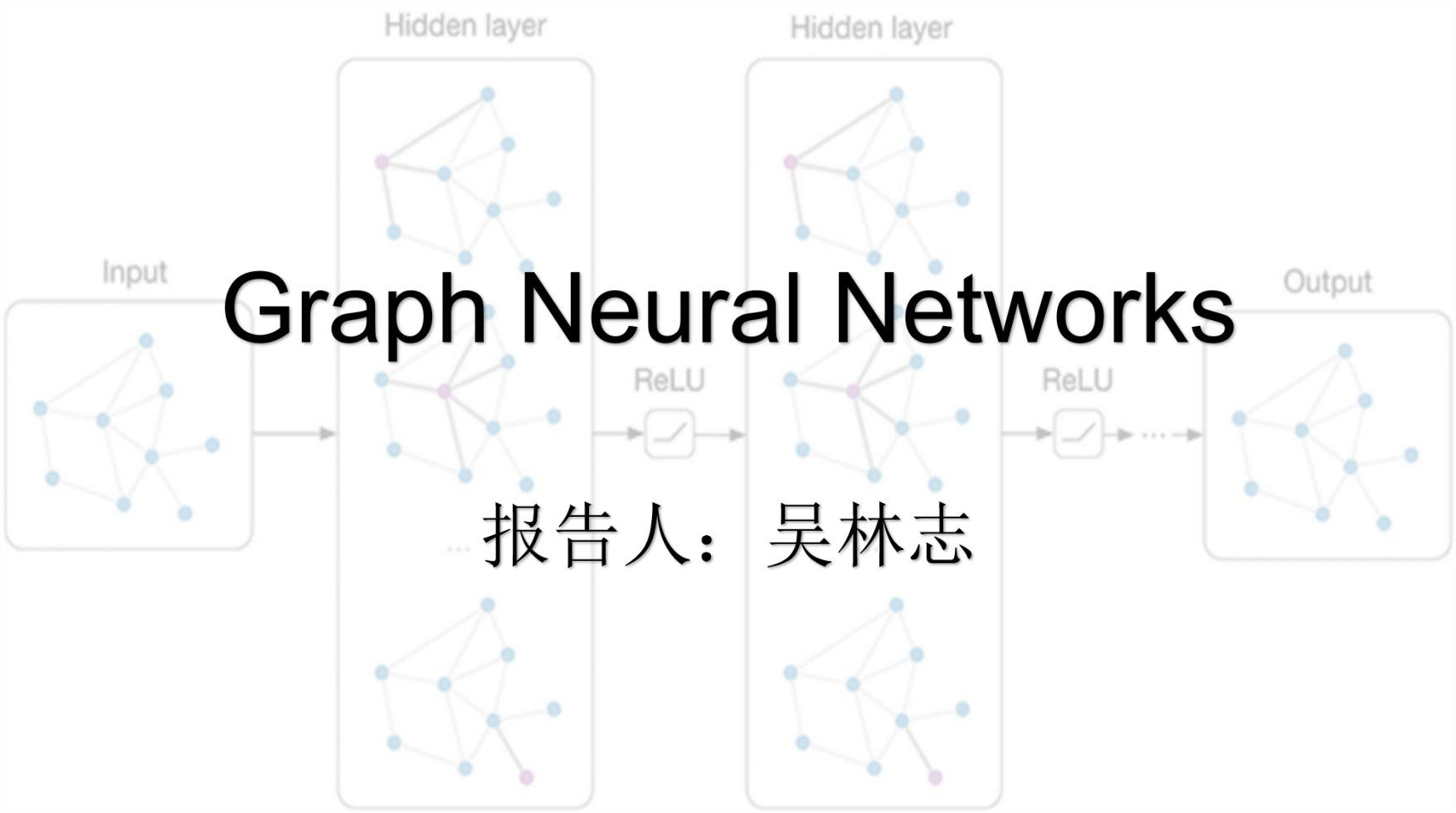


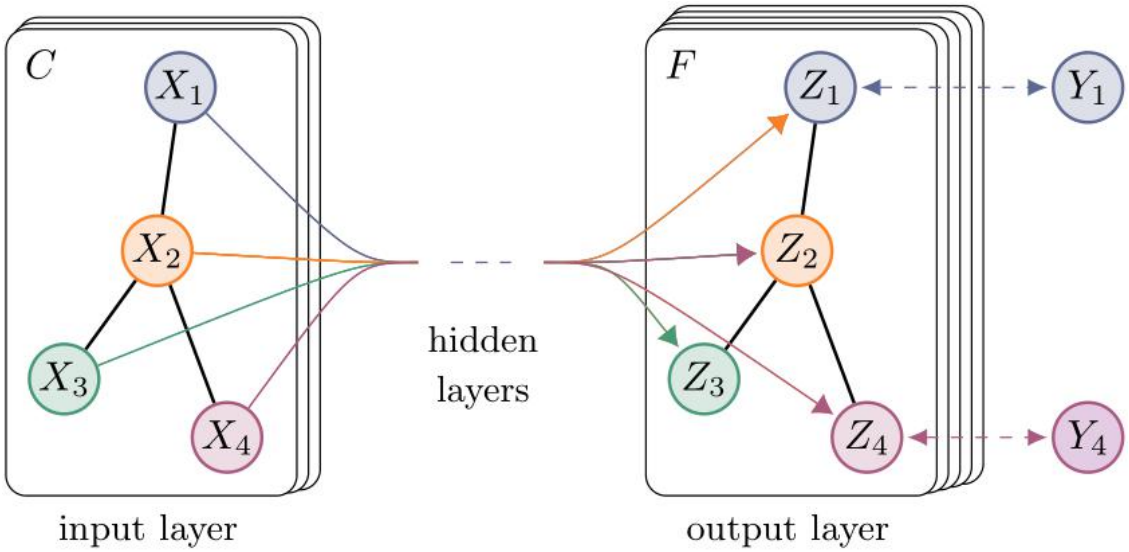
Graph Neural Networks

报告人：吴林志

相关论文

- GCN: Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017
- Text GCN: Graph Convolutional Networks for Text Classification, AAAI 2019
- GraphSAGE: Inductive Representation Learning on Large Graphs, NIPS 2017
- GAT: Graph Attention Networks, ICLR 2018

Graph Convolutional Network



$$H^{(l+1)} = f(H^{(l)}, \hat{A}) \Rightarrow \begin{cases} H^{(0)} = X \\ H^{(1)} = \sigma(\hat{A}XW^{(0)}) \\ H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^{(l)}) \end{cases}$$

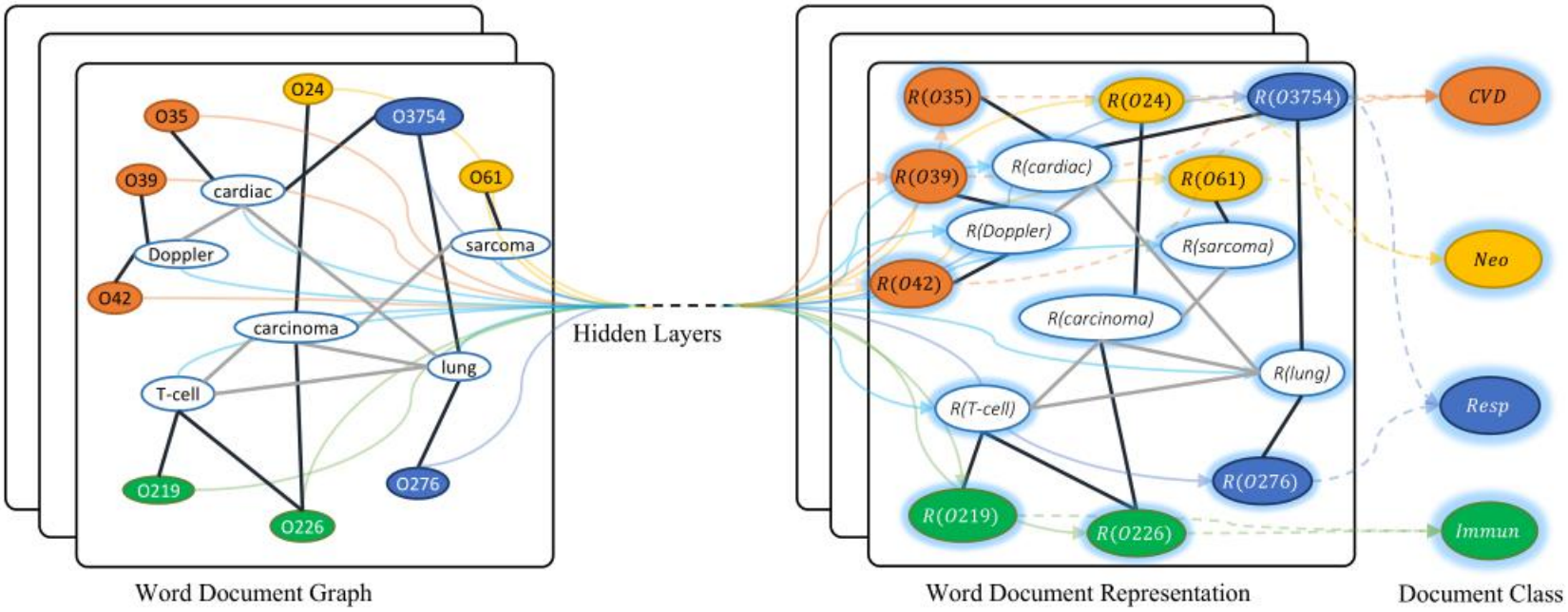
$$\hat{A} = \tilde{D}^{-1}\tilde{A} \quad \hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} \quad \tilde{D}_{ii} = \sum_j \tilde{A}_{ij} \quad \tilde{A} = A + I_N$$

例子

Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

拉普拉斯矩阵L = D - A (度矩阵-邻接矩阵)

Text GCN



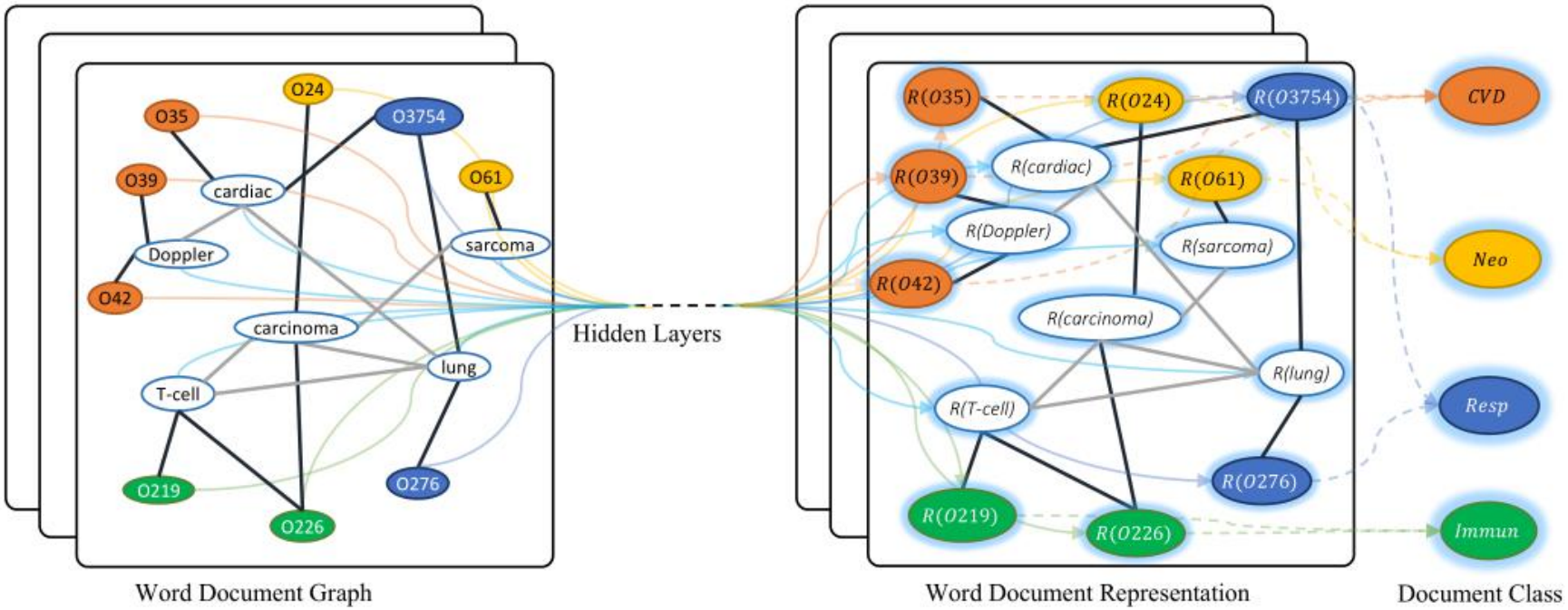
$$Z = \mathrm{softmax}\left(\tilde{A}\,\mathrm{ReLU}\left(\tilde{A}XW^{(0)}\right)W^{(1)}\right)$$

$$\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

$$D_{ii} = \sum_j A_{ij}$$

$$\mathcal{L} = -\sum_{d\in\mathcal{Y}_D}\sum_{f=1}^{F} Y_{df}\ln Z_{df}$$

如何将非结构化的文本表示成图结构？



- 在整个文本语料库上构建异构文本图 (实际上是一个大矩阵)
- 图的节点由文档和词组成
- 每个节点的向量都是one-hot形式表示

如何将非结构化的文本表示成图结构？

- 文档-词的边基于词在文档中的出现信息，使用TF-IDF作为边的权重；词-词的边基于词的全局词共现信息，使用PMI计算两个词节点连线的权重。

$$A_{ij} = \begin{cases} \mathrm{PMI}(i,j) & i\text{和}j\text{是词语而且}\mathrm{PMI}(i,j) > 0 \\ \mathrm{TF-IDF}(i,j) & i\text{是文档}j\text{是词语} \\ 1 & i = j \\ 0 & \text{其他} \end{cases}$$

TF-IDF and PMI

- TF-IDF表示一个词在文档中的重要性

$$TF-IDF=\frac{\text{词}w_i\text{在}doc_j\text{中出现的次数}}{doc_j\text{中所有词出现的次数和}}*\log\frac{\text{语料库中}doc\text{总数}}{1+\text{包含词}w_i\text{的}doc\text{数目}}$$

- PMI词与词间的互信息，大于0表示词与词之间的语义相关性越高，反之则较小或不存在

$$\text{PMI}(i,j)=\log\frac{p(i,j)}{p(i)p(j)}$$

$$p(i,j)=\frac{\#W(i,j)}{\#W}$$

$$p(i)=\frac{\#W(i)}{\#W}$$

#W: 滑动窗口的总数量
#W(i): 包含单词i的滑动窗口数量 (在一个语料库中)
#W(i, j): 同时包含单词i和单词j的滑动窗口的数量

Text GCN的特点

- 巧妙地将文档分类问题转为图节点分类问题，可以捕捉图的全局信息，从而很好地表示节点的特征
- 一个简单的双层Text GCN已经取得出色地效果
- 不能快速生成embeddings和预测没有见过的测试文档 (需要在训练的时候就加入网络中学习对应的embedding)

inductive / transductive learning

- 主要区别在于待预测的数据是否在训练过程中已经见(用)过！
- 直推式transductive：训练过程中，测试数据可见(测试集为训练提供无标签数据)
- 归纳式inductive：训练过程中，测试数据不可见(测试集只用于评估模型)

GCN的局限：

1、无法完成inductive任务 (即处理动态图问题)

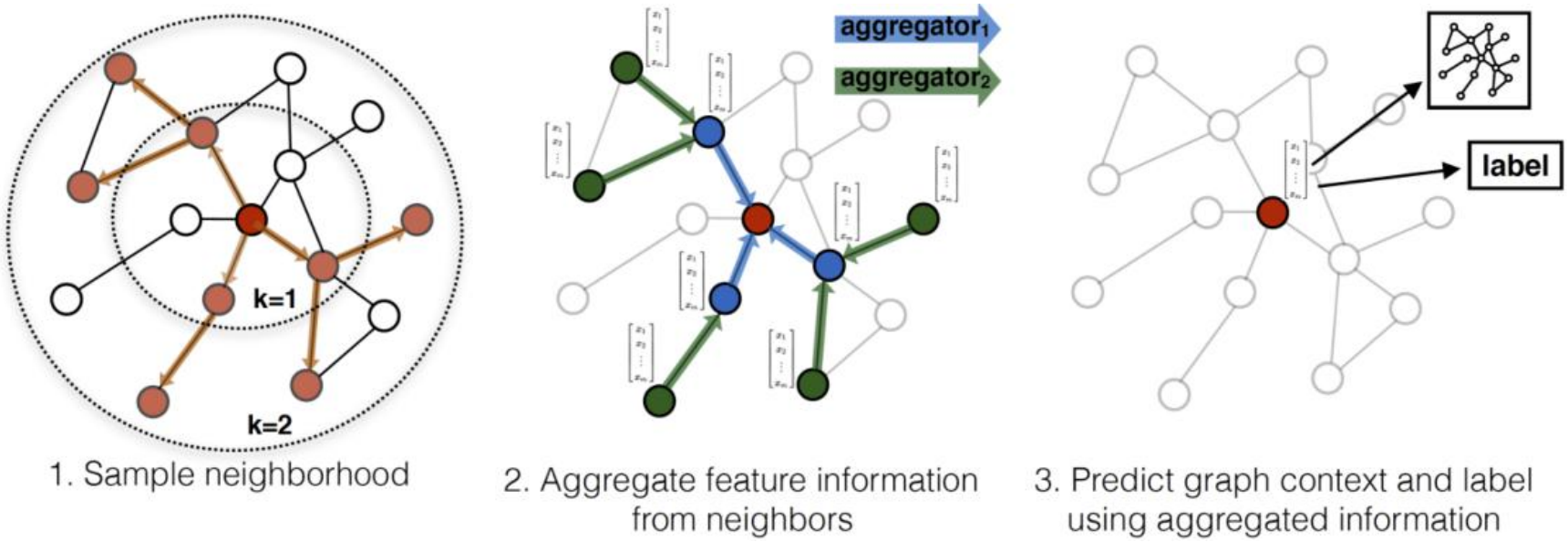
2、处理有向图的瓶颈，不容易实现分配不同的学习权重给不同的邻居

3、对于一个图结构训练好的模型，不能运用于另一个图结构 (没法快速表示新节点)

通用Inductive框架-GraphSAGE

- Transductive方法只能对一个固定的图生成embedding，不能对图中未见过的新节点生成embedding
- GraphSAGE是为了学习一种节点表示方法，即如何通过从一个顶点的局部邻居**采样**并**聚合**顶点特征，而不是为每个顶点训练单独的embedding

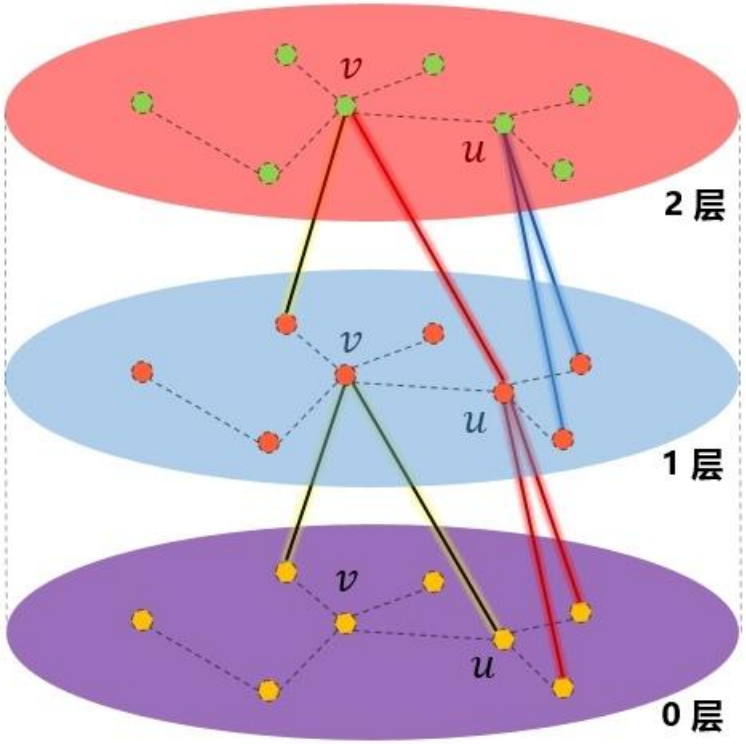
通用Inductive框架-GraphSAGE



1. 对图中每个顶点邻居顶点进行采样，因为每个节点的度是不一致的，为了计算高效， 为每个节点采样**固定数量**的邻居
2. 根据聚合函数聚合邻居顶点蕴含的信息 (Mean/LSTM/Pooling)
3. 得到图中各顶点的向量表示供下游任务使用

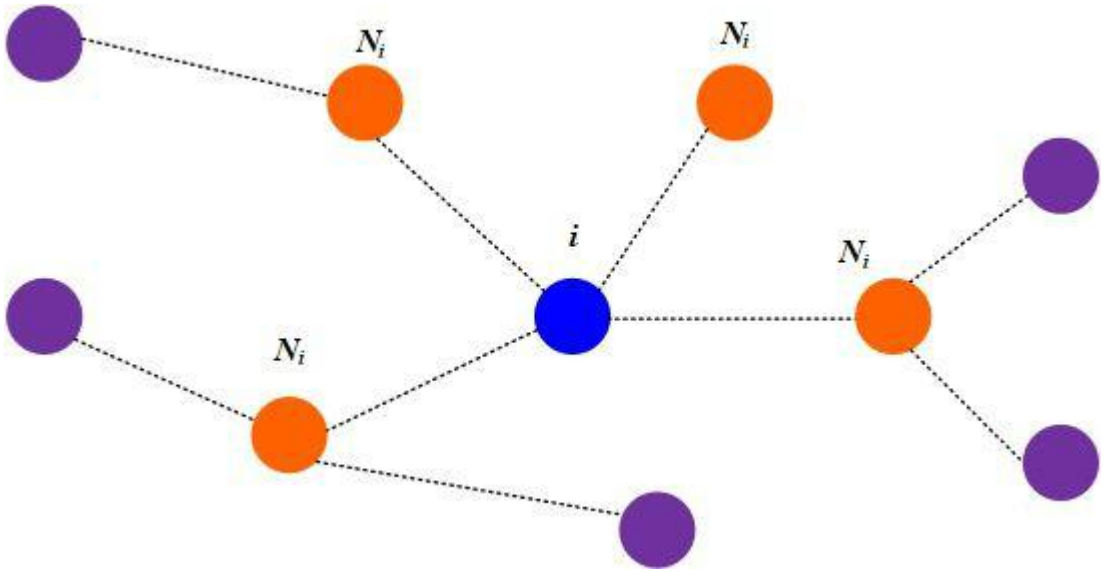
通用Inductive框架-GraphSAGE

- 每一层的node的表示都是由上一层生成的，跟本层的其他节点无关 (基于层的采样)
- 聚合K次，就可以扩展到K阶邻居



Graph Attention Network

- 引入masked self-attentional layers来改进GCN的缺点；
- 对不同的相邻节点分配相应的权重，既不需要矩阵运算，也不需要事先知道图结构



Graph Attentional Layer

- 和一般attention机制一样，GAT的计算也分为两步：**计算注意力系数**和**加权求和**

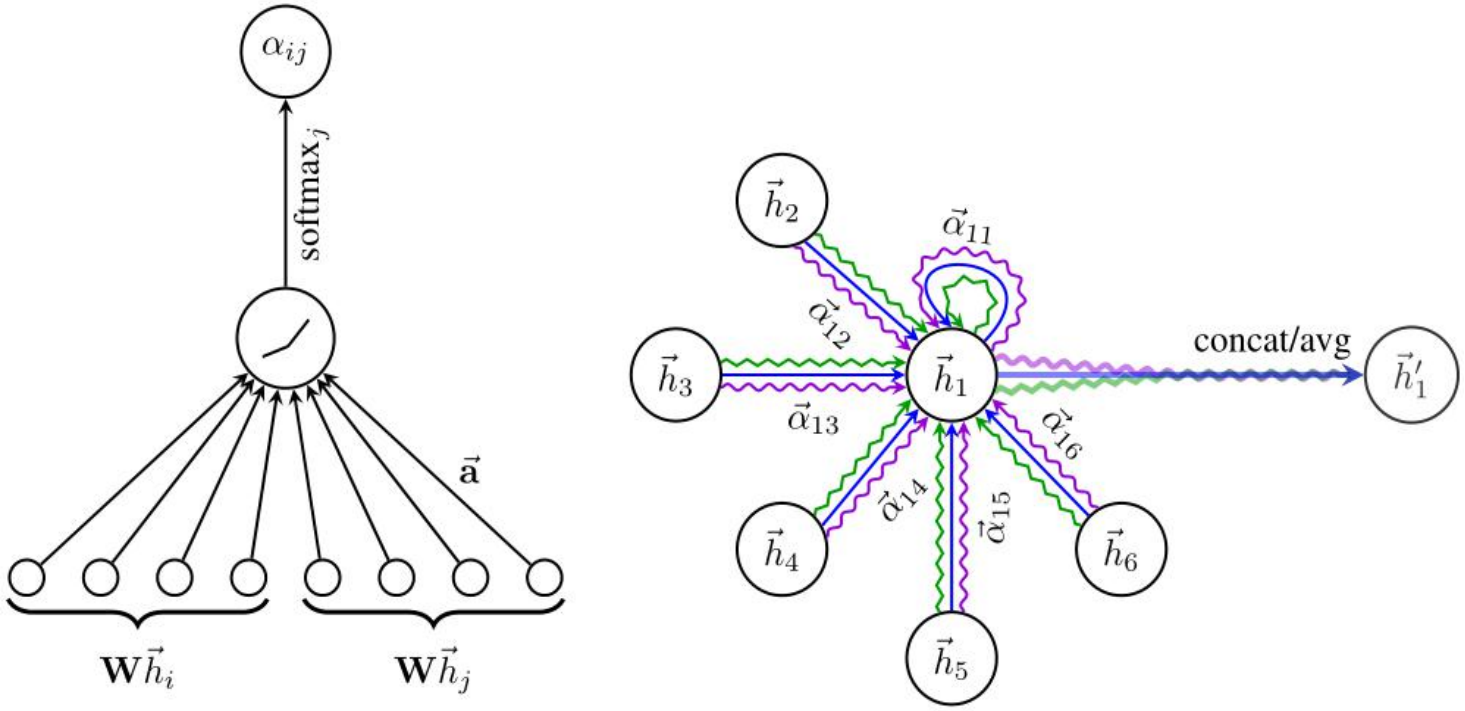
$$input: \mathbf{h} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}, \vec{h}_i \in \mathrm{R}^F \quad output: \mathbf{h}' = \{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\}, \vec{h}'_i \in \mathrm{R}^{F'}$$

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j) = \mathrm{LeakyReLU}(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \,\|\, \mathbf{W}\vec{h}_j]), \| \text{表示拼接}$$

$$\alpha_{ij} = \mathrm{softmax_j}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k\in N_i}\exp(e_{ik})} = \frac{\exp(\mathrm{LeakyReLU}(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \,\|\, \mathbf{W}\vec{h}_j]))}{\sum_{k\in N_i}\exp(\mathrm{LeakyReLU}(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \,\|\, \mathbf{W}\vec{h}_k]))}$$

$$\vec{h}'_i = \sigma\left(\sum_{j\in N_i}\alpha_{ij}\mathbf{W}\vec{h}_j\right) \overset{Multi-Head}{\Rightarrow} \begin{cases} concat: \vec{h}'_i = \Big\|_{k=1}^{K}\sigma\left(\sum_{j\in N_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right) \\ average: \vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in N_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right) \end{cases}$$

Graph Attentional Layer



GAT优点

- 计算高效 (并行化)　$O(|V|FF' + |E|F')$
- 为邻接节点分配不同的权重，考虑到节点特征之间的相关性
- 不需要事先得到整个图结构或所有顶点的特征 (只需访问目标节点的领接节点)
- 不要求图是无向的
- 能够运用于inductive任务中

VGCN-BERT: Augmenting BERT with Graph Embedding for Text Classification

用图嵌入增强BERT进行文本分类

Vocabulary Graph

- Normalized PMI

$$NPMI\ (i,j) = -\frac{1}{\log\ p(i,j)} \log \frac{p(i,j)}{p(i)\,p(j)} \in [-1,1]$$

$$p(i,j) = \frac{\#W(i,j)}{\#W}$$

$$p(i) = \frac{\#W(i)}{\#W}$$

#W: 滑动窗口的总数量

#W(i): 包含单词i的滑动窗口数量 (在一个语料库中)

#W(i, j): 同时包含单词i和单词j的滑动窗口的数量

- 窗口大小为整个句子长度
- 当两个词的NPMI大于某个阈值(0~0.3)时，建立一条边

Vocabulary GCN

- 通用GCN：对于输入n个节点的图，每个节点特征维度为m，新的节点表示H计算如下:

$$H = \tilde{A} X W$$

$$\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

$$D_{ii} = \sum_{j} A_{ij}$$

$$X \in \mathrm{R}^{n \times m}, W \in \mathrm{R}^{m \times h}$$

Vocabulary GCN

- VGCN的构建是基于词表而不是文档
- 对于单个文档，假设其表示为一个由词表中的词构成的行向量x，则单层卷积结果为：

$$h = (\widetilde{A}\,x^T)^T W = x\widetilde{A}\,W$$

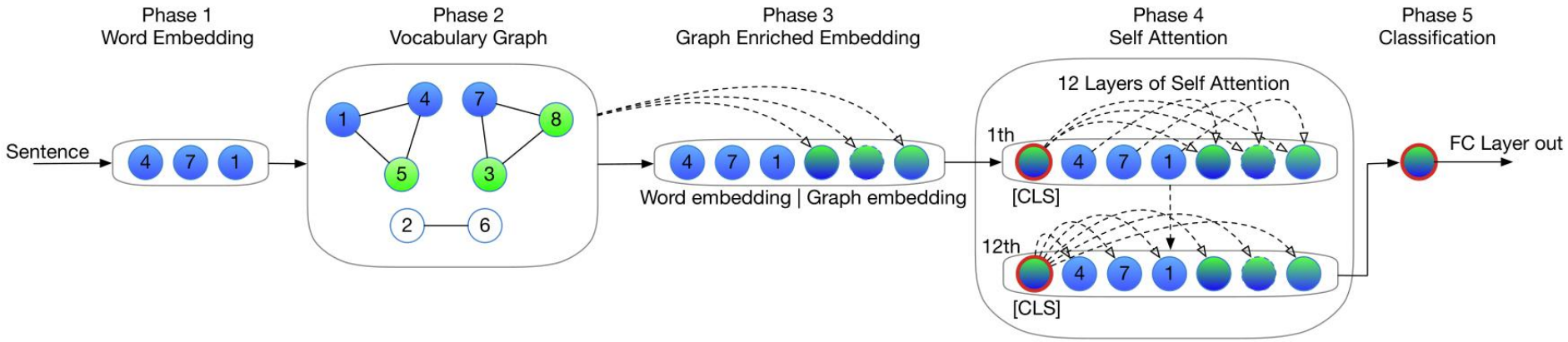
$$x \in 1 \times |V|, \widetilde{A} \in |V| \times |V|, W \in |V| \times h$$

- 2层的VGCN：

$$\mathbf{VGCN} = \mathrm{ReLU}(X_{mv}\widetilde{A}_{vv}W_{vh})W_{hc}$$

- m为batch大小(文档数量)，v为词表大小，h为隐层大小，c为类别数或句子嵌入维度
- X_{mv}是包含文档特征的向量(bow vector / word embedding)

Integrating VGCN into BERT



$$\mathbf{G_{embedding}} = \mathrm{ReLU}(X_{mev}\tilde{A}_{vv}W_{vh})W_{hg}$$

- g为图嵌入大小，其维度和每个词嵌入相同；m为batch大小；e为词嵌入维度；v为词表大小

Although it's a bit smug and repetitive, this documentary engages your brain in a way few current films do (=>innovation). **[Positive]**

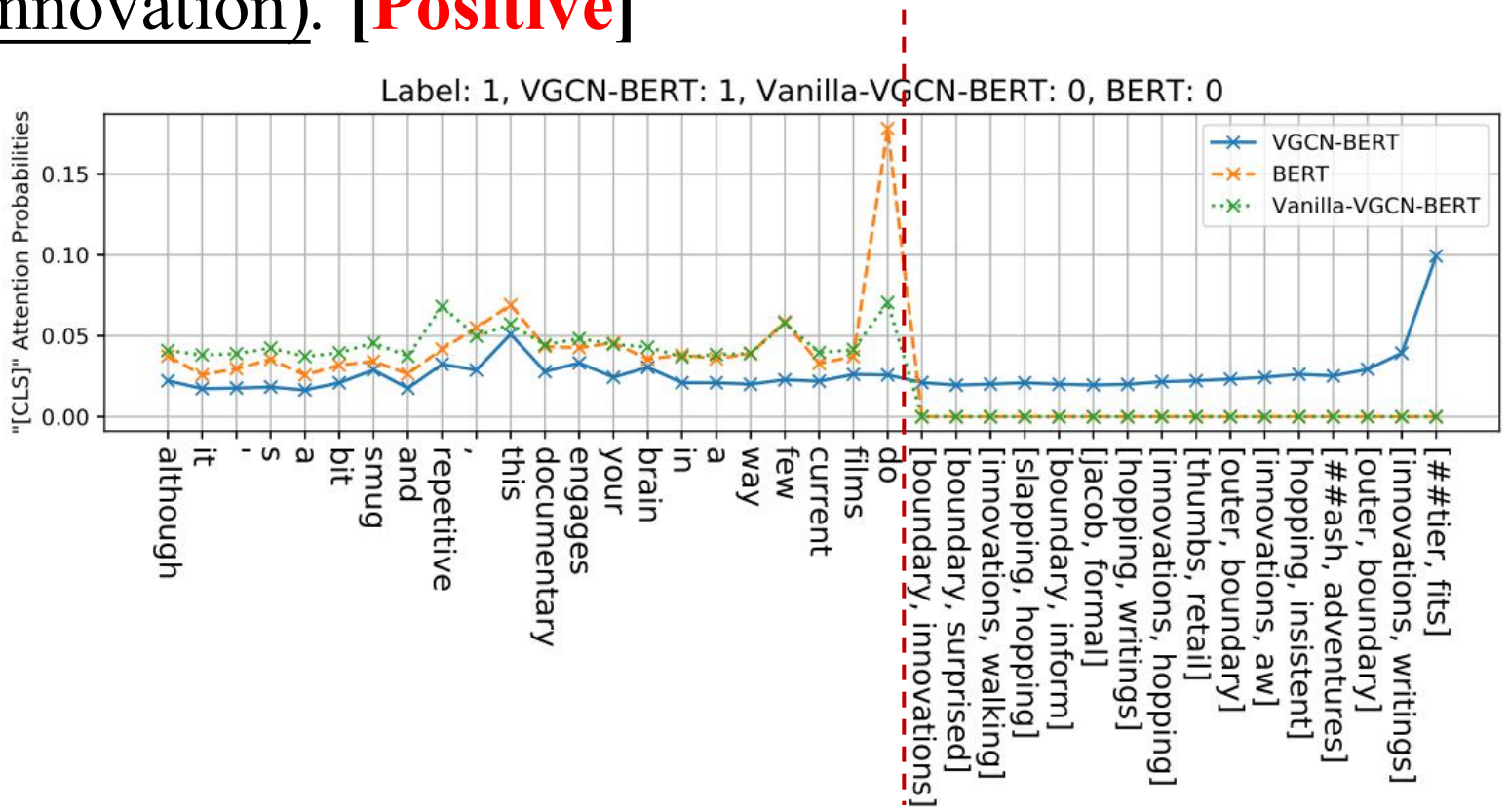


Fig. 2: Visualization of the attention that the token [CLS] (used as sentence embedding) pays to other tokens. The first part corresponds to word embeddings of the sentence. The second part is the graph embedding. [word1, word2] indicates the approximate meaning of a dimension in graph embedding.

文末干货

https://github.com/LindgeW/graph-based-deep-learning-literature

https://github.com/IndexFziQ/GNN4NLP-Papers

https://github.com/thunlp/GNNPapers

https://github.com/nnzhan/Awesome-Graph-Neural-Networks

https://github.com/benedekrozemberczki/awesome-graph-classification