2020-2021学年秋季学期

自然语言处理

Natural Language Processing

授课教师： 胡玥

助　教： 于静

自然语言处理
Natural Language Processing

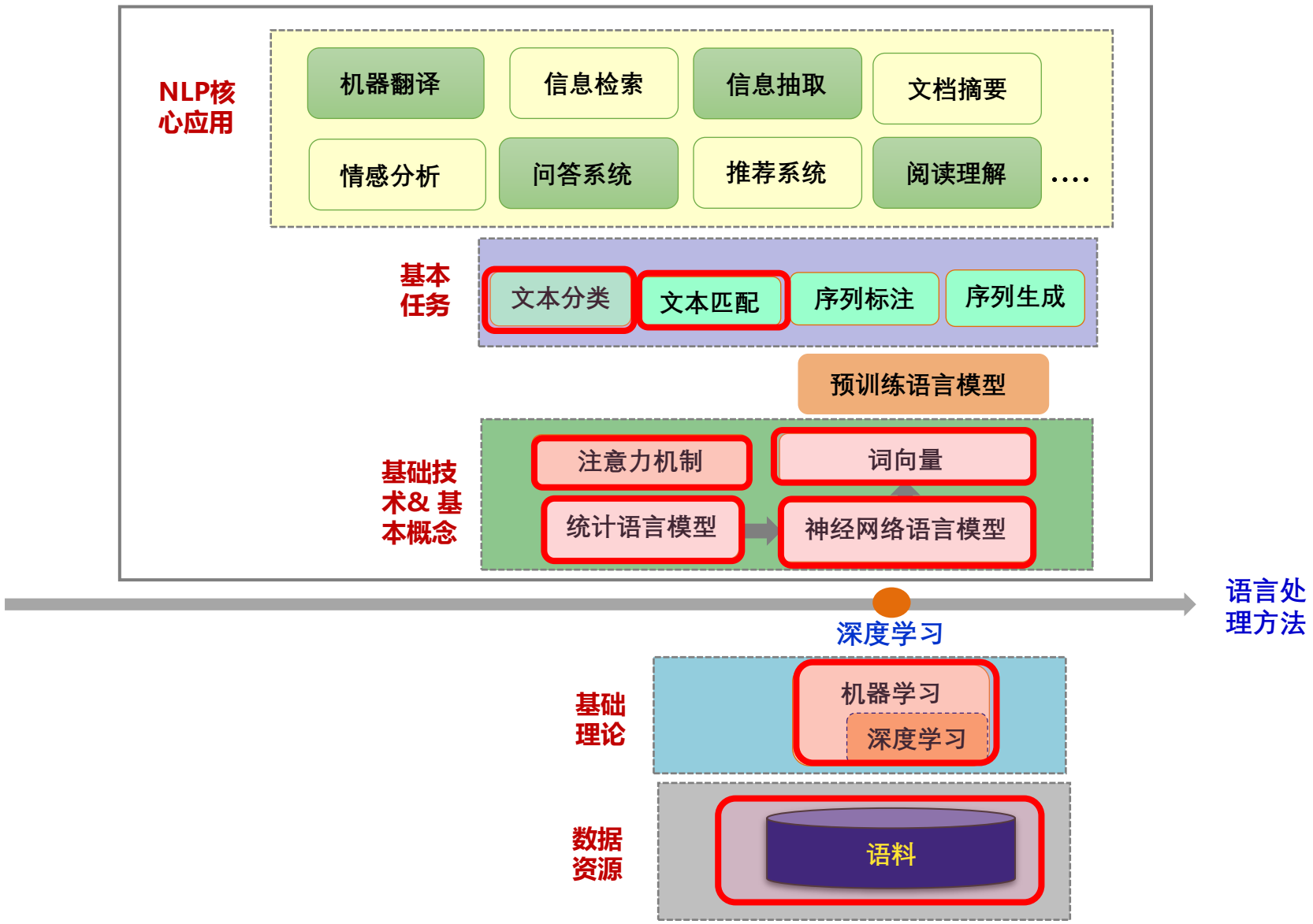
第 12 章　文本匹配

授课教师：　胡玥

授课时间：2020.10

课程编码 201M4005H　课程名称 自然语言处理　授课团队名单 胡玥 、于静

基于深度学习的自然语言处理课程内容



概 要

本章主要内容：

本章主要围绕句子级/文档级的文本匹配任务介绍三类匹配方法及模型包括：

1. 句子编码匹配模型 2.交互聚合匹配模型

2. 本章教学目的：

使学生理解并掌握文本匹配的各种方法及经典模型，为后继的自然语言处理任务打好基础。

内 容 提 要

12.1 概述

■ 文本匹配任务

文本匹配是自然语言处理中的一个核心问题，很多自然语言处理的任务都可以抽象成文本匹配问题，例如信息检索可以归结成查询项和文档的匹配，问答系统可以归结为问题和候选答案的匹配，对话系统可以归结为对话和回复的匹配等等。

■ 文本匹配问题描述

给定一段文本 q_i ,和另一段文本的列表 $D_i = \{d_{i1}, d_{i2}, \ldots. d_{in}\}$根据任务给定关系，计算所有的文本对$<q_i, D_i^{(j)}>$ 针对任务关系的匹配度打分，从中选出与q_i匹配度最高的文本段

不同的任务有不同的关系

12.1 概述

■ **自然语言处理中句子对相关的研究任务：**

1. Semantic Textual Similarity(STS)：判断两个句子的语义相似程度；

2. Natural Language Inference (NLI)：也叫 Recognizing Textual Entailment (RTE)，判断两个句子在语义上是否存在推断关系[蕴涵entailment, 矛盾contradiction and中立neutral 关系）相对任务1更复杂，不仅仅是考虑相似，而且也考虑了推理；

3. Paraphrase Identification (PI): 判断两个句子是否表达同样的意思；

4. Question Answering (QA) :主要是指选择出来最符合问题的答案，是在给定的答案中进行选择，而不是生成；

5. Machine Comprehension (MC) :判断一个句子和一个段落之间的关系，从大段落中找出存在答案的小段落，对比的两个内容更加复杂一些。

12.1 概述

■ 匹配方法

★ 规则方法： 不同任务需要专门构建特征规则

★ 统计方法： 特征工程+算法（PRanking / margin/ SVM/LR……）

以上二种传统的文本匹配方法主要集中在人工定义特征之上的关系学习，焦点在于如何人工提取的特征和设置合适的文本匹配学习算法来学习到最优的匹配模

★ 深度学习方法： （深度文本匹配模型）

能够从大量的样本中自动提取出词语之间的关系并能结合短语匹配中的结构信息和文本匹配的层次化特性，更精细地描述文本匹配问题.

本章讨论：句/文档粒度的基于深度学习的匹配方法

12.1 概述

■ **匹配方法分类** （不同的文献有不同的分类体系）

参考文献【2】按文档匹配过程中交互匹配粒度和方式将匹配方法分为三类：

- 基于单语义文档表达的深度学习模型
- 基于多语义文档表达的深度学习模型
- 直接建模匹配模式的深度学习模型

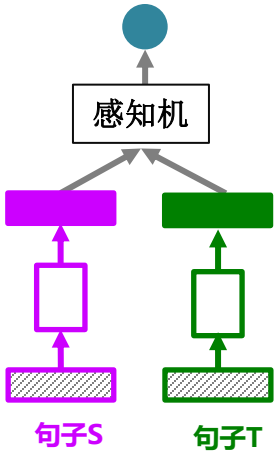
参考文献【1】按文档匹配模型结构将匹配方法分为二类：

- 句子编码模型（孪生网络）
- 交互聚合模型（匹配整合）

本章采用文献【1】的分类体系

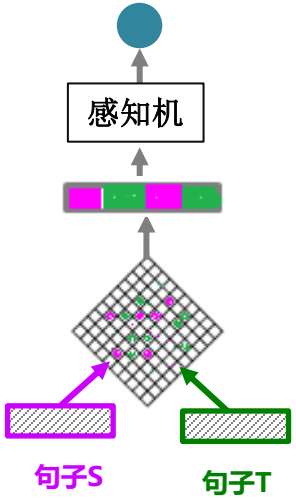
12.1 概述

- 句子编码模型（孪生网络）



模型先得到句子S/T的句向量，然后通过对比句向量计算匹配度。

- 交互聚合模型（匹配整合）



模型通常在聚合之前使用不同的单词对齐机制，直接捕获匹配特征形成含有对齐信息的聚合向量然后用聚合向量计算匹配度

内 容 提 要

12.1 概述

12.2 句子编码匹配模型

12.3 交互聚合匹配模型

12.2 句子编码匹配模型

★ DSSM：(基于DNN的经典模型)

- **动机**：用神经网络生成代表句子语义的句向量，用计算向量的距离的方法取代传统的匹配算法

- **模型结构**

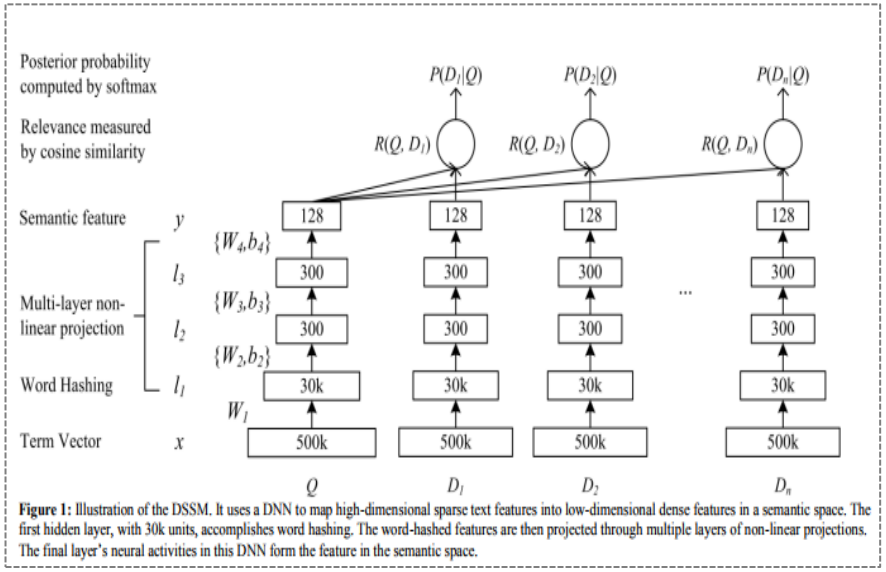


Figure 1: Illustration of the DSSM. It uses a DNN to map high-dimensional sparse text features into low-dimensional dense features in a semantic space. The first hidden layer, with 30k units, accomplishes word hashing. The word-hashed features are then projected through multiple layers of non-linear projections. The final layer's neural activities in this DNN form the feature in the semantic space.

输出：
$$P(D|Q) = \frac{exp(\gamma R(Q,D))}{\sum_{D' \in \mathbf{D}} exp(\gamma R(Q,D'))}$$

$$R(Q,D) = cosine(y_Q, y_D) = \frac{y_Q^T y_D}{\|y_Q\| \, \|y_D\|}$$

句表示：128维的句子向量表示

$$l_1 = W_1 x$$
$$l_i = f(W_i l_{i-1} + b_i), i = 2, \ldots, N-1$$
$$y = f(W_N l_{N-1} + b_N)$$

激活函数
$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

每个句子通过哈希映射到30K维向量；
输入给三层全连接神经网络

输入：Q 和文档集 D

Huang P S, et al. Learning deep structured semantic models for web search using clickthrough data. CIKM, 2013

12.2 句子编码匹配模型

■ 模型学习

训练数据：点击日志里包含了用户搜索的query和用户点击的doc，可以假定如果用户在当前query下对doc进行了点击，则该query与doc是相关的。由此，可以通过点击日志构造训练集与测试集。

输出打分：通过softmax 可以把query 与样本 doc 的语义相似性转化为后验概率：

$$P(D|Q) = \frac{exp(\gamma R(Q,D))}{\sum_{D' \in \mathbf{D}} exp(\gamma R(Q,D'))}$$

其中 gamma是一个softmax函数的平滑因子， D表示被排序的候选文档集合，在实际中，对于正样本，每一个（query， 点击doc）对，使用 (Q, D^+) 表示；对于负样本，随机选择4个曝光但未点击的doc。

优化目标： 通过极大似然估计来最小化损失函数：

$$L(\Lambda) = -log \prod_{(Q,D^+)} P(D^+|Q)$$

模型通过随机梯度下降（SGD）来进行优化，最终可以得到各网络层的参数W_i ,b_i

12.2 句子编码匹配模型

- 模型预测

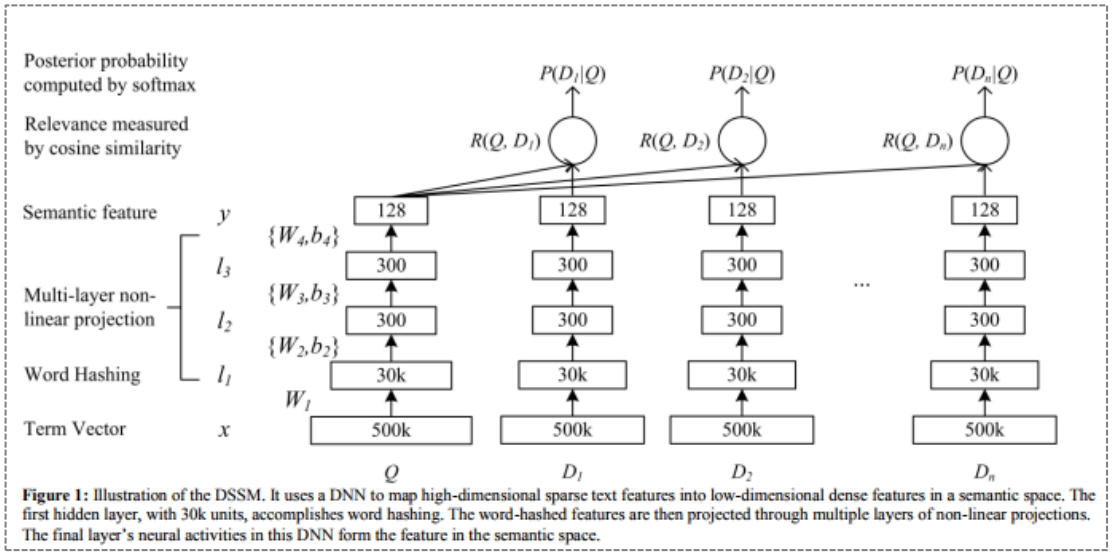


Figure 1: Illustration of the DSSM. It uses a DNN to map high-dimensional sparse text features into low-dimensional dense features in a semantic space. The first hidden layer, with 30k units, accomplishes word hashing. The word-hashed features are then projected through multiple layers of non-linear projections. The final layer's neural activities in this DNN form the feature in the semantic space.

DSSM模型特点：

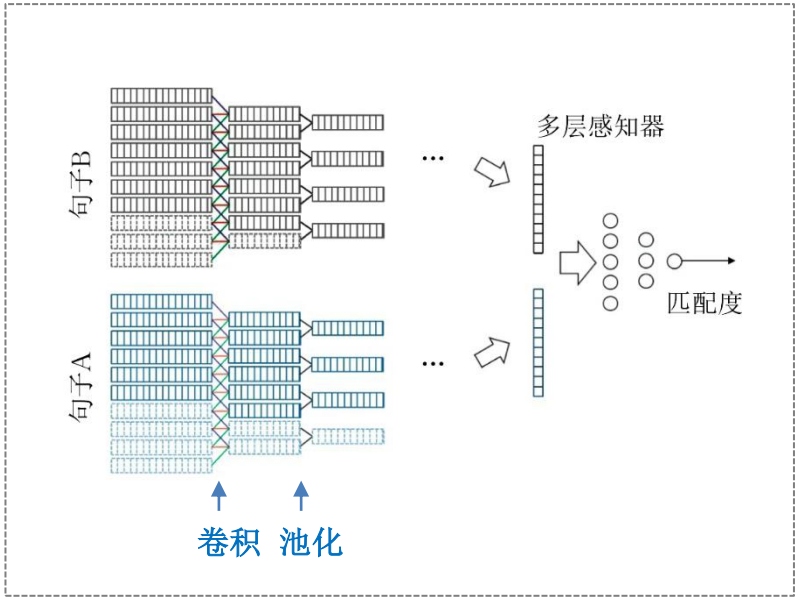
优点：省去了人工的特征工程传统的输入层是用 Embedding 的方式或者主题模型的方式来直接做词的映射（这样会给整个模型引入误差）DSSM 采用统一的有监督训练，不需要在中间过程做无监督模型的映射，因此精准度会比较高。

缺陷：1、没有考虑到单词之间的时序联系，2、相似度匹配用的余弦相似度是一个无参的匹配公式。

12.2 句子编码匹配模型

★ ARC-I (基于CNN)

■ 模型结构



输入：句子A和B

运算关系：

1. padding句子A/B到定长
2. 分别做多轮的卷积+池化运算
3. 拼接两个向量
4. 输入给多层感知机

输出：句子A和B的匹配度

优势：减少参数量，建模全局信息

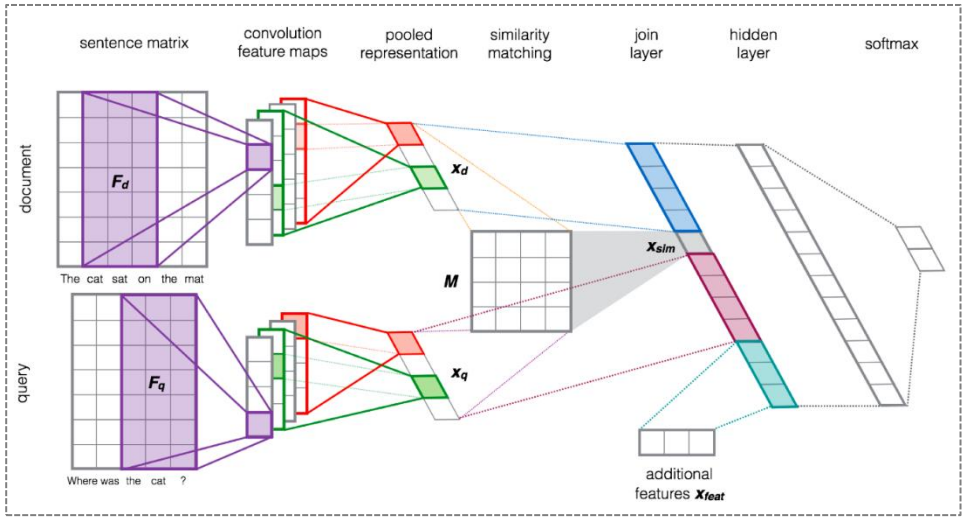
缺点：在于将两个句子encode成句向量之后再用多层感知机进行分类，这种方法就很明显没有体现出句子之间的交互操作

Hu, Baotian, ， Convolutional Neural Network Architectures for Matching Natural Language Sentences.” NIPS (2014).

12.2 句子编码匹配模型

★ CDNN (基于CNN)

■ 模型结构



输入：句子d和q

运算关系：

1. 分别做卷积+池化运算，得到句向量
2. 计算相似度 $\mathbf{X_{sim}} = \boldsymbol{X_q^T M X_d}$
3. 计算单词重叠数等其他特征
4. 拼接句向量、相似度和其他特征
5. 输入给多层感知机

输出：句子d和q的匹配度

■ 创新点：引入相似矩阵

Aliaksei , et al. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks." SIGIR (2015).

12.2 句子编码匹配模型

★ CDNN (基于CNN)

■ 模型学习

优化目标： 最小化交叉熵损失函数

$$\begin{aligned} \mathcal{C} &= -\log \prod_{i=1}^{N} p(y_i|\mathbf{q}_i, \mathbf{d}_i) + \lambda \|\theta\|_2^2 \\ &= -\sum_{i=1}^{N} [y_i \log \mathbf{a}_i + (1-y_i)\log(1-\mathbf{a}_i)] + \lambda \|\theta\|_2^d \end{aligned}$$

$$\theta = \{\mathbf{W}; \mathbf{F}_q; \mathbf{b}_q; \mathbf{F}_d; \mathbf{b}_d; \mathbf{M}; \mathbf{w}_h; b_h; \mathbf{w}_s; b_s\}$$

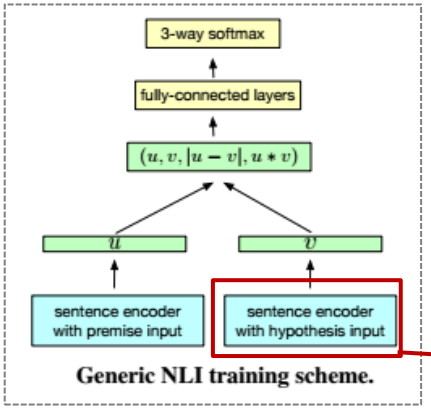
优势：增加了匹配特征

12.2 句子编码匹配模型

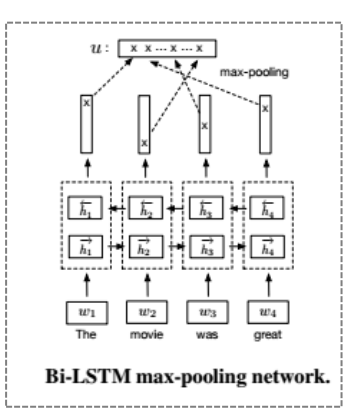
★ InferSent (基于RNN)

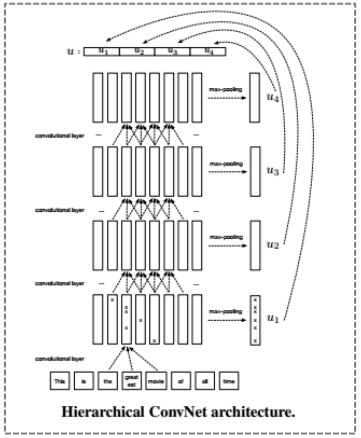
■ 动机：在文本蕴含任务上建立适合各种任务的通用句表示（句向量）

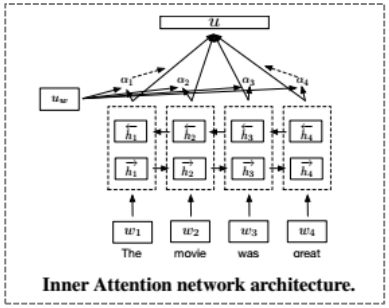
通用文本蕴含架构



各种句表示方法：







BiGRU-last network

Bi-LSTM mean-pooling network

LSTM network

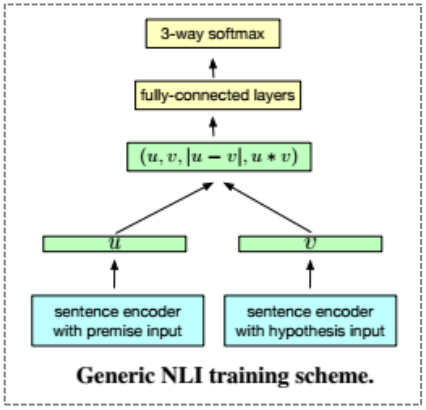
GRU network

Alexis et al.. Supervised learning of universal sentence representations from natural language inference data. (EMNLP). 2017

12.2 句子编码匹配模型

- 实验：

通用文本蕴含架构



Generic NLI training scheme.

Model	dim	NLI dev	NLI test	Transfer micro	Transfer macro
LSTM	2048	81.9	80.7	79.5	78.6
GRU	4096	82.4	81.8	81.7	80.9
BiGRU-last	4096	81.3	80.9	82.9	81.7
BiLSTM-Mean	4096	79.0	78.2	83.1	81.7
Inner-attention	4096	82.3	82.5	82.1	81.0
HConvNet	4096	83.7	83.4	82.0	80.9
BiLSTM-Max	4096	**85.0**	**84.5**	**85.2**	**83.7**

Table 3: **Performance of sentence encoder architectures** on SNLI and (aggregated) transfer tasks. Dimensions of embeddings were selected according to best aggregated scores (see Figure 5).

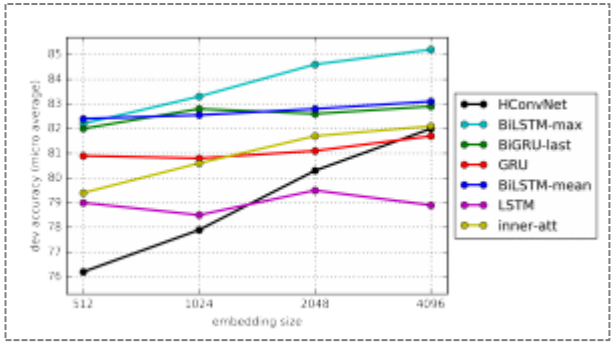
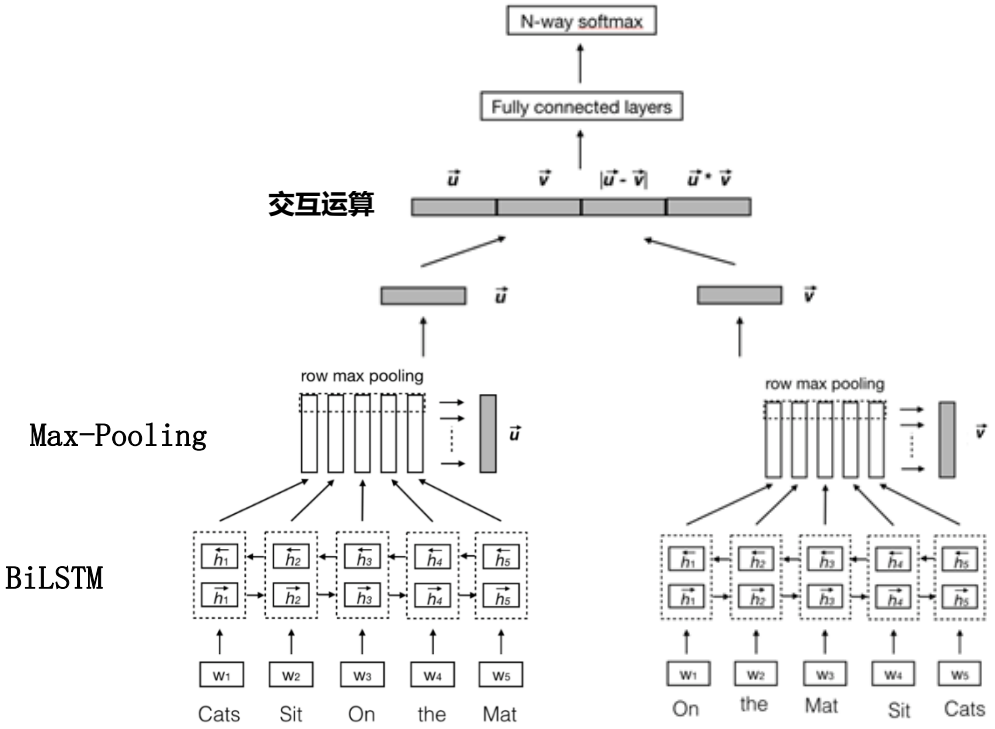


Figure 5: **Transfer performance w.r.t. embedding size** using the micro aggregation method.

结论：BiLSTM-Max 效果最好

12.2 句子编码匹配模型

■ 模型结构：



输出：句子A和B的匹配度

运算关系：

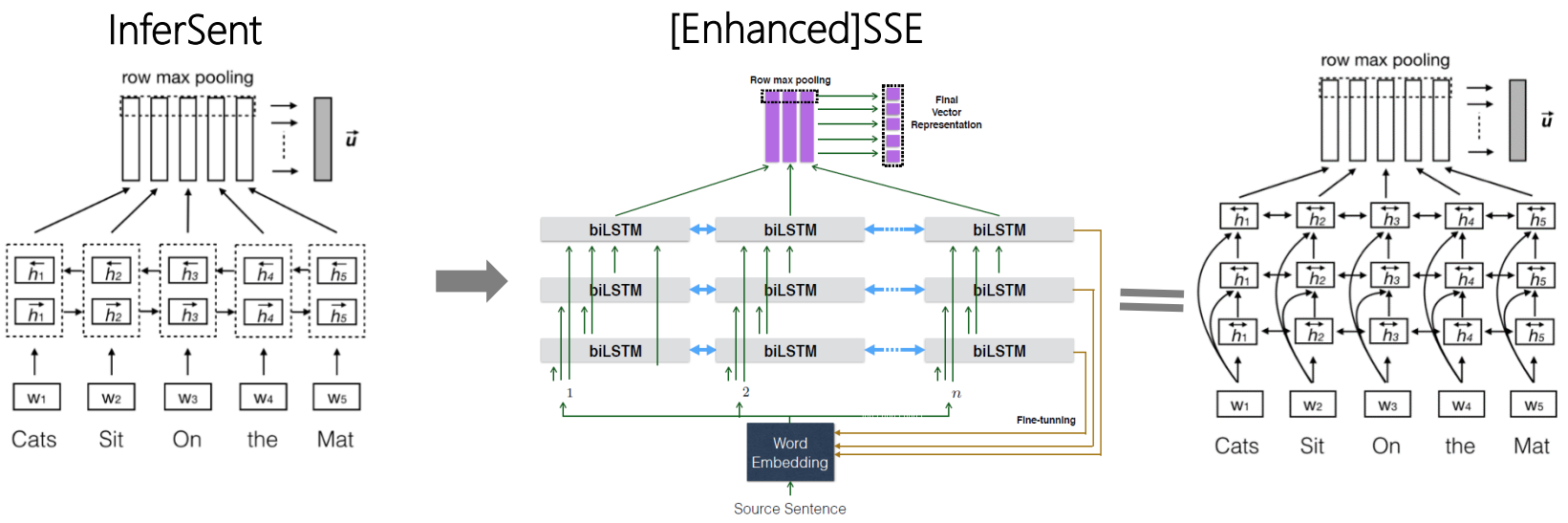
1. 将A，B分别通过BiLSTM-Max表示成句向量
2. 将2个句向量交互运算输入给多层感知机

输入：句子A和B

12.2 句子编码匹配模型

★[Enhanced]SSE

- 句表示：



$$x_t^1 = w_t$$

$$x_t^i = [w_t, h_t^{i-1}, h_t^{i-2}, ...h_t^1] \quad (\text{for } i > 1)$$

$$h_t^i = \text{bilstm}^i(x_t^i, t), \forall t \in [1, 2, ..., n]$$

句表示：$H^m = (h_1^m, h_2^m, ..., h_n^m)$

$$v = max(H^m)$$

12.2 句子编码匹配模型

- 实验：

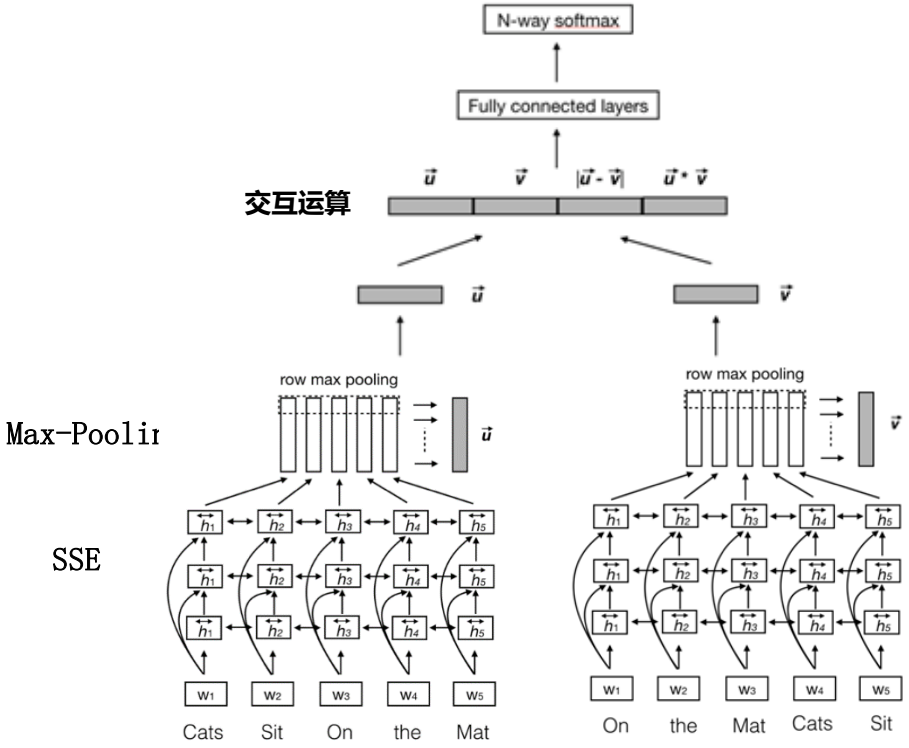
Model	Accuracy		
	SNLI	Multi-NLI Matched	Multi-NLI Mismatched
CBOW (Williams et al., 2017)	80.6	65.2	64.6
biLSTM Encoder (Williams et al., 2017)	81.5	67.5	67.1
300D Tree-CNN Encoder (Mou et al., 2015)	82.1	–	–
300D SPINN-PI Encoder (Bowman et al., 2016)	83.2	–	–
300D NSE Encoder (Munkhdalai and Yu, 2016)	84.6	–	–
biLSTM-Max Encoder (Conneau et al., 2017)	84.5	–	–
Our biLSTM-Max Encoder	85.2	71.7	71.2
Our Shortcut-Stacked Encoder	**86.1**	**74.6**	**73.6**

Final Test Results on SNLI and Multi-NLI datasets.

结论：SSE 效果最好

12.2 句子编码匹配模型

■ 模型结构：



输出：句子A和B的匹配度

运算关系：

1. 将A，B分别通过SSE表示成句向量
2. 将2个句向量交互运算输入给多层感知机

输入：句子A和B

优势：捕获更深层语义表示

内 容 提 要

12.1 概述

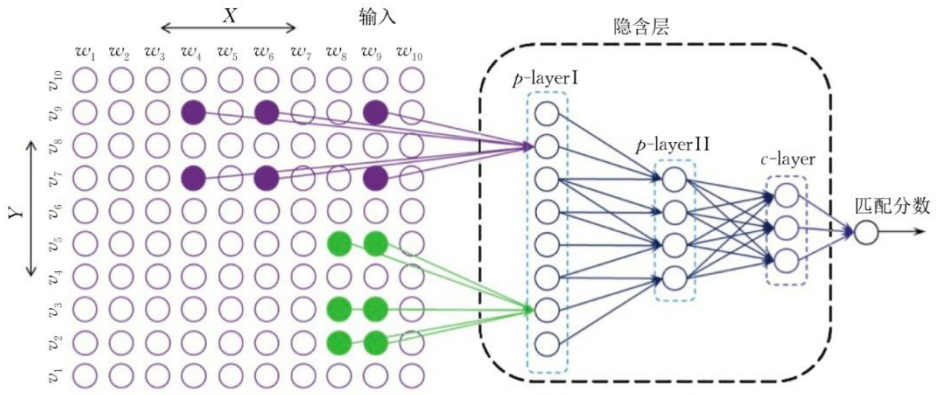
12.2 句子编码匹配模型

12.3 交互聚合匹配模型

12.3 交互聚合匹配模型

★ DM (基于DNN)

该模型的提出基于文本匹配过程的两个直觉：1）Localness:两个语义相关的文本应该存在词级别的共现模式（co-ouccurence pattern of words）；2）Hierarchy:共现模式可能在不同的词抽象层次中出现



1. 基于词袋模型，把句子X/Y表达成词向量序列
2. 根据预训练好的主题模型筛选局部块
3. 同一个局部块的单词统一连接到下一层的隐节点
4. 输入给多层感知机

优势：善于捕捉主题层面上的匹配

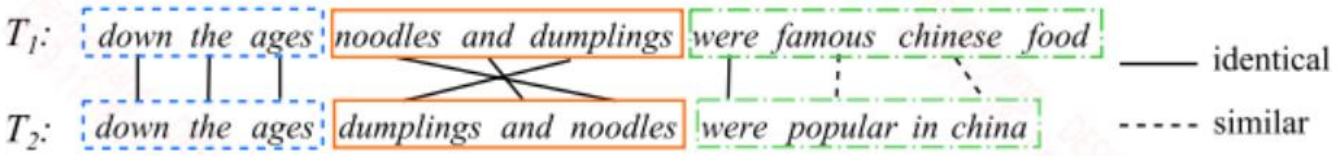
缺点：忽略了词在句子中的顺序

Lu, Zhengdong and Hang Li. “A Deep Architecture for Matching Short Texts.” NIPS (2013).

12.3 交互聚合匹配模型

★ Match Pyramid (基于CNN)　　早期论文，attention还没有在NLP中广泛的应用

例：两个语义相同的句子，在词级别上就有很多相同或相似对应的词

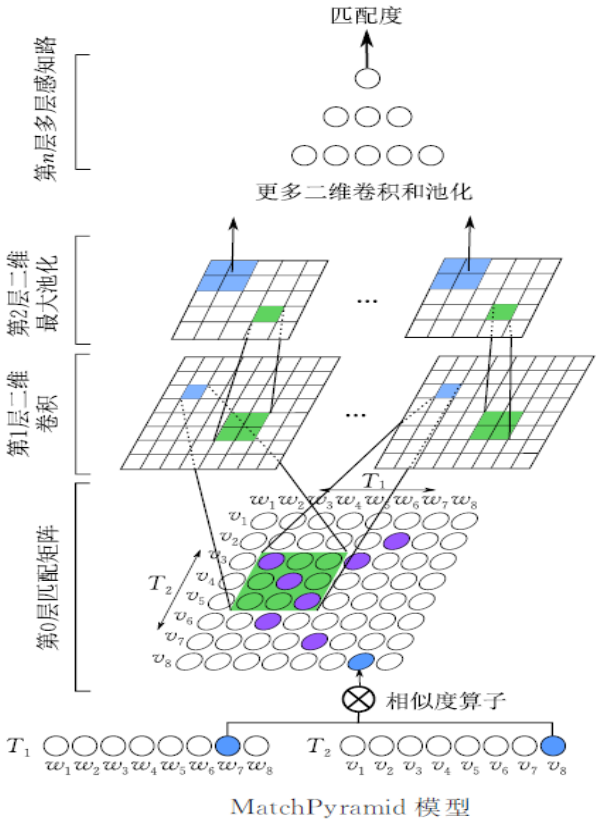


思路：将文本匹配问题转换成图像识别的问题进行求解

Pang, Liang, et al. Text Matching as Image Recognition." AAAI (2016).

12.3 交互聚合匹配模型

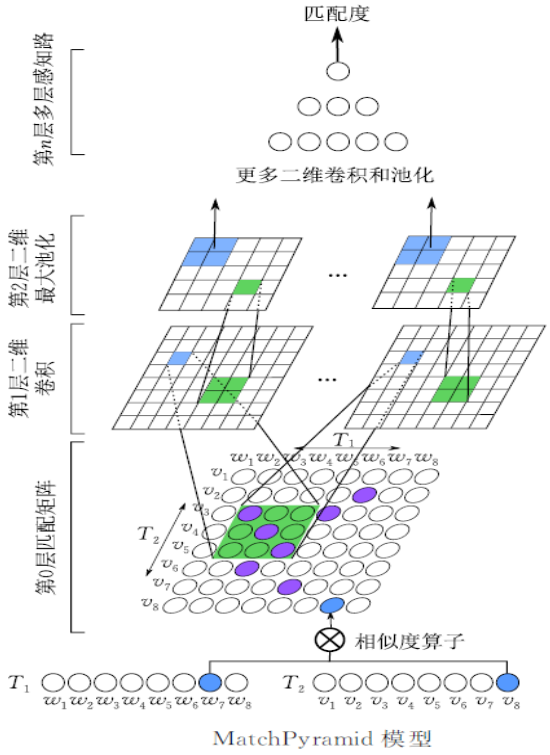
Match Pyramid 模型结构



Match Pyramid 模型包含三部分：

- 匹配矩阵层
- 卷积层
- MLP层

12.3 交互聚合匹配模型

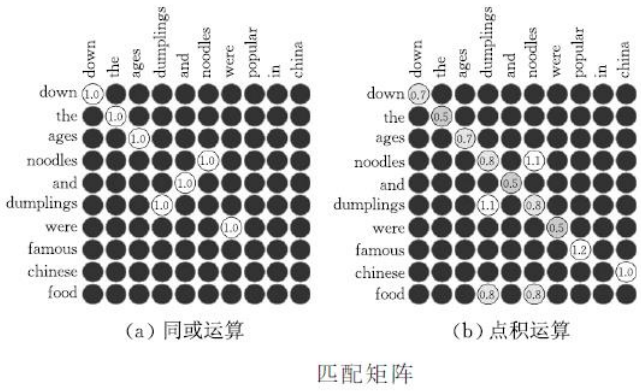


- **匹配矩阵层**

两个句子中的词两两计算相似度得到一个矩阵

$$\mathbf{M}_{ij} = w_i \otimes v_j.$$

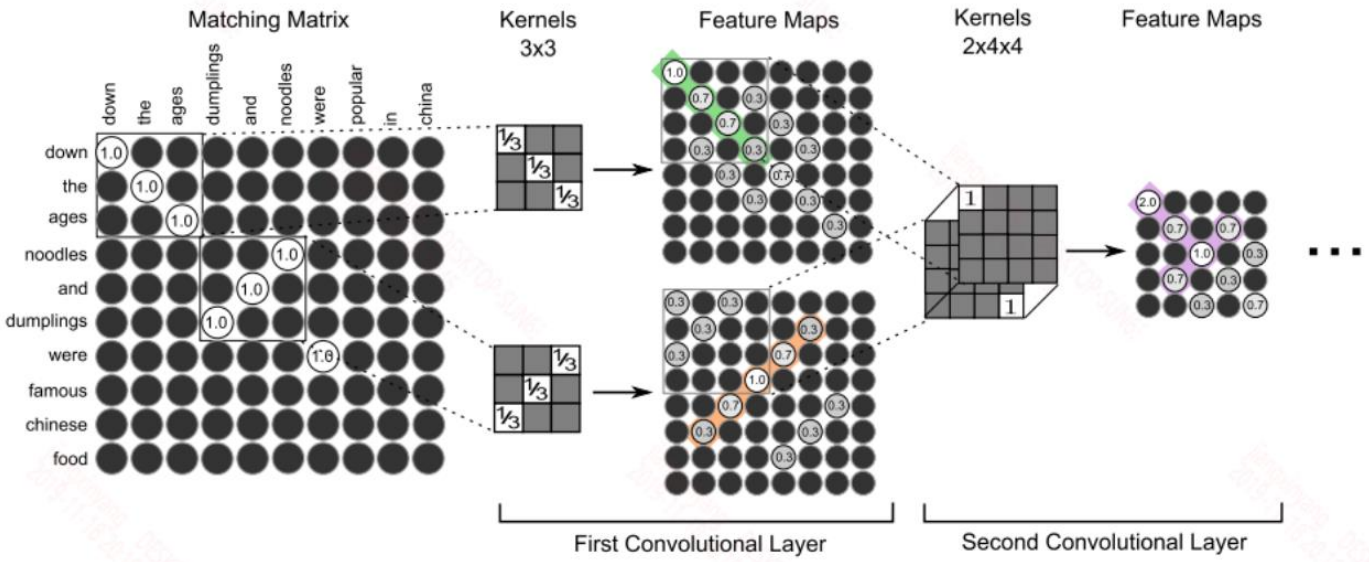
句子1中的第i个词和第2个句子中的第j个词的相似度值



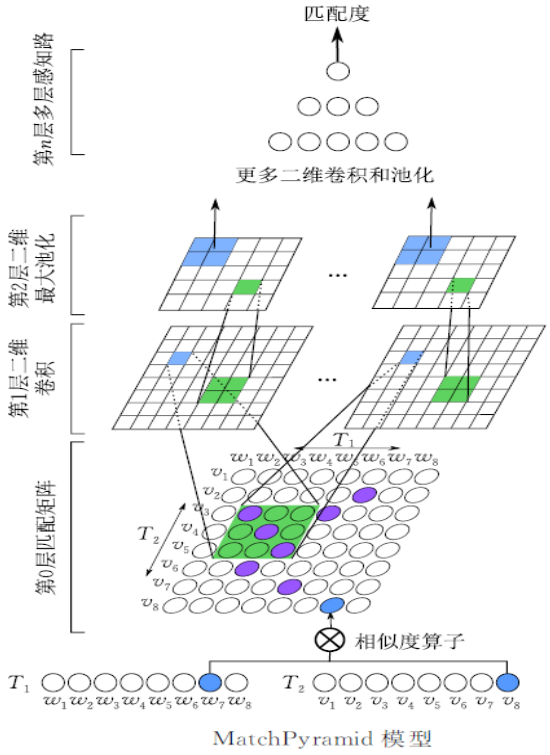
相似度定义：同或关系、余弦相似度或点积

12.3 交互聚合匹配模型

- 卷积层



12.3 交互聚合匹配模型



- **MLP层**

  采用两层的全连接层

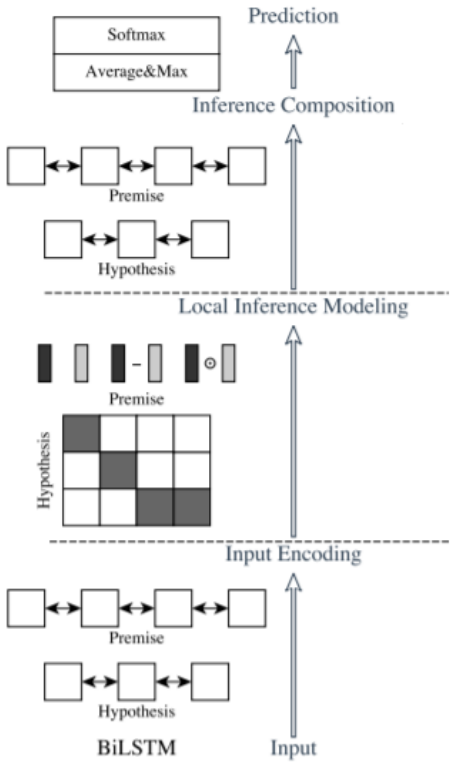
优势：匹配矩阵可以包含最细粒度的匹配信息

12.3 交互聚合匹配模型

★ ESIM (基于RNN)

ESIM模型主要是用来做文本推理的，给定一个前提premise p 推导出假设hypothesis h

该模型也可以做文本匹配，判断目标是两个序列是否是同义句

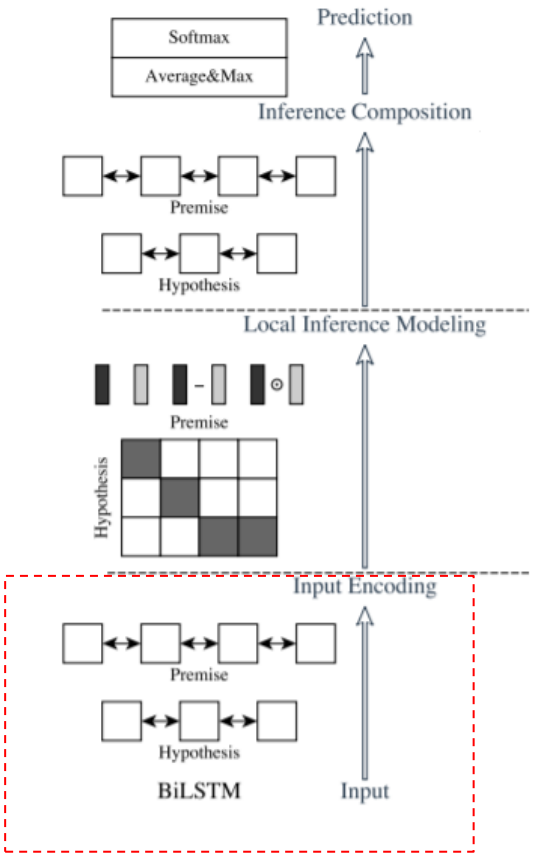


ESIM模型包含四部分：

- Input Encoding
- Local Inference Modeling
- Inference Composition
- Prediction

Qian Chen, et al. Enhanced LSTM for natural language inference. (ACL). 2017

12.3 交互聚合匹配模型



- **Input Encoding**

输入： 句子X(p) 和 Y(h)

把句子X/Y表达成词向量序列，输入第一层BiLSTM进行编码

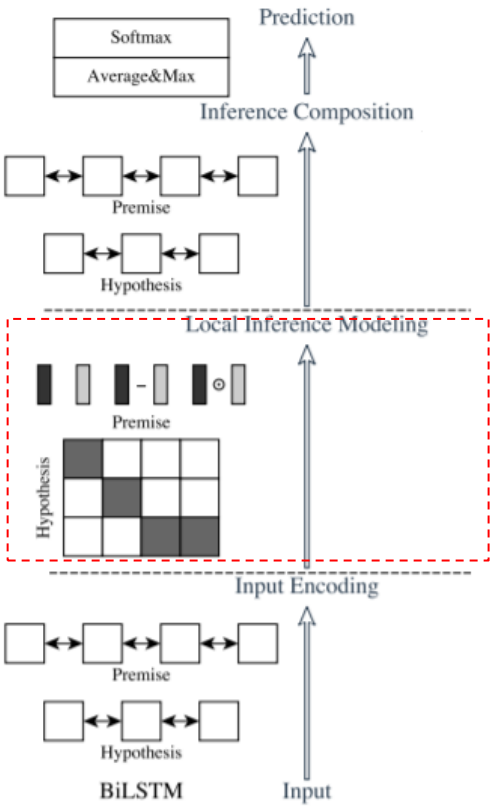
双向的LSTM最后把其隐藏状态的值保留下来，分别记为$\bar{a}i$ 和 $\bar{b}j$

$$\bar{a_i} = BiLSTM(a, i), i \in [1, ..., l_a]$$

$$\bar{b_j} = BiLSTM(b, j), j \in [1, ..., l_b]$$

其中，a与b表示p与h

12.3 交互聚合匹配模型



- **Local Inference Modeling**

把上一轮拿到的特征值做差异性计算

采用attention机制，其中的计算方法：

$$e_{ij} = \bar{a_i}^T \bar{b_j}$$

然后根据attention weight计算出a与b的权重加权后的值

$$\tilde{a_i} = \sum_{j=1}^{l_b} \frac{exp(e_{ij})}{\sum_{k=1}^{l_b} exp(e_{ik})} \bar{b_j}, i \in [1, ..., l_a]$$

$$\tilde{b_j} = \sum_{i=1}^{l_a} \frac{exp(e_{ij})}{\sum_{k=1}^{l_a} exp(e_{kj})} \bar{a_i}, j \in [1, ..., l_b]$$

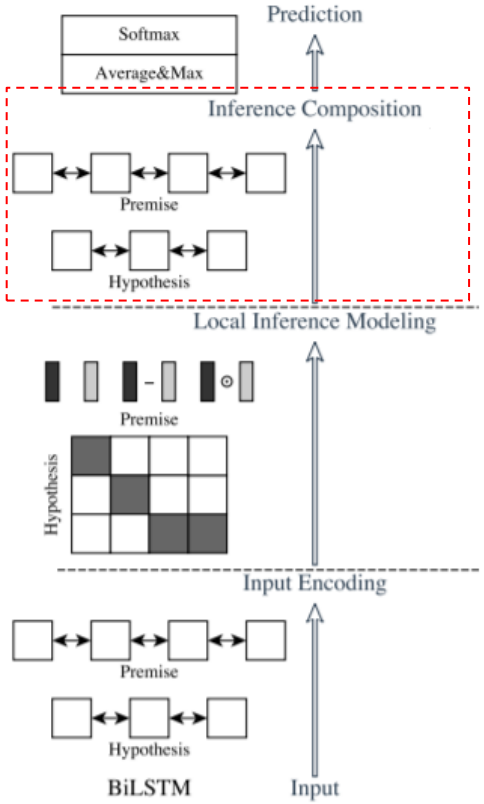
然后分别对这两个值做差异性计算（$\bar{a} - \tilde{a}$，$\bar{a} \odot \tilde{a}$）

最后将各值进行拼接

$$m_a = [\bar{a}; \tilde{a}; \bar{a} - \tilde{a}; \bar{a} \odot \tilde{a}]$$

$$m_b = [\bar{b}; \tilde{b}; \bar{b} - \tilde{b}; \bar{b} \odot \tilde{b}]$$

12.3 交互聚合匹配模型



- **Inference Composition**

把之前的值再一次送到了BiLSTM中，这次主要是用于捕获推理信息

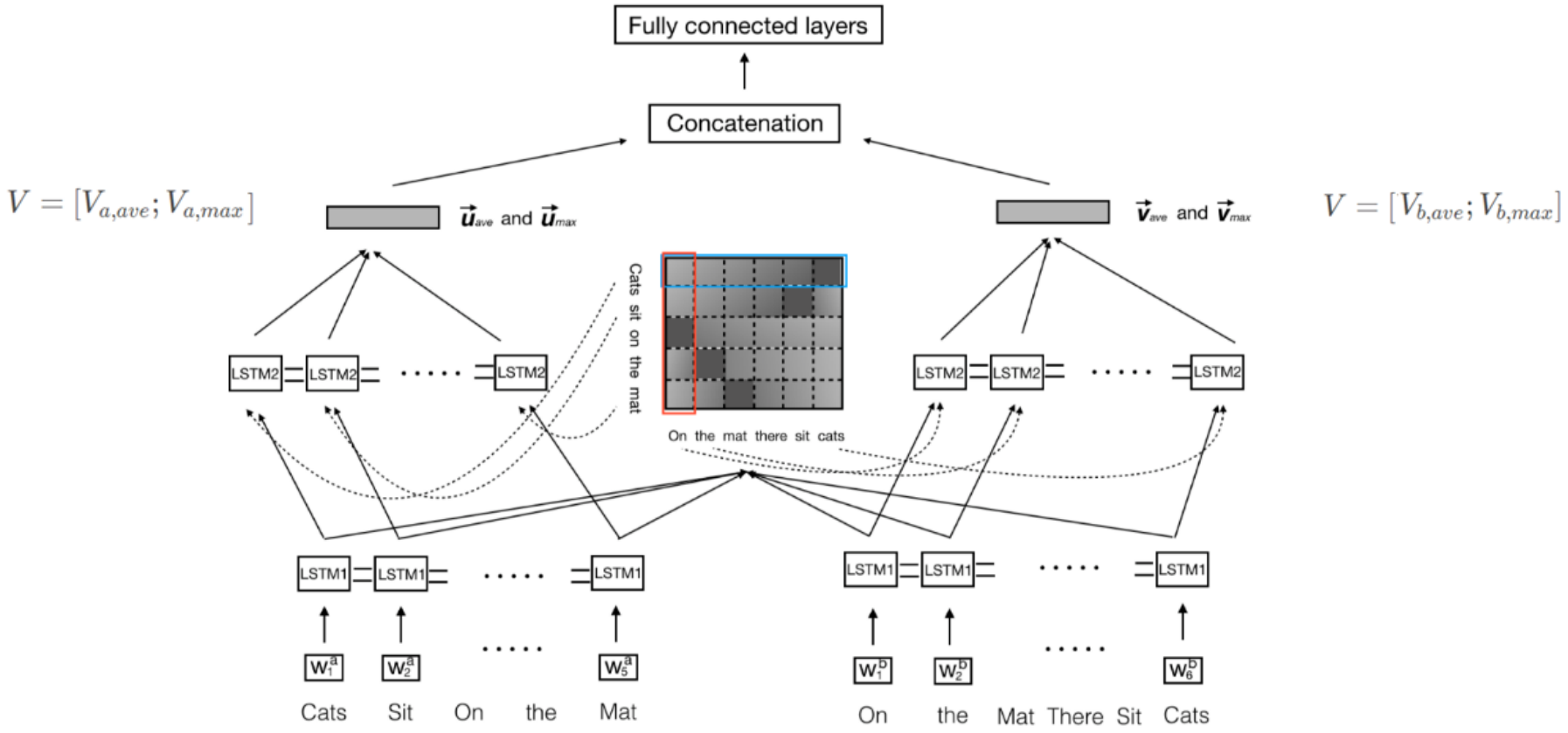
最后把BiLSTM得到的值进行池化操作，分别是最大池化与平均池化，并把池化之后的值再一次的拼接起来。

$$V_{a,ave} = \sum_{i=1}^{l_a} \frac{V_a, i}{l_a}, \quad V_{a,max} = max_{i=1}^{l_a} V_{a,i}$$

$$V_{b,ave} = \sum_{j=1}^{l_b} \frac{V_b, j}{l_b}, \quad V_{b,max} = max_{j=1}^{l_b} V_{b,j}$$

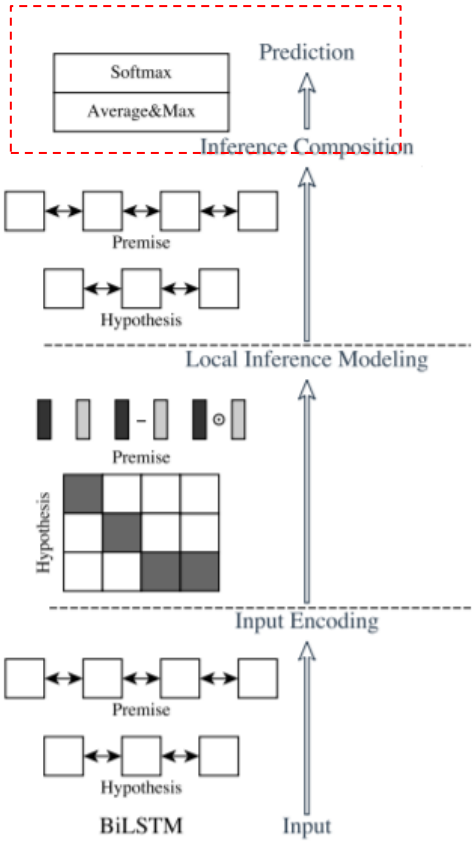
$$V = [V_{a,ave}; V_{a,max}; V_{b,ave}; V_{b,max}]$$

12.3 交互聚合匹配模型



优势：Attention机制更好地建模句子关系

12.3 交互聚合匹配模型



- **Prediction**

最后把V 送入到全连接层，采用tanh激活函数，得到的结果送到softmax层

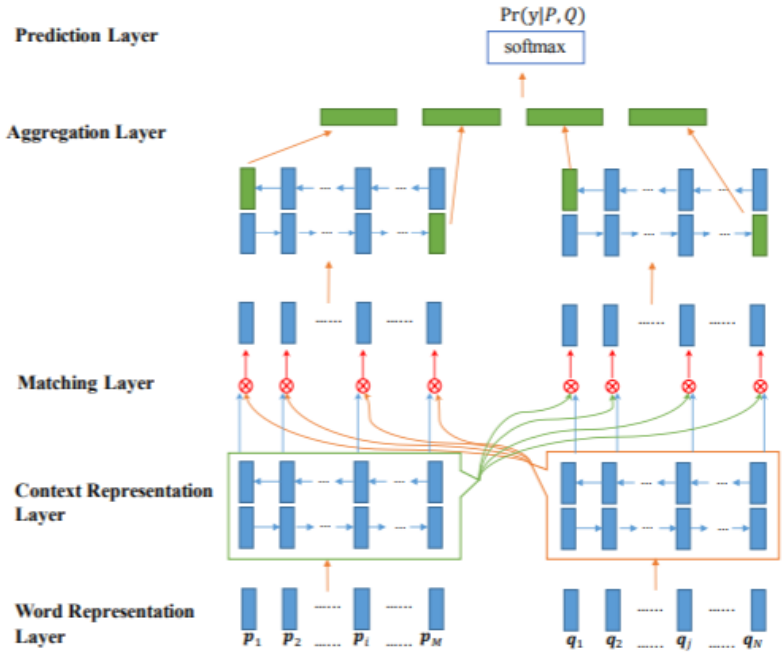
输出：句子X和Y的匹配度

ESIM与BiMPM在相似度匹配任务中都是使用较多的模型，但是ESIM训练速度快

12.3 交互聚合匹配模型

★ BiMPM(基于RNN)

该模型主要用于做文本匹配，即计算文本相似度。创新点在于采用了双向多角度匹配，采用matching-aggregation的结构，把两个句子之间的单元做相似度计算，最后经过全连接层与softamx层得到最终的结果

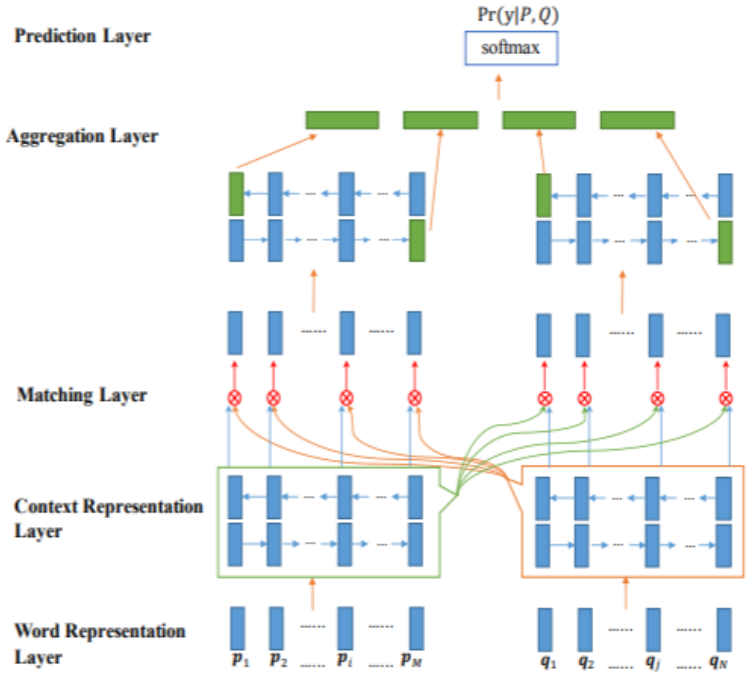


ESIM模型包含五部分：

- Word Representation Layer
- Context Representation Layer
- Matching Layer
- Aggregation Layer
- Prediction Layer

Zhiguo Wang, et al. Bilateral Multi-Perspective Matching for Natural Language Sentences

12.3 交互聚合匹配模型

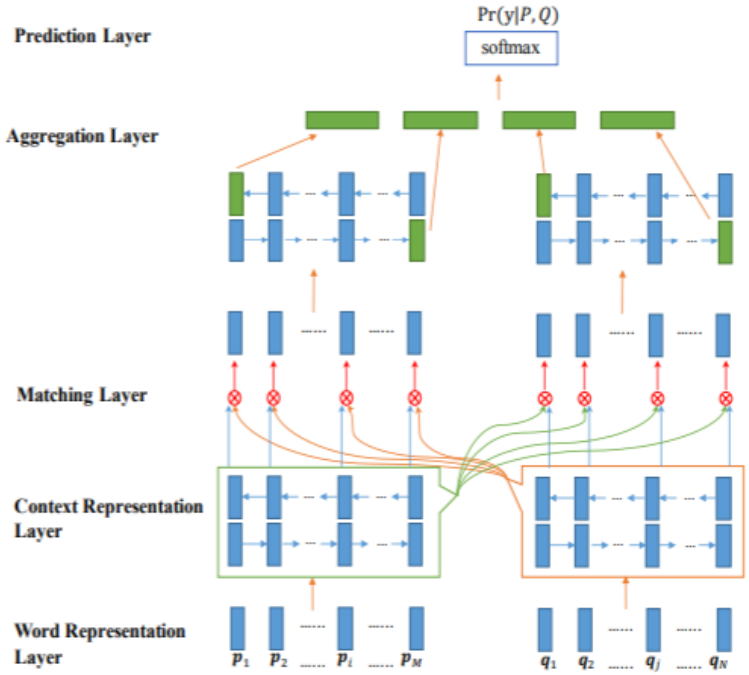


- **Word Representation Layer**

输入：序列 P 和 Q

该层把输入的序列P与Q转变为d维的向量表示，即是文本的向量化表示embedding

12.3 交互聚合匹配模型



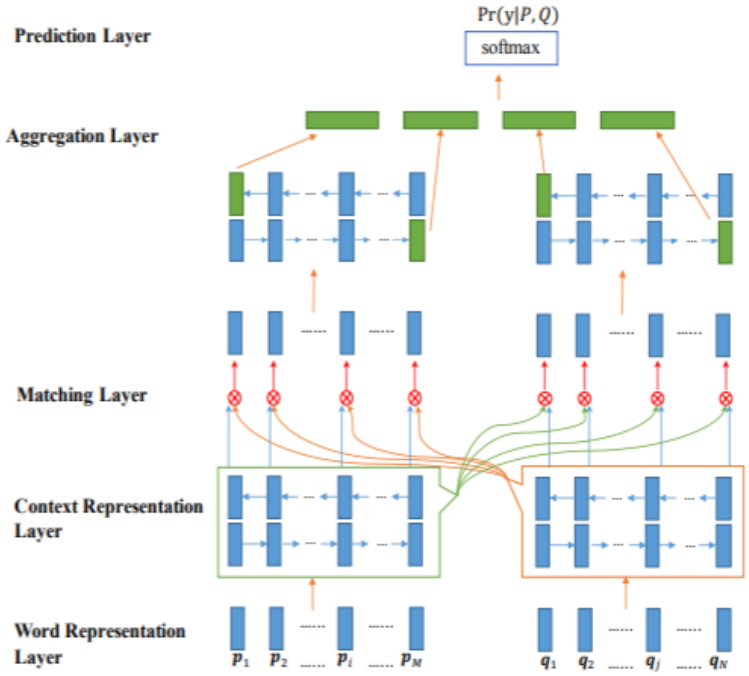
- **Context Representation Layer**

提取P与Q的上下文信息，把上一层拿到的embedding输入到BiLSTM中，分别得到两个方向不同时刻的context embedding

$$\overrightarrow{\boldsymbol{h}}_i^p = \overrightarrow{\text{LSTM}}(\overrightarrow{\boldsymbol{h}}_{i-1}^p, \boldsymbol{p}_i) \qquad i = 1, ..., M$$
$$\overleftarrow{\boldsymbol{h}}_i^p = \overleftarrow{\text{LSTM}}(\overleftarrow{\boldsymbol{h}}_{i+1}^p, \boldsymbol{p}_i) \qquad i = M, ..., 1$$

$$\overrightarrow{\boldsymbol{h}}_j^q = \overrightarrow{\text{LSTM}}(\overrightarrow{\boldsymbol{h}}_{j-1}^q, \boldsymbol{q}_j) \qquad j = 1, ..., N$$
$$\overleftarrow{\boldsymbol{h}}_j^q = \overleftarrow{\text{LSTM}}(\overleftarrow{\boldsymbol{h}}_{j+1}^q, \boldsymbol{q}_j) \qquad j = N, ..., 1$$

12.3 交互聚合匹配模型



- **Matching Layer**

该层是模型的核心层，采用多维度匹配：把每一个序列不同时刻的context与另一个序列的所有时刻的context做一个比较，并且考虑了两个方向，用multi-perspective 的匹配方法，获取两个句子细粒度的联系信息，

12.3 交互聚合匹配模型

多维度匹配

多角度匹配的相似度函数 $m = f_m(v_1, v_2; w)$

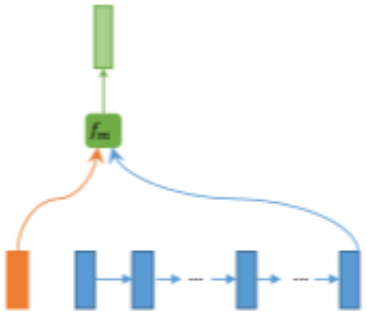
其中v_1与v_2表示的是两个d维度的向量，$W \in R^{l \times d}$是权重，其维度为(l, d)其中l表示的是匹配的角度数量，结果m是一个l维度的向量，$m = [m_1, ..., m_k, ..., m_l]$，每一个$m_k$表示的是第$k$个角度的匹配结果，其值的相似度计算方法如下

$$m_k = cosine(W_k \cdot v_1, W_k \cdot v_2)$$

f_m实际上又有四种策略来求相似度，分别看下

12.3 交互聚合匹配模型

（a）Full-Matching



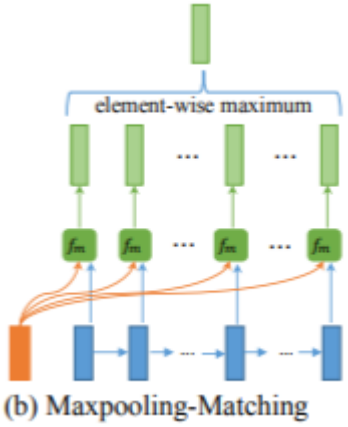
把每一个时刻的context与另一个序列的尾的context做一个相似度计算

$$\overrightarrow{m}_i^{full} = f_m(\overrightarrow{h}_i^p, \overrightarrow{h}_N^q; W^1)$$

$$\overleftarrow{m}_i^{full} = f_m(\overleftarrow{h}_i^p, \overleftarrow{h}_1^q; W^2)$$

$$m_k = cosine(W_k \cdot v_1, W_k \cdot v_2)$$

（b）Maxpooling-Matching



把P序列所有时刻的context与Q序列所有时刻的context做一个相似度计算，最后只保留相似度最大的那个值

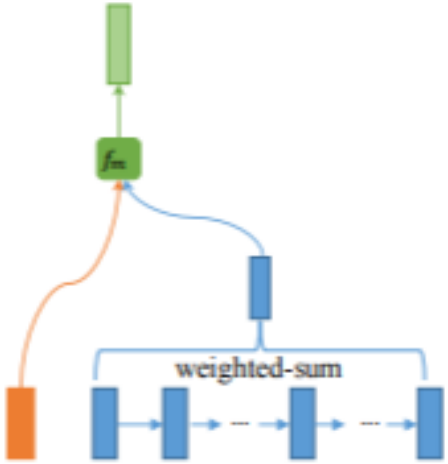
$$\overrightarrow{m}_i^{max} = \max_{j \in (1...N)} f_m(\overrightarrow{h}_i^p, \overrightarrow{h}_j^q; W^3)$$

$$\overleftarrow{m}_i^{max} = \max_{j \in (1...N)} f_m(\overleftarrow{h}_i^p, \overleftarrow{h}_j^q; W^4)$$

where $\max_{j \in (1...N)}$ is element-wise maximum.

12.3 交互聚合匹配模型

(c) Attentive-Matching



(c) Attentive-Matching

先计算P序列每一个时刻的context与Q序列每一个时刻的context的相似度

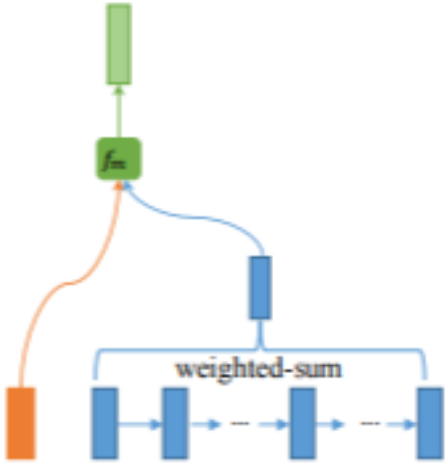
$$\overrightarrow{\alpha}_{i,j} = cosine(\overrightarrow{h}_i^p, \overrightarrow{h}_j^q) \qquad j = 1, ..., N$$
$$\overleftarrow{\alpha}_{i,j} = cosine(\overleftarrow{h}_i^p, \overleftarrow{h}_j^q) \qquad j = 1, ..., N$$

然后把得到的相似度作为对应时刻的权重与另一个序列做加权平均，针对整个序列计算出attentive vector

$$\overrightarrow{h}_i^{mean} = \frac{\sum_{j=1}^{N} \overrightarrow{\alpha}_{i,j} \cdot \overrightarrow{h}_j^q}{\sum_{j=1}^{N} \overrightarrow{\alpha}_{i,j}}$$
$$\overleftarrow{h}_i^{mean} = \frac{\sum_{j=1}^{N} \overleftarrow{\alpha}_{i,j} \cdot \overleftarrow{h}_j^q}{\sum_{j=1}^{N} \overleftarrow{\alpha}_{i,j}}$$

12.3 交互聚合匹配模型

(c) Attentive-Matching



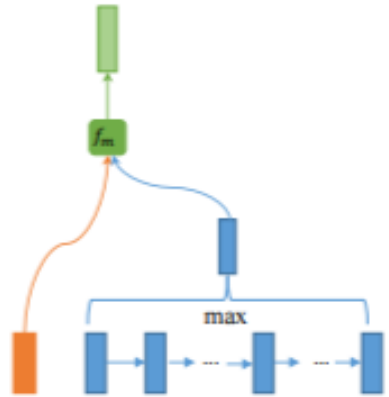
(c) Attentive-Matching

最后把每一个时刻的context与attentive vector做一个相似度计算

$$\overrightarrow{\boldsymbol{m}}_i^{att} = f_m(\overrightarrow{\boldsymbol{h}}_i^p, \overrightarrow{\boldsymbol{h}}_i^{mean}; \boldsymbol{W}^5)$$

$$\overleftarrow{\boldsymbol{m}}_i^{att} = f_m(\overleftarrow{\boldsymbol{h}}_i^p, \overleftarrow{\boldsymbol{h}}_i^{mean}; \boldsymbol{W}^6)$$

（d）Max-Attentive-Matching

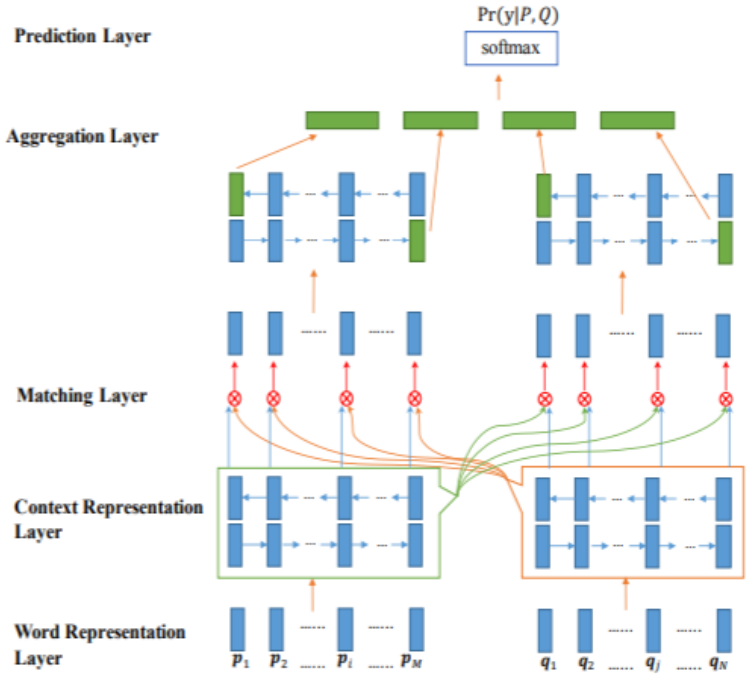


(d) Max-Attentive-Matching

该方法其实和Attentive-Matching方法类似，不过是把最后生成attentive vector的方法改为了求最大值而不是加权平均

序列的每一个时间步长都通过这四种策略得到相似度的值，由于是双向的，最终将生成的8个向量串联起来作为的每个时间步长的匹配向量。

12.3 交互聚合匹配模型



- **Aggregation Layer**

采用的依旧是BiLSTM模型把上一层的结果聚集在一起。

- **Prediction Layer**

两个全连接层与一个softmax层

输出： 序列 P 和 Q 的匹配度

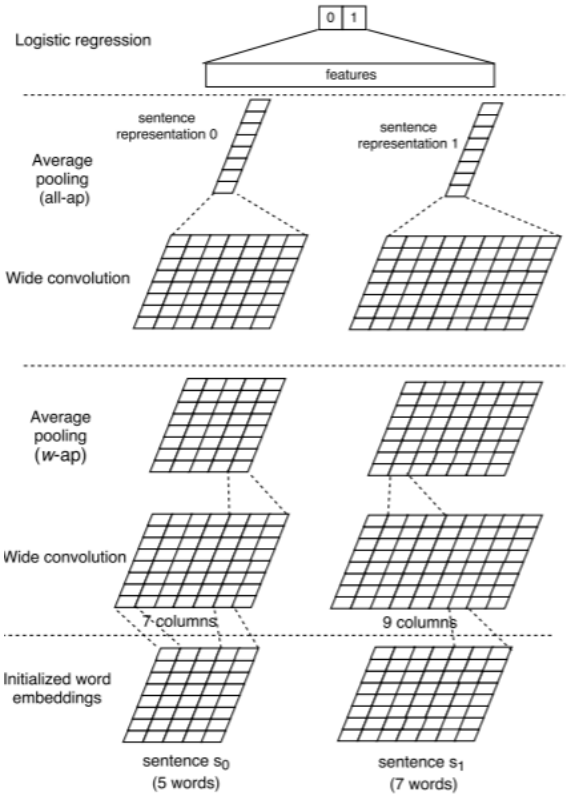
BiMPM特点：1. 同时兼顾了词向量和字向量 2. 更加注重句子之间交互信息，从不同层次不同粒度来匹配待比较的句子，加大了比较范围。效果提升，训练难度大。

12.3 交互聚合匹配模型

★ ABCNN (基于CNN 和 Attention)

论文包括两部分：**1.** 仅有CNN的基础模型BCNN **2.** 添加了attention的ABCNN

1. BCNN: Basic Bi-CNN



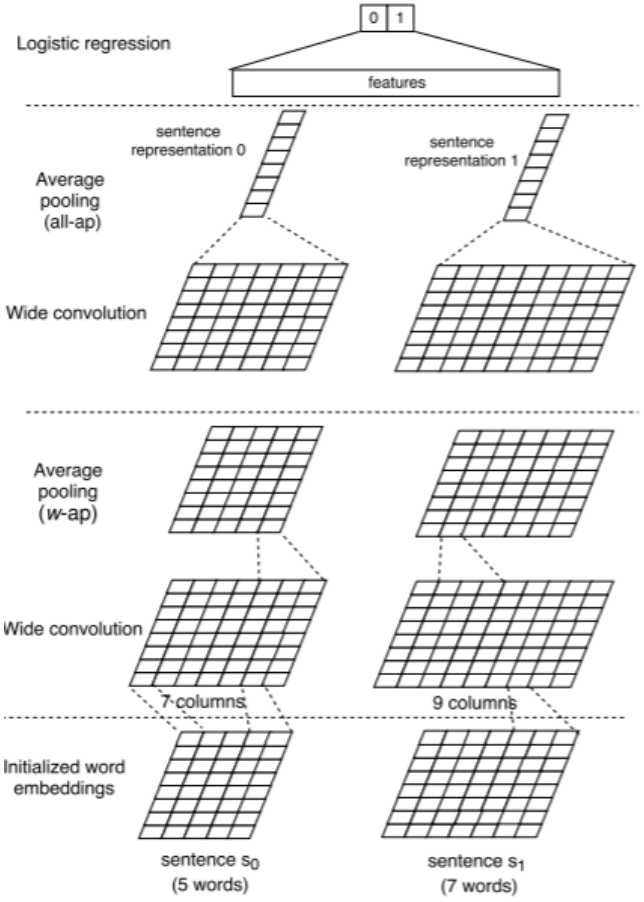
BCNN 模型包含四部分：

- Input Layer
- Convolution Layer
- Average Pooling Layer
- Output Layer

Wenpeng Yin, et al. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs

12.3 交互聚合匹配模型

1. BCNN: Basic Bi-CNN



- **Input Layer**

  该层做词嵌入，可用word2vec，glove，elom等方法

- **Convolution Layer（卷积层）**

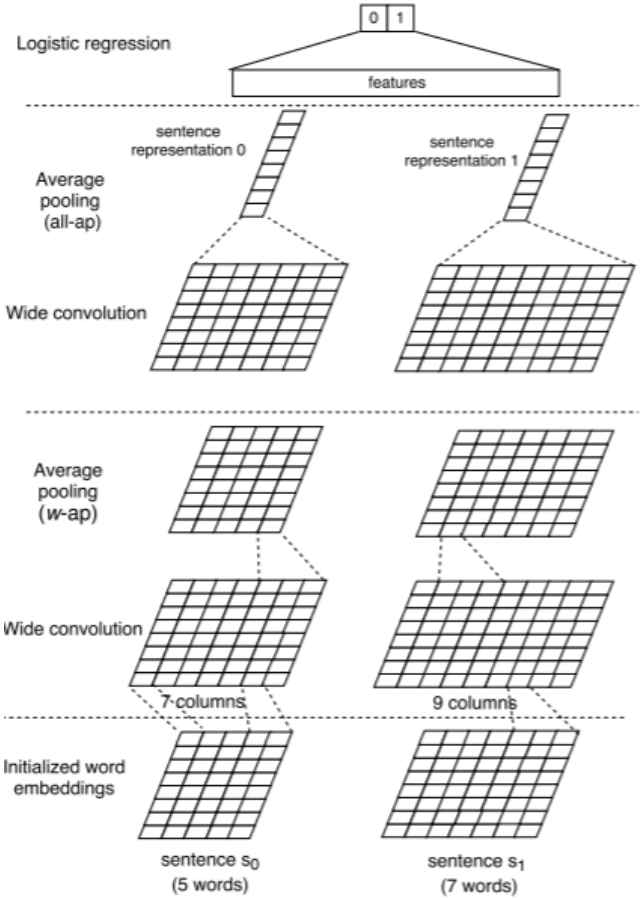
  采用宽卷积得到一个长度为

  $$sent\_length + w_s - 1$$

  的向量，（w为窗口宽度）

12.3 交互聚合匹配模型

1. BCNN: Basic Bi-CNN



- **Average Pooling Layer（平均池化层）**
  - 中间的池化层是w-ap :

    使用滑动窗口的形式，以窗口宽度 w

    对卷积输出进行 Average Pooling

    pooling 层后仍然会变回 $sent\_length$

    可以使 conv-pooling 层就无限叠加
  - 最后层是all-ap（大小与输入相同，保证输出维度相同）
- **Output Layer**

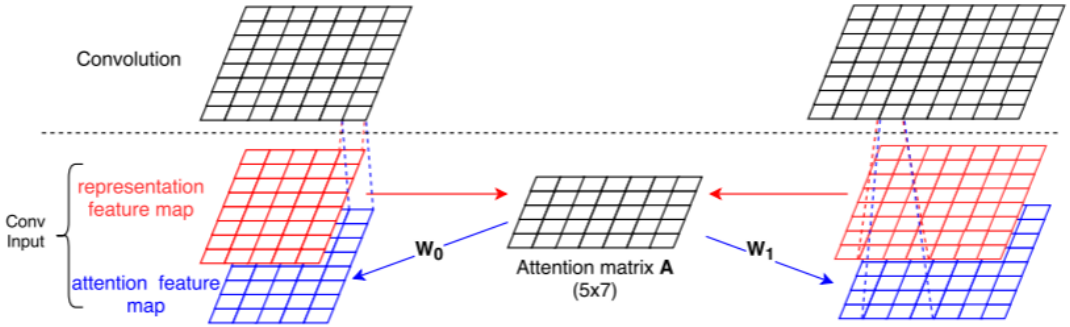
  简单的二分类

12.3 交互聚合匹配模型

2. ABCNN: Attention-Based BCNN

(a) ABCNN-1

通过对输入句子的向量表示进行 attention 操作，从而影响卷积网络



(a) One block in ABCNN-1

- 针对两个不同的序列，生成attention matrix A

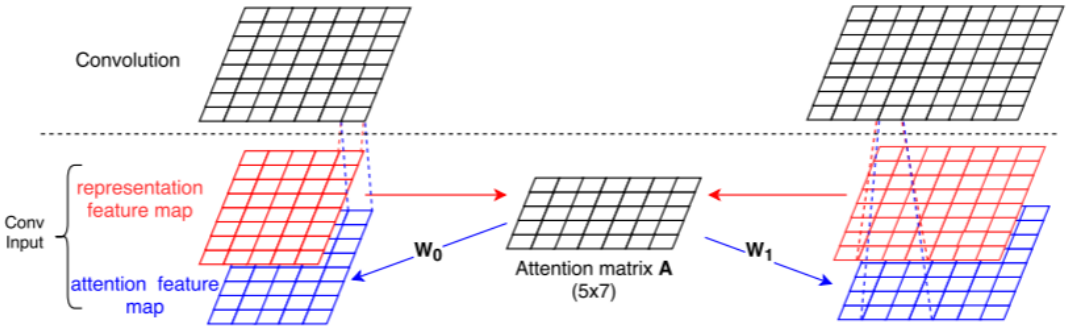
$A_{i,j} = match\_score(F_{0,r}[:,i], F_{1,r}[:,j])$　　$F_{i,r} \in R^{d \times s}$ 表示句子的向量表示

match-score：　$\frac{1}{1+|x-y|}$　（可以用多种方式进行计算）

- 得到了attention矩阵A，则可以计算句子的attention特征

12.3 交互聚合匹配模型

（a） ABCNN-1



(a) One block in ABCNN-1

- 得到了attention矩阵A，则可以计算句子的attention特征

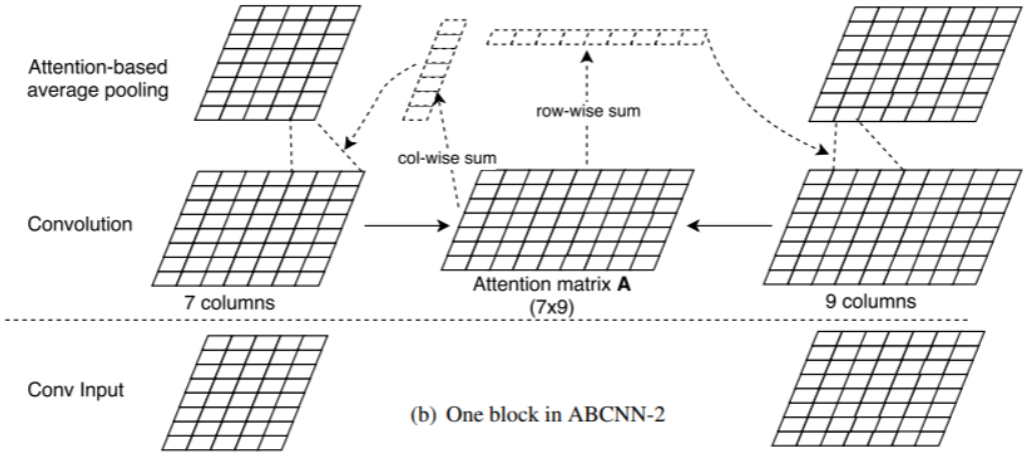
$$F_{0,a} = W_0 \cdot A^T \qquad F_{1,a} = W_1 \cdot A$$

其中：$W_0 \in R^{d \times s}, W_1 \in R^{d \times s}$ 是模型参数

- 将原始的句子向量 $F_{i,r}$ 和attention 特征向量 $F_{i,a}$ 进行叠加作为卷积层的输入向量

（b） ABCNN-2

对 conv 层的输出进行 attention，从而对卷积层的输出结果进行加权

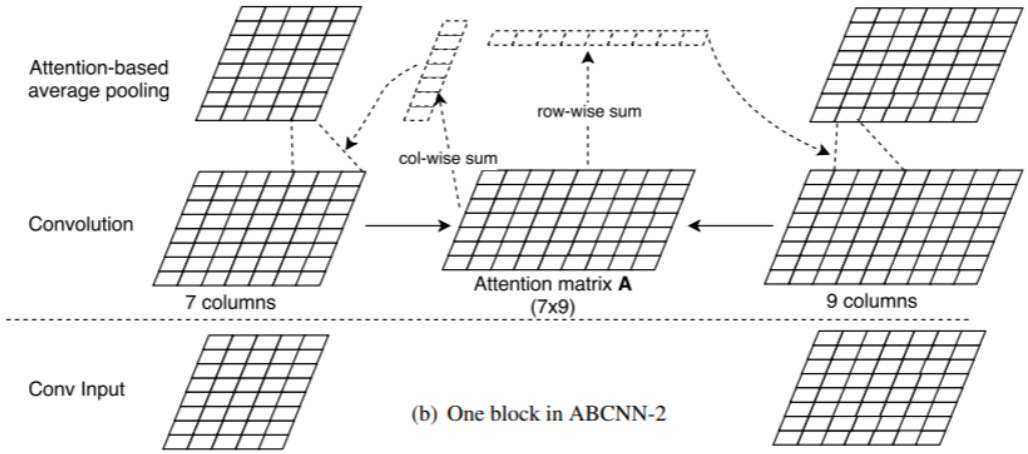


- 针对卷积后两个不同的序列，生成attention matrix A （方法同ABCNN-1）

  分别为两个句子计算它们的 conv 输出和 attention 矩阵 Average Pooling

12.3 交互聚合匹配模型

（b） ABCNN-2



- pooling： 根据计算出的 Attention 权重向量来计算得到

$$a_{0,j} = \Sigma A[j,:] \qquad a_{1,j} = \Sigma A[:,j]$$

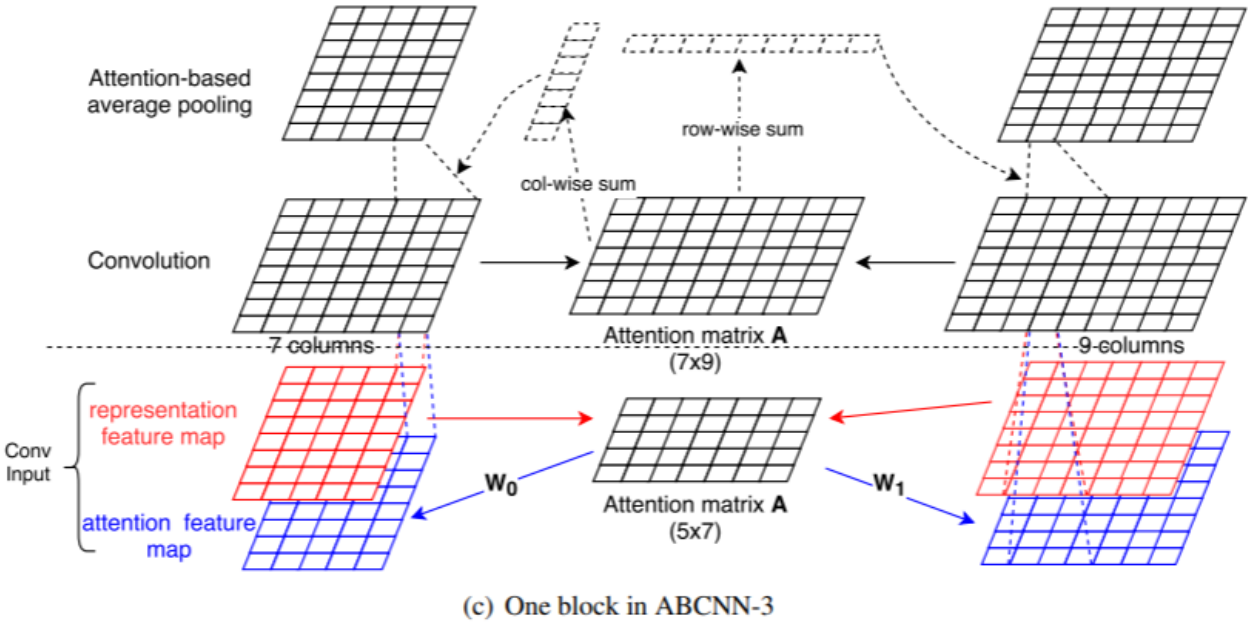
利用这个 attention 值对卷积层的输出进行加权

$$F_{i,r}^{p}[:,j] = \Sigma_{k=j:j+w} a_{i,k} \cdot F_{i,r}^{c}[:,k], j = 1...s_i$$

conv 层的输出即是 pooling 层的输入最后池化后的结果维度和输入维度是相同的

12.3 交互聚合匹配模型

(c) ABCNN-3



(c) One block in ABCNN-3

- ABCNN-3 是 ABCNN-1 和 ABCNN-2结构进行叠加

12.3 交互聚合匹配模型

实验结果：

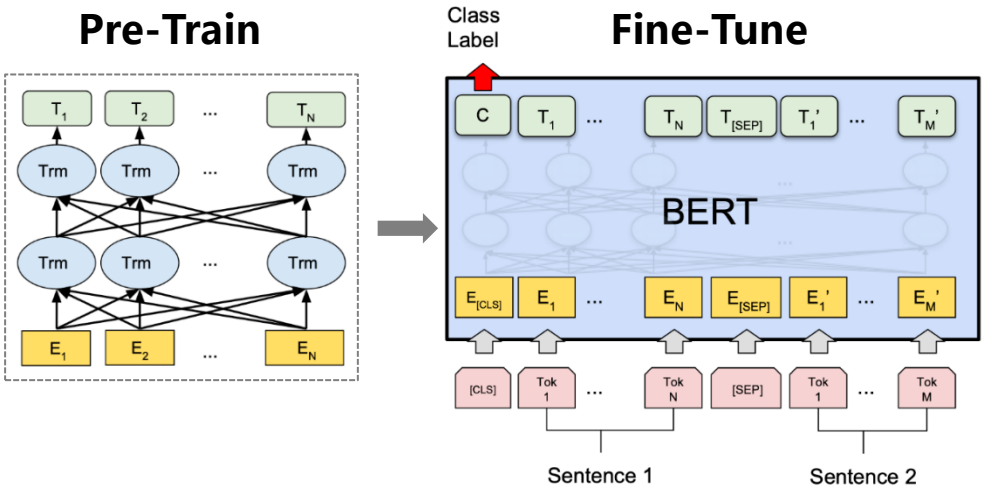
	method	MAP	MRR
Baselines	WordCnt	0.4891	0.4924
	WgtWordCnt	0.5099	0.5132
	CNN-Cnt	<u>0.6520</u>	<u>0.6652</u>
	Addition	0.5021	0.5069
	Addition(+)	0.5888	0.5929
	A-LSTM	0.5347	0.5483
	A-LSTM(+)	0.6381	0.6537
BCNN	one-conv	0.6629	0.6813
	two-conv	0.6593	0.6738
ABCNN-1	one-conv	0.6810*	0.6979*
	two-conv	0.6855*	0.7023*
ABCNN-2	one-conv	0.6885*	0.7054*
	two-conv	0.6879*	0.7068*
ABCNN-3	one-conv	0.6914*	**0.7127***
	two-conv	**0.6921***	0.7108*

Table 3: Results on WikiQA. Best result per column is bold. Significant improvements over state-of-the-art baselines (underlined) are marked with $*$ (t-test, $p < .05$).

12.3 交互聚合匹配模型

★ BERT (基于Transformer)

■ 模型结构



输入：句子X和Y

运算关系：

1. 预训练双向Transformer语言模型BERT
2. 取BERT最后一层[CLS]的表示作为句子关系的表示
3. 输入分类器

输出：句子X和Y的匹配度

优势：更深层的语义表达更好的通用语义表示

Devlin, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL (2018).

12.3 交互聚合匹配模型

★ ConSeqNet

■ 动机：

先前的文本蕴含任务模型都是利用给定的文本对问题进行分析，没有利用外部资源

该文提出一种利用外部知识增强任务

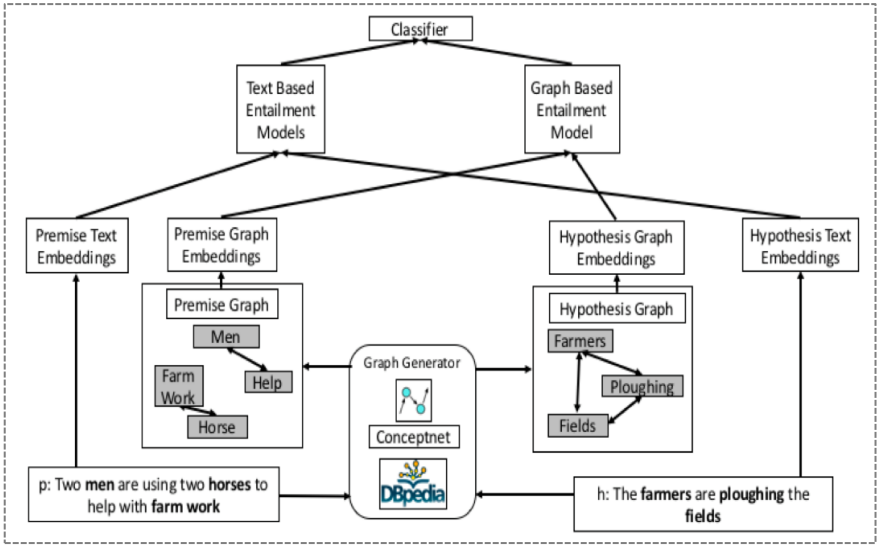
文本蕴含任务(text entailment)：给定一个前提文本（premise），根据这个前提去推断假说文本（hypothesis）与premise的关系，一般分为蕴含关系（entailment）和矛盾关系（contradiction）中立关系（neutral）；蕴含关系：表示从premise中可以推断出hypothesis；矛盾关系：hypothesis与premise矛盾；中立关系：hypothesis与premise无关。文本蕴含的结果就是这几个概率值。

Wang, Xiaoyan, Improving Natural Language Inference Using External Knowledge in the Science QuestionsDomain.” AAAI (2018).

12.3 交互聚合匹配模型

★ ConSeqNet （以图的形式利用外部知识）

■ **模型架构：**



模型分二个独立通道：

1. 文本通道

输入句子p 和 h经过Text Based Model 得到融合文本匹配向量，输入分类器

2. 图形通道

输入句子p 和 h经过Text Graph Model 得到的图信息向量，输入分类器

输出：由文本通道生成向量和图形通道生成向量联合产生输出

优势：引入外部的知识

Wang, Xiaoyan, Improving Natural Language Inference Using External Knowledge in the Science QuestionsDomain.” AAAI (2018).

12.3 交互聚合匹配模型

1. 文本通道 Text Based Model

- 用 Gmatch-LSTM 表示文本向量

match-LSTM：

- 用双向LSTM 产生文本编码
- Word-by-Word Attention

$E_{ij} = \mathbf{p}_i \cdot \mathbf{h}_j$ for the hypothesis:

$$\alpha_j = \sum_{i=1}^{K} \frac{\exp(E_{ij})}{\sum_{k=1}^{K} \exp(E_{kj})} \mathbf{p}_i$$

- Matcher

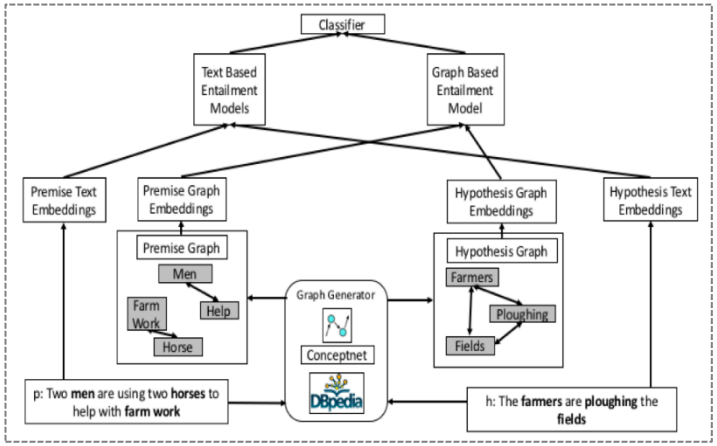
$$\tilde{\mathbf{h}}_j = [\mathbf{h}_j; \alpha_j; \mathbf{h}_j - \alpha_j; \mathbf{h}_j \odot \alpha_j]$$

$$\{\mathbf{h}_j^{\mathrm{m}}\}_{j=1:J} = \mathrm{BiLSTM}(\{\tilde{\mathbf{h}}_j\}_{j=1:J})$$

- Pooling

$$\mathbf{x}_{\mathrm{out}}^{\mathrm{text}} = \max([\mathbf{h}_1^{\mathrm{m}}, \mathbf{h}_2^{\mathrm{m}}, \ldots, \mathbf{h}_J^{\mathrm{m}}])$$

$\mathbf{x}_{\mathrm{out}}^{\mathrm{text}}$ is then used for classification.



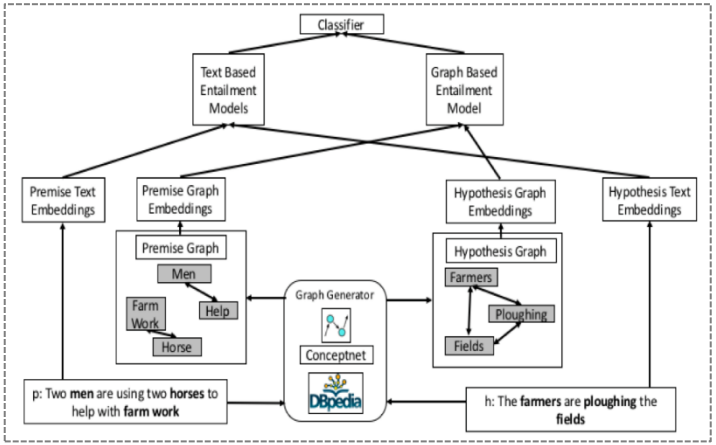
12.3 交互聚合匹配模型

2. 图形通道　Graph Based Model

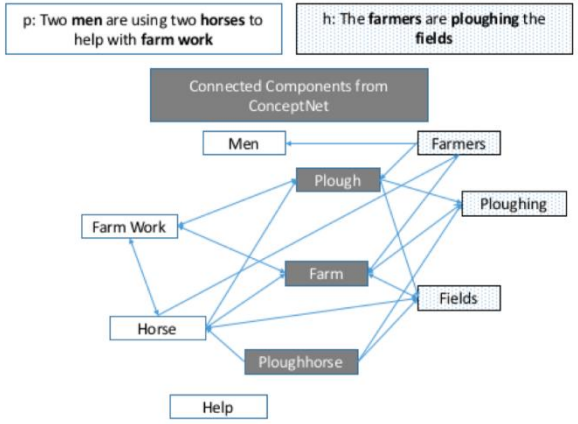
- 利用知识库 conceptnet 产生 p 和 h 的图拓扑结构

方法：句子中的实体或短语在知识库中出现，抽取出来作为图结点

利用知识库 conceptnet可构建“Concepts only Graph”，“One-Hop”，“Two-Hop”三种知识图



$G_p = (V_p, E_p)$　　$G_h = (V_h, E_h)$

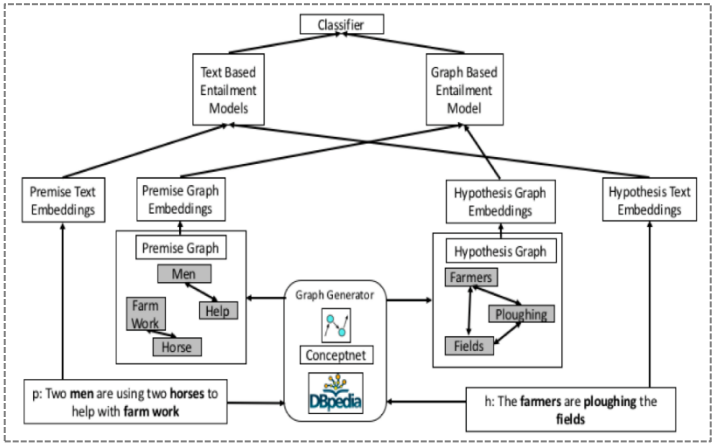


12.3 交互聚合匹配模型

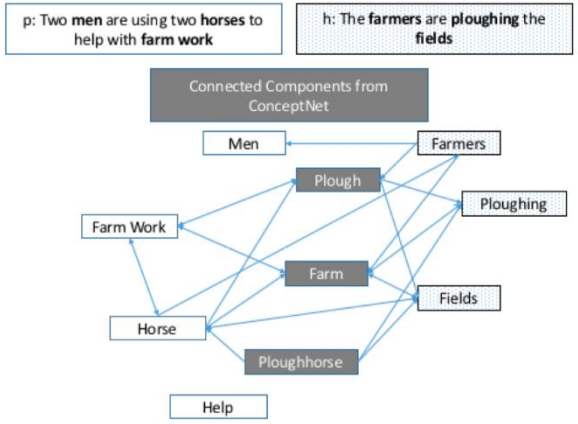
2. 图形通道　Graph Based Model

- Gmatch-LSTM for Concepts Only Graphs

将图数据按照文本数据处理：将图中结点按照在句子中出现的顺序展开结点中的concept 用知识图嵌入表示，如 TransH 。利用Gmatch-LSTM 可将图表示为向量



$G_p = (V_p, E_p)$　　$G_h = (V_h, E_h)$



2. 图形通道 Graph Based Model

- GconAttn for General Graphs (Graph Concepts Attention Model)

1. Context Encoding :GconAttn model directly starts with the concept embeddings.

2. Word-by-Word Attention

$$\beta_i = \sum_{j=1}^{J} \frac{\exp(E_{ij})}{\sum_{k=1}^{J} \exp(E_{ik})} \mathbf{h}_j$$

$$\alpha_j = \sum_{i=1}^{K} \frac{\exp(E_{ij})}{\sum_{k=1}^{K} \exp(E_{kj})} \mathbf{p}_i$$

其中： $E_{ij} = \mathbf{p}_i \cdot \mathbf{h}_j$

3. Matcher

$$\mathbf{p}_i^{\mathbf{m}} = F\left([\mathbf{p}_i; \beta_i; \mathbf{p}_i - \beta_i; \mathbf{p}_i \odot \beta_i]\right)$$

$$\mathbf{h}_j^{\mathbf{m}} = F\left([\mathbf{h}_j; \alpha_j; \mathbf{h}_j - \alpha_j; \mathbf{h}_j \odot \alpha_j]\right)$$

其中： $\mathbf{s} \in \{\mathbf{h}, \mathbf{p}\}$

4. Pooling

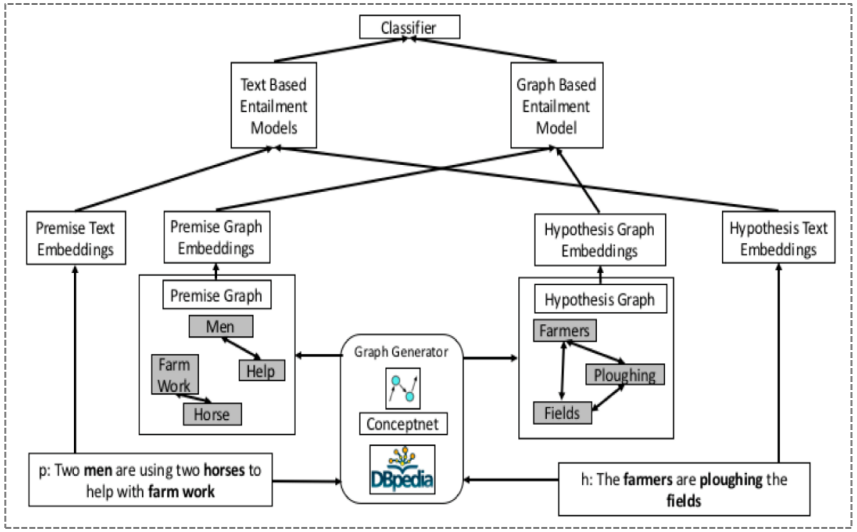
$$\mathbf{s}'_{\max} = \max\left([\mathbf{s}_1^{\mathbf{m}}, \mathbf{s}_2^{\mathbf{m}}, \ldots, \mathbf{s}_N^{\mathbf{m}}]\right)$$

$$\mathbf{s}'_{\text{avg}} = \text{avg}\left([\mathbf{s}_1^{\mathbf{m}}, \mathbf{s}_2^{\mathbf{m}}, \ldots, \mathbf{s}_N^{\mathbf{m}}]\right)$$

$$\mathbf{x}_{\text{out}}^{\text{graph}} = [\mathbf{p}'_{\max}; \mathbf{p}'_{\text{avg}}; \mathbf{h}'_{\max}; \mathbf{h}'_{\text{avg}}]$$

12.3 交互聚合匹配模型

3. Merging Text and Graph Models



将Text Based Model 产生的 $\mathbf{x}_{\text{out}}^{\text{text}}$

与 Graph Based Model产生的 $\mathbf{x}_{\text{out}}^{\text{graph}}$

连接进行分类输出

$$\mathbf{pred} = F\left(\left[\mathbf{x}_{\text{out}}^{\text{text}}; \mathbf{x}_{\text{out}}^{\text{graph}}\right]\right)$$

12.3 交互聚合匹配模型

- **实验结果：**

Model	Dev	Test
Decomp-Attn (Parikh et al. 2016)	75.4	72.3
DGEM* (Khot, Sabharwal, and Clark 2018)	79.6	77.3
DeIsTe (Yin, Roth, and Schütze 2018)	82.4	82.1
BiLSTM-Maxout (Mihaylov et al. 2018)	-	84.0
match-LSTM (Wang and Jiang 2015)	88.2	84.1
Our implementation		
match-LSTM (GRU)	88.5	84.2
match-LSTM+WordNet* (Chen et al. 2018)	88.8	84.3
match-LSTM+Gmatch-LSTM* (**ConSeqNet**)	**89.6**	**85.2**

Table 1: Performance of entailment models on SciTail in comparison to our best model that uses match-LSTM as the text and the graph model with *Concepts Only* graph and CN-PPMI embeddings. * indicates the use of external knowledge in the approach.

推荐工具：

MatchZoo 是一个通用的文本匹配工具包，旨在方便大家快速的实现、比较、以及分享最新的深度文本匹配模型。

Keras Version: https://github.com/NTMC-Community/MatchZoo

PyTorch Version: https://github.com/NTMC-Community/MatchZoo-py

Transformer相关模型的工具包

All Version: https://github.com/huggingface/transformers

参考文献：

- Neural Network Models for Paraphrase Identification, Semantic Textual Similarity, Natural Language Inference, and Question Answering. COLING 2018.
- 庞亮,兰艳艳,徐君,郭嘉丰,万圣贤,程学旗. 深度文本匹配综述. 计算机学报. 2016, Vol.39, No.128.
- Huang P S, He X, Gao J, et al. Learning deep structured semantic models for web search using clickthrough data. CIKM, 2013
- Hu, Baotian, Zhengdong Lu, Hang Li and Qingcai Chen. "Convolutional Neural Network Architectures for Matching Natural Language Sentences." NIPS, 2014.
- Lu, Zhengdong and Hang Li. "A Deep Architecture for Matching Short Texts." NIPS (2013).
- Wang, Mingxuan, Zhengdong Lu, Hang Li and Qun Liu. "Syntax-based Deep Matching of Short Texts." Arxiv (2015).
- Pang, Liang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan and Xueqi Cheng. "Text Matching as Image Recognition." AAAI (2016).
- Alexis Conneau, Douwe Kiela, Holger Schwenk, Loıc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. EMNLP 2018.
- Yixin Nie and Mohit Bansal. Shortcut-stacked sentence encoders for multi-domain inference. In Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP. Arxiv, 2017.
- Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced LSTM for natural language inference. ACL 2017.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT, 2018.
- Xiaodong Liu, Pengcheng He , Weizhu Chen, Jianfeng Gao. "Multi-Task Deep Neural Networks for Natural Language Understanding." Arxiv (2019).
- https://blog.csdn.net/weixin_38526306/article/details/88425997

在此表示感谢!

谢谢各位！

Q&A