KOREA UNIVERSITY

DSBA Open QA Study

Open-Domain Question Answering Paper Review 3

DSBA
Data Science & Business Analytics

고려대학교 산업경영공학과

Data Science & Business Analytics Lab

발표자 : 허재혁

00

발표 목차

Contents

01 DenSPI: Real-Time Open-Domain Question Answering with Dense-Sparse Phrase Index
논문 소개

Real-Time Open-Domain Question Answering with Dense-Sparse Phrase Index

Minjoon Seo[1,5*] **Jinhyuk Lee**[6*] **Tom Kwiatkowski**[2],
Ankur P. Parikh[2] **Ali Farhadi**[1,3,4] **Hannaneh Hajishirzi**[1,3]
University of Washington[1] Google Research[2] Allen Institute for AI[3] XNOR.AI[4]
Clova AI, NAVER[5] Korea University[6]
{minjoon,ali,hannaneh}@cs.washington.edu
{tomkwiat,aparikh}@google.com jinhyuk_lee@korea.ac.kr

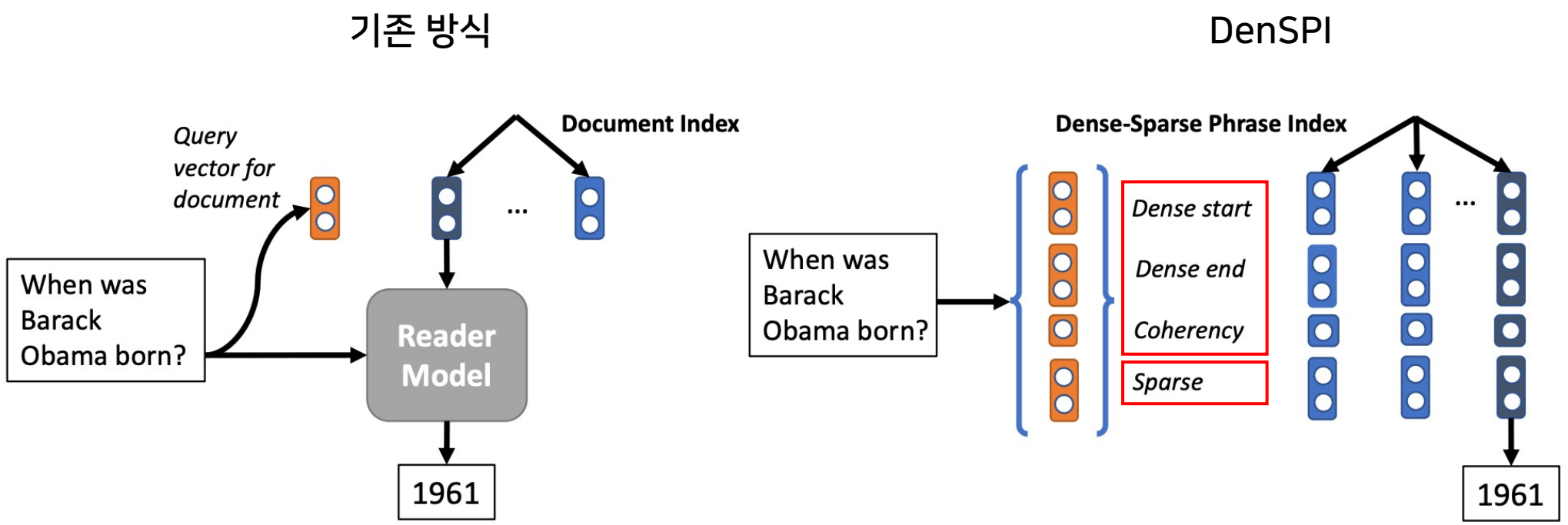
82 citations

1. 기존 Retriever + Reader로 구성된 two-stage를 one-stage로 줄임
2. Query-agnostic indexable representation 제안
3. DrQA [Chen et al., 2017] 대비 6,000배 연산량 감소, 44배 빠른 추론 속도 (CPU)

01 DenSPI: Real-Time Open-Domain Question Answering with Dense-Sparse Phrase Index

방법론

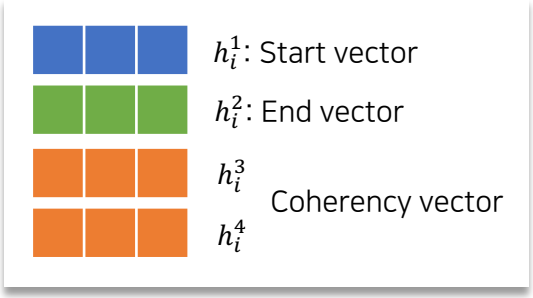
- ✓ Phrase Index를 통해 Document에 대한 Retriever 과정 없이 바로 query와 비교
- ✓ Dense (통사적, 의미적 정보) + Sparse (어휘적 정보)를 함께 사용하여 phrase vector 구성

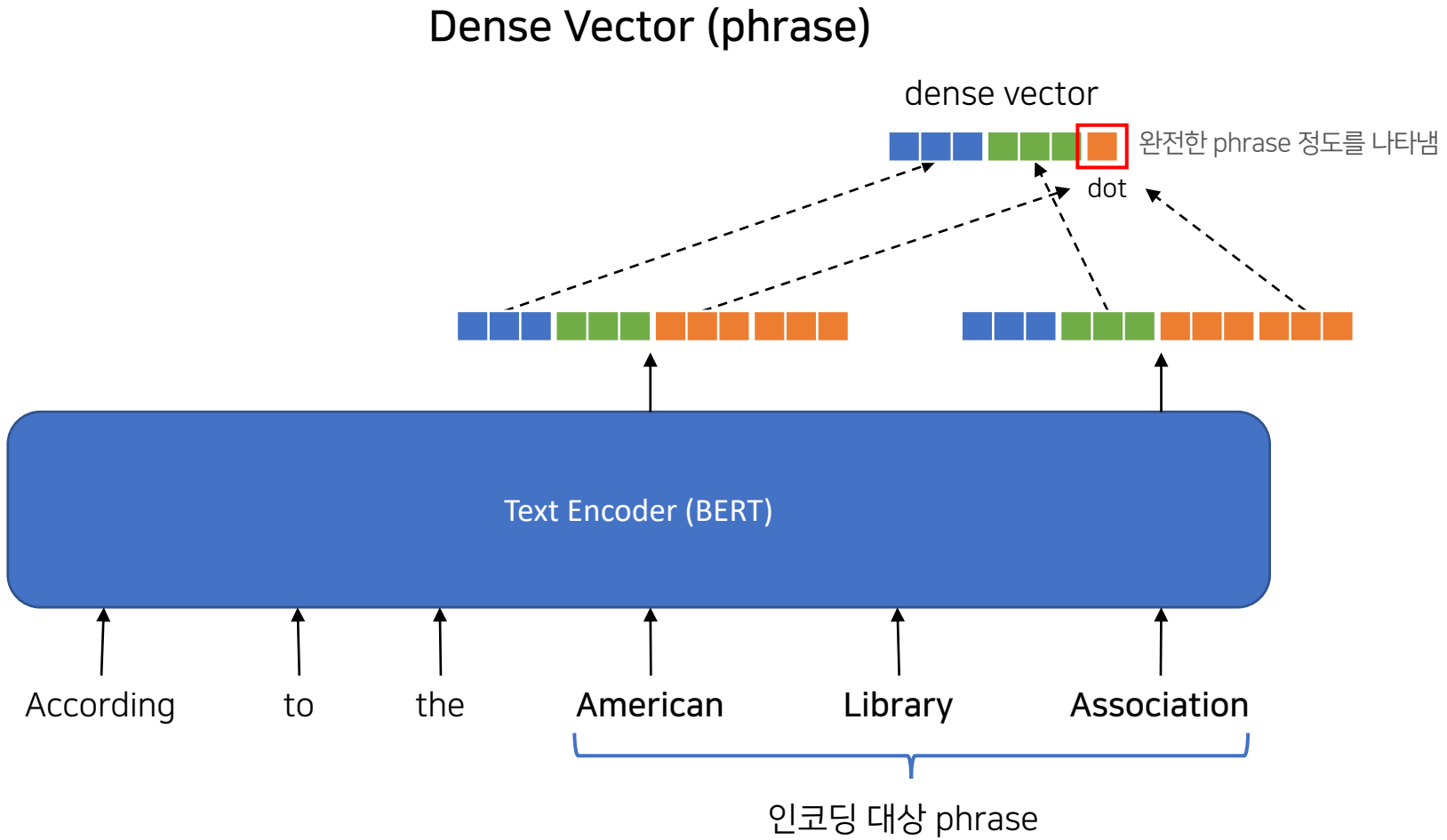


01 DenSPI: Real-Time Open-Domain Question Answering with Dense-Sparse Phrase Index

방법론

- ✓ Text Encoder의 output token dimension을 네 등분해서 start, end, coherency vector로 사용
- ✓ Target span의 start index 부분의 start vector와 end index 부분의 end vector 그리고 coherence scaler를 이용하여 phrase vector 구성
- ✓ Dense embedding model은 SQuAD v1.1 로 학습

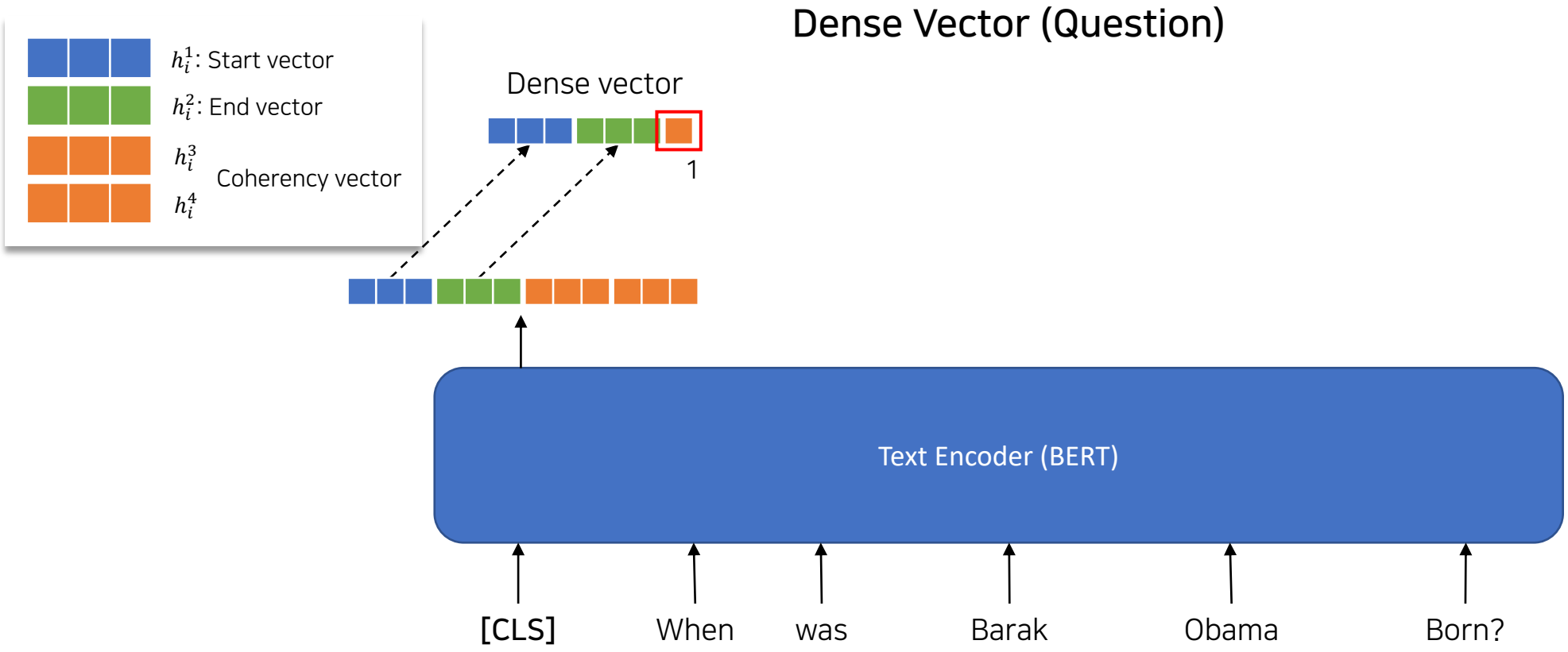




01 DenSPI: Real-Time Open-Domain Question Answering with Dense-Sparse Phrase Index

방법론

- ✓ Text Encoder의 output token dimension을 네 등분해서 start, end, coherency vector로 사용
- ✓ Question의 경우 [CLS] token output을 통해 계산



Sparse Vector

- ✓ DrQA 의 방식 기반으로 사용 [Chen et al. 2017]
- ✓ TF-IDF document & paragraph vector (Wikipedia)
- ✓ Unigram & Bigram (dim = 17 M)

01 DenSPI: Real-Time Open-Domain Question Answering with Dense-Sparse Phrase Index

방법론

- ✓ Wikipedia: 3 billion tokens, 60 billion phrases (phrase 당 4 KB) 총 240 TB
- ✓ Pointer, Filtering, Quantization을 통해 연산 비용 감소

Indexing

- ✓ **Pointer:** start, end vector 재사용
  - 240 TB -> 12 TB
- ✓ **Filter:** single layer – binary classifier로 start, end vector만 추출
  - 12 TB -> 4.5 TB
- ✓ **Quantization:** float 32 -> int8 (4 bytes -> 1 byte)
  - 4.5 TB -> 1.5 TB

01 DenSPI: Real-Time Open-Domain Question Answering with Dense-Sparse Phrase Index

방법론

- ✓ 검색 방법에 따라 실험 비교
- ✓ 결론적으로 Hybrid 가 성능 대비 속도 측면에서 가장 효율적인 방법

Search

- ✓ Sparse-first Search
- ✓ Dense-first Search
  - ✓ Faiss 사용 -> reranking sparse vector
- ✓ Hybrid
  - ✓ Start index 만 사용해서 먼저 탐색 -> reranking 결과 리스트

01 DenSPI: Real-Time Open-Domain Question Answering with Dense-Sparse Phrase Index

실험

- ✓ 초 당 처리하는 단어 수가 DrQA에 비해 약 6,000 배 빠름 + 성능도 높음
- ✓ 그러나 BERT-Large에 비해 성능이 낮음 (decomposability gap)
- ✓ Quenstion에 대한 golden answer(phrase)가 없는 SQuAD-Open 데이터에 대해 더 다양한 문서 탐색 가능

SQuAD v1.1

	Model	EM	F1	W/s
Original	DrQA	69.5	78.8	4.8K
	BERT-Large	84.1	90.9	51
Query-Agnostic	LSTM+SA	49.0	59.8	-
	LSTM+SA+ELMo	52.7	62.7	-
	DENSPI (dense only)	73.6	81.7	28.7M
	+ Linear layer	66.9	76.4	-
	+ Indep. encoders	65.4	75.1	-
	− Coherency scalar	71.5	81.5	-

SQuAD-Open

	F1	EM	s/Q	#D/Q
DrQA	-	29.8	35	5
R^3	37.5	-	-	-
Paragraph ranker	-	30.2	-	20
Multi-step reasoner	39.2	31.9	-	-
MINIMAL	42.5	34.7	-	10
BERTserini	46.1	38.6	115	-
Weaver	-	42.3	-	25
DENSPI-SFS	42.5	33.3	0.60	5
DENSPI-DFS	35.9	28.5	0.51	815
–sparse scale=0	16.3	11.2	0.40	815
DENSPI-Hybrid	44.4	36.2	0.81	817

01 DenSPI: Real-Time Open-Domain Question Answering with Dense-Sparse Phrase Index

한계점

✓ 독립적인 인코딩 방식으로 인해 기존 방식 대비 속도는 빠르지만 성능 차이 존재

Decomposability gap

$$\hat{a} = \underset{a}{\operatorname{argmax}}\, F_\theta(a, q, d)$$ 기존 방식

$$\hat{a} = \underset{a}{\operatorname{argmax}}\, G_\theta(q) \cdot H_\theta(a, d)$$ Query-agnostic

question과 answer/document 간 attention 정보가 없음

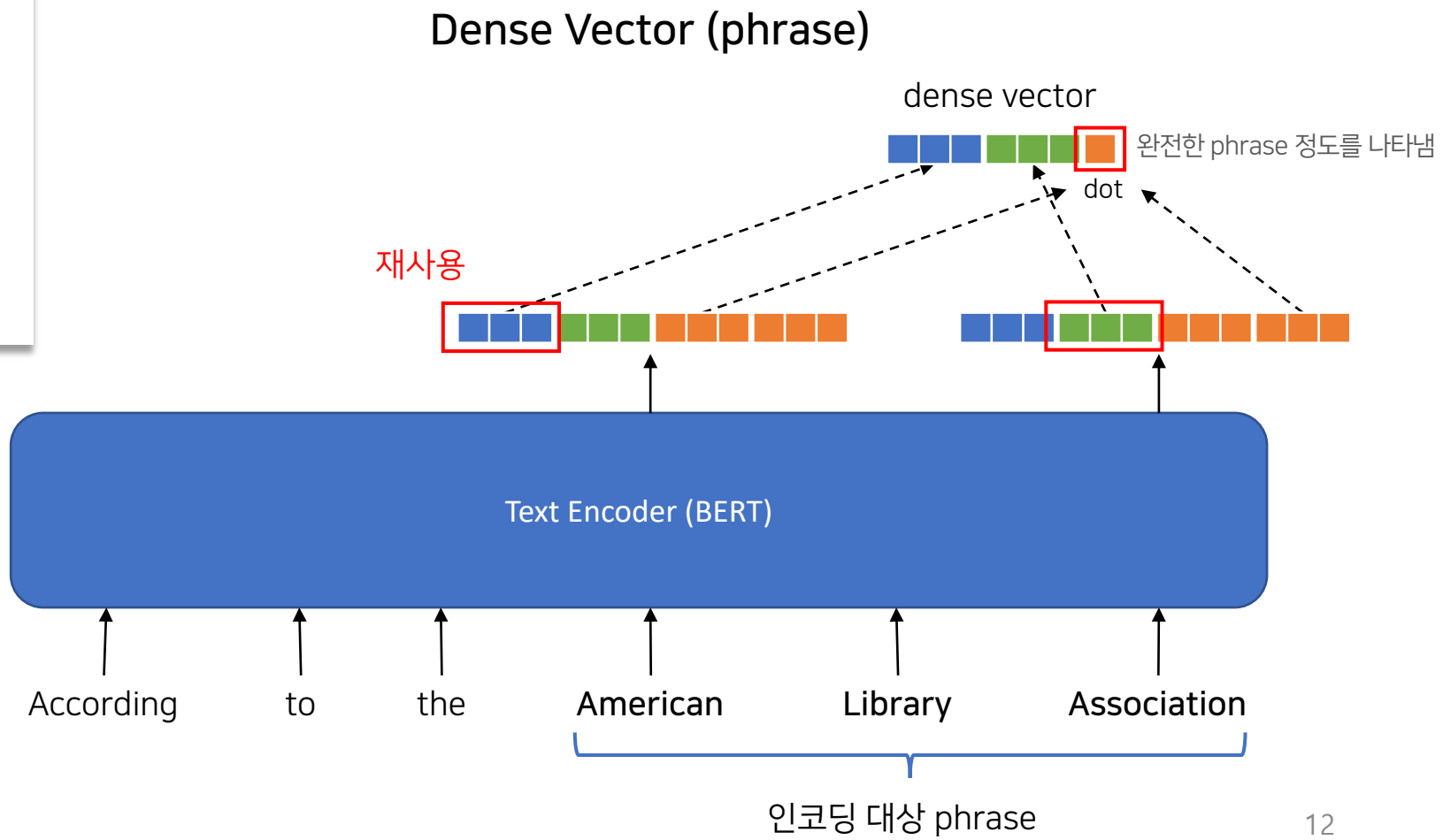
01 DenSPI: Real-Time Open-Domain Question Answering with Dense-Sparse Phrase Index

궁금

1. Pointer에서 재사용의 의미: 같은 토큰에 해당하는 embedding vector를 재사용한다?
   - ➢ Context에 따라 같은 단어여도 다른 embedding vector를 가지기 때문에 이런 의미가 아닌 것 같다고 생각

Indexing

- ✓ **Pointer:** start, end vector 재사용
- ✓ **Filter:** single layer – binary classifier로 start, end vector만 추출

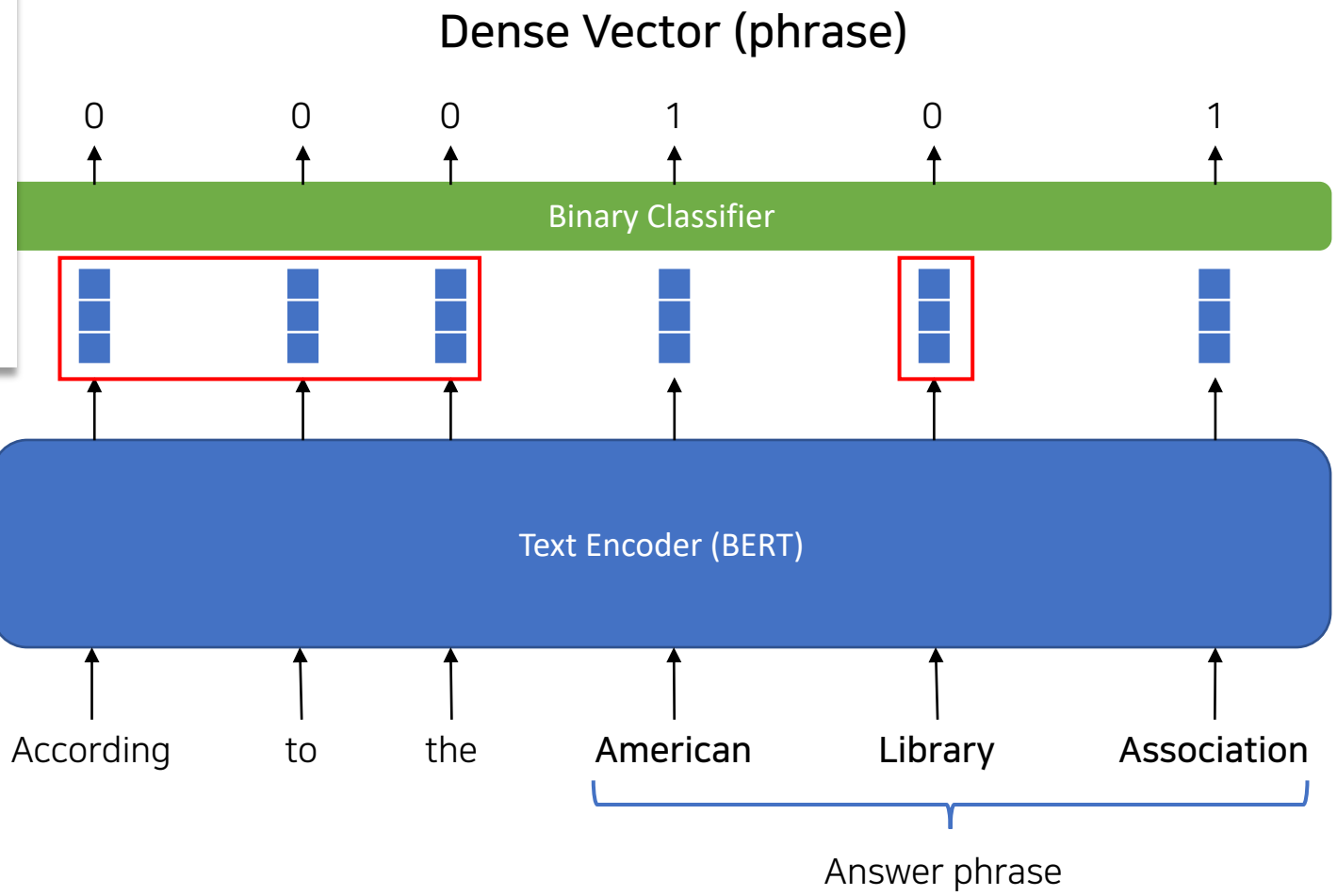


01 DenSPI: Real-Time Open-Domain Question Answering with Dense-Sparse Phrase Index 궁금

2. Filter 의미: answer phrase가 아닌 것 같은 vector를 저장하고 있을 필요가 없어짐
   - ➢ Dense embedding model 학습을 통해 phrase indexing을 하기 위해 binary classifier로 하는 방법 말고 기존에는 어떤 방법 사용?

Indexing

- ✓ **Pointer:** start, end vector 재사용
- ✓ **Filter:** single layer – binary classifier로 start, end vector만 추출



PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text

Haitian Sun Tania Bedrax-Weiss William W. Cohen
Google AI Language
{haitiansun,tbedrax,wcohen}@google.com

101 citations

1. Knowledge Bases와 Text를 통해 학습하는 GRAFT-Net을 기반으로 발전
2. 반복적으로 Knowledge bases와 Text를 통해 graph를 확장하여 retrieve 하는 방식
3. Multi-hop QA에 적합한 모델

GRAFT-Net 과 차이점

- ✓ Heuristic하게 graph를 구성하지 않음
  - ➢ 필요 이상으로 크게 구성될 수 있음
  - ➢ 정답이 그래프 안에 포함되지 않을 수 있음
- ✓ Knowledge Bases와 Corpus로부터 그래프를 확장함

02 PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text

방법론

- ✓ Document로 부터 entity 탐지를 위해 entity linker [Ji et al., 2014] 를 사용
- ✓ 여기서 document는 single sentence를 의미
- ✓ Graph-CNN은 question-answer pair를 통해 학습

Algorithm 1 PullNet

1: Initialize question graph G_q^0 with question q and question entities, with $\mathcal{V}^0 = \{e_{q_i}\}$ and $\mathcal{E}^0 = \emptyset$.

→ Quenstion 만으로 subgraph 구성

2: **for** $t = 1, \cdots, T$ **do**

3: Classify and select the entity nodes in the graph with probability larger than ϵ

$\{v_{e_i}\} = \texttt{classify\_pullnodes}\,(G_q^t, k)$

→ 학습된 Graph-CNN을 통해 나타낸 node representation으로 subgraph 내 entity 중 확장해야하는 entity를 상위 k개 선택

4: **for all** v_e in $\{v_{e_i}\}$ **do**

5: Perform pull operation on selected entity nodes

$\{v_{d_i}\} = \texttt{pull\_docs}\,(v_e, q)$

→ Lucene [McCandless et al., 2010] (IDF 기반 retrieval system)으로 document 추출

$\{v_{f_i}\} = \texttt{pull\_facts}\,(v_e, q)$

→ Question을 구성하는 각 token을 LSTM에 입력값으로 넣어 last-state representation과 relation에 대한 embedding vector를 통해 유사도 계산하여 추출 (상위 k개 인지 언급 x)

Look-up table 활용

6: **for all** v_d in $\{v_{d_i}\}$ **do**

7: Extracted entities in new document nodes

$\{v_{e(d)_i}\} = \texttt{pull\_entities}\,(v_d)$

→ Entity linker를 통해 document에서 entity 추출

8: **for all** v_f in $\{v_{f_i}\}$ **do**

9: Extract head and tail of new fact nodes

$\{v_{e(f)_i}\} = \texttt{pull\_headtail}\,(v_f)$

→ 해당 fact node들로 부터 subject entity 인지 object entity인지 반환

10: Add new nodes and edges to question graph

$G_q^{t+1} = \texttt{update}\,(G_q^t)$

→ 위에서 찾은 entity들을 더해서 edge 생성하여 subgraph 업데이트

11: Select entity node in final graph that is the best answer

$v_{\text{ans}} = \texttt{classify\_answer}\,(G_q^T)$

→ 학습된 Graph-CNN으로 subgraph 내의 entity 중 answer 예측

02 PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text

실험

✓ Multi-hop 에 대해 높은 성능 향상이 있음을 보임

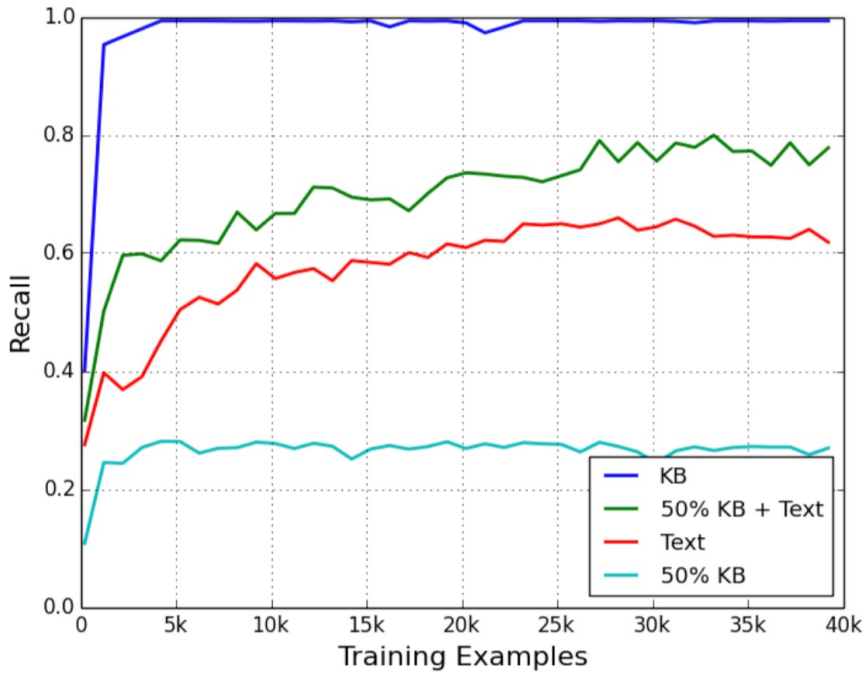
	MetaQA (1-hop) / wikimovies				MetaQA (2-hop)				MetaQA (3-hop)			
	KB	Text	50% KB	50% KB + Text	KB	Text	50% KB	50%KB + Text	KB	Text	50% KB	50% KB + Text
KV-Mem*	96.2	75.4	63.6	75.7	82.7	7.0	41.8	48.4	48.9	19.5	37.6	35.2
GraftNet*	97.0	82.5	64	91.5	94.8	36.2	**52.6**	69.5	77.7	40.2	59.2	66.4
PullNet (Ours)	97.0	84.4	65.1	**92.4**	**99.9**	**81.0**	52.1	**90.4**	**91.4**	**78.2**	**59.7**	**85.2**
KV-Mem	93.9	76.2	–	–	–	–	–	–	–	–	–	–
GraftNet	96.8	**86.6**	**68.0**	**92.6**	–	–	–	–	–	–	–	–
VRN	**97.5**	–	–	–	89.9	–	–	–	62.5	–	–	–

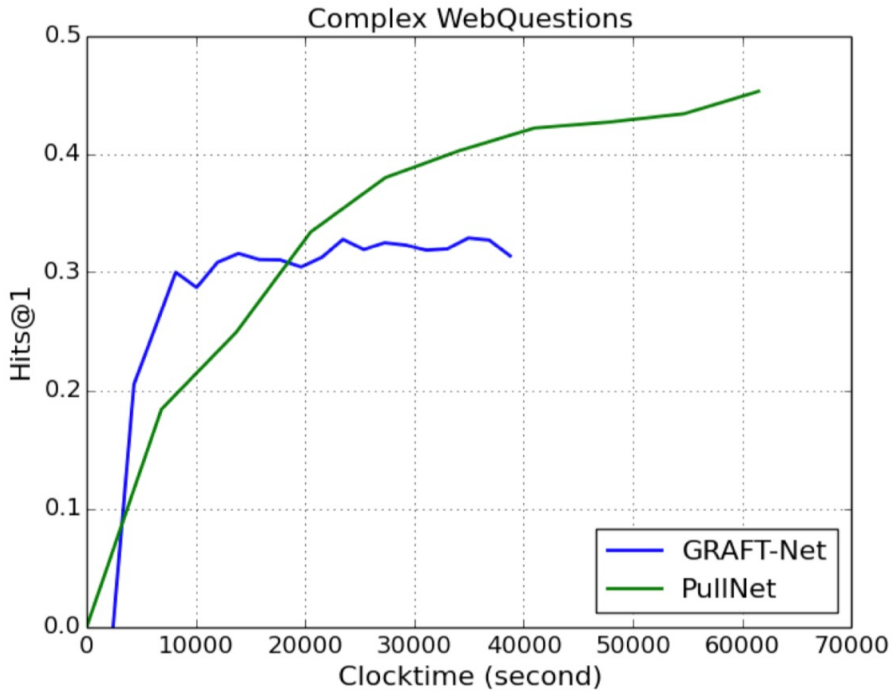
	WebQuestionsSP			
	KB	Text	50% KB	50% KB + Text
KV-Mem*	46.7	23.2	32.7	31.6
GraftNet*	66.4	24.9	48.2	49.7
PullNet (Ours)	**68.1**	24.8	**50.3**	**51.9**
GraftNet	**67.8**	**25.3**	47.7	49.9
NSM	69.0 (F1)	–	–	–

	Complex WebQuestions (dev)			
	KB	Text	50% KB	50% KB + Text
KV-Mem*	21.1	7.4	14.8	15.2
GraftNet*	32.8	10.6	26.1	26.9
PullNet (Ours)	**47.2**	**13.1**	**31.5**	**33.7**

02 PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text 실험

- ✓ Knowledge Bases 으로 PullNet이 금방 정답 entity를 찾아낼 수 있음을 보임
- ✓ GRAFT-Net에 비해 학습이 더 잘 되는 것을 보임





EOD