# 文本匹配算法

## 套路知几许：

### 1. 1-d卷积

分为宽卷积和窄卷积两种

宽卷积的结果是：卷积完，宽度不变为embedding size，但是长变了

目标：

（1）为了让每个词或者字都进行同样卷积的次数（更加平等对待边缘词），不像窄卷积，最中间的词可能被卷积了k次，而边缘词只卷积了1次

（2）卷积神经网络为了达到深度的目标，可以采用1-d宽卷积+移动池化，这样经过一轮的卷积池化之后，图片/文本 的size依然还是和原来一样。这样就可以无限堆叠cnn+pooling

窄卷积

窄卷积卷积完成之后，会造成宽度依然是embedding size，但是长度变小。但是变小了之后依然还是可以继续1-d卷积的，只是越卷积长越小，意味着1-d卷积的层是有限的。

### 2. 输入层

虽然dssm及其改进算法是deep learning做qq匹配的鼻祖，但是由于dssm出生的时候word2vec还没有出生。导致其输入是one-hot的优化算法。现在都用word2vec了

### 文本表示层

Cnn、rnn、lstm、gru、attention、match paramid

### 4. 文本交互

（1）一些算法直到拼接特征进行分类前再将其拼接，之前特征之间无交叉

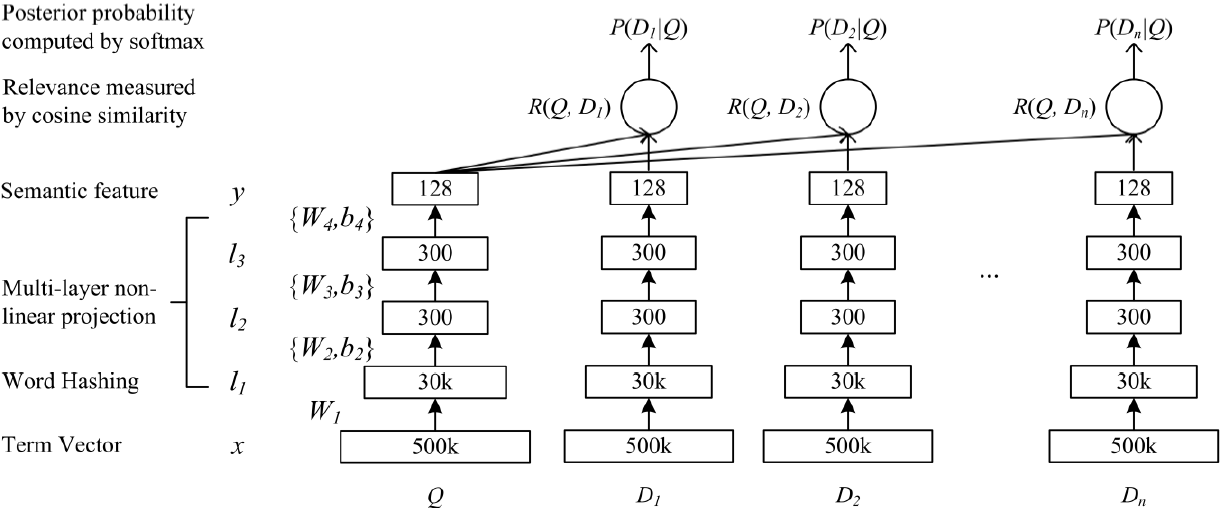
譬如：

Arci

Dssm

（2）也可以在前面就进行文本之间的交互

## DSSM算法



输入跟 CBOW一样。

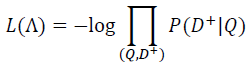
将字做one-hot表示。第一层将一句话中出现的词所对应的one-hot表示进行加和得到一个向量表示。

然后全连接-激活-全连接-激活最终得到一句话的128维的向量表示。激活函数采用tanh。

输出层是Q和D+计算cosine相似度，和D-计算cosine相似度，负采样算法采样出4个负样本（论文里的超参数，可调）。

这样可以计算出一个四维的向量表示。

计算softmax。

损失函数为：

通过反向传播即可得到参数。

CNN-DSSM

相对与DSSM而言：

1. 输入层做了变化。 词向量表示
2. 表示层发生了变化。 卷积神经网络

相对于textcnn，cnn-dssm的卷积核仅用了3维，意味着三个词三个词的卷积，作者用的是字向量。

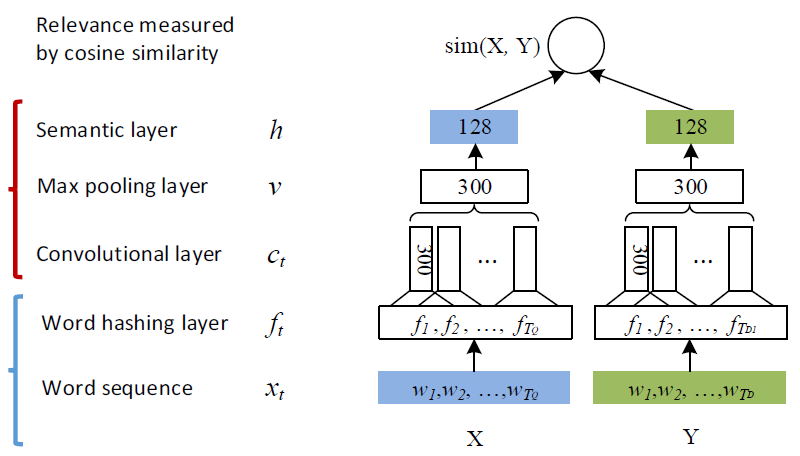
输出层和dssm一致。

优点：

相对于bow而言，用了更多了上下文信息。但是相对于rnn而言，这个信息还不充分。

同时，相对于transformer而言深度也不够。

如果句子太长，可用transformer-xl



## LSTM-DSSM

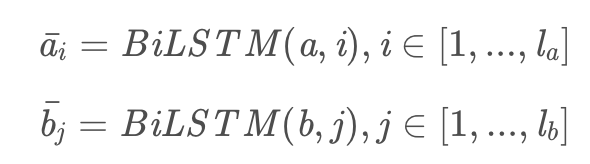
同cnn-dssm一致。不过是将cnn改成了lstm

## ESIM

这是一个比较复杂的算法。

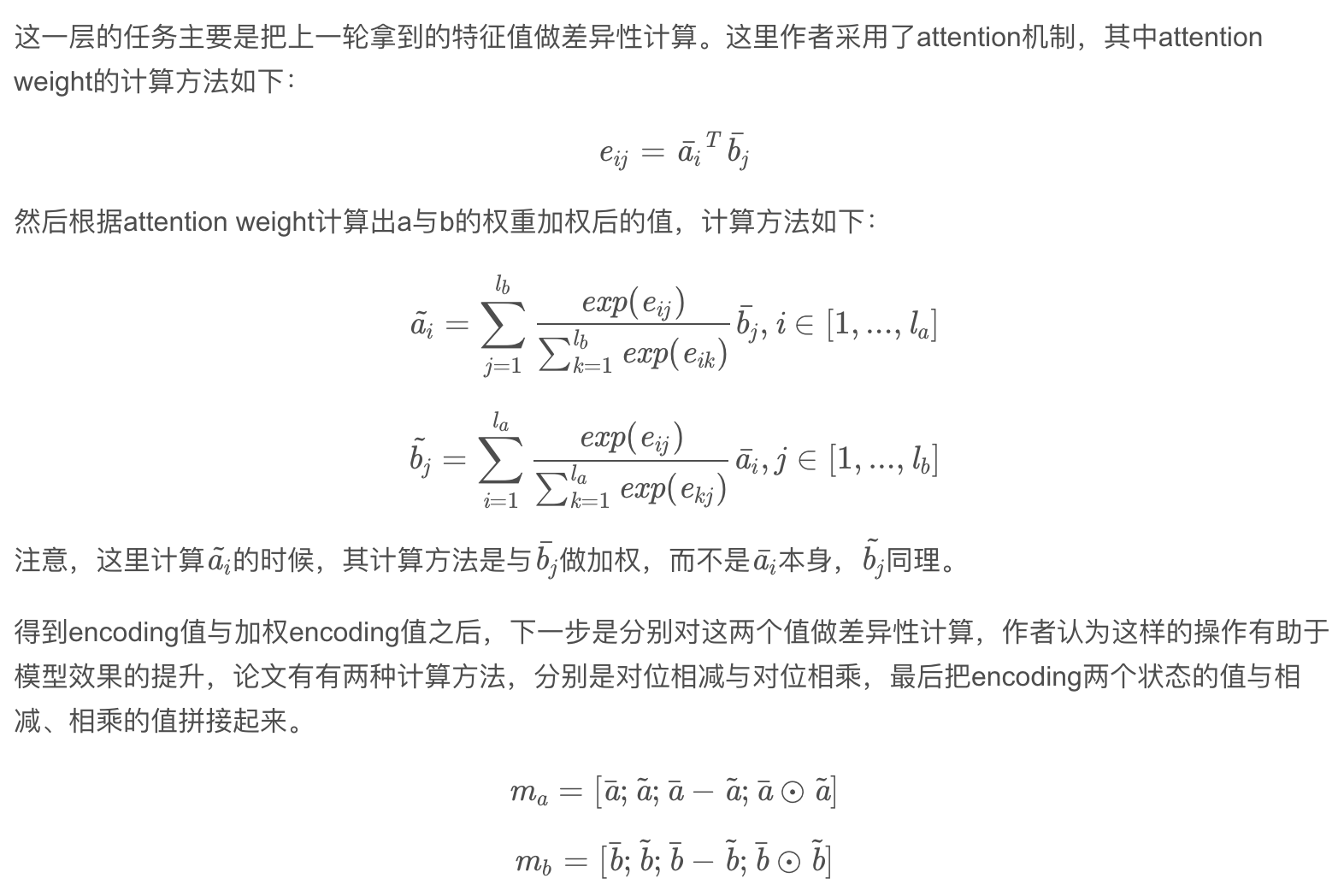
1. Input Encoding

词向量作为输入，接入bilstm

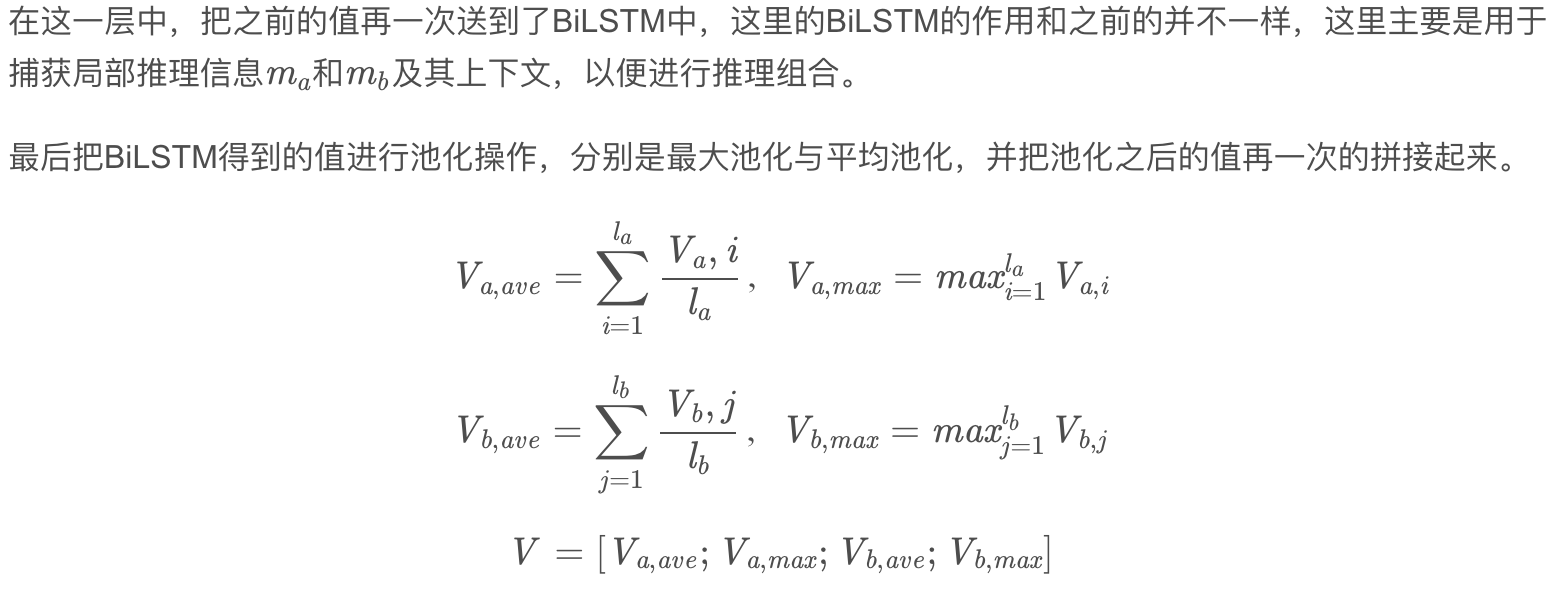


1. Local Inference modeling

接入attention，然后比较attention之前和之后的差异



1. Inference Composition

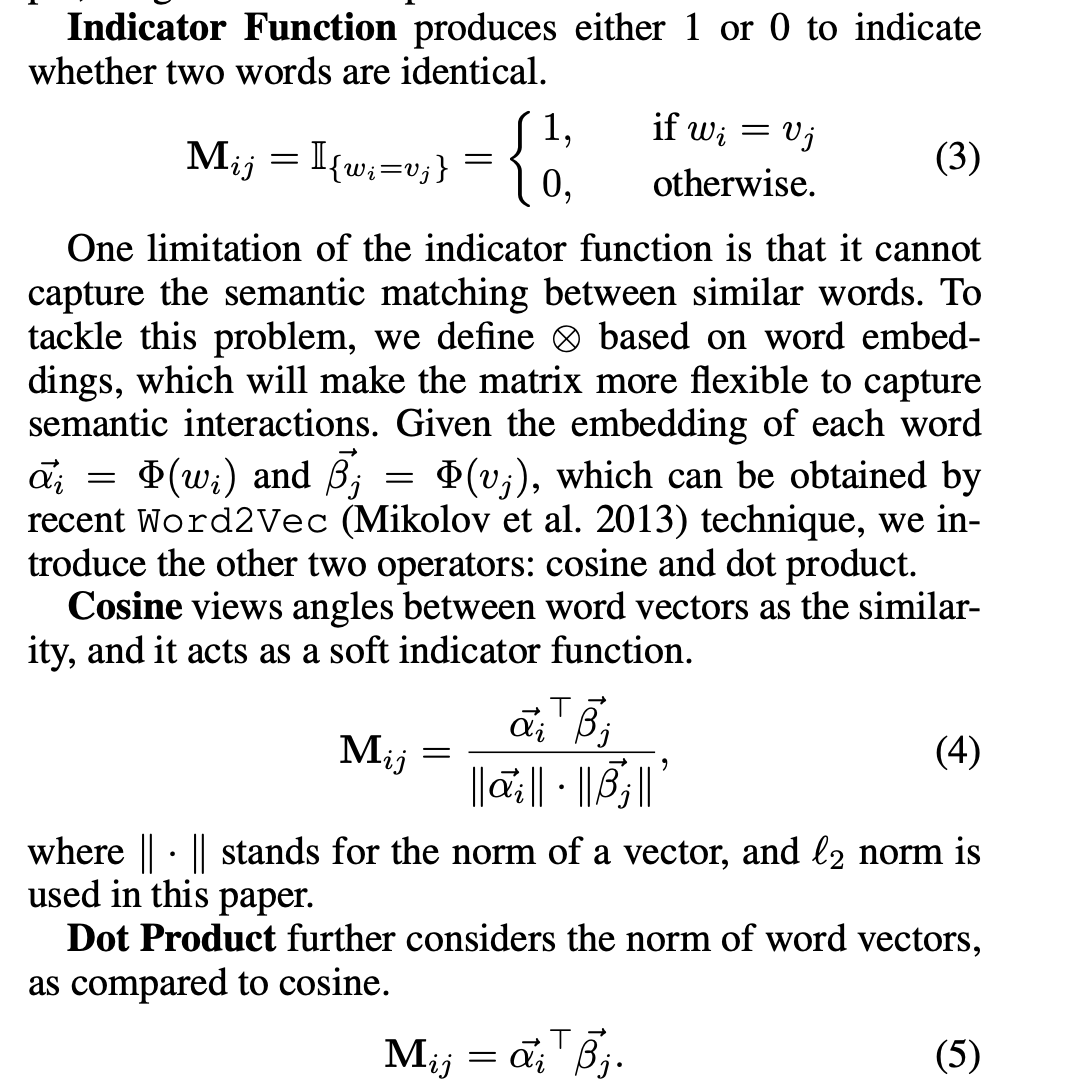


1. Prediction

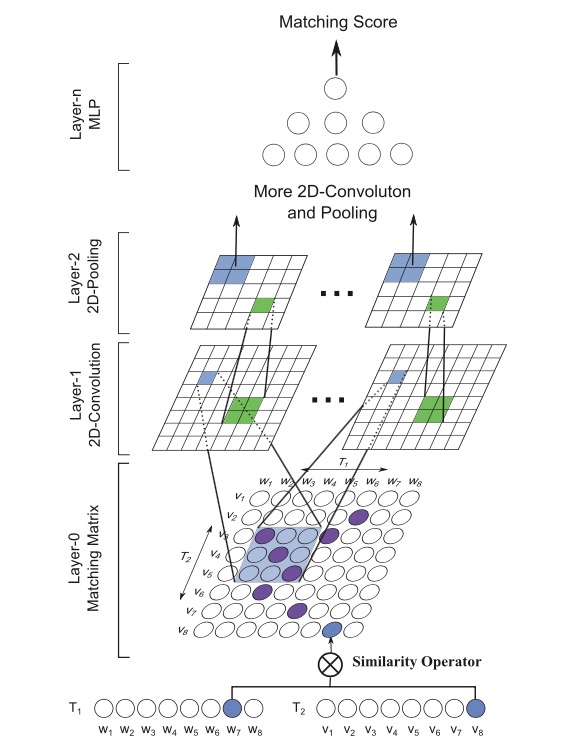
最后把VVV送入到全连接层，激活函数采用的是tanhtanhtanh，得到的结果送到softmax层。

## Match pyramid

相似度计算可以选：



2. 转换成图片之后，接着2d卷积、池化、mlp、logistic之后即可。



## ARCI

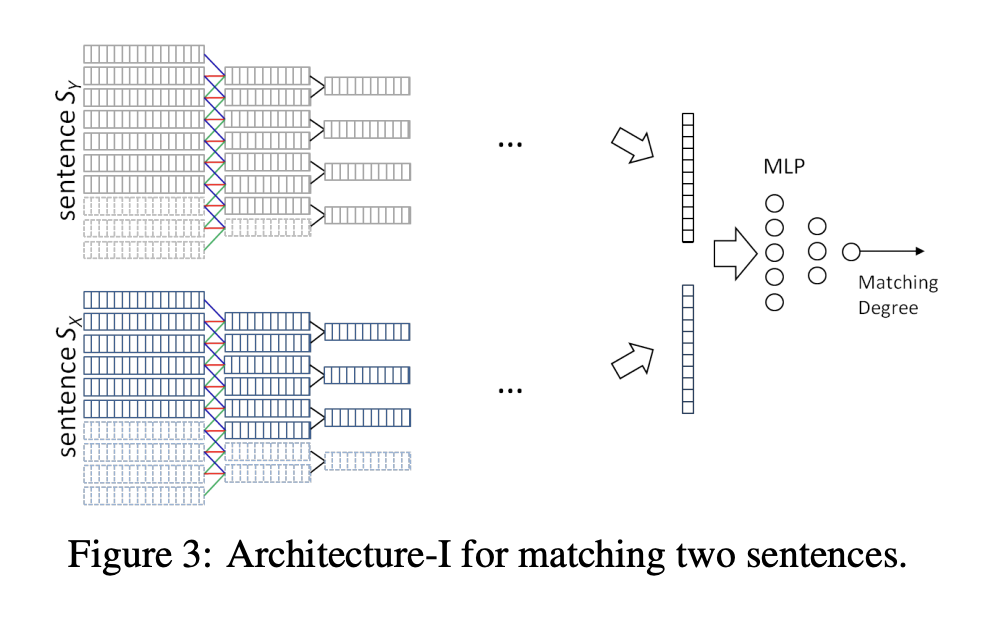
1. 输入是句子的向量表示

2. 进行1-d卷积，没有进行宽卷积

3. 并不是进行卷积，是做了pooling

4. 中间可以继续1-d卷积和pooling

5. 也可以直接pooling成一列向量接mlp，然后接logistic回归



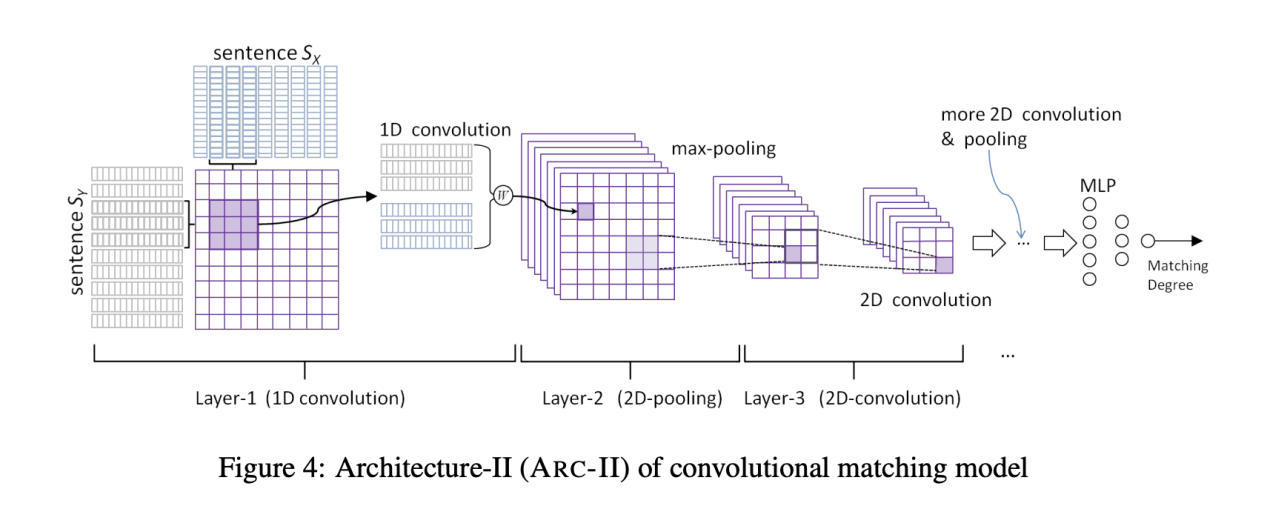
## ARCII

1. 两个句子分别做1-d卷积，窄卷积不是宽卷积，下面最左边的图有一定误解性，1-d卷积进行在长的维度上进行卷积，宽的维度为1

2. 对卷积完的图，做2-d poolding

3. 进行2-d卷积，卷积核的size并不是宽度等于embedding，而是可以将卷积核的size定作为超参数

4. 最后接最大池化，接mlp即可



## BCNN

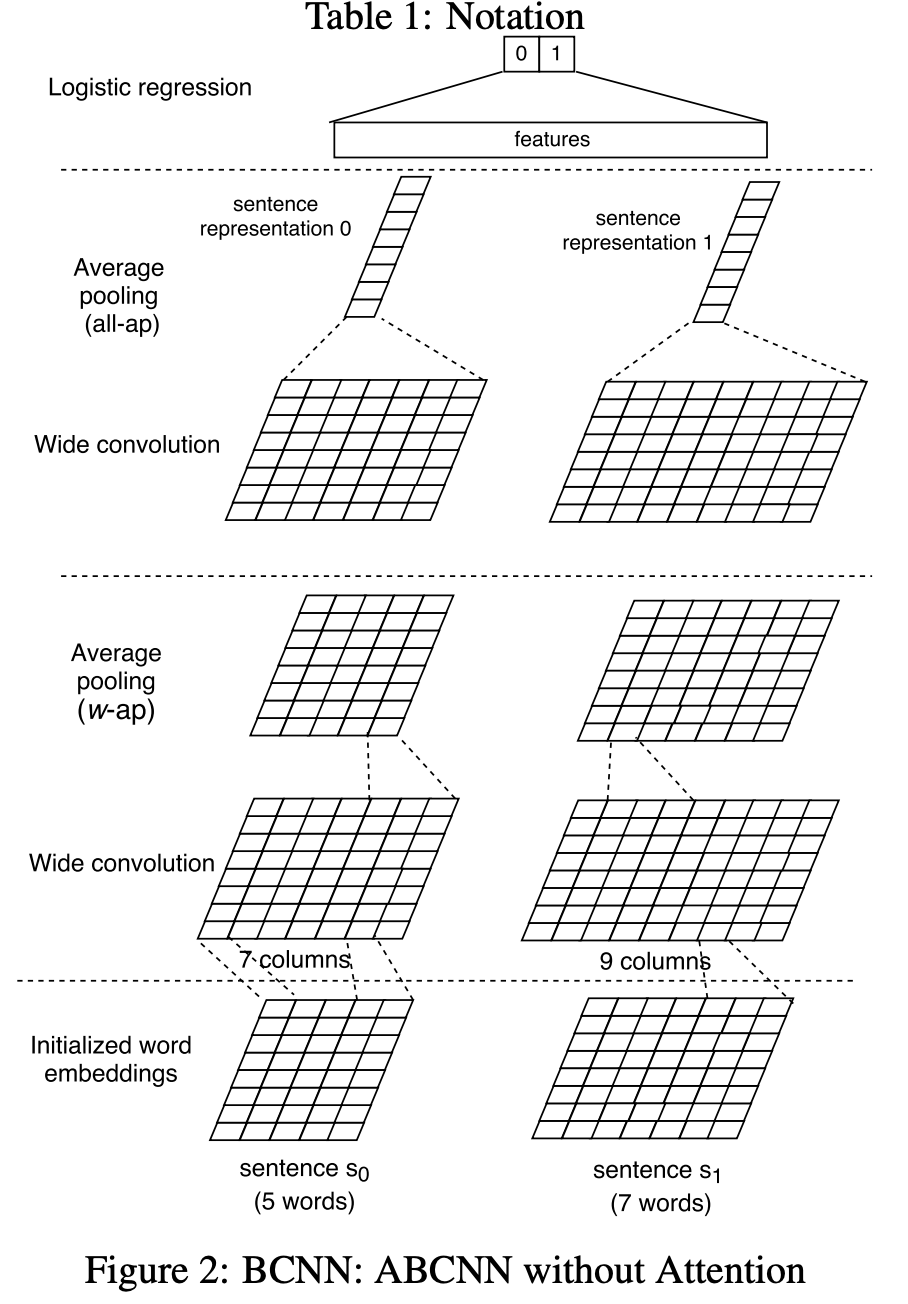
1. 输入层为两个句子的词向量表示
2. 进行宽卷积，即1-d卷积，

具体做法是：

1）假如卷积核size为3\*1，为了使得第一个词和最后一个词都被卷积3次，则，需要在前后各补两列0，然后进行卷积操纵，卷积完成之后得到的feature map为7\*embedding size和9\*embedding size

2）进行w-ap，1维的移动池化，池化完成之后，feature map为7\*embedding size和9\*embedding size

3）宽卷积+移动池化 这两个操作可以无限加深，然后接全池化接全联接和logistic regression即可



## ABCNN1

思想：

宽卷积前做attention，对attention后对矩阵再做宽卷积

1. 输入和bcnn一样，是词向量矩阵
2. 两个句子的词向量矩阵做attention，得到一个attention矩阵
3. 两个句子的词向量矩阵分别与attention矩阵相乘得到两个矩阵，对这两个矩阵分别进行宽卷积等操作，和bcnn一样

## ABCNN2

思想：

对宽卷积做attention，相对于abcnn1来说，attention更加注重了边界。在pooling的时候，因为pooling是对原句子矩阵做。

1. 输入和bcnn一样

2. 分别对两个embedding矩阵做宽卷积

3. 对两个宽卷积矩阵做attention

4. 对attention矩阵对行和列分别进行加和

5. 在移动poolding对时候，将attention矩阵加和得到对两列矩阵作为权重，来做pooling

6. 后面对就和bcnn一致

## ABCNN3

思想：

综合了abcnn1 和 abcnn2

进行了两次attention

