Outline

- 卷积神经网络(CNN)
- 文本卷积

文本卷积

- 在NLP中通常使用的是一维卷积(1D卷积)

  -E.g. torch.nn.Conv1d

  -通常使用CV中的术语进行说明

- NLP中的两种主要范式:

  - **Context window modeling**: 主要用于序列输出任务，对每一个位置利用卷积考虑上下文情况进行表示

  - **Sentence modeling**: 进行整个句子的建模，使用卷积提取n元语法信息，通过池化组合，并获得句子整体表示

文本1D卷积-窄卷积

句子：tentative deal reached to keep government open

tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3

卷积

步长=1

t,d,r	−1.0
d,r,t	−0.5
r,t,k	−3.6
t,k,g	−0.2
k,g,o	0.3

不使用padding
输出特征映射大小
=7-3+1=5

- 使用大小=3的卷积核，形状实际为3*d

3	1	2	−3
−1	2	1	−3
1	1	−1	1

d

通道 channel
该卷积核进行跨通道操作

文本1D卷积-宽卷积

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

$$\frac{3-1}{2} = 1 \rightarrow$$

Ø,t,d	−0.6
t,d,r	−1.0
d,r,t	−0.5
r,t,k	−3.6
t,k,g	−0.2
k,g,o	0.3
g,o,Ø	−0.5

- 使用大小=3的卷积核

3	1	2	−3
−1	2	1	−3
1	1	−1	1

如何保持输出特征映射的宽度？

文本1D卷积-宽卷积

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

→

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

- 设置多个卷积核，*e.g.* 使用3个大小=3的卷积核

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

捕获不同类别的输入特征

多通道 vs 多滤波器

- 多通道
  - 如果使用一个通道，那么每一个词语只能用一个index来代替，这样的输入能够提供的信息非常有限→预训练词向量
  - 多通道的输出有两个相同的拷贝，其中一份会随着反向传播进行更新，另一份保持不变

- 多滤波器
  - 一个大小为n的滤波器一次查看一个窗口大小为n的范围内的局部的词语序列，多个滤波器可以捕获不同方面的信息
  - E.g. 属性情感分类，第一个滤波器关注aspect，第二个关注文本情感
  - 对于不同的滤波器，可以有不同的大小.

(全局)最大池化

- 利用卷积的输出(不同方面的特征)对文本进行总结、压缩
- 最大池化max pooling代表选择一个范围内最大的值，其他的值被丢弃
- 池化实际完成了降维
- 缺点：可能造成信息的损失

Ø,t,d	-0.6	0.2	1.4
t,d,r	-1.0	1.6	-1.0
d,r,t	-0.5	-0.1	0.8
r,t,k	-3.6	0.3	0.3
t,k,g	-0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	-0.5	-0.9	0.1

在整个列上做**全局**最大池化

max pooling over time

max p	0.3	1.6	1.4

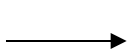
最显著的特征

(全局)平均池化

- 平均池化ave pooling取一个范围内最大的平均
- 一般最大池化比平均池化好，原因在于NLP中信息比较稀疏，平均池化弱化了我们想要查看的显著特征

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

在每一个列上平均池化



ave p	−0.87	0.26	0.53

池化方式调用：
torch.nn.functional

局部池化

- 针对局部的池化方式

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

kernel=2
步长=2

最大池化 →

Ø,t,d,r	-0.6	1.6	1.4
d,r,t,k	-0.5	0.3	0.8
t,k,g,o	0.3	0.6	1.2
g,o,Ø,Ø	-0.5	-0.9	0.1

平均池化 →

Ø,t,d,r	-0.8	0.9	0.2
d,r,t,k	…	…	…
t,k,g,o	…	…	…
g,o,Ø,Ø	-0.5	-0.9	0.1

(全局) k-max pooling

- NLP中常见的使用方式。对每个通道内的特征映射值，选择k个最大的值，相当于保留一项功能，或者说一个方面，在这个文本中最显著的k个特征值，k是给定的
- max pooling可以看成是k=1的特殊情况
- k-max pooling后得到的这些数值的顺序和它们在原来矩阵中的顺序相同

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

2-max池化 →

2-max p	−0.2	1.6	1.4
	0.3	0.6	1.2

(全局)动态k-max pooling

- 广泛使用的方式。动态体现在，k值不是事先设定的，而是一个函数，该函数和文本长度、网络深度有关。
- 整个模型中，动态k-max pooling一般加在中间的卷积层后面，最后一层卷积后是k值固定的k-max pooling
- 动态的k值，可以更加平滑地提取高阶和更长范围的特征

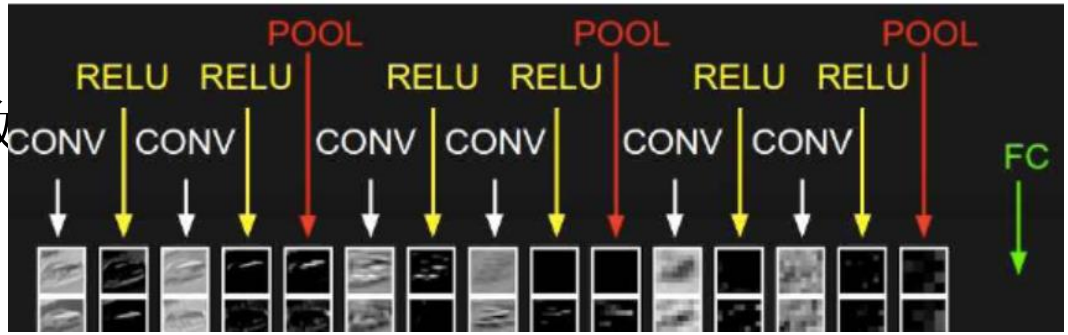
$$k_l = \max(k_{top}, \left\lceil \frac{L-l}{L} s \right\rceil)$$ [ ]代表向上取整

l: 当前卷积层的层数

L: 模型中使用的卷积总层数

s: 文本长度

k_{top}: 最后一层池化的k值



变体: dilation卷积

- 卷积中的降维、增加感受野的方式：池化、增加层数、增大卷积核
- 池化可能丢失信息，增加层数和卷积核大小会增加参数数量 → dilation(膨胀卷积/空洞卷积)
- 膨胀卷积：在计算邻域加权和的时候，跳过某些行，就像给卷积核插入空洞，相当于增加了卷积核的大小

1	Ø	0.0	0.0	0.0	0.0
2	tentative	0.2	0.1	−0.3	0.4
3	deal	0.5	0.2	−0.3	−0.1
4	reached	−0.1	−0.3	−0.2	0.4
5	to	0.3	−0.3	0.1	0.1
6	keep	0.2	−0.3	0.4	0.2
7	government	0.1	0.2	−0.1	−0.1
8	open	−0.4	−0.4	0.2	0.3
9	Ø	0.0	0.0	0.0	0.0

感受野大小=5

滤波器大小=3

2	3	1
1	-1	-1
3	1	0

1	3	1
1	-1	-1
3	1	-1

每次跳过1行，步长=1

1,3,5	0.3	0.0
2,4,6		
3,5,7		

变体: dilation

- 一次跳过m行，即每两个元素之间插入了m个空洞，令m+1为膨胀率，称为d (dilation rate). d=1为普通卷积
- 卷积核的有效大小$k' = k + (k - 1) \times (d - 1)$，例子中k'=5，即一次卷积读取数据跨越了5行，相当于让输入变得分散
- 设置不同膨胀率时，感受野就不一样，即设置不同膨胀率可以获取多尺度信息

1	Ø	0.0	0.0	0.0	0.0
2	tentative	0.2	0.1	−0.3	0.4
3	deal	0.5	0.2	−0.3	−0.1
4	reached	−0.1	−0.3	−0.2	0.4
5	to	0.3	−0.3	0.1	0.1
6	keep	0.2	−0.3	0.4	0.2
7	government	0.1	0.2	−0.1	−0.1
8	open	−0.4	−0.4	0.2	0.3
9	Ø	0.0	0.0	0.0	0.0

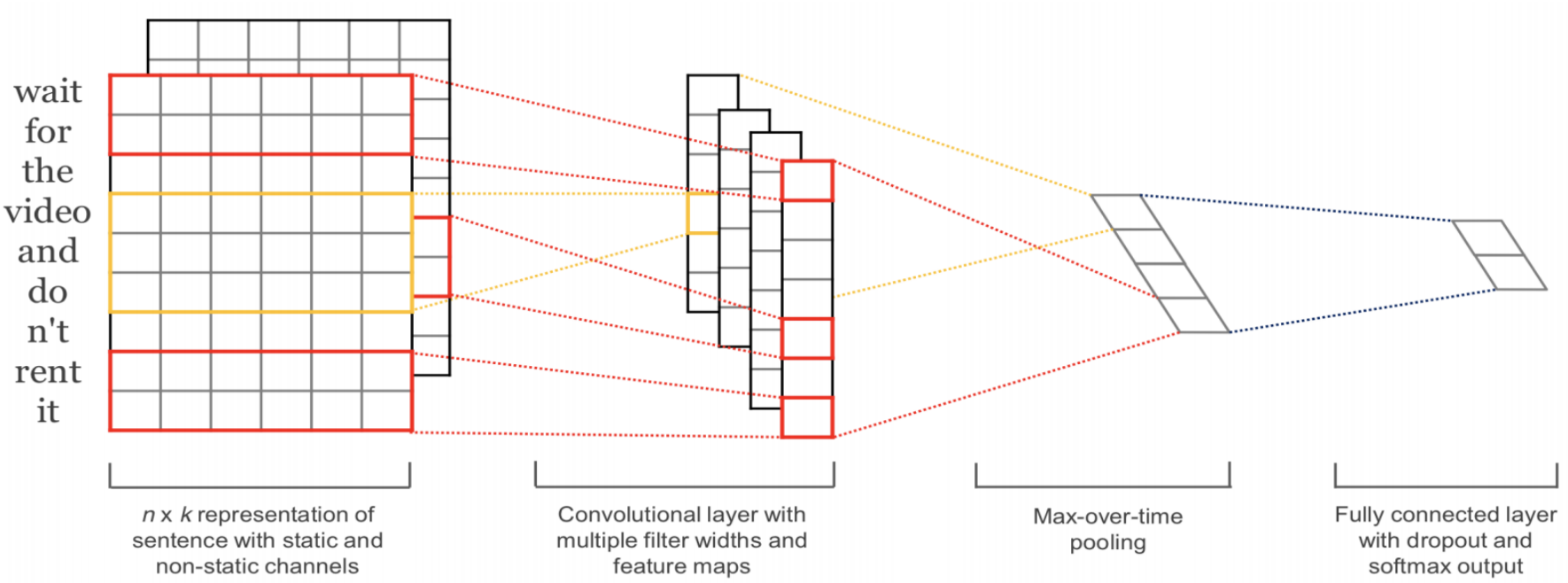
滤波器

2	3	1	1	3	1
1	-1	-1	1	-1	-1
3	1	0	3	1	-1

每次跳过1行，即D=2

1,3,5	0.3	0.0
2,4,6		
3,5,7		

文本CNN：例子1



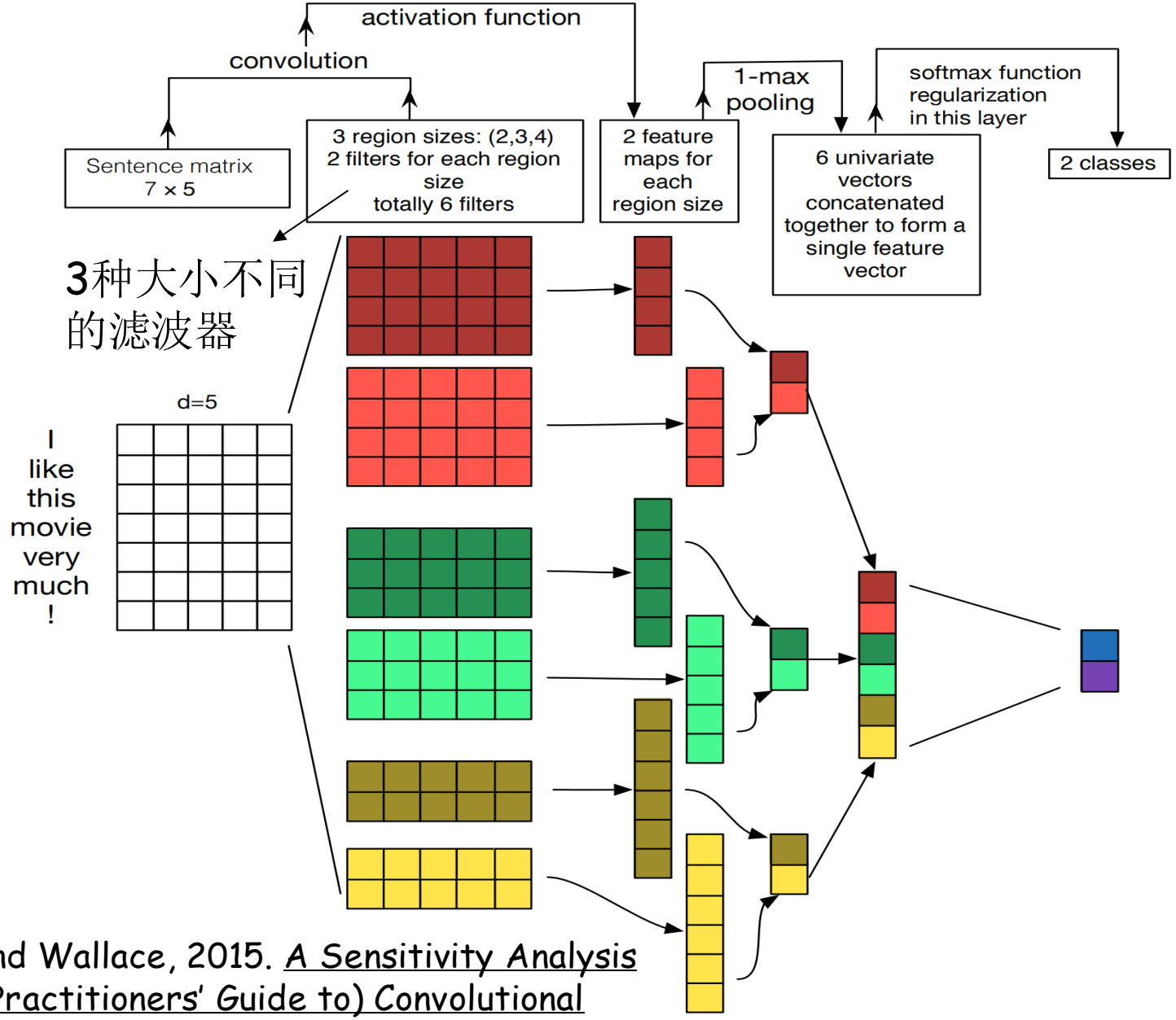
CBOW预训练词向量
static: 在训练过程中不更新词向量
non-static: 更新词向量

大小不同的多滤波器
h×k，**h** 是滤波器的大小

最大池化，保留表达文本语义最重要的信号

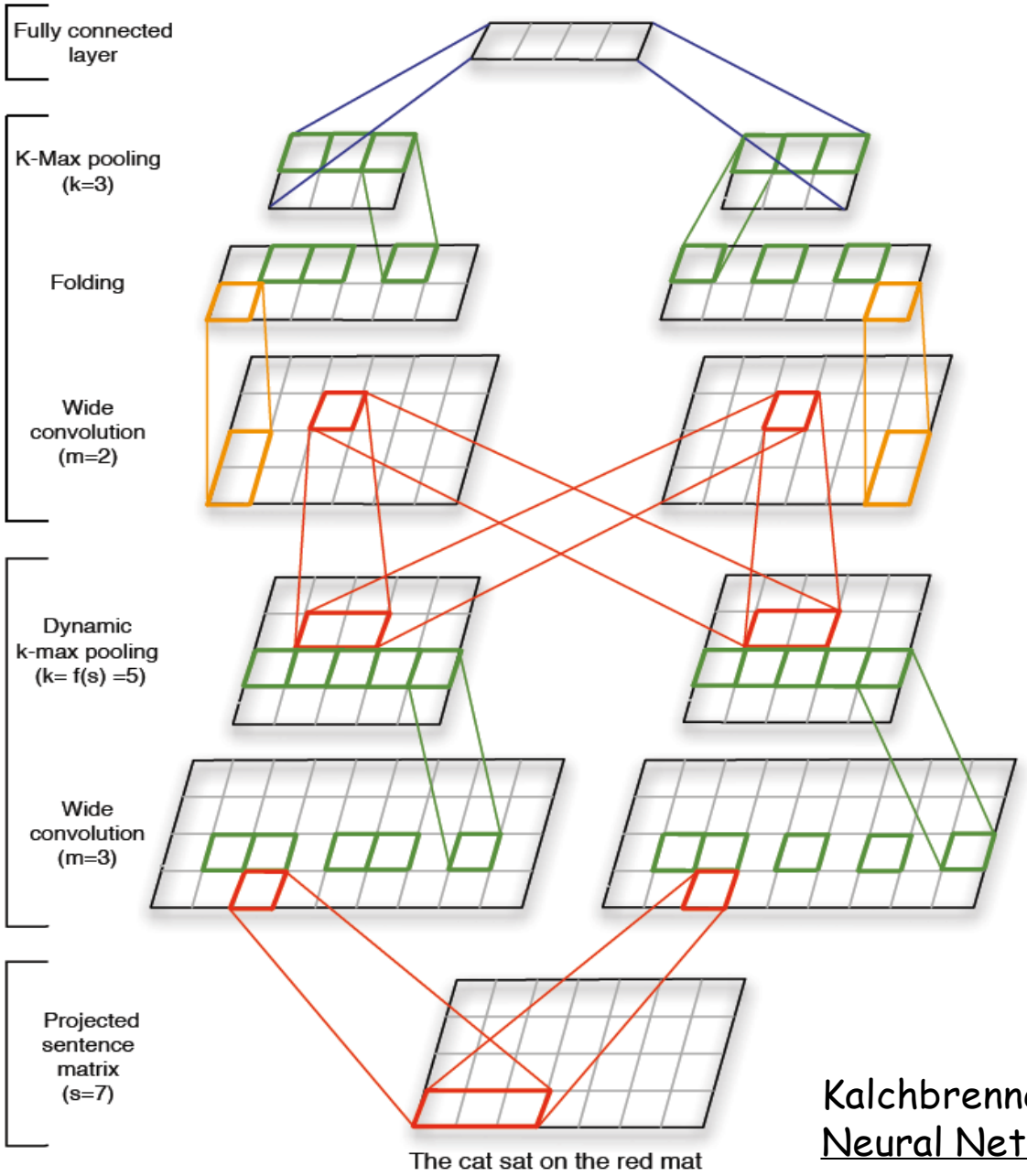
Kim, 2014. EMNLP. Convolutional Neural Networks for Sentence Classification

文本CNN：例子2



Zhang and Wallace, 2015. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification

文本CNN：例子3



1) 2个大小为3的宽卷积

2)非线性激活

3) 动态k-max池化

4) 以上可重复多次

5) Folding层

6) k-max pooling, k=3

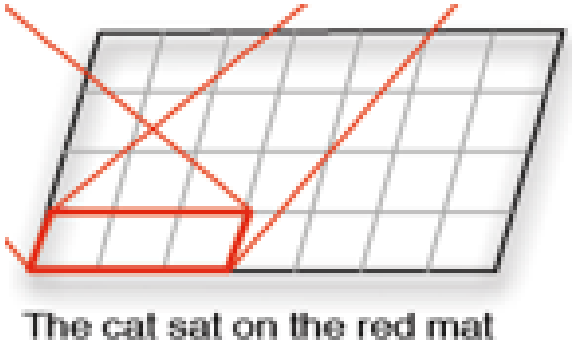
7) 全连接层

多层CNN

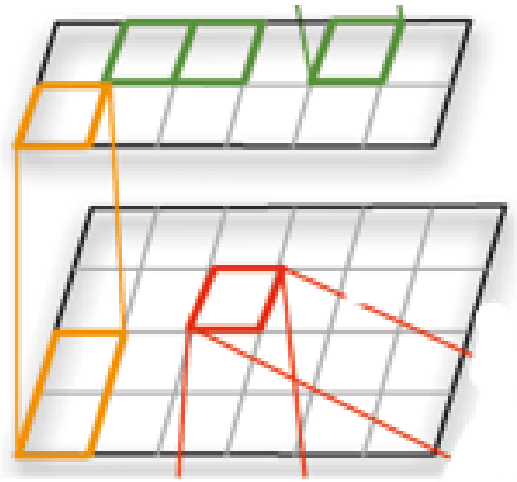
Kalchbrenner et al, 2014. ACL. A Convolutional Neural Network for Modelling Sentences

文本CNN：例子3

- Folding层：考虑不同行之间的联系，将两行中同一列上的数据进行相加，行数减小一半，好像“拦腰折叠”。
- 该过程没有引入新的参数，但是能够在全连接层之前提前考虑到词向量维度上的某些关联。



输入



Folding操作

文本CNN：例子4

Hybrid-Siamese Convolutional Neural Network (HSCNN)

- 任务：
  - ✓ 多标签分类（一个样本对应一个或多个类别标签）
  - ✓ 样本分布不均衡

vs 多分类 (类别总数很多)

- 孪生网络：
  - ✓ 框架中包含两个相同的部分，包括结构、参数…
  - ✓ 从数据中去学习一个相似性度量，用这个度量去比较和匹配新的未知类别的样本

Loss	Objectives
BCE	$-\sum_c \sum_{v\in\{0,1\}} y_{cv} \log p_{cv}$
WCE	$-\sum_c \alpha_c \sum_{v\in\{0,1\}} y_{cv} \log p_{cv}$
Focal	$-\sum_c \sum_{v\in\{0,1\}} y_{cv}(1-p_{cv})^{\gamma} \log p_{cv}$

对于某样本：C个类别，依次看，第i个类别的预测是否准确

多标签 HSCNN

Yang et al, 2020. EMNLP. HSCNN: A Hybrid-Siamese Convolutional Neural Network for Extremely Imbalanced Multi-label Text Classification

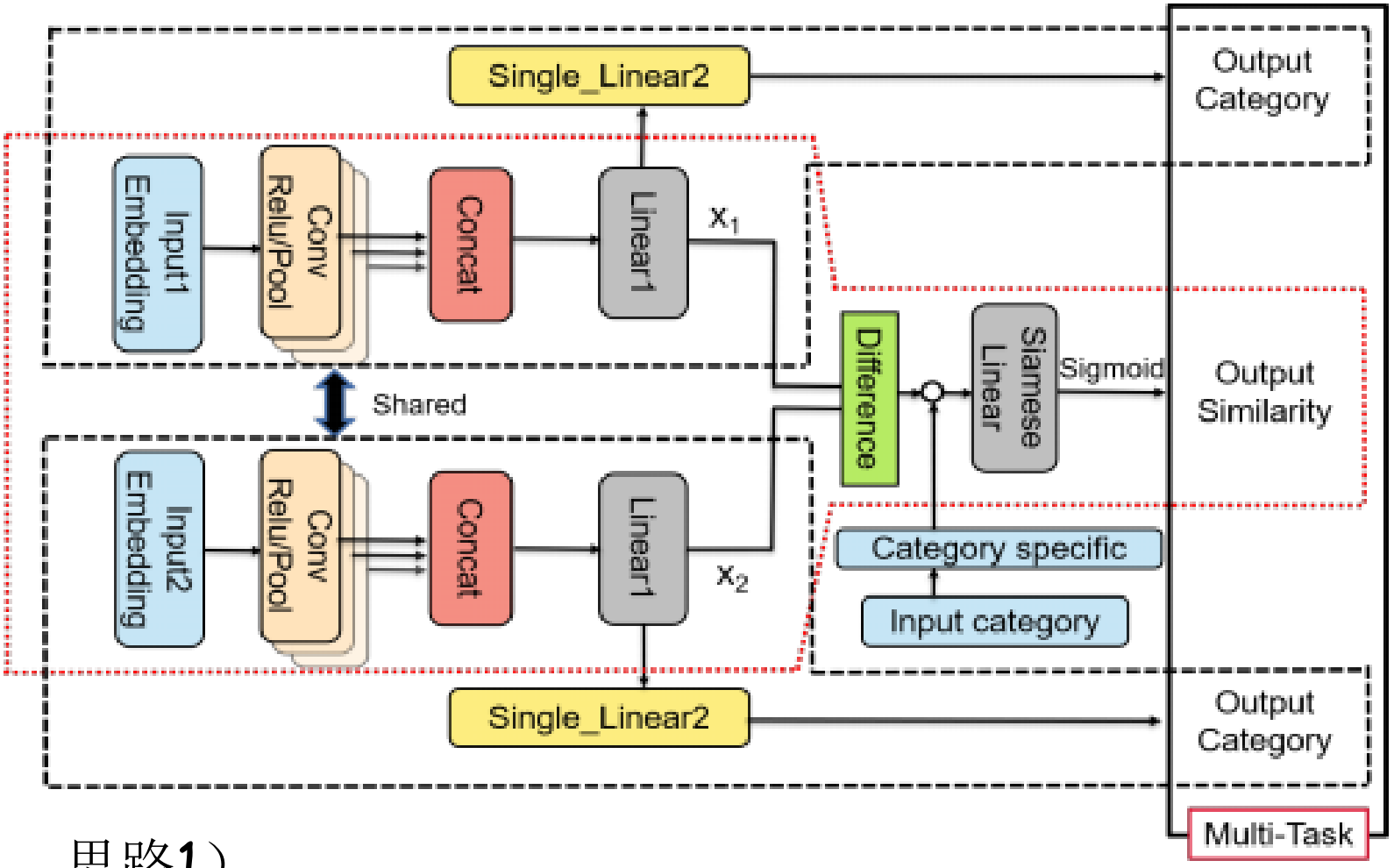
多标签分类

- 目标类别之间互斥，一个样本可能对应1个或多个类别
- 例子：电影的类型，假设一共有C个类型标签

Q: 生活中还有哪些多标签分类的例子？怎么处理该任务？

1) 转化为C个二分类
2) 转化为文本生成任务，输出类别的序列
3) 统计所有标签的组合（K种），进行K分类
4) 第一个分类器预测标签个数，第二个分类器取top-n个结果
5) 。。。

HSCNN



思路1）

Yang, Li, Fukumoto et al, 2020. EMNLP. HSCNN: A Hybrid-Siamese Convolutional Neural Network for Extremely Imbalanced Multi-label Text Classification

```
class TextCNN(nn.Module):
    def __init__(self, config):
        super(TextCNN, self).__init__()
        self.is_training = True
        self.dropout_rate = config.dropout_rate
        self.num_class = config.num_class
        self.config = config

        self.embedding = nn.Embedding(num_embeddings=config.vocab_size,
                                      embedding_dim=config.embedding_size)
        self.convs = nn.ModuleList([
                nn.Sequential(nn.Conv1d(in_channels=config.embedding_size,
                                        out_channels=config.feature_size,
                                        kernel_size=h),
#                               nn.BatchNorm1d(num_features=config.feature_size),
                              nn.ReLU(),
                              nn.MaxPool1d(kernel_size=config.max_text_len-h+1))
                    for h in config.window_sizes
                    ])
        self.fc = nn.Linear(in_features=config.feature_size*len(config.window_sizes),
                            out_features=config.num_class)
        if os.path.exists(config.embedding_path) and config.is_training and config.is_pretrain:
            print("Loading pretrain embedding...")
            self.embedding.weight.data.copy_(torch.from_numpy(np.load(config.embedding_path)))

    def forward(self, x):
        embed_x = self.embedding(x)
# batch_size x text_len x embedding_size  -> batch_size x embedding_size x text_len
        embed_x = embed_x.permute(0, 2, 1)
        out = [conv(embed_x) for conv in self.convs]  #out[i]:batch_size x feature_size*1
        out = torch.cat(out, dim=1)  # 对应第二个维度（行）拼接起来，比如说5*2*1,5*3*1的拼接变成5*5*1
        out = out.view(-1, out.size(1))
        #print(out.size())  # 32*400
        out = self.fc(out)
        return out
```

也可以使用conv2d实现
输入和滤波器形状最后添加一个维度1

class torch.nn.Conv1d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True)

Batch normalization

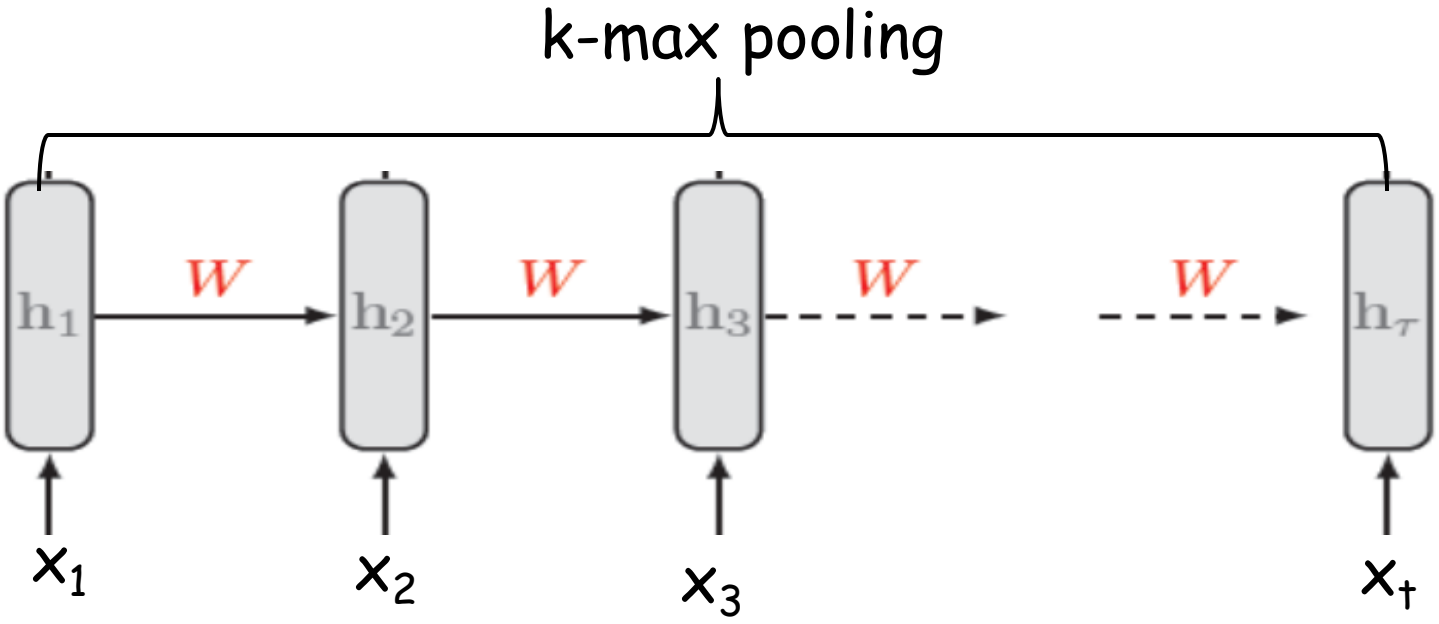
conv1d实际在最后一个维度进行

卷积扩展

- ✓ 卷积与循环结构的结合
- ✓ 图卷积网络

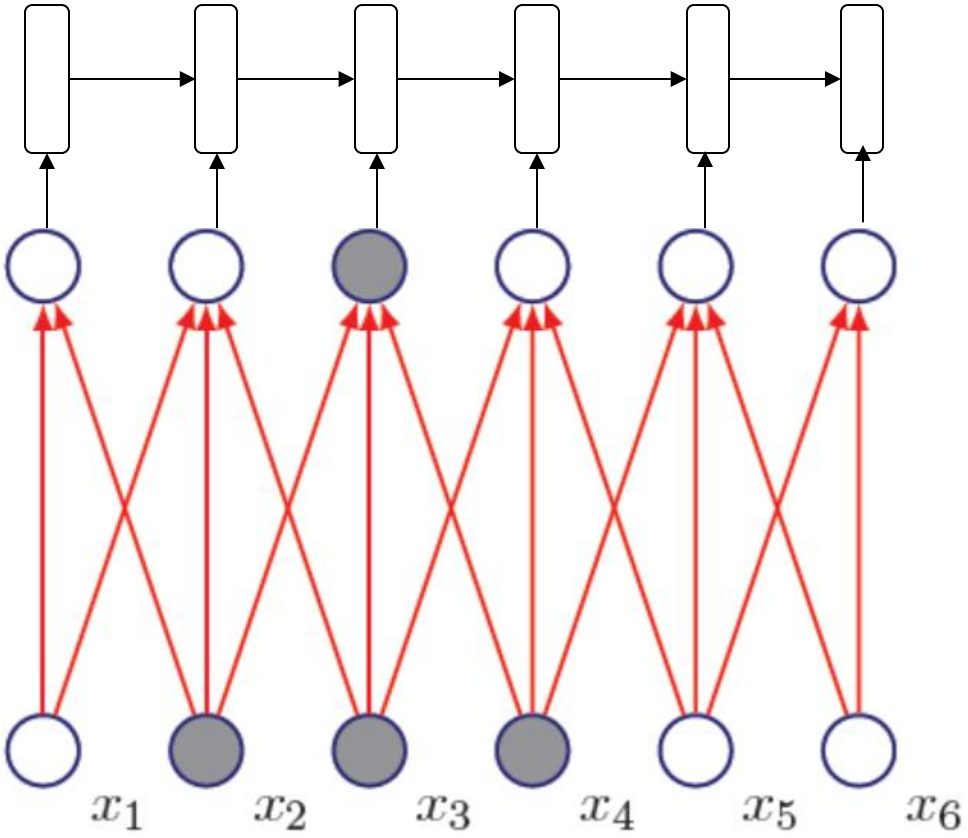
CNN结合RNN

- 循环神经网络+池化



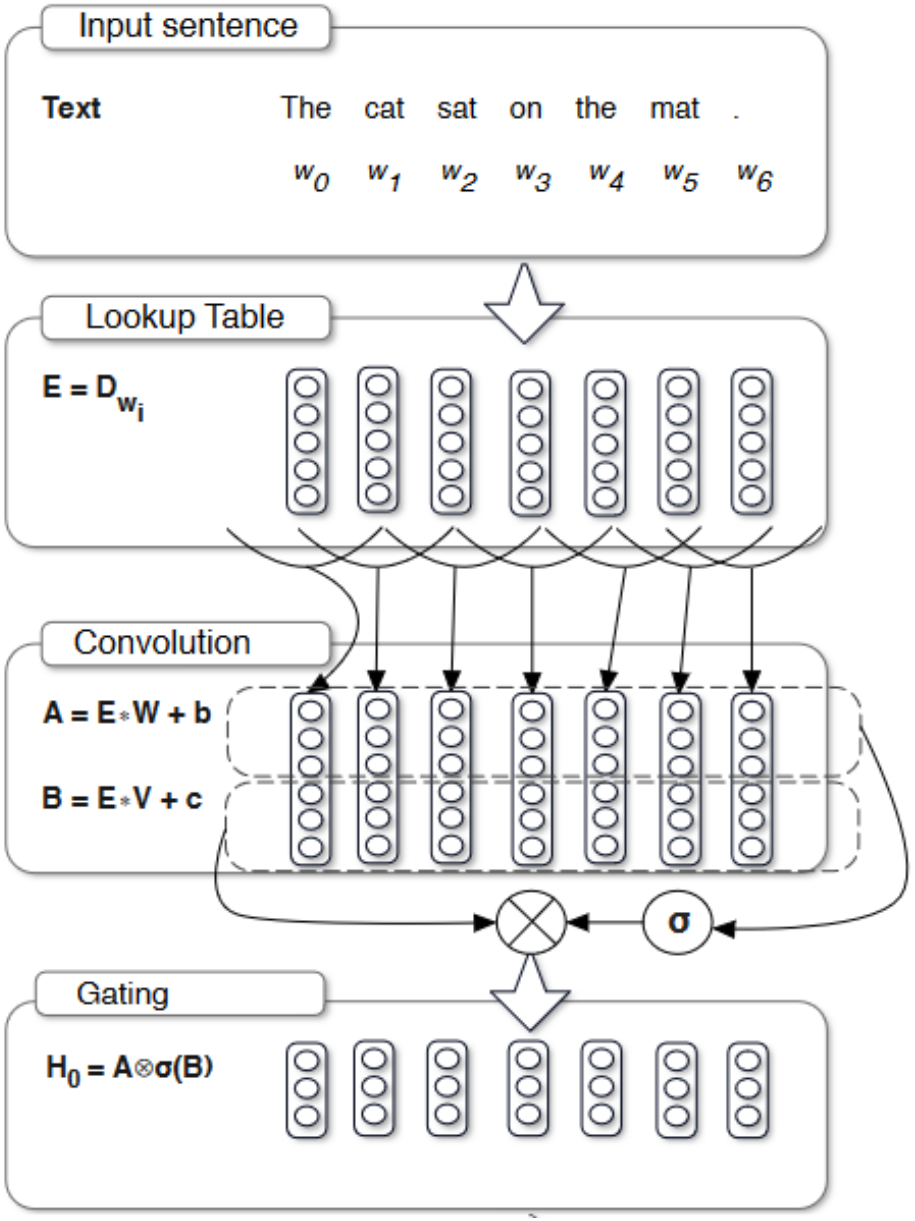
CNN结合RNN

- 卷积+循环神经网络



CNN结合RNN

- 门控卷积
- 上一页高阶版



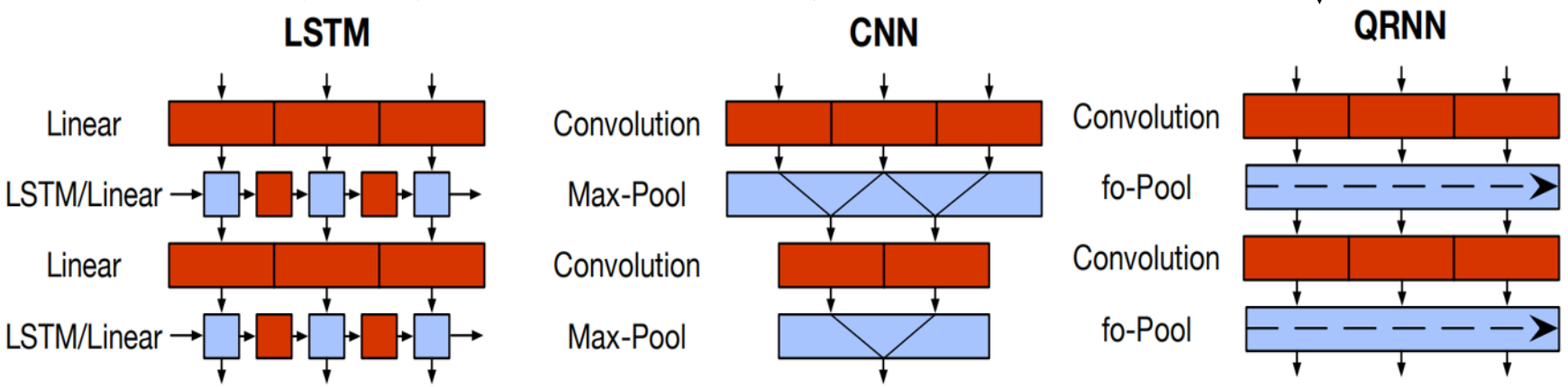
Quasi-Recurrent Neural Network (QRNN)

- Quasi:类似的、疑似的、拟的
- RNN的链式结构很适合处理序列文本，但存在长距离依赖问题；同样，无法并行化计算，也无法同时针对文档不同部分状态进行计算，比较耗时。
- 对比CNN：通过滤波器可以并行计算输入序列

- 想法: 如何把CNN和RNN的优点结合起来？

Quasi-Recurrent Neural Network (QRNN)

- 将CNN和RNN系模型相结合
- RNN利用往期信息+ CNN的可并行化



- 时间步上的卷积：

$$\mathbf{Z} = \tanh(\mathbf{W}_z * \mathbf{X})$$
$$\mathbf{F} = \sigma(\mathbf{W}_f * \mathbf{X})$$
$$\mathbf{O} = \sigma(\mathbf{W}_o * \mathbf{X})$$

W: 滤波器

if filter width=2

$$\mathbf{z}_t = \tanh(\mathbf{W}_z^1 \mathbf{x}_{t-1} + \mathbf{W}_z^2 \mathbf{x}_t)$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_f^1 \mathbf{x}_{t-1} + \mathbf{W}_f^2 \mathbf{x}_t)$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o^1 \mathbf{x}_{t-1} + \mathbf{W}_o^2 \mathbf{x}_t).$$

通过卷积的方式计算细胞候选、遗忘门、输出门

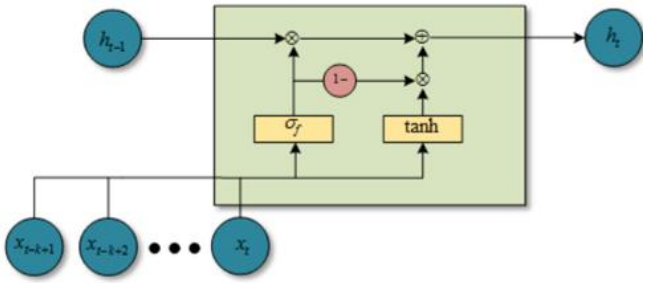
Quasi-Recurrent Neural Network

- RNN形式的池化　　Can be parallelized outside the filter

f-pooling: 使用遗忘门的动态平均池化

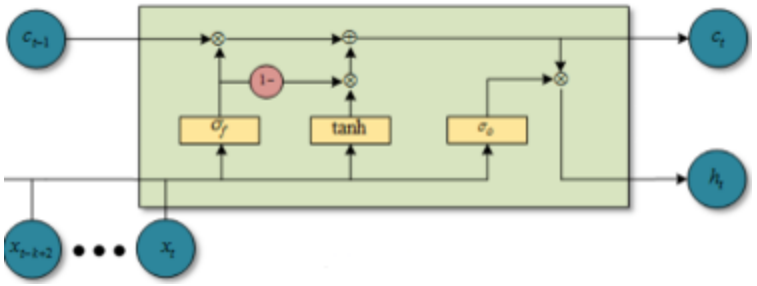
$$\mathbf{h}_t = \mathbf{f}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{z}_t$$

h and c 使用零初始化



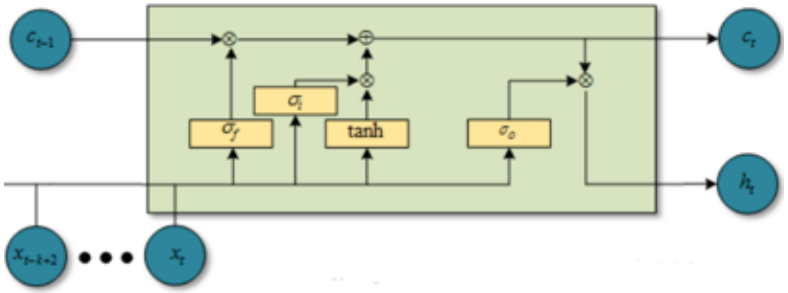
fo-pooling: 继续考虑输出门

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{z}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{c}_t.$$



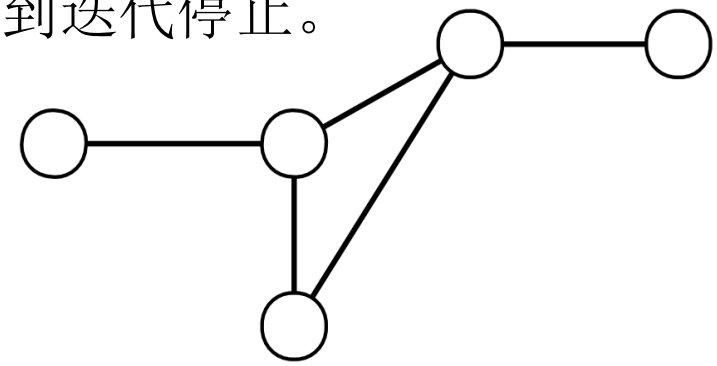
ifo-pooling: 考虑输入、遗忘、输出门

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{z}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{c}_t.$$

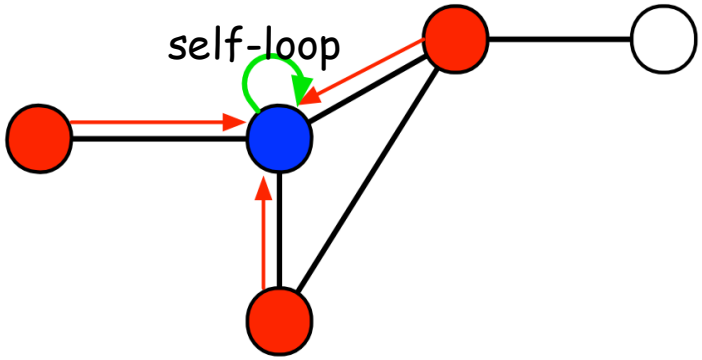


图卷积网络

- 图卷积：Graph Convolution Network (GCN)，将卷积操作用于图
- 文本的图建模：句子或token作为节点，句子或token的关联关系作为边。更新多轮，每轮节点更新使用邻居节点的表示进行计算，直到迭代停止。



Undirected graph



Update of the blue node

如：$h_v = ReLU(\sum_{u \in N(v)} Wx_u + b)$

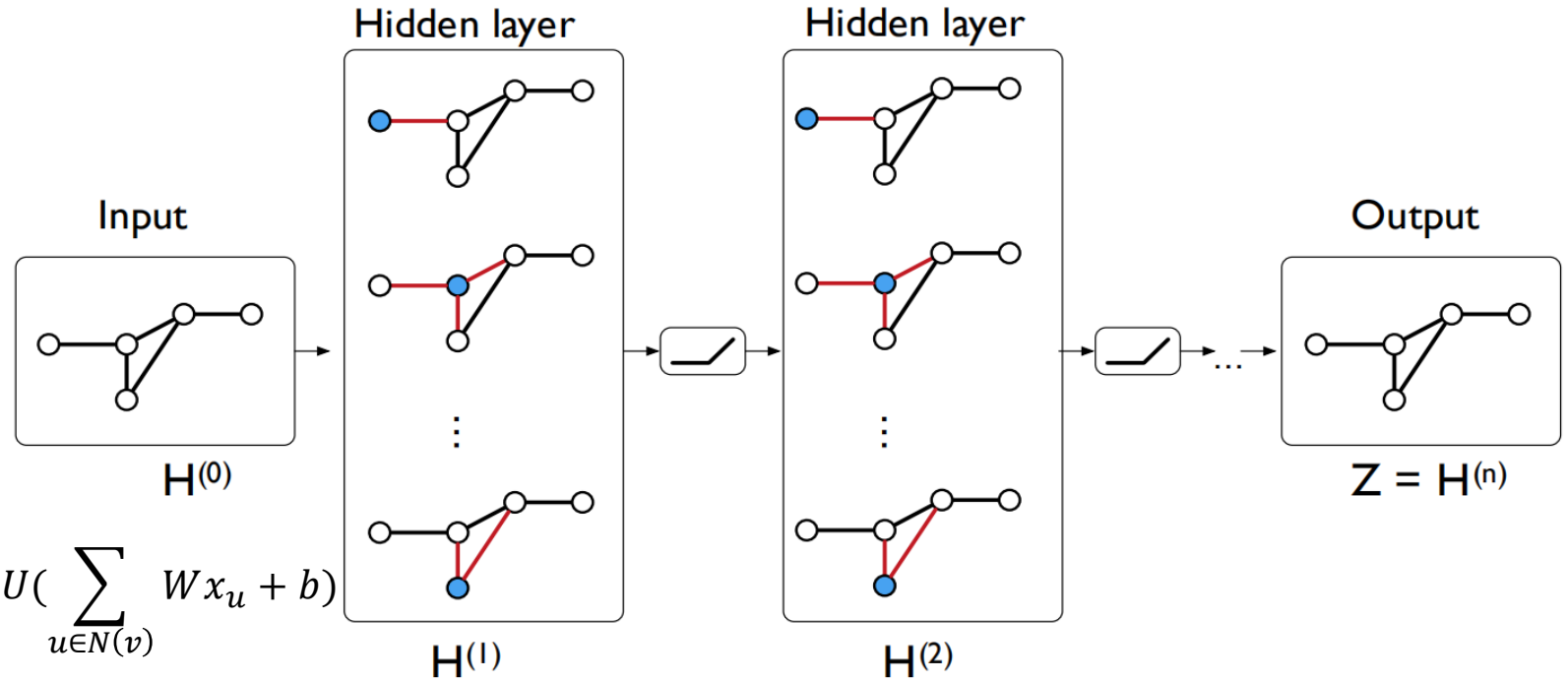
节点v更新后的表示

邻居节点，包括自身

Marcheggiani and Titov, 2017. EMNLP. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling

图卷积

- 图模型的更新可以迭代多次



$$h_v = ReLU(\sum_{u \in N(v)} W x_u + b)$$

$$h_v^{k+1} = ReLU(\frac{1}{|N(v)|} \sum_{u \in N(v)} W^k h_u^k + b^k)$$

Marcheggiani and Titov, 2017. EMNLP. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling

门控图卷积 (GatedGCN)

- 在基础图卷积之上

$$h_i^{\ell+1} = f_{\text{G-VCNN}}^{\ell}\left(h_i^{\ell} , \{h_j^{\ell} : j \to i\} \right) = \text{ReLU}\left(U^{\ell} h_i^{\ell} + V^{\ell} \sum_{j \to i} h_j^{\ell} \right)$$

- 加入门控机制

$$h_i^{\ell+1} = f_{\text{G-GCNN}}^{\ell}\left(h_i^{\ell} , \{h_j^{\ell} : j \to i\} \right) = \text{ReLU}\left(U^{\ell} h_i^{\ell} + \sum_{j \to i} \eta_{ij} \odot V^{\ell} h_j^{\ell} \right)$$

Edge gates $\eta_{ij} = \sigma\left(A^{\ell} h_i^{\ell} + B^{\ell} h_j^{\ell} \right)$

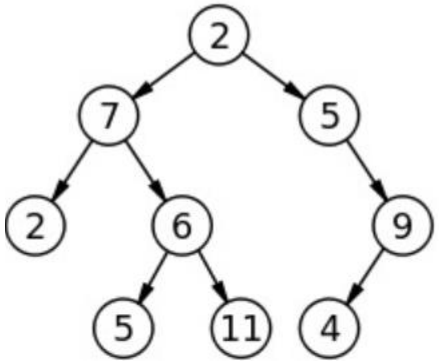
- 进一步加入残差

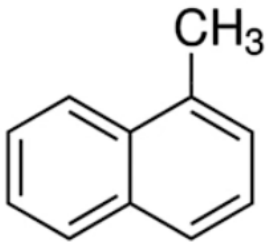
$$h_i^{\ell+1} = f^{\ell}\left(h_i^{\ell} , \{h_j^{\ell} : j \to i\} \right) + h_i^{\ell}$$

Bresson and Laurent. 2018. ICLR. Residual gated graph convnets

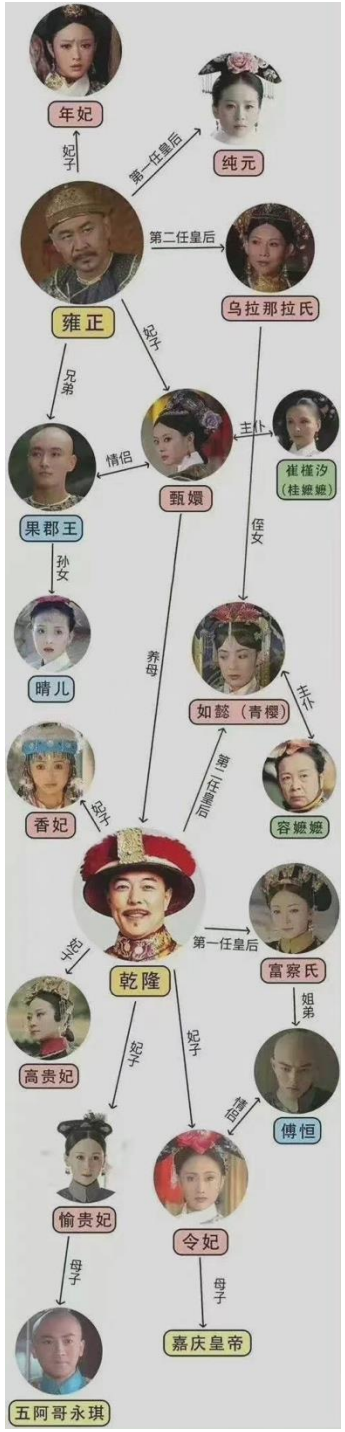
图数据

- Graph = (V, E), V=nodes, E=edges





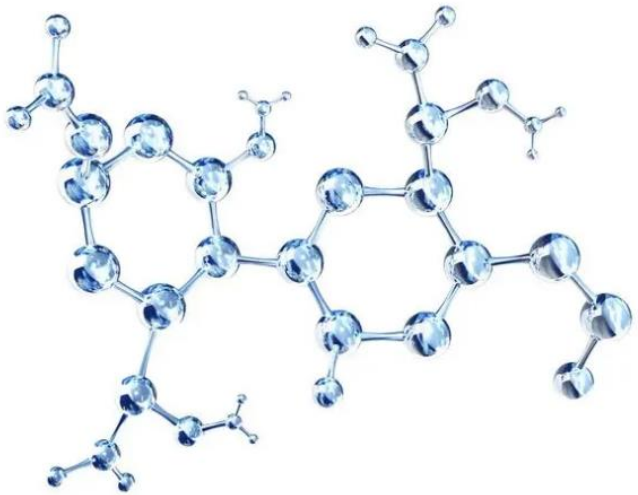
图类型	同构图	多关系图	异构图
# 节点类型	1	1	>1
# 边类型	1	>1	>=1



图数据任务

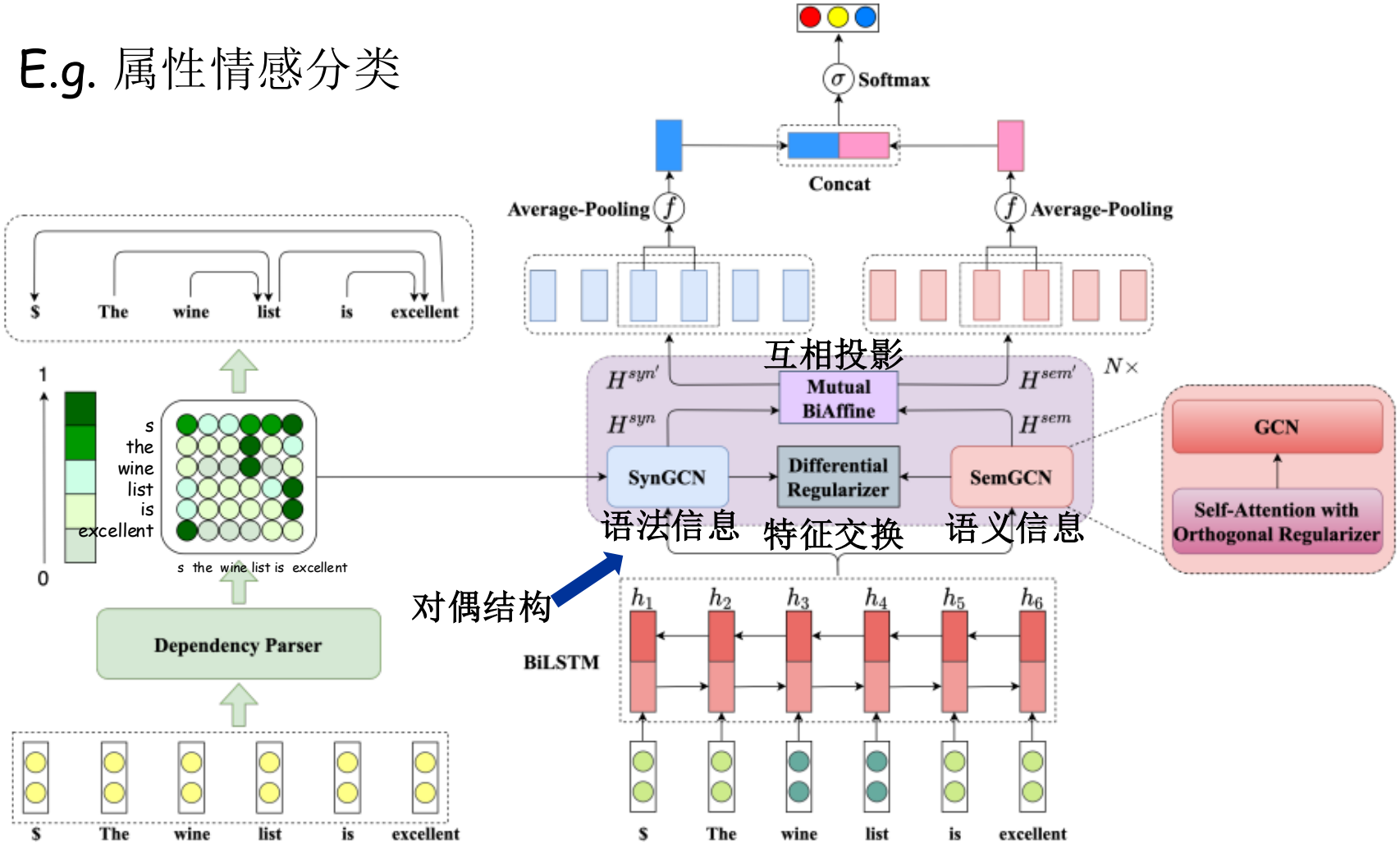
- 节点分类
- 边分类（需要显式学习边的表示）
- 图表示学习
- 图分类
- 图回归

要进行图整体的表示(readout)。可以对所有节点表示进行聚合(aggregation)，操作包括：平均化，求和，加权和，LSTM，最大池化。。。



图卷积案例：文本分类

- E.g. 属性情感分类



Li et al, 2021. ACL. Dual Graph Convolutional Networks for Aspect-based Sentiment Analysis

作业：卷积用于属性情感分类

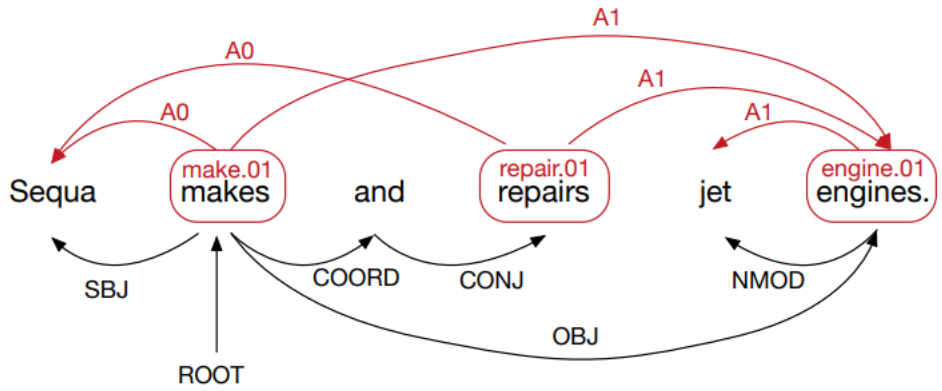
- 任务介绍：第4章PPT
- 方法：参考 Dual Graph Convolutional Networks for Aspect-based Sentiment Analysis
- 数据集：Restaurants数据集（忽略"conflict"标签的样本）
- 基础版： 只用实现SynGCN-head，即使用左侧GCN直接进行预测

```
<sentence id="2200">
    <text>The design and atmosphere is just as good.</text>
    <aspectTerms>
        <aspectTerm term="design" polarity="positive" from="4" to="10"/>
        <aspectTerm term="atmosphere" polarity="positive" from="15" to="25"/>
    </aspectTerms>
    <aspectCategories>
        <aspectCategory category="ambience" polari
```

Models	Restaurant	
	Accuracy	Macro-F1
SynGCN-head	82.93	75.29
SynGCN	83.74	76.97
SemGCN	83.29	76.30
DualGCN w/o BiAffine	82.84	75.31
DualGCN w/o R_O&R_D	82.93	75.79
DualGCN w/o R_O	83.56	77.43
DualGCN w/o R_D	83.65	76.34
DualGCN	**84.27**	**78.08**

图卷积案例：SRL

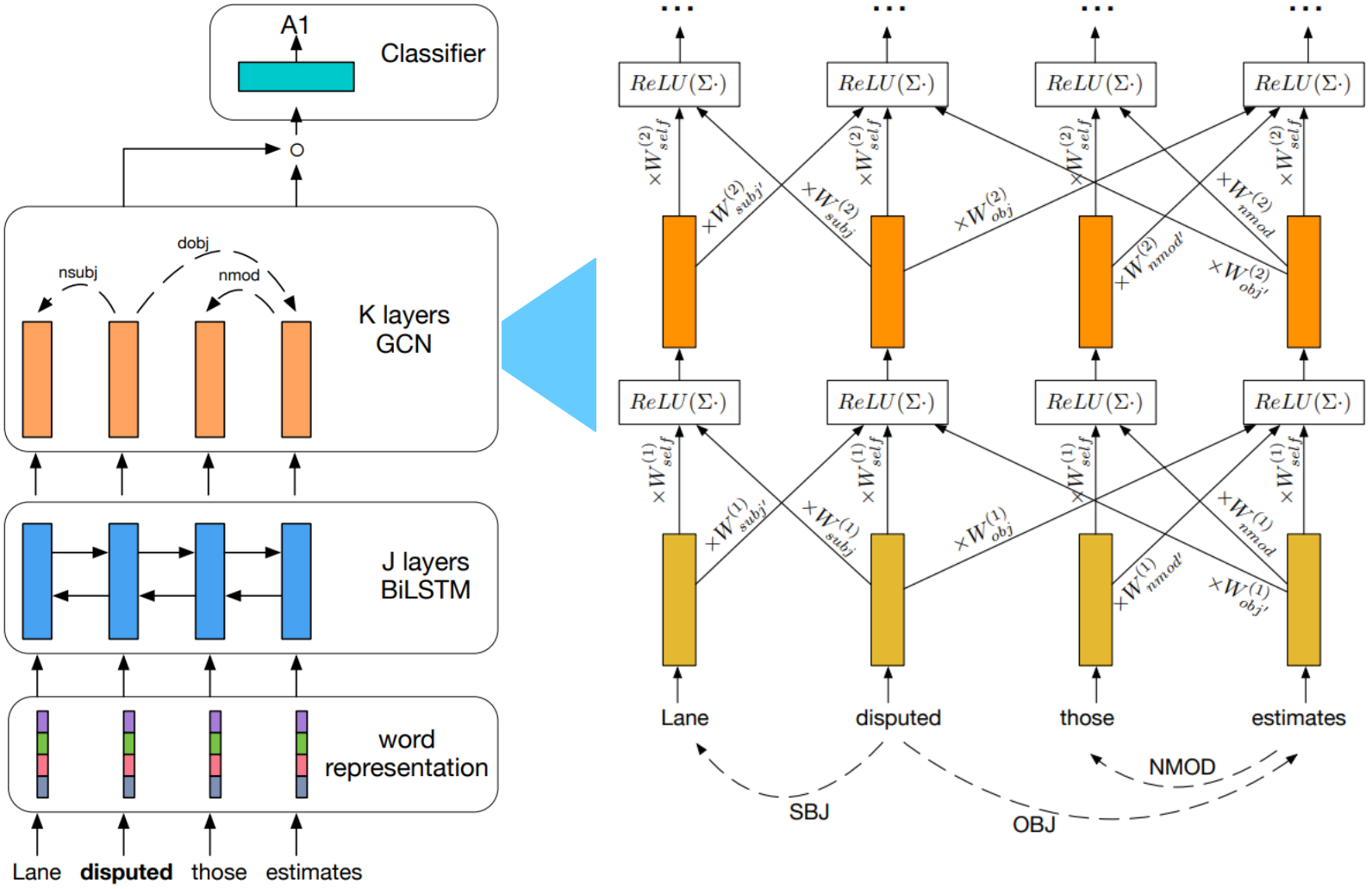
- SRL (Semantic role labeling 语义角色标注)任务：挖掘文本中“谁对谁做了什么”，即确定句子的谓词，并研究句子中各成分和谓词的关系，具体成分和关系类型使用预定义的类型



语义角色和依存紧密相关

Marcheggiani et al, 2017. EMNLP. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling

图卷积案例：SRL



显然，该模型也可以用于解决其他NLP任务