阅读理解实验阶段性总结

徐俊

2016 年 12 月

Contents

1 摘要

阅读理解任务从七月底开始以来，至今已经快五个月，在此梳理一下开始直接的实验思路和进展。时间上来看，从八月到九月中旬，调研并达到 baseline 的效果；九月下旬到十月下旬针对数据和模型进行分析，在进行多项改进 NN 模型未果之后，大方向选择在 log-linear 的框架下融入经典 NLP 特征以及 NN 特征；十一月至今，大方向调整为如何利用经典 NLP 特征来提升任务。出于简洁直观考虑，本文按照实验类别（纯粹的模型增强实验、在 log-linear 框架下融入经典特征以及在 NN 框架下融入经典特征）分别介绍。

2 任务概述

参见文章《两种阅读理解模型框架的概要介绍》。两个模型框架见 Figure 1和 Figure 2。

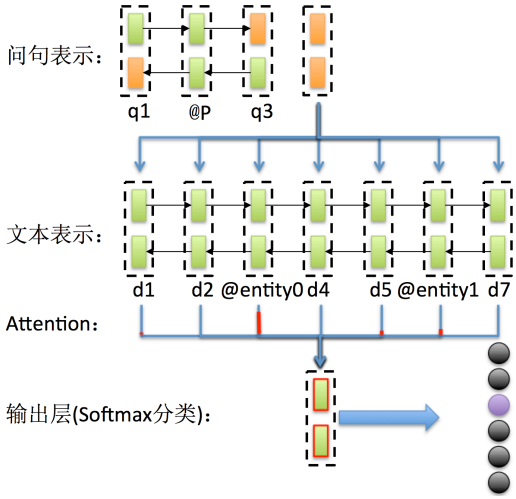

Figure 1: Attention Reader

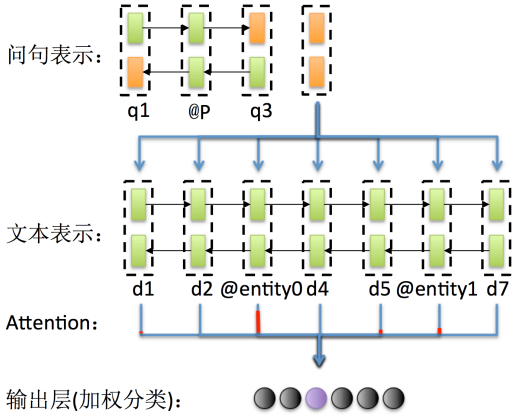

Figure 2: Attention-Sum Reader

3 模型增强实验

这部分实验包括所有不涉及融入新特征的 NN 实验。主要分为两个大的类型：更加重视局部信息以及 Attention Reader 同 Attensum Reader 的融合模型 (Attention-Attensum Reader)。

3.1 更加重视局部信息

1. **动机**：分析负例数据的时候发现，目标词很多时候并不是通常意义上的句子核心实体。比如问句是“探险家怎么怎样，然后乘坐帆船，从 @placeholder 抵达 @entity1，……”，在这样的句子中目标词很难通过问句来刻画，故而想到更多的利用目标词周围信息来提升性能；

2. **做法**：做法一只用目标词前后各三个词，作为“问句”，其他不变；做法二是仅利用 @placeholder 处的隐层状态作为问句表示；做法三是将原问句的表示同做法一中“新问句”的表示做拼接，作为问句表示；做法四是将原问句表示同做法二获得的表示做拼接，共同作为最终的问句表示；

3. **结果**：当时 attentive reader baseline 性能在 71%，而这四组模型的结果（未经严格调参的情况下）性能均没有超过 70%；

4. **结论以及后续**：这种做法并没有显著提升，而随着 attentive reader baseline 调整到 72.6%，这一部分的实验并没有继续下去，原因是感觉这个方向较为不可行；

3.2 Attention-Attensum Reader

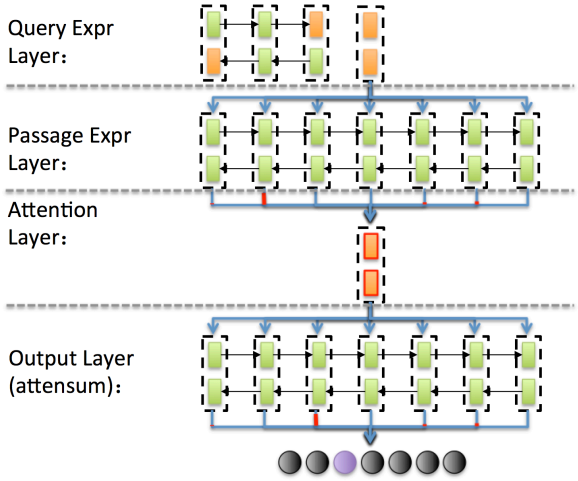

Figure 3: Attention-Attensum Reader。同 Attention Reader 相比，仅仅在输出层有所不同。Attentive Reader 为每个 @entity 符号赋予一个特定的权重向量（不同文本中即使 @entity 所代表的词已经完全不同了，仍旧使用同一个权重向量），而 Attention-Attentive Reader 的每个 @entity 的权重向量则是该词在 RNN 中对应时刻的隐层状态。相比较而言，Attention-Attentive Reader 中的权重向量选取更合理。

1. **动机**：@entity 实体在不同文章中表示不同的词，而在 softmax 解码的时候，一个 @entity 符号对应唯一的权重向量，这就使得 attention reader 在解码的时候实际上是“混乱”的，至少这加大了训练的难度；

2. **做法**：需要的其实就是为每个 @entity 符号寻找一个对应的有实际意义的权重向量，那么可以直接使用 RNN 编码过程中 @entity 符号对应时刻的隐层状态作为权重向量。

3. **目前状态**：截至目前，已经尝试运行三次，均因服务器故障以及资源问题等原因未能获得顺利运行；

4. **结论以及后续**：继续做；但是在写作本文的时候，发觉一个潜在的问题，就是抛开所谓 attention，softmax，权重向量、隐层状态等等概念，实际上从数学角度而言 attention layer 同 output layer 在做同一件事情，做两次同样的事情是否是有意义的？当然另外一角度也说明这个模型本身非常便于扩展叠加多层。

4 log_linear 系列实验

4.1 Baseline

Baseline 采用八种特征，包括候选实体是否出现在文章中、出现在问句中、频率、第一次出现在文章中的位置、n-gram 特征、依存特征等等。Chen 的文章报出来的结果是 67.1%，我这边实现出来的结果是 65.8%。训练使用开源的 lambdaMART。

4.2 增加 NN 输出结果为新特征的实验

4.2.1 lambdaMART

将 NN 给每个候选词打的分数作为特征添加到 lambdaMART，做了两组实验，第一组是直接将分数（概率）作为特征；第二组是将 NN 给出的概率最高的那个词设置特征为 1 其余为 0；两组实验均没有达到 65.8%，感觉不太对，于是换成 SVM 再做一下。

4.2.2 SVM

直接使用 libsvm 训练，单机器一直没有训练出来结果，训练一个礼拜未果，放弃；
使用线性核的 rankSVM 来训练，性能是 62%，但是奇怪的是如果使用开发集作为训练数据却可以获得 68% 的效果；
非线性核的 rankSVM 也没有跑出来；
使用公司的 LTR 来做，gbrank 效果是 67%。
之后没有继续该方向的实验。

5 NN 框架下融入经典特征的系列实验

5.1 Attentive Reader

5.1.1 Baseline

Figure 1 是经典的 Attentive Reader 模型框架图。从最开始的调研到初步实现，再到调参，性能只能抵达 65%，在陈丹琦公布源码之后，找到其中诸多 trick，之后 baseline 性能抵达 72.6%，同丹琦报出来的结果基本持平。

这一部分的总结是：网络的维度、embedding 的维度以及数据预处理，对于性能影响最大，而其他超参基本上没有本质影响。调参的时候太过谨慎，每次只调一个，造成时间和计算资源的低效率使用。

5.1.2 经典特征丰富模型输入

1. **动机**：做负例数据分析的过程中，一方面发现部分情况下（25%）目标词并没有被很好的刻画，比如目标词分明是个人、或者是机构，而返回的答案却是个“山”、在这种情况下，借助经典特征来帮助 Reader“刻画”目标词便成了一个可行的方向。

2. **做法**: 提取依存关系的类型作为特征，加入到输入层，也即输入到 RNN 中的是词向量和对于特征向量的拼接。依存关系是有方向的，根据提取入边或者出边的关系类型作为特征分别作了一组实验；此外，将依存关系中父节点的词作为特征以及 postag 作为特征，也分别做了一组实验；上述四组实验均采用直接拼接的方法，一次对应，做了四组词向量和特征向量过一个非线性层预处理的实验；

3. **结果**：由于做的时候是为了做 ensemble，所以没有对单独的模型做特别的调参，就综合效果而言，加入依存关系类型的实验效果在 69% 左右，加入依存父节点词作为特征的实验效果在 71%，计入 postag 性能在 70%，而非线性并没有对性能造成明显影响；

4. **猜想**：一方面可能是因为特征对于任务作用有限，而这八组实验均没有超过 baseline（即使没有严格调参这也能说明问题了）；另一方面是任务本身的问题，@entity 在不同文章中表示不同的内容，这直接导致词向量、对应时刻 RNN 隐层状态以及解码矩阵中的权重向量，均失去明确的意义，为任务带来混乱。

5.1.3 经典特征用于调整 attention 机制

1. **动机**：同上一节。

2. **做法**：Attentiive Reader 中 Attention 层，取 query 表示和文章中各个时候的隐层状态做点乘，然后 softmax 获得“权重”。将特征在这个层面直接加入，query 表示向量拼接一个目标词的特征向量，而文章各个时刻的隐层状态也拼接各自时刻的词语特征向量，如此直接利用特征干预筛选过程。具体做法上，有三种，其一，仅用 01 标识出 entity 的位置，这样便于权重集中到 @entity 等有效候选实体上面；其二，文章中仅仅 @entity 具有特征，其他普通词的特征向量全部置为 UNK；其三，文章中各个词义均有特征向量。特征向量用的是依存关系中父节点位置上的词。

3. **结果**：由于代码 bug，双向 rnn 返回的隐层状态是反向的，这对于上一节的做法么有影响，却对于本节的做法直接干扰，进而训练所得结果均是没有提升；调整 bug 之后，attentive reader 出现 NAN 问题，原因不明；由此此时已经调试完成 Attensum Reader，而且训练速度更快性能相当，所以转到 attensum reader 上做实验了，本节实验暂停。

5.2 Attensum Reader

5.2.1 Baseline

Figure 2 是经典的 Attensum Reader 模型框架图。直接利用 attention 过程中的权重作为词输出的概率，唯一的不同是需要将同一个词对应的权重加和作为最终该词的概率。baseline 效果是 72.5%，而如果使用数据预处理将所有答案全部替换为 @entity0，则出现 99% 的效果。

5.2.2 经典特征丰富模型输入

暂时没有做，鉴于 Attention Reader 的实验，这一部分暂缓。

5.2.3 经典特征用于调整 attention 机制

1. **初始 attensum**：

   baseline：72.65%；高峰抵达 73%；同时观测到 500 句左右，集中到 entity 的概率不足 0.5。

   baseline+ 仅有 entity 上有特征其他词语特征为 UNK：72%；同 baseline 一样，500 多句集中到 entity 的概率不足 0.5 去掉针对特征 embedding 的 L2 正则之后，性能没有提升，有 300 多句集中到 entity 的概率不足 0.5。

   baseline+ 所有词语均有特征：72.1%；同 baseline 一样，500 多句集中到 entity 的概率不足 0.5。

   baseline+entity 用特征 1 标示其他词语用 0 标示：72.3%；仅用 01 表示 entity 位置，去掉针对 embedding 的 L2 正则，优化损失函数（鼓励概率集中到 entity 上）全部句子集中到 entity；再度优化之后，性能徘徊在 72.5% 左右。

2. **改变 TF 输入方式以及去掉损失优化以及仅在 entity 上面解码**：

   baseline+ 仅有 entity 上有特征其他词语特征为 UNK：70%；

   baseline+ 所有词语均有特征：正在做；

   baseline+entity 用特征 1 标示其他词语用 0 标示：等待做；

   baseline+ 特征选取 Postag：等待做；

5.3 基于经典特征做 Rerank 的实验

1. **动机**：利用特征来帮助选择，同前面两节将特征融入 NN 的做法不同，这边是基于 NN 给出候选答案，然后基于特征在候选答案中进一步筛选。

2. **做法**：利用多种 NN 模型，attentive reader、attentive reader+ 经典特征丰富，共计五种模型，然后每个模型随机选择两个中间结果（性能已经稳定），这样每个文本-问句对共有十个候选答案。一般而言，这十个候选答案中只有两三个不同的 @entity；利用目标词的特征向量和这些候选词的特征向量点乘，算的权重，权重最大的那个候选答案成为最终的结果。

3. **结果**：性能抵达 71%；而实际上，如果仅仅利用这十个候选答案进行投票，则性能抵达 75%（此处还出过一个比较严重的乌龙事件，前后占用十天时间）。

4. **目前状态**：如此简单的二次筛选就能取得不错的结果，但是接下来怎么做就没有具体规划了。如何更合理的利用经典特征，这是个开放问题。

6 当前实验总结

总体而言，将经典特征整合进 NN 的实验，有两种方式。其一是丰富输入特征，在 attentive Reader 中并没有取得很好的效果，在 Attensum Reader 中暂时没有做。其二是将特征用于调整 attention

机制，之前在 Attentive Reader 中的实验由于 bug 导致之前的结果丧失意义，而调整之后的代码出现 NAN，于是暂停去做 Attensum Reader 的对应实验；Attensum Reader 的 baseline 性能正常，在 72.5% 左右，而其他已经完成的实验均没有明显突破。

7 下一步实验计划

1、迁移到 lego，实验室的机器饱和，速度缓慢。
2、需要做的实验是**将特征用于调整 attention 机制**下 Attensum Reader 的剩余实验以及 Attentive 的实验。此外，Attentio_Attensum Reader 的实验也要继续。
3、考虑更换数据集，目前的数据集收到 @entity 这种没有固定语义的现象影响严重，各种模型均受此影响。