Word2vec

对于CBOW和Skip-gram两个模型，word2vec给出了两套框架，他们分别基于hierarchical(我理解为哈夫曼树)和负采样来进行设计。

**Hierarchical Softmax**

和之前的神经网络语言模型相比，霍夫曼树的所有内部节点就类似之前神经网络隐藏层的神经元,其中，根节点的词向量对应投影后的词向量，而所有叶子节点就类似于之前神经网络softmax输出层的神经元，叶子节点的个数就是词汇表的大小。在霍夫曼树中，隐藏层到输出层的softmax映射不是一下子完成的，而是沿着霍夫曼树一层一层完成的，因此这种softmax取名为"Hierarchical Softmax"。

**负采样**

由于训练词向量模型的目标不是为了得到一个多么精准的语言模型，而是为了获得它的副产物——词向量。所以要做到的不是在几十万个token中艰难计算下一词，而只需能做到在几个词中找到对的那个词就行。

**缺点是上下文无关(static）**

因而为了让句子有一个整体含义(context)，大家会在下游具体的NLP任务中基与词向量的序列做encoding操作。

Bi-LSTM

注: 添加到lstm的缺点之中

但是利用LSTM对句子进行建模还存在一个问题：无法编码从后到前的信息。在更细粒度的分类时，如对于强程度的褒义、弱程度的褒义、中性、弱程度的贬义、强程度的贬义的五分类任务需要注意情感词、程度词、否定词之间的交互。举一个例子，“这个餐厅脏得不行，没有隔壁好”，这里的“不行”是对“脏”的程度的一种修饰，通过BiLSTM可以更好的捕捉双向的语义依赖，即Bi-LSTM实现上下文相关（context）。

从前到后和后到前分别做一遍LSTM的encoding操作，从而获得两个方向的token(词)联系，进而获得句子的context。

**不完全双向**

模型的前向和后向LSTM两个模型是分别训练的，最后得到的隐层向量直接拼接得到结果向量，并且在最后的Loss function中也是前向和后向的loss function直接相加，并非完全同时的双向计算。

**自己看见自己**

经过两层的双向操作，每个位置上的输出就已经带有了原本这个位置上的词的信息了。这样的“窥探”会导致模型预测词的任务变得失去意义，因为模型已经看到每个位置上是什么词了。

BERT（Bidirectional Encoder Representation from Transformers）

Transformer舍弃了RNN的循环式网络结构，完全基于注意力机制来对一段文本进行建模。Transformer所使用的注意力机制的核心思想是去计算一句话中的每个词对于这句话中所有词的相互关系，然后认为这些词与词之间的相互关系在一定程度上反应了这句话中不同词之间的关联性以及重要程度。因此再利用这些相互关系来调整每个词的重要性（权重）就可以获得每个词新的表达。这个新的表征不但蕴含了该词本身，还蕴含了其他词与这个词的关系，因此和单纯的词向量相比是一个更加全局的表达。

**Masked Language Model**

在输入一句话的时候，随机地选一些要预测的词，然后用一个特殊的符号来代替它们。尽管模型最终还是会看到所有位置上的输入信息，但由于需要预测的词已经被特殊符号代替，所以模型无法事先知道这些位置上是什么词，这样就可以让模型根据所给的标签去学习这些地方该填的词了。

**sentence-level representation**

预测输入BERT的两端文本是否为连续的文本，作者指出引入这个任务可以更好地让模型学到连续的文本片段之间的关系。类似word2vec的单词级负采样。

**位置编码**

**Transformer模型并没有捕捉顺序序列的能力，也就是说无论句子的结构怎么打乱，Transformer都会得到类似的结果。**

**为了解决这个问题，论文中在编码词向量时引入了位置编码（Position Embedding）的特征。具体地说，位置编码会在词向量中加入了单词的位置信息**

**Transformer —— attention is all you need**

Transformer中抛弃了传统的CNN和RNN，整个网络结构完全是由Attention机制组成。更准确地讲，Transformer由且仅由self-Attenion和Feed Forward Neural Network组成。

对比RNN

作者采用Attention机制的原因是考虑到RNN（或者LSTM，GRU等）的计算限制为是顺序的，也就是说RNN相关算法只能从左向右依次计算或者从右向左依次计算，这种机制带来了两个问题：

(1) 时间片 t 的计算依赖 t-1 时刻的计算结果，这样限制了模型的并行能力；

(2) 顺序计算的过程中信息会丢失，尽管LSTM等门机制的结构一定程度上缓解了长期依赖的问题，但是对于特别长期的依赖现象,LSTM依旧无能为力。

基本结构

transformer的本质上是一个Encoder-Decoder的结构。

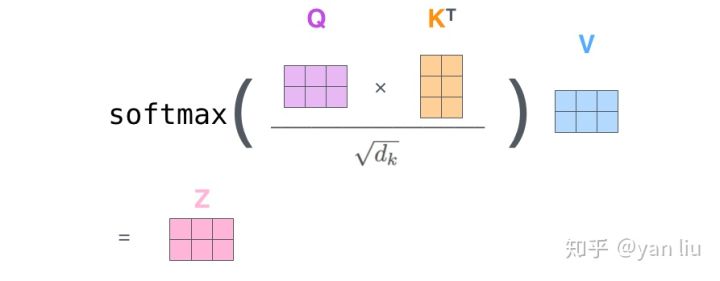
首先通过Word2Vec等词嵌入方法将输入语料转化成特征向量，从编码器输入的句子首先会经过一个自注意力（self-attention）层，这层帮助编码器在对每个单词编码时关注输入句子的其他单词。自注意力层的输出会传递到前馈（feed-forward）神经网络中。每个位置的单词对应的前馈神经网络都完全一样。

接下来我们看看Transformer的一个核心特性，在这里输入序列中每个位置的单词都有自己独特的路径流入编码器。在自注意力层中，这些路径之间存在依赖关系。而前馈（feed-forward）层没有这些依赖关系。因此在前馈（feed-forward）层时可以并行执行各种路径。

self-attention(**自注意力机制)**

其核心内容是为输入向量的每个单词学习一个权重。比如当模型处理这个单词“it”的时候，自注意力机制会允许“it”与“animal”建立联系。当我们在编码器#5（栈中最上层编码器）中编码“it”这个单词的时，注意力机制的部分会去关注“The Animal”，将它的表示的一部分编入“it”的编码中。

计算自注意力的第一步就是从每个编码器的输入向量（每个单词的词向量）中生成三个向量。也就是说对于每个单词，我们创造查询向量、键向量和值向量。它们是通过3个不同的权值矩阵由嵌入向量 X 乘以三个不同的权值矩阵 W^Q W^K， W^V 得到的。



multi-headed attention(多头注意力机制)

Multi-Head Attention相当于 h个不同的self-attention的集成（ensemble）。

如，当编码“it”一词时，一个注意力头集中在“animal”上，而另一个则集中在“tired”上，从某种意义上说，模型对“it”一词的表达在某种程度上是“animal”和“tired”的代表。

优点

(1)设计创新，因为其抛弃了在NLP中最根本的RNN或者CNN并且取得了非常不错的效果。

(2)算法的并行性非常好，符合目前的硬件（主要指GPU）环境。

缺点: 粗暴的抛弃RNN和CNN虽然非常炫技，但是它也使模型丧失了捕捉局部特征的能力

TCP

三次握手

第一次握手：建立连接时，客户端发送syn包（syn=j）到服务器，并进入SYN\_SENT状态，等待服务器确认；SYN：同步序列编号（Synchronize Sequence Numbers）。

第二次握手：服务器收到syn包，必须确认客户的SYN（ack=j+1），同时自己也发送一个SYN包（syn=k），即SYN+ACK包，此时服务器进入SYN\_RECV状态；

第三次握手：客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK(ack=k+1），此包发送完毕，客户端和服务器进入ESTABLISHED（TCP连接成功）状态，完成三次握手。

四次握手

（1）客户端A发送一个FIN，用来关闭客户A到服务器B的数据传送。

（2）服务器B收到这个FIN，它发回一个ACK，确认序号为收到的序号加1。和SYN一样，一个FIN将占用一个序号。

（3）服务器B关闭与客户端A的连接，发送一个FIN给客户端A（报文段6）。

（4）客户端A发回ACK报文确认，并将确认序号设置为收到序号加1（报文段7）。

这是因为服务端的LISTEN状态下的SOCKET当收到SYN报文的建连请求后，它可以把ACK和SYN（ACK起应答作用，而SYN起同步作用）放在一个报文里来发送。但关闭连接时，当收到对方的FIN报文通知时，它仅仅表示对方没有数据发送给你了；但未必你所有的数据都全部发送给对方了，所以你可以未必会马上会关闭SOCKET,也即你可能还需要发送一些数据给对方之后，再发送FIN报文给对方来表示你同意现在可以关闭连接了，所以它这里的ACK报文和FIN报文多数情况下都是分开发送的.