Welcome

MSRA Researcher新手，现阶段从事多模态大模型预训练方向。对NLP感兴趣或对博客内容有疑惑及意见建议的，欢迎评论或添加我微信。此外如果有需要内推的同学，也欢迎来骚扰我。联系方式详见concat页面。

前往GitHub (https://github.com/Dod-o)

论文|记忆网络之Memory Networks

6月 14, 2019 (http://www.pkudodo.com/2019/06/)

论文|记忆网络之Memory Networks

作者 Dodo (http://www.pkudodo.com/author/root/) 在 (http://www.pkudodo.com/2019/06/14/1-13/)
NLP (http://www.pkudodo.com/category/nlp/),
对话系统 (http://www.pkudodo.com/category/%e5%af%b9%e8%af%9d%e7%b3%bb%e7%bb%9f/) 标签
memory network (http://www.pkudodo.com/tag/memory-network/),
综述 (http://www.pkudodo.com/tag/%e7%bb%bc%e8%bf%b0/)

Memory NetWorks介绍

在文本的处理上，由于很多地方对记忆的需要，因此诞生了RNN及LSTM。但RNN和LSTM也只能用于短时间内的记忆（一般来说也就十几个step）。所以如果文本较长的话，RNN和LSTM也无能为力了。

当然也有一种方法，就是直接扩大RNN和LSTM的隐状态大小，让其可以存储更多的信息。但相对于这种方式，我们更希望能够任意地增加记忆量，同时能够对模型做尽可能小的改变。

Memory Networks正是从这一角度出发得到的产物。直观来讲，Memory Networks可以理解成正常的model额外加一个记忆模块。

就好像我们使用的CPU（普通的model），它内部是有一个很小的ram的（我本科阶段是嵌入式方向，目前见过cpu内部最小的ram只是kb的量级），但如果要运行操作系统或软件的话，内部ram就不够了（memory不够），所以通常会外扩一个ram芯片（Memory Networks），这样cpu的运行并没有什么改变，只不过在需要memory来配合推理的时候，直接从Memory Networks中存取就可以了。

memory network在提出后又经历了几次变体，这次先写《memory networks》，这是memory networks被facebook首次提出的原始论文。后续又经历了几次变体。下一篇就会讲它end-to-end 的memory network，这是对于原始MemNN的一次改进。

《Memory Networks》

Memory Networks一共有5个部分，分别是memory m、I、G、O、R，其中

memory m：m由一系列的mi组成，其中mi是单个向量，每个mi都表示某一个方面的记忆存储。当要更新记忆的时候，就操作使用新记忆去更新相应的mi，要写入记忆时直接写入mi即可。至于i的范围则依据实际的使用策略不同而作相应的修改。

I：输入特征映射，它负责将输入转换成内部的特征表示形式。比如说输入的是文本，就需要它将其转换成model可以看懂的数学形式（比如说word转换成embedding形式、文本转换成向量等），也可以认为是数据的预处理的一个过程。用I(x)表示运算过程，其中x是输入。

G：更新记忆，根据当前的memory状态（m）、当前的输入（I(x)）、要更新的记忆（mi），得到更新后的记忆，并更新mi。用mi=G(mi, I(x), m)表示，其中i为m的数目范围内的任意数，具体怎么指定i的值，后续会解释。

O：输出映射，根据当前的记忆状态以及当前的输入，确定当前的输出，但输出是内部特征表示形式（比如说一个稠密的向量），因此还需要一个步骤作转换。用o=O(I(x), m)表示。

R：回应，负责将O步骤输出的o转换成系统交互的形式，例如文本或系统动作。用r=R(o)表示，最简单的理解就是把o输入一个rnn，使用seq2seq或者类似的结构输出对应的文本或动作即可。

一个文本的例子——最基本的模型应用

怎么把数据存到memory

最简单的一个方法就是顺序存，比如说定义有n个m，即mi中i的取值范围为0-999。那么每次要往memory存的时候，就找到下一个空着的memory，将输入x原封不动地放进去（这是最简单的策略，比如说x是个句子，就把句子直接放到memory中），之后N+1，下次就会往下一个memory存。

$$\mathbf{m}_N = x, N = N + 1.$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/15605145611.png)由于G是负责更新memory的，所以在这种策略中，G只是很简单的原封不动地输出x。

怎么提取指定记忆

那么存记忆已经搞定了，另一个主要的问题是取记忆，比如说下面这段小故事：

> 1. Joe去了厨房。
> 2. Fred去了厨房。
> 3. Joe拿起了牛奶。
> 4. Joe去了办公室。
> 5. Joe放下了牛奶。
> 6. Joe上了厕所。
>
> 提问：牛奶现在在哪里？

一共6句话，根据当前的存储策略，6句话直接以原始形式存在了m中，这个时候根据提问怎么知道哪句话是和提问相关的呢?

如果我们自己做这个题，就会把每个句子再看一遍，定位到了第五句——Joe放下了牛奶。memory network做的也是这件事情，它使用了O模块，去计算当前输入x和每一个m的相关程度，找到最相关的句子mi，公式如下:

$$o_1 = O_1(x, \mathbf{m}) = \arg\max_{i=1,\ldots,N} s_O(x, \mathbf{m}_i)$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/15605153261.png)

其中so是一类计算相关程度的函数，它最后会输出一个得分，x和所有的m遍历计算，直到找到得分最高的即为最相关的句子i。

> 其中score函数如下:
>
> $$s(x, y) = \Phi_x(x)^\top U^\top U \Phi_y(y).$$
>
> **(http://www.pkudodo.com/wp-content/uploads/2019/06/15605160191.png)**
> U的大小为[n，D]，Φx和Φy是D维向量，Φ(x)负责将x映射到内部向量表示形式（比如说最简单的就是词袋模型）。所以最后的s(x，y)的结果是一个实数，表示了分数。

但是这其中存在一个问题，比如说我们找到了最相关的i是5，放下了牛奶，但牛奶在哪里我们仍然不知道。所以对于人来说，会往前看一句，发现第4句"Joe去了办公室"直接暗示了牛奶在办公室。第4句和第5句结合才能得到牛奶在办公室的答案。

因此不光要找到最相关的句子，还应该找到第二相关的句子（paper中用k来定义一共找前k个相关的句子，这里k=2）。

$$o_2 = O_2(x, \mathbf{m}) = \arg\max_{i=1,\ldots,N} s_O([x, \mathbf{m}_{o_1}], \mathbf{m}_i)$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/15605155411.png)

值得注意的是，并不是简单地找到前2个相关句子，在找第2个句子时，需要以找到的第一个句子为条件。像上面的式子中，需要以x和m01为条件去找mo2。作者没有提到这么做的原因，但我认为是为了能够在有了第一个句子的情况下，再去找到一个另外的能够有最强代表性的句子，而不是说两个句子其实指代的是同一个方面，这样跟找一个句子也没什么差别。

输出

在这个例子中，输出并不要求是字符串，只要一个word即可，比如说"办公室"（这里引用的是原文中的例子，在英文中是单个的"office"，如果是中文情况下，可以加一个rnn来生成字符串）。

所以输出函数的输入是当前输入、m01，m02和字典，输出是字典当中最有可能的那个单词，公式如下:

$$r = \text{argmax}_{w \in W}\ s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], w)$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/15605159251.png)

训练

k=2时的目标函数如下：

$$\sum_{\bar{f} \neq \mathbf{m}_{o_1}} \max(0, \gamma - s_O(x, \mathbf{m}_{o_1}) + s_O(x, \bar{f})) + \tag{6}$$

$$\sum_{\bar{f}' \neq \mathbf{m}_{o_2}} \max(0, \gamma - s_O([x, \mathbf{m}_{o_1}], \mathbf{m}_{o_2}]) + s_O([x, \mathbf{m}_{o_1}], \bar{f}')) + \tag{7}$$

$$\sum_{\bar{r} \neq r} \max(0, \gamma - s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], r) + s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], \bar{r}])) \tag{8}$$

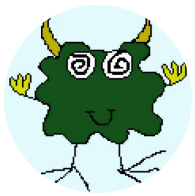
(http://www.pkudodo.com/wp-content/uploads/2019/06/15605163021.png)

（6）表示挑出来的句子是否是正确的句子，（7）表示在第一个句子被正确挑出来的情况下，第二个句子是否是正确的句子。（8）最后输出的是否是正确的答案。

可以看到loss其实对train的各个方面都做了一个约束，最后的performance效果也是非常好。

其他

论文的后半部分补充了很多小技巧，例如memory如果非常大的解决方法、遇到新词等。这里就不再过多展开了。



Dodo

9条评论

匿名 **发布于7:26 上午 - 1月 28, 2023**

To the pkudodo.com webmaster, Your posts are always well-cited and reliable.

()

回复

匿名 发布于9:55 下午 - 1月 9, 2023

()

This web site was... How would you say it? Relevant!! Last but not least I have found a thing that assisted me. Many thanks! sasilu.se/map17.php uv lampa 36w naglar

回复

匿名 发布于9:05 下午 - 1月 9, 2023

()

Thanks for ones wonderful submitting! I undoubtedly savored looking through it, you're an excellent creator.I will make sure you bookmark your web site and can usually come back extremely quickly. I want to stimulate you to carry on your fantastic creating, Have got a good holiday break weekend!

回复

匿名 发布于9:01 下午 - 1月 9, 2023

()

Pretty! This has long been an extremely superb submit. Numerous many thanks for furnishing this data.

回复

匿名 发布于3:19 上午 - 10月 17, 2022

()

An excellent share! I have just forwarded this onto a coworker who had been conducting a little bit homework on this. And he in actual fact bought me meal due to the fact that I stumbled upon it for him... lol. So let me reword this.... Thanks for the food!! But yeah, thanks for investing some time to take a look at this subject here on your website.

回复

匿名 发布于12:53 上午 - 9月 2, 2022

bunadisisj nsjjsjsisjsmizjzjjzjzz zumzsert

回复

()

匿名 **发布于4:51 下午 - 6月 1, 2021**

赘婿?

回复

()

匿名 **发布于6:02 下午 - 10月 4, 2020**

感谢博主！讲得很清晰！！

回复

()

匿名 **发布于10:47 上午 - 10月 15, 2019**

Теперь буду знать

回复

()

发表评论

内容(链接请去除http协议头，否则会被误判为垃圾评论)

设置显示昵称(选填)

发送信息

近期文章

- 论文|记忆网络之Memory Networks (http://www.pkudodo.com/2019/06/14/1-13/)
- 梳理|对话系统中的DST (http://www.pkudodo.com/2019/06/09/1-12/)
- 论文阅读|How Does Batch Normalizetion Help Optimization (http://www.pkudodo.com/2019/05/23/1-11/)
- 学习规划|机器学习和NLP入门规划 (http://www.pkudodo.com/2019/03/20/1-10/)
- 面试体会|微软、头条、滴滴、爱奇艺NLP面试感想 (http://www.pkudodo.com/2019/03/10/1-9/)

文章归档

- 2019年6月 (http://www.pkudodo.com/2019/06/)
- 2019年5月 (http://www.pkudodo.com/2019/05/)
- 2019年3月 (http://www.pkudodo.com/2019/03/)
- 2018年12月 (http://www.pkudodo.com/2018/12/)
- 2018年11月 (http://www.pkudodo.com/2018/11/)
- 2018年10月 (http://www.pkudodo.com/2018/10/)
- 2016年9月 (http://www.pkudodo.com/2016/09/)

标签

DST (http://www.pkudodo.com/tag/dst/)

K近邻 (http://www.pkudodo.com/tag/k%e8%bf%91%e9%82%bb/)

LSI (http://www.pkudodo.com/tag/lsi/)

memory network (http://www.pkudodo.com/tag/memory-network/)

MQTT (http://www.pkudodo.com/tag/mqtt/)

NLP (http://www.pkudodo.com/tag/nlp/)

s3c2440 (http://www.pkudodo.com/tag/s3c2440/)

SLU (http://www.pkudodo.com/tag/slu/)

SVM (http://www.pkudodo.com/tag/svm/)

VSM (http://www.pkudodo.com/tag/vsm/)

体会 (http://www.pkudodo.com/tag/%e4%bd%93%e4%bc%9a/)

决策树 (http://www.pkudodo.com/tag/%e5%86%b3%e7%ad%96%e6%a0%91/)

分类器 (http://www.pkudodo.com/tag/%e5%88%86%e7%b1%bb%e5%99%a8/)

实现 (http://www.pkudodo.com/tag/%e5%ae%9e%e7%8e%b0/)

对话系统 (http://www.pkudodo.com/tag/%e5%af%b9%e8%af%9d%e7%b3%bb%e7%bb%9f/)

感想 (http://www.pkudodo.com/tag/%e6%84%9f%e6%83%b3/)

感知机 (http://www.pkudodo.com/tag/%e6%84%9f%e7%9f%a5%e6%9c%ba/)

支持向量机 (http://www.pkudodo.com/tag/%e6%94%af%e6%8c%81%e5%90%91%e9%87%8f%e6%9c%ba/)

文章相似度 (http://www.pkudodo.com/tag/%e6%96%87%e7%ab%a0%e7%9b%b8%e4%bc%bc%e5%ba%a6/)

最大熵 (http://www.pkudodo.com/tag/%e6%9c%80%e5%a4%a7%e7%86%b5/)

朴素贝叶斯 (http://www.pkudodo.com/tag/%e6%9c%b4%e7%b4%a0%e8%b4%9d%e5%8f%b6%e6%96%af/)

机器学习 (http://www.pkudodo.com/tag/%e6%9c%ba%e5%99%a8%e5%ad%a6%e4%b9%a0/)

爬虫 (http://www.pkudodo.com/tag/%e7%88%ac%e8%99%ab/)

统计学习方法 (http://www.pkudodo.com/tag/%e7%bb%9f%e8%ae%a1%e5%ad%a6%e4%b9%a0%e6%96%b9%e

统计方法学习 (http://www.pkudodo.com/tag/%e7%bb%9f%e8%ae%a1%e6%96%b9%e6%b3%95%e5%ad%a6%e

综述 (http://www.pkudodo.com/tag/%e7%bb%bc%e8%bf%b0/)

网易云歌单 (http://www.pkudodo.com/tag/%e7%bd%91%e6%98%93%e4%ba%91%e6%ad%8c%e5%8d%95/)

规划 (http://www.pkudodo.com/tag/%e8%a7%84%e5%88%92/)

详解 (http://www.pkudodo.com/tag/%e8%af%a6%e8%a7%a3/)

逻辑回归 (http://www.pkudodo.com/tag/%e9%80%bb%e8%be%91%e5%9b%9e%e5%bd%92/)

逻辑斯蒂回归 (http://www.pkudodo.com/tag/%e9%80%bb%e8%be%91%e6%96%af%e8%92%82%e5%9b%9e%e

面试 (http://www.pkudodo.com/tag/%e9%9d%a2%e8%af%95/)