一．文献的阅读和总结报告

1. 背景和模型简介

要跨越机器和人类的阅读理解能力之间的鸿沟，有三个主要挑战:
1) 推理能力。单段问答模型倾向于在与问题相匹配的句子中寻找答案，而这一过程不涉及复杂的推理。
2) 可解释性。显式的推理路径能够验证逻辑的严密性，这对 QA 系统的可靠性至关重要。
3) 可伸缩性。人类可以在大容量记忆中轻松地通过知识进行推理。而现有的 OA 系统暂时都还做不到这一点。

人类的认知过程对解决这类问题有值得借鉴的地方。根据双重过程理论的说法，我们的大脑首先通过一个被称为系统 1 的内隐的、无意识的和直觉的过程跟随注意力检索相关信息，然后在此基础上进行另一个明确、有意识和可控的推理过程，即系统 2。系统 1 可以根据请求提供资源，而系统 2 可以通过在工作记忆中执行顺序思维来更深地挖掘关系信息，虽然速度较慢，但具有人类特有的理性。对于复杂的推理，这两个系统是协同的，以迭代的方式执行快速和缓慢的思考。

文章中提出的认知图 QA (CogQA)受到双过程理论的启发，其由功能不同的系统 1 和 2 模块组成。系统 1 从段落中提取与问题相关的实体（entities）和回答候选项（answer candidates），并对其语义信息进行编码。提取出来的实体被组织成一个认知图(图 1)。然后系统 2 对图执行推理过程，并收集线索来指导系统 1 更好地提取下一跳实体。对上述过程进行迭代，直到找到所有可能的答案，然后根据系统 2 的推理结果选择最终的答案。这是一种基于 BERT 和图神经网络(GNN) 的高效实现。

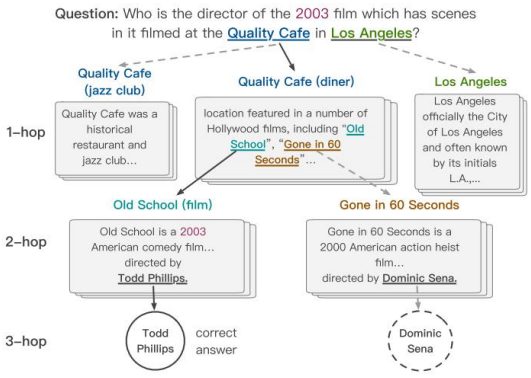

Figure 1: An example of cognitive graph for multi-hop QA. Each *hop node* corresponds to an entity (e.g., "Los Angeles") followed by its introductory paragraph. The circles mean *ans nodes*, answer candidates to the question. Cognitive graph mimics human reasoning process. Edges are built when calling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

该模型的优势如下：
•这一大规模多跳阅读理解 QA 基于人类认知。

•文章提出的框架中的认知图结构提供了有序和实体级的可解释性，并适合关系推理。
•文章基于 BERT 和 GNN 的实现在所有指标上都大大超过了之前的工作和其他竞争对手。

2. 模型细节

模型采用有向图结构构建认知图。认知图 G 的每个节点对应一个实体或可能的答案 x，记作 node x。提取模块系统 1：读取实体 x 的介绍段落 para[x]，并从段落中提取答案的候选和有用的下一跳实体。然后这些新节点扩展了图 G，为推理模块系统 2 提供了显式结构。本文假设系统 2 通过计算节点的隐藏表示 X 来进行基于深度学习的推理，而不是基于规则的推理。因此，在提取跨度时，系统 1 还需要将 para[x]总结进一个语义向量作为初始隐藏表示。然后系统 2 基于图结构更新 X 作为下游预测的推理结果。

可解释性是由于认知图中明确的推理路径而享有的。除了简单的路径外，认知图还可以清楚地显示出联合或循环的推理过程，在这个过程中，新出现的“前辈”节点可能会为答案带来新的线索。文章框架中的线索是一个形式灵活的概念，其会参考前人的信息来指导 System 1 更好地提取跨度。除了新添加的节点，那些有新传入边的节点也需要重新访问，由于新的线索。它们都被称为边界节点（frontier nodes）。

可伸缩性意味着 QA 的时间消耗不会随着段落数量的增加而显著增加。这些段落可以通过在本文的框架中迭代地通过线索扩展来得以发现。

算法 I 描述了 CogQA 的过程。初始化之后，开始了图展开和推理的迭代过程。每一步访问一个边界节点 x，系统 1 在线索和问题 Q 的指导下读取 para[x]，提取 span 并生成语义向量 sem[x,Q,clues]。同时，系统 2 更新隐藏表示 X，并为后续节点 y 准备线索[y, G]，最后根据 X 进行预测。

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 $\arg\max_{\text{answer node } x} \mathcal{F}(\mathbf{X}[x])$

3. 程序实现

CogQA 框架实施的主要部分是确定系统 1 和系统 2 的具体模型，以及线索的形式。
文章的实现使用 BERT 作为系统 I, GNN 作为系统 2。同时，clue[x,G]是 x 的前任节点段落（从这些段落中提取出 x）中的句子。为了方便系统 1 的训练，直接通过原始的句子作为线索，而不是任何形式的计算隐藏状态，因为原始的句子是自成体系的，独立于之前迭代步骤的计算，这样在不同的迭代步骤上的训练就可以解耦，从而在训练过程中提高效率。图节点的隐藏表示 X 每次都由 GNN 的一个传播步骤更新。整体模型如图 2 所示。

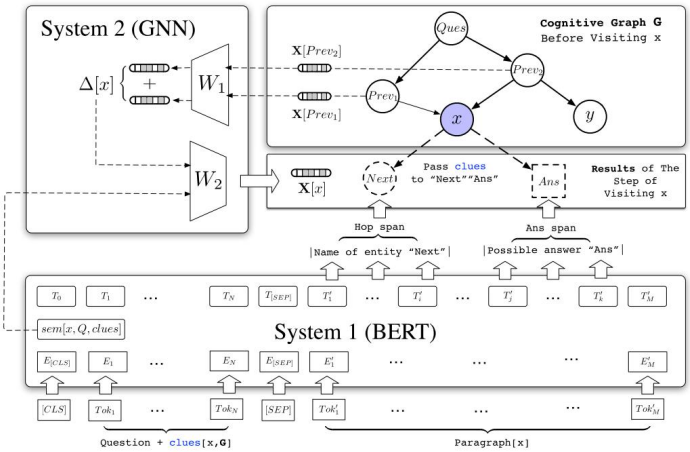

Figure 2: Overview of CogQA implementation. When visiting the node x, System 1 generates new hop and answer nodes based on the $clues[x, \mathcal{G}]$ discovered by System 2. It also creates the inital representation $sem[x, Q, clues]$, based on which the GNN in System 2 updates the hidden representations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

3.1 系统 1

使用 BERT 作为系统 1，其在访问节点 x 时的输入如下:

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

Clue[x, G]是从前一个节点传递的句子。BERT 的输出向量记为$\mathbf{T} \in \mathbb{R}^{L \times H}$, 其中 L 为输入序列的长度，H 为隐藏表示的维数大小。
值得注意的是，对于答案节点 x, Para[x]可能缺失。因此，不提取 span，但仍然可以基于“Sentence A”部分计算 sem[x, Q，clue]。在从问题中提取 1 跳节点来初始化 G 时，不计算语义向量，输入中只存在 Q 部分。
span extraction.答案和下一跳实体具有不同的属性。答案提取在很大程度上依赖于问题的特征。例如，“New York City”比“2019”更可能是 where 问题的答案，而下一跳实体通常是那些实体中的描述与问题中的语句匹配的实体。因此，文章分别预测 answer spans 和 next-hop spans。
引入“指针向量”S_{hop}, E_{hop}, S_{ans}, E_{ans} 作为额外的可学习参数来预测 targeted spans。第 i 个输入令牌为一个 answer span$P_{ans}^{start}[i]$ 的开始概率可以通过如下方式计算:

$$P_{ans}^{start}[i] = \frac{e^{\mathbf{S}_{ans} \cdot \mathbf{T}_i}}{\sum_j e^{\mathbf{S}_{ans} \cdot \mathbf{T}_j}} \qquad (1)$$

设$P_{ans}^{end}[i]$为第 i 个输入令牌为一个 answer span 的结束的概率，计算公式相同。只关注那些 K 个最大的开始概率{$start_k$}对应的的位置。对于每个 k，结束位置 end_k 为:

$$end_k = \underset{start_k \le j \le start_k + maxL}{\arg\max} P_{ans}^{end}[j] \quad (2)$$

其中 maxL 是 span 的最大可能长度。

为了识别不相关的段落，利用负采样技术来训练系统 1 来产生一个负的限值。在顶部 K 个 span 中，那些开始概率小于负阈值的将被丢弃。由于第 0 个令牌[CLS]被预先训练成为下一个句子预测任务合成所有的输入令牌，所以在实现中，$P_{ans}^{start}[0]$ 扮演了阈值的角色。

通过将剩余的预测 answer span 作为新的“答案节点”来扩展认知图。紧接着是同样的过程来扩展“下一跳节点”，即将 S_{ans}、E_{ans} 替换为 S_{hop}、E_{hop}。

Semantics Generation 如前所述，BERT 在位置 0 的输出具有总结序列的能力。因此，最直接的方法是使用 T_0 作为 sem[x, Q, clues]。然而，BERT 中的最后几层主要负责转换 span 预测的隐藏表示。在本文的实验中，他们使用位置为 0 的第三到最后一层的输出作为 sem[x, Q, clues]表现最好。

二．代码阅读和调通

目前仅大致略读了代码，但尚未搭建环境和调通代码。