mp.weixin.qq.com

【NLP笔记】文本生成？还不快上知识库

本文本来是写ACL2020一篇对话系统论文《Dynamic Fusion Network for Multi-Domain End-to-end Task-Oriented Dialog》笔记，发现需要太多的前序知识，于是顺便按时间顺序整理了一下记忆网络和知识库任务导向的几篇经典论文。

【知识库导向】

知识库任务导向可以融合进很多任务中，如ERNIE就是在bert的基础上利用了知识库；之前在一个信息抽取的比赛任务冠军方案中，看到加入知识库也是一个上分点，在抽取人物或者作品的时候，知识库能帮你确认这是什么类型；另外，比如知识库在对话文本生成的场景中，如果不结合外部知识库，生成的任务语义都只能源于训练数据，无法获得一些常识性知识，如：

”我想喝星巴克“

”附近没有星巴克呢，有其他咖啡店请问你需要么“

如果你的训练数据中没有”星巴克“是”咖啡店“的知识，那捕获不到这种语义，而依赖知识库，相当于给一个已经十岁的孩子再教育，比刚生出来开始教爬容易多了。

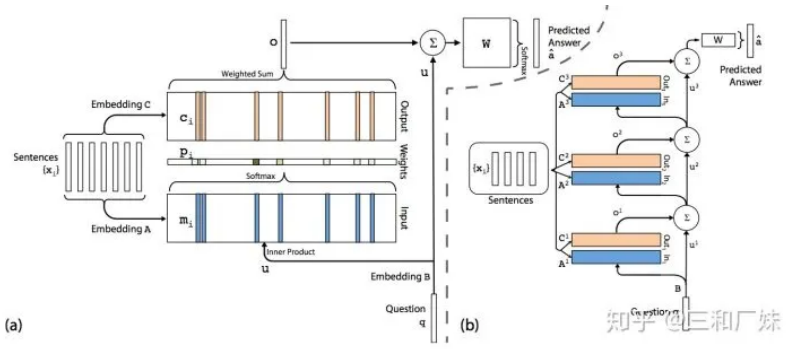
【记忆网络】

MemNN也是NLP的一个分支，它最大的特点是不像普通的编码结构如LSTM和CNN之类的会对信息压缩成hidden state，从hidden state中抽取特征，这类方法产生的记忆太小了，在压缩过程中损失了很多有用信息，MemNN是将所有的信息存在一个外部memory中，和inference一起联合训练，得到一个能够存储和更新的长期记忆模块，最大限度的保存有用信息。

通过几篇论文具体了解一下：

1. MemNN记忆网络原理简介

MemNN主要包括2种操作：



embedding：sentence经embedding过程产生2个矩阵：input(图中 embedding A) 和output matrix （embedding B）

inference: 计算得到和上述内部向量之间的相关性，具体是三个步骤。

input memory representation过程会计算问题向量Embedding B 和input matrix (Embedding A)的点积后归一化，得到和input matrix维度一致的概率向量p，即问题和各记忆向量的相关程度；

output memory representation过程将output matrix(Embedding c)按概率向量p进行加权求和，得到输出向量o，相当于选取了相关性最高的记忆向量组合；

output calculation是将输出向量转化为所需答案的格式，得到各单词相对答案的概率，运算是全连接型的矩阵乘累加；

inference是通过多层神经网络来推断语句和问题的相关性(图中右边的多个叠加)

MemNN 存在的问题

MemNN对输入内容的保存没有经过大幅度的压缩(参数都是embedding)，信息完整性很高，这样在问答推理上相比RNN等压缩模型很有优势，不过带来的问题就是存储空间会随着内容的增大而线性增加，内存带宽需求的增加。

由于MemNN计算的特点是从sentence生成的多个向量中选择相关性最大的产生答案，因此中间结果矩阵会是一个很稀疏的矩阵，只有相关性较强的部分才有值，其他不相关的几乎都是0，因此密集运算加速器（如GPU等）效果就不好了，需要软件和硬件着重考虑如何进行稀疏性的优化。

2. KV-MemNN 字典型记忆网络

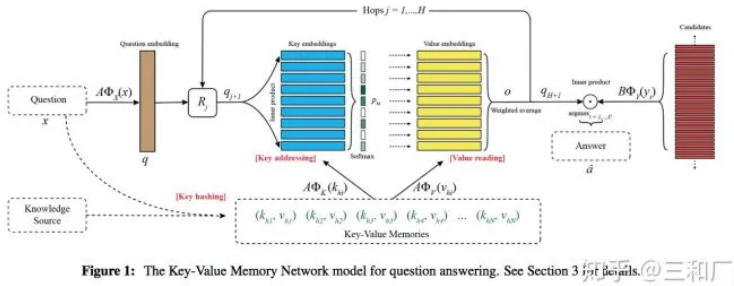


Figure 1: The Key-Value Memory Network model for question answering. See Section 3 for details.

在MemN2N中把context线性变换成了一个整体的embedding作为memory，而KV-MemNN的不同之处在于引入了外部知识源，将其中的memory变成了(key, value)键值对。

主要包括下面三个步骤：

Key Hashing：使用的是倒排索引的方法，选择一个大小为N的k-v对集合，从知识库中选出潜在的候选记忆，在消除停用词的情况下保证key对应的单词在query中出现；

Key Addressing：阶段主要是利用Hashing的结果（候选记忆）去和query经过线性变换后的结果计算一个相关概率（relevance probability）, 与MemNN中 的inner product 类似

$$p_{h_i} = Softmax(A\phi_X(x) \cdot A\phi_K(k_{h_i}))$$

x 其中 是query， 是特征筛选器， 是矩阵, 初始时

ϕ

$$A\phi_X(x) = R_1$$

A

value reading: key被设计的和query相关(图中蓝色矩阵)，value被设计的和answer相关（图中黄色部分），所以使用Addressing得到的probability和kv中的v进行weight sum操作

$$o = \sum_i p_{h_i} A\phi_V(v_{h_i})$$

表示即根据query的偏重注意力从中读取出来知识源中有价值的记忆，

q 记忆会多跳循环更新，虽然这部是无监督的，但是多跳更新直观上与推理能对应上 的多跳更新

$$q = A\phi_X(x), q_2 = R_1(q+o), q_3 = R_2(q_2+o)\dots$$

p 的更新

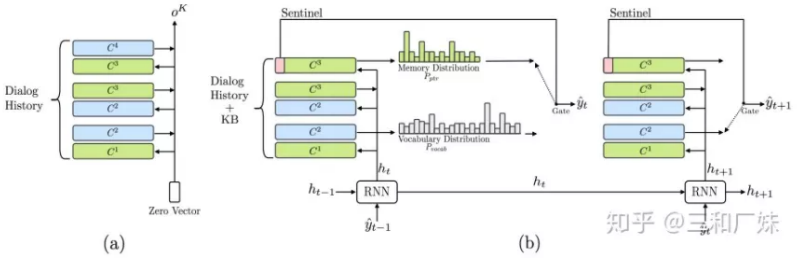
$$p_{h_i} = Softmax(q_{j+1}^T A\phi_K(k_{h_i}))$$

最后label与model的输出点乘后softmax后求loss

$$a = Softmax(q_{H+1}^T B\phi_Y(y_i))$$

文中作者还尝试了很多K-V的表示形式， 感兴趣大家可以读原文;

3. Mem2seq 记忆网络文本生成



Mem2Seq是一个使用带有指针网络思想的多跳注意力机制的生成模型，这种方法有效的结合了KB的信息且Mem2Seq学习如何生成动态的查询来控制memory的访问。它与KV-MemNN的区别在于知识源的表达形式以及它是在应用在seq的生成中，decoder的每一步都用到了记忆并对记忆更新。

C^K Encoder: 上图中a部分是encoder的核心， 表示第k跳记忆，query要经过K-hop的更新，类似从历史对话中找到这是真正的query是什么，从上面几种方案中可以看出，这个也是memNet的常规操作；

P_{vocab} Decoder: 上图b部分描述解码，每一个时刻，将产生两个分布：词表分布 和记忆部分分布 记忆部分分布是指对话历史以及KB的信息。

计算如下：

P_{ptr} 上式表明，当生成的词与记忆存储器中的词相等时，使用记忆存储器的词，即完成复制功能，当生成的词不在记忆存储器中时， 指向一个特殊字符，模型将使用词表分布 来生成输出。

$$ptr_i = \begin{cases} max(z), if \exists z.s.t.\ y_i = u_z \\ n+l+1, otherwise \end{cases}$$

4. GLMP-任务型对话中全局到局部的记忆指针网络

GLMP是最接近我们要看的这篇论文的内容了，仔细介绍一下：

1. 解决的问题是如何有效地在任务型对话系统中嵌入知识库

p_{vocab} p_{ptr}

- Mem2Seq的改进在于将decoder变成了PointNetwork，将copy以及生成思想和记忆网络结合在一起，有效的实现了任务型对话知识库嵌入；
- 动态的大量的知识库的嵌入无疑相当于对模型引入一个巨量噪声，而且加大模型计算等方面的开销(知识库难于编码以及解码)，为了有效地在任务型对话系统中嵌入知识库，原文提出了全局到局部的记忆指针网络(GLOBAL-TO-LOCAL MEMORY POINTER NETWORKS，GLMP, 全局到局部的记忆指针网络)；

2. GLMP结构

【Encoder部分】

Global Memory Encoder编码对话历史，输出的两个量：全局上下文表示和全局记忆指针

全局上下文表征

$H = (h_1^e, \dots h_i^e)$ 编码上下文 使用了一个context rnn(其实就是双向GRU)来编码用户的一句话，得到每一个时间步

- 编码记忆 每一hop的可训练嵌入矩阵与Mem2Seq相似

$C = (C^1, \dots C^{K+1})$ • 上下文与记忆的链接 为了克服MN的弊端--建模记忆之间相关性比较困难，因此将得到的隐状态加到dialogue memory representation中，即

$$c_i^k = c_i^k + h_{m_i}^e, where\ m_i \in X\ \forall k \in [1, K+1]$$

$M = [B; X] = (m_1, \dots m_{n+l})$ encoder端的输入为： ， 其中 表示的是上文提到的三元组；（B是历史对话信息，X是三元组，m对话也转成3元 组信息， 统一表达形式，n+l外部三元组和历史对话拼接起来的字数）

m_i 终于得到了全局上下文的表达为：

$$p_i^k = Softmax((q^k)^T c_i^k)$$

$$o^k = \sum_i p_i^k c_i^{k+1}, q^{k+1} = q^k + o^k$$

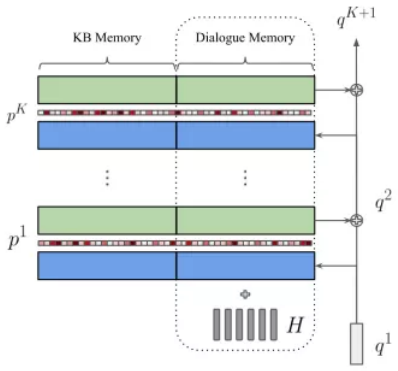
q^{k+1} 全局上下文表征 的是（ 双向GRU最后隐状态)经过K-hop迭代产生的)

2. 全局记忆指针(Global Memory Pointer)

q^1 全局记忆指针用来过滤其余知识库的噪声

h_n^e • 首先使用编码器的最后一个隐层状态 查询外部知识直到最后一跳（只是最后一条计算吧），做内积相似度计算，执行Sigmoid函数(0-1的值)，最后获得的memory分布即为全局内存指针G，最终它被传递给解码器使用。

训练全局记忆指针的生成需要添加额外的辅助任务，使用标签为 来检查 中的 Object词是否在对应的真实响应 中出现，如果出现则为1，没有为0。



$$g_i = Sigmoid(q^K c_i^K) \qquad g_i^l = \begin{cases} 1 & if(Object(m_i) \in Y \\ 0 & otherwise \end{cases}$$

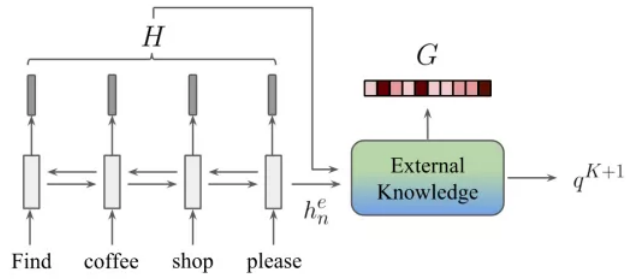
最终交叉熵loss为：

$$Loss_g = -\sum_{i=1}^{n+l}[g_i^l log(g_i) + (1-g_i^l) log(1-g_i)]$$

因为Sigmoid是一个二分类的函数，非真即假，添加额外的辅助任务训练全局记忆指针就是为了过滤知识库，保留有用的知识传递给decoder来实例化slot， 那就得到了一个全局记忆的指针；

h_n^e 【decoder部分】

- $G^{label} = (g_1^l, \dots g_{n+l}^l)$ decoder使用了粗略空槽的RNN(sketch rnn)，先用一个占位符占着， 然后利用全局记忆指针过滤掉的外部知识库来寻找槽位具体相关信息 ， 最后使用局部记忆指针(local memory point)来实例化未填充的slot值，这个slot的值可能是知识库中的信息或者生成的内容。
  - m_i 首先有产生一个粗糙(sketch)带有未填充slot值(但是有slot标签)的响应 , sketch RNN 是一个单层的GRU，但是它的生成单词表中有Sketch Tag，例如 会产生“@poi is @distance away”来替代“Starbucks is 1 mile away.”
  - Y 在每一个时间步，sketch RNN 的隐状态有两个作用：



1. 如果判断结果是生成而非copy, 那h的作用预测(生成)下一个词（就是decoder的用法）,使用 (t时间步d维的隐状态）的计算公式为生成下一个词可以表示为：

h_t^d

$$h_t^d = GRU(C^1(\tilde{y}_{t-1}^d), h_{t-1}^d) \qquad p_t^{vocab} = Softmax(W h_t^d)$$

$$y_t^s = argmax(P^{vocab})$$

loss表示为

$$Loss_v = \sum_{t=1}^{m} -log(P_t^{vocab}(y_t^s))$$

2. 作为外部知识库的查询向量，当生成的结果是Tag时，之前编码的全局记忆指针就会递给外部知识库，来确定这个tag填什么，起到过滤外部知识库作用；

h_t^d 作为查询向量来与过滤后的外部知识库做PointNetwork，产生的分布就是局部记忆指针(Local memory pointer，L)

计算如下（copy point 原理部分，有生成目标就生成，没有就copy）：

$$L_t^{label} = \begin{cases} max(z) & if \exists zs.ty_t = Object(m_z) \\ n+l+1 & otherwise \end{cases}$$

$$Loss = \sum_{t-1}^{m} -log(L_t(L_t^{label}))$$

用几句话描述GLMP

引入了知识库和记忆网络结合

decoder引入了pointer Net

encoder 编码记忆网络和上下文（包括知识库实体)并生成全局的关注点指针（用于过滤噪声），decoder部分借鉴pointerNet， 用隐状态生成局部指针来确定copy时指向那个实体
DFU：多领域端到端任务导向的动态融合对话网络

终于到了这篇Dynamic Fusion Network for Multi-Domain End-to-end Task-Oriented Dialog 主角了，其实有了GLMP的前序知识，这篇看起来就简单多了。

相对GLMP这篇文章要解决的主要问题是：在任务型对话中如何快速在不同领域迁移学习。任务型对话是领域强相关的，不同领域数据和模型相差巨大，作者设计了一个基于GLMP的架构DF-Net，既能自动学习到不同领域的相关性又能学到各个域特有的知识。

加强版的encoder和decoder模块

H_{enc}^f 在GLMP的基础上，encoder和decoder 将混合域隐节点和特有域隐节点的拼接后融合成 ，其中 指shared， 指 domain-specific

s

d

$$(H_{enc}^s, H_{enc}^d) \rightarrow H_{enc}^f$$

$$(h_{dec,t}^s, h_{dec,t}^d) \rightarrow h_{dec,t}^f$$

具体加强方式

$$H_{enc}^{f} = W_2(LeakyReLU(W_1[H_{enc}^{s}, H_{enc}^{d}]))$$

动态融合

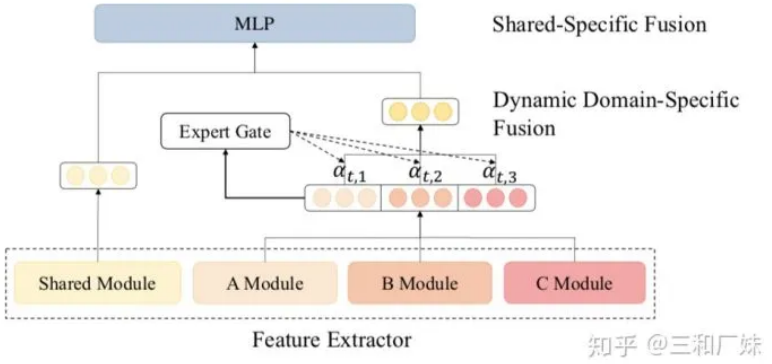
作者认为加强版的encoder和decoder即使融合进了不同的域的信息，但是忽略了不同域的细粒度相关性。因此设计动态融合的架构是：

各个域数据首先各自先GLMP，从而获得所有域中各自域的特定特征

私有特征都由动态domain-specific特征融合模块进行融合， 作者借鉴了Mixture-of-Experts mechanism (MoE) 混合专家机制，可以看出预测属于某私域的概率分布，实质上看起来就是一个加了一个辅助任务来计算每个域的attention

$$\alpha_t = Softmax(W * h_{dec,t} + b)$$

$$h_{dec,t}^{df} = \sum_i \alpha_{t,i}, h_{dec,t}^{d_i}$$



共享特征由动态shared特征融合模块融合 共享特征的融合就是指将原始encoder和decoder替换为加强版

对抗学习

作者最后为了更好的训练对模型做了一些调整，引入了对抗学习来更好的学习域之间共有的特征

引入了梯度反转层

$L = \gamma_b L_{basic} + \gamma_m L_{moe} + \gamma_a L_{adv}$ 最后的loss为 ，看过GLMP我们也知道 也是多个混合loss组成， 是在混合loss的基础上再混合，真是一个各种混合的任务，但是，实验结果非常强，在少量数据下，模型的trans能力比之前最好的高了十几个点

L_{basic}

L

	SMD					Multi-WOZ 2.1				
Model	BLEU	F1	Navigate F1	Weather F1	Calendar F1	BLEU	F1	Restaurant F1	Attraction F1	Hotel F1
Mem2Seq (Madotto et al., 2018)	12.6	33.4	20.0	32.8	49.3	6.6	21.62	22.4	22.0	21.0
DSR (Wen et al., 2018)	12.7	51.9	52.0	50.4	52.1	9.1	30.0	33.4	28.0	27.1
KB-retriever (Qin et al., 2019b)	13.9	53.7	54.5	52.2	55.6	-	-	-	-	-
GLMP (Wu et al., 2019a)	13.9	60.7	54.6	56.5	72.5	6.9	32.4	38.4	24.4	28.1
Shared-Private framework (Ours)	13.6	61.7	56.3	56.5	72.8	6.6	33.8	39.6	26.6	29.3
Dynamic Fusion framework (Ours)	**14.4***	**62.7***	**57.9***	**57.6***	**73.1***	**9.4***	**35.1***	**40.0***	**28.1***	**30.0***