ACL2019

Dynamically Fused Graph Network for Multi-hop Reasoning

Lin Qiu[1†] **Yunxuan Xiao**[1†] **Yanru Qu**[1†]
Hao Zhou[2] **Lei Li**[2] **Weinan Zhang**[1] **Yong Yu**[1]
[1] Shanghai Jiao Tong University [2] ByteDance AI Lab, China
{lqiu, kevinqu, yyu}@apex.sjtu.edu.cn
{xiaoyunxuan, wnzhang}@sjtu.edu.cn
{zhouhao.nlp, lilei.02}@bytedance.com

Task and Motivation

- Multi-hop text-based QA on open domain HotpotQA
- Multi-hop: many difficult questions require multiple supporting evidence from scattered text across two or more documents.
- Most existing approaches focus on finding the answer to a question within a single paragraph.

Question Answering

根据supporting information的形式，可以分为KBQA，TBQA， KB和TB混合的QA，以及其他类型的QA比如VQA
根据推理的复杂性，可以分为single-hop和multi-hop

KBQA： SimpleQuestions
TBQA: SQuAD（Single Hop）, HotpotQA (Multi Hop)
Mixed QA： WikiHop (Multi Hop), ComplexWebQuestions (Multi Hop)

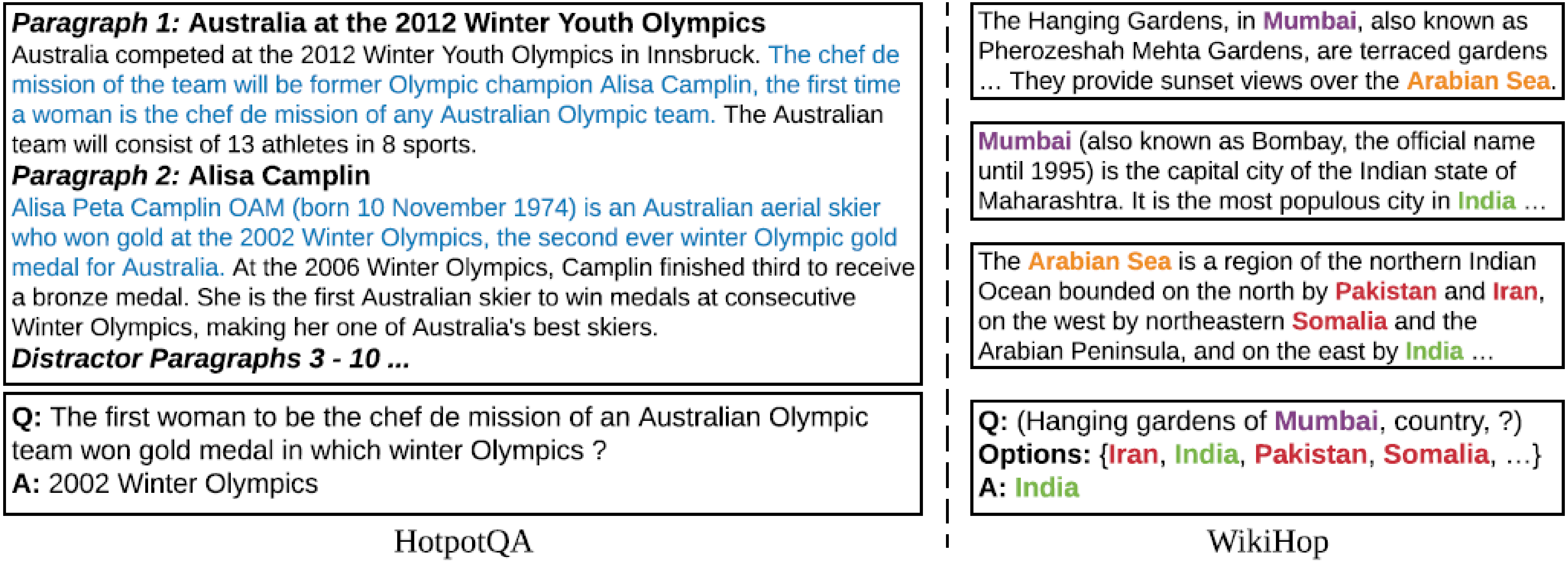


Figure 2: Comparison between HotpotQA (left) and WikiHop (right). In HotpotQA, the questions are proposed by crowd workers and the blue words in paragraphs are labeled supporting facts corresponding to the question. In WikiHop, the questions and answers are formed with relations and entities in the underlying KB respectively, thus the questions are inherently restricted by the KB schema. The colored words and phrases are entities in the KB.

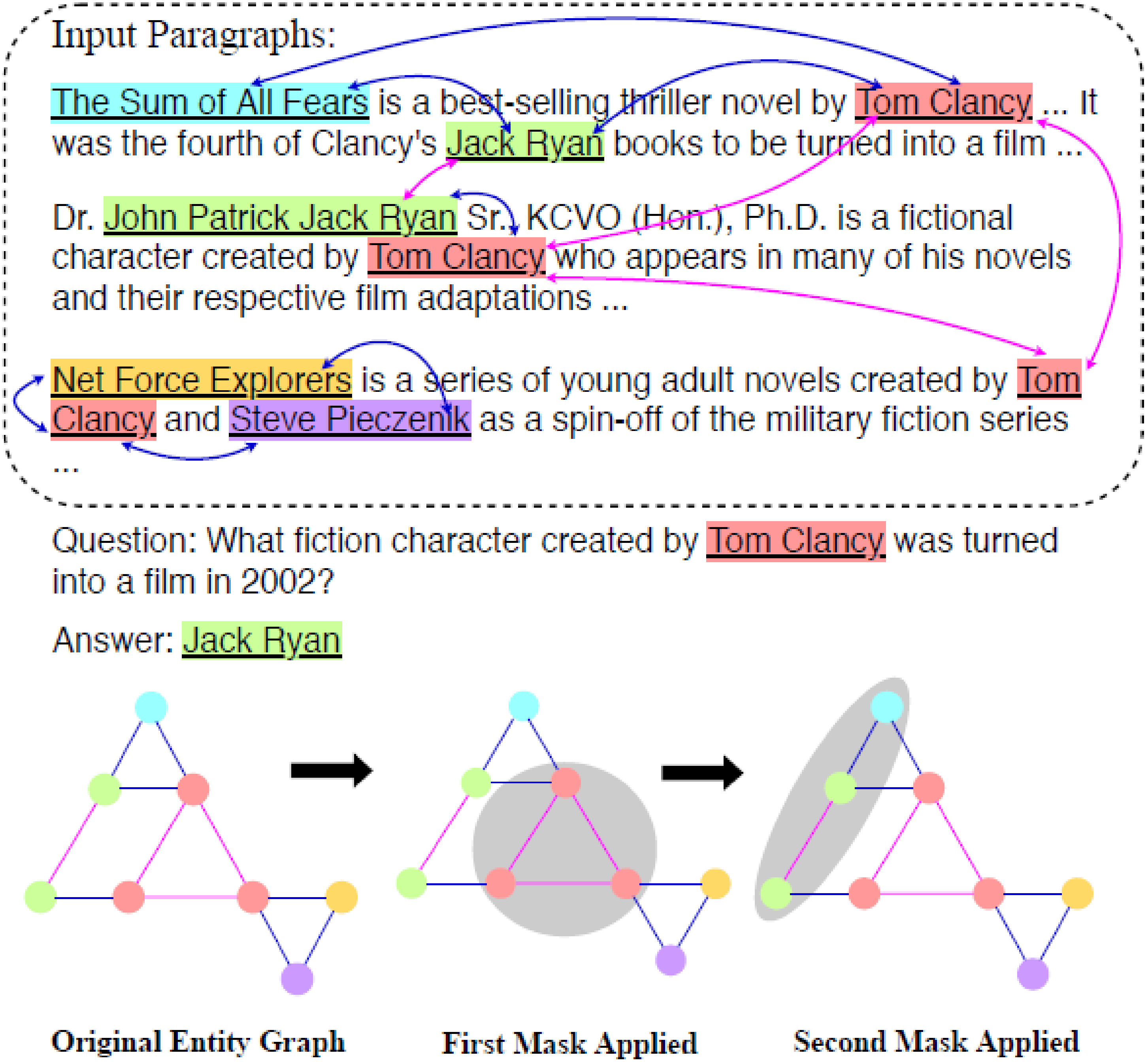


Figure 1: Example of multi-hop text-based QA. One question and three document paragraphs are given. Our proposed DFGN conducts multi-step reasoning over the facts by constructing an entity graph from multiple paragraphs, predicting a dynamic mask to select a sub-graph, propagating information along the graph, and finally transfer the information from the graph back to the text in order to localize the answer. Nodes are entity occurrences, with the color denoting the underlying entity. Edges are constructed from co-occurrences. The gray circles are selected by DFGN in each step.

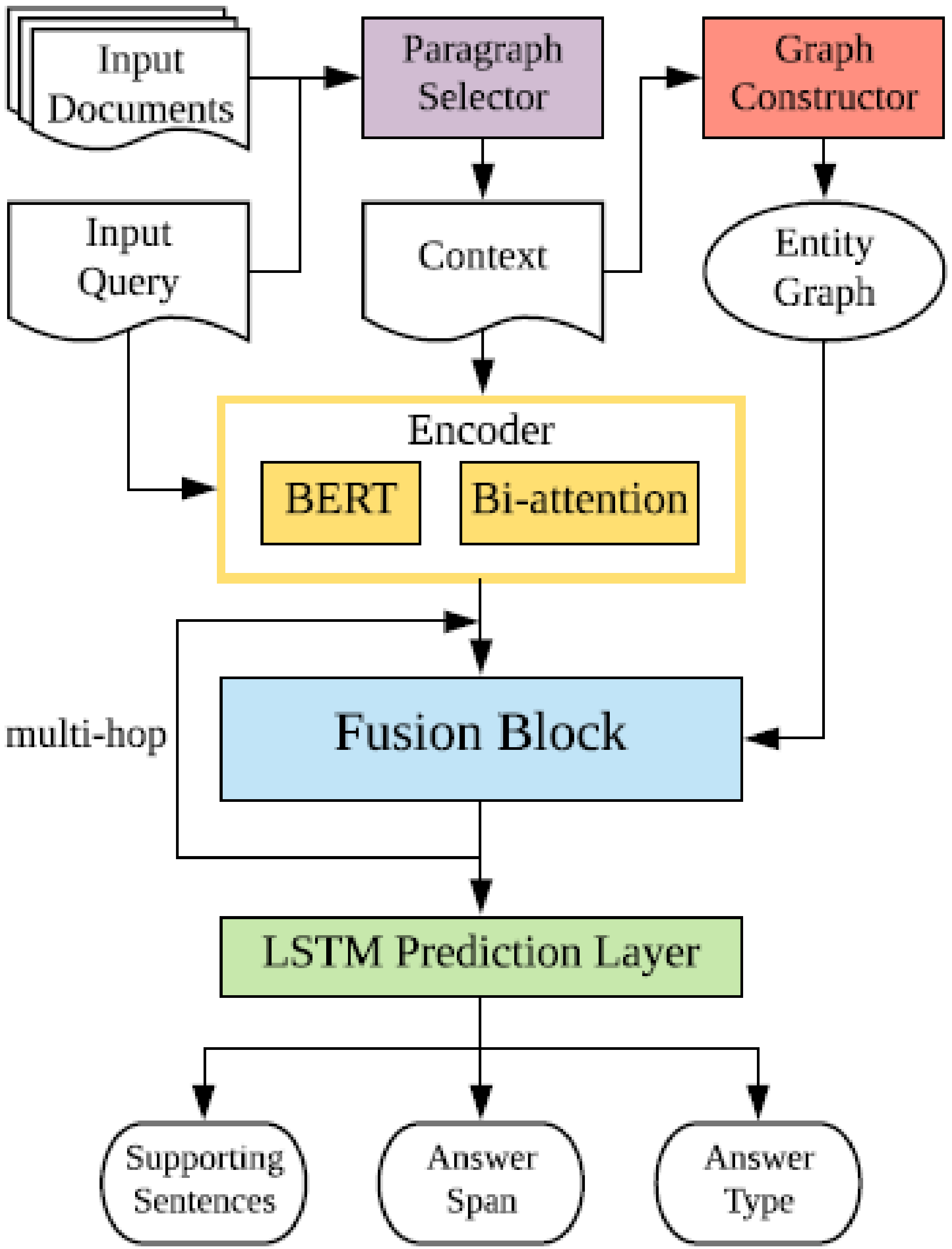
Challenges for Multi-hop QA

- Not every document contain relevant information, requires filtering out noises from multiple paragraphs and extracting useful information.
- previous work on multi-hop QA (e.g. WikiHop) usually aggregates document information to an entity graph, and answers are then directly selected on entities of the entity graph.
- However, in a more realistic setting, the answers may even not reside in entities of the extracted entity graph. Thus, existing approaches can hardly be directly applied to open-domain multi-hop QA tasks like HotpotQA.

Method

- Paragraph Selection
- Constructing Entity Graph
- Encoding Query and Context
- Reasoning with the Fusion Block
- Prediction

Method



Input Documents
Paragraph Selector
Graph Constructor
Input Query
Context
Entity Graph
Encoder
BERT
Bi-attention
multi-hop
Fusion Block
LSTM Prediction Layer
Supporting Sentences
Answer Span
Answer Type

Paragraph Selection

- Since not every piece of text is relevant to the question, we train a sub-network to select relevant paragraphs.
- The sub-network is based on a pre-trained BERT model followed by a sentence classification layer with sigmoid prediction.
- Training labels are constructed by assigning 1's to the paragraphs with at least one supporting sentence for each Q&A pair.

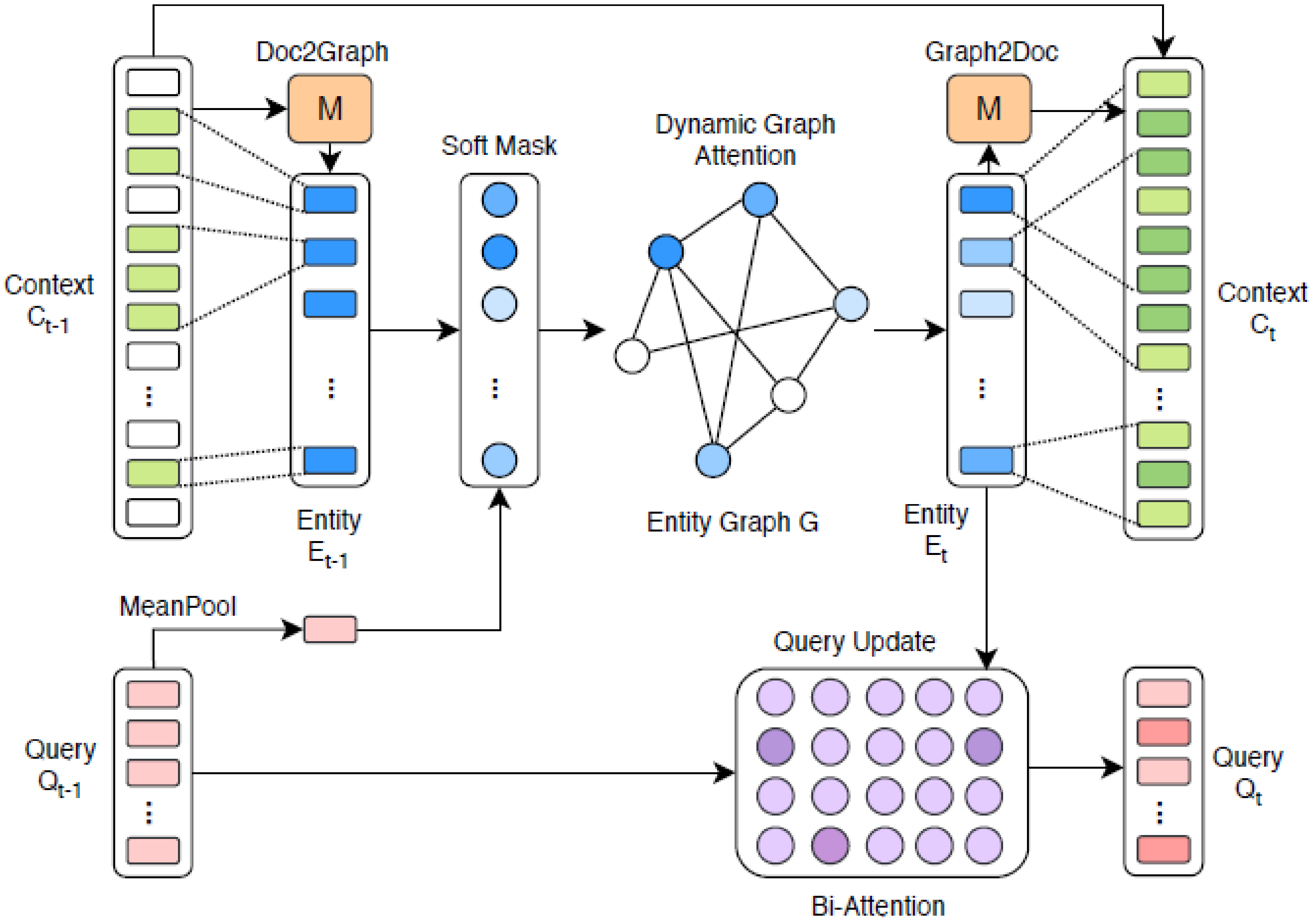
Constructing Entity Graph

- Stanford corenlp toolkit to recognize named entities from the context C
- The entity graph is constructed with the entities as nodes and edges :
- 1. for every pair of entities appear in the same sentence in C (sentence-level links)
- 2. for every pair of entities with the same mention text in C (context-level links)
- 3. between a central entity node and other entities within the same paragraph (paragraph-level links).

Encoding Query and Context

- BERT + Bi-Attetion

Reasoning with the Fusion Block



$$\tilde{\mathbf{q}}^{(t-1)} = \mathrm{MeanPooling}(\mathbf{Q}^{(t-1)}) \tag{1}$$

$$\gamma_i^{(t)} = \tilde{\mathbf{q}}^{(t-1)}\mathbf{V}^{(t)}\mathbf{e}_i^{(t-1)}/\sqrt{d_2} \tag{2}$$

$$\mathbf{m}^{(t)} = \sigma([\gamma_1^{(t)}, \cdots, \gamma_N^{(t)}]) \tag{3}$$

$$\tilde{\mathbf{E}}^{(t-1)} = [m_1^{(t)}\mathbf{e}_1^{(t-1)}, \ldots, m_N^{(t)}\mathbf{e}_N^{(t-1)}] \tag{4}$$

Prediction

$$\mathbf{O}_{sup} = \mathcal{F}_0(\mathbf{C}^{(t)}) \tag{11}$$

$$\mathbf{O}_{start} = \mathcal{F}_1([\mathbf{C}^{(t)}, \mathbf{O}_{sup}]) \tag{12}$$

$$\mathbf{O}_{end} = \mathcal{F}_2([\mathbf{C}^{(t)}, \mathbf{O}_{sup}, \mathbf{O}_{start}]) \tag{13}$$

$$\mathbf{O}_{type} = \mathcal{F}_3([\mathbf{C}^{(t)}, \mathbf{O}_{sup}, \mathbf{O}_{end}]) \tag{14}$$

$$\mathcal{L} = \mathcal{L}_{start} + \mathcal{L}_{end} + \lambda_s \mathcal{L}_{sup} + \lambda_t \mathcal{L}_{type} \tag{15}$$

Results

Model	Answer		Sup Fact		Joint	
	EM	F1	EM	F1	EM	F1
Baseline Model	45.60	59.02	20.32	64.49	10.83	40.16
GRN*	52.92	66.71	52.37	84.11	31.77	58.47
DFGN(Ours)	55.17	68.49	49.85	81.06	31.87	58.23
QFE*	53.86	68.06	**57.75**	**84.49**	**34.63**	59.61
DFGN(Ours)†	**56.31**	**69.69**	51.50	81.62	33.62	**59.82**

DFGN模型的贡献

- 前人重点在如何建图，本文重点关注如果在已经建好的图上做多跳的推理
- 使用Doc2Graph和Graph2Doc，使得从图上得到的推理信息回归到文本中来获取answer和supporting fact

Arxiv 2020.4

Is Graph Structure Necessary for Multi-hop Reasoning?

Nan Shao[†], **Yiming Cui**[‡†], **Ting Liu**[‡], **Shijin Wang**[†§], **Guoping Hu**[†]
[†]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, China
[‡]Research Center for Social Computing and Information Retrieval (SCIR), Harbin Institute of Technology, Harbin, China
[§]iFLYTEK AI Research (Hebei), Langfang, China
[†§]{nanshao,ymcui,sjwang3,gphu}@iflytek.com
[‡]{ymcui,tliu}@ir.hit.edu.cn

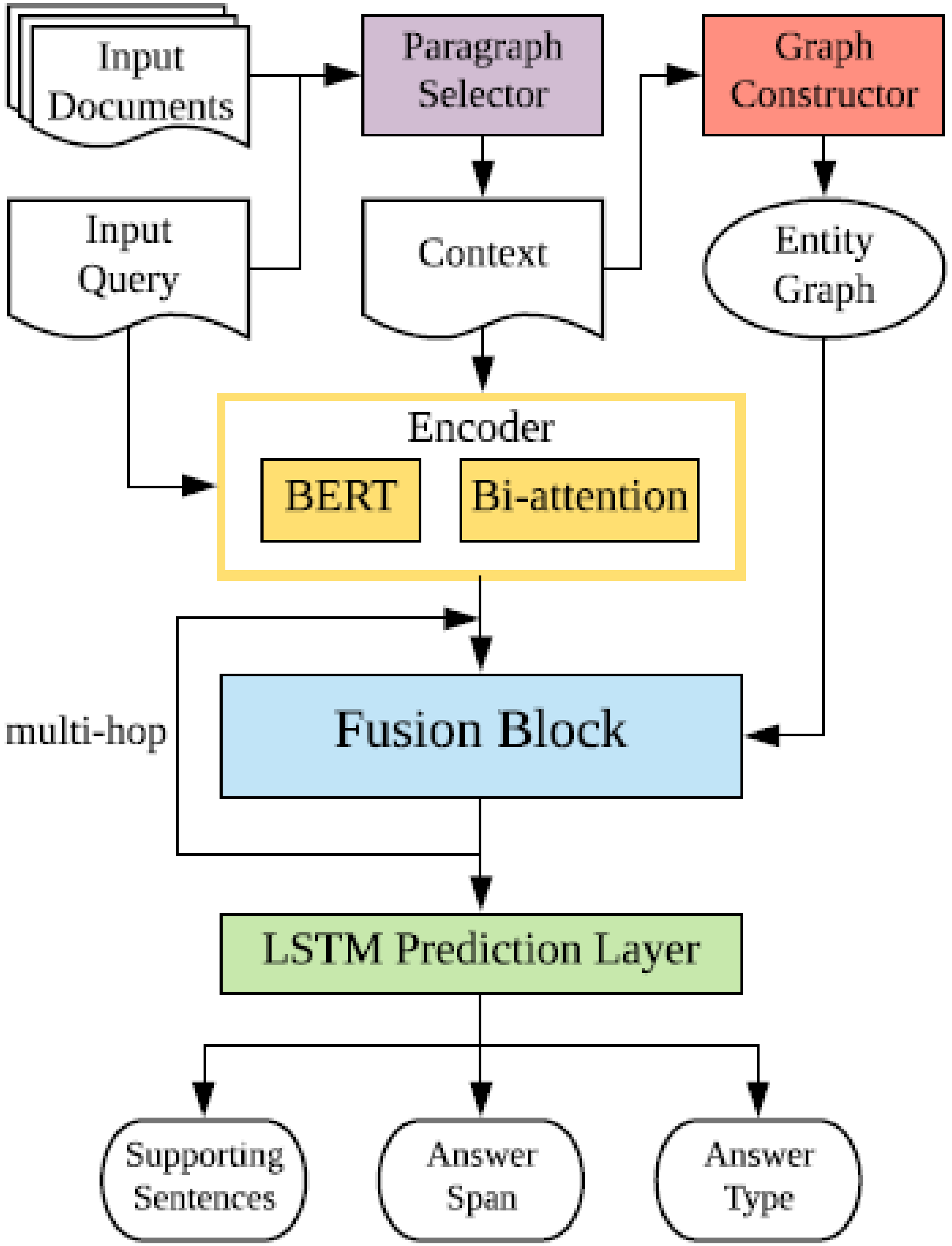
动机和结论

- 探究图结构对于多跳推理是否是有必要的（DFGN on HotportQA）

- 结论1：合理使用预训练语言模型也能做到多跳推理
- 结论2：GAT是Transformer的特例，self-attention也能学会graph attention所需要的邻接矩阵

Previous works argue that a fancy graph structure is a vital part of their models and demonstrate that by ablation experiments. However, in experiments, we find when we use the pre-trained models in the fine-tuning approach, removing entire graph structure may not hurt the final results. Therefore, in this paper, we aimed to answer the following question: **How much does graph structure contribute to multi-hop reasoning?**

To explain the results of experiments, we point out that graph-attention (Veličković et al., 2018) is a special case of self-attention. The adjacency matrix based on manually defined rules and the graph structure can be regarded as prior knowledge, which could be learned by self-attention or Transformer (Vaswani et al., 2017). We design the experiments to show when we model text as an entity graph, both graph-attention and self-attention can achieve comparable results. When we treat texts as a sequence structure, only a 2-layer Transformer could achieve similar results as DFGN.


Input Documents
Paragraph Selector
Graph Constructor
Input Query
Context
Entity Graph
Encoder
BERT
Bi-attention
multi-hop
Fusion Block
LSTM Prediction Layer
Supporting Sentences
Answer Span
Answer Type

Model	Joint	
	EM	F1
Baseline (Yang et al., 2018)	10.83	40.16
QFE (Nishida et al., 2019)	34.63	59.61
DFGN (Qiu et al., 2019)	33.62	59.82
SAE (Tu et al., 2019a)	38.81	64.96
TAP2 (Glass et al., 2019)	39.77	69.12
EPS+BERT	42.47	70.48
HGN (Fang et al., 2019)	43.57	71.03
Our Model	**44.67**	**72.73**

Table 1: Results on the test set of HotpotQA in the Distractor setting.

Graph Structure May Not Be Necessary

Setting	Joint EM	Joint F1
Baseline (Fine-tuning)	45.91	**73.93**
w/o Graph	**45.98**	73.78
Baseline (Feature-based)	36.45	63.75
w/o Graph	32.26	59.76

Table 2: Ablation of graph structure under different settings.

Graph Attention Versus Self Attention

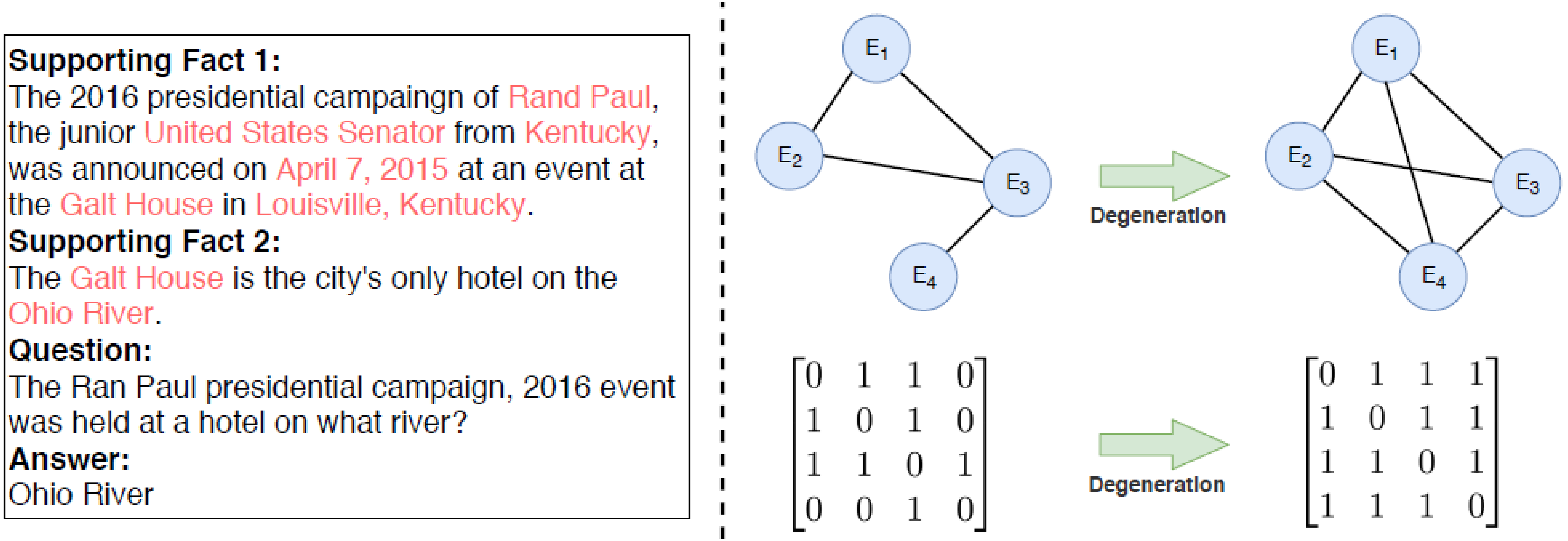
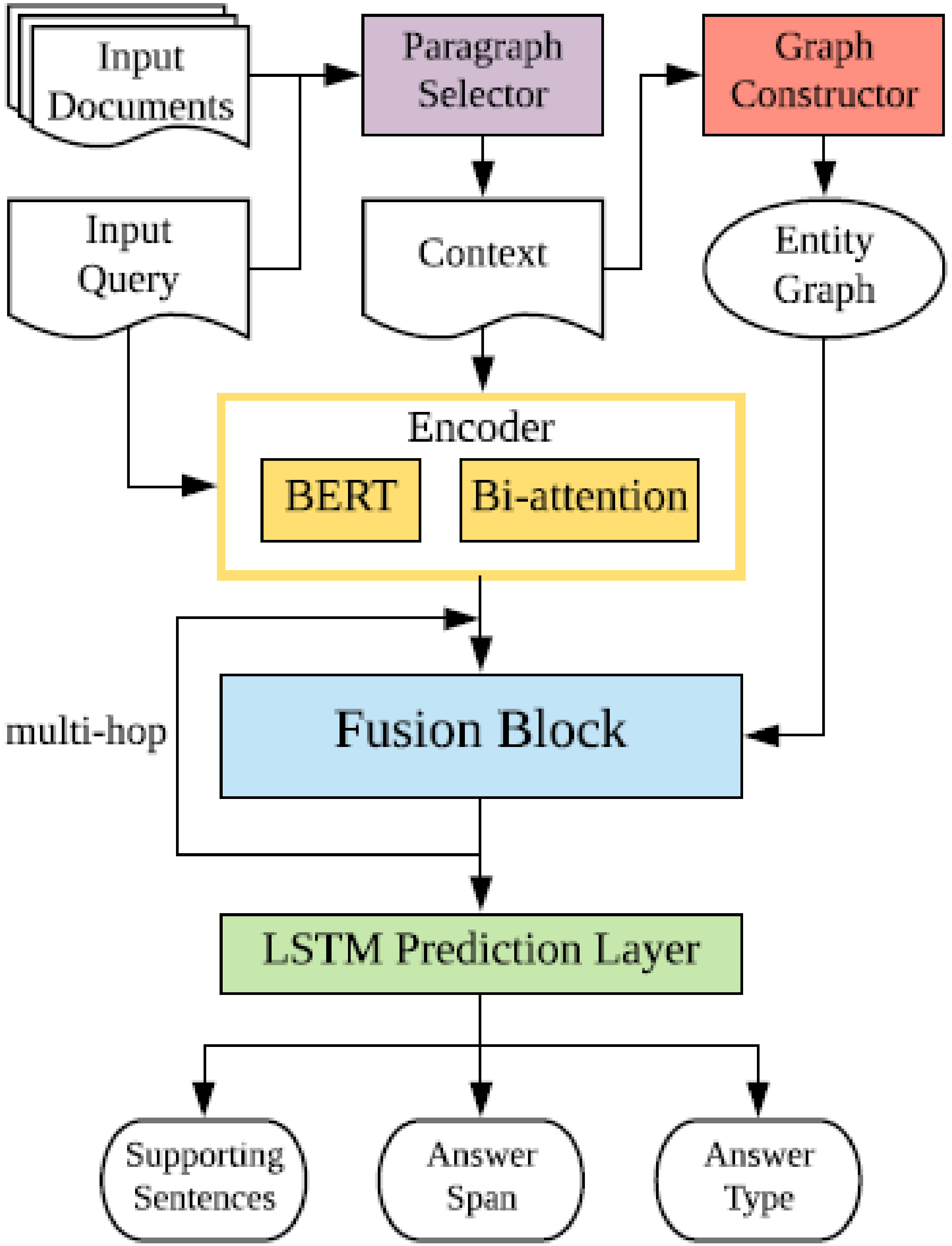


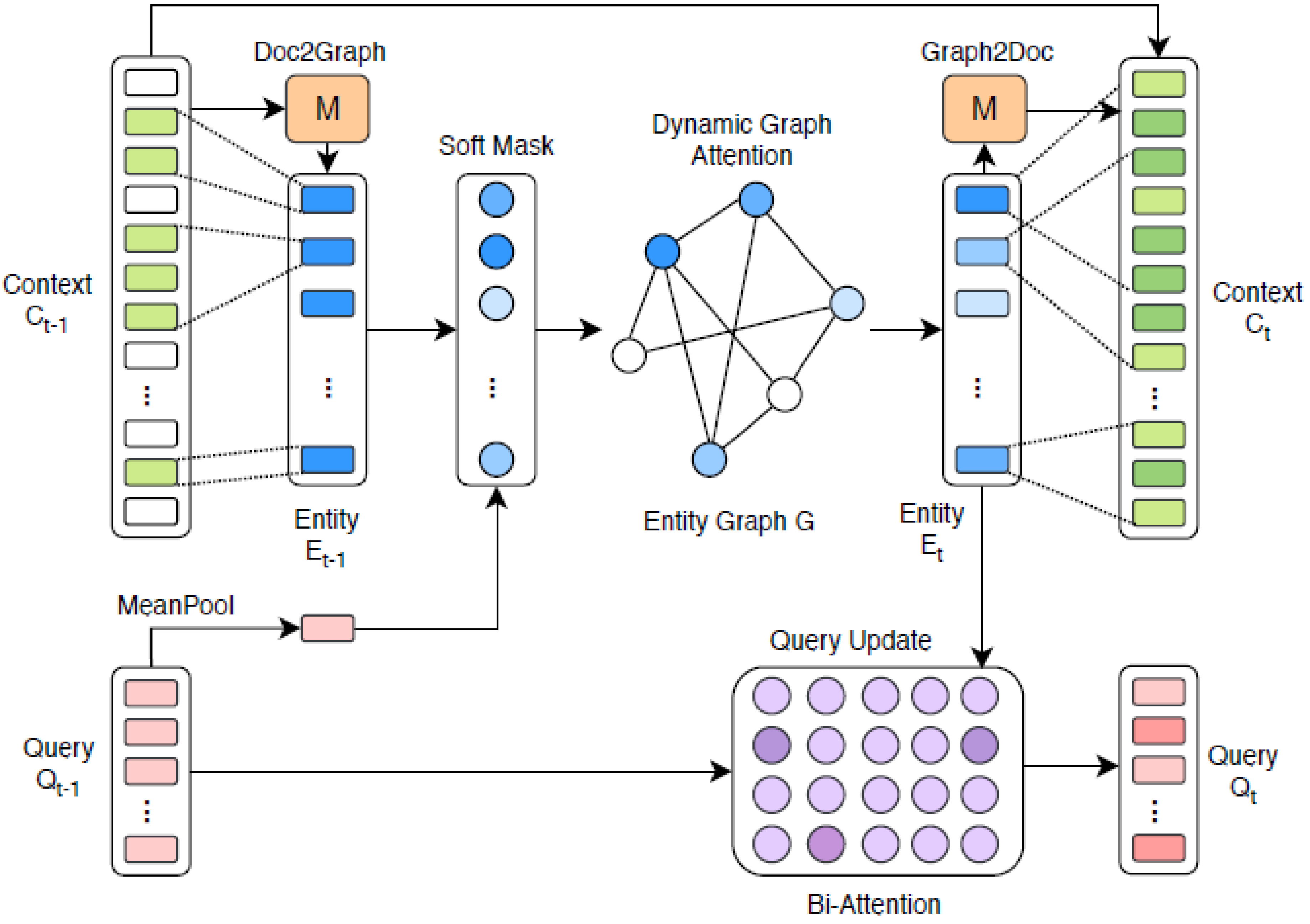
Figure 1: Entities in raw texts are modeled as an entity graph and handled by graph attention networks. When the entity graph are fully connected, a graph-attention layer will degenerate into a vanilla self-attention layer.

Experiments

Setting	Joint EM	Joint F1
Baseline	32.26	59.76
+ Graph Fusion Block	**36.45**	63.75
+ Self Attention	35.41	61.77
+ Graph Attention	35.79	61.91
+ Transformer	36.23	**63.82**
+ Masked Transformer	35.19	62.48

Table 4: Performance comparison in terms of joint EM and F1 scores under different module settings.


Input Documents
Paragraph Selector
Graph Constructor
Input Query
Context
Entity Graph
Encoder
BERT
Bi-attention
multi-hop
Fusion Block
LSTM Prediction Layer
Supporting Sentences
Answer Span
Answer Type


Doc2Graph
M
Soft Mask
Dynamic Graph Attention
Graph2Doc
M
Context C_{t-1}
Entity E_{t-1}
Entity Graph G
Entity E_t
Context C_t
MeanPool
Query Update
Query Q_{t-1}
Bi-Attention
Query Q_t

Experiments

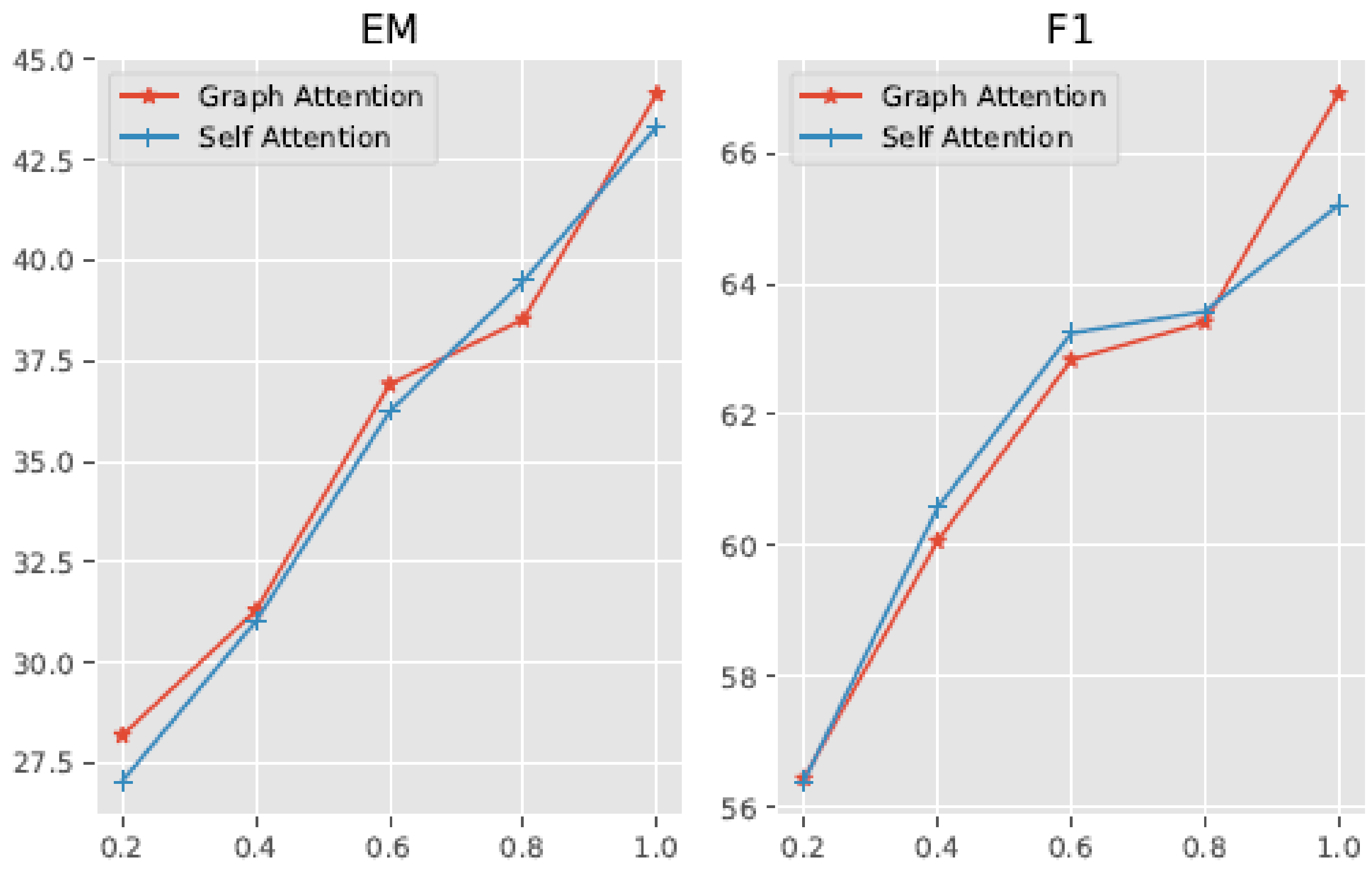


Figure 2: Results of graph-attention and self-attention on examples with different density of the adjacency matrix.

Thanks!