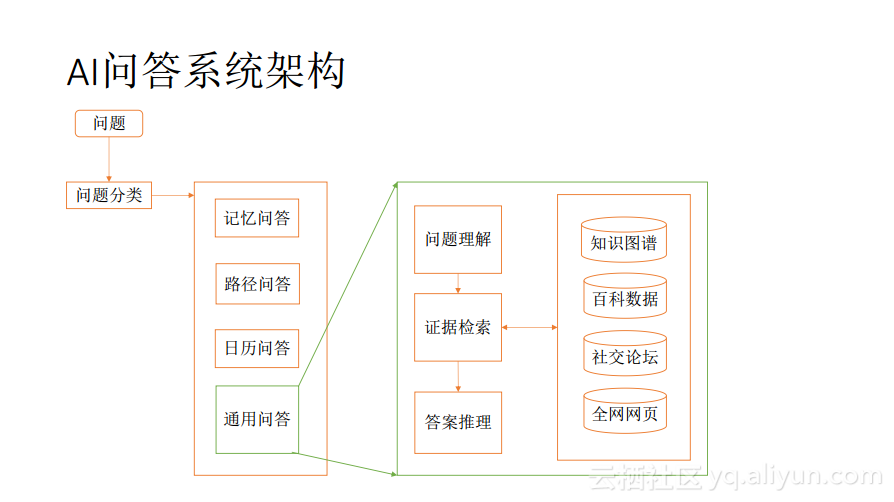
**来源：AI是如何回答你提出的问题的？揭秘智能问答系统背后的深度学习网络**

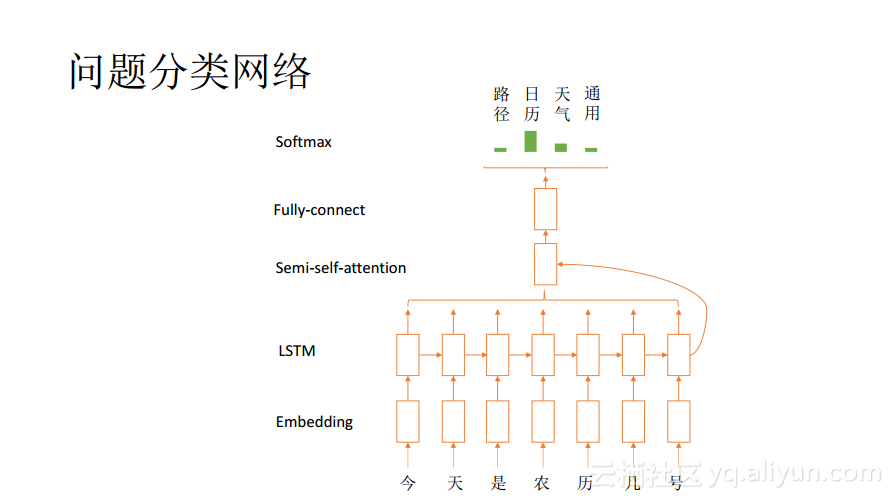
**问答AI将会无处不在**  
本文将为大家分享问答系统背后的原理以及深度学习在问答系统中的应用。本文中所谓的“问答”就是“一问一答”，也就是用户提出问题，AI系统给出回答。随着人工智能和物联网技术的飞速发展以及两者之间的相互融合和落地，越来越多的设备将会被植入问答AI，比如随身会携带的耳机、手表、戒指，家里的音响、台灯等，汽车和机器人等，最终AI将会无处不在。未来，大家可以随时随地与AI进行交互，提出问题并获得答案，或者通过AI来操控物理世界。问答AI将会成为一个非常重要的入口，其地位就相当于现在的搜索引擎。



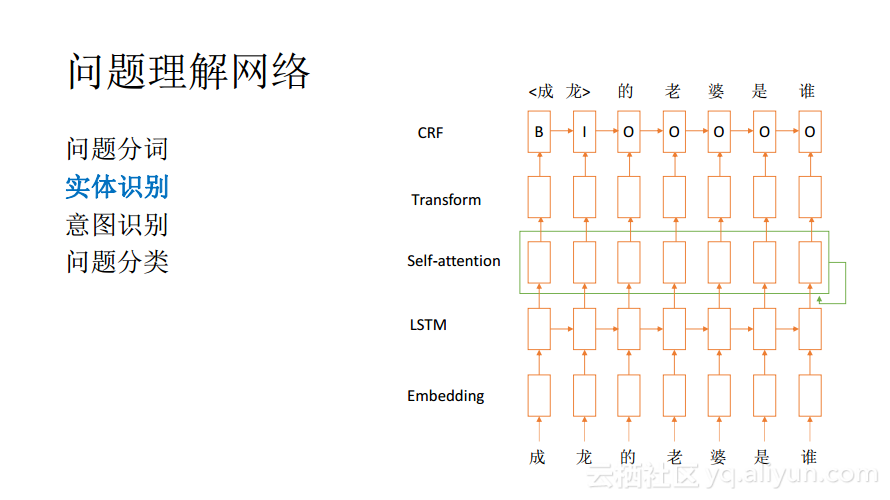
**人类如何回答问题**  
在考虑设计一个AI问答系统之前，不妨先来考虑一下人类是如何回答问题的。比如提出一个问题，“现任美国总统是谁？”大家可能瞬间就能够给出答案，因为像这样的简单问题在大脑里已经有了记忆，人类可以根据记忆瞬间地给出回答。还有一些问题则需要垂直领域的知识，经过科学的计算和缜密的推理才能给出答案，比如数学或者物理相关的问题。此外，还有一些问题需要多个领域进行综合分析才能给出回答，比如在回答“肺炎的患者吃什么食物比较合适？”这个问题时，不仅需要知道医学领域中对于肺炎的病理，还需要知道食物和营养学等相关知识，只有结合以上这几个领域的知识才能够给出比较合理的答案。而面对更加复杂的问题，仅凭借人类大脑里面的知识储备可能还不够，还需要借助外部工具搜集一些数据和资料才能进行回答。  
  
**AI问答系统架构**  
人类回答问题的基本逻辑就是对复杂的问题进行分解，分而治之。AI问答系统也借鉴了这样的逻辑，把整个系统进行分解，然后将不同的问题放入到不同的垂直问答领域中做回答。当然，还会有一个非常通用的通用问答。在通用问答里面主要包含了3大部分，即问题理解，证据检索以及答案推理。AI收到问题后，先对问题进行各种各样的分析、处理和理解；接着基于分析结果从不同的数据源检索出相关的证据，也就是检索可能包含答案的数据；然后，进行综合地阅读理解和推理并且给出答案。问答AI的各个子系统都可以用深度学习实现。



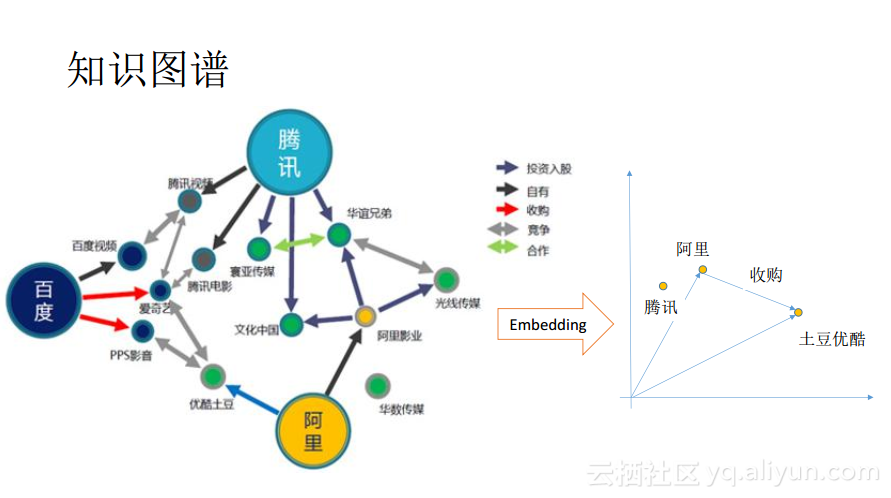
**问题分类网络**  
接下来为大家分享通用问答这条流水线上面的各个子系统背后的深度学习模型。首先为大家分享问题分类网络，问题其实就是一段话，也就是一个符号序列。如果想用神经网络处理一个符号序列，则首先需要将其embedding到向量空间中去，因为这个符号序列是具有上下文的语义上的相关性的，所以需要借助LSTM来进行处理，将上下文的语义信息能够融合到词的向量中去。因为序列是变长的，所以为了便于后续的处理需要将整个序列转化成一个固定维度的向量，转化方法比较多，简单粗暴的方法就是将序列里面的所有的向量加起来再取平均；稍微复杂一些的方法则可以考虑使用多个卷积层和pooling层，逐步将长序列转化成为固定长度的向量；semi-self-attention是简单而高效的方法，把序列的最后一个状态作为key，和序列自己做一个attention pooling。之后再做一些非线性的变化，最终使用Softmax来输出类别。如下图所示的就是一个问题分类网络。



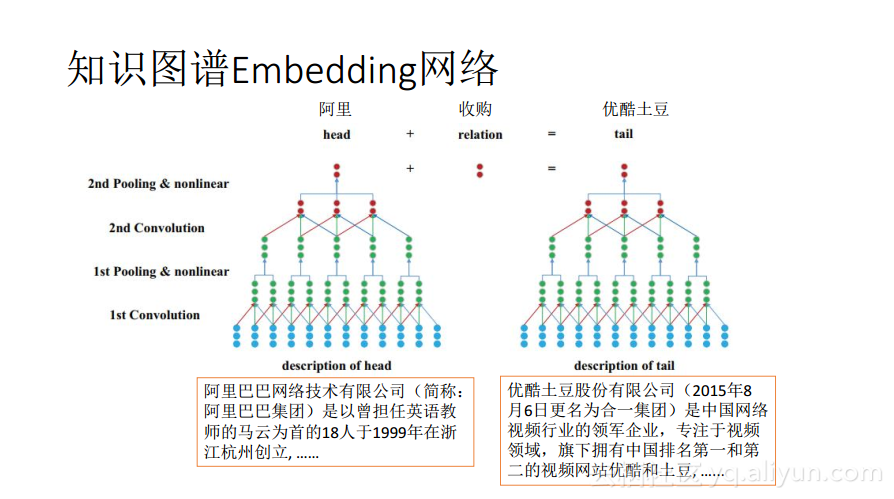
**问题理解网络**  
其实上面所提到的问题分类也可以认为是一种问题理解的方式，因为问题理解本身没有明确的定义，只要是有助于回答后面问题的处理就都可以加入进来。一般而言，问题理解中的处理大致有这样几个：问题分词、实体识别、意图识别以及问题分类。其中的意图识别也是分类问题，而分词和实体识别都是序列标注问题。接下来以实体识别为例分享序列标注问题。



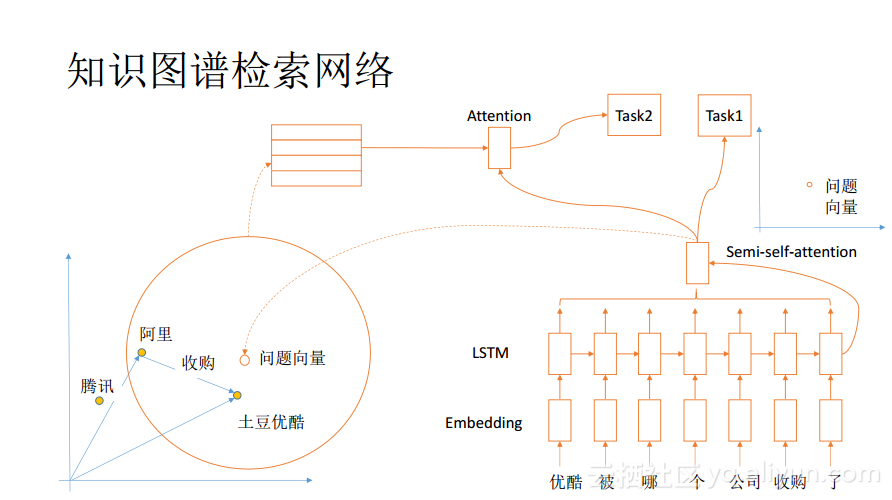
实体识别的目标是识别问题中哪些词属于实体，并且属于什么实体，比如“成龙的老婆是谁？”这个问题就包含了一个人名实体，当然其他的场景下面可能还有地名、机构名称以及道路名称等，这些都是实体。实体的标志方法一般是用B、I、O这种符号表示，其中B，也就是Beginning代表着实体词的开始；I代表Inside也就说这个词还是在这个实体里面；O代表Outside也就是说这个词已经不属于这个实体了。在前面的这个问题中“成龙”这个词就构成了实体。序列标注模型一般会在下面使用深度学习，上面则使用CRF类来做解码。  
  
深度学习的网络与前面所提到的问题识别其实是大同小异的，最下面两层也负责将序列转化成为空间中的序列，并且尽可能多地将语义的信息包含在其中。在问题理解网络模型中间会加一个Self-attention层，虽然LSTM能够捕获上下文的一些信息，但是在实际中，其对于较长句子的分析能力还是不足的，并且也不能很好地捕获一些比较低频的问题的语义。而Self-attention层能够将序列中的任意两个词都进行关联并捕获他们之间语义的关系，这样就不会受限于长度，可以在任意远的两个词之间捕获语义的相关性。所以其对于语义的处理，特别是标注都是很有帮助的。上图所展现的就是一个问题理解中的深度学习网络。  
  
**知识图谱**



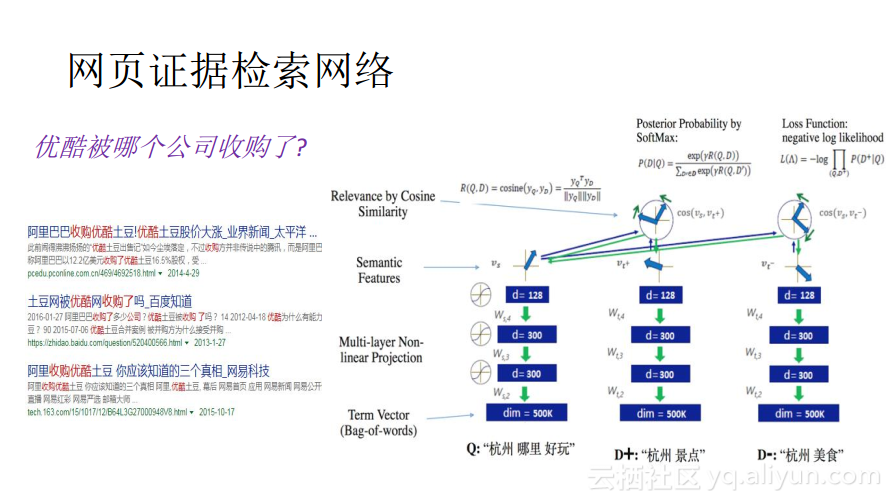
完成了以上的步骤后就需要去考虑检索证据。证据的来源比较多，一个重要的来源就是知识图谱。在过去二三十年的互联网化以及信息化进程中，各行各业都沉淀了很多的知识，这些知识往往都是以知识图谱的形式保存下来的。知识图谱是典型的图结构，节点就表示实体，边则表示实体之间的关系。如果想要在知识图谱上使用深度学习方法做检索，首先需要考虑的就是将知识图谱embedding到一个向量空间里面去，再在向量空间中进行语义检索。我们面临的第一个问题，是如何将一个复杂的知识图谱embedding到一个向量空间中去？大家不妨思考一下知识图谱具有什么样的特点，其所具备的一个非常重要的特点就是它由三元组组成，而三元组具有非常强烈的约束，使用符号的形式进行表达是很容易理解的，比如通过<阿里，收购，优酷土豆>这个三元组就能了解这样的一种关系。但是将知识图谱中的三元组embedding到向量空间中去之后，应该如何去描述这种关系呢？这是一个非常关键的问题。  
  
**知识图谱Embedding网络**  
有一句话叫做“简单的就是美的”，实体可以变成向量，而向量之间最简单的代数计算就是相加或者相减，所以可以考虑让三元组中的前两个向量相加，等于第三个向量，通过简单的相加关系来表达知识图谱中符号上的逻辑关系。在如下图所示的例子中，阿里+收购=优酷土豆。



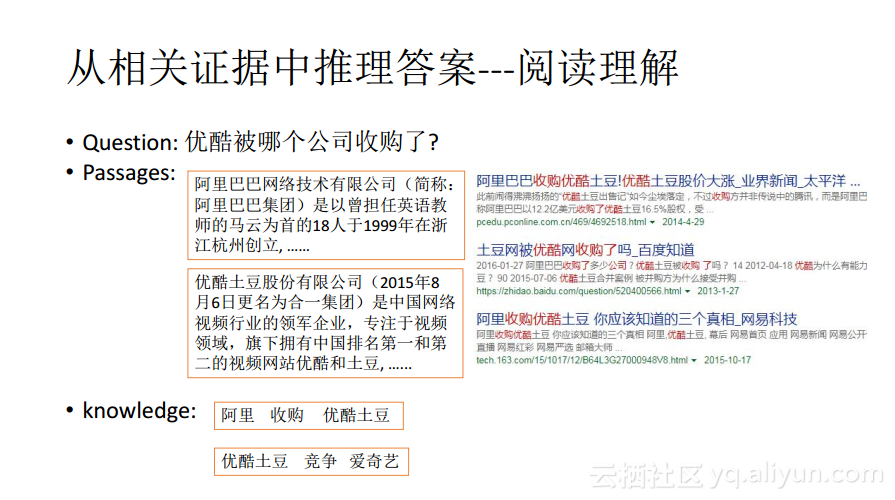
一个知识图谱往往是非常庞大的，很多时候就会有几百万个实体，以及几十亿个关系，所以需要设计一个损失函数来尽量多地满足等式关系，最终驱动实体和边embedding到向量空间的比较好的点上面去。因为现在知识图谱里面除了实体的信息之外还可能有其他的信息，比如对实体以及关系的描述，这种的描述往往是自然语言的形式，比如句子、段落甚至是一篇文章。像这些信息也是可以用序列embedding的方式先将其转化成为一个向量，在向量空间里也强迫其满足等式关系，将有助于实现整个知识图谱的embedding，有助于后续语义上的处理。  
  
**知识图谱检索网络**  
当知道了如何embedding之后，接下来需要考虑如何检索知识图谱。此时，知识图谱里面的实体已经变成向量了，而问题也可以表达成向量，自然就可以使用问题向量在知识图谱中寻找接近的向量，这样所找出来的向量很可能与问题是语义相关的，甚至很有可能就包含了答案。但是在实际中这种做法可能会存在问题，因为问题表达学习与知识图谱的embedding是两个独立的任务，学习出来的向量空间实际上存在于不同的坐标系里面，在不同的坐标系里面，即便坐标相同，所表示的内容也可能是不相同的。所以需要将其做进一步的对齐，只有在坐标对齐之后才能判断两个坐标系中的点是否接近，从而判断是否是真正地语义上的相近。对齐两个坐标系可以用Attention机制来实现，当然需要标注数据，设计新的Task来驱动Attention里面的参数收敛，使得两坐标系内的向量是可比较的。如果标注的数据量够多，也可以对问题向量学习里面的网络参数同时做fine tune，并且能够起到更好的效果提升。在对齐之后就可以将问题向量丢到知识图谱的向量空间中去了，将最相关或者最近的实体和关系找出来，这样就能够获得最优的证据。



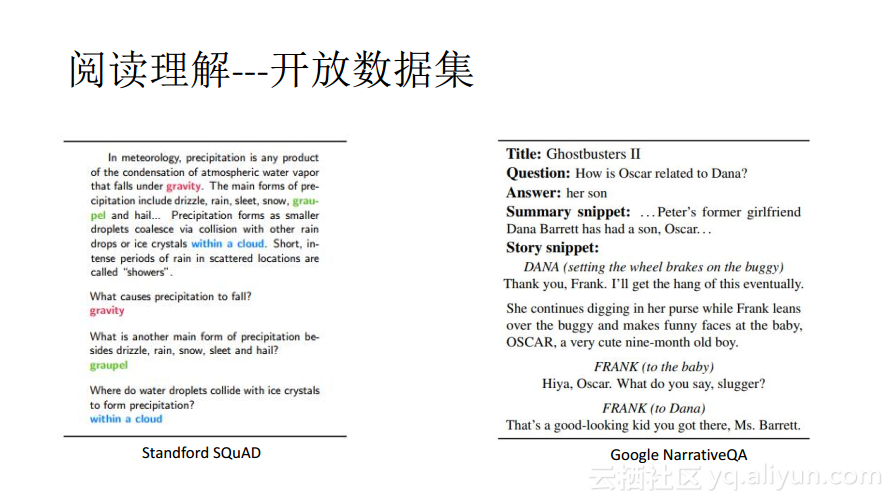
**网页证据检索网络**  
除了知识图谱是一个重要的证据来源之外，其实还有一个更大的证据来源在于互联网。在互联网中找证据就是一个经典的搜索Ranking的问题，其核心就是衡量问题与网页之间的语义相关性，找出与问题最为相关的网页。用神经网络解决这个问题，需要把网页文本和问题embedding到向量空间中，还需要设计一个相似性函数，一般而言可以使用COS相似性，用COS值来衡量语义，经过不断的学习之后使得学习出来的向量的COS值能够代表语义上的相近性，最终就可以用COS值来找寻与问题最为相关的网页。



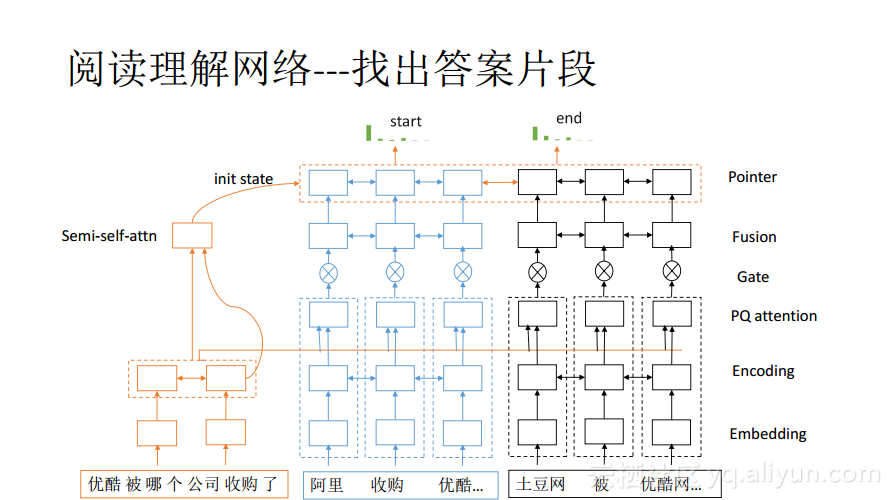
**从相关证据中推理答案--阅读理解**  
接下来分享当将证据检索完之后接下来该如何做。前面提到知识图谱和网页中的数据形式是具有一定差异的，还有一些证据会来自其他数据源比如百科、论坛以及图片视频等，而这些不同形态的数据连起来就会非常复杂。如何去进行综合阅读和推理是一个非常复杂的问题，也是强AI的问题，当解决了这个问题也就能跨越到强AI的阶段。



而到目前为止，这个问题还没有一个非常好的解决方案，所以在该问题上的研究和讨论也非常火爆。最近也有很多公司和研究机构发布了许多简化的开放的数据集供大家测试和研发算法，比如Standford的SQuAD数据集，这个数据集里面有500多个文章以及10万多个问题，而去年百度、谷歌以及微软也都发布了一些数据集，这些数据集也更大并且更加复杂，像百度和微软所发布的数据集中段落的数目有100多万，问题也有20多万，并且这些数据集也更加接近于真实的场景。



**阅读理解网络---找出答案片段**  
当面对复杂的阅读理解问题时，该如何设计深度学习模型来解决这个问题呢？其实这里还不能称之为“解决”只能是“探索性解决”，因为现在还没有一个特别好的解决方案。这里为大家分享一个目前在SQuAD数据集上表现比较好的网络结构。



在这个网络结构里输入是一个问题和多个段落，目标是在多个段落里面找出答案片段，当然也可能有多个答案片段。自然也需要先对这些问题和段落进行embedding，在此基础之上再进行进一步地处理。中间层的处理方式与前面所提到的问题分类以及序列标注都有所不同，在中间层的PQ attention是将段落和问题进行关联，让段落的表达能够蕴含问题的语义，具体的做法就是让段落中的每一个词都单独地对整个问题进行attention，这样做完之后就能够使得每个段落中的每个词都能够包含问题，也就是一定程度上将问题是什么以及段落中的词与问题的关联强度如何展现出来。因为段落往往比较长，会有很多杂音，所以上面再加一个Gate层，对无关的词起到过滤抑制的作用。在此之上再加一层Fusion进行聚合，其功能就是将Gate层锐化的信息再做一个平滑，同时把前后的信息在此进行融合，使其能够捕获答案与段落关联的更多信息。在最上面是Pointer层，这里就是找答案片段的层，在其下的那些层里面是将不同的段落单独处理的，而在Pointer层中则是将所有的段落串起来作为一个段落看待，在里面寻找最佳答案；Pointer层是一个RNN，初始状态是用问题向量，输出是两个Softmax层，一个表示答案的开始，另一个表示答案的结束。  
  
**答案生成网络---自然语句生成**  
当将段落中的答案找出来之后，还需要形成一个通顺的语句并反馈给用户，而有时候答案片段可能有多个，更需要将其串接起来形成自然语句。生成自然语句是一个序列的generation问题，与机器翻译或者图像caption问题是类似的。 网络结构包含两大部分:encoder和decoder， 在Decoder部分，可以认为和上述问题都是一样的，但是在Encoder部分就会有很大的区别，因为其数据来源和结构完全不同。在机器翻译里面，Encoder所处理的就是一个序列，而在阅读理解里面，Encoder所处理的数据就既有问题也有段落，还会包含了已经找出来的答案信息。  
  
在Encoder中最下面的两层Embedding和Encode中可以完全共享寻找答案网络最下面的两层参数，这里的Start和End就是阅读理解中的Poiner层输出的结果，将其构成二值化序列放进来。当然除此之外，如果还有其他的特征也可以放入到Encoder里面来。Encoder的信息会输出给Decoder，Decoder中有一个非常关键的层就是Attention，Attention能够准确地从问题或者段落里面找一些关键词出来，使得必要的词能够被输出到最终的答案语句中去。Decoder还会生成一些非关键词，起到顺滑和链接的作用。

