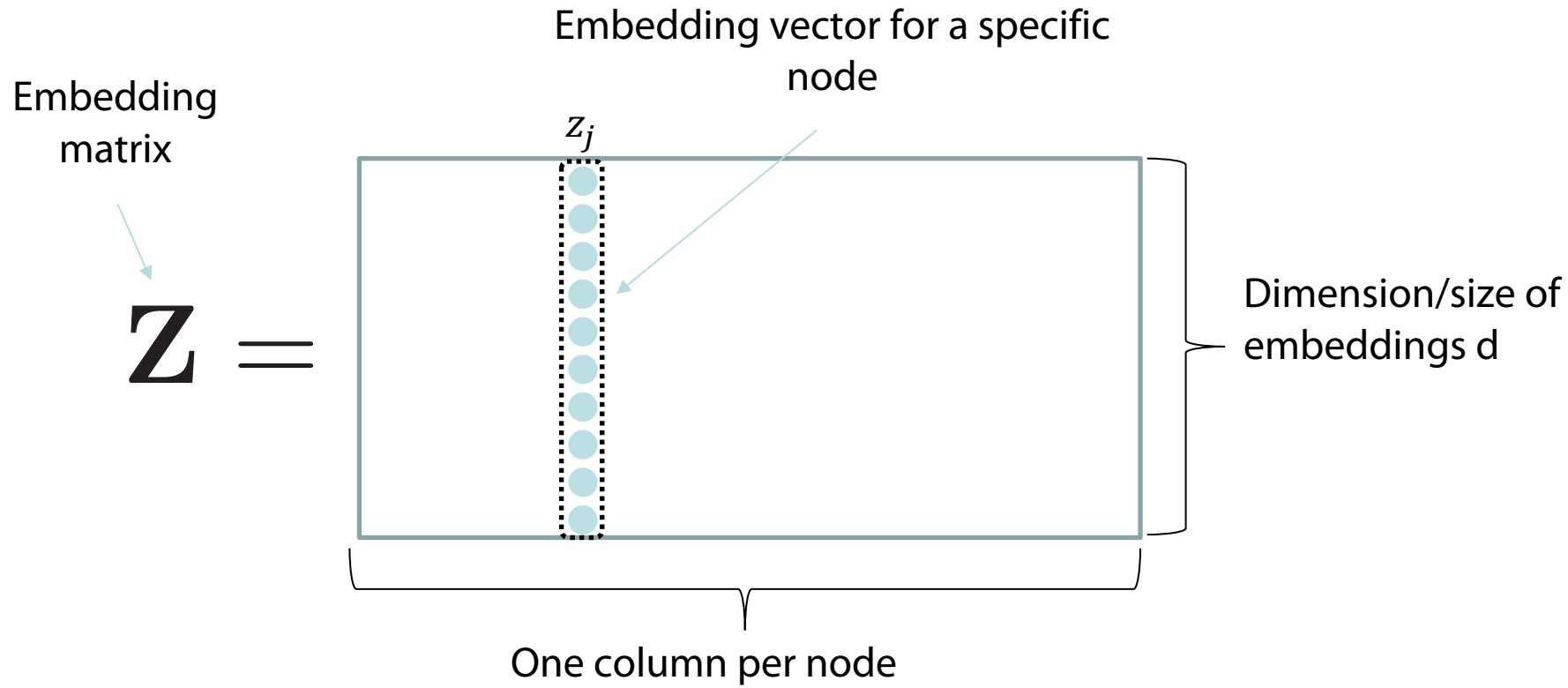
More or less clever approaches, but appropriate similarity features should better be found automatically

$\mathbf{z}_u^\top \mathbf{z}_v \approx$ Probability that *u* and *v* co-occur in a random walk over the network

Hamilton et al. Representation Learning on Graphs: Methods and Applications. IEEE Data Engineering Bulletin on Graph Systems. **2017**.

From "Shallow" to "Deep"

- Shallow: Define features based on selected features



From “Shallow” to “Deep”

- Limitations of shallow encoding:
  - O(|Vd|) parameters needed
    - No parameter sharing
    - Every node has its own unique embedding vector
  - Inherently “transductive” (not inductive)
    - Impossible to generate embeddings for nodes that were not seen during training
  - Does not incorporate node features
    - Many graphs have nodes with features that we can and should leverage

- Need to find embeddings based on holistic view on graph

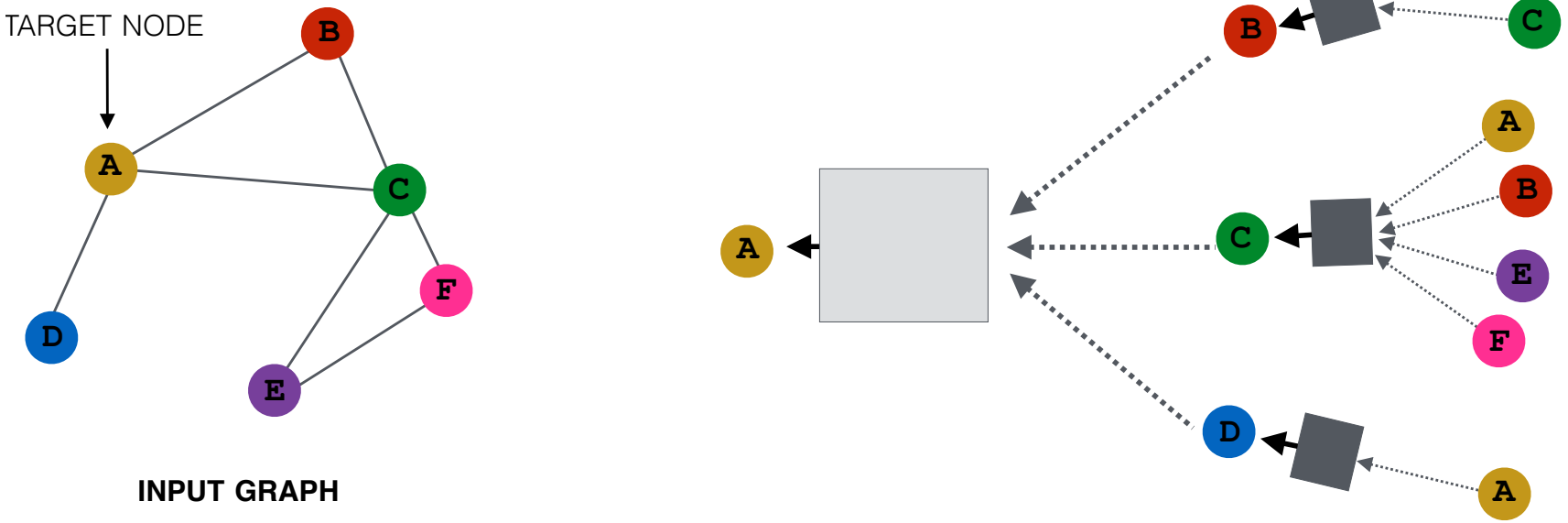
Scarselli et al. The Graph Neural Network Model. IEEE Transactions on Neural Networks. **2005.**

Setup

- Assume we have a graph G:
  - V is the vertex set.
  - $\mathbf{A}$ is the adjacency matrix (assume binary).
  - $\mathbf{X} \in \mathbf{R}^{m \times |V|}$ **is a matrix of nodes and their features.**
    - Categorical attributes, text, image data
      - E.g., profile information in a social network.
    - Node degrees, clustering coefficients, etc.
    - Indicator vectors (i.e., one-hot encoding of each node)

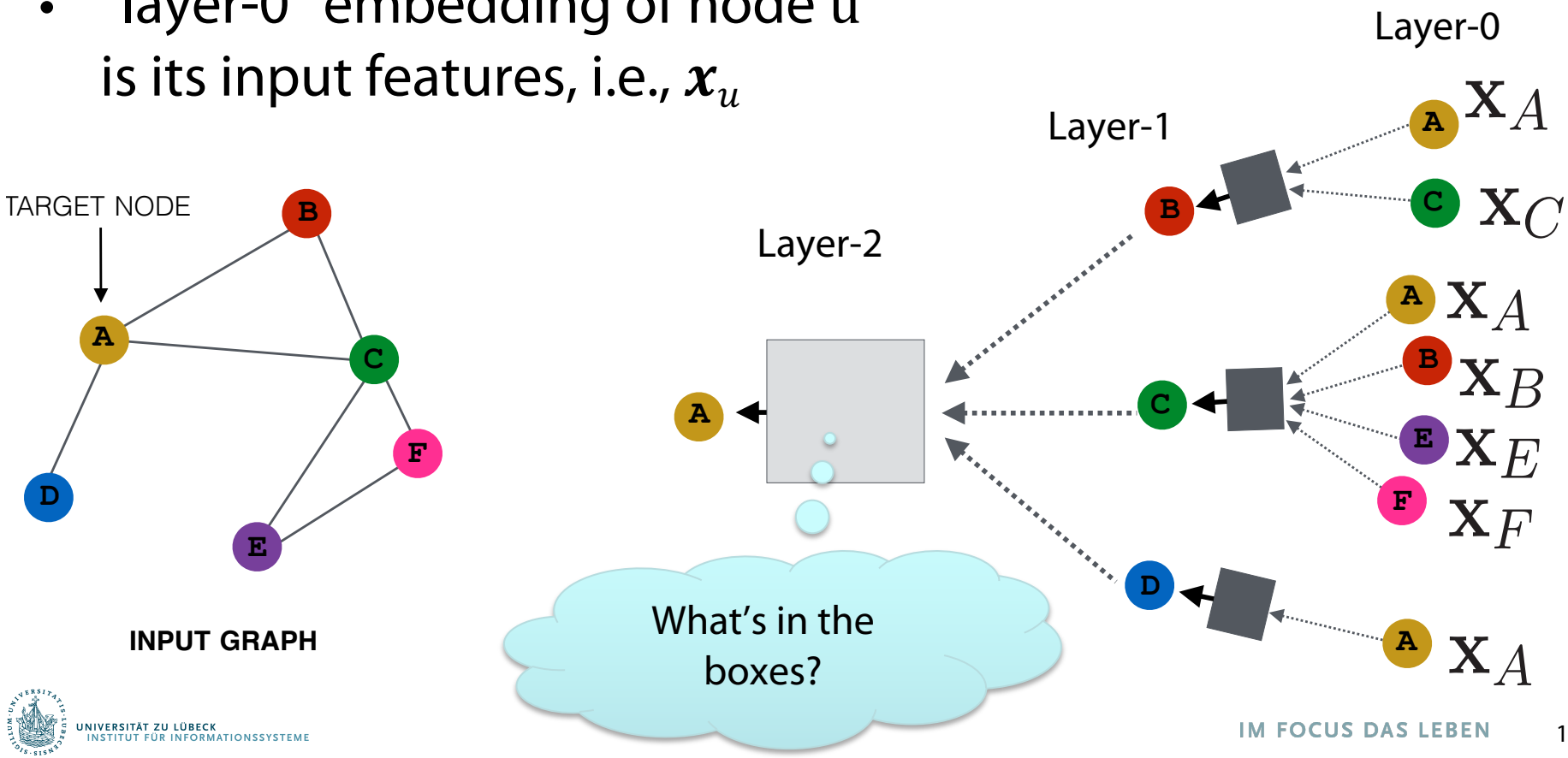
Neighborhood Aggregation

- Key idea
  - Generate node embeddings based on local neighborhoods
  - Nodes aggregate information from their neighbor
  - Computation graph for every node



Neighborhood Aggregation

- Nodes have embeddings at each layer
- Model can be of arbitrary depth
- "layer-0" embedding of node u is its input features, i.e., $\boldsymbol{x}_u$



Graph Networks (GNNs)

- Basic approach: Average neighbor messages and apply a linear transformation with non-linear normalization
- Define a loss function on the embeddings, $\mathcal{L}(z_u)$

Initial "layer 0" embeddings are equal to node features

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

previous layer embedding of *v*

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k > 0$$

kth layer embedding of v

non-linearity (e.g., ReLU or tanh)

average of neighbor's previous layer embeddings

Unsupervised Training

trainable matrices
(i.e., what we learn)

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

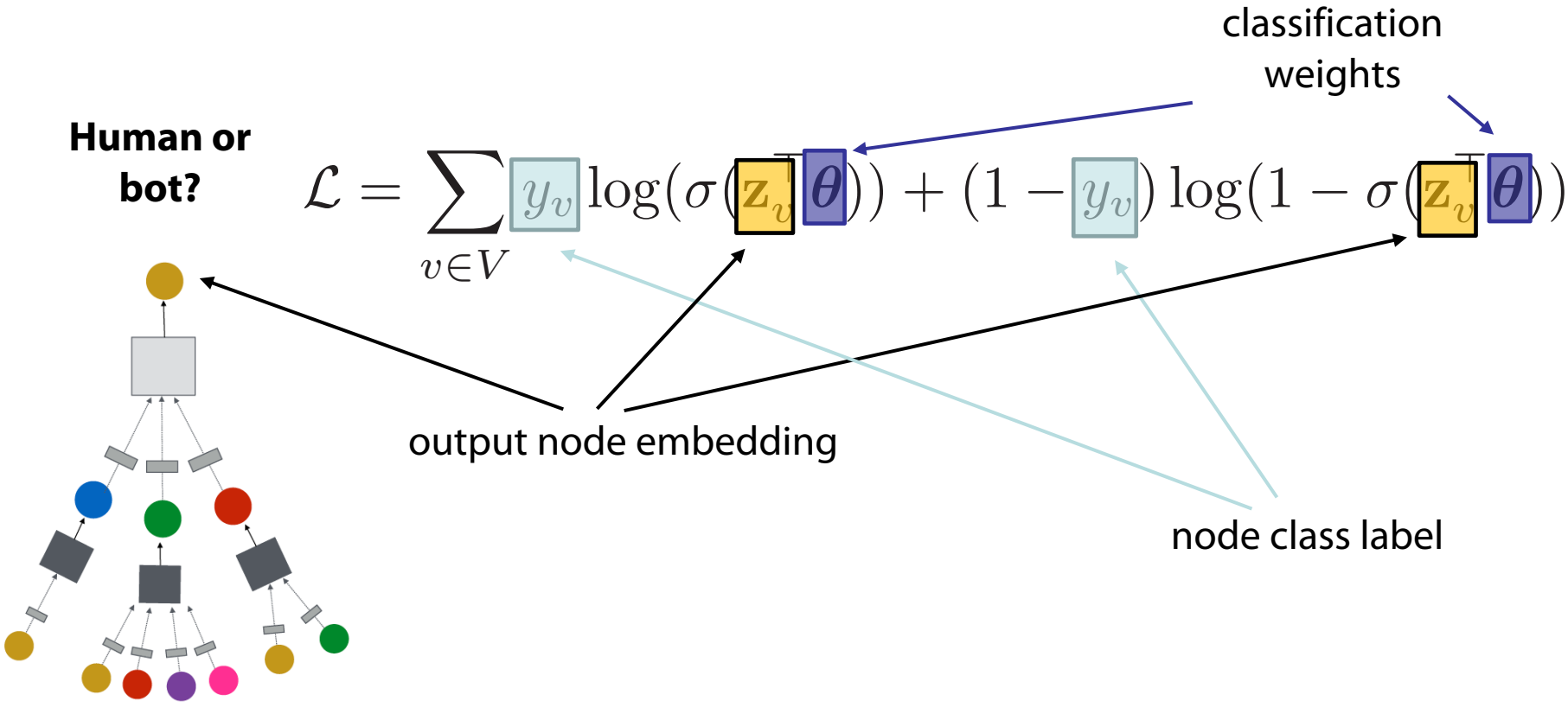
$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k \in \{1, ..., K\}$$

$$\mathbf{z}_v = \mathbf{h}_v^K$$

- After K-layers of neighborhood aggregation, we get output embeddings for each node
- Feed these embeddings into any loss function …
- and run stochastic gradient descent to train the aggregation parameters

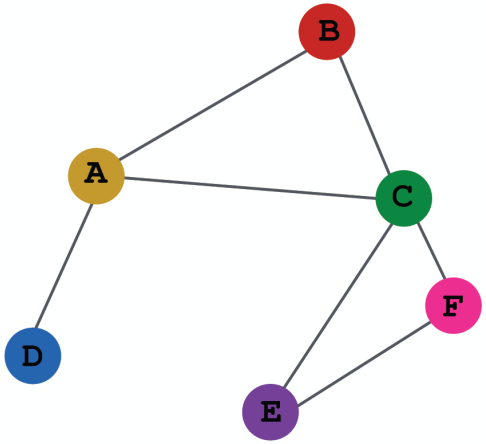
Supervised Training

- E.g., based on node classification $y_v \in \{0, 1\}$:

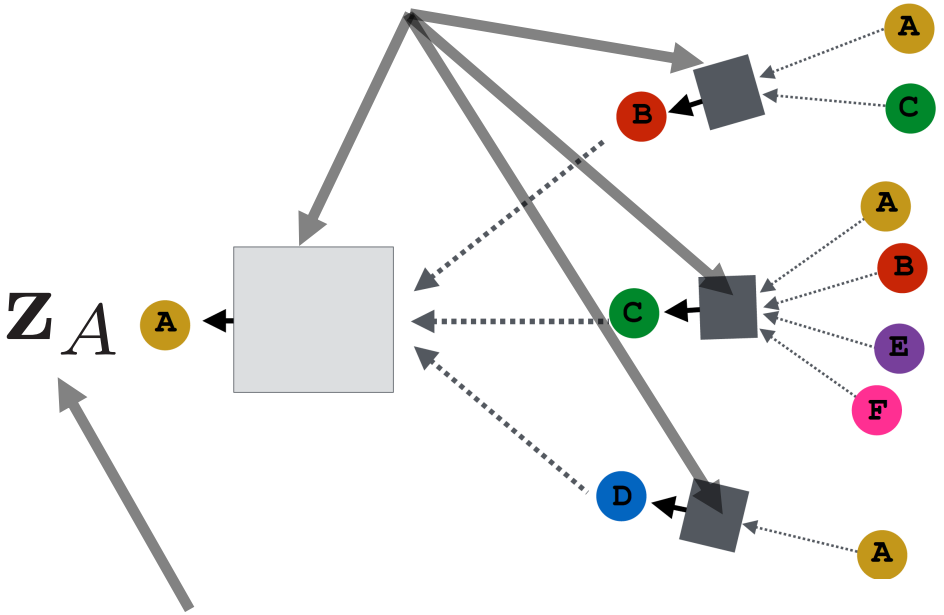


Overview of Model Design

1) Define a neighborhood aggregation function.

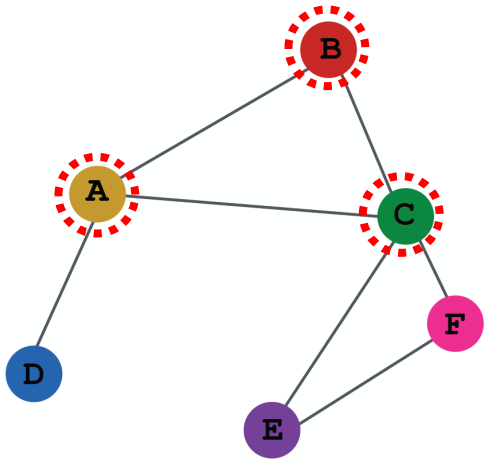


INPUT GRAPH



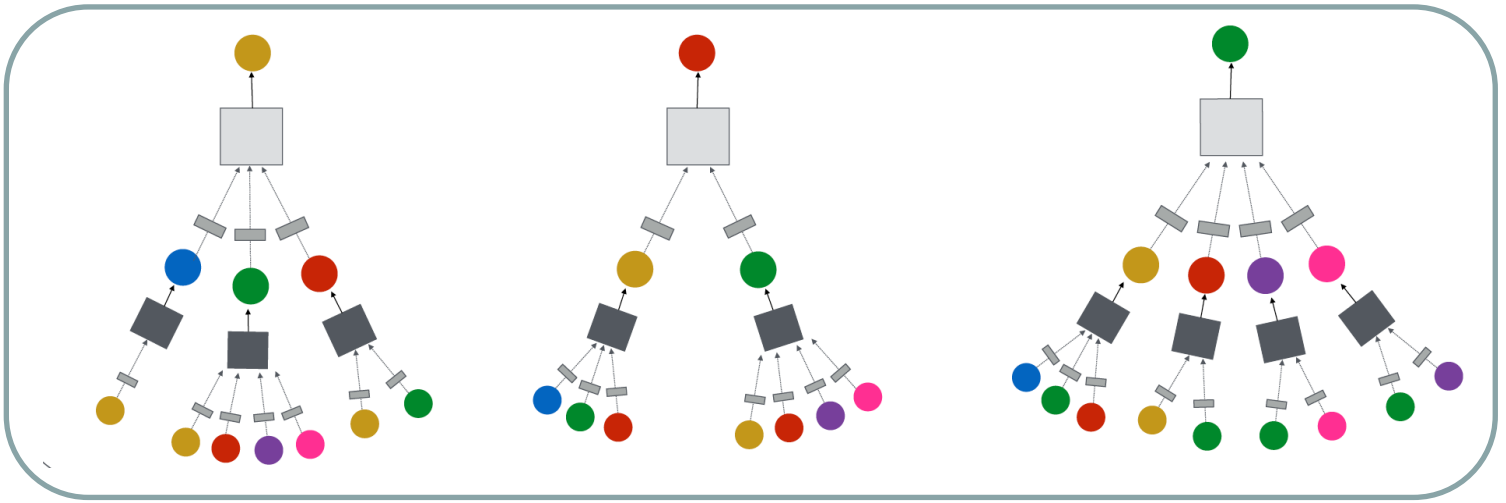
2) Define a loss function on the embeddings, $\mathcal{L}(z_u)$

Overview of Model Design

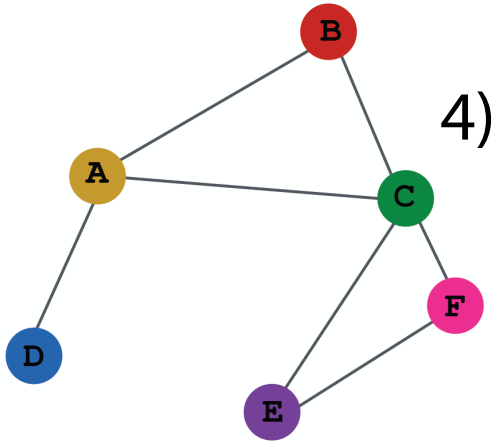


INPUT GRAPH

3) Train on a set of nodes, i.e., a batch of compute graphs



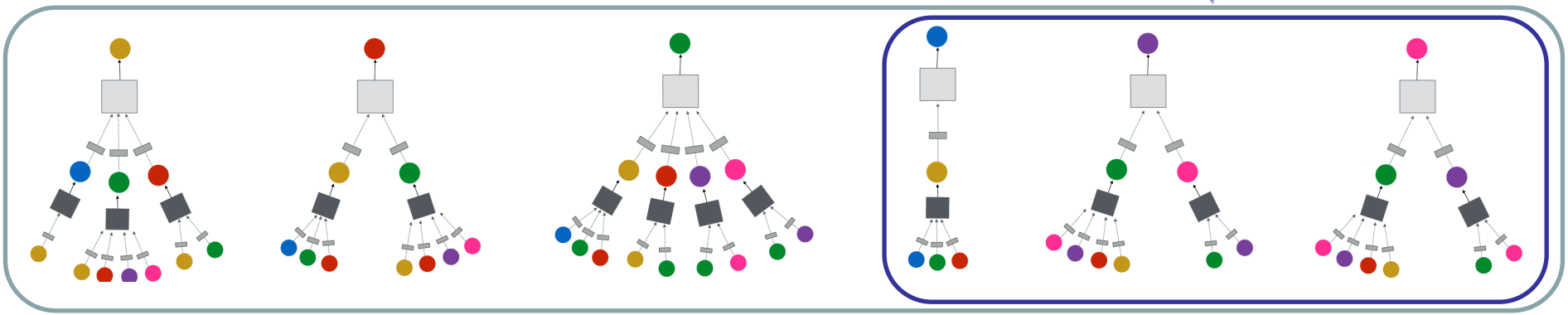
Overview of Model



INPUT GRAPH

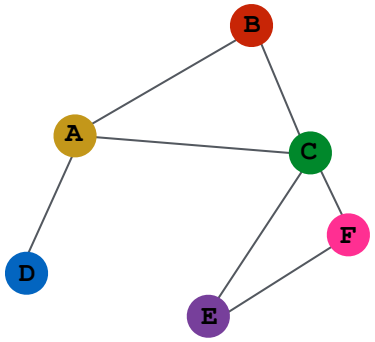
4) Generate embeddings for nodes as needed

Even for nodes we never trained on!!!!

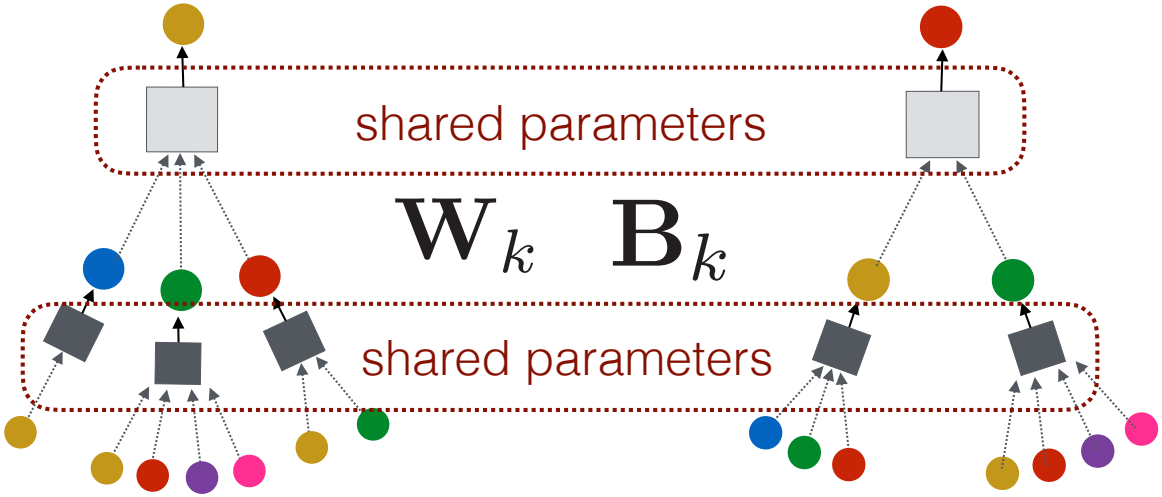


Inductive Capability

- Same aggregation parameters are shared for all nodes.
- Number of model parameters is sublinear in |V| …
- … and we can generalize to unseen nodes!



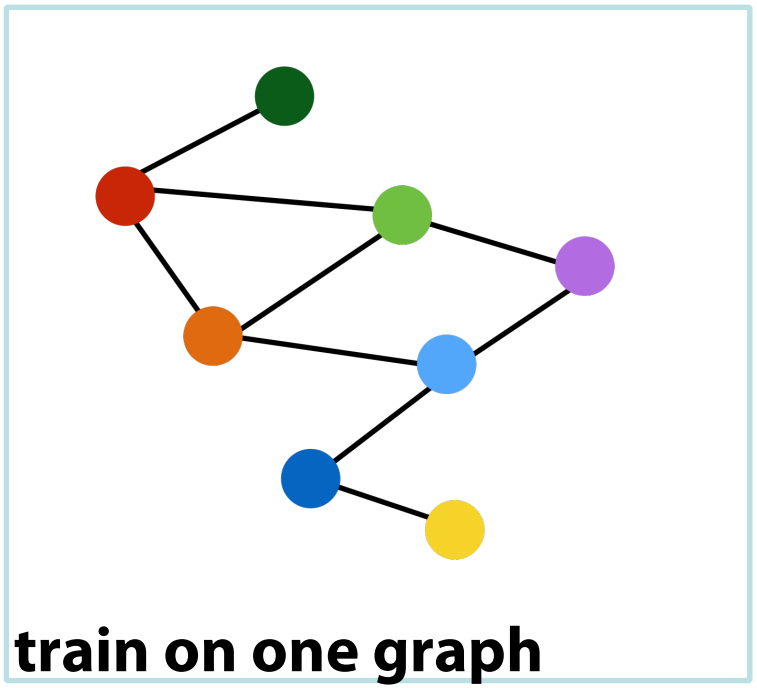
INPUT GRAPH



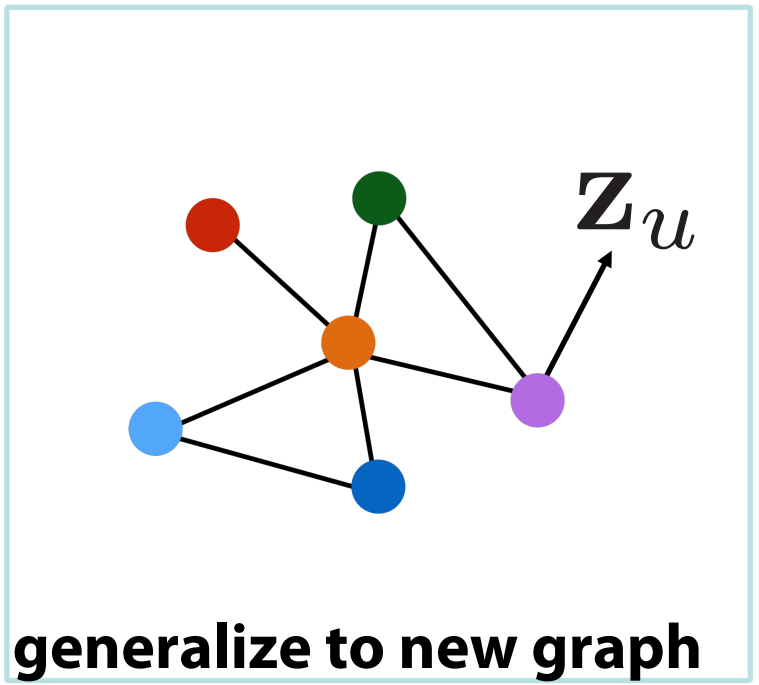
Compute graph for node A

Compute graph for node B

Inductive Capability



train on one graph

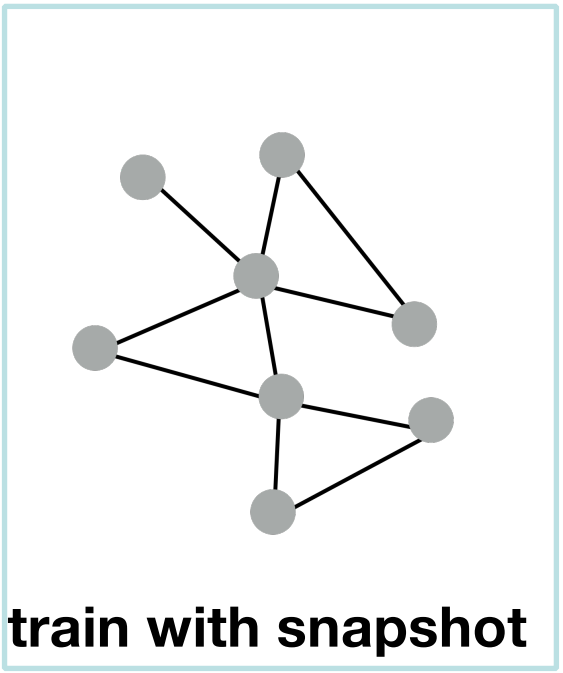


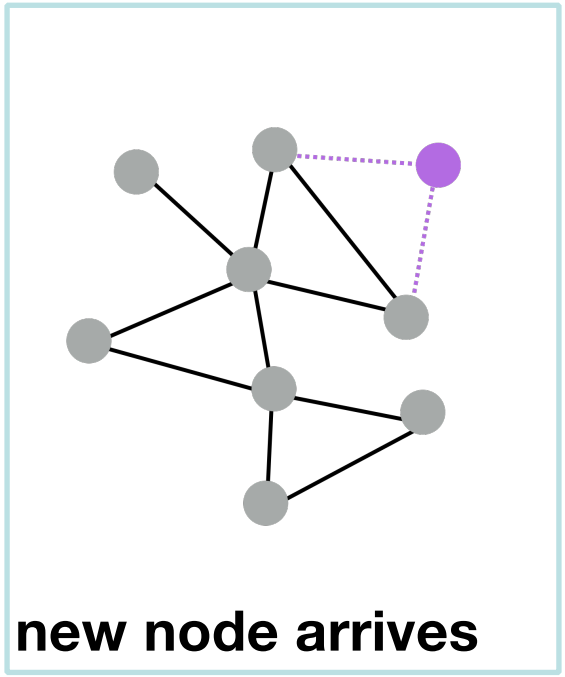
generalize to new graph

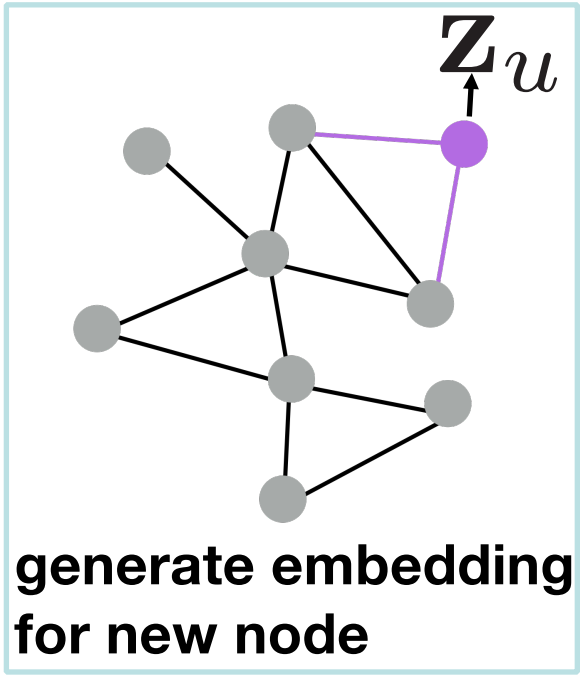
Inductive node embedding → generalize to entirely unseen graphs

e.g., train on protein interaction graph from model organism A and generate embeddings on newly collected data about organism B

Inductive Capability







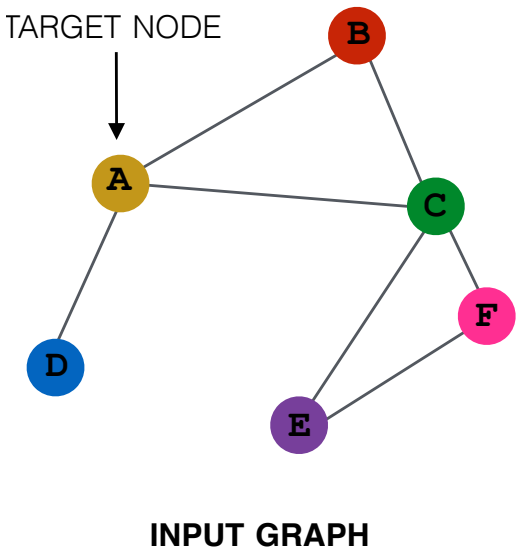
Many application settings constantly encounter previously unseen nodes.

e.g., Reddit, YouTube, GoogleScholar, ….

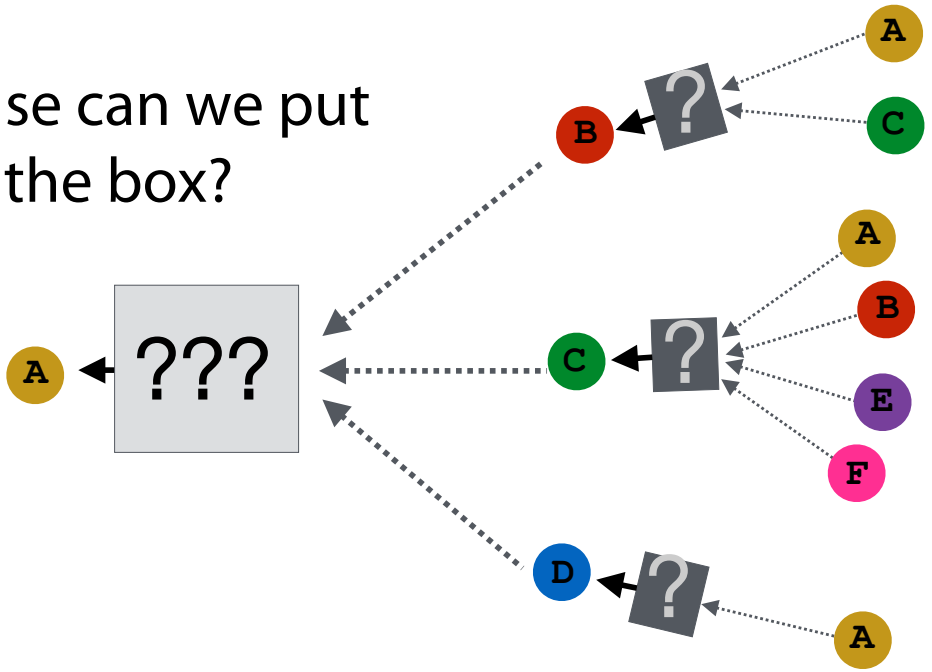
Need to generate new embeddings “on the fly”

Neighborhood Aggregation

- Key distinctions are in how different approaches aggregate messages



What else can we put in the box?



Graph Convolutional Networks (GCNs)

- Slight variation on the neighborhood aggregation idea:

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v) \cup v} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

same matrix for self and neighbor embeddings

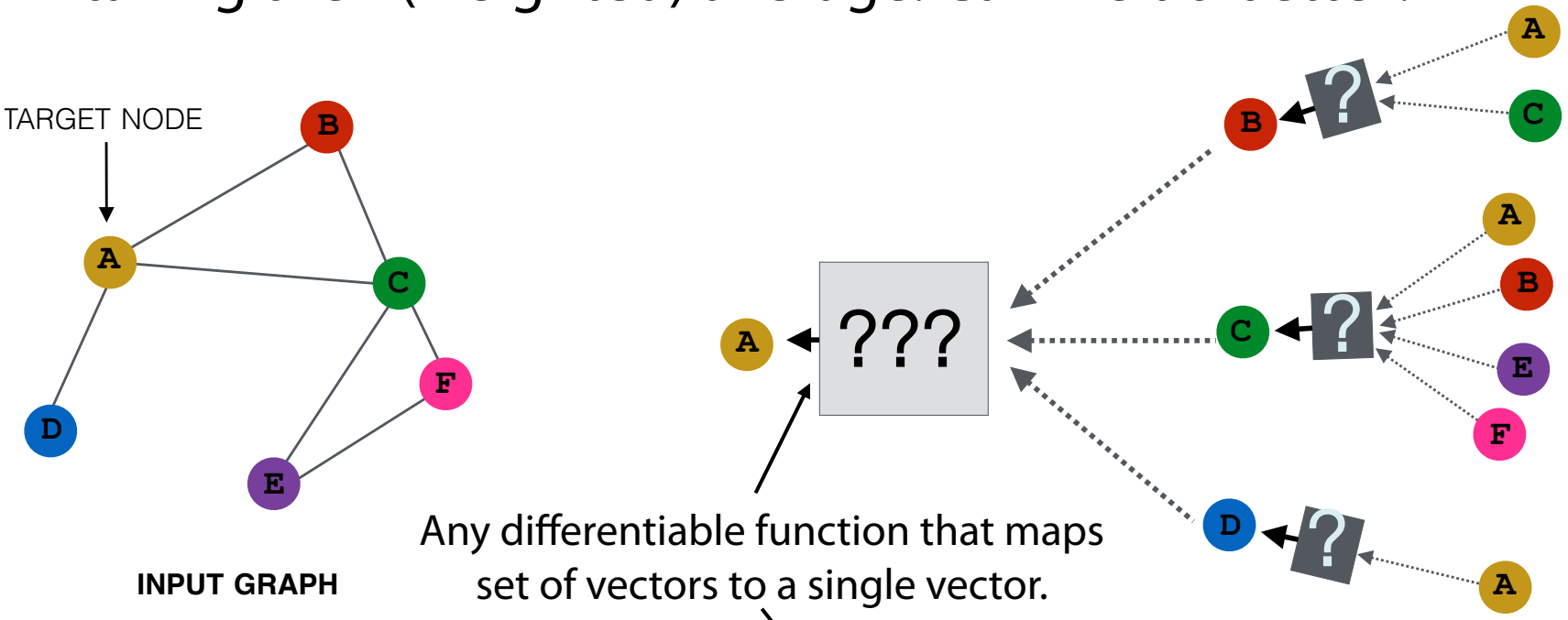
per-neighbor normalization

- Empirically, this configuration to give the best results
  - More parameter sharing
  - Down-weights high degree neighbors

Semi-supervised classification with graph convolutional networks
TN Kipf, M Welling. In Proc. 5th International Conference on Learning Representations (ICLR-17), **2017**.

GraphSAGE (SAmple and aggreGatE)

- So far we have aggregated the neighbor messages by taking their (weighted) average. Can we do better?



$$\mathbf{h}_v^k = \sigma\left(\left[\mathbf{A}_k \cdot \text{AGG}(\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\}), \mathbf{B}_k \mathbf{h}_v^{k-1}\right]\right)$$

concatenate self embedding and neighbor embedding

GraphSAGE Variants

- Mean:

$$\text{AGG} = \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}$$

- Pool:
  - Transform neighbor vectors and apply symmetric vector function

element-wise mean/max

$$\text{AGG} = \gamma\left(\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\}\right)$$

- LSTM-based RNNs:
  - Apply LSTM to random permutation of neighbors (LSTMs work on seqences)

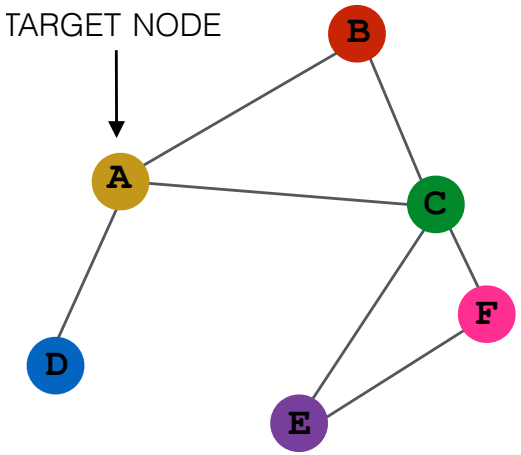
$$\text{AGG} = \text{LSTM}\left([\mathbf{h}_u^{k-1}, \forall u \in \pi(N(v))]\right)$$

- Transformers (attention)?

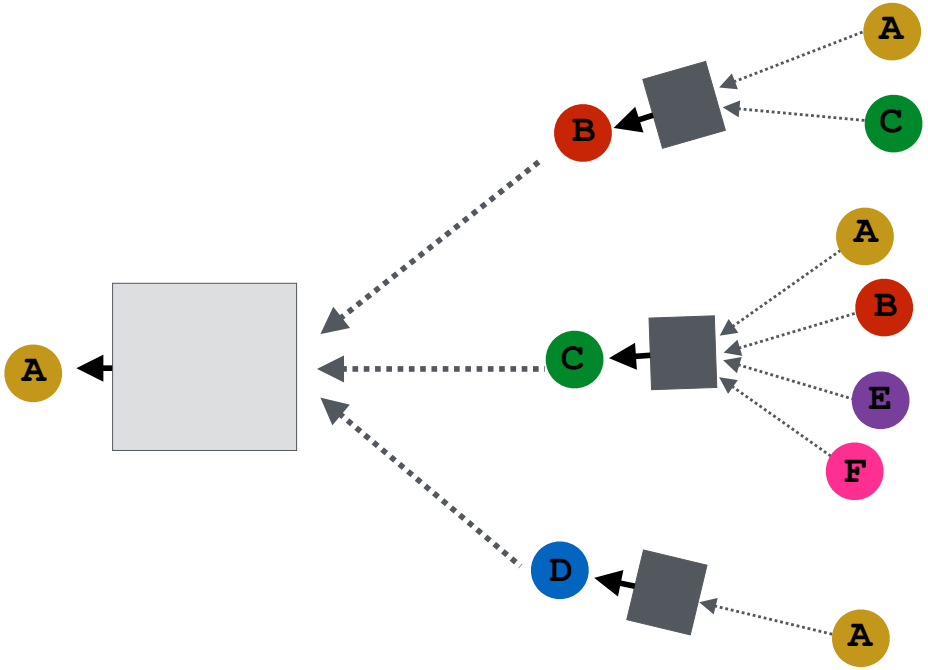
William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In Proc. NIPS'17. **2017**.

Neighborhood Aggregation

- GCNs and GraphSAGE generally only 2-3 layers deep
- What if we want to go deeper?
  - Overfitting from too many parameters.
  - Vanishing/exploding gradients during backpropagation

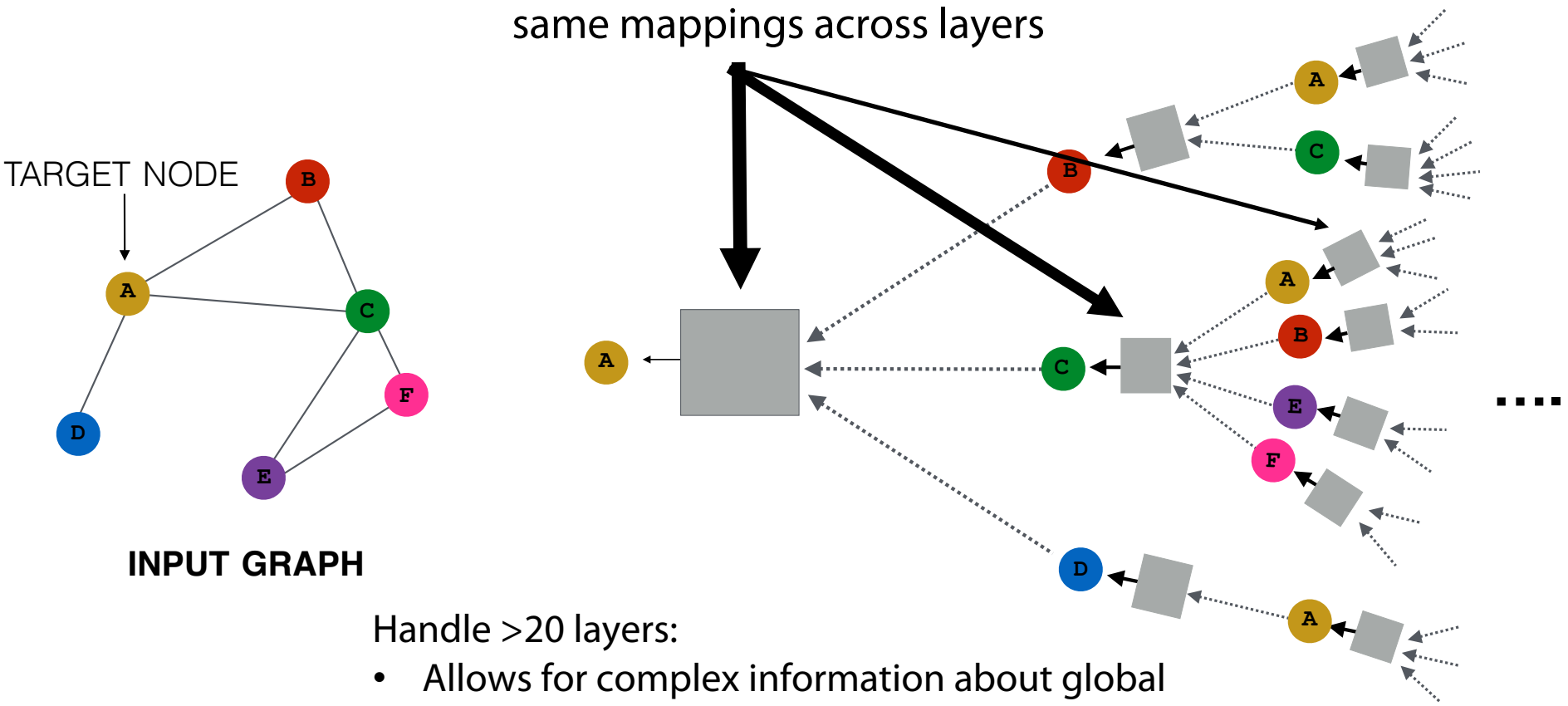


INPUT GRAPH



Gated Graph Networks

- Use techniques from recurrent networks
- Parameter sharing across layers, recurrent state update



Handle >20 layers:

- Allows for complex information about global graph structure to be propagated to all nodes

Neighborhood aggregation with RNN state update

1. Get "message" from neighbors at step k:

$$\mathbf{m}_v^k = \mathbf{W} \sum_{u \in N(v)} \mathbf{h}_u^{k-1}$$

Aggregation function does not depend on k

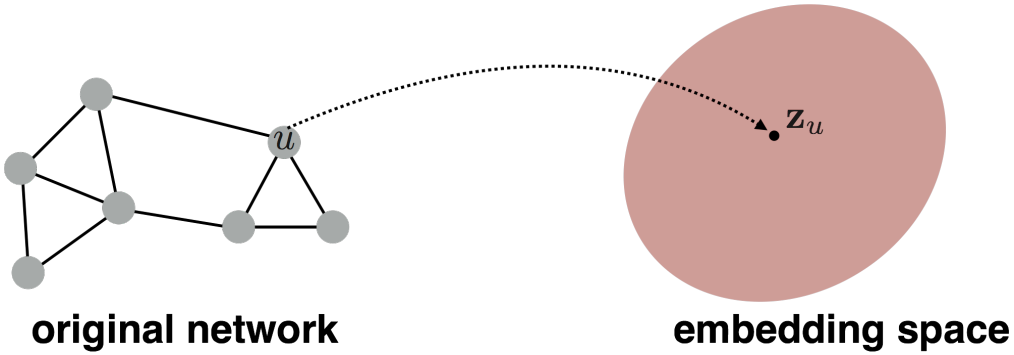
2. Update node "state" using Gated Recurrent Unit (GRU)
   New node state depends on the old state and the message from neighbors:

$$\mathbf{h}_v^k = \mathrm{GRU}(\mathbf{h}_v^{k-1}, \mathbf{m}_v^k)$$

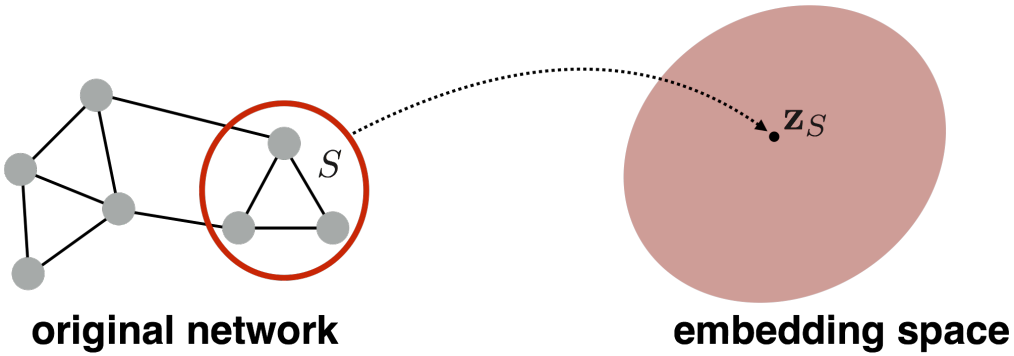
Yujia Li Richard Zemel Marc Brockschmidt Daniel Tarlow,
Gated Graph Sequence Neural Networks
Proceedings of ICLR'16. **2016**.

(Sub)graph Embeddings

- So far we have focused on node-level embeddings…



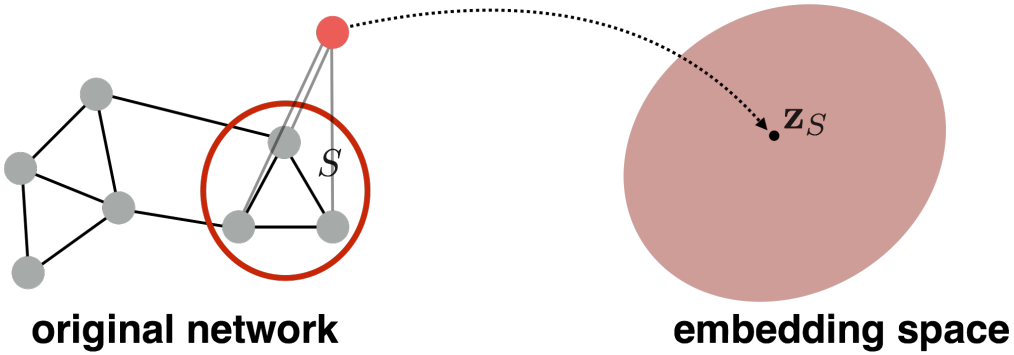
- But what about subgraph embeddings?



(Sub)graph Embeddings

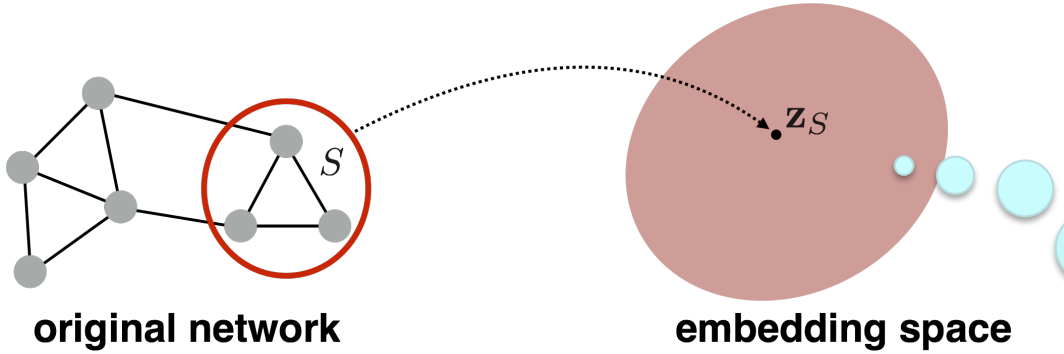
How to embed (sub)graphs with millions or billions of nodes?

- Use representative as a virtual node



Li et al. Gated Graph Sequence Neural Networks. In Proc. ICLR. **2016**.

- Sum or average node embeddings: $\mathbf{z}_S = \sum_{v \in S} \mathbf{z}_v$



How to do the analog of CNN "pooling" on networks?

Duvenaud et al. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In Proc. ICML **2016**.

Summary so far

- Key idea: Generate node embeddings based on local neighborhoods.
  - GraphSAGE
    - Generalized neighborhood aggregation
  - Gated Graph Networks
    - Neighborhood aggregation + recursion (same mappings for a layer) + GRUs
  - Graph Convolutional Networks
    - Average neighborhood information and stack computational networks

Recent Advances in Graph Networks

- Attention-based neighborhood aggregation (Weightings for neighbors)
  - Graph Attention Networks (Velickovic et al., 2018)
  - GeniePath (apaptive receptive paths) (Liu et al., 2018)
- Generalizations based on spectral convolutions (eigen-decomposition of graph Laplacian L)
  - Geometric Deep Learning (Bronstein et al., 2017)
  - Mixture Model CNNs (Monti et al., 2017)
- Speed improvements via subsampling
  - FastGCNs (Chen et al., 2018)
  - Stochastic GCNs (Chen et al., 2017)

Graph Networks, Embeddings, and KGs

- Graph networks allow for the computation of embeddings for nodes in a KG
- With embeddings, existence of links between nodes can be estimated (KG completion)
  - See also, e.g., node2vec
- If nodes originate from words …
- … we have another way to embed nodes
  - See also, e.g., word2vec
  - KG completion based on word embeddings

node2vec: Scalable Feature Learning for Networks. A. Grover, J. Leskovec. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), **2016**