Document Modeling with GAT for Multi-grained Machine Reading Comprehension

基于GAT的多粒度机器阅读理解的文档建模

ACL 2020

背景

任务：

- Machine reading comprehension (MRC)
- 数据集：Natural Questions
- Wiki页面，文档长度长
- 长答案：候选集里选择
- 短答案：长答案里得到

Example

Question: where is the bowling hall of fame located

Wikipedia page: International Bowling Hall of Fame

Long answer: The World Bowling Writers (WBW) International Bowling Hall of Fame was established in 1993 and is located in the International Bowling Museum and Hall of Fame , on the International Bowling Campus in Arlington , Texas .

Short answer: Arlington , Texas

思想：

- 现有方法会分开做两个子任务，从而忽略了两者之间的依赖性
  - pipeline：先从候选集里选长答案，再从长答案里抽取短答案；
  - 基于BERT：从文档里抽取短答案，再从候选集里选出包含短答案的长答案
- 利用GAT和BERT，提出多粒度的MRC：词，句子，段落，文档
- 使用GAT来获得不同层级的表示，同时可以对不同层次之间的依赖关系进行建模，同时做这两个任务

方法

数据预处理：

- 词切分+文档切片
  - 一个实例：[CLS]问题[SEP]文档片段[SEP]
  - 每个文档片段：7个段落，18个句子

- 每个长答案候选项前打特殊标识：[Paragraph=N]，[Table=N]，[List=N]
  - 文档前几段或者表格里更可能包含答案

- 根据文档片段内是否存在答案打5类标记：short，yes，no，long，no-answer
  - Short：长+短答案为实体list
  - Yes：长+短答案为yes
  - No：长+短答案为no
  - Long：长+不包括短答案
  - No-answer：不包含答案

- 每个NQ平均生成30个样例，删去97%不包含答案的样例，总共660,000训练样例，350,000条训练样例包含长答案，270,000条训练样例包含短答案

方法

输入：

Formally, we define an instance in the training set as a six-tuple

$$(\boldsymbol{c}, S, l, s, e, t).$$

$\boldsymbol{c} = ([\mathrm{CLS}], Q_1, ..., Q_{|Q|}, [\mathrm{SEP}], D_{i,1}, ..., D_{i,|D_i|}, [\mathrm{SEP}])$ defines the document fragment D_i along with a question Q of the instance, $|Q| + |D_i| + 3 = 512$ corresponding to the data preprocessing method.

- c：文档片段
- S：长答案候选集
- l：候选集S中的目标长答案候选
- s和e：指向短答案位置的start和end，∈（0,511）
- t：带注释的答案类型，t=0.1.2.3.4

目标：

Our goal is to learn a model that identifies a long answer candidate l and a short answer span (s, e) in l and predicting their scores for evaluation.

- 学出长答案候选l和l里面的短答案跨度s和e

方法

Graph建模：

- 词，句子，段落，文档片段

- 段落：长答案
- 词：短答案

- 增加词-段落、词-文档、句子-文档层次之间的边

- 都是双向边

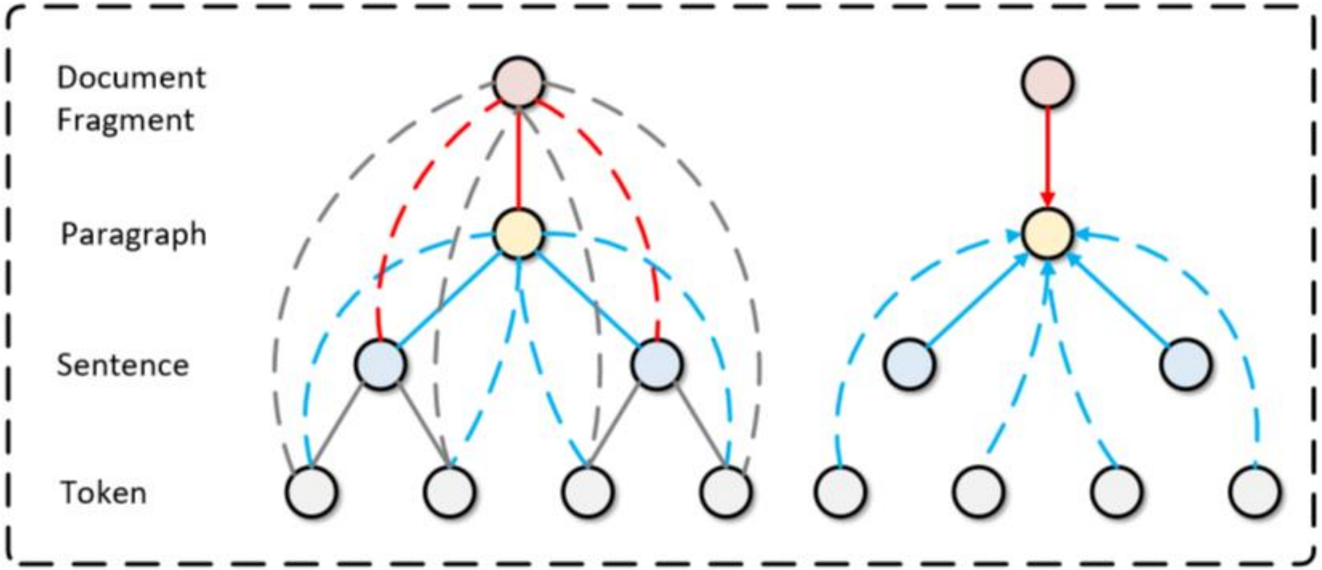


Figure 4: The graph on the left is an illustration of the graph integration layer. The graph on the right shows the incoming information when updating a paragraph node. The solid lines represent the edges in the hierarchical tree structure of a document while the dash lines stand for the edges we additionally add.

方法：Graph Initialization

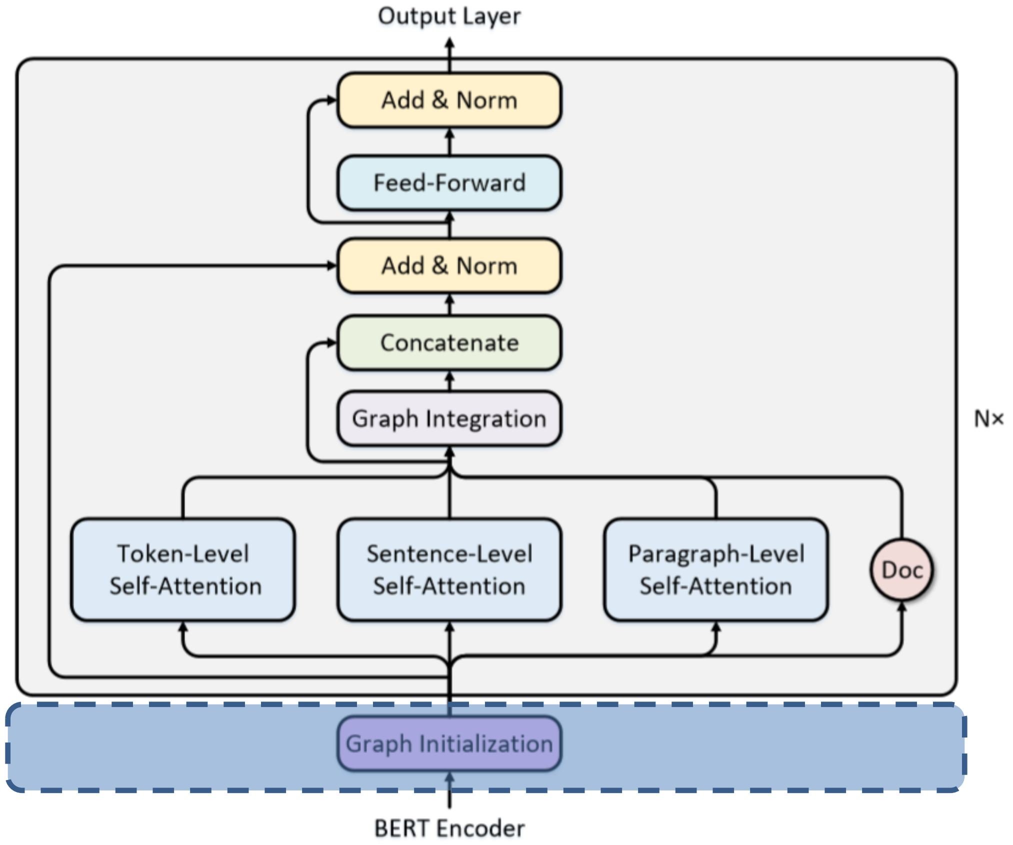
Graph初始化：

- Token：BERT
- 除token外，其他节点初始化：

$$\boldsymbol{h}_i^0 = \operatorname*{average}_{j \in \mathcal{N}_i, o_j+1=o_i} \{\boldsymbol{h}_j^0 + \boldsymbol{a}_{ij}\} + \boldsymbol{b}_{o_i}$$

$$o_i \in \{0, 1, 2, 3\}$$

$$\boldsymbol{a}_{ij}, \boldsymbol{b}_{o_i} \in \mathbb{R}^{d_h}$$



方法：GAT

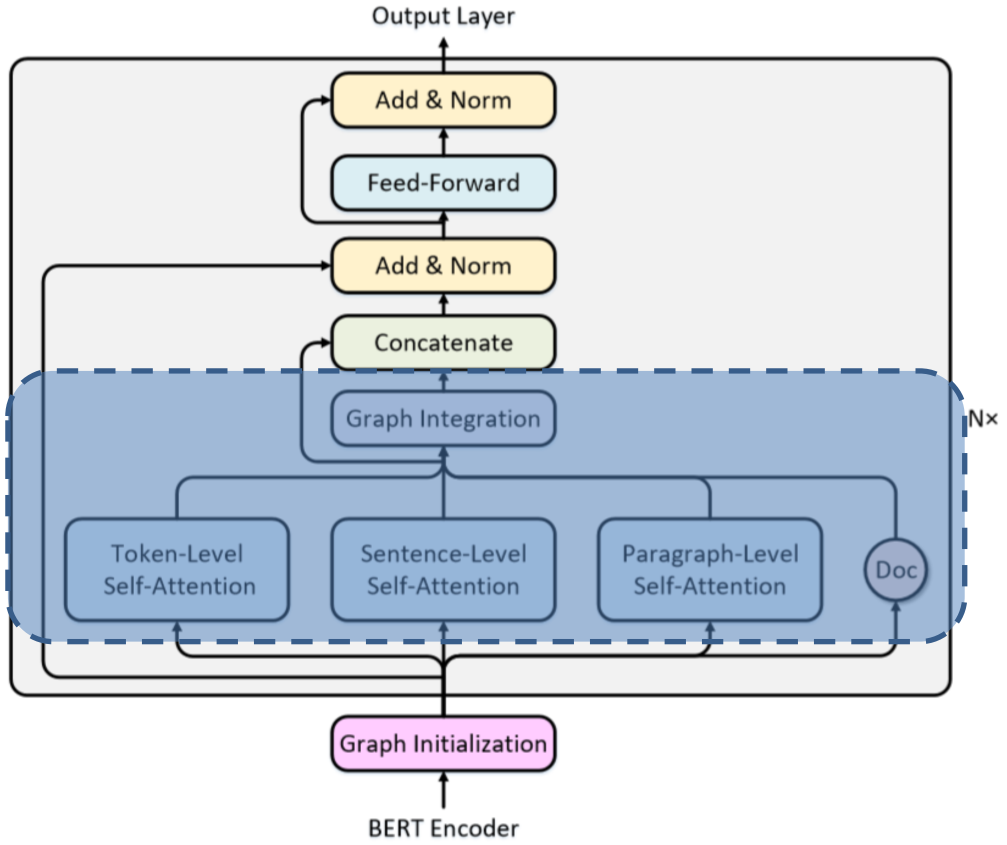
Graph：$\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$

- V：节点
- X：节点特征 $(\boldsymbol{h}_1, ..., \boldsymbol{h}_{|\mathcal{V}|})$

$i \in \mathcal{V}$ has its own representation $\boldsymbol{h}_i \in \mathbb{R}^{d_h}$ where d_h is the hidden size of our model.

- E：边

$(\mathcal{E}_1, ..., \mathcal{E}_K)$ where K is the number of edges



多头注意力机制

- m个head
- j→i：

$$e_{ij} = \frac{(\boldsymbol{h}_i \mathbf{W}^{\mathrm{Q}}) (\boldsymbol{h}_j \mathbf{W}^{\mathrm{K}})^{\mathrm{T}}}{\sqrt{d_z}}$$

$$\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$

$$\boldsymbol{z}_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij} \boldsymbol{h}_j \mathbf{W}^{\mathrm{V}}$$

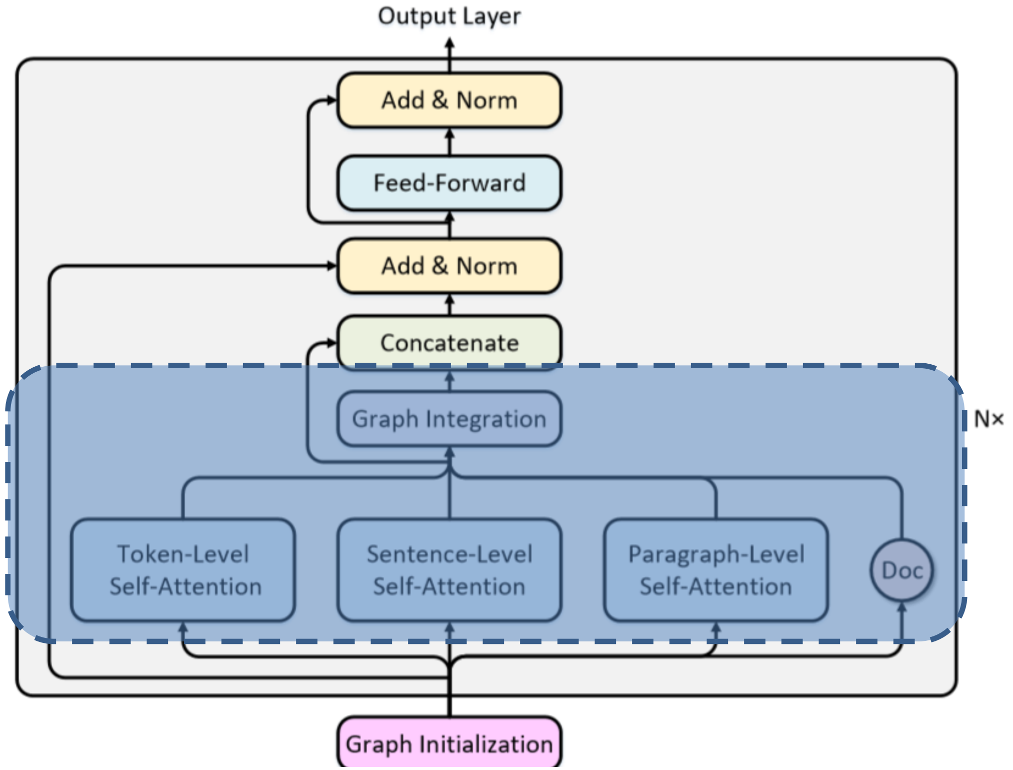
$$\boldsymbol{z}_i' = \Big\Vert_{k=1}^{m} \boldsymbol{z}_i^k$$

方法：GAT

关系嵌入：$\boldsymbol{a}_{ij}^{K}, \boldsymbol{a}_{ij}^{V} \in \mathbb{R}^{d_z}$

- 多粒度节点之间建立相对位置信息的embedding

- Self-attention：两个节点的相对距离
- 集成层：句子在段落中的相对位置



$$e_{ij} = \frac{\left(\boldsymbol{h}_i \mathbf{W}^{\mathrm{Q}}\right)\left(\boldsymbol{h}_j \mathbf{W}^{\mathrm{K}}\right)^{\mathrm{T}}}{\sqrt{d_z}}$$

$$\boldsymbol{z}_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij} \boldsymbol{h}_j \mathbf{W}^{\mathrm{V}}$$

$$\begin{aligned} e_{ij} &= \frac{\left(\boldsymbol{h}_i \mathbf{W}^{\mathrm{Q}}\right)\left(\boldsymbol{h}_j \mathbf{W}^{\mathrm{K}}\right)^{\mathrm{T}} + \boldsymbol{h}_i \mathbf{W}^{\mathrm{Q}} \left(\boldsymbol{a}_{ij}^{\mathrm{K}}\right)^{\mathrm{T}}}{\sqrt{d_z}} \\ \boldsymbol{z}_i &= \sum_{j \in \mathcal{N}_i} \alpha_{ij} \left(\boldsymbol{h}_j \mathbf{W}^{\mathrm{V}} + \boldsymbol{a}_{ij}^{\mathrm{V}}\right). \end{aligned}$$

方法:

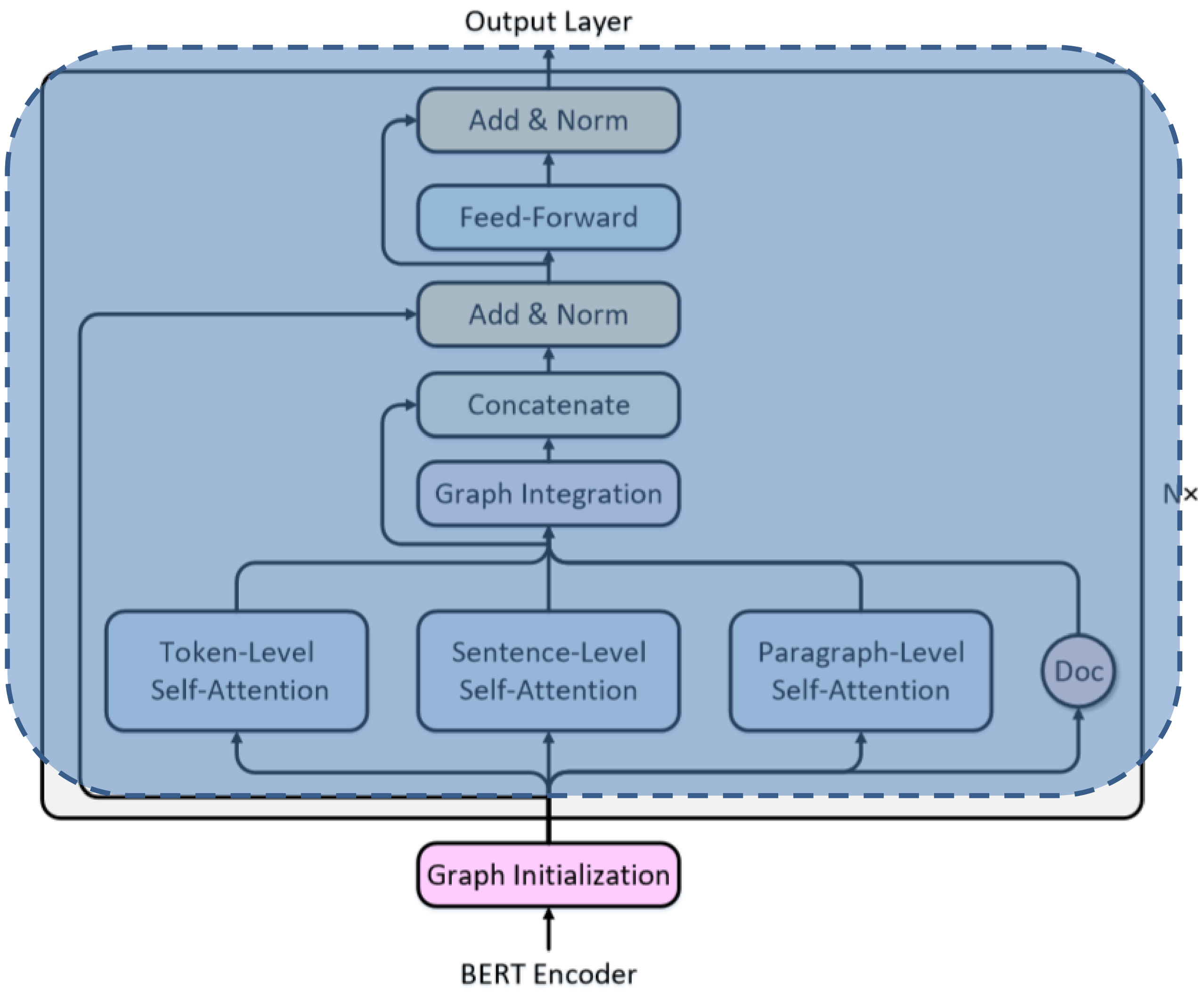
自注意力机制:

- 全连通GAT
- 相同粒度：token，句子，段落
- 每个层内也有关系嵌入：节点的相对距离

Graph Integration:

- 进行多粒度级别的信息传递

Feed-Forward:
- 全连通
- 用 GLEU 做非线性激活单元



方法:

输出：

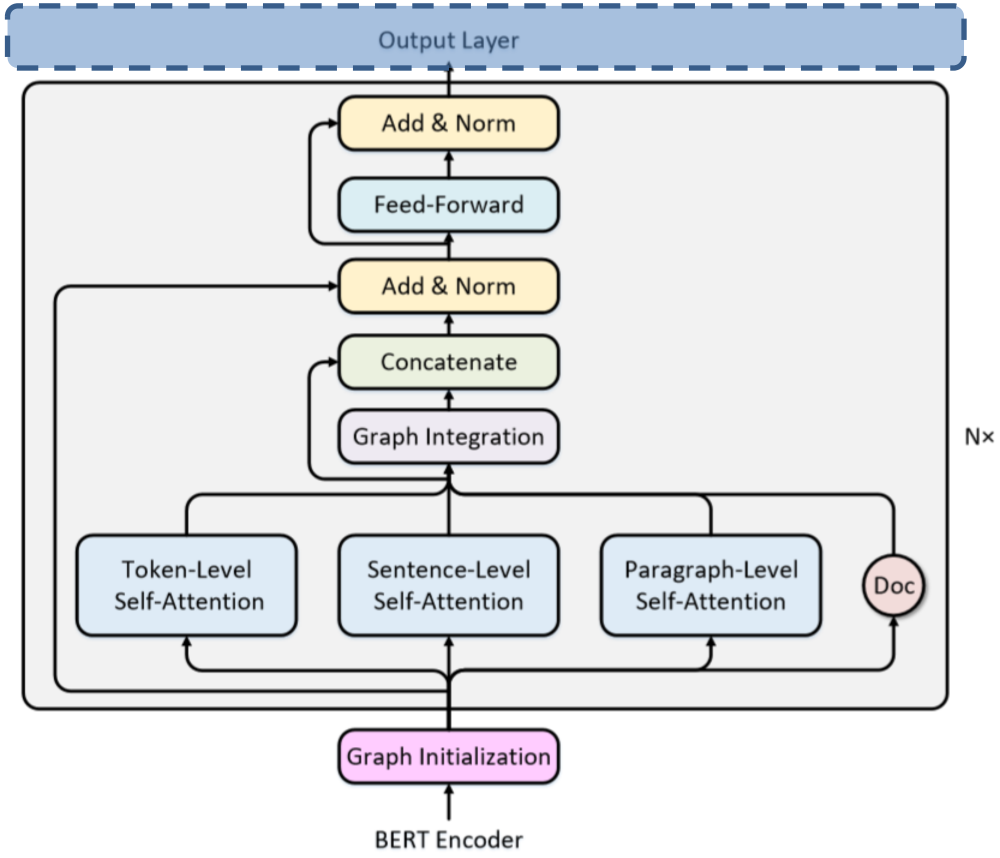
$$L(\theta) = -\frac{1}{N}\sum_{i}^{N}[\log p(s,e,t,l \mid \mathbf{c}, S)]$$
$$= -\frac{1}{N}\sum_{i}^{N}[\log p_s(s \mid \mathbf{c}, S) + \log p_e(e \mid \mathbf{c}, S)$$
$$+ \log p_t(t \mid \mathbf{c}, S) + \log p_l(l \mid \mathbf{c}, S)],$$

- 目标函数定义为所有训练实例的预测偏差的负对数似然
- s和e是短答案预测，l是长答案预测，t是答案类型预测

$$p_s(s \mid \mathbf{c}, S) = \mathrm{softmax}(f_s(s, \mathbf{c}, S; \theta))$$

- f是评分函数
- 如果没有短答案

$$g(\mathbf{c}, S) = f_t(t > 0, \mathbf{c}, S; \theta) - f_t(t = 0, \mathbf{c}, S; \theta);$$
$$g(\mathbf{c}, S, l) = f_l(l, \mathbf{c}, S; \theta) - f_l(l = [\mathrm{CLS}], \mathbf{c}, S; \theta);$$
$$g(\mathbf{c}, S, s, e) = f_s(s, \mathbf{c}, S; \theta) + f_e(e, \mathbf{c}, s; \theta) - f_s(s = [\mathrm{CLS}], \mathbf{c}, S; \theta) - f_e(e = [\mathrm{CLS}], \mathbf{c}, S; \theta).$$



实验

整体架构：

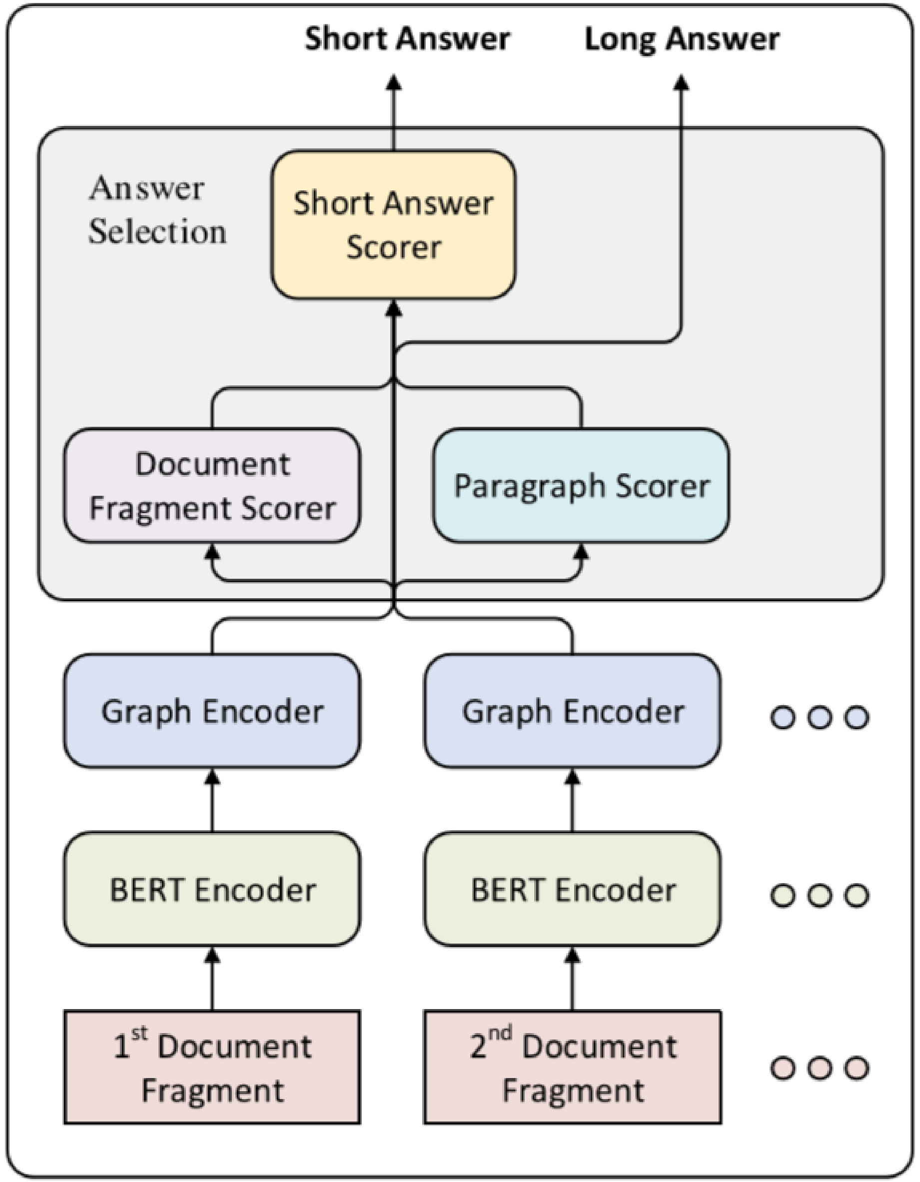
- 数据预处理，问题和文档片段

  ↓
- BERT编码器和图初始化，得到问题和文档片段的初步表示

  ↓
- 图神经网络（自注意力+集成层）

  ↓
- 答案选择模块，对这些表示进行打分

  ↓
- 模型预测，使用Pipeline策略，先预测长答案，再预测短答案，将基于BERT的方法视为基线模型



实验

模型设置：

- Model-I：基线模型，首先预测短答案，根据短答案的位置在长答案候选项中进行选择
- Model-II：只使用average-pooling初始化图，并进行Pipeline预测
- Model-III：在Model-II的基础上加入了两层图神经网络编码器

BERT：

- BERT-base：在SQuAD2.0数据集上微调过的BERT-base-uncased模型
- BERT-large：在SQuAD2.0数据集上微调过的BERT-large-uncased模型
- BERT-syn：Google在SQuAD2.0上提交的模型，经过了N-gram masking以及synthetic self-training的预训练

实验 https://github.com/DancingSoul/NQ_BERT-DM

	Long Answer Dev			Long Answer Test			Short Answer Dev			Short Answer Test		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
DocumentQA	47.5	44.7	46.1	48.9	43.3	45.7	38.6	33.2	35.7	40.6	31.0	35.1
DecAtt + DocReader	52.7	57.0	54.8	54.3	55.7	55.0	34.3	28.9	31.4	31.9	31.1	31.5
$\text{BERT}_{\text{joint}}$	61.3	68.4	64.7	64.1	68.3	66.2	59.5	47.3	52.7	**63.8**	44.0	52.1
+ 4M synthetic data	62.3	70.0	65.9	65.2	68.4	66.8	60.7	50.4	55.1	62.1	47.7	53.9
BERT-syn+Model-III	72.4	73.0	72.7	-	-	-	60.1	54.1	56.9	-	-	-
+ ensemble 3 models	**74.2**	**73.6**	**73.9**	**73.7**	**75.3**	**74.5**	**64.0**	**54.9**	**59.1**	62.6	**55.3**	**58.7**
Single Human	80.4	67.6	73.4	-	-	-	63.4	52.6	57.5	-	-	-
Super-annotator	90.0	84.6	87.2	-	-	-	79.1	72.6	75.7	-	-	-

Model	LA. F1	SA. F1
BERT-base+Model-I	63.9	51.0
BERT-base+Model-II	67.7	50.9
BERT-base+Model-III	**68.9**	**51.9**
$\text{BERT}_{\text{joint}}$	64.7	52.7
BERT-large+Model-I	66.0	52.9
BERT-large+Model-II	70.3	53.2
BERT-large+Model-III	**70.7**	**53.8**
BERT-syn+Model-I	67.8	56.1
BERT-syn+Model-II	72.2	56.7
BERT-syn+Model-III	**72.7**	**56.9**

Model	LA.F1	SA.F1
0-layer	67.7	50.9
1-layer	68.8	51.2
2-layer	**68.9**	**51.9**
3-layer	**68.9**	**51.9**
4-layer	**68.9**	51.7

Model	LA. F1	SA. F1
BERT-base+Model-III	**68.9**	**51.9**
-Graph module	63.9	51.0
-Long answer prediction	65.1	51.4
-Short answer prediction	68.2	-
-Relational embedding	68.8	51.7
-Graph integration layer	68.3	51.1
-Self-attention layer	68.4	51.2

THANK YOU FOR WATCHING