论文阅读笔记

Step6

MF1833063, 史鹏, spwannasing@gmail.com

2019 年 7 月 15 日

1 Cognitive Graph for Multi-Hop Reading Comprehension at Scale

这篇文章基于Bert和GNN，在迭代中逐步构建出cognitive graph：$\mathcal{G}$，图中的每一个节点都和一个实体或者一个可能的答案有关。由两部分组成：implicit extraction（System 1）和explicit reasoning（System 2）。

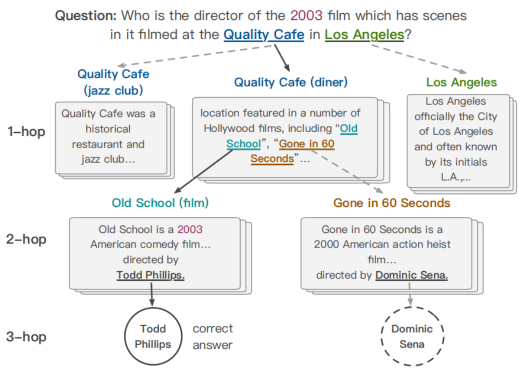


Figure 1: An example of cognitive graph for multi-hop QA. Each *hop node* corresponds to an entity (e.g., "Los Angeles") followed by its introductory paragraph. The circles mean *ans nodes*, answer candidates to the question. Cognitive graph mimics human reasoning process. Edges are built when calling an entity to "mind". The solid black edges are the correct reasoning path.

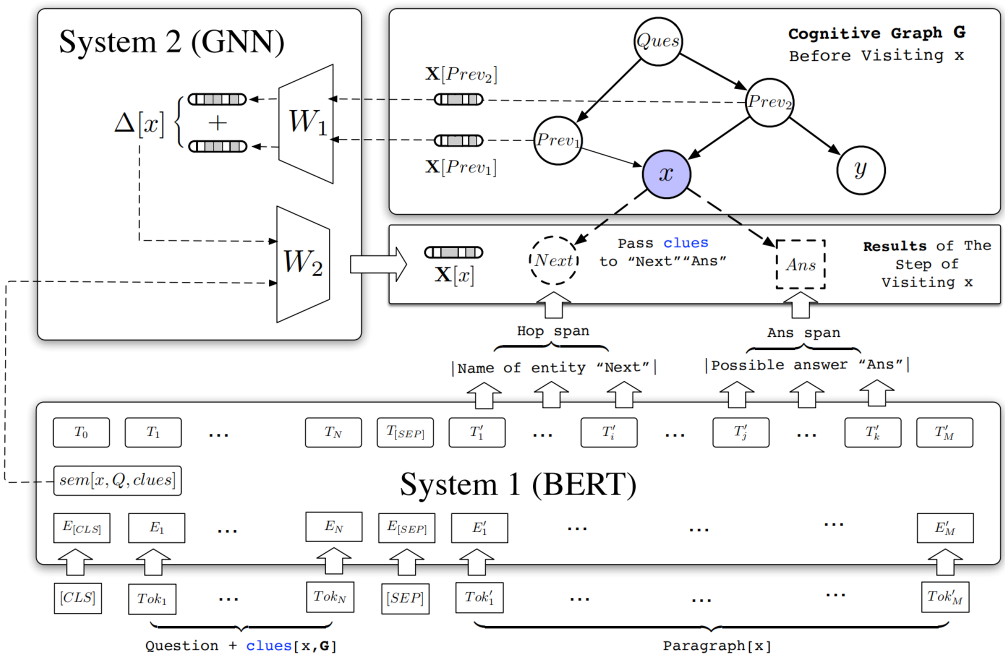


Figure 2: Overview of CogQA implementation. When visiting the node x, System 1 generates new hop and answer nodes based on the $clues[x, \mathcal{G}]$ discovered by System 2. It also creates the inital representation $sem[x, Q, clues]$, based on which the GNN in System 2 updates the hidden representations $\mathbf{X}[x]$.

Algorithm 1: Cognitive Graph QA

Input:
System 1 model $\mathcal{S}_1$, System 2 model $\mathcal{S}_2$,
Question Q, Predictor $\mathcal{F}$,Wiki Database $\mathcal{W}$

Initialize cognitive graph $\mathcal{G}$ with entities mentioned in Q and mark them *frontier nodes*
repeat
  pop a node x from frontier nodes
  collect $clues[x, \mathcal{G}]$ from predecessor nodes of x
   // eg. $clues$ can be sentences where x is mentioned
  fetch $para[x]$ in $\mathcal{W}$ if any
  generate $sem[x, Q, clues]$ with $\mathcal{S}_1$ // initial $\mathbf{X}[x]$
  if *x is a hop node* **then**
    find hop and answer spans in $para[x]$ with $\mathcal{S}_1$
    for *y in hop spans* **do**
      if *$y \notin \mathcal{G}$ and $y \in \mathcal{W}$* **then**
        create a new hop node for y
      if *$y \in \mathcal{G}$ and edge$(x, y) \notin \mathcal{G}$* **then**
        add edge (x, y) to $\mathcal{G}$
        mark node y as a frontier node
    end
    for *y in answer spans* **do**
      add new answer node y and edge (x, y) to $\mathcal{G}$
    end
  end
  update hidden representation $\mathbf{X}$ with $\mathcal{S}_2$
until *there is no frontier node in $\mathcal{G}$ or $\mathcal{G}$ is large enough*;
Return $\underset{\text{answer node } x}{\arg\max}\ \mathcal{F}(\mathbf{X}[x])$

[注] 本文选取的数据集是HotpotQA，HotpotQA 是一个大型问答数据集，它包含约 11.3 万个具备上述特征的问答对。也就是说，这些问题要求问答系统能够筛选大量的文本文档，以找到与生成答案相关的信息，并对找到的多个支撑性事实进行多步推理，从而得出最终答案。

System 1从段落中提取与问题相关的实体和answer candidate，并对其语义信息进行编码。提取的实体被组织成一个Cognitive Graph。然后，系统2对图进行推理，并收集线索指导系统1更好地提取下一跳实体。

System 1(Bert)：

$$\underbrace{[CLS]Question[SEP]\,\text{clues}[x,\mathcal{G}][SEP]}_{\text{Sentence } A}\ \underbrace{\text{Para}[x]}_{\text{Sentence } B} \tag{1.1}$$

System 2(GNN)：

$$\begin{aligned}\Delta &= \sigma\left(\left(AD^{-1}\right)^{T}\sigma\left(\mathbf{X}W_1\right)\right)\\ \mathbf{X}' &= \sigma\left(\mathbf{X}W_2+\Delta\right)\end{aligned} \tag{1.2}$$

2 Exploiting Explicit Paths for Multi-hop Reading Comprehension

本文基于WikiHop数据集，在该数据集中问题以三元组的形式出现$(h_e, r, ?)$，h_e代表head entity，r代表head entity和未知的tail entity之间的关系。任务是从给定的candidates集合中选出一个答案:$(c_1, c_2, \ldots, c_N)$。

所做的工作是在预测答案的同时，将推理的path展示出来。方法很intuitive，但难的是想到并去做这个工作。

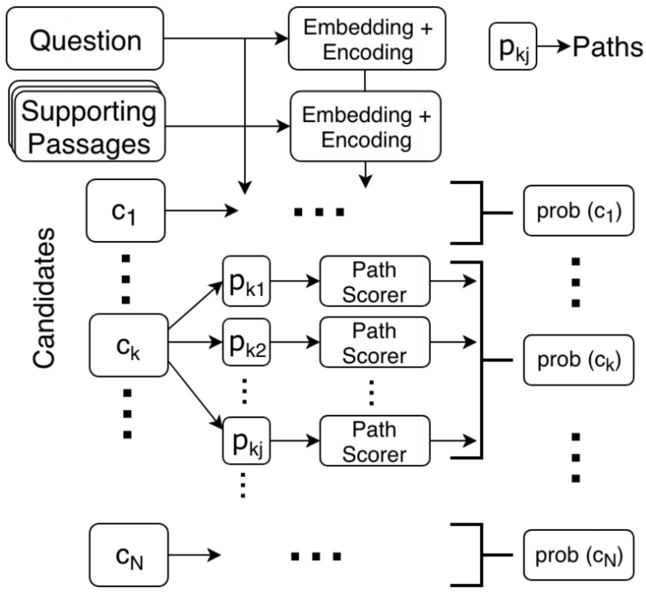


图 1: Architecture of the proposed model

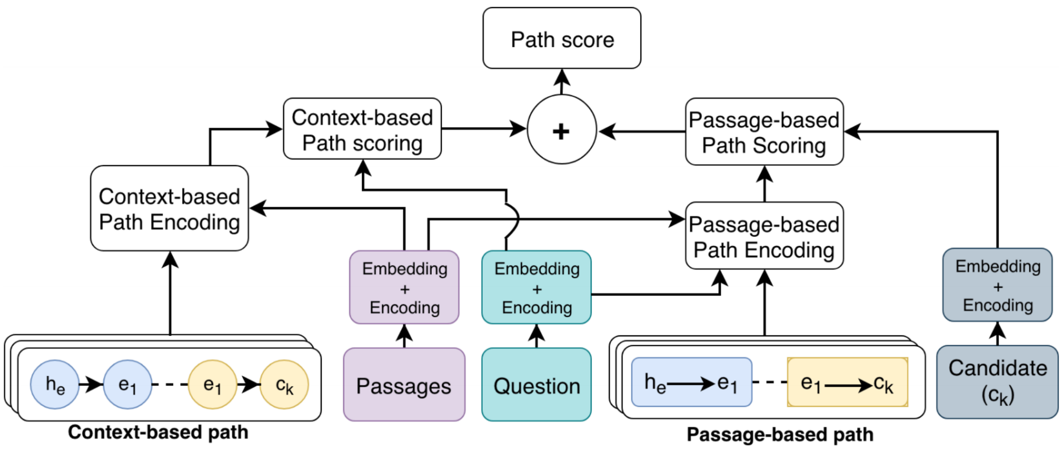


图 2: Architecture of the proposed path scoring module

模型的流程是：

1. 提取Path 在含有h_e的第一篇文章中，找出在那句话或者下一句话中出现的所有实体，然后在其它的passage中寻找这些实体，如果该passge也包含candidate中的单词那么则构建出了一个path。为每一个candidate构建一个从h_e的路径。

2. 对所有的path进行encoding以及score，选出answer以及输出path。

[1] Context-based Path Encoding:

对于2-hop的path：$(h_e, e_1), (e_1, c_k)$:

$$\mathbf{g}_{h_e} = \mathbf{s}_{p_1,i_1} \| \mathbf{s}_{p_1,i_2} \tag{2.1}$$

$$\mathrm{FFL}(\mathbf{a}, \mathbf{b}) = \tanh\left(\mathbf{a}\mathbf{W}_a + \mathbf{b}\mathbf{W}_b\right) \tag{2.2}$$

$$\mathbf{r}_{h_e,e_1} = \mathrm{FFL}\left(\mathbf{g}_{h_e}, \mathbf{g}_{e_1}\right) \tag{2.3}$$

$$\mathbf{x}_{ctx} = \mathrm{FFL}\left(\mathbf{r}_{h_e,e_1}, \mathbf{r}_{e_1,c_k}\right) \tag{2.4}$$

[2] Passage-based Path Encoding:

首先计算相似矩阵：$\mathbf{A}_p \in \mathbb{R}^{T\times U}$，然后分别计算question-aware passage和passage-aware question：$\mathbf{S}_p^{q_1} = \mathbf{AQ}$和$\mathbf{Q}_p = \mathbf{A}^\top \mathbf{S}_p$，根据更新的question表示再计算$\mathbf{S}_p^{q_2} \in \mathbb{R}^{T\times H}$，其中$\mathbf{S}_p^{q_2} = \mathbf{A}\mathbf{Q}_p$

然后将两次计算的结果拼接：$S_p^q \in \mathbb{R}^{T\times 2H} = \mathbf{S}_p^{q_1} \| \mathbf{S}_p^{q_2}$。

$$\begin{aligned} a_t^p &\propto \exp\left(\mathbf{s}_{p,t}^q \mathbf{w}^\top\right) \\ \tilde{\mathbf{s}}_p &= \mathbf{a}^p \mathbf{S}_p^q \end{aligned} \tag{2.5}$$

$$\mathbf{x}_{psg} = \mathrm{FFL}\left(\tilde{\mathbf{s}}_{p1}, \tilde{\mathbf{s}}_{p_2}\right) \tag{2.6}$$

[3] Path Scoring:

$$\tilde{\mathbf{q}} = \left(\mathbf{q}_0 \| \mathbf{q}_U\right) \mathbf{W}_q \tag{2.7}$$

$$\mathbf{y}_{x_{ctx},q} = \mathrm{FFL}\left(\mathbf{x}_{ctx}, \tilde{\mathbf{q}}\right) \tag{2.8}$$

$$z_{ctx} = \mathbf{y}_{x_{ctx},q} \mathbf{w}_{ctx}^\top \tag{2.9}$$

$$z_{psg} = \tilde{\mathbf{c}}_k \mathbf{x}_{psg}^\top \tag{2.10}$$

$$z = z_{ctx} + z_{psg} \tag{2.11}$$

3 Relational inductive biases, deep learning, and graph networks

主要介绍了一些GNN的相关应用方法，是Section 1 的论文中System 2的组成部分。

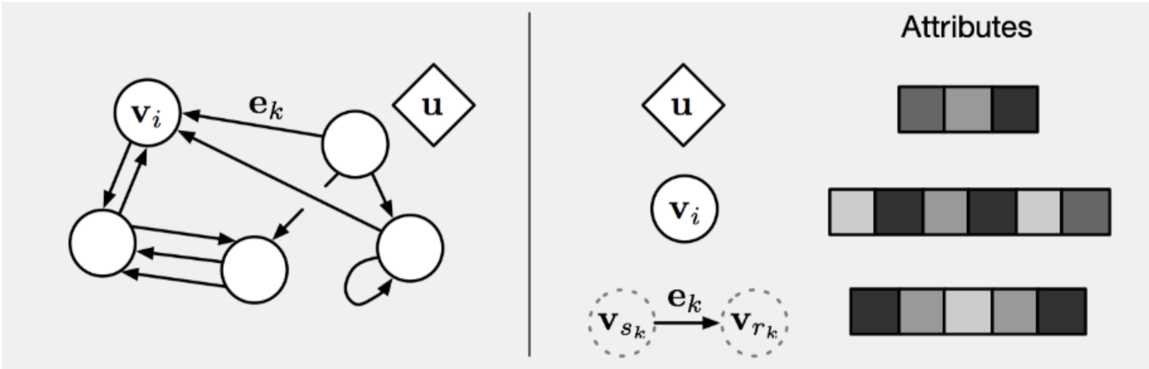


图 3: GNN的定义

GNN中节点更新方式：

$$
\begin{aligned}
\mathbf{e}'_k &= \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{sk}, \mathbf{u}\right) & \overline{\mathbf{e}}'_i &= \rho^{e\rightarrow v}\left(E'_i\right) \\
\mathbf{v}'_i &= \phi^v\left(\overline{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) & \overline{\mathbf{e}}' &= \rho^{e\rightarrow u}\left(E'\right) \\
\mathbf{u}' &= \phi^u\left(\overline{\mathbf{e}}', \overline{\mathbf{v}}', \mathbf{u}\right) & \overline{\mathbf{v}}' &= \rho^{v\rightarrow u}\left(V'\right)
\end{aligned}
\tag{3.1}
$$

Algorithm 1 Steps of computation in a full GN block.

```
function GraphNetwork(E, V, u)
    for k ∈ {1 ... N^e} do
        e'_k ← φ^e (e_k, v_{r_k}, v_{s_k}, u)            ▷ 1. Compute updated edge attributes
    end for
    for i ∈ {1 ... N^n} do
        let E'_i = {(e'_k, r_k, s_k)}_{r_k=i, k=1:N^e}
        ē'_i ← ρ^{e→v} (E'_i)                            ▷ 2. Aggregate edge attributes per node
        v'_i ← φ^v (ē'_i, v_i, u)                        ▷ 3. Compute updated node attributes
    end for
    let V' = {v'}_{i=1:N^v}
    let E' = {(e'_k, r_k, s_k)}_{k=1:N^e}
    ē' ← ρ^{e→u} (E')                                    ▷ 4. Aggregate edge attributes globally
    v̄' ← ρ^{v→u} (V')                                    ▷ 5. Aggregate node attributes globally
    u' ← φ^u (ē', v̄', u)                                 ▷ 6. Compute updated global attribute
    return (E', V', u')
end function
```

4 BAM! Born-Again Multi-Task Networks for Natural Language Understanding

多任务模型的性能通常比它们的单任务对应对象差,本文就提出了了一种通用的解决方案。通过Knowledge Distillation，Multi-Task Distillation和Teacher Anneall来解决这个问题。主要思想是对于每一个Task，由Teacher(Teacher的结构和student一样)来“教导”它们该如何去做，在训练一定时间之后，“student”再去向golden answer学习。理由是teacher提供的answer的预测是一个分布，比golden answer的one-hot能够提供更多的信息。比如在图像分类的时候，该图片为马，one-hot只会标注这是一只马，而Teacher预测的distribution不仅能知道是马，还能够知道，相比于自行车飞机什么的，更有可能是一只驴。

1.Kownledge Distillation:

传统的one-hot：$\mathcal{L}(\theta)=\sum_{x_\tau^i, y_\tau^i \in \mathcal{D}_\tau} \ell\left(y_\tau^i, f_\tau\left(x_\tau^i, \theta\right)\right)$

Distillation:$\mathcal{L}(\theta)=\sum_{x_\tau^i, y_\tau^i \in \mathcal{D}_\tau} \ell\left(f_\tau\left(x_\tau^i, \theta^{\prime}\right), f_\tau\left(x_\tau^i, \theta\right)\right)$

2.Multi-Task Distillation:

$$\mathcal{L}(\theta)=\sum_{\tau \in \mathcal{T}} \sum_{x_\tau^i, y_\tau^i \in \mathcal{D}_\tau} \ell\left(f_\tau\left(x_\tau^i, \theta_\tau\right), f_\tau\left(x_\tau^i, \theta\right)\right)$$

3.Teacher Annealing:

$$\ell\left(\lambda y_\tau^i+(1-\lambda) f_\tau\left(x_\tau^i, \theta_\tau\right), f_\tau\left(x_\tau^i, \theta\right)\right)$$

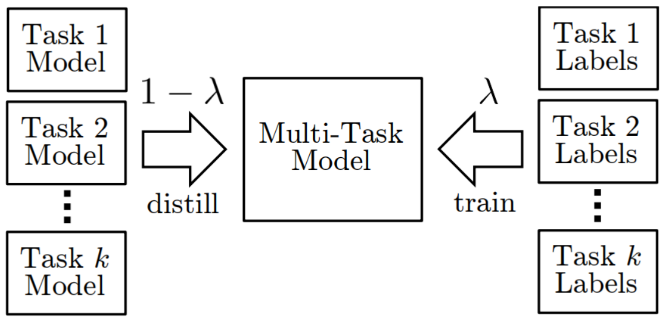
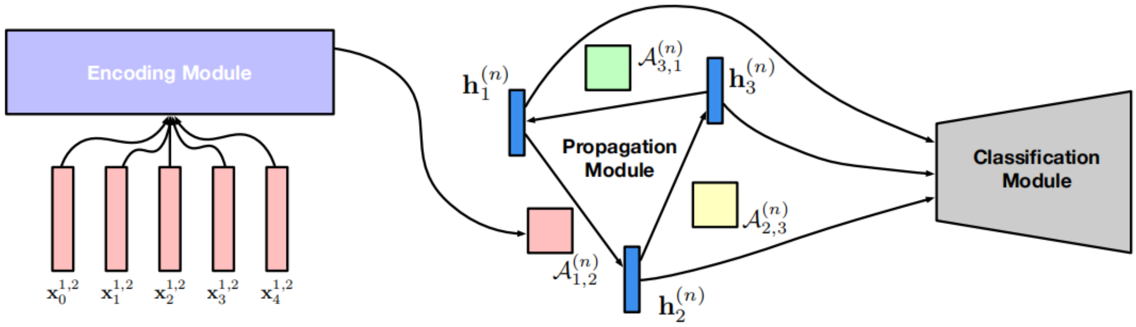


Figure 1: An overview of our method. λ is increased linearly from 0 to 1 over the course of training.

5 Graph Neural Networks with Generated Parameters for Relation Extraction

本文是刘知远老师组里的工作，是GNN在推理和关系抽取上的一次应用：Graph Neural Network with Generated Parameter（GP-GNNs）。



结构分为三部分：

1. Encoding Module:

$$s = (x_0, x_1, \ldots, x_{l-1}) \tag{5.1}$$

$$E\left(x_t^{i,j}\right) = \left[\boldsymbol{x}_t; \boldsymbol{p}_t^{i,j}\right] \tag{5.2}$$

$$\mathcal{A}_{i,j}^{(n)} = f\left(E\left(x_0^{i,j}\right), E\left(x_1^{i,j}\right), \cdots, E\left(x_{l-1}^{i,j}\right); \theta_e^n\right) \tag{5.3}$$

i,j分别是对应的entity的索引，x_t是一个sequence中t位置的word。p_t是相对位置向量，即表示该单词是否在实体i，实体j中还是都不在。

2. Propagation Module:

$$\mathbf{h}_i^{(n+1)} = \sum_{v_j \in \mathcal{N}(v_i)} \sigma\left(\mathcal{A}_{i,j}^{(n)} \mathbf{h}_j^{(n)}\right) \tag{5.4}$$

3. Classification Module:

$$\mathcal{L} = g\left(\mathbf{h}_{0:|\mathcal{V}|-1}^0, \mathbf{h}_{0:|\mathcal{V}|-1}^1, \ldots, \mathbf{h}_{0:|\mathcal{V}|-1}^K, Y; \theta_c\right) \tag{5.5}$$

$$\boldsymbol{r}_{v_i,v_j} = \left[\left[\boldsymbol{h}_{v_i}^{(1)} \odot \boldsymbol{h}_{v_j}^{(1)}\right]^\top; \left[\boldsymbol{h}_{v_i}^{(2)} \odot \boldsymbol{h}_{v_j}^{(2)}\right]^\top; \ldots; \left[\boldsymbol{h}_{v_i}^{(K)} \odot \boldsymbol{h}_{v_j}^{(K)}\right]^\top\right] \tag{5.6}$$

$$\mathcal{L} = \sum_{s \in S} \sum_{i \neq j} \log \mathbb{P}\left(r_{v_i,v_j} | i, j, s\right) \tag{5.7}$$

6 Multi-hop Reading Comprehension across Multiple Documents by Reasoning over Heterogeneous Graphs

本文所解决的问题是跨文档的多跳阅读理解，提出的模型是Heterogeneous Document-Entity Graph（HDEGraph）。

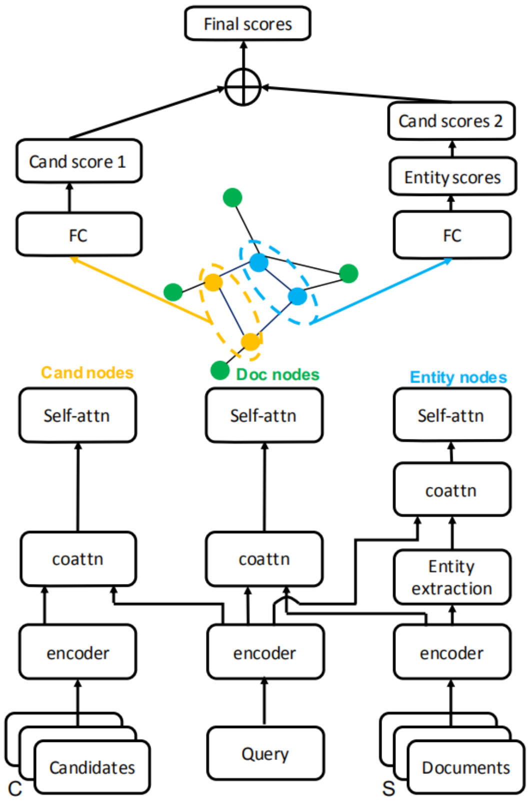


Figure 2: System diagram. S and C are the number of support documents and candidates respectively. We use yellow nodes to represent query-aware candidate representation, blue nodes to represent extracted query-aware entity representation and green nodes to represent query-aware document representation.

1. Context Encoding

   以support document的encoding距离，candidates和query的情况类似。

$$\mathbf{A}_{qs}^{i} = \mathbf{H}_{s}^{i}\left(\mathbf{H}_{q}\right)^{\top} \in \mathbb{R}^{l_{s}^{i} \times l_{q}} \tag{6.1}$$

$$\begin{gathered}\mathbf{C}_{q} = \operatorname{softmax}\left(\mathbf{A}_{qs}^{\top}\right)\mathbf{H}_{s} \in \mathbb{R}^{l_{q} \times h} \\ \mathbf{C}_{s} = \operatorname{softmax}\left(\mathbf{A}_{qs}\right)\mathbf{H}_{q} \in \mathbb{R}^{l_{s} \times h}\end{gathered} \tag{6.2}$$

$$\mathbf{D}_{s} = f\left(\operatorname{softmax}\left(\mathbf{A}_{qs}\right) C_{q}\right) \in \mathbb{R}^{l_{s} \times h} \tag{6.3}$$

$$\mathbf{S}_{ca} = \left[\mathbf{C}_{s}; \mathbf{D}_{s}\right] \in \mathbb{R}^{l_{s} \times 2h} \tag{6.4}$$

$$\begin{gathered}\mathbf{a}_{s} = \operatorname{softmax}\left(MLP\left(\mathbf{S}_{ca}\right)\right) \in \mathbb{R}^{l_{s} \times 1} \\ \mathbf{s}_{sa} = \mathbf{a}_{s}^{\top}\mathbf{S}_{ca} \in \mathbb{R}^{1 \times 2h}\end{gathered} \tag{6.5}$$

2. Reasoning over HDE graph

$$\mathbf{z}_i^k = \sum_{r \in \mathcal{R}} \frac{1}{|\mathcal{N}_i^r|} \sum_{j \in \mathcal{N}_i^r} f_r\left(\mathbf{h}_j^k\right) \tag{6.6}$$

$$\mathbf{u}_i^k = f_s\left(\mathbf{h}_i^k\right) + \mathbf{z}_i^k \tag{6.7}$$

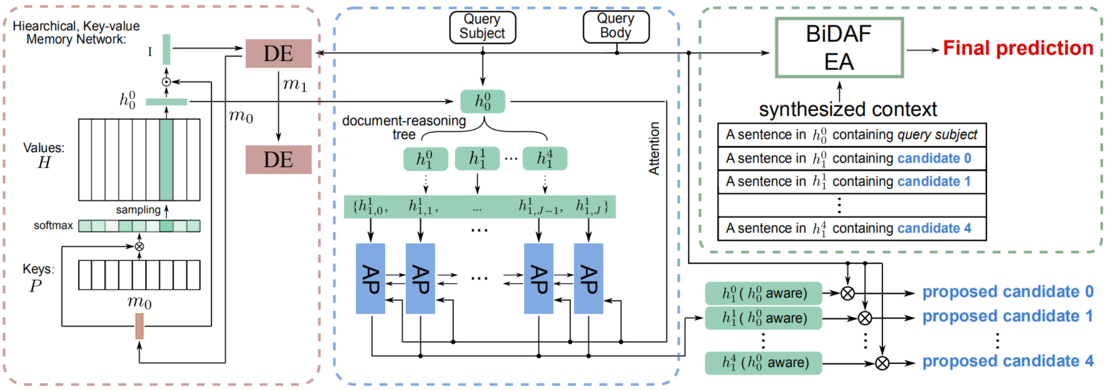
$$\begin{gathered} \mathbf{g}_i^k = \operatorname{sigmoid}\left(f_g\left(\left[\mathbf{u}_i^k; \mathbf{h}_i^k\right]\right)\right) \\ \mathbf{h}_i^{k+1} = \tanh\left(\mathbf{u}_i^k\right) \odot \mathbf{g}_i^k + \mathbf{h}_i^k \odot \left(1 - \mathbf{g}_i^k\right) \end{gathered} \tag{6.8}$$

3. Score accumulation

$$\mathbf{a} = f_C\left(\mathbf{H}^C\right) + ACC_{\max}\left(f_E\left(\mathbf{H}^E\right)\right) \tag{6.9}$$

7 Explore, Propose, and Assemble: An Interpretable Model for Multi-Hop Reading Comprehension

本文也是应用于multi-hop multi-document的一篇文章，提出了Explore-Propose-Assemble reader (EPAr)模型。



1. Retrieval and Encoding:
   使用TF-IDF算法来先筛选出一部分的document，减少计算量。

2. Document Explorer：
   用document-level的representation和word-level的representation分别作为Memory Network的Key和value。

   Read：

   $$x_n = p_n^T \mathbf{W_r} m^t \quad \chi = \operatorname{softmax}(x) \quad P(d_i) = \chi_i \tag{7.1}$$

   Write：

   $$\begin{aligned} w_k &= h_k^T \mathbf{W_w} m & \tilde{h} &= \sum_{k=1}^{K} h_k \omega_k \\ \omega &= \operatorname{softmax}(w) & m^{t+1} &= \mathbf{GRU}\left(\tilde{h}, m^t\right) \end{aligned} \tag{7.2}$$

3. Answer Proposer

   $$\begin{aligned} e_i^k &= \mathbf{v}^T \tanh\left(\mathbf{W_h} \hat{h}_{cct}^i + \mathbf{W_s} s^k + \mathbf{b}\right) \\ a^k &= \operatorname{softmax}\left(e^k\right); \quad c^k = \sum_i a_i^k h_{cct}^i \\ y^k &= \mathbf{LSTM}\left(\hat{h}_T^{k-1}, s^{k-1}, c^{k-1}\right) \\ w^k &= \boldsymbol{\alpha}\left(y^k, u_s\right) + \boldsymbol{\alpha}\left(y^k, u_b\right); \epsilon = \operatorname{softmax}(w) \end{aligned} \tag{7.3}$$

   $$a = \sum_{k=1}^{K} \hat{h}_T^k \epsilon_k; \quad \text{Score }_l = \boldsymbol{\beta}\left(c_l, a\right) \tag{7.4}$$

4. Evidence Assembler：
   将相关的sentence拼接起来，送入传统的单document算法模型。

5. Joint Optimization