# 论文阅读笔记

## Step1

MF1833063, 史鹏, spwannasing@gmail.com

2019 年 3 月 13 日

# 1 Chinese Poetry Generation with Planning based Neural Network

这篇论文主要介绍了一种seq2seq的模型，用于生成古诗，根据输入的文本：一句话或者一篇文章，生成表达对应含义的古诗。

1. 提出了一种“planning-based”诗歌生成框架，这个框架明确的规划每一行的子主题。
2. 修改了RNN的encoder-decoder框架，使其同时支持对于子主题以及前面已经生成的行进行编码，逐行的生成诗句。

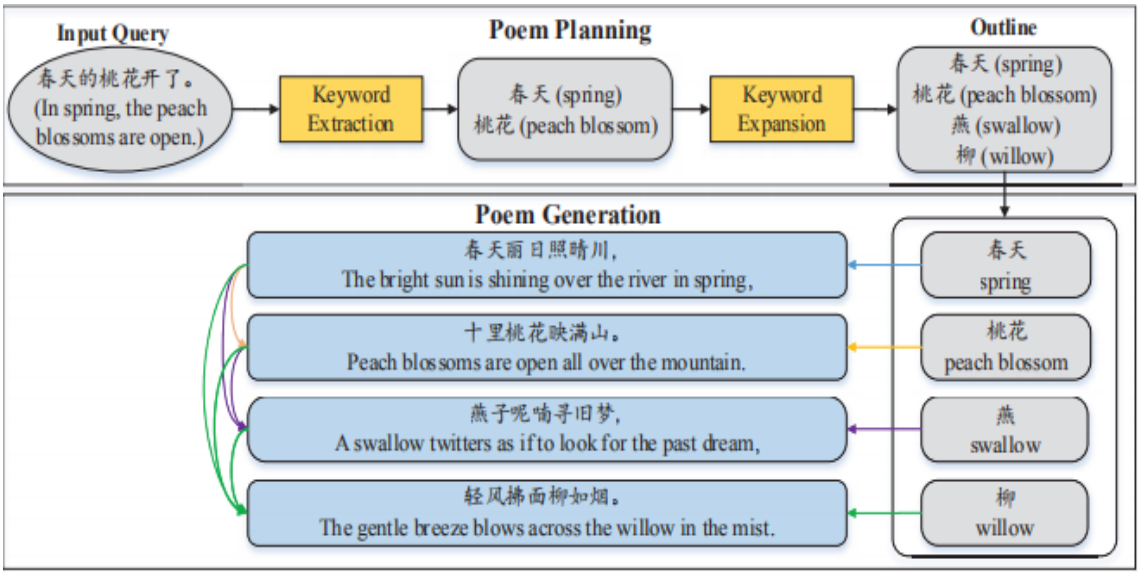



图 1: 整体工作流程

3. 对于提取用户输入的文本中的关键词使用了TextRank算法。
   TextRank算法基于PageRank，用于为文本生成关键字和摘要。

$$WS(V_i)=(1-d)+d*\sum_{V_j\in\in(V_i)}\frac{w_{ji}}{\sum_{V_k\in Out(V_j)}w_{jk}}WS(V_j)$$

图 2: 权重计算公式

思想：（1）如果一个单词出现在很多单词后面的话，那么说明这个单词比较重要
（2）一个TextRank值很高的单词后面跟着的一个单词，那么这个单词的TextRank值会相应地因此而提高

4. 扩充关键词，用户输入的文本提取出的关键词的数量可能不足，这篇论文提出了两种方式解决这个问题。

[1] RNNLM-based方法

使用RNN语言模型来根据已有的关键词序列预测下一个关键词,自动对训练数据根据TextRank算法生成每行诗的关键词，用这些关键词序列来训练RNNLM。

[2] Knowledge-based方法

利用外部的知识，比如wiki百科，抽取出在这些预料中，已有关键词附近的词，同时这些词还要是名词或者形容词，而且这个词在诗歌训练集中出现过。

5. 诗歌生成

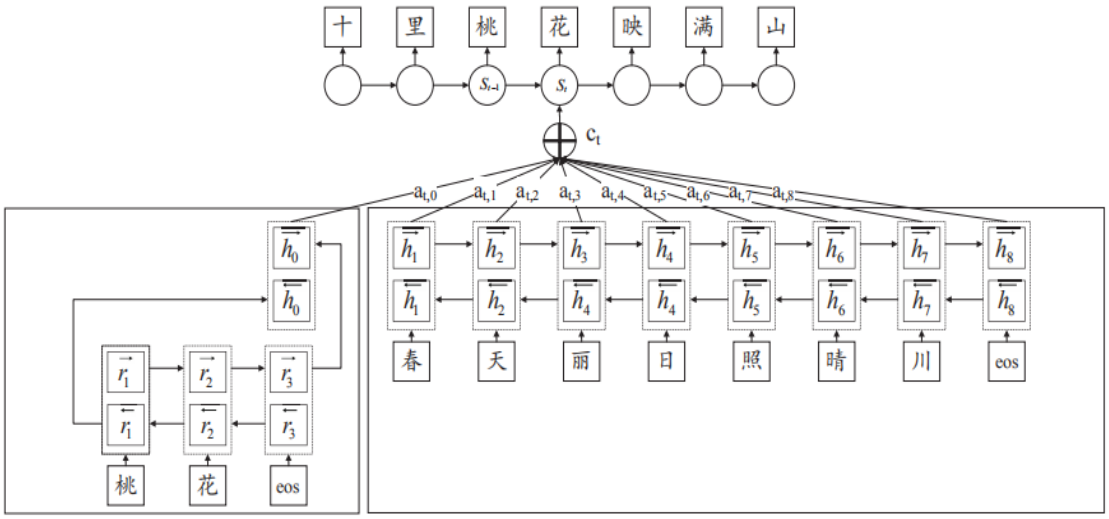



图 3: 诗歌生成模型

$$y_t = \arg\min_y P(y|s_t, c_t, y_{t-1})$$

$$s_t = f(s_{t-1}, c_{t-1}, y_{t-1})$$

$$c_t = \sum_{j=0}^{T_h-1} a_{tj} h_j$$

$$a_{tj} = \frac{exp(e_{tj})}{\sum_{t=0}^{T_h-1} exp(e_{tk})}$$

$$e_{tj} = v_a^T tanh(W_a s_{t-1} + U_a h_j)$$

# 2 MEMORY NETWORKS

描述了一种新的学习模型，叫做记忆网络，记忆网络推理结合了长期记忆组件和推理组件，学习如何共同使用这些组件。

1. RNN等网络结构，因为总是将序列编码到固定长度，所以其记忆能力有限，难以记住一些复杂的信息，甚至是复制自己这种简单的任务都不可以。

2. 本文的中心思想是将可读写的记忆组件与成功的推理学习策略结合起来。记忆网络由四个组件组成。I、G、O、R

   I：input feature map–将输入转化为内部的特征表示。

   G:generalization–根据输入更新“记忆”内的参数，我们称之为泛化，因为在这个阶段，网络有机会压缩和压缩它的记忆，以供将来使用。

   O:output feature map–根据输入和当前的记忆，来输出

   R:response–将上一步的输出转化为文字

3. 最简单的G的形式是将I（x）存入它记忆的槽（slot）内。

$$m_{H(x)} = I(x)$$

   这里面H（x)的作用是选择一个合适的槽。更复杂的G变体可以根据当前输入x中的新特征，返回并更新先前存储的内存(可能是所有内存)。如果内存满了，当H选择替换哪个内存时，它也可以实现一个“遗忘”过程，例如，H可以计算每个内存的效用，选择作用最小的。

4. 推理的核心是O和R,O通常对于输入x使用寻找”k supporting memories”的方法。这里假设k=2,则

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

$$o_2 = O_2(x, m) = \arg\max_{i=1,\ldots,N} s_O([x, m_{o1}], m_i)$$

   对于图4中的例子，$m_{o1}$ = “Joe left the milk”,$m_{o2}$ = “Joe travelled to the office” 然后在$[x, m_{o1}, m_{o2}]$的基础上，对各单词score。选出得分最高的单词。

Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk.
Joe travelled to the office. Joe left the milk. Joe went to the bathroom.
Where is the milk now? A: office
Where is Joe? A: bathroom
Where was Joe before the office? A: kitchen

图 4: 例子

# 3 Teaching Machines to Read and Comprehend

当前缺少大规模的问答预料，本文的贡献是用一种特殊的方法生成了大规模的有监督阅读理解数据。另外还提出了基于Attention的深度神经网络来进行阅读和回答问题(baseline)。

1. 观察到，摘要和句子及其相关文档可以很容易地通过简单的实体检测和算法转换为上下文查询答案。

| Original Version | Anonymised Version |
|---|---|
| **Context** | |
| The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the "Top Gear" host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon "to an unprovoked physical and verbal attack." ... | the *ent381* producer allegedly struck by *ent212* will not press charges against the " *ent153* " host , his lawyer said friday . *ent212* , who hosted one of the most - watched television shows in the world , was dropped by the *ent381* wednesday after an internal investigation by the *ent180* broadcaster found he had subjected producer *ent193* " to an unprovoked physical and verbal attack . " ... |
| **Query** | |
| Producer **X** will not press charges against Jeremy Clarkson, his lawyer says. | producer **X** will not press charges against *ent212* , his lawyer says . |
| **Answer** | |
| Oisin Tymon | *ent193* |

图 5: 示例

2. 计算某单词作为答案的概率：$p(a|d,q) \propto exp(W(a)g(d,q)$,W(a)是关于a索引的矩阵，g(d,q)返回的是一个关于document和query的向量。以下讨论的问题即为获取到g(d,q)的几种方式。

3. 提出了三种利用神经网络进行计算的方式。

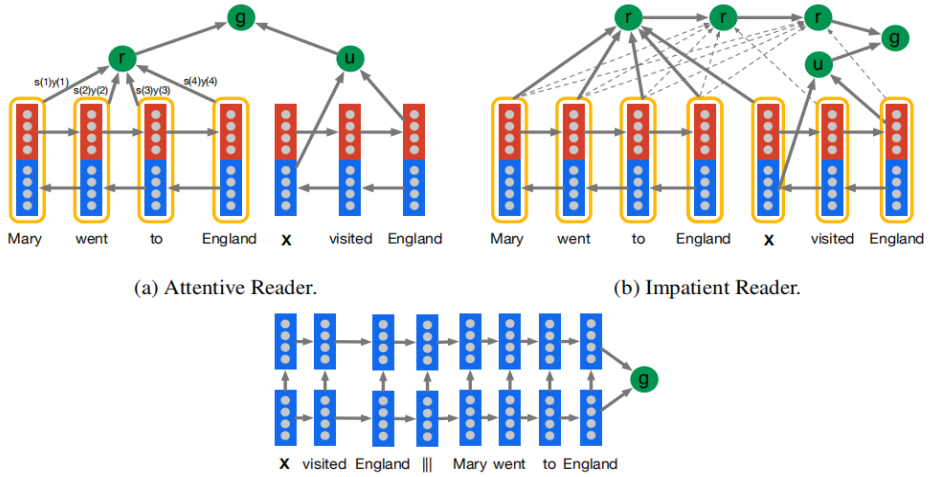
[1] The Deep LSTM Reader-将query和document用分隔符连接起来，作为一整个sequence送入LSTM，然后把最后一个时间步的每一层连接起来。

$$g^{LSTM}(d,q) = y(|d|,|q|)$$

[2] The Attentive Reader-对于document中的每一个token都进行相似度计算，得到一个“attention”权重，按权重相加。

$$g^{AR}(d,q) = tanh(W_{rg}r + W_{ug}u)$$

[3] The Impatient Reader-Attentive读者可以专注于上下文文档中最有可能为查询提供答案的段落。我们可以通过对模型进行更深层次的改造，使模型具有更强的可重复性从文档中读取每个查询token。每检索到query中的一个token就将已检索过的query中的tokens和 document都再看一遍。 $g^{IR}(d,q) = tanh(W_{rg}r(|q|) + W_{qg}u)$




(a) Attentive Reader.

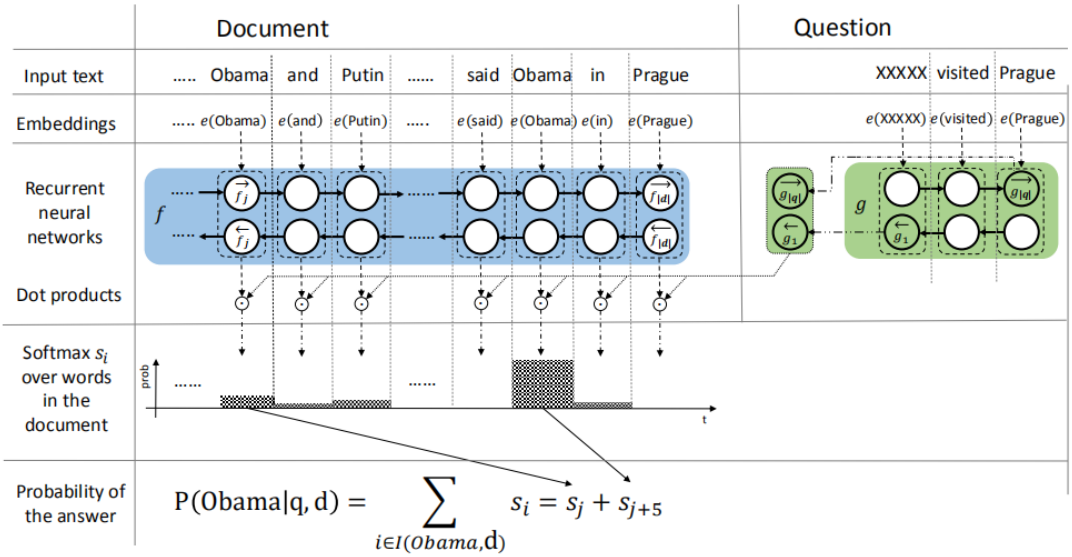
(b) Impatient Reader.

(c) A two layer Deep LSTM Reader with the question encoded before the document.

图 6: 三种计算方式

# 4 Text Understanding with the Attention Sum Reader Network

提出了一个新的、简单的模型，它使用注意直接从上下文中选择答案，而不是像通常那样使用文档中的混合词表示来计算答案。




工作流程：

（1）计算query的embedding

（2）计算document中每个词的contextual embedding

（3）点乘，选择最有可能的答案

# 5 A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task

主要目标是，弄清楚究竟需要对语言多深的理解才能够完成Reading comprehension的任务。其次提出一个超过目前state-of-art 5%的模型，并且认为这是这个任务上所能够达到的最高水平。

1. Entity-Centric Classifier-为每一个entity设计一个向量$f_{p,q}(e)$，以及学习一个权重矩阵$\theta$使得正确答案的分数比其它答案都要高。

$$\theta^T f_{p,q}(a) > \theta^T f_{p,q}(e), \forall e \in E \cap \backslash \{a\}$$

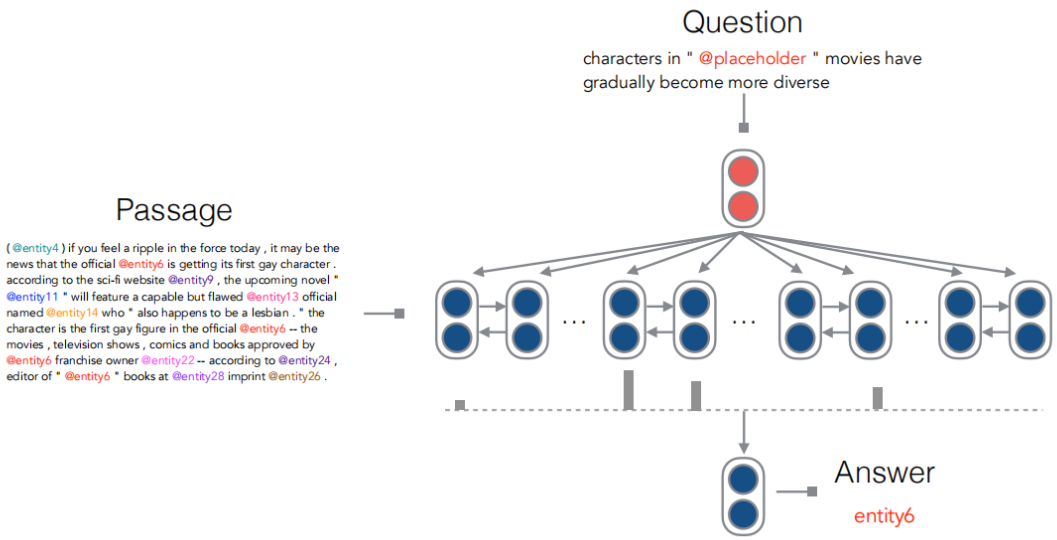



图 7: 网络结构

2. 整体类似Attentive Readder模型，区别就是改进了Attentive Reader里面计算g(d,q)函数的计算方法，将tanh替换为先计算softmax再加权求和。

# 6 Long Short-Term Memory-Networks for Machine Reading

本文针对RNN对于处理固定输入的局限性，提出了一种基于LSTM的改进模型。

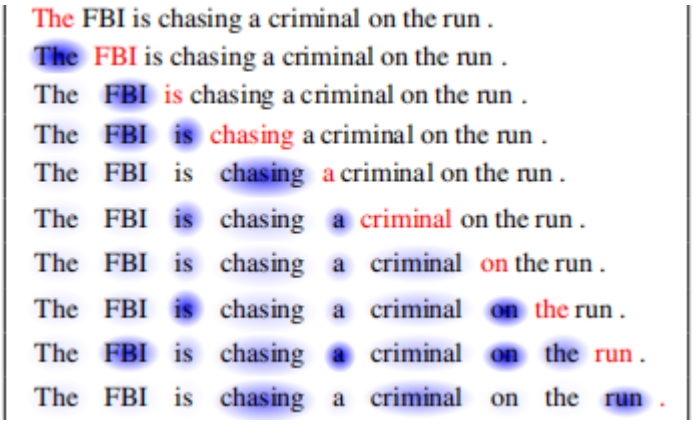



图 8: example

1. 本文所提出的结构的中心思想是使用多个记忆槽(memory slot)来存储输入的表示，而每个槽的read和write操作被建模为Attention机制。Attention不需要监督信息，能够发现输入的token之间的关系。

2. Long Short-Term Memory (LSTM)

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} W \cdot [h_{t-1}, x_t]$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

$$h_t = o_t \odot tanh(c_t)$$

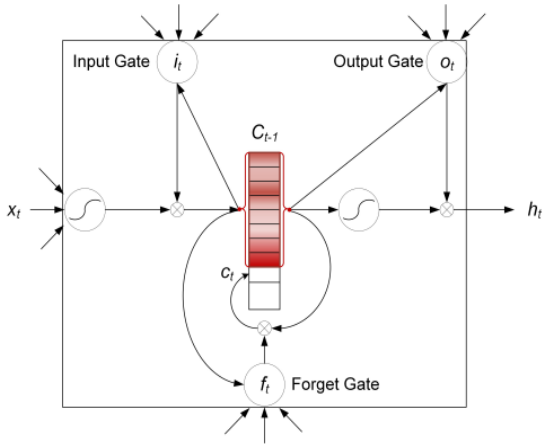
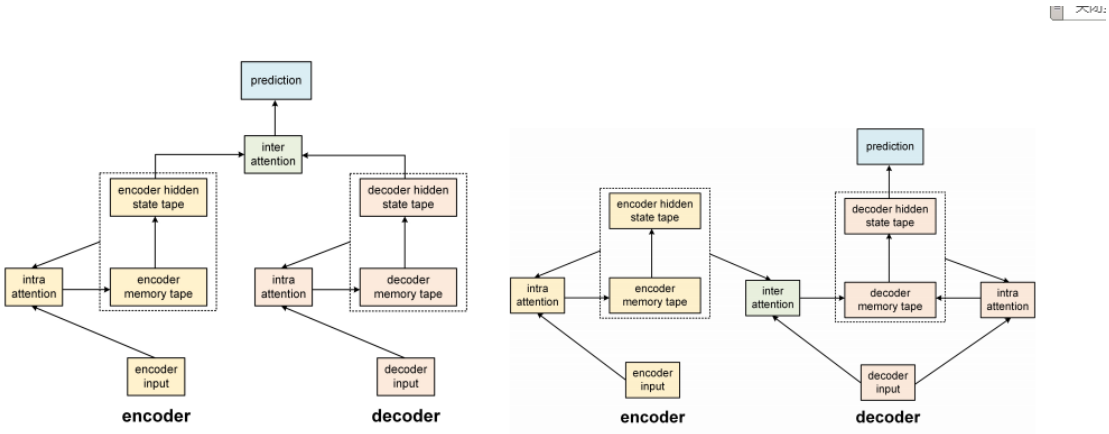



Figure 2: Long Short-Term Memory-Network. Color indicates degree of memory activation.

图 9: 网络结构

3. Long Short-Term Memory-Network（LSTMN）
   将标准LSTM中的memory cell替换为memory network，这样就可以将每一个input的embedding都存入一个不同的memory slot直到达到数量上限。而且这样的设计还可以使得能够通过一层neural attention来推理tokens之间的关系。因为在read的时候就是根据目前的token，在所有存储的memories中寻找有用的内容。
   对于input gate和forget gate的更新则首先将所有的memories加权相加，剩余的步骤一样。




(a) Decoder with shallow attention fusion. (b) Decoder with deep attention fusion.

图 10: seq2seq的两种形式

# 7 Key-Value Memory Networks for Directly Reading Documents

提出了一种全新的键值对记忆神经网络，来改善直接阅读文档的能力。因为它可以在读取memory操作中的寻址（K）和输出（V）两个阶段使用不同的编码。

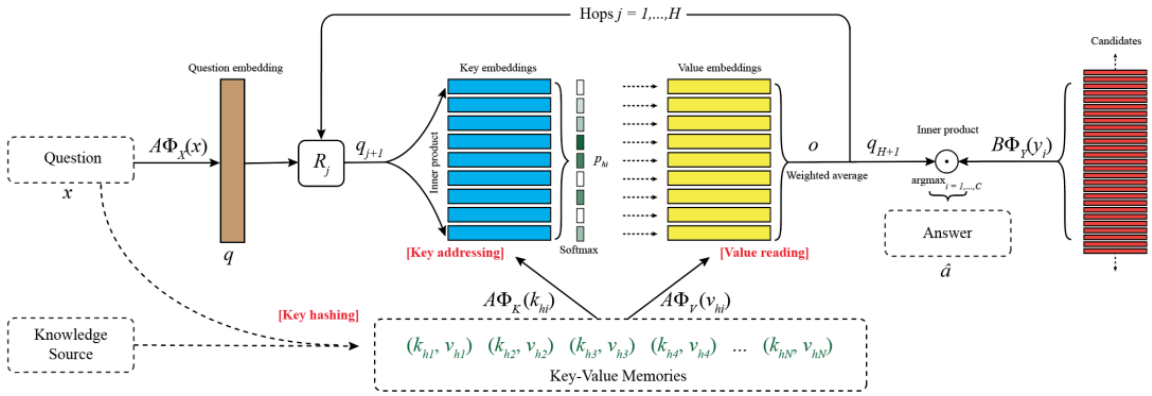



图 11: 网络结构

1. 首先，将事实存储在键值结构化内存中，然后再对它们进行推理，以预测答案。
2. 存储器的设计使得模型能够学习使用键来寻址与问题相关的存储器，相应的值随后被返回。此结构允许模型为当前任务编码先验知识，并利用键和值之间可能存在的复杂转换。
3. 在每一step中，从memory收集到的信息都会被添加到原始的query中。将memory slots定义为一些成对的向量$(k_1, v_1)..., (k_M, v_M)$,然后将问题定义为x。寻址及读取步骤如下。
4. Key Hashing
   question可以被用来预先选择一个大数组的一个子集，比如通过倒排索引的方式。
5. Key Addressing
   每一个key都会被指定一个对应的概率：
   $$p_{h_i} = Softmax(A\Phi_X(x) \cdot A\Phi_K(k_{h_i}))$$
   Φ是一个D维的特征映射(比如可以根据词袋模型来进行映射)，而A是一个$d \times D$维矩阵。
6. Value Reading
   根据对应key分配的概率，最后的value加权相加。
   $$o = \sum_i p_{h_i} A\Phi_V(v_{h_i})$$
   在一开始$q = A\Phi_X(x)$,当接受到ressult o之后，query就更新成$q_2 = R_1(q + o)$,R是一个$d \times d$的矩阵。一直重复下去，在每一个hop使用一个不同的矩阵$R_j$,直到H个hops之后。因此寻址的公式就变成了：
   $$p_{h_i} = Softmax(q_{j+1}^T \cdot A\Phi_K(k_{h_i}))$$

这样做的原因是可以将新的证据合并到查询中，以便在后续访问中集中并检索更相关的信息。然后就可以计算得到

$$\hat{a} = argmax_{i=1,\ldots,C} Softmax(q_{H+1}^T B \Phi_Y(y_i))$$

# 8 Modeling Human Reading with Neural Attention

人类在阅读的时候，有时候会仔细看一些单词，有时候又会快速的略过一些单词。在这篇论文中，提出了一个新的方法来对这种行为建模，使用无监督的方法，将自动编码与neural attention结合起来。这个模型解释了，人类在阅读的精确性与所消耗的注意力之间的一个权衡。


**Figure 2:** Top: Heatmap showing human fixation probabilities, as estimated from the ten readers in the Dundee corpus. In cases of track loss, we replaced the missing value with the corresponding reader's overall fixation rate. Bottom: Heatmap showing fixation probabilities simulated by NEAT. Color gradient ranges from blue (low probability) to red (high probability); words without color are at the beginning or end of a sequence, or out of vocabulary.

图 12: 根据模型预测的人类阅读注意力模型和真实的人类阅读注意力模型

1. 实现这个模型的一个关键是计算“surprisal”，通过语言模型来计算。它测量一个单词在上下文中的可预测性，定义为给定前面单词的当前单词的条件概率的负对数。

2. NEAT (NEural Attention Tradeoff).Encoder和Decoder都使用LSTM。Reader每次读取一个单词$w_i$然后将单词序列转换为一系列向量$h_0, ..., h_3$ 每一个向量都是固定维度的对所有已经读过的单词的编码。最后一个$h_3$则会作为decoder的输入。因为它能够读取到encoder的输出，所以理论上可以分配最大的概率给实际上集中注意力读的那些单词序列。

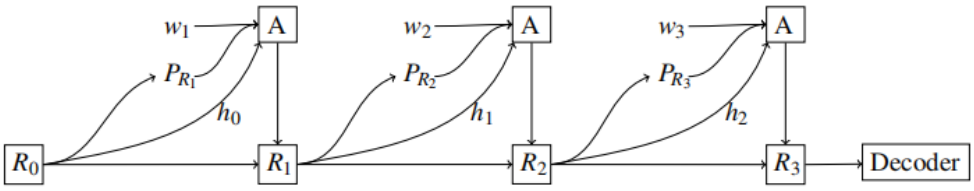



**Figure 1:** The architecture of the proposed model, reading a three-word input sequence $w_1, w_2, w_3$. $R$ is the reader network and $P_R$ the probability distribution it computes in each time step. $A$ is the attention network. At each time step, the input, its probability according to $P_R$, and the previous state $h_{i-1}$ of $R$ are fed into $A$, which then decides whether the word is read or skipped.

图 13: Encoder结构图

3. first:对下一个step要出现的单词进行概率预测$P_R$。
   second：对skip操作建模为只有部分单词能够被送入Reader，而在其它情况会被送入一些无用的特殊向量。Attention模块A决定下一个单词是否被送入Reader 我们假设，选择跳过的单词不仅要考虑到先前的上下文，而且还要考虑到单词本身的预览（即能否通过前文预测到）。

4. 计算surprisal

$$Surp(w_i|w_{1...i-1} = -logP_R(w_i|w_{1...i-1}, \omega_{1...i-1})$$

模型的目标函数

$$L(\omega|w,\theta) = -\sum logP_R(w_i|w_{1...i-1}, \omega_{1...i-1}; \theta) - \sum logP_{decoder}(w_i|w_{1...i-1}; h_N; \theta)$$

# 9 LEARNING RECURRENT SPAN REPRESENTATIONS FOR EXTRACTIVE QUESTION ANSWERING

RASOR: RECURRENT SPAN REPRESENTATION,能够显示的计算候选答案。整个网络结构分为以下几个部分。

1. Question-Focused Passage Word Embedding

[1] Question-independent passage word embedding:lookup 预训练的词向量。

[2] Passage-aligned question representation:在这个数据集中，问题段落对通常在正确的答案跨度附近包含大量的词汇重叠或相似。作者提出来；利用passage soft-alignment得到question representation，利用的是neural attention。这里FFNN表示的是前向神经网络中的全连接层，表示非线性的映射

$$s_{ij} = FFNN(p_i) \cdot FFNN(q_j)$$

$$a_{ij} = \frac{exp(s_{ij})}{\sum_{k=1} nexp(s_{ik})}$$

$$q_i^{align} = \sum_{j=1}^{n} a_{ij} q_j$$

[3] Passage-independent question representation：基于neural attention得到通过Bi-LSTM的question向量：

$$\begin{aligned} \{q'_1, \ldots, q'_n\} &= \text{BILSTM}(\mathbf{q}) \\ s_j &= w_q \cdot \text{FFNN}(q'_j) && 1 \le j \le n \\ a_j &= \frac{\exp(s_j)}{\sum_{k=1}^{n} \exp(s_k)} && 1 \le j \le n \\ q^{indep} &= \sum_{j=1}^{n} a_j q'_j \end{aligned}$$

根据以上三点，最后的$p_i$向量为以上三个向量连接起来。

2. RaSoR:Recurrent Span Representation

$$\{p_1^{*'}, ..., p_m^{*'}\} = BILSTM(\{p_1^*, ..., p_m^*\})$$

$$h_{a_{start}} = [p_{a_{start}}^{*'}, p_{a_{end}}^{*'}]$$

3. Score Answer Span

$$s_a = w_a \cdot FFNN(h_a)$$

$$P(a|q,p) = \frac{exp(s_a)}{\sum_{a' \in A(p)} exp(s_{a'})}$$

$$f(p,q) = \underset{a \in A(p))}{argmax} P(a|p,q)$$

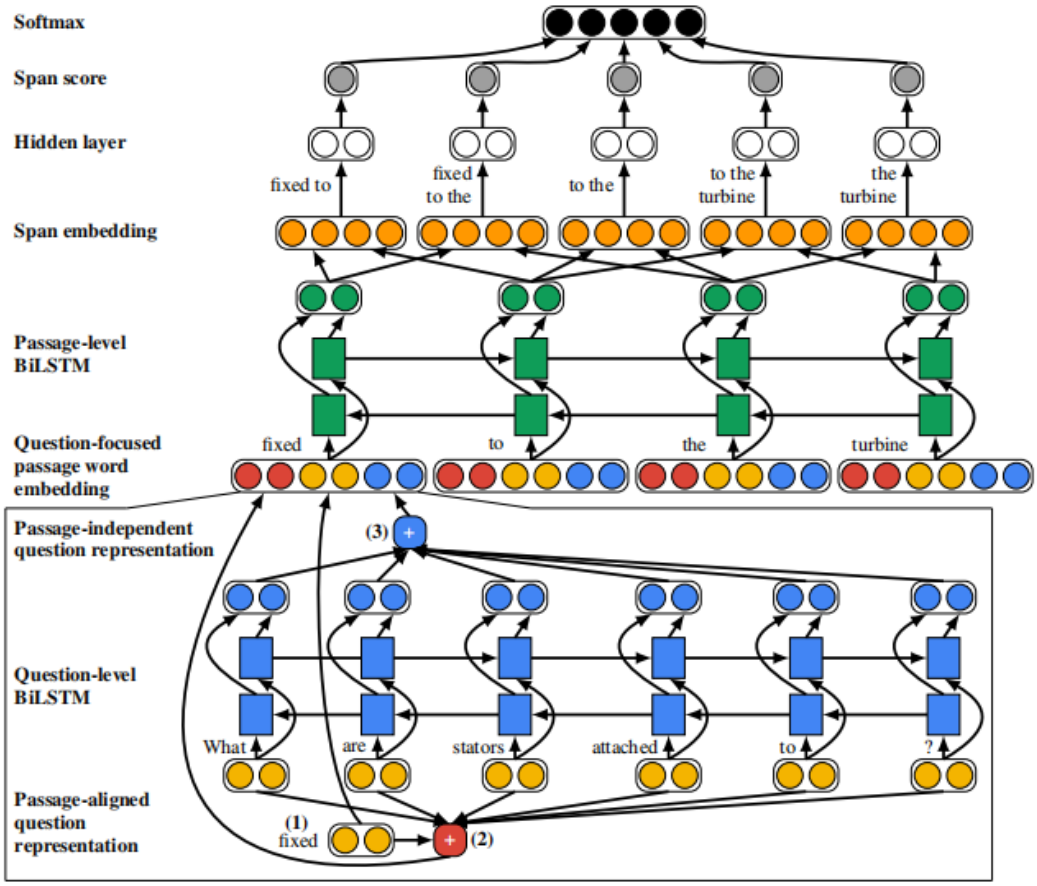



Figure 1: A visualization of RASOR, where the question is *"What are the stators attached to?"* and the passage is *"...fixed to the turbine ..."*. The model constructs question-focused passage word embeddings by concatenating **(1)** the original passage word embedding, **(2)** a passage-aligned representation of the question, and **(3)** a passage-independent representation of the question shared across all passage words. We use a BiLSTM over these concatenated embeddings to efficiently recover embedding representations of all possible spans, which are then scored by the final layer of the model.

图 14: 网络结构图

# 10 Multi-Perspective Context Matching for Machine Comprehension

提出了MPCM模型，在模型中首先根据question乘以一个相关的权重矩阵得到passage向量，然后送进BiLSTM。与以往不同的是，这篇文章预测的是答案的起始位置和结束位置。

1. 对答案的起点和终点做了一个独立性假设，因此大大简化了模型。

$$A^* = \underset{1 \le a_b \le a_e \le N}{argmax}\ Pr(a_b|Q,P)Pr(a_e|Q,P)$$

模型由以下六层组成。

2. Word Representation Layer
word Embedding+character composed embedding(通过将组成word的每个character送入LSTM计算得到)

3. Filter Layer

$$r_{ij} = \frac{q_i^T p_j}{\|q_i\| \cdot \|p_j\|}$$
$$r_j = max r_{ij}$$
$$p'_j = r_j \cdot p_j$$

4. Context Representation Layer
送入BiLSTM进行计算

5. Multi-Perspective Context Matching Layer

$$m_j = [\overrightarrow{m}_j^{full}; \overleftarrow{m}_j^{full}; \overrightarrow{m}_j^{max}; \overleftarrow{m}_j^{max}; \overrightarrow{m}_j^{mean}; \overleftarrow{m}_j^{mean}]$$

$$\overrightarrow{\boldsymbol{m}}_j^{full} = f_m(\overrightarrow{\boldsymbol{h}}_j^p, \overrightarrow{\boldsymbol{h}}_M^q; \boldsymbol{W}^1) \qquad \overrightarrow{\boldsymbol{m}}_j^{max} = \max_{i \in (1...M)} f_m(\overrightarrow{\boldsymbol{h}}_j^p, \overrightarrow{\boldsymbol{h}}_i^q; \boldsymbol{W}^3)$$
$$\overleftarrow{\boldsymbol{m}}_j^{full} = f_m(\overleftarrow{\boldsymbol{h}}_j^p, \overleftarrow{\boldsymbol{h}}_1^q; \boldsymbol{W}^2) \qquad \overleftarrow{\boldsymbol{m}}_j^{max} = \max_{i \in (1...M)} f_m(\overleftarrow{\boldsymbol{h}}_j^p, \overleftarrow{\boldsymbol{h}}_i^q; \boldsymbol{W}^4)$$

(a) full (b) max

$$\overrightarrow{\boldsymbol{m}}_j^{mean} = \frac{1}{M}\sum_{i=1}^{M} f_m(\overrightarrow{\boldsymbol{h}}_j^p, \overrightarrow{\boldsymbol{h}}_i^q; \boldsymbol{W}^5)$$
$$\overleftarrow{\boldsymbol{m}}_j^{mean} = \frac{1}{M}\sum_{i=1}^{M} f_m(\overleftarrow{\boldsymbol{h}}_j^p, \overleftarrow{\boldsymbol{h}}_i^q; \boldsymbol{W}^6)$$

(c) mean

6. Aggregation Layer
运用BiLSTM，使得passage的每一个Step都能够和周围联系起来。

7. Prediction Layer

使用两个不同的前馈网络，分别送入softmax，得到结果。

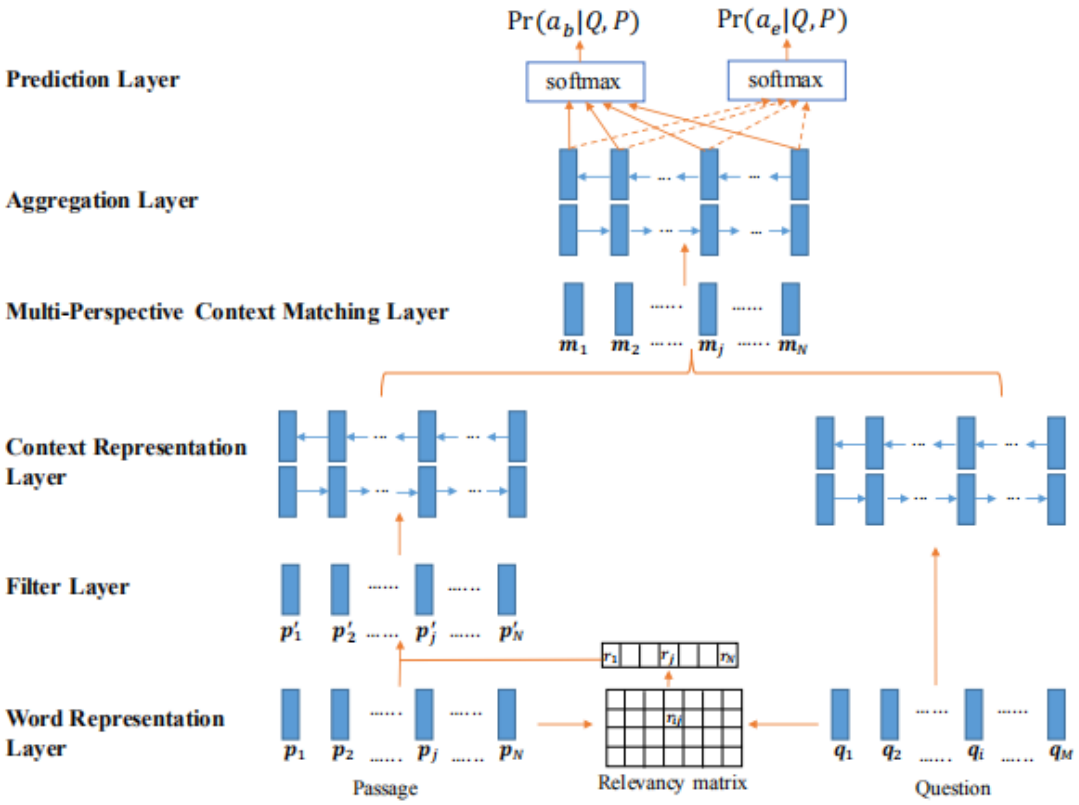



图 15: 网络结构

# 11 Natural Language Comprehension with the EpiReader

将回答问题分为两个步骤：1Extractor选出候选答案；2Reasoner对候选答案打分选出最终答案。

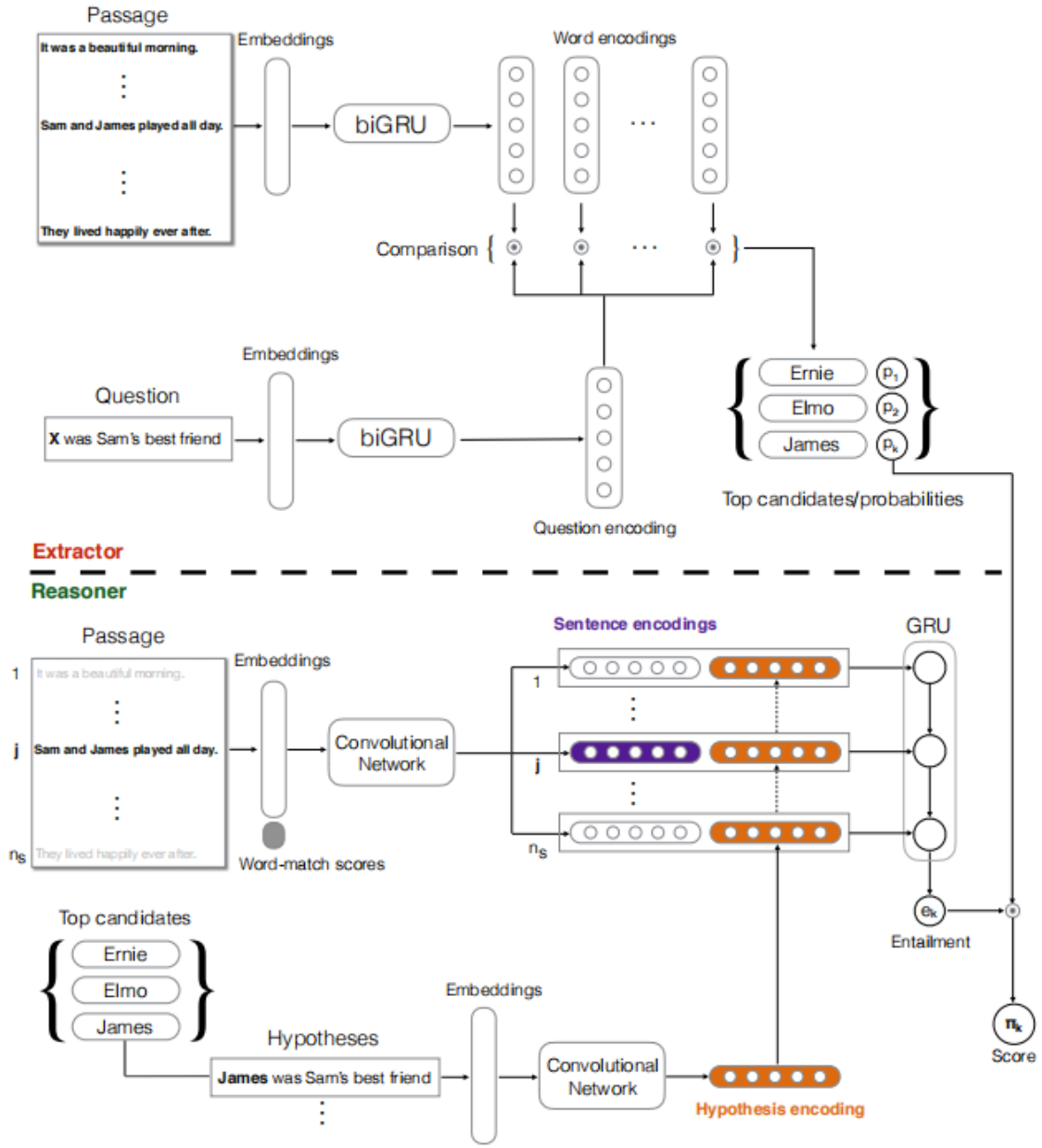



图 16: 网络结构

1. The Extractor

   将passage和question分别送入对应的BiGRU，取出passage对应的每一个step的输出$f(t_i) \in \mathbb{R}^{2d}$,以及question最后一个step的输出$g(Q) \in \mathbb{R}^{2d}$,然后计算相似度

$$s_i \propto exp(f(t_i) \cdot g(Q))$$

从而得到每个单词作为答案（或者说candidate）的概率：

$$P(w|P,Q) = \sum_{i:t_i=w} s_i$$

2. The Reasoner

Reasoner通过“文本蕴含”来推测假设$H_k$和passage的关系。对passage的每句话以及每个假设首先进行Embedding操作（根据预训练的W来lookup）然后送入convolutional architecture。

[1] 矩阵M：$2 \times |S_i|$,第一行是sentence中每一个embedding和candidate的内积，第二行是sentence中每个embedding和question中每个单词的内积的最大值。

[2] 矩阵$S_i$加上M（应该是拼接？）

[3] 卷积操作，augmented $S_i$的filters大小是$F^S \in \mathbb{R}^{(D+2)\times m}$, $H_k$的filters大小则是$F^H \in \mathbb{R}^{D\times m}$，然后经过池化层得到text的表示$r_{S_i} \in \mathbb{R}^{N_F}$，还有假设的表示$r_{H_k} \in \mathbb{R}^{N_F}$ 然后就可以计算相似度

$$\zeta = r_{S_i}^T R r_{H_k}$$

这里的R是可以通过train学得的。将相似度与之前计算的表示连接起来。

$$x_{ik} = [\zeta, r_{S_i}, r_{H_k}]^T$$

将$N_S$长度的序列送入GRU，将GRU的最后一个step的hidden state送入一个fully-connected layer，得到一个标量$y_k$，$e_k \propto exp(y_k)$

3. Combine

$$\mathcal{L}_E = \mathop{\mathbb{E}}_{(\mathcal{Q},\mathcal{T},a^*,A)} [-\log P(a^* \mid \mathcal{T}, \mathcal{Q})]$$

(a) Extractor

$$\mathcal{L}_R = \mathop{\mathbb{E}}_{(\mathcal{Q},\mathcal{T},a^*,A)} \left[ \sum_{\hat{a}_i \in \{\hat{a}_1,\ldots,\hat{a}_K\} \setminus a^*} [\gamma - \pi^* + \pi_{\hat{a}_i}]_+ \right]$$

(b) Reasoner

$$\mathcal{L}_{ER} = \mathcal{L}_E + \lambda \mathcal{L}_R$$

(c) total