机器学会阅读将是人工智能在处理和理解人类语言进程中一个里程碑式的事件，是一个真正AI必须达到的标准。最近一家叫做[Maluuba](http://www.maluuba.com/)的科技公司，号称开发了目前最领先的机器阅读理解系统EpiReader[10]，成为了业界的领跑者，也被媒体盛赞。本文是一篇机器阅读理解的综述文章，系统地总结和对比一下最近阅读过的相关paper。

问题描述

首先定义下机器阅读理解问题的数学模型。

问题可以表述为一个三元组(d,q,a)，这里d是指原文document，q是指问题query或者question（不同的paper可能称呼不同，但指的是同一回事），a是answer，即问题的答案。这个答案是来自一个固定大小的词汇表A中的一个词。我们要解决的问题就变成了：给定一个document-query对(d,q)，从A中找到最合适的答案a。

经常听到这么一句话，没有分类解决不了的问题。虽然有一点夸张，但这个问题是一个典型的多分类问题，预测候选列表中每个word或者entity的条件概率，最大的那个就是正确答案。其实很多问题也都是这样，尤其是在生成问题上，给定context，来predict每一个word的概率。

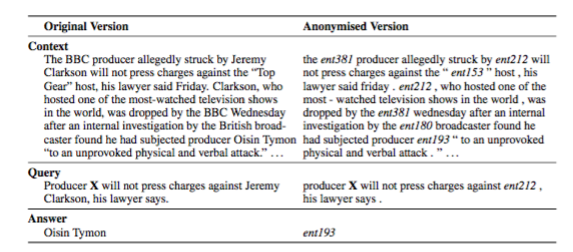
这里不同的paper在词汇表A中有些不同，有的paper将A定义为document和query中的所有词，而有的将A定义为所有的entity，而有的将会附带一个包括正确答案在内的10个候选答案，并且每个答案的词性都一致。

语料

语料对于NLP的研究有着十分重要的基础作用，尤其是大规模的语料为研究相关任务带来了革命性的变化。前些年的语料都非常小，比如MCTest。从2015年开始，出现了两大主流的大型数据集。

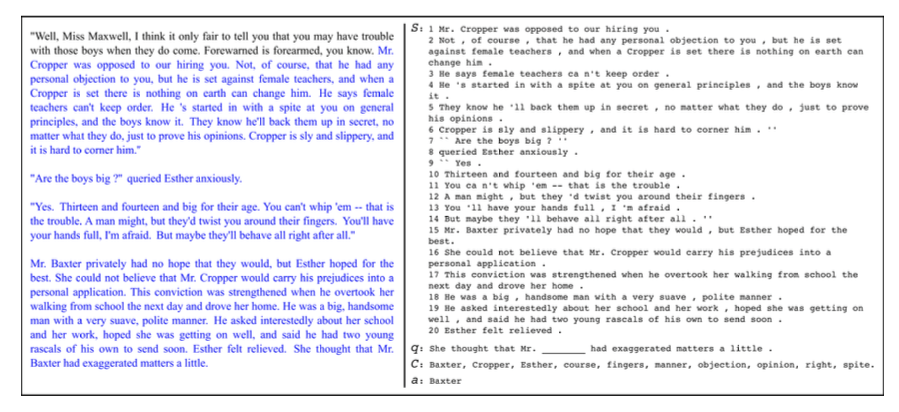
1、CNN/Daily Mail[9]

数据集构建基本的思路是受启发于自动文摘任务，从两个大型的新闻网站CNN和Daily Mail中获取数据源，用abstractive的方法生成每篇新闻的summary，用新闻原文作为document，将summary中去掉一个entity作为query，被去掉的entity作为answer，从而得到阅读理解的数据三元组(document,query,answer)。这里存在一个问题，就是有的query并不需要联系到document，通过query中的上下文就可以predict出answer是什么，也就失去了阅读理解的意义。因此，本文提出了用一些标识替换entity和重新排列的方法将数据打乱，防止上面现象的出现。处理之后的效果见下图：

[](http://rsarxiv.github.io/2016/06/13/Teaching-Machines-to-Read-and-Comprehend-PaperWeekly/fig1.png)

2、Children’s Book Test(CBT)[3]

CBT的数据均来自Project Gutenberg，使用了其中的与孩子们相关的故事，这是为了保证故事叙述结构的清晰，从而使得上下文的作用更加突出。每篇文章只选用21句话，前20句作为document，将第21句中去掉一个词之后作为query，被去掉的词作为answer，并且给定10个候选答案，每个候选答案是从原文中随机选取的，并且这10个答案的词性是相同的，要是名词都是名词，要是命名实体都是实体，要是动词都是动词。例子看下图：

[](http://rsarxiv.github.io/2016/06/16/THE-GOLDILOCKS-PRINCIPLE-READING-CHILDREN%E2%80%99S-BOOKS-WITH-EXPLICIT-MEMORY-REPRESENTATIONS-PaperWeekly/fig1.png)

左图为电子书的原文，右图为构建成数据集之后的几个元素，document、query、answer和candidate。数据集根据词性一共分为四类，第一类是Named Entity，第二类是Nouns，第三类是Verbs，第四类是Preposition。其实阅读理解问题的难度在于前两种词性，后面的两种用语言模型通过query的上下文就可以预测出来，不需要借助于document的信息。这个数据集并没有像CNN那样做替换和重排的处理，反而是鼓励大家用更少的信息来做阅读理解。

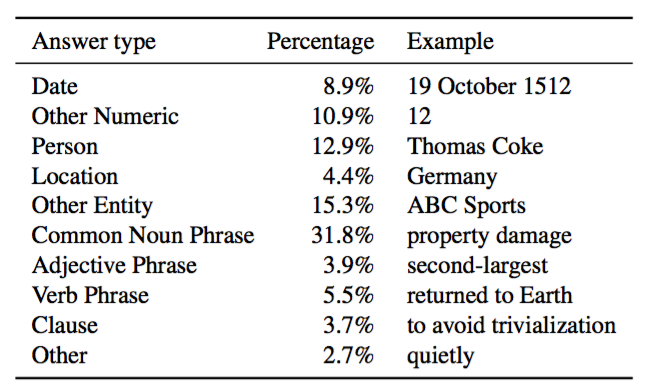
说完最流行的两个数据集，接下来介绍一下昨天刚刚在arxiv上submit的一个数据集。

3、Stanford Question Answering Dataset(SQuAD)[1]

该数据集的构建分为三个步骤：

* 在Wikipedia中选取质量排名在10000以内的article，（这里用了 Project Nayuki’s Wikipedia’s internal PageRanks来做rank），从每篇文章中提取出paragraph，经过一系列处理之后得到了23215个paragraph，涉及了很宽泛的话题。
* 然后雇佣了crowdworkers给每个paragraph提问和回答，而且鼓励workers用自己的话来提问。（这一点和CNN中用abstractive的思路很像，只不过是用了人工来做。）
* 第三步是让crowdworkers来用原文中的text（word或者是span）来回答这个问题，如果无法用原文回答的话，直接提交问题。

这个数据集的答案类型非常丰富，看下表：

[](http://rsarxiv.github.io/2016/06/18/%E6%95%99%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E9%98%85%E8%AF%BB/fig1.png)

4、LAnguage Modeling Broadened to Account for Discourse Aspects(LAMBADA)[11]

该数据集的paper于2016年6月20日于arxiv上，特补充在此。

这个数据集的数据源来自Book Corpus，一共包括10022个passage，平均每个passage包括4.6句话+1句target，每个passage大约75个单词。在选择数据的过程用了很多的人力来做，按照以下的过程：

* 第一个人阅读全文之后来猜target word，如果猜对了。
* 第二个人继续阅读全文来猜target word，如果猜对了。
* 更多人不让阅读全文，只能读target sentence来猜target word，直到猜对或者达到猜的人数上限，比如说10.如果没有人猜的对，就将该数据归入LAMBADA中。

这种方法很费时费力，但从质量上得到了保证，所获得的数据集都可以保证通过阅读全文之后一定会得到正确结果，不阅读全文一定找不到结果，避免了语言模型通过分析target sentence直接生成target word这种情况，给研究者提供了一个质量更高的数据集。

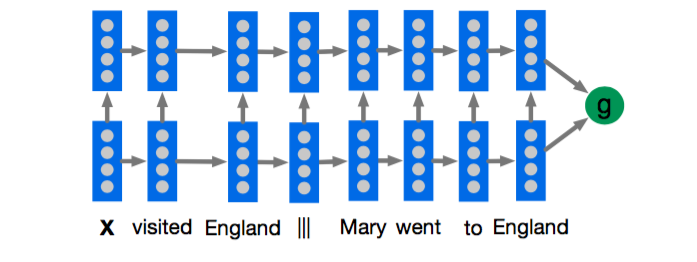
模型

本文所说的模型是指neural模型，人工features的模型就不介绍了。

1、Deep LSTM Reader / Attentive Reader / Impatient Reader[9]

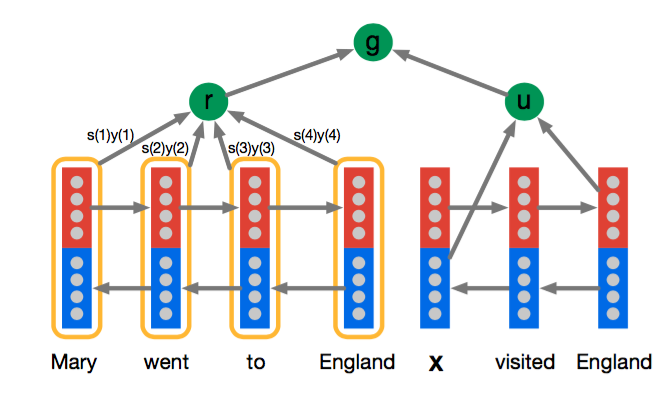
这个模型是配套CNN/Daily Mail数据集的模型，只是作为后面研究的baseline，所以效果不会太好。

* Deep LSTM Reader

[](http://rsarxiv.github.io/2016/06/13/Teaching-Machines-to-Read-and-Comprehend-PaperWeekly/fig2.png)

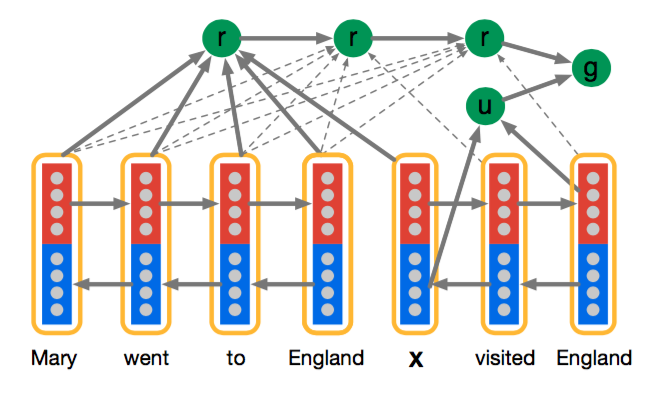
看上图，其实非常简单，就是用一个两层LSTM来encode query|||document或者document|||query，然后用得到的表示做分类。

* Attentive Reader

[](http://rsarxiv.github.io/2016/06/13/Teaching-Machines-to-Read-and-Comprehend-PaperWeekly/fig3.png)

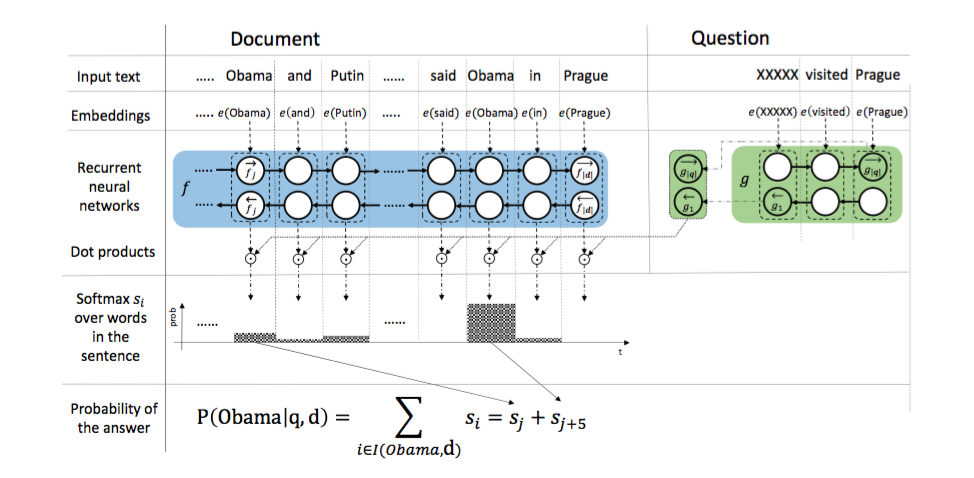
这个模型将document和query分开表示，其中query部分就是用了一个双向LSTM来encode，然后将两个方向上的last hidden state拼接作为query的表示，document这部分也是用一个双向的LSTM来encode，每个token的表示是用两个方向上的hidden state拼接而成，document的表示则是用document中所有token的加权平均来表示，这里的权重就是attention，权重越大表示回答query时对应的token的越重要。然后用document和query的表示做分类。

* Impatient Reader

[](http://rsarxiv.github.io/2016/06/13/Teaching-Machines-to-Read-and-Comprehend-PaperWeekly/fig4.png)

这个模型在Attentive Reader模型的基础上更细了一步，即每个query token都与document tokens有关联，而不是像之前的模型将整个query考虑为整体。感觉这个过程就好像是你读query中的每个token都需要找到document中对应相关的token。这个模型更加复杂一些，但效果不见得不好，从我们做阅读理解的实际体验来说，你不可能读问题中的每一个词之后，就去读一遍原文，这样效率太低了，而且原文很长的话，记忆的效果就不会很好了。

2、Attention Sum Reader[6]

[](http://rsarxiv.github.io/2016/06/14/Text-Understanding-with-the-Attention-Sum-Reader-Network-PaperWeekly/fig1.png)

**step 1** 通过一层Embedding层将document和query中的word分别映射成向量。

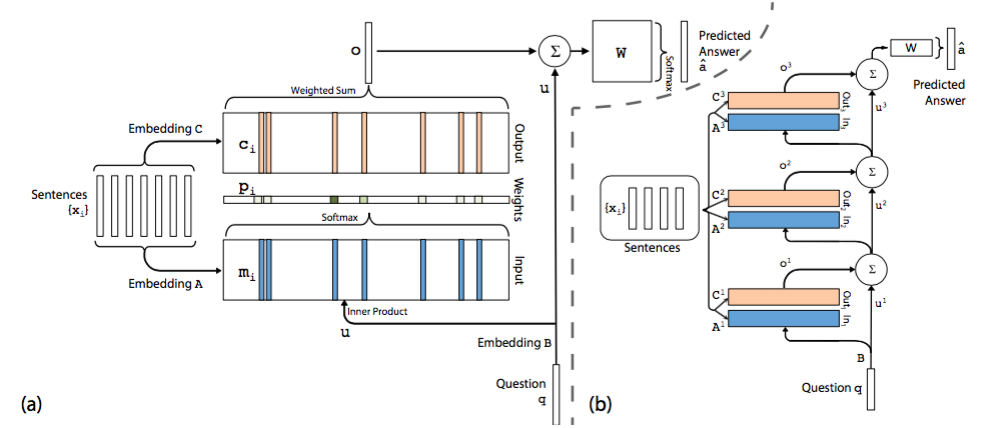
**step 2** 用一个单层双向GRU来encode document，得到context representation，每个time step的拼接来表示该词。

**step 3** 用一个单层双向GRU来encode query，用两个方向的last state拼接来表示query。

**step 4** 每个word vector与query vector作点积后归一化的结果作为attention weights，就query与document中的每个词之前的相关性度量。

**step 5** 最后做一次相同词概率的合并，得到每个词的概率，最大概率的那个词即为answer。

3、Memory Networks[3][5]

[](http://rsarxiv.github.io/2016/06/16/THE-GOLDILOCKS-PRINCIPLE-READING-CHILDREN%E2%80%99S-BOOKS-WITH-EXPLICIT-MEMORY-REPRESENTATIONS-PaperWeekly/fig2.png)

模型将document中的每一个word保存为一个memory m(i)，每个memory本质上就是一个向量，这一点与embedding是一回事，只是换了一个名词。另外每个word还与一个输出向量c(i)相关联。可以理解为每个word表示为两组不同的embedding A和C。同样的道理，query中的每个单词可以用一个向量来表示，即对应着另一个embedding B。

在Input memory表示层，用query向量与document中每个单词的m(i)作内积，再用softmax归一化得到一组权重，这组权重就是attention，即query与document中每个word的相关度，与昨天的AS Reader模型有些类似。

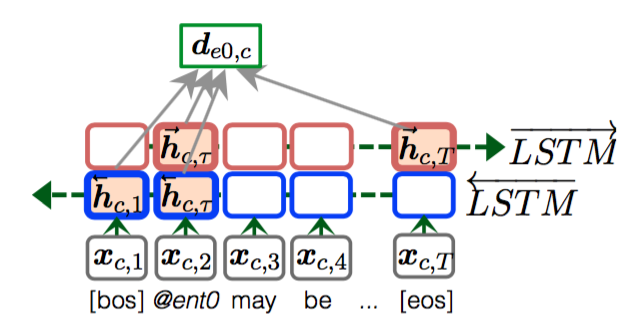
接下来，将权重与document中的另一组embedding c(i)作加权平均得到Output memory的表示。

最后，利用query的表示和output memory的表示去预测answer。

然后，介绍下右图的多层模型。根据单层模型的结构，非常容易构造出多层模型。每一层的query表示等于上一层query表示与上一层输出memory表示的和。每一层中的A和C embedding有两种模式，第一种是邻接，即A(k+1) = C(k)，依次递推；第二种是类似于RNN中共享权重的模式，A(1) = A(2) = … = A(k),C(1) = C(2) = … = C(k)。其他的过程均和单层模型无异。

本文模型的特点是易于构造更多层的模型，从而取得更好的效果。后面Gate Attention Reader模型正式借助了这个思想。

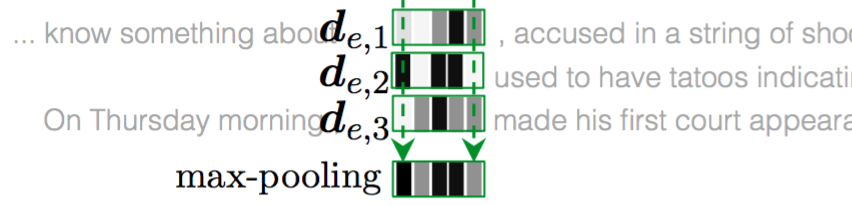
4、Dynamic Entity Representation[7]

[](http://rsarxiv.github.io/2016/06/17/Dynamic-Entity-Representation-with-Max-pooling-Improves-Machine-Reading-PaperWeekly/fig1.png)

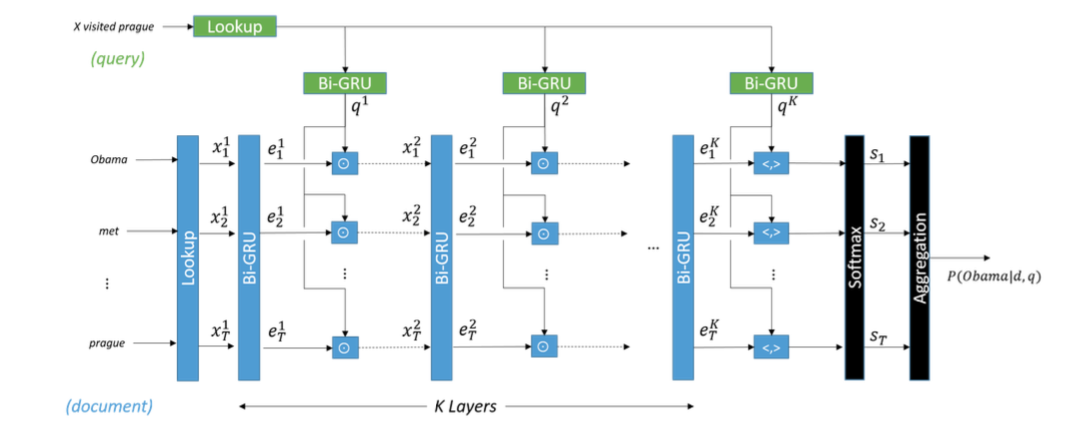
计算出entity的动态表示之后，通过attention mechanism计算得到query与每个entity之间的权重，然后计算每个entity在document和query条件下的概率，找到最终的answer。

query向量的计算与动态entity计算过程类似，这里需要填空的地方记作placeholder，也是包括四个部分，其中两个是表示placeholder上下文的last hidden state，另外两个是表示placeholder的hidden state。

模型的整个计算过程就是这样。如果遇到一个entity在document中出现多次的情况，该entity就会会有不同的表示，本文采用CNN中常用的max-pooling从各个表示中的每个维度获取最大的那一个组成该entity最终的表示，这个表示包括了该entity在document中各种context下的信息，具有最全面的信息，即原文中所说的accumulate information。如下图：

[](http://rsarxiv.github.io/2016/06/17/Dynamic-Entity-Representation-with-Max-pooling-Improves-Machine-Reading-PaperWeekly/fig2.png)

5、Gate Attention Reader[8]

[](http://rsarxiv.github.io/2016/06/12/Gated-Attention-Readers-for-Text-Comprehension-PaperWeekly/fig1.png)

**step 1** document和query通过一个Lookup层，使得每个词都表示成一个低维向量。

**step 2** 将document中的词向量通过一个双向GRU，将两个方向的state做拼接获得该词的新表示。同时也将query通过一个双向GRU，用两个方向上的last hidden state作为query的表示。

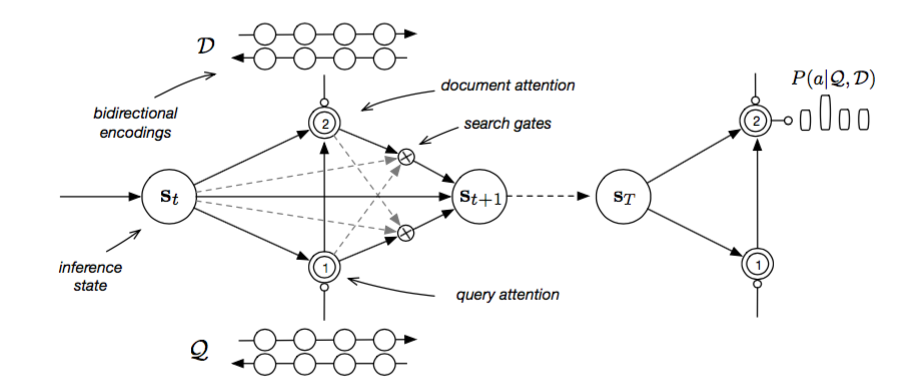
**step 3** 将document中每个词的新表示与query的新表示逐元素相乘得到下一个GRU层的输入。

**step 4** 重复step 2和3，直到通过设定的K层，在第K层时，document的每个词向量与query向量做内积，得到一个最终的向量。

**step 5** 将该向量输入到softmax层中，做概率归一化。

**step 6** 因为document中有重复出现的词，聚合之后得到最终的分类结果，即确定应该填哪个词。

6、Iterative Alternating Attention[2]

[](http://rsarxiv.github.io/2016/06/18/%E6%95%99%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E9%98%85%E8%AF%BB/fig2.png)

**step 1** 将document和query通过一个Lookup层，使得每个词都表示成一个低维向量。

**step 2** 将document和query中的词向量通过一个双向GRU，将两个方向的state拼接获得该词的新表示。

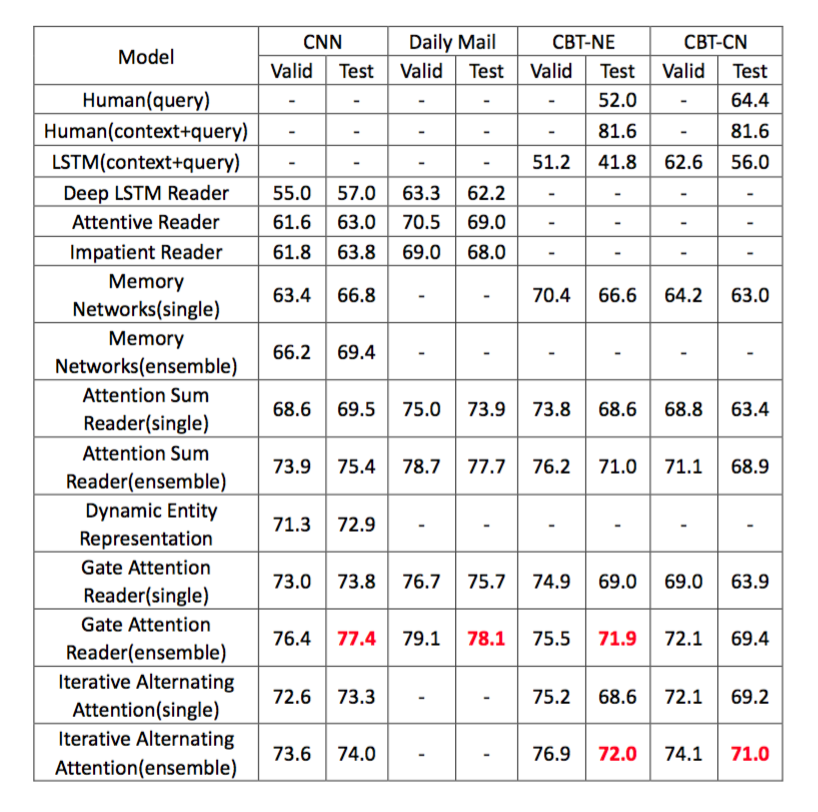
**step 3** 这一步文章中称为Iterative Alternating Attention，1）根据前一个inference状态s(t-1)来计算query的attention，得到query glimpse q(t)，对应图中的1;2)根据前一个状态s(t-1)和当前的query glimpse q(t)来计算document的attention，得到document glimpse d(t)，对应图中的2；3)用GRU来将前一个状态s(t-1)和当前的query glimpse q(t)和当前的document glimpse d(t)做处理得到当前的状态s(t)。

**step 4** 重复step 3 直到t达到给定的T为止。

**step 5** 用最后得到的每个词向量进行归一化，并且聚合相同的词概率，得到预测结果。

结果

所有模型都是在两大主流数据集上进行对比[2][8]，结果如下：

[](http://rsarxiv.github.io/2016/06/18/%E6%95%99%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E9%98%85%E8%AF%BB/fig3.png)

Maluuba公司的模型在CBT数据集上是最好的，在CNN/Daily Mail数据集上并没有最测试，而Gate Attention Reader占据了CNN数据集上的头把交椅。

model ensemble可以将single model的效果提升很多，是一种非常有效的技术。从第一个模型Attentive Reader到最后一个模型Iterative Alternating Attention时间跨度大概是半年左右的时间，阅读理解的正确率提升了近20个百分点。

CBT数据集上包含了人工测试的结果，最高的准确率为81.6%，而目前计算机可以达到的最高正确率是72%，离人类仍有不小的差距，需要更多更牛的model涌现出来。

思考

读了一周的Machine Reading Comprehension paper，有几点思考：

1、大规模语料的构建是nlp领域研究进步的重要基础和保证，深度学习模型尤其是端到端+注意力模型利用大规模的语料进行训练和学习，极大地提升了计算机阅读理解的效果。而且每出现一个新的数据集，都会弥补之前数据集存在的问题，对模型提出了更高的要求，从而提高了该领域的研究水平。同时，也给很多需要毕业的童鞋提供了一个新的刷榜平台。

2、模型的提出应该更多地联系人类是如何解决这个问题的，比如attention、copy mechanism等等优秀的模型。attention借助了人类观察一个事物的时候，往往第一眼会先注意那些重要的部分，而不是全部这个行为方式。copy mechanism而是受启发于当人类不了解一个事物或者该事物只是一个named entity，而又需要表达它的时候，需要“死记硬背”，需要从context中copy过来。模型Dynamic Entity Representation用一种变化的眼光和态度来审视每一个entity，不同的context会给同样的entity带来不同的意义，因此用一种动态的表示方法来捕捉原文中entity最准确的意思，才能更好地理解原文，找出正确答案。实际体会中，我们做阅读理解的时候，最简单的方法是从问题中找到关键词，然后从原文中找到同样的词所在的句子，然后仔细理解这个句子最终得到答案，这种难度的阅读理解可能是四、六级的水平，再往高一个level的题目，就需要你联系上下文，联系关键词相关联的词或者句子来理解原文，而不是简单地只找到一个句子就可以答对题目。

3、[4]和[6]中用简单的模型反而会得到更好的结果，[9]中用了非常复杂的注意力机制反而并没有太好的结果。[2][8]都证明了用多层结构比单层结构更优秀。那么，是不是多层简单的模型会有更好的效果？这是一个需要动手实践来研究的问题。

4、当前的这些模型都是纯粹的data-driven，并没有考虑人工features进来。我一直坚信，如果做一个准确率非常高的系统的话，neural models+features是必须的，针对具体的问题，做具体的分析；但是如果是对于学术界研究model的话，提出更牛的neural models比纯粹的刷榜更有意义。

参考文献

[1] [SQuAD: 100,000+ Questions for Machine Comprehension of Text](http://cn.arxiv.org/pdf/1606.05250v1)

[2] [Iterative Alternating Neural Attention for Machine Reading](http://arxiv.org/pdf/1606.02245v3.pdf)

[3] [THE GOLDILOCKS PRINCIPLE: READING CHILDREN’S BOOKS WITH EXPLICIT MEMORY REPRESENTATIONS](https://arxiv.org/pdf/1511.02301.pdf)

[4] [A Thorough Examination of the CNN Daily Mail Reading Comprehension Task](http://arxiv.org/pdf/1606.02858v1.pdf)

[5] [End-To-End Memory Networks](http://arxiv.org/pdf/1503.08895v5.pdf)

[6] [Text Understanding with the Attention Sum Reader Network](https://arxiv.org/pdf/1603.01547.pdf)

[7] [Dynamic Entity Representation with Max-pooling Improves Machine Reading](http://www.cl.ecei.tohoku.ac.jp/publications/2016/kobayashi-dynamic-entity-naacl2016.pdf)

[8] [Gated-Attention Readers for Text Comprehension](http://cn.arxiv.org/pdf/1606.01549v1)

[9] [Teaching Machines to Read and Comprehend](http://rsarxiv.github.io/2016/06/13/Teaching-Machines-to-Read-and-Comprehend-PaperWeekly/)

[10] [Natural Language Comprehension with the EpiReader](http://cn.arxiv.org/pdf/1606.02270.pdf)

[11] [The LAMBADA dataset:Word prediction requiring a broad discourse context](http://cn.arxiv.org/pdf/1606.06031)