Word Embeddings & Graph Neural Networks for Automatic Reasoning over Knowledge Graphs

Davide Buffelli

Outline

- **Introduction to Graph Embeddings**
- **Past Approaches**
- **Graph Neural Networks**
    - **General Message Passing Framework**
    - **Graph Convolutional Networks**
    - **GraphSage**
    - **Real-World Use Cases**
- **Paper Presentation: “Infusing Knowledge into the Textual Entailment Task Using Graph Convolutional Networks”**

Notation

Graph $G = (V, E)$ ➡ (A, X)

$A =$ **Adjacency Matrix**

$A \in \mathbb{R}^{n \times n}$, $a_{i,j} = \begin{cases} 1 & \textbf{if } (i,j) \in E \\ 0 & otherwise \end{cases}$

$\tilde{A} = A + I$ **(with self loops)**

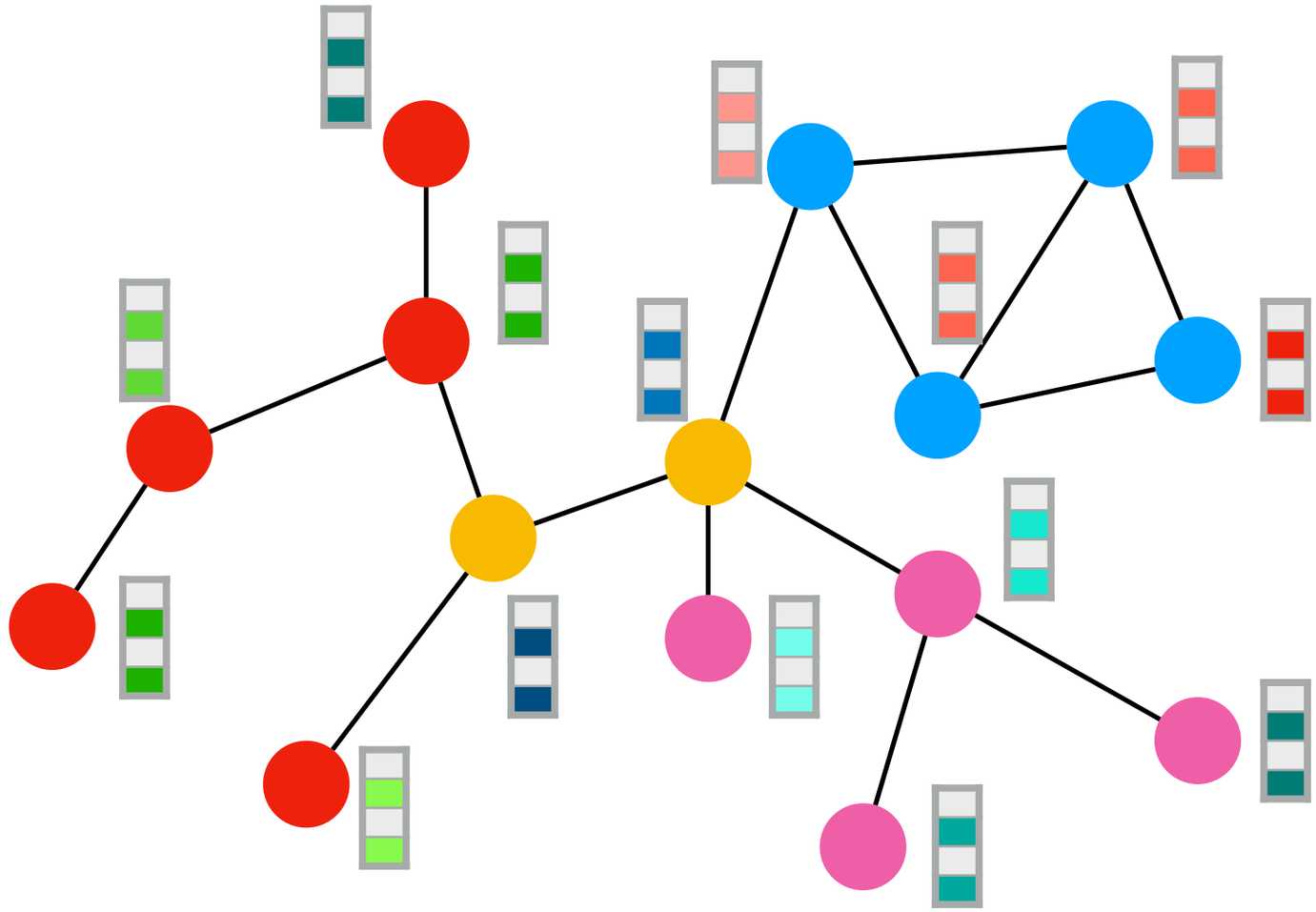
$X =$ **Feature Matrix**

$X \in \mathbb{R}^{n \times d}$

$D =$ **Degree Matrix**

$D \in \mathbb{R}^{n \times n}, d_{i,i} = \sum_{j} a_{i,j}$

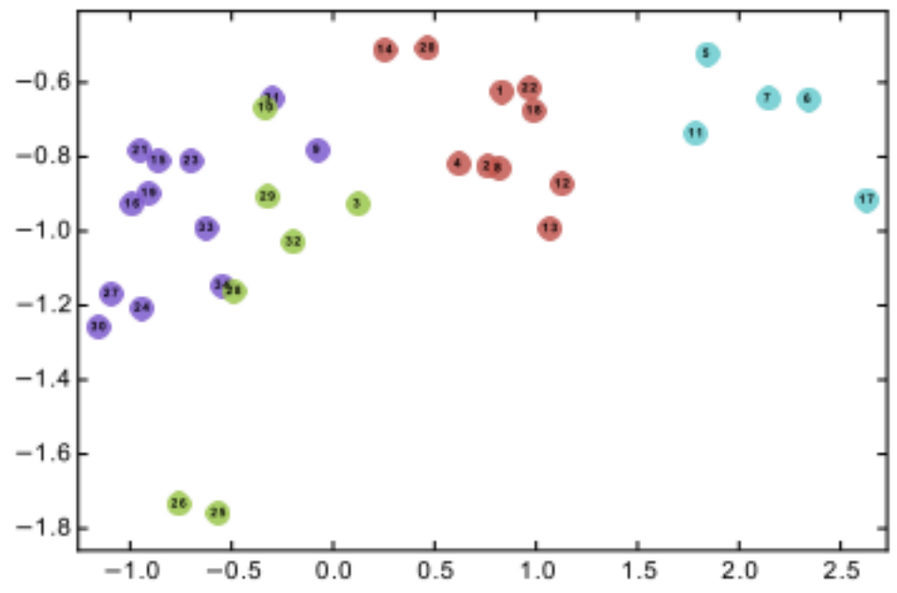
$\tilde{D}, \tilde{d}_{i,i} = \sum_{j} \tilde{a}_{i,j}$



Graph Embeddings

Goal: assign a *meaningful* vector representation to each node

(a) Input: Karate Graph



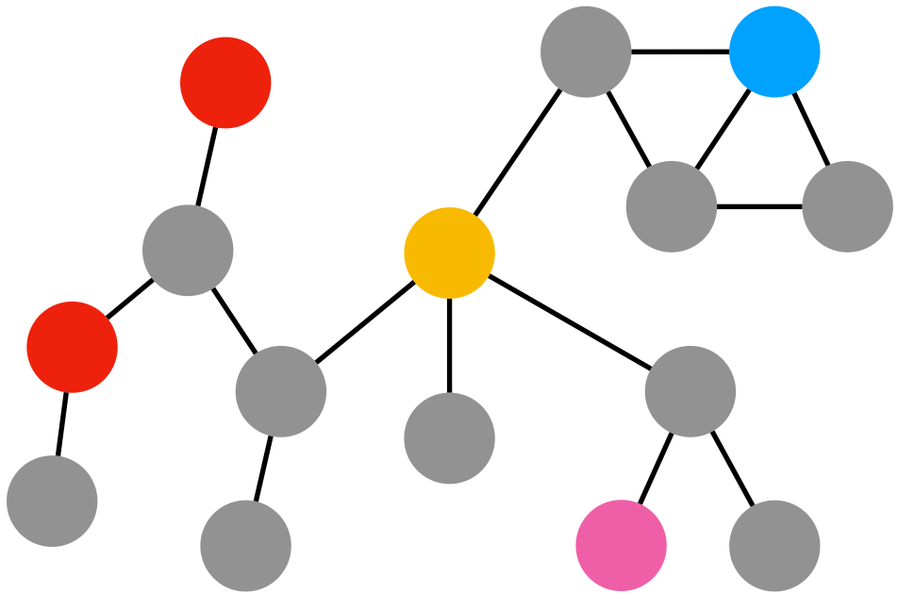
(b) Output: Representation

Applications

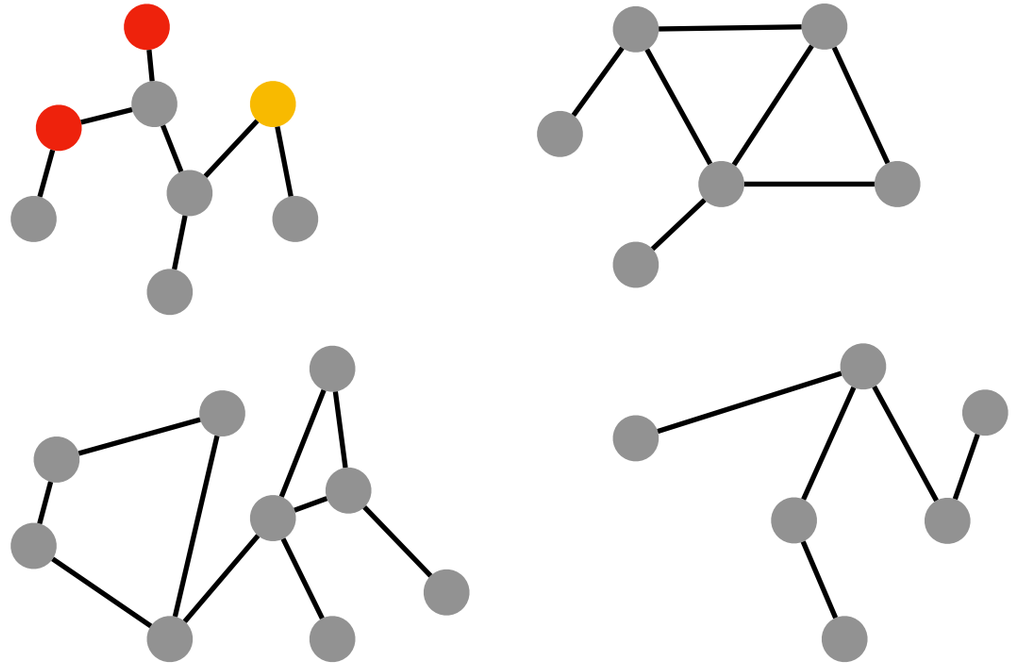
- Node Classification
- Link Prediction
- Graph Classification

Recommender Systems,
Social Network Analysis,
Community Detection,
Transaction Networks (e.g. anomaly detection),
chemoinformatics (e.g. protein classification),
…

Two Scenarios:



Transductive



Inductive

Past Approaches

First Approaches

View the problem as a *dimensionality reduction* task.

Input: Adjacency Matrix (or graph-related matrix e.g. all-pairs shortest path matrix)

“Basic” Techniques: Principal Components Analysis (PCA), Multidimensional scaling (MDS)

“**Advanced” Techniques:** preserve some properties obtained from the graph (e.g. pairwise distances)

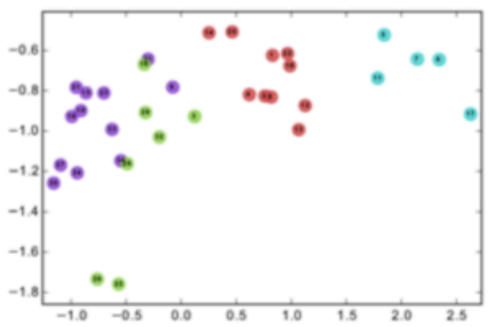
- Isomap - *A Global Geometric Framework for non-linear Dimensionality Reduction, (Science, 2000)*
  https://web.mit.edu/cocosci/Papers/sci_reprint.pdf
- Local Linear Embeddings (LLE) - *Nonlinear dimensionality reduction by locally linear embedding, (Science, 2000)*
  http://www.robots.ox.ac.uk/~az/lectures/ml/lle.pdf
- Laplacian eigenmaps (LE) - *Laplacian eigenmaps and spectral techniques for embedding and clustering, (NeurIPS 2002)*
  http://web.cse.ohio-state.edu/~belkin.8/papers/LEM_NIPS_01.pdf

Random Walk Approaches

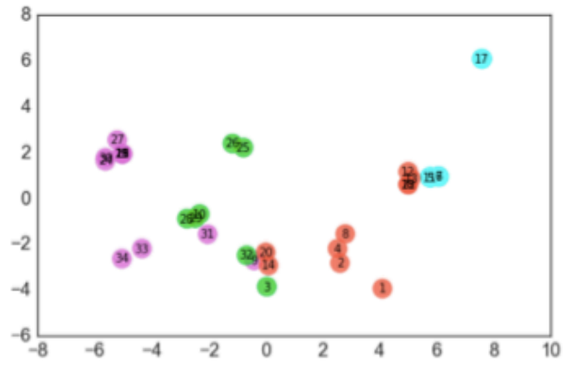
DeepWalk

DeepWalk: Online learning of social representations, (KDD 2014) https://arxiv.org/abs/1403.6652

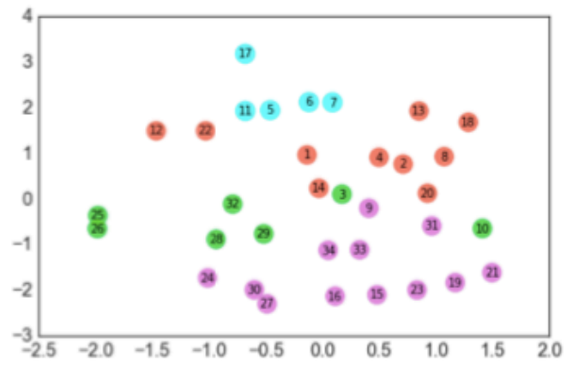
Idea: a random walk is a phrase and each node in the random walk is a word in the phrase -> use the SkipGram model (*word2vec*)



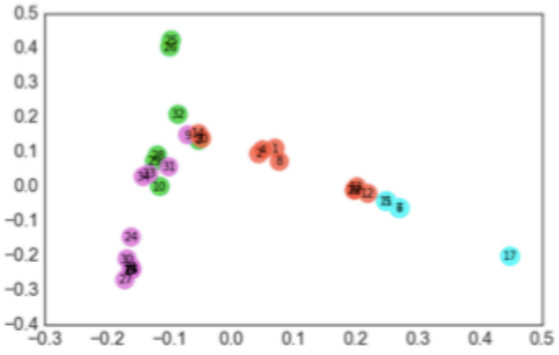
(a) Output: DeepWalk



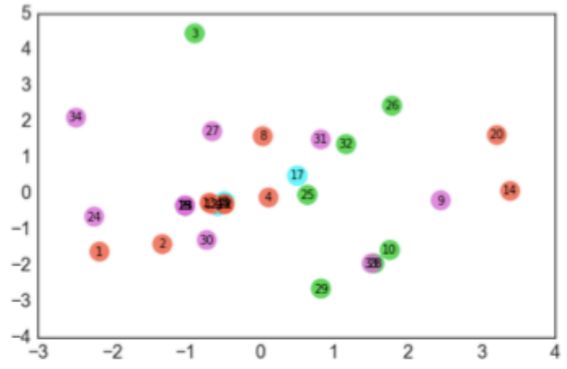
(b) Output: PCA



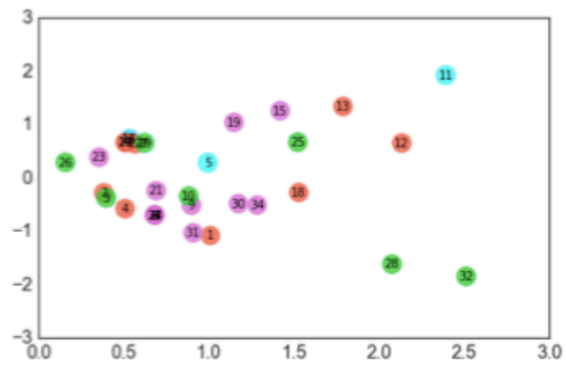
(c) Output: MDS



(d) Output: LLE



(e) Output: LE



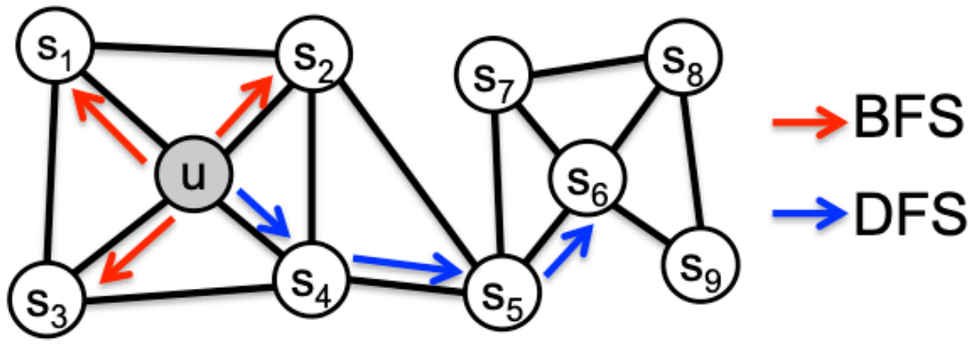
(f) Output: SVD

Random Walk Approaches

node2vec

node2vec: Scalable Feature Learning for Networks, (KDD 2016) https://arxiv.org/abs/1607.00653

introduces a biased random walking procedure which combines BFS style and DFS style neighborhood exploration



Planetoid

Revisiting Semi-Supervised Learning with Graph Embeddings, (ICML 2016) https://arxiv.org/abs/1603.08861

Introduces an inductive variant and is capable of taking node features into consideration

Graph Neural Networks

Graph Neural Networks

Message Passing Framework

Neural Message Passing for Quantum Chemistry, (ICML 2017) https://arxiv.org/abs/1704.01212

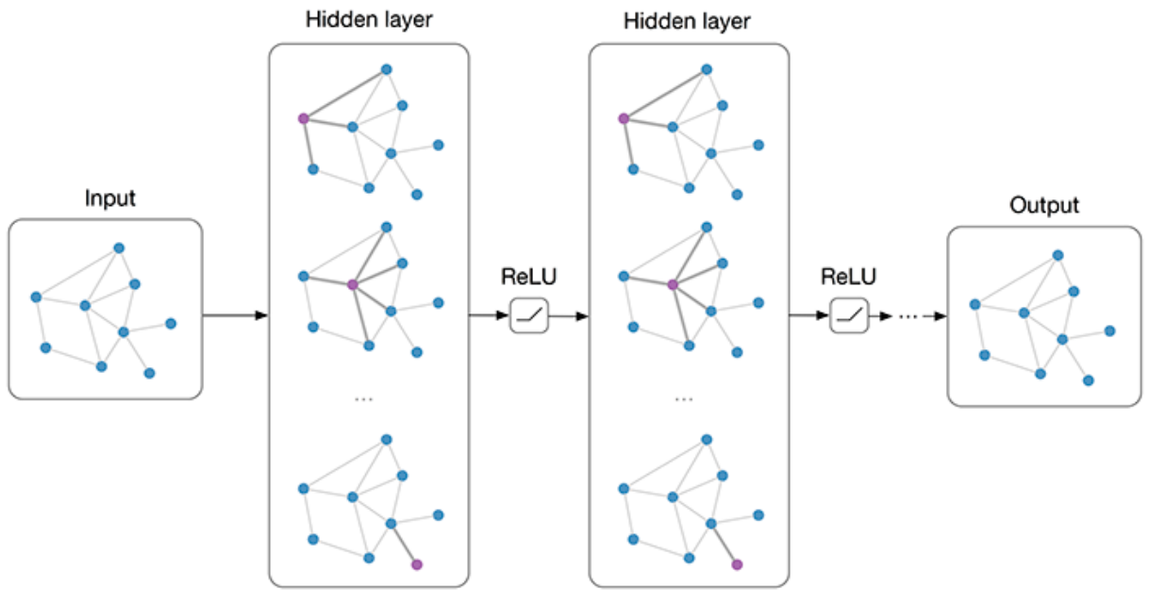
Most models can be expressed in the following 3-steps formulation ($H^0 = X$, $l = 1,..,L$):

Create the message: $m_{ij}^l = MSG(h_i^{l-1}, h_j^{l-1}, e_{ij})$

Aggregate messages from neighbours: $M_i^l = AGG(\{m_{ij}^l | v_j \in N(v_i)\})$

Update node representations: $h_i^l = UPDATE(\{M_i^l, h_i^{l-1}\})$

Final embedding of i-th node: h_i^L

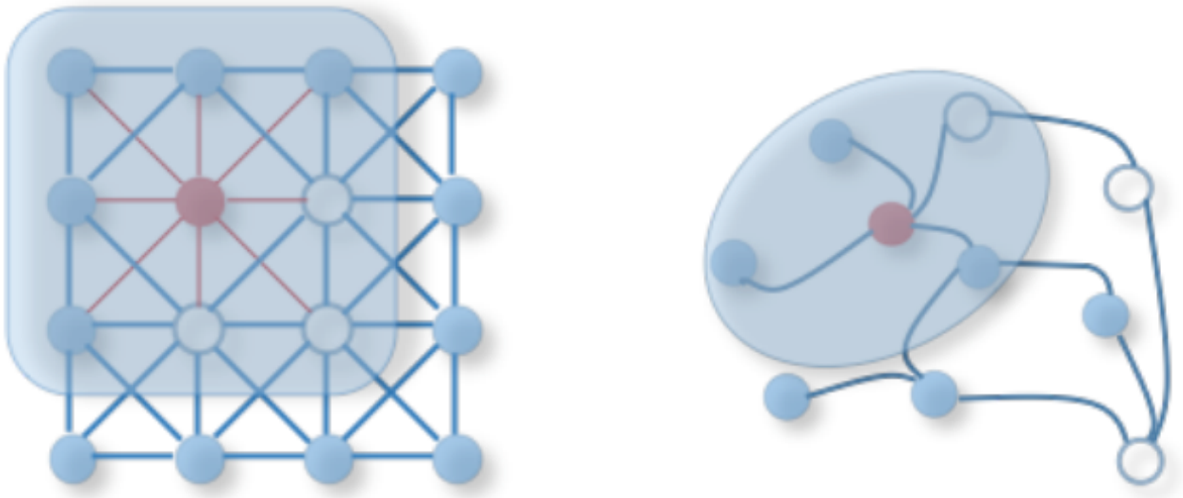


An aggregation process of k iterations makes use of the subtree structures of height k rooted at every node

Graph Convolutional Networks

Graph Convolutional Networks

Idea: generalize Convolutional Neural networks (CNNs) to the graph domain



Desired Properties:

- Exploit local connectivity
- Exploit compositionality of data
- Weight-sharing between nodes/neighborhoods
- Computational and storage efficiency

Spectral-based Methods

Laplacian Matrix: $L = D - A$

Normalized Laplacian matrix: $\mathbf{L} = \mathbf{I_n} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$

Can be factored: $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$

Graph Fourier Transform: $\mathscr{F}(\mathbf{x}) = \mathbf{U}^T\mathbf{x}$

Graph Convolution with filter g:

$$\begin{aligned}\mathbf{x} *_G \mathbf{g} &= \mathscr{F}^{-1}(\mathscr{F}(\mathbf{x}) \odot \mathscr{F}(\mathbf{g})) \\ &= \mathbf{U}(\mathbf{U}^T\mathbf{x} \odot \mathbf{U}^T\mathbf{g})\end{aligned} \tag{1}$$

where $\odot$ denotes the Hadamard product. If we denote a filter as $\mathbf{g}_\theta = diag(\mathbf{U}^T\mathbf{g})$, then the graph convolution is simplified as

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}\mathbf{g}_\theta\mathbf{U}^T\mathbf{x} \tag{2}$$

Spectral-based methods follow this definition. The difference is in the choice of filter g (usually a learnable filter).

Drawbacks of Spectral Methods

Computationally expensive

Filters are domain dependent

Filters are not localized in space (solved by ChebNet & GCNs)

Load the whole graph into memory

Any change in the graph means a change in the eigenbasis

No clear definition of the Laplacian matrix on directed graphs

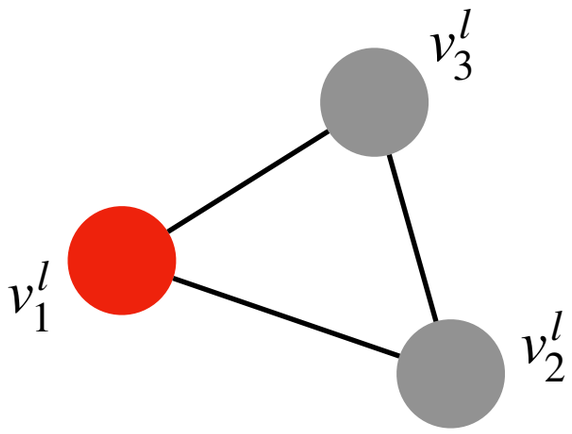
Graph Convolutional Networks (GCN)

Semi-supervised classification with graph convolutional networks, (ICLR 2017) https://arxiv.org/abs/1609.02907

Propagation rule:

$$H^{(0)} = X$$

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$



$$v_1^{l+1} = \sigma\left[\left(\frac{v_1^l}{deg(v_1)+1} + \frac{v_2^l}{\sqrt{deg(v_1)+1}\sqrt{deg(v_2)+1}} + \frac{v_3^l}{\sqrt{deg(v_1)+1}\sqrt{deg(v_3)+1}}\right)W^{(l)}\right]$$

$$\sigma = \textbf{ReLU}(x) = \max(0,x)$$

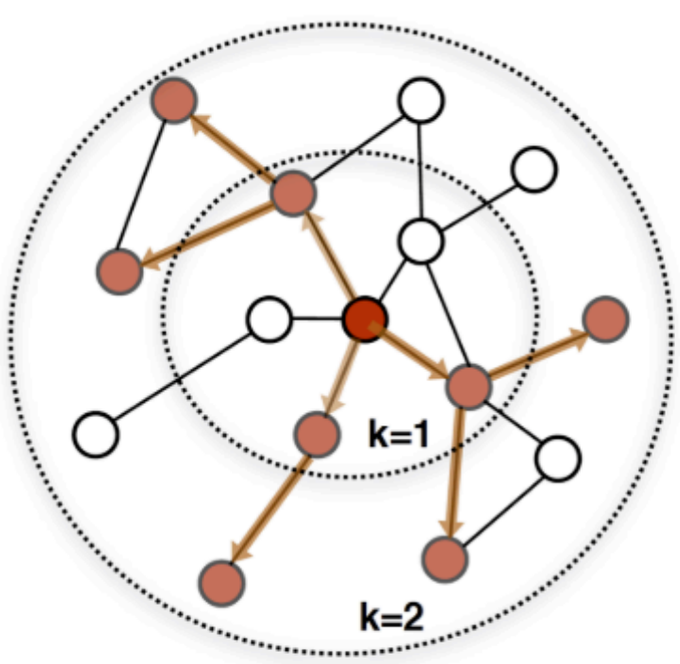
Nice Mathematical properties:

- **1st order Chebyshev polynomial approximation of spectral graph convolutions**

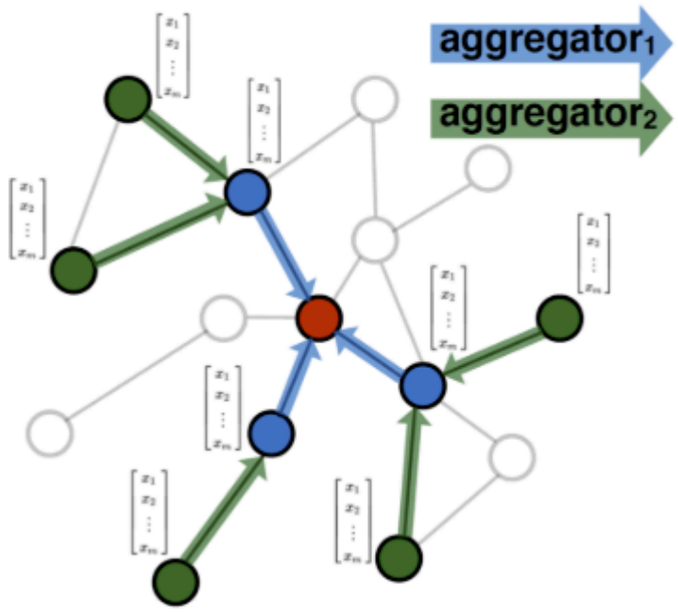
GraphSage

Inductive Representation Learning on Large Graphs, (NeurIPS 2017) https://arxiv.org/abs/1706.02216

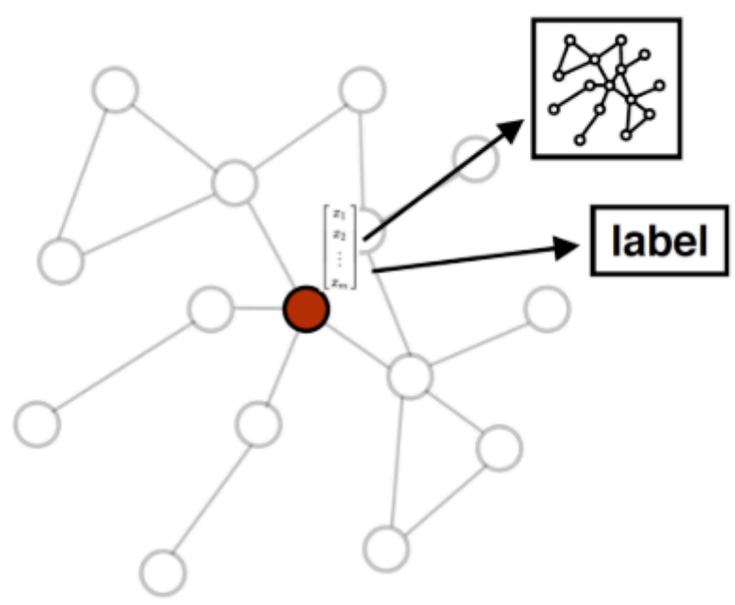
- **First Inductive GCN Framework**
- **Introduced the Sample and Aggregate strategy**
- **Train a set of aggregator functions instead of an embedding vector for each node**



1. Sample neighborhood



2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information

GraphSage

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;

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

  for $v \in \mathcal{V}$ **do**

    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;

    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$

  end

  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$

end

$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

Real-World Use Cases

Pinterest's recommendation system (3 billion nodes and 18 billion edges)

Graph Convolutional Neural Networks for Web-Scale Recommender Systems, (KDD 2018) *https://arxiv.org/abs/1806.01973*

Uber's recommendation system

https://eng.uber.com/uber-eats-graph-learning/

Twitter Cortex (team led by Micheal M. Bronstein)

https://blog.twitter.com/en_us/topics/company/2019/Twitter-acquires-Fabula-AI.html
https://cortex.twitter.com/en/research.html

Alibaba's recommendation system

AliGraph: A Comprehensive Graph Neural Network Platform (KDD 2019) *https://arxiv.org/abs/1902.08730*

KDD 2019 Applied Data Science

https://www.zdnet.com/article/apple-alibaba-amazon-and-the-gang-promote-state-of-the-art-in-ai-and-knowledge-discovery-with-graphs/

Paper Presentation:
“Infusing Knowledge into the Textual Entailment Task Using Graph Convolutional Networks”

Kapanipathi et al., AAAI 2020 https://arxiv.org/abs/1911.02060

The Textual Entailment Task

Input

Text (or Premise)
e.g. “Maurita and Jade both were at the scene of the car crash.”

Hypothesis
e.g. “Multiple people saw the accident.”

Output

Positive Entailment
Hypothesis can be proven by premise

Neutral Entailment
Premise says nothing about the truthness of the hypothesis

Negative Entailment
Premise contradicts the hypothesis

Applications question answering, information extraction, summarization, evaluation of machine translation systems

The Textual Entailment Task

Demo *https://demo.allennlp.org/textual-entailment/*

Demo Usage

Enter text or

Choose an example...

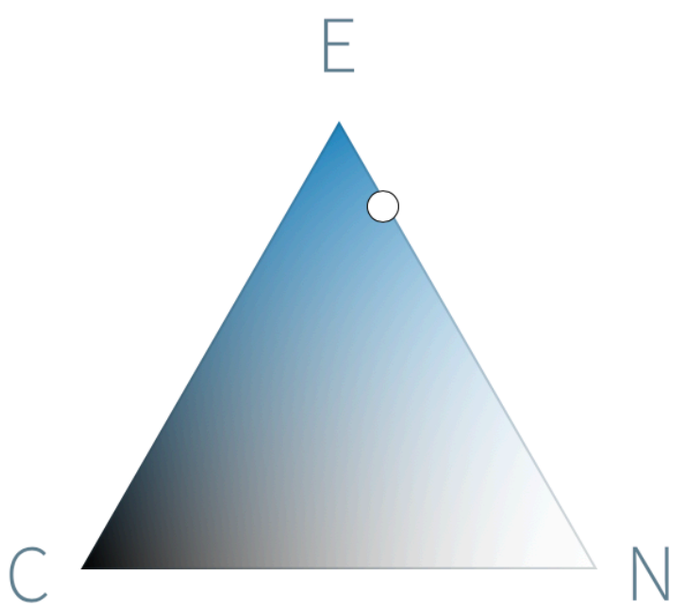
Premise

Maurita and Jade both were at the scene of the car crash.

Hypothesis

Multiple people saw the accident.

It is **very likely** that the premise **entails** the hypothesis.



Judgment	Probability
Entailment	81.1%
Contradiction	0.8%
Neutral	18.1%

KG-augmented Entailment System (KES)

Input:
- Premise
- Hypothesis

Word Embedding model
(e.g. word2vec, BERT)

Text Embeddings

$P = (p_1, .. p_n)$

$H = (h_1, .., h_m)$

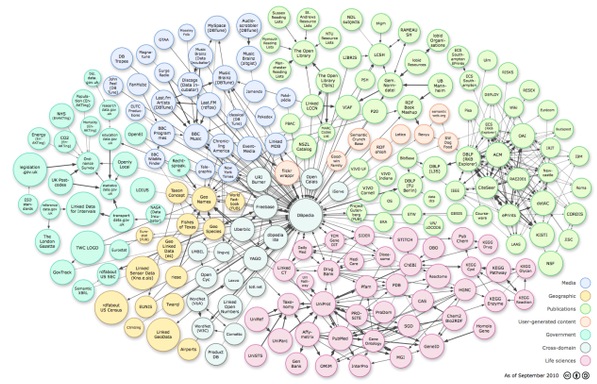
Standard Text-based Model

NLI Model

Output (global embedding)

$t_{out} \in \mathbb{R}^k$

Knowledge Base



Reasoning over Knowledge Base

Subgraph Extraction

GNN

Output (subgraph embedding)

$g_{out} \in \mathbb{R}^d$

Final Classifier
(e.g. MLP)

KG-augmented Entailment System (KES)

Input:
- Premise
- Hypothesis

Word Embedding model
(e.g. word2vec, BERT)

Text Embeddings

$P = (p_1, .. p_n)$

$H = (h_1, .., h_m)$

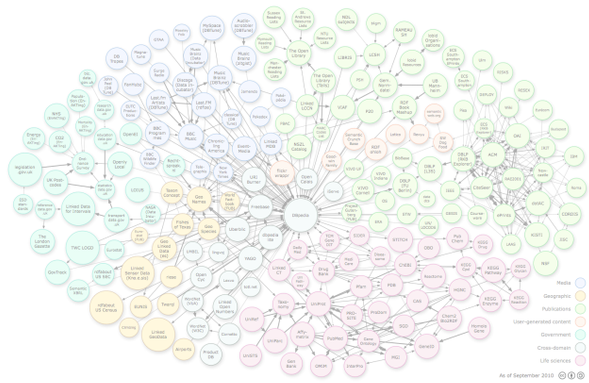
Standard Text-based Model

NLI Model

Output (global embedding)

$t_{out} \in \mathbb{R}^k$

Knowledge Base



Reasoning over Knowledge Base

Subgraph Extraction

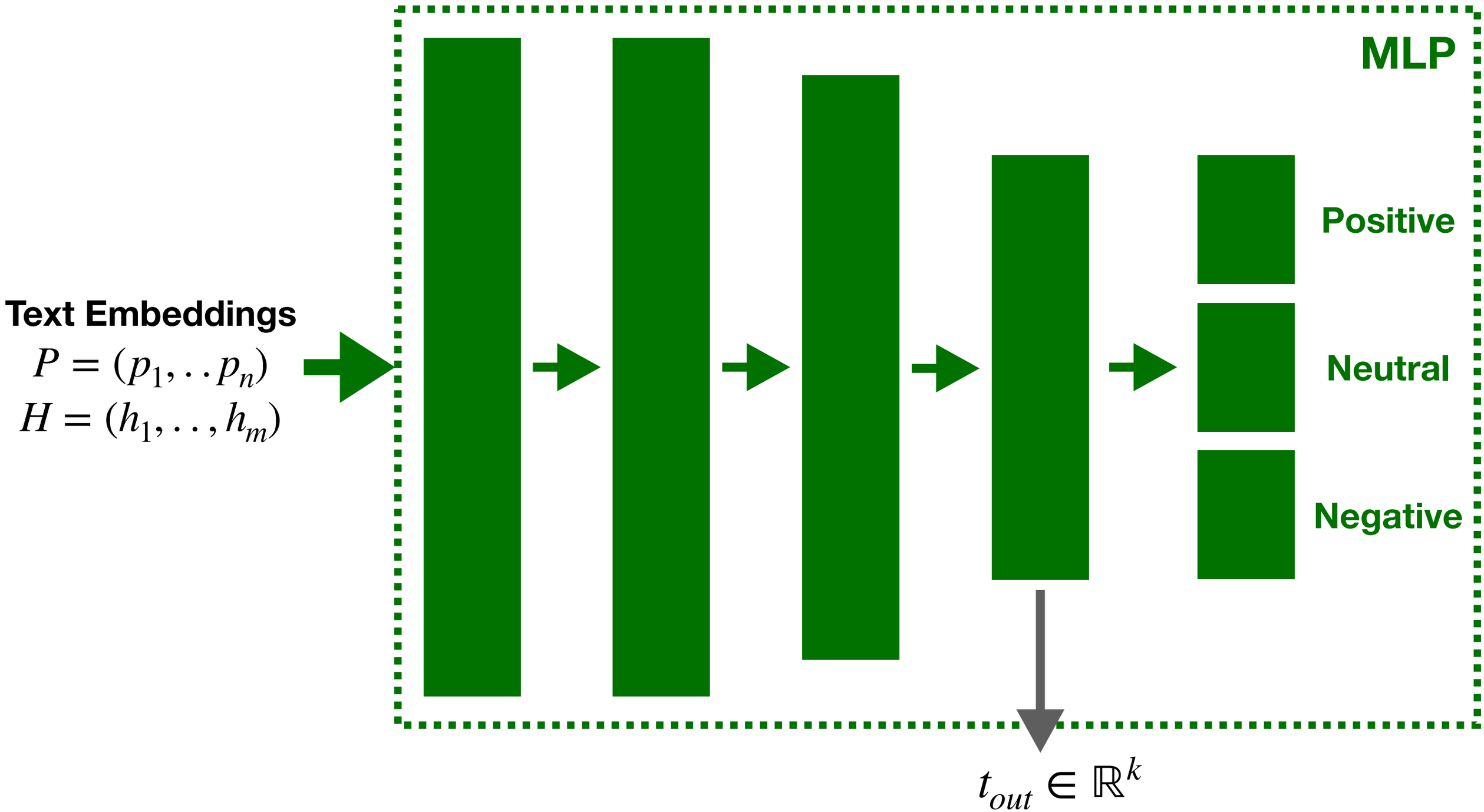
GNN

Output (subgraph embedding)

$g_{out} \in \mathbb{R}^d$

Final Classifier
(e.g. MLP)

Standard Text-based Model



KG-augmented Entailment System (KES)

Input:
- Premise
- Hypothesis

Word Embedding model
(e.g. word2vec, BERT)

Text Embeddings

$P = (p_1, .. p_n)$

$H = (h_1, .., h_m)$

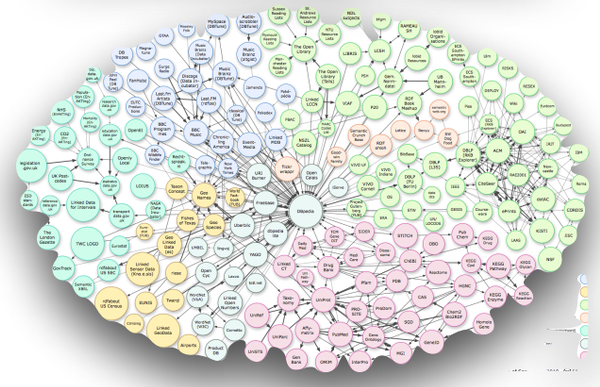
Standard Text-based Model

NLI Model

Output (global embedding)

$t_{out} \in \mathbb{R}^k$

Knowledge Base



Reasoning over Knowledge Base

Subgraph Extraction

GNN

Output (subgraph embedding)

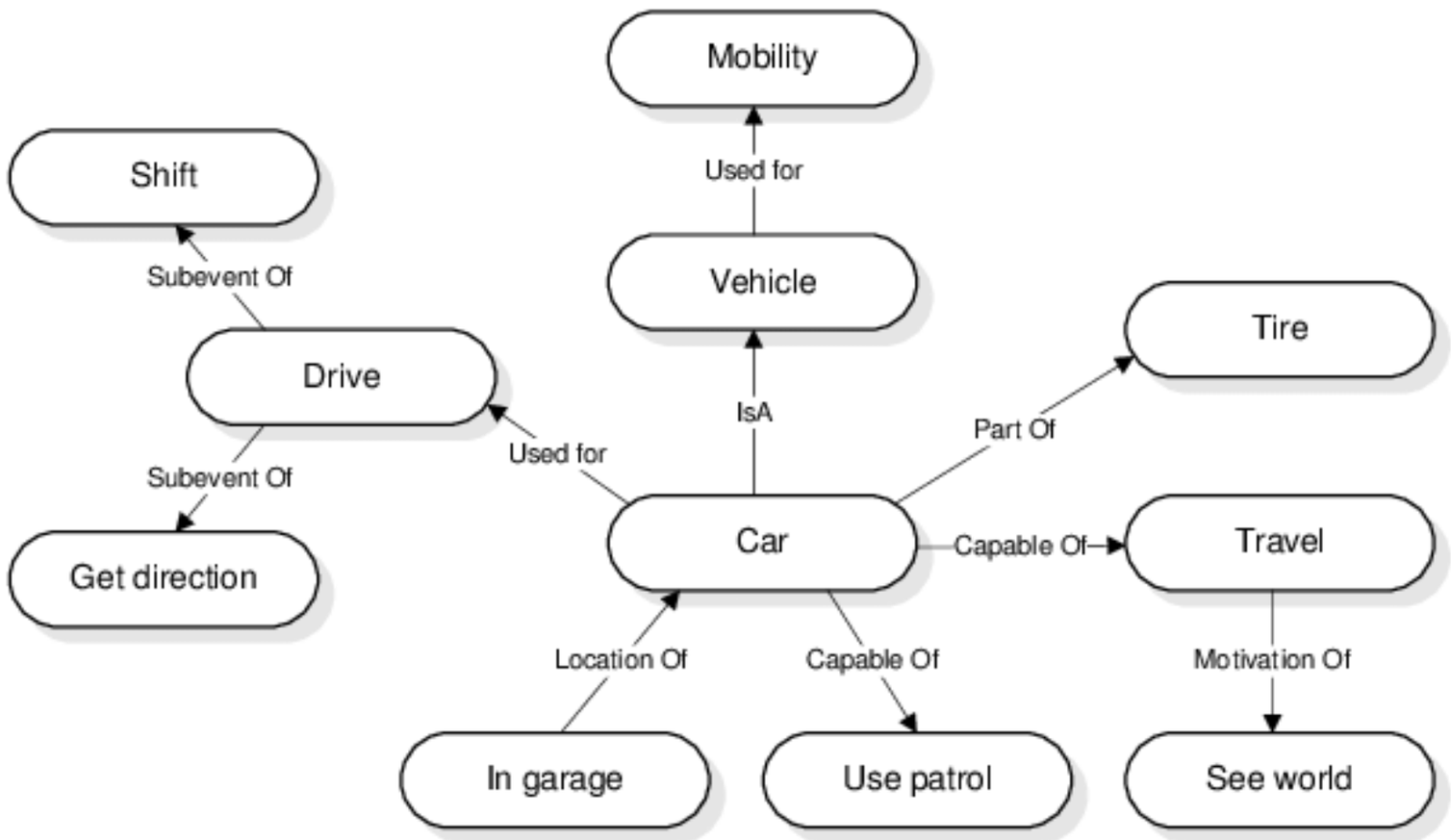
$g_{out} \in \mathbb{R}^d$

Final Classifier
(e.g. MLP)

Knowledge Base

Concept Net

“ConceptNet is a freely-available semantic network, designed to help computers understand the meanings of words that people use.” http://conceptnet.io



KG-augmented Entailment System (KES)

Input:
- Premise
- Hypothesis

Word Embedding model
(e.g. word2vec, BERT)

Text Embeddings

$P = (p_1, .. p_n)$

$H = (h_1, .., h_m)$

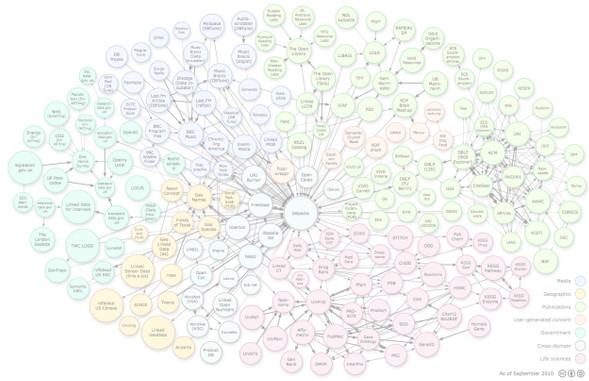
Standard Text-based Model

NLI Model

Output (global embedding)

$t_{out} \in \mathbb{R}^k$

Knowledge Base



Reasoning over Knowledge Base

Subgraph Extraction

GNN

Output (subgraph embedding)

$g_{out} \in \mathbb{R}^d$

Final Classifier
(e.g. MLP)

Reasoning over Knowledge Base

Inputs: Text Embeddings $P = (p_1, .. p_n)$, $H = (h_1, \ldots, h_m)$

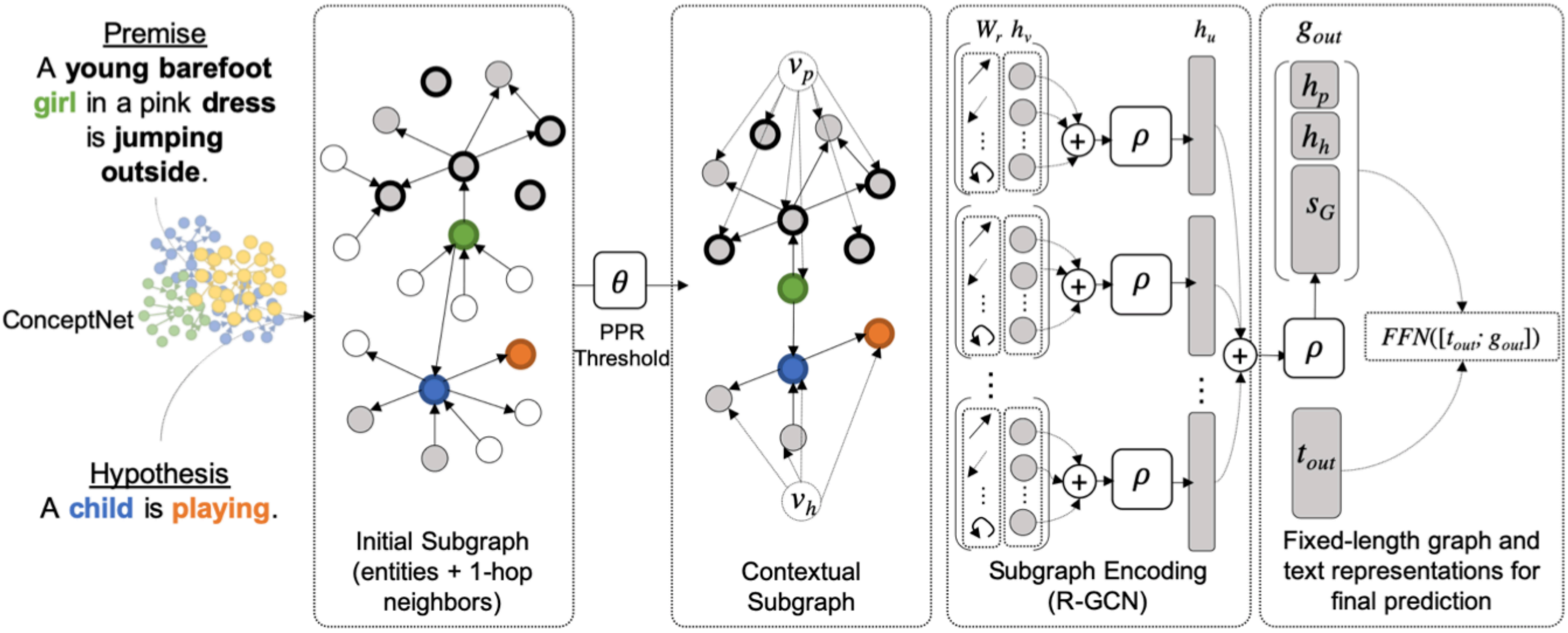
1. map the terms in premise and hypothesis text to concepts in KG by performing a max-substring match -> S is the set of mapped nodes.
2. Expand subgraph induced by S to include 1-hop neighbours
3. PPR filtering:
   3.1 perform random walks with restart with a bias towards the nodes in S
   3.2 exclude all nodes with a PPR score lower than θ
4. Add self-loops, and create premise supernode v_p and hypothesis supernode v_h
5. Encode resulting subgraph with R-GCN:

$$h_u^{l+1} = \rho\left(\sum_{r\in\mathcal{R}}\sum_{v\in\mathcal{N}_{u,r}}\frac{1}{c_{u,r}}W_r^l h_v^l\right) = \rho\left(\sum_{r\in\mathcal{R}}\textbf{GCN}_r(A_r, H)\right)$$

$$s_G = \rho\left(\sum_{v\in V} W h_v^L\right)$$

$$g_{out} = [s_G; h_{v_p}^L; h_{v_h}^L]$$

Reasoning over Knowledge Base



Results

Models	Scitail		MultiNLI		SNLI		BreakingNLI	
	Text	**KES**	**Text**	**KES**	**Text**	**KES**	**Text**	**KES**
match-LSTM	82.54	82.22 (0.6)	71.32	71.67 (0.8)	83.60	83.94 (0.6)	65.11	**78.72**
BERT+match-LSTM	89.13	**90.68** (0.2)	77.96	76.73 (0.6)	85.78	85.97 (0.6)	59.42	**77.59**
HBMP	81.37	**83.49** (0.2)	69.27	68.42 (0.6)	84.61	83.84 (0.2)	60.31	**63.60**
DecompAttn	76.57	72.43 (0.8)	64.89	**71.93** (0.6)	79.28	**85.56** (0.6)	51.3*	**59.83**
Existing Models with KG	**Text**	**Text+Graph**	**Text**	**Text+Graph**	**Text**	**Text+Graph**	**Text**	**Text+Graph**
KIM (Chen et al. 2018)	-	**NE**	-	76.4*	-	88.6*	-	83.1*
ConSeqNet (Wang et al. 2019)	84.2*	85.2*	71.32	70.9	83.60	83.34	65.11	61.12

Table 1: Entailment accuracy results of KES with different text models compared to text-only entailment models (Text). Bold values indicate where KES improves performance. PPR θ-values are shown in parentheses.*Reported values from related work.

PPR	Scitail (17.74*)		SNLI (11.5*)		MultiNLI (17.5*)	
	Edges	**Nodes**	**Edges**	**Nodes**	**Edges**	**Nodes**
0.2	42.65	10.14	80.29	19.83	76.27	16.15
0.4	26.72	7.48	25.70	8.15	33.82	6.48
0.6	15.53	4.35	14.08	4.65	23.97	3.44
0.8	11.67	3.04	9.98	3.18	20.27	2.05
ConSeqNet	0	0 (17.74*)	0	0 (11.5*)	0	0 (17.5*)
	No edges or new concepts are added from ConceptNet.					
KIM	Features based on fixed WordNet relations. No new concepts are added.					

Table 2: Average number of nodes and edges (not explicitly mentioned in text) in combined premise and hypothesis subgraphs by PPR threshold. *Average number of concepts explicitly mentioned in each premise and hypothesis text.