2.1 Word Embedding介绍

*Word Embedding词嵌入，是一种词的表示形式，可以简单地理解为将词进行分布式向量化表示；词向量在深度学习模型的应用领域中有着非常重要的地位，类比计算机视觉领域、语音识别领域，图片、语音等想要被深度学习算法所识别，就必须转换成对应的像素矩阵，音频块，那么在自然语言处理领域，词语想要被深度学习算法所识别，就必须转化成为词向量，只有将实体的词抽象成为数学描述，才能进行建模，应用到更多的自然语言处理的任务中去；（原创）*

NLP中最直观，也是最常用的词的表示方法之一就是one-hot Representation，即独热表示，这种方法把词表示为一个很长的向量，这个向量的维度是词表大小，其中绝大多数元素为0，只有一个维度的值为1，这个维度就代表了当前的词，这种one-hot Representation如果采用稀疏方式存储，即给每个词分配一个数字ID，会是非常的简洁，但也存在着很多的不当之处，首先，向量的维度会随着句子的词的数量类型增大而增大，其次，任意两个词之间都是孤立的，无法表示出词与词之间的相关信息，而这一点却是其致命的缺点；

Word embedding是一种分布式的特征表述，相对于one-hot表示法，它将向量每一个元素由整形改为浮点型，变为整个实数范围的表示，将原来稀疏的巨大维度压缩嵌入到一个更小维度的空间，如图示：



2-3 word embedding维度空间压缩变换图

向量的不同维度用来表征不同特征，不同维度上就代表着不同的语义，例如苹果和红旗都是红色的，苹果和香蕉都是水果。所以编码中编入的信息包含颜色和属性信息的话，则实质上往不同空间维度上投影，则语义相似性度量的结果是不一样的。如下图所示：

针对本文所研究的答案选择排序任务，如图。。所示，首先要做的是，将问题和答案进行预处理，将句子中的词语转换 成可以被神经网络识别的特征表示形式，即词向量，进而将词向量进行连接生成句子向量，最终以向量矩阵的形式输入到神经网络模型中；（原创）

2.2 经典的word embedding模型

Harris在1954年提出的分布假说（distributional hypothesis）为将语义融入到词表示中提供了理论基础，即上下文相似的词，其语义也相似，词的语义由其上下文决定，到目前为止，基于分布假说的词表示方法根据建模的不同，主要可以分为基于矩阵的分布表示、基于聚类的分布表示和基于神经网络的分布表示，word embedding就是基于神经网络的分布表示，其核心是上下文的表示以及上下文与目标词之间的关系的建模。为了选择一种模型刻画目标词与其上下文之间的关系，需要在词向量中捕获到一个词的上下文信息，而统计语言模型正具有捕捉上下文信息的能力，所以，常用的方法就是利用语言模型构建上下文与目标词之间的关系，通过训练语言模型，获得词向量；

常见的几种word embedding模型有如下几种：

2.2.1神经网络语言模型

Bengio等人在2001年发表的文章中提出了使用一个三层的神经网络来构建语言模型，这也是训练语言模型的最经典之作，模型如下图所示：



图中最左方的是目标词的前个词，根据这个词预测目标词，表示词所对应的词向量，模型共享同一个词向量库，通过查找的方式找出词语对应的词向量；网络的输入层（input layer）将这个向量进行查找，取出后首位相拼连，形成一个维的向量，记为，隐藏层（hidden layer）直接使用计算得到，是一个偏置项，是输入层到隐藏层的参数矩阵，使用作为激活函数，输出层（output layer）的每个节点表示下一个词为的未归一化log概率，最终，输出词的概率分布为：



式中的是隐藏层到输出层的参数矩阵，整个模型的对数计算集中在和隐藏层的矩阵乘法中，参数矩阵包含了输入层到输出层的直连边，也就是输入层直接到输出层的一个线性变换，当置为0时，表示不存在输入层到输出层的直连边。

2.2.2 Log Bi-Linear模型

2.2.3 CBOW模型和Skip\_gram 模型

Mikolov等人在2013年发表的文章中提出了word2vec，从此word2vec便成了word embedding中最为人所知的模型，word2vector代码中包含了两个模型CBOW和Skip-Gram，其训练目标相较之前的NNLP有所不同，是得到一组较好的词向量而不是一个较好的语言模型。

为了得到较高效的训练模型，在CBOW模型中，移除了前向反馈神经网络中的hidden layer,直接将中间层的embedding layer与输出层的softmax layer连接，在输出层使用分层softmax优化方法代替传统的softmax方法；忽略上下文环境的序列信息，将输入的所有词向量信息均汇总到同一个embedding layer，并将future words纳入到上下文环境

CBOW模型等价于一个词袋模型的向量乘以一个embedding矩阵，从而得到一个连续的embedding向量，这也是CBOW模型名称的由来；CBOW模型依然是从上下文对目标词语的预测中学习到词向量的表达，上下文取自存在于目标词前面和后面的各个特征词，即，如图2-7左侧所示，为CBOW模型的原理图。

Skip-Gram模型不同于其他模型的是，它从目标词语对上下文的预测中学习到词向量，如图2-7右侧所示，这个模型的名称源于其在训练时会对上下文环境里的词进行采样，如果将Skip-Gram模型的前向计算过程写成数字形式，则：



是embedding层矩阵里的列向量，也被称为的输入向量，是softmax层矩阵里的行向量，也被称为的输出向量，因此，通过计算输入词的输入向量和输出词的输出 向量之间的相似关系来学习模型参数，即词向量；





图2-8 。。。。。。。。。。。。。。。。。。。。

2.2CNN介绍

2.2.1 CNN介绍

卷积神经网络（Convolutional Neural Network）最早是应用在计算机视觉当中，而如今，CNN也早已应用于自然语言处理的各个任务中，卷积神经网络有三个重要的环节：卷积，池化，激活函数；

卷积运算可以理解为作用于矩阵的一个滑动窗口函数，滑动窗口也可以称作卷积核、滤波器或特征检测器，在图像中滤波器通常是对图像的一小块区域进行计算，而在文本中，是以一句话所构成的词向量矩阵作为输入，每一行代表一个词的词向量，所以在处理文本时，滤波器通常覆盖上下几行的词，滤波器的宽度与输入词向量的宽度相同，通过这种方式，多个连续词之间的特征就能够被捕捉到，并且能够在同一类特征计算时共享权重；

池化层一般是在卷积层之后，池化层对输入做降采样，池化的过程实际上是对卷积层分区域求最大值或者对每个卷积层求最大值，在自然语言处理中，通常是对整个输出层做池化，每个卷积层只有一个输出值，池化首先是可以输出一个固定大小的矩阵，这对于自然语言处理当中输入句子的长度不一的情况有非常大的作用。例如，如果使用200个卷积核，并对每个输出使用最大池化，那么无论卷积核的尺寸是多大，也无论输入数据的维度或者单词个数如何变化，都将得到一个200维的输出，这可以应对不同长度的句子和不同大小的卷积核，总是得到一个相同维度的输出结果，用作最后的分类；另外池化层在降低数据维度的同时还能够保留显著的特征，每一种卷积核都是用来检测一种特定的特征。在句子分类中，每一种卷积核可以用来检测某一种含义的词组，如果这种类型的含义的词语出现了，该卷积核的输出值就会非常大，通过池化过程就能够尽可能地将该信息保留下来；

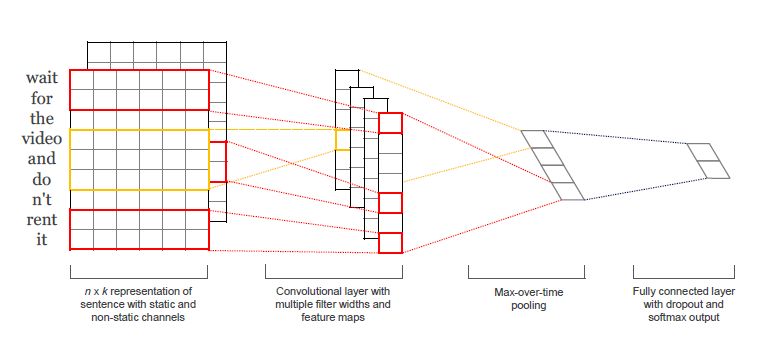


图2-1 卷积神经网络处理自然语句的过程示意图

图2-1示意了卷积神经网络处理自然语句的过程，输入为个词语的词向量组成的矩阵，为卷积窗口的大小，首先通过卷积操作，。。。。。。。。。。。。。。。。。。。。。。。

2.2 LSTM介绍

人类并不是每时每刻都从一片空白的大脑开始他们的思考，在阅读一篇文章的时候，都是基于自己已经拥有的对先前所见词的理解来推断当前词的真实含义，而不会将所有的东西都全部丢弃，然后用空白的大脑进行思考，也就是说人类的思想具有持久性，传统的神经网络并不能做到这点，例如，假设希望对电影中的每个时间点的事件类型进行分类，传统的神经网络很难来对这个问题进行处理，很难使用电影中先前的事件推断后续的事件；

RNN解决了这个问题，RNN不仅会学习当前时刻的信息，也会依赖之前的序列信息，由于其特殊的网络模型结构决定了其信息保存的能力，所以RNN对处理时间序列和语言文本序列问题有独特的优势，递归神经网络都具有一连串重复神经网络模块的形式，每个神经网络会把消息传递给下一个，如果将这个循环展开，则如下图所示：



图1：RNN网络的链式结构图

在上图所示的神经网络中，输入为，输出为，上的环允许将每一步产生的信息传递到下一步中，这种链式的特征揭示了RNN本质上是与序列和列表相关的，它们是对于这类数据最自然的神经网络架构；在标准的RNN中，这个重复的模块只有一个非常简单的结构，例如一个tanh层，如下图所示：



图2：标准RNN网络中重复模块的内部结构

长短期记忆网络（Long Short Term Memory Network,LSTM)，是一种改进之后的循环神经网络，可以解决RNN无法处理长距离的依赖所导致的梯度消失问题，以及由于依赖激活函数和网络参数而可能导致的梯度爆炸问题，近几年来被广泛应用，LSTM通过可能的设计来避免长期依赖问题，记住长期的信息在实践中是LSTM的默认行为，而非需要很大的代价才能获得的能力。

 图3：LSTM网络的内部结构

LSTM的关键点是细胞状态，也叫单元状态，就是穿过图中的水平线，单元状态类似传送带，它贯穿整个链条，只有一些线性的相互作用，这使得信息很容易的能以不变的方式向下流动；LSTM有能力向单元状态中移除或添加信息，通过门结构来管理，包括“遗忘门”，“输出门”，“输入门”，通过门让信息选择性通过，来去除或增加信息到单元状态，模块中sigmoid层输出0到1之间的数字，描述了每个成分应该通过门限的程度，0表示不让任何成分通过，而1表示让所有成分通过。

“遗忘门”决定从“单元状态”中丢弃什么信息；比如完形填空中填“他”或者“她”的问题，细胞状态可能包含当前主语的类别，当看到新的代词，希望忘记旧的代词。如上图所示，左边第一个即为遗忘门，将上一时刻的输出和这一时刻的输入进行拼接，然后判断以多大的程度来保留这部分信息，即得到概率值，如下式所示，









“输入门”决定放什么信息到单元状态中；如图，第二个sigmoid层决定什么值需要更新，如式（2）所示，代表的是以多大的概率来更新信息；tanh层创建一个新的候选值向量，如式（3）所示，表示现在的全部信息，这两个操作都是为状态更新做准备；接着更新“细胞状态”；更新为，如式4）所示，把旧状态与相乘，就丢弃掉了确定需要丢弃的信息，然后将乘以，以确定要更新的信息，通过相加操作得到新的细胞状态；

接着根据“细胞状态”得到输出，首先运行一个sigmoid层来确定细胞状态在哪个部分将输出，如式（5）所示，接着用tanh处理细胞状态，得到一个在-1~1之间的值，再将它和sigmoid门的输出相乘，如式（6）所示，得到确定输出的部分。





2.3 注意力机制介绍

2.3.1 Encoder-Decoder框架介绍

在Attention Mechanism被广泛应用于自然语言处理领域的大潮流中，大部分的AM都是基于Encoder-Decoder框架，Encoder-Decoder框架可以看作是一种自然语言处理领域的研究模式，应用场景非常广泛，下图是自然语言处理领域里常用的Encoder-Decoder框架最抽象的一种表示：

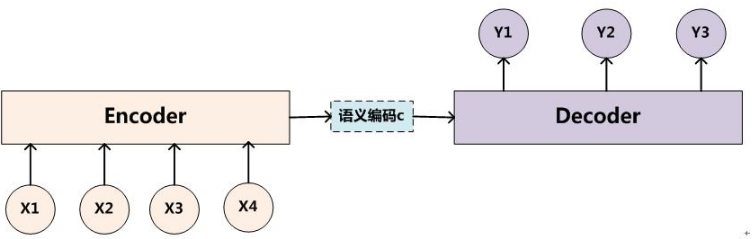


图2-4 自然语言处理领域里的Encoder-Decoder框架

对于机器翻译来说，就是对应不同语言的句子，比如是英语句子，是对应的中文句子翻译。再比如对于文本摘要来说，就是一篇文章，就是对应的摘要；再比如对于对话机器人来说，就是某人的一句话，就是对话机器人的应答，但在上图中展示的Encoder-Decoder模型是没有体现出“注意力机制”的，其目标句子Y中每个单词的生成过程如下：







其中是decoder的非线性变换函数。从这里可以看出，在生成目标句子的单词时，不论生成中的哪个单词，他们使用的句子的语义编码都是一样的，没有任何区别。而语义编码是由句子的每个单词经过Encoder 编码产生的，这意味着不论是生成哪个单词，句子中任意单词对生成某个目标单词来说影响力都是相同的，没有任何区别，没有引入注意力模型在输入句子比较短的时候问题不大，但是如果输入句子比较长，此时所有语义完全通过一个中间语义向量来表示，单词自身的信息已经消失，便会丢失很多的细节信息，所以需要将注意力机制引入到Encoder-Decoder框架中。

2.3.2 Attention Mechanism介绍

Attention Mechanism(AM)是一种从认知心理学中引入的概念，它模拟人脑的注意力模型，强调了在某一个特定时刻，人脑对于资源进行不均衡分配的核心思想，举例来说，当观赏一副画时，虽然可以看到整幅画的全貌，但在某个特定的时刻深入观察时，人的意识和注意力的焦点是集中在画面中的某一个部分的，即人脑对整幅图的关注是不均衡的，是有一定权重上的区分的。

在自然语言处理领域，AM最突出的成就是在机器翻译任务中，下图为以RNN为基本网络模型，并引入Attention机制的翻译模型；



图2-5 引入注意力机制的翻译模型

图中的是经过Decoder后的译文词语，是输入的待encoder的词汇，从图中可以看到，Attention机制的主要思想体现在，当前译文词语不仅和之前的历史信息有关系,还选择性的使用了输入句子中每个词汇的某种输出信息，如图所示，的输出信息以大小的程度决定当前译文词语。这样，每个解码器的输出词汇通过一个权值矩阵，以不同的程度依赖于所有的输入信息；

称为注意力分配概率，它的计算十分关键，以下图为例，介绍的计算过程，对于采用RNN的Decoder来说，如果要生成单词，需要计算生成时的输入句子单词对来说的注意力分配概率分布，那么可以利用时刻的隐层节点状态去一一和输入句子中每个单词对应的RNN隐层节点状态进行对比，即通过函数来获得目标单词和每个输入单词对应的对齐可能性，然后函数的输出经过Softmax进行归一化就得到了符合概率分布取值区间的注意力分配概率分布数值；



图2-6 注意力分配概率的一般计算过程

2.3数据集收集