## 深度学习

自2015年开始，**深度学习**用于KB-QA的方法取得了很好的效果，并逐渐成为了KB-QA的主流方法。也涌现出了一批使用深度学习提升传统的语义解析、信息抽取和向量建模方法的优秀论文。本期，我们先以深度学习提升向量建模方法为例，作为深度学习篇的上篇，为大家进一步揭开知识库问答的面纱。

我们在揭开知识库问答KB-QA的面纱4·向量建模篇介绍了KB-QA（强烈建议在阅读本文章之前阅读该文章）中介绍了传统方法之一的向量建模（Vector Modeling）方法，该方法具有操作性强，不需要任何手工的特征等优点。今天，我们将介绍一篇利用深度学习对该向量建模方法进行提升的论文，来自北京航空航天大学和Microsoft研究院的 Question Answering over Freebase with Multi-Column Convolutional Neural Networks（文章发表于2015年的ACL会议）。

该文章使用了卷积神经网络的一种变体（作者称为multi-column）从三个方面（**答案路径Answer Path，答案上下文信息Answer Context，答案的类型Answer Type**）对问题和答案的分布式表达进行学习，使得该分布式表达相比之前的向量建模方法能够包含更多有效的特征。该方法在WebQuestion数据集上测试，取得了40.8的F1-score。

向量建模方法的再思考

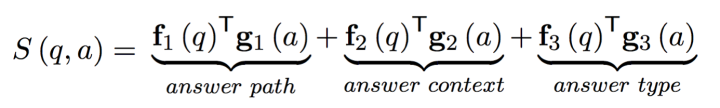
在上期的向量建模方法中，我们谈到该方法本身存在一些缺陷。

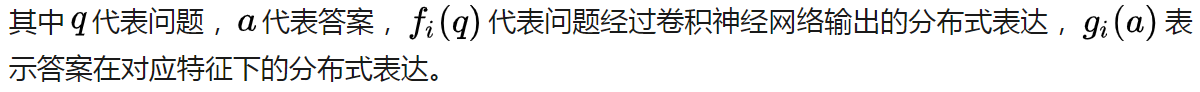
首先是对于**问题的向量化**。对问题的向量化，传统向量建模方法采用了类似词袋模型的方式，相当于它并未考虑问题的语言顺序（比如 “*谢霆锋的爸爸是谁？” 谢霆锋是谁的爸爸？*这两个问题用该方法得到的表达是一样的，然而这两个问题的意思显然是不同的）。

对于这个缺陷，我们可以使用深度学习的模型对问题进行向量化，比如使用循环神经网络（Recurrent Nerual Networks, RNNs）、卷积神经网络（Counvoulutional Nerual Networks, CNNs ）等模型提取问题特征，这样的方式考虑了语言的顺序，并且提取特征的能力也更加强大。

第二个缺陷是，在对答案进行向量化的时候，直接将答案的路径（问题主题词到答案实体的路径）和上下文信息（答案实体周围的知识库子图）一起作为答案特征，通过multi-hot的方式对答案进行向量化。事实上，这样的形式不利于模型区分答案的特征（仅仅根据答案的multi-hot向量是不好区分哪些是答案的类型，哪些来自答案的上下文，哪些来自问题主题词到答案实体的路径）。

因此我们可以将问题的特征表示拆解开，用三个向量分别表示答案的三个特征，即**答案路径（Answer Path）**，**答案上下文信息（Answer Context）**，**答案类型（Answer Type）**，对于每一个答案特征向量，都用一个卷积网络去对问题进行特征提取，将提取出的分布式表达和该答案对应特征向量的分布式表达进行点乘，这样我们就可以得到一个包含三部分的得分函数：



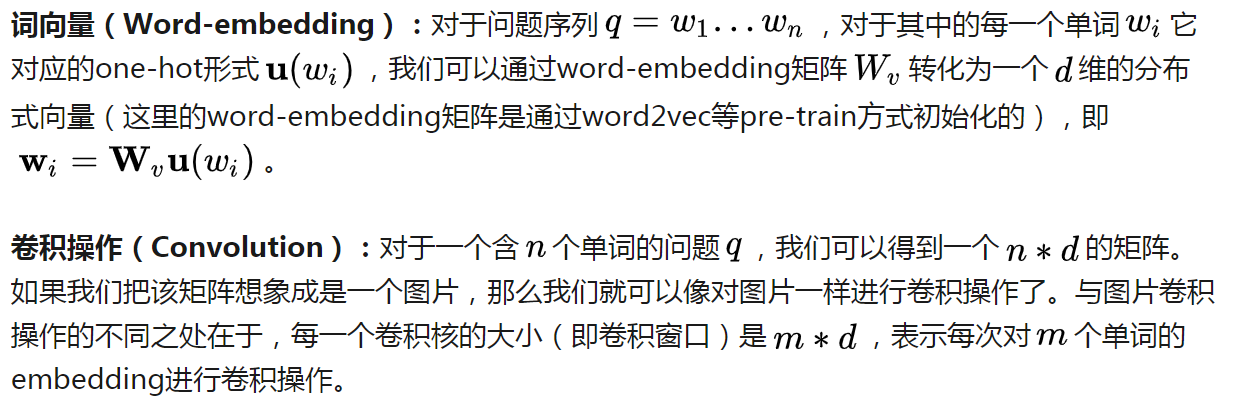
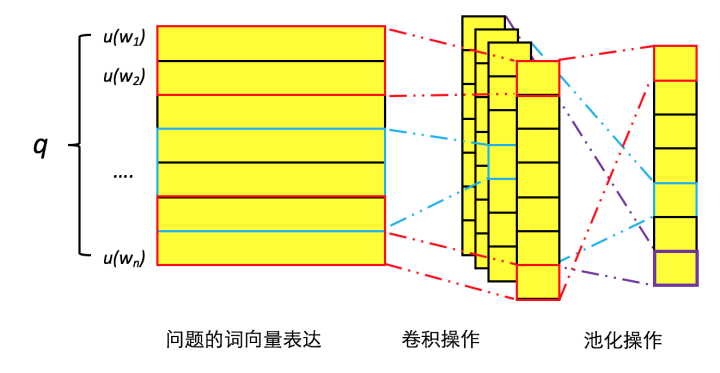
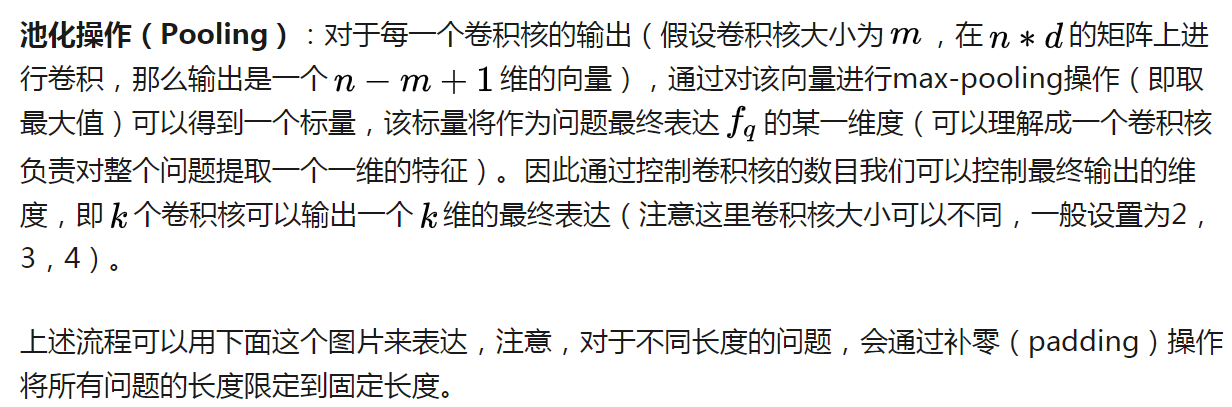
有了得分函数，我们就可以像向量建模方法一样，通过定义margin-based ranking损失函数对模型参数进行训练。



Multi-Column卷积神经网络

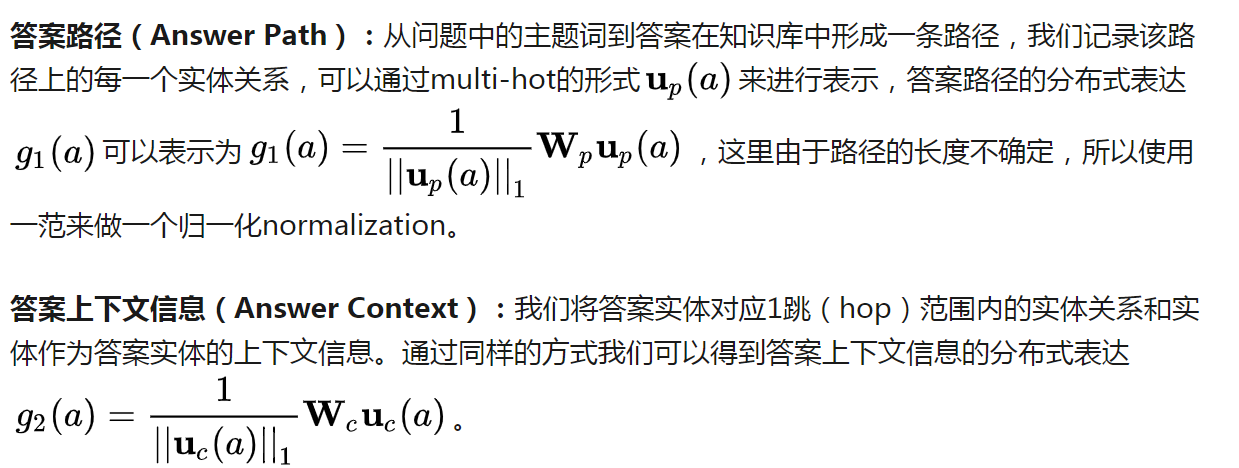
对于问题的特征提取，作者使用Multi-Column卷积神经网络，其结构实质上是共享word-embedding层的三个text-CNNs，text-CNNs模型在文本分类问题上取得了很好的效果。（这两个概念是Deep Learning for NLP中的重要概念，我在这里只作简单介绍，不再展开深入讨论，想深入了解的朋友可以点击进入查看相应的论文）。

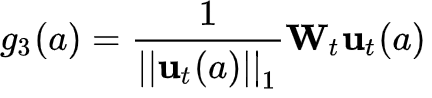
（如果对text-CNNs了解的朋友可以跳过下面这一段）

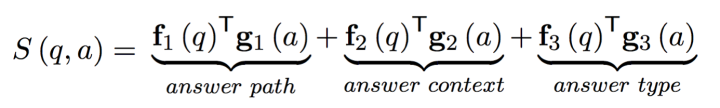
这样，我们通过三个text-CNNs，在共享word-embedding的情况下，就可以得到f(q1),f(q2)和f(q3)。（事实上，在这篇文章中所使用的卷积操作，对于每一个column只采用了一个卷积核，一个卷积核对一个卷积窗口的卷积结果并非一个值而是一个向量，max-pooling作用在每一个卷积窗口的卷积结果上，具体方式可以参看后面的图。个人认为这样的卷积方式减少了参数，显得特征提取更加粗粒度，效果很可能不如text-CNNs）

接下来，我们用三个向量来分别表示答案的三种特征。

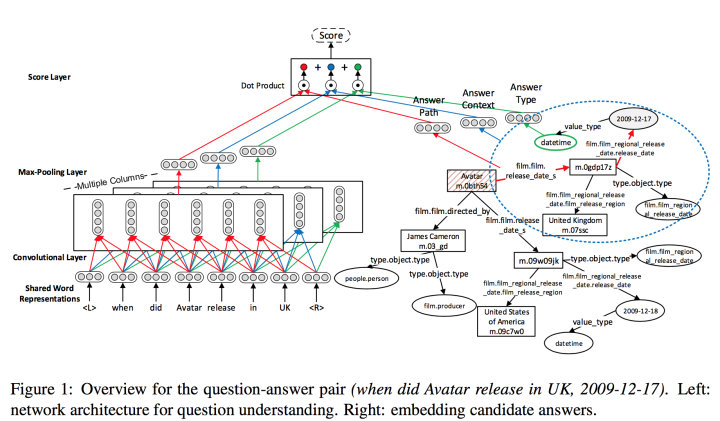


答案类型（Answer Type）：在信息抽取中我们提到，答案的类型是一个很重要的特征。类型是一种特殊的实体关系，比如时间2009-12-17 的类型是 datetime，James Cameron 的类型是people.person 和 film.producer。 在实际操作中，可以在freebase里通过实体关系common.topic.notable.types 来查询实体对应的所有类型。通过同样的方式，我们可以得到相应的分布式表达。注意，如果候选答案是一个值，那么就用该值的类型（string/float/datetime）作为答案类型，比如答案是2009-12-17，那么类型就是string。

至此，我们得到了包含个三部分的得分函数：



整个流程如下图所示：



（图中方块带红色斜线的为主题词，红色箭头表示路径，绿色椭圆表示答案类型，蓝色虚线椭圆表示上下文信息范围）

对于问题*“when did Avatar release in UK”*和它的答案*2009-12-17，*我们通过multi-column卷积神经网络提取三种问题的分布式表达，再通过答案的路径、上下文信息和类型得到答案相应的三种分布式表达，通过分别点乘再求和的方式得到最终的答案-问题对得分。

我们可以通过我在揭开知识库问答KB-QA的面纱4·向量建模篇介绍了KB-QA提到的同样的方式构造损失函数和多任务学习来训练模型参数。

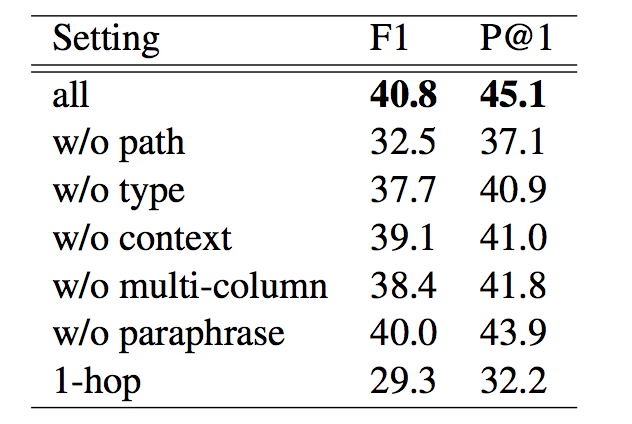
实验分析与总结

模型的参数这里不再赘述。值得一提的是这篇文章采用的关于候选答案的寻找方式。和向量建模方法类似，也是寻找问题中主题词n跳范围内（一般取两跳）的实体作为候选答案集合。然而对于有些问题，其候选答案集合很大，因此我们采取一种启发式方法，如果同一路径候选答案数超过200，则随机选择200个作为候选。如果某个候选答案得分最高，那么我们把它所在的答案路径中的所有其他节点（不在200个候选集合的节点）也计算得分，选出得分最高的作为最终答案。

实验依旧是在benchmark——WebQuestion上进行，取得了40.8的F1-score，这个性能击败了几乎当时所有的传统方法。并且我个人认为如果引入现在的一些深度学习技巧（诸如batch normalization、Adam等）并且加大模型的参数数量，改进word-embedding预训练等，效果还有上升的空间。我相信该方法如果面对**更大规模的数据集**会比传统方法取得更好的效果。

由于这篇文章考虑了答案的三个因素（路径、上下文、类型），为了探究哪个因素对最终效果影响最大。作者通过测试 在得分函数中去除某一因素、只使用single column、不进行多任务训练、只使用1 hop范围作为候选答案（该顺序对应下表的顺序）等情况下的模型性能，来对模型进行分析。

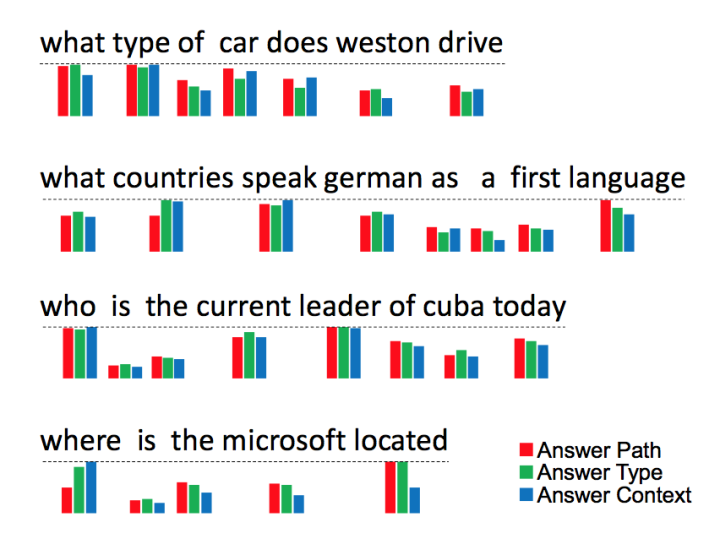
那么你猜猜对于答案的路径、上下文、类型这三个因素，去除哪个因素之后对性能影响最大呢？实验结果如下表



其中w/o表示without，可以看出答案的路径对效果影响最大（问题主题词到答案实体的路径其实相当于问题到答案的推理过程），其次是答案的类型，影响最小的是答案的上下文信息（仔细想想，其实答案实体的上下文信息很多对寻找该问题的答案没有帮助，比如询问奥巴马的出生地，那么奥巴马的其他很多信息如社会关系、职位等等对答案没有帮助，相当于噪音）。

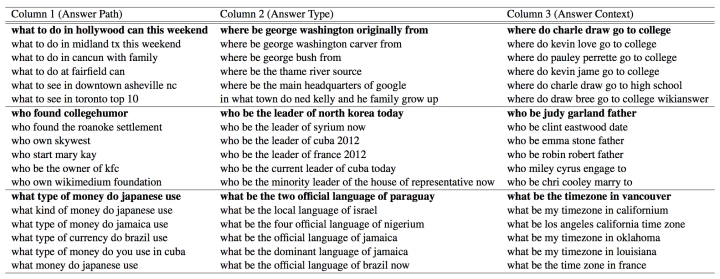
分布式表达，虽然没有传统的信息抽取和语义解析那么具有解释性，但是，我们却可以通过衡量分布式表达的余弦（Cosine）距离和通过得分函数来观察到一些可解释性。为此，作者还进行了两个实验。

首先，我们想知道对于一个问题，我们的模型最关注这个问题的哪个单词，即显著性单词检测（Salient words detection） 。我们可以进行这样的实验，要检测问题中的某一个单词的显著性，我们可以将该单词替换为某些停顿词（如to is a等），计算替换了单词之后的表达，再计算该表达和原始表达的欧式距离（可以多替换几种停顿词取平均距离）作为显著性得分，这里给出了一些例子：



我们可以看出来，**问题词**（wh-）、问题词**相关的名词**和**动词**都对找出最终答案有重要影响。

我们也可以通过问题表达之间的cosine距离来寻找**相似的问题**，如下表：



我们通过计算问题（字体加粗）的表达，通过cosine距离寻找最近的问题表达所对应的问题，可以看出这些问题在语义上还是很相似的。如相同语义*“who found/own/start \*”* 和 *“who be the owner of \*”*的距离很接近*。*

最后，作者也列举了一些**错误分析**（通过对错误情况进行分析，对我们提出改进的方法很有帮助），错误主要有以下三种情况：

**候选答案生成**：有些问题的主题词是难以正确提取出来的，比如缩写词和表达不全，如问题“where did jfk and his wife live*”*，很难将jfk这个缩写词对应到*John F. Kennedy*这个人名上，这样会导致我们无法得到正确的候选答案集合。要解决这种情况，可能需要对问题进行一些预处理。

**问题歧义：**对于数据集中有些有歧义的问题，难以获得和正确答案相应的关系，如问题*“who is aidan quinn”*，答案是演员，我们很难通过该问题*who is*推断出和职业相关。这种情况该怎么办呢？

**时序敏感（Time-Aware）问题：**对于问题中带有 first / second 这种与时间顺序相关的词语，如*“who is johnny cash’s****first****wife”*，答案可能给出的是second wife的名字（模型只关注到了wife而忽略了first的含义，并没有进行额外的推理）。对于这种情况，可能需要定义专门（ad-hoc）的操作，注意的是，这一点是该类方法相比**语义解析**方法的一个缺点。

那么，下一期，我们就将介绍深度学习对语义解析方法的提升，我们将以Micorsoft ACL 2015年的Outstanding paper为例，为大家带来揭开知识库问答KB-QA的面纱6·深度学习中篇，进一步揭开KB-QA的面纱。

在上期，我们介绍了深度学习对传统向量建模KB-QA方法进行提升的一篇代表论文，可以看出它的效果击败了当时所有的传统方法。本期，我们将以**深度学习提升语义解析**方法的一篇代表作为例，作为深度学习篇的中篇，为大家进一步揭开知识库问答的面纱。

我们在揭开知识库问答KB-QA的面纱2·语义解析篇中介绍了传统方法之一的语义解析（Semantic Parsing）方法，该方法相比向量建模方法有更强的解释性，具有一定的推理能力。今天，我们将介绍一篇利用深度学习对该语义解析方法进行提升的论文，来自Microsoft公司的Semantic Parsing via Staged Query Graph Generation:Question Answering with Knowledge Base（文章发表于2015年的ACL会议，是当年的Outstanding Paper）。

该文章分析了传统语义解析方法的不足，受信息抽取和向量建模方法的启发，将语义解析过程转化成查询图（Query graph）分阶段生成的过程，使用了卷积神经网络来提升自然语言到知识库关系的映射。该方法在WebQuestion数据集上测试，取得了**52.5的F1-score**，该性能远超当时的所有方法。

语义解析方法的再思考

让我们先回想一下传统的语义解析方法，它的思想是把自然语言问题转化为逻辑形式，通过逻辑形式转化为查询语句，在知识库中查询得出最终答案。在进行语义解析生成逻辑形式的过程中，主要是在提取自然语言问题中的信息和利用训练好的语法解析器进行解析，这一过程**几乎没有使用到知识库里的信息**。而在向量建模和信息抽取方法中，我们不仅对问题进行了特征提取，还借助知识库确定了候选答案范围（相比语义解析中的词汇映射要在大范围的知识库实体关系中寻找映射，这样的方式使得搜索范围大大减小），并将候选答案在知识库中的信息作为特征。相比之下，可以看出传统的语义解析方法和知识库本身的联系是**不够紧密**的（Decoupled from KB），也就是说，**传统语义解析方法对知识库的利用还不够**。

再看看语义解析的第一步，词汇映射（Lexicon）。要将自然语言中的谓语关系映射到知识库中的实体关系，是一件很困难的事情，仅仅通过**统计方式**进行映射，效果并不好。如果我们能考虑知识库的信息，是不是能将词汇映射的范围缩小？使用深度学习的办法通过分布式表达来代替基于统计方法的词汇映射，会不会取得更好的效果？

在语义解析的过程中，如何更好的去利用知识库的知识，缩小语义解析树的搜索范围，并获取更多有益的特征信息？就让我们带着疑问，看一下本文的作者是如何解决这些问题的。

什么是查询图

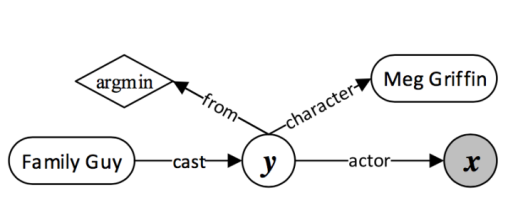
我们来考虑这样一个问句*“Who first voiced Meg on Family Guy?"*(谁是第一个为Family Guy里的MegGriffin角色配音的人，*注：Family Guy是美国的一部动画片，MegGriffin是其中的一个角色，有两个人先后为其配音过*)

可以看出，这个问题是有一定难度的，我们在前一期谈到，对于深度学习的向量建模法来说，*first*这种时序敏感（Time-Aware）词常常会被模型忽略而得出错误答案。语义解析方法可以将first解析为逻辑形式的聚合函数（*arg min*），但它又难以将问题中的*Meg*这一缩写词通过词汇表映射为知识库中的*MegGriffin*。

想一想我们人如果给定知识库会怎么去寻找答案？首先我们也许不知道Meg具体是指哪个角色，但是我们可以先去知识库里搜Family Guy，在它对应的知识库子图中搜索和Meg很接近的实体，也就是说我们一开始就借助知识库，帮我们缩小了范围，这样我们就很容易找到Meg其实对应的是MegGriffin。我们可以借助这样的思想来对我们的语义解析进行改进。

**为了更好的去利用知识库**，我们用一种图的形式来代替语法解析树表示逻辑形式，这个图被称为**查询图**（*query graph*）。

问句*“Who first voiced Meg on Family Guy?"*对应的查询图如下图所示：



查询图可以分为以下四个部分：

**知识库实体**，在图中用圆角矩形表示。**中间变量**，在图中用白底圆圈表示。**聚合函数**，用菱形表示。**lambda变量（答案）**，在图中用灰底圆圈表示。图中实体节点到答案变量的路径可以转化为一系列join操作，不同路径可以通过intersection操作结合到一起，因此，该查询图在不考虑聚合函数argmin的情况下可以转化为一个lambda表达式，即：



（如果你不懂 lambda表达式，没有关系，上式表示 我们要寻找x，使得在知识库中存在实体y，满足 1. y和FamilyGuy存在cast关系；2. y和x存在actor关系；3.y和MegGriffin存在character关系，这里我们可以把y想象成是一个中间变量，通过对它增加约束来缩小它的范围，通过它和答案x的关系来确定答案x）

有了查询图，通过将其转化为lambda表达式就可以在知识库中查询得到答案。那么，如何去构造查询图呢？

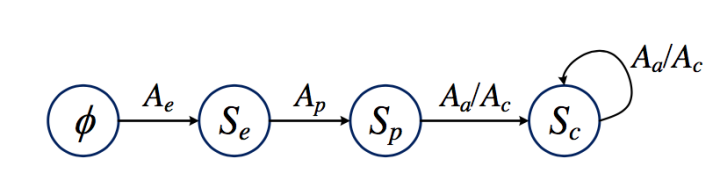
查询图的阶段生成

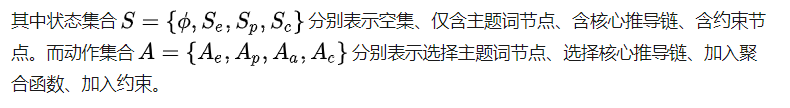
我们先看看查询图的构成成分。

问题中的主题词（可以看作是一个根节点）到答案变量的这条路径（如Family Guy - y - x）包含了所有的中间变量，这条路径可以看作是从问题到答案的一个核心推导过程，我们将其称作**核心推导链（core inferential chain）。**

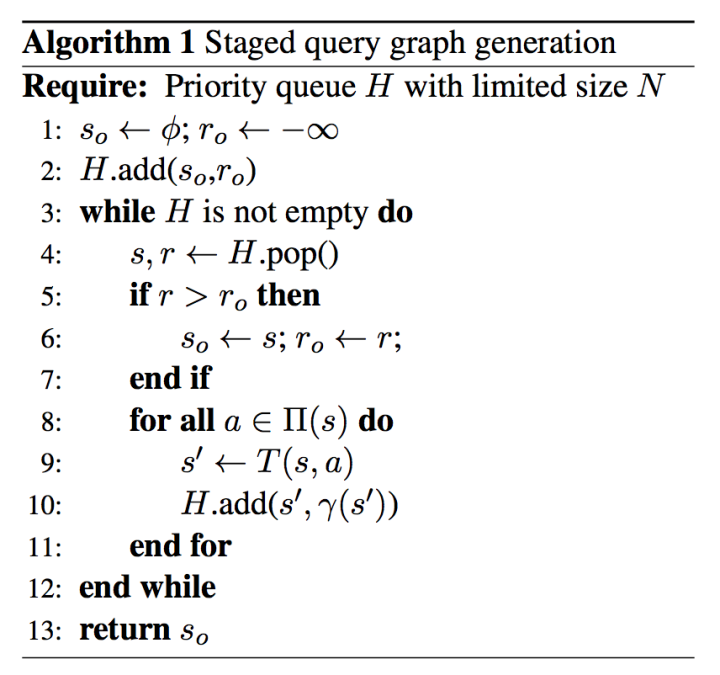
而对于核心推导链里的中间变量，我们可以对它加一些**约束**（要求它与其他实体具有一定的关系，如 y - character -> Meg Griifin）和**聚合函数**（如 y - from -> arg min）。

因此我们查询图的生成，可以分为以下几个步骤：**确定主题词**，**确定核心推导链**，是**否增加约束和聚合**。整个过程可以用下面的这个有限状态机自动机表示：





因此我们查询图可以分阶段生成，这个生成的过程实质上是一个搜索。依照我们的有限状态自动机，根据图所处的状态s，我们可以确定在该状态下可以采取的动作的集合（比如当前我们处在状态，根据有限自动机我们的动作为选择主题词节点，假设检测出来问句中有3个主题词候选，那么我们的动作集合大小为3）。因此，我们的查询图生成实际上是一个搜索过程，如果对这个搜索不加任何限制，那么这个搜索是指数级复杂度的。因此对于每一个状态s，我们可以用奖励函数（reward function）对它进行评估，奖励函数r得分越高表示这个状态对应的查询图和正确的语义解析表达越接近。我们用一个对数线性（log-linear）模型来学习奖励函数（这里涉及的一些概念不禁让人想起增强学习）。有了奖励函数，我们用best-first的策略利用优先队列进行启发式搜索，算法流程如下：

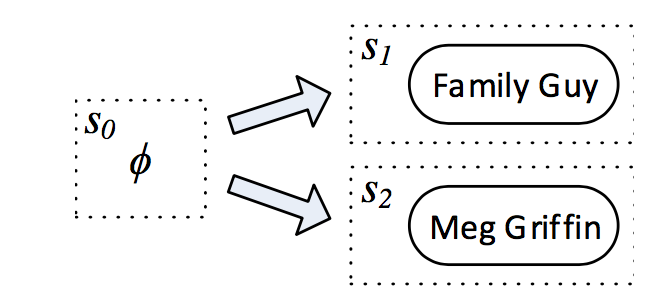


其中T(s,a)代表在s状态下采取动作a后得到的新状态，我们将优先队列的大小N限制为1000。上述算法可以简单概括为：每次从队列中取出得分最高的状态分别执行动作集中的每一个动作生成一批新的状态并压入优先队列，始终记录得分最高的状态，最终将得分最高的状态作为最后的查询图。

接下来，我们来看看每一种动作是怎么执行的，以及如何去构造奖励函数。我们依旧以问题*“Who first voiced Meg on Family Guy?"*为例。

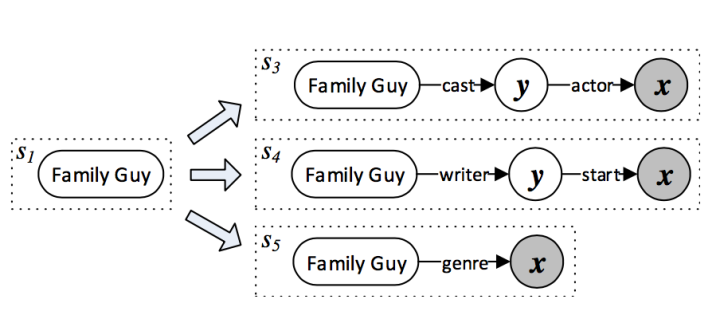
主题词链接

我们的第一种动作（action），就是从问题中确定主题词，这个操作称为**主题词链接**（Linking Topic Entity）。作者使用了S-MART作为实体链接系统，该系统是针对带噪音的短文本设计的，适合用于对问句提取主题词，它会为相应的 实体-自然语言短语 链接对 给出**链接得分**（Linking Score）。我们最多保留得分最高的10个实体作为候选，第一步如图所示：

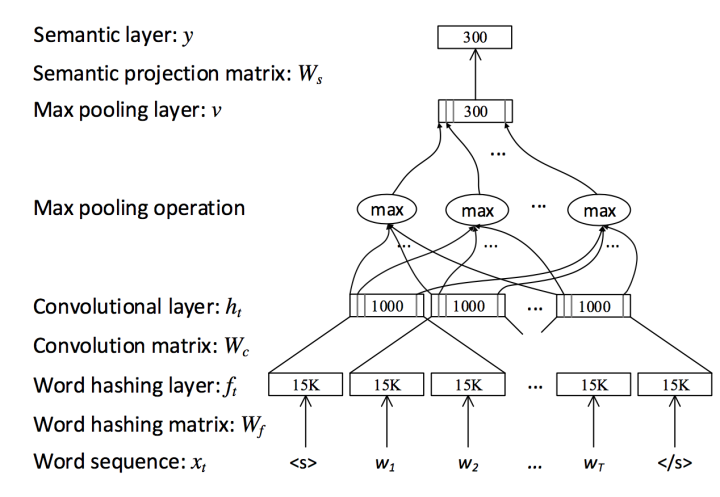


核心推导链

接下来，我们确定核心推导链。对于每一个候选的主题词，将它在知识库中对应的实体节点周围长度为1的路径（如下图s5）和长度为2且包含CVT节点的路径（如下图s3s4）作为核心推导链的候选（CVT，即复合值类型 Compound Value Types，是freebase中用于表示复杂数据而引入的概念，不了解的朋友可以点击该链接）。如下图：



核心推导链其实就是将自然语言问题映射为一个**谓语序列**（如cast-actor），因此我们可以用**卷积神经网络**来对这个映射进行打分，如下图所示：

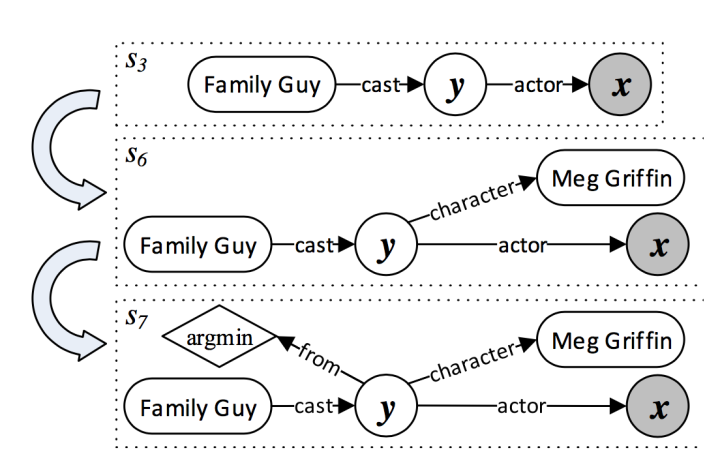


我们将自然语言和谓语序列分别作为输入，分别经过两个不同的卷积神经网络得到一个300维的分布式表达，利用表达向量之间的相似度距离（如cosine距离）计算自然语言和谓语序列的**语义相似度得分**。由于我们上期我们已对卷积神经网络做过介绍，因此这里我们对它不再赘述。需要注意的是，这里的输入采用的是**字母三元组**（letter-trigram）的方式，这是一个非常有趣的方式，类似于character-CNN。每个单词都将它拆分成几个 字母三元组，作为CNN的输入。比如单词*who*可以拆为#-w-h，w-h-o，h-o-#。每个单词通过前后添加符号#来区分单词界限（并且单词最短只含一个字母，添加两个#可以保证能形成至少一个字母三元组）*。*

采用**字母三元组的好处**在于：1.减小输入维度，这样输入维度可以稳定在字母集大小+1(#号)的三次方，即，而不是字典大小（同时可以处理一些字典中不存在的词和一些低频词，如缩写词等等）。2.相同语义的词语可能因为词根等缘故，前缀或者后缀会比较相似，这样能更好的提取单词语义的特征。3.对于现实生活中的用户，有时候可能会发生单词拼写错误，但错误拼写不会对这种输入方式造成太大影响。

增加约束和聚合函数

我们通过增加约束和聚合函数的方式扩展查询图，缩小答案的范围，以增加准确率，如下图



如何去增加约束和聚合函数呢？作者采用了基于一些简单规则的方式，比如当实体链接检测到句子中出现其他实体，那么我们可以增加一个约束。又比如句子中出现了first等时序敏感词，我们可以增加聚合节点。具体来说，根据以下规则确定**是否要为CVT节点添加约束节点或者聚合节点**：

1.约束实体出现在问句中

2.约束谓词表示事件的结束时间，但没有值（这表示它是当前事件）

3.问题中出现约束实体名称的一些单词

4.谓语是*people.marriage*.*type\_of\_union*（这说明关系是否是家庭伴侣关系、婚姻关系还是民事关系）

5.问句中包含单词 *first*或者 *oldest*，并且谓语是from形式的谓语（表明事件的起始时间）

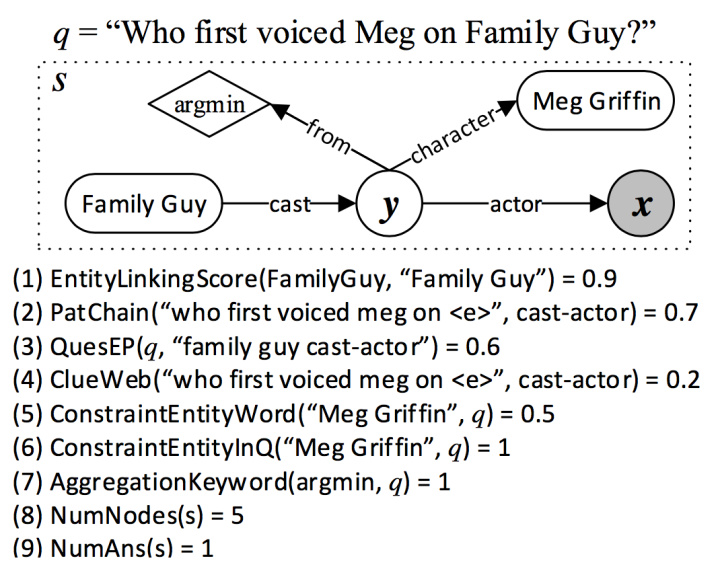
6.问句中包含单词 *last*, *latest* 或 *newest* ，并且谓语是to形式的谓语（表明事件的结束时间）

而对于答案节点，如果包含以下之一的谓语，我们会添加一个**约束节点**：

*people.person.gender / common.topic.notable types / common.topic.notable\_for*

奖励函数的特征定义

我们用对数线性模型训练奖励函数，因此我们要确定输入向量，和信息抽取以及传统语义解析方法一样，我们手工定义一个特征向量来表征整个查询图的信息，将它作为对数线性模型的输入。我们先来对特征有个主观上的感受，例如问题“Who first voiced Meg on Family Guy?” 对应的查询图，它的特征如下图所示：



具体来说，我们从 **主题词链接**、**核心推导链**、**增加约束聚合**三个方面定义特征。

**a.主题词链接特征**：实体链接得分（EntityLinkingScore），由实体链接系统给出。

如 EntityLinkingScore(FamilyGuy,"Family Guy")=0.9

b.核心推导链特征：

1.PatChain：将问句中的主题词替换为实体符号，和谓语序列同时输入两个不同的CNN，根据CNN输出的表达求语义相似度作为特征。

如: *PatChain("Who first voiced Meg on <e>", cast-actor) =0.7*

2.**QuesEP**：将谓语序列和主题词的规范名称（canonical name）连接（concatenate）起来作为输入，和问题求语义相似度。

如: QuesEP(q,“family guy cast-actor”) = 0.6

3.**ClueWeb**：用ClueWeb来训练一个更加*in-domain*的模型。如果一句话包含两个实体和谓语，那么就把这句话和谓语作为一组 数据对 输入模型进行训练。*注意：ClueWeb的输入和PatChain是一样的，但是其模型是用不同数据训练的。*

*从这定义的三个特征可以看出，这其实是一个ensemble模型，将三种模型的输出结果进行了一个log-linear组合。*

c.约束聚合特征：

对于CVT节点有以下特征：

1.约束实体是否出现在问句中 如*ConstraintEntityInQ("Meg Griffin",q)=1*

2.是否是当前发生的事件

3.是否是当前发生的事件，且问句中包含关键词“currently”, “current”, “now”, “present” 和“presently”

4.约束实体单词出现在问句中的百分比 如ConstraintEntityWord("Meg Griffin",q)=0.5

5.约束谓语的类型是people.marriage.type\_of\_union

6.问题中是否包含“first” 或 “oldest” ，谓语是from形式谓语，并且CVT节点按该from性质排序是第一

7.问题中是否包含“last”, “latest” 或 “newest” ，谓语是to形式谓语，并且CVT节点按该to性质排序是最后

对于答案节点有以下特征：

1.性别一致性（男性）：约束谓语是*gender*，并且问句中出现了以下男性关键词中的一个{“dad”, “father”, “brother”, “grandfather”, “grandson”, “son”, “husband”}

2.性别一致性（女性）：约束谓语是*gender*，并且问句中出现了以下女性关键词中的一个{“mom”, “mother”, “sister”, “grandmother”, “granddaughter”, “daughter”, “wife”}

3.当约束谓语是 *notable\_types* 或 *notable\_for* 时，约束实体单词出现在问题中的百分比

d.总体特征

查询图对应的答案数量NumAns和查询图的节点数NumNodes

模型学习

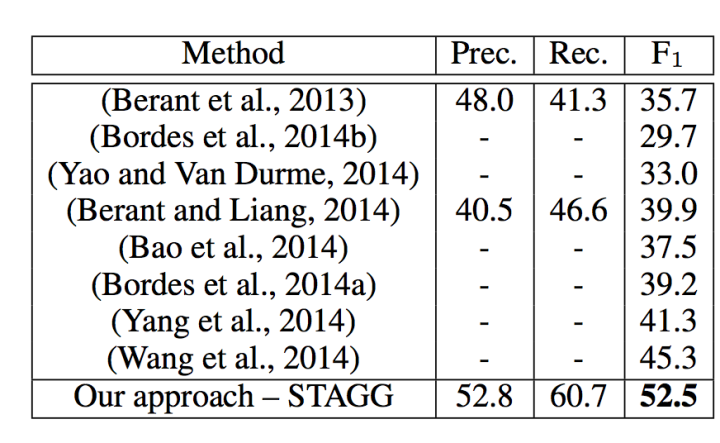
在信息抽取中，我们的模型是在进行二分类（根据特征向量判定候选答案是否是正确答案），而在本文中，我们对模型不进行二分类，而是根据查询图对应的实体和真实答案的F1-score进行**排名**。基于lambda-rank算法对一个一层的神经网络进行训练。这样做的好处是，有些查询图虽然查询得到的答案和真实答案不完全相同，但根据它的相同程度（F1-score）也可以说它比完全错误的查询图要好。

论文实验与总结

在训练数据上，通过实体链接系统确定候选实体，候选实体到正确答案的知识库路径（长度限制为2）作为核心推导链的正样本，错误查询图中的路径作为负样本。根据训练数据，作者生成了17,277个F1-score不为0的查询图（正样本）和1.7M完全错误的查询图（负样本）对卷积神经网络进行训练。

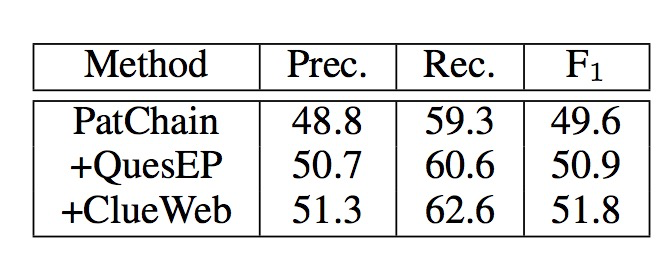
对于奖励函数的训练，为每个问题生成了4000个样例（包含所有正确的查询图和随机选择的负样本）以F1-score作为排名标准来训练排名器（ranker）。

该方法与当时的所有baseline进行了比较，效果如下



可以看出该方法取得了相当大的提升，也因此获得了当年的Outstanding paper。

本方法使用到了外部的实体链接系统，作者也比较了使用Freebase Search API时的性能，F1-score会下降约4.1。同时，作者也对核心推导链所涉及的三个特征的性能进行了比较，核心推导链三个特征的性能如下表：



我们可以发现，其实只使用PatChain的性能就已经很好了（达到了惊人的49.6），原因是WebQuestion里50%的问题可以只是用核心推导链就可以得出正确答案。

最后作者进行了**错误分析**，随机选择100个答错的问题，发现35%的错误来自核心推导链构建错误，23%来自约束错误，8%来自实体链接错误，剩下34%的错误来自于标签错误或不完整已经问题中的实体有歧义等。也就是说有34%的错误是数据的问题，这再一次显示出了该方法的强大。

由于本期内容较多，我们再做一个快速的回顾：

1.考虑到传统语义解析与KB结合**不够紧密**，作者提出了**查询图**的概念

2.查询图的构造由**实体链接**系统**确定主题词**，**核心推导链**，增加**约束**和**聚合**这几种操作构成

3.对于查询图的每一个状态，我们都用一个**奖励函数**对它进行评价，使用**优先队列**进行**启发式搜索**构建查询图

4.通过查询图的**实体链接得分**、**核心推导链三个特征**、**约束聚合手工特征**以及**全局特征**作为输入向量，训练单层神经网络作为**排名器**得到奖励函数

5.核心推导链使用**卷积神经网络**（**letter-trigram**作为输入）进行训练，并且**ensemble**了三个不同数据训练的模型

总的来说，我们可以看出，该方法几乎融合了传统语义解析、深度学习、信息抽取等方法的优点，还使用了部分手工特征（对数据进行了仔细观察和分析），确实是一个很令人惊叹的方法。

在深度学习篇的中篇和上篇，我们可以看到都使用了卷积神经网络对模型进行提升。下一期，我们将进入深度学习篇的下篇，看如何使用更加**复杂的深度学习模型**进行KB-QA，为大家进一步揭开KB-QA的面纱。

## 记忆网络

在上两期，我们分别介绍了使用深度学习的卷积神经网络对向量建模和语义解析方法进行提升的两篇经典论文。今天我们将为大家带来两篇使用深度学习其他两种经典模型进行KB-QA的文章，分别是**记忆网络（Memory Network）**和**带注意力机制的循环神经网络**（Recurrent Neural Networks with Attention Mechanism）。它们都是深度学习用于自然语言处理领域（Deep Learning for NLP）中相当火热的方法，就让我们一起来看看如何将它们应用到KB-QA中吧。

*（由于内容较多，我们将本文拆成2篇文章）*

什么是记忆网络

*（了解记忆网络的朋友可直接跳过该段）*

记忆网络由Facebook AI Research的Jason Weston等人提出，其论文发表在2015年ICRL会议。对于很多问题（如自动问答、对话系统等），我们都希望我们的模型能够像我们人一样具有记忆机制，能够记住一些信息（上下文、知识库等），为此我们发明了很多的记忆模型。一个广为人知的记忆模型就是**长短时记忆网络**（Long Short-Term Memory, LSTM）。然而它的记忆实质上是一种**内在（Internal）表达**，只能记住在使用时所输入的一些历史信息。它就像计算机的内存一样，能够存储一些使用时的历史输入信息（把所有的历史输入信息都压缩到一个低维空间上，这样也会损失很多信息），一旦使用完毕，记忆就会被抹去，记忆量也非常有限。

我们能不能找到一种**外部（External）表达**的记忆，就像计算机的硬盘一样，能够长期存取大量的记忆信息呢？在使用的时候再去记忆库中读取相关的记忆。对于我们的KB-QA问题，我们能否将知识库里的知识直接存储在我们的模型中呢？近年来，人们在这方面的研究作出了不懈努力，发明了一些能够提供外部记忆的模型，其中有两个最著名的模型，一个是由Google DeepMind团队提出的神经图灵机（Neural Tuning Machine），而另一个就是今天要讲的记忆网络。

记忆网络其实是一种框架结构，它里面的每一个模块（也称组件，Component）都可以根据应用套用不同的模型，也可以根据应用的需要增加其他的模块。总体来说它可以分为以下5个模块：

**输入模块I**（Input）：**将外部输入转化为内部表达**。我们可以把输入模块想象成人类的感知器官，接受外部的原始输入并将其转化为能够被大脑处理的生物信号。具体来说，输入模块用于将外部的输入（如自然语言句子、图片等）x转化为内在的分布式表达I(x)。输入模块可以根据应用的需要选用不同的模型，诸如CNN、LSTM、bag-of-words等等。

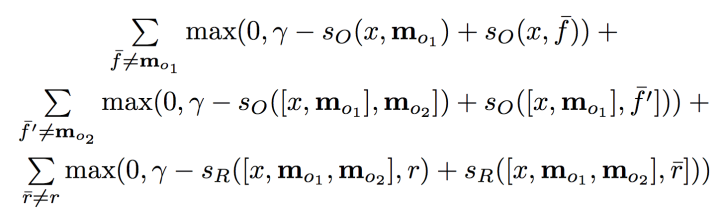
**记忆模块M**（Memory）：**用于存储记忆**。每一条记忆都是一个**分布式表达**向量mi，每一个记忆有一个索引地址号i，所有记忆组成了我们的记忆模块。我们的记忆模块就像是计算机中的硬盘一样，存储着一条条的信息，每一个信息都有一个物理地址。我们可以记忆模块进行记忆的读取，修改，删除，也可以插入新的记忆信息。

泛化模块G（Generalization）：根据外部输入的内在表达I(x)去更新记忆G(mi,I(x),m)。我们可以根据新的输入对记忆模块中的每一个记忆都进行更新并增加新的记忆，也可以仅对部分相关记忆作针对性更新。最简单的方式是，直接将输入存储在记忆模块中而不对其他的记忆进行修改，即。这里H(x)是一个为输入选择记忆槽（为输入确定记忆的索引地址）的函数，在记忆存储满的时候，也可根据H(x)函数对记忆进行替换。更复杂一点，我们也可以引入一些遗忘机制等等。

输出模块O（Output）：根据外部输入的内在表达去选择记忆模块中与之最相关的记忆。该记忆也称为支撑记忆（Support Memory），即。这个选择的方式有很多，最简单的方法就是将I(x)和记忆向量mi投影到同一个向量空间中，通过比较相似度得分（cosine，点乘等）来选择最相似度得分最高的记忆作为输出。当然我们也可以根据应用的需要，选择多个记忆作为输出。选择多个记忆的方式也有很多种，最简单的方法就是选择相似度得分最高的k个记忆作为输出，论文中作者提出了另一种方法，先选择得分最高的作为第一个输出o1，再选择与I(x)得分加上与o1得分 最高的记忆作为第二个输出，以此类推（这种方法更像是在进行推理，先将输入作为推理的支点，选择与输入最相关的记忆作为第一条线索，再根据输入和第一条线索去确定下一条线索）。

回答模块R（Response）：根据输入的内在表达和输出模块提供的记忆，输出满足应用需要格式的结果。它就相当于encoder-decoder模型中的decoder部分，将作为输入，根据应用的需要使用不同的模型输出最终的结果（分类器、语言模型、RNN、CNN等）。

如何去训练记忆网络呢？给定训练数据 输入输出对(x,r) 我们可以通过梯度下降算法优化margin-based ranking损失函数，去训练记忆网络中的所有参数。假设输出模块每次输出两条支撑记忆，我们的损失函数如下：

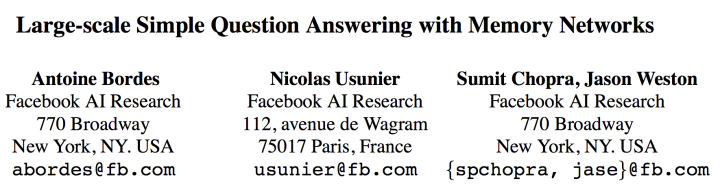


其中是随机采样的一些负样本。需要注意的是，这里我们在构造损失函数的时候，需要提供输出模块输出**正确记忆的标签**（即上式的），即记忆的正样本。但是在一般情况下，我们的训练数据都只有 输入输出对 ，难以提供关于记忆的标签。针对这一问题，纽约大学和Facebook AI Research的Sukhbaatar等人提出了端到端学习（End-to-End）的记忆网络，感兴趣的朋友可以点进去看看。

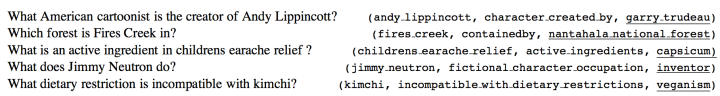
关于记忆网络还有很多的技巧（trick），我们可以通过**哈希**或**聚类**的方法提高查找记忆的效率，通过单词的上下文来处理未见过的单词等等，这里我们不再展开。

可以看出相比LSTM的内部记忆，记忆网络可以存储大量的外部记忆，并且它具有很强的扩展性（记忆网络自提出之后吸引了大量研究，出现了各式各样的变体和应用，如同现在的生成对抗网络一样火爆）。那么如何将记忆网络应用在KB-QA中呢？

基于记忆网络的KB-QA

我们以Facebook AI Research的Bordes等人在2015年发表的论文Large-scale Simple Question Answering with Memory Networks为例，介绍如何将记忆网络应用于KB-QA，该论文的作者Bordes也是我们在第四期中介绍的KB-QA向量建模方法经典论文的一作。

这篇论文初步尝试了将记忆网络应用于KB-QA中。作者认为现在的KB-QA对于解决**只依赖一个知识三元组**的简单问题（称为 Simple Question Answering）仍然有困难，为此作者构建了一个更大的简单问题数据集，称作**SimpleQuestions。**该数据集的每个问题都依据一个知识三元组知识，进行人工构建问题，数据集最终一共包含了108,442个问题-答案对，相比之前只含8000多个问题-答案对的benchmark数据集WebQuestion，其数据量大了很多。该数据集的部分数据如下图所示*（下划线表示答案）*：

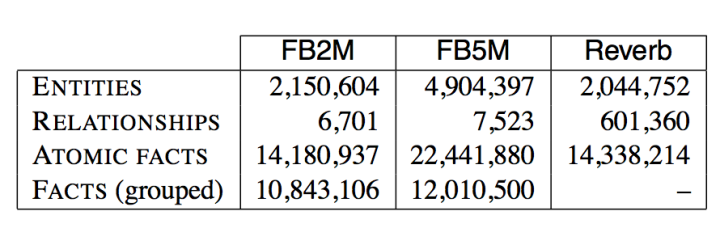
接下来我们看看作者如何使用记忆网络。其整体思想是将知识库里的知识存储到记忆模块M中，问题经过输入模块I转化为分布式表达，输出模块O选择与问题最相关的支撑记忆（由于SimpleQustions的问题只依赖一个知识，所以只需要选择一条记忆），回答模块R将该记忆对应三元组的宾语作为最终答案输出。最后，为了测试记忆网络的泛化能力，在模型训练完毕后，我们将Reverb中提取的三元组（我们在揭开知识库问答KB-QA的面纱2·语义解析篇中对Reverb作过介绍，它的知识三元组是自然语言形式，如(“Obama”, “was also born in”, “August 1961”)，知识三元组抽取自ClueWeb）作为新的知识，用泛化模块G将新知识存储到记忆模块中，在不经过re-training的情况下使用该记忆回答问题，测试模型的泛化性能。

具体来说，整体流程可以分为以下三步：

1.**存储知识**。首先我们要将我们的知识库里的知识，存储到记忆网络中。作者使用Freebase的两个子集FB2M（含2M实体和5K实体关系）和FB5M（含5M实体和7K实体关系）分别作为知识库。使用**输入模块I**来处理数据。

由于一个问句可能有多个答案，并且对于一个问题**输出模块O**只选择一个支撑记忆，我们先对知识做两种预处理：

a.将具有相同主语和实体关系的三元组进行**合并**（Group），这样每一条知识将包含k个不同的宾语，即。合并的原因在于我们想用一条记忆回答具有多个答案的问题*（我们将合并前的知识三元组称为Atomic Facts，合并后三元组的称为Facts）*，合并后和合并前的知识库大小如下：



b.去除中间节点。有些知识中日期会链接两个实体以区分某一事实的时间范围，我们可以将中间节点（mediator node）去除转化成一个**二阶关系**，这样我们就把长度为2的路径压缩成了长度为1的路径，即压缩为一个三元组。这个操作使得WebQuestion里的能被单一关系回答的问题数量从65%上升到86%。

预处理完知识后，我们的**输入模块I对**知识进行预处理并存储到记忆中。这里使用词袋模型bag-of-symbol的方法，用一个维的multi-hot向量f(y)来表示每一条知识并作为记忆。的大小为知识库实体和实体关系的大小之和，主语实体s和实体关系r对应向量维的值为1，宾语实体oi对应维的值设为1/k。

2.**训练记忆网络**。我们用问题-答案对来训练记忆网络。

首先用**输入模块I**来处理输入的自然语言问句q，我们使用n-gram词袋模型（bag-of-ngrams）方法，用一个Nv维的multi-hot向量g(q)来表示每一个问句。Nv的大小是字典大小，字典包含所有问题中出现的单词和所有知识库实体的自然语言别称（这个别称可能由多个单词构成，我们用1个n-gram来表示）。

对于输入g(q)，我们的**输出模块O**要在记忆中寻找一个与之最相关的支撑记忆。为了避免遍历整个记忆模块里的每一条知识，我们先确定一个**候选范围**。确定方式如下，将问句中的所有n-gram与知识库实体别称进行匹配，以确定候选实体，将含有候选实体作为主语的知识作为我们的候选支撑记忆。我们将记忆和问题投影到一个低维分布式空间，通过cosine相似度作为得分函数，来寻找最相关的支撑记忆，即：。这里我们需要学习的参数就是两个权值矩阵。

和我在揭开知识库问答KB-QA的面纱4·向量建模篇中提到的训练方法一样，我们构建margin-based ranking损失函数，也进行多任务的训练，通过多任务训练让语义相同的问题的分布式表达相似。这里不再赘述。

需要注意的是，我们知道构建margin-based ranking损失函数需要提供支撑记忆的正样本和负样本，由于SimpleQuestions数据集的每个问题都有对应的知识标签，因此我们已经有支撑记忆的正确标签。但是对于WebQuestion数据集，我们没有正确的支撑记忆标签，作者通过类似之前寻找候选支撑记忆的方式去得到标签。

训练完毕后，作者在WebQuestion和SimpleQuestions上做了测试，在WebQuestion上取得了42.2的F1-Score，在SimpleQuestion上正确率63.9%。作者发现如果仅用WebQuestion来训练模型，那么在SimQuestion测试集上正确率只有46.6%，说明了WebQuestion数据集的数据量还是具有一定的局限性。

3.**测试网络泛化能力**。使用**泛化模块G**来连接新的知识库Reverb到我们的记忆中，通过实体链接和实体别名匹配等方式，来匹配已有记忆中的实体和新知识库里的实体（这种方式只能匹配到新知识库中17%的实体）。新知识库中剩下的实体和所有的关系都用词袋模型表示，因此我们可以用一个Nv+Ns维的向量h(y)来表示新知识并将其存储到记忆中。同样的，输出模块在寻找支撑记忆时的相似度得分函数为，其中矩阵Wvs直接由之前训练好的Wv,Ws拼接（concatenate）而成。

我们将该新的知识库存储到记忆中，在不经过re-training的情况下用Reverb的一个测试集（含691个问题）进行测试，达到了67%的正确率，接近当时state-of-the-art的方法。

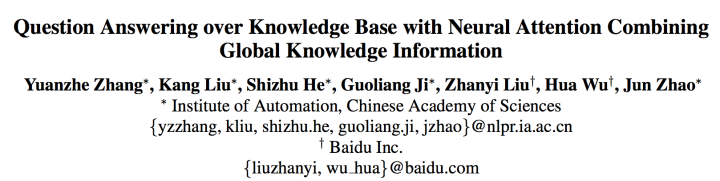
可以看出该方法使用了一个很基础版本的记忆网络来解决**简单问题**的KB-QA。这篇论文使用的方法很简单，本质上和传统向量建模方法很接近。但是我们也可以看出，记忆网络这种结构具有的很强大的**扩展性**，它的每个模块都有很多改进的空间。总的来说，**记忆网络是一种很适合进行KB-QA的深度学习框架，**相信今后会有很多使用记忆网络进行KB-QA的优秀文章**。**

此外，个人认为使用更加**复杂的记忆网络**是未来深度学习解决KB-QA的一个很有前景的途径。记忆网络的框架给了我们很多的提升空间：引入更多的技巧，使用更合理的模型作为记忆网络的组件，在记忆选择中引入推理机制，注意力机制和遗忘机制，将多源的知识库存入记忆等等。

下篇文章中，我们再一起看看如何将注意力机制引入KB-QA中。

## 注意力机制

我们紧接上篇揭开知识库问答KB-QA的面纱7·深度学习下篇（一）的内容，为大家介绍注意力机制的KB-QA

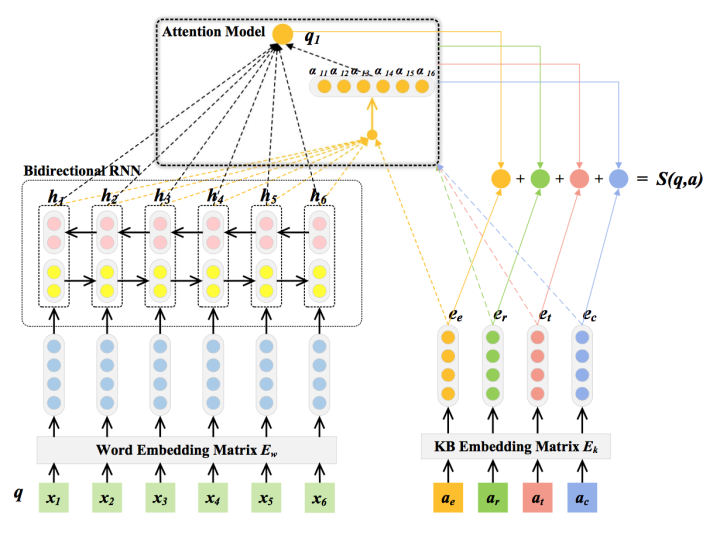
文章Question Answering over Knowledge Base with Neural Attention Combining Global Knowledge Information由中科院自动化所刘康老师等人在2016年发表在arxiv上，是一篇相对较新的文章。该文章和我在深度学习上篇分享的文章类似，也是使用深度学习对向量建模方法进行提升，不同于之前使用CNN提取问句特征，而该文章使用双向LSTM并结合问题引入注意力机制提取问句特征，在WebQuestion上取得了42.6的F1-Score，击败了之前的Multi-Column CNN。

深度学习提升向量建模方法的大体框架都很接近：根据问题确定主题词，根据主题词确定候选答案，通过候选答案和问题的分布式表达相似度得分确定最终答案。而方法的核心在于学习问题和候选答案的分布式表达，其实相关的方法都是在这两个部分做文章。这篇文章的想法在于，对于不同的答案，我们关注问题的焦点是不同的，我们根据候选答案的信息，来引入注意力机制，对同一个问题提取出不同的分布式表达。

比如 对于问题 "who is the president of France?"，其中之一的答案是实体“Francois Holland”，我们通过知识库可以知道Francois Holland 是一个总统，因此我们会更加关注问句中的 “president” 和 “France” 单词，而根据Francois Holland的类型person，我们会更关注问句中的疑问词who。

（如果你之前没有接触过注意力机制也没有关系，可以直接往下看，也可以看看关于注意力机制最经典的文章，Yoshua Bengio等人在2015年ICLR发表的Neural Machine Translation by Jointly Learning to Align and Translate，该文章提出的encoder-decoder with attention mechanism模型几乎可以横扫大部分的NLP问题）

该方法的整体框架如下图所示：



具体来说，可分为以下三个步骤：

1.**将候选答案转化为分布式表达**

我们从多个方面考虑答案的特征：答案实体、答案上下文环境（知识库中所有与答案实体直接相连的实体）、答案关系（答案与问题主题词之间的实体关系）、答案类型。每一种特征都可以用Vk维的multi-hot向量表示，Vk即知识库实体和实体关系的数量之和。我们通过Embedding矩阵Ek将每一种特征转化为低维的分布式表达，我们就得到了四种关于答案的分布式表达（其中由于答案上下文环境涉及的实体较多，我们取这些实体的embedding均值作为上下文环境的embedding）。

2.将自然语言问题转化为分布式表达

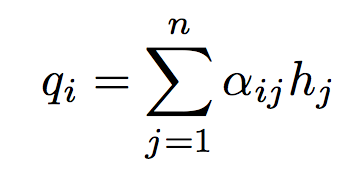
将问句中的每一个单词经过Embedding矩阵Ew转化成word-embedding，使用双向LSTM（bi-LSTM）提取问句特征。bi-LSTM第j时刻的输出记作hj，使用bi-LSTM的好处在于hj既包含了第j个单词之前的信息，又包含了该单词之后的信息。

3.在得分函数中引入注意力机制

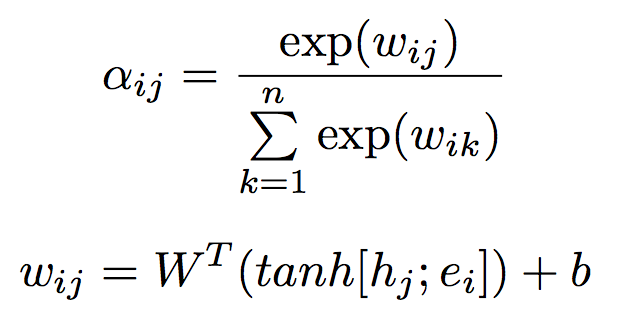
我们希望我们问句的分布式表达对于四种不同的答案特征有不同的表达（根据答案的特征对于问题有不同的关注点），第i种答案的分布式表达ei 对应的问句分布式表达记作qi，我们的得分函数定义为四种对应表达的点乘之和，即：



对于一般的LSTM，我们通常将最后一个时刻的输出hT作为句子的最终表达，而在这里，我们引入注意力机制，根据问题的特征，给予每一时刻的输出不同程度的关注（对bi-LSTM每一时刻的输出进行加权求和），即：



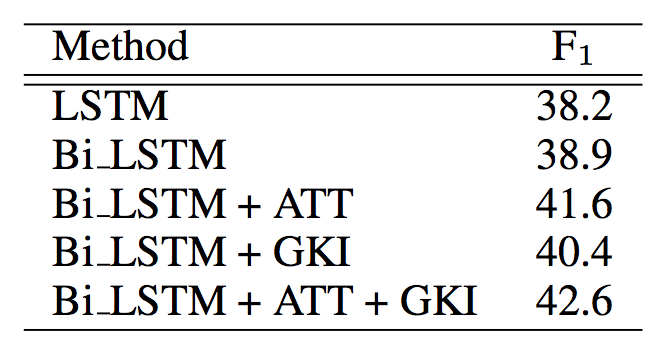
其中的权重系数aij取决于bi-LSTM第j时刻的输出hj和第i种答案特征的分布式表达ei，因此我们可以使用一个单层的神经网络去学习这个权重，并通过Softmax对权重进行归一化，公式如下：



4.**OOV问题**  
特别一提的是，在测试的过程中，我们的候选答案可能从未在训练集中出现过，因此它对应的分布式表达是没有被我们的模型训练过的（这个问题称为*the problem of out of vocabulary*, OOV，之前的文章很少有考虑过这个问题的）。为了解决该问题，作者利用TransE对知识库进行训练，训练实体和实体关系对应的Embedding矩阵Ek（实际操作中，作者通过轮流训练KB-QA模型和TranE的方式训练并共用Embedding矩阵Ek，每训练一个epoch的KB-QA就训练100个epoch的TransE）。这样，我们就利用了整个知识库的特性，预先对每一个知识库实体都进行了训练，使得相似实体的分布式表达也很相似。因此，即使遇到KB-QA训练集中未遇到的候选答案实体，KB-QA模型也能将它视作是在训练集中出现过的某个和它分布式表达相似的实体，这样就减轻了OOV问题所带来的破坏性。

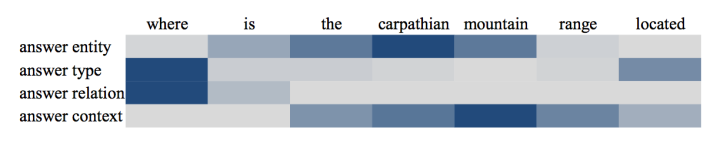
（关于TransE：TransE是知识图谱补全的经典方法，它借鉴了word-embedding的思想，能够将知识库中的实体和实体关系用分布式向量表达。其主要思想是对于一个知识三元组（s,r,o），我们希望主语实体的分布式表达e(s)加上关系实体的分布式表达e(r)能够尽量接近宾语实体的分布式表达e(o)，因此我们可以构建类似的margin-rank损失函数通过正样本和采样负样本进行训练。TransE提出之后还出现了大量的改进算法，诸如TransH、TransR、TransG、TranSparse、TransD等等。之后如果有机会我会专门写文章或专栏对知识图谱补全相关的方法进行介绍）

在实验环节，作者对模型进行了分析，分析使用注意力机制（记作ATT）、利用知识库全局信息使用TransE训练实体embedding（记作GKI）以及bi-LSTM对性能的影响，在WebQuestion测试集上F1-Score的结果如下：



可以看出ATT和GKI这两个机制都对模型性能有一定的提升，最终模型取得了42.6的F1-score，击败了几乎所有的深度学习提升向量建模的方法（记忆网络是42.2而Multi-column CNN是41.3）。当然这个方法在WebQuestion上的F1-Score距离我们在深度学习中篇所提到的语义解析方法（F1-Score 52.5）还有一定的距离，但论文中也提到相比该方法设置大量的人工特征（很多特征是对训练集观察得到的），这个方法具有更强的适应性和可扩展性。

注意力机制还有另外一个好处，那就是**可视化**，通过可视化每个单词的权重，可以得到一些可解释性，如下图：



图中颜色越深表示权重越大，我们可以通过该图看出一些符合人类直觉的解释（如对应答案的类别，我们会更加关注疑问词等等）。

深度学习篇小结

我们在KB-QA深度学习篇中，为大家介绍了使用CNN、RNN、记忆网络、双向LSTM、注意力机制等深度学习的火热方法与KB-QA结合的几篇经典文章。我们可以看到深度学习具有传统方法所不具有的**高效性、可扩展性、适应性、鲁棒性**等优点。我相信，在**未来几年的KB-QA已离不开深度学习**。

此外，个人认为使用更加**复杂的记忆网络**是未来深度学习解决KB-QA的一个很有前景的途径。记忆网络的框架给了我们很多的提升空间：引入更多的技巧，使用更合理的模型作为记忆网络的组件，在记忆选择中引入推理机制，注意力机制和遗忘机制，将多源的知识库存入记忆等等。

本期之后，我会不定时的更新一些KB-QA新的有趣的文章，欢迎大家一起来探讨和交流KB-QA相关的知识。

这里再分享一些**公开数据集的下载地址**，方便想要复现的朋友们使用：

WebQuestion: [https://github.com/percyliang/sempre](http://link.zhihu.com/?target=https%3A//github.com/percyliang/sempre)  
Freebase: [https://github.com/percyliang/sempre](http://link.zhihu.com/?target=https%3A//github.com/percyliang/sempre) (or [https://developers.google.com/freebase/](http://link.zhihu.com/?target=https%3A//developers.google.com/freebase/))

wikianswers: [http://knowitall.cs.washington.edu/paralex/](http://link.zhihu.com/?target=http%3A//knowitall.cs.washington.edu/paralex/)

接下来，我们会考虑写一些与KB相关的其他系列文章，比如**基于KB的VQA**，**知识图谱推断与补全**等，欢迎有意向合作或投稿的朋友与我联系~