

# DeepIntoDeep

# Various NLP Tasks

발표자: 조혜진

# Various NLP Tasks

조혜진

Artificial Intelligence in Korea University(AIKU)

Department of Computer Science and Engineering, Korea University

# Contents

---

- BoW, TF-IDF
- Topic Modeling
- Tagging
  - Part-of-speech tagging
  - NAR
- Coreference Resolution
- Retrieval
  - Sparse
  - Dense (아래 QA에서 다룰 예정)
- Question Answering

# BoW, TF-IDF

# BoW, TF-IDF

---

- Bag of Words (BoW)
  - 단어의 순서는 고려하지 않고, 오직 등장 횟수만 고려하여 문서(document)를 표현하는 방법
  - How?
    1. Vocabulary 생성: 각 단어에 해당하는 정수 인덱스 부여  
E.g. “To live as a monster or to die as a good man”  
→ {"To": 0, "live": 1, "as": 2, "a": 3, "monster": 4, "or": 5, "die": 6, "good": 7, "man": 8}
    2. 해당 단어의 등장 횟수가 매핑 되는 인덱스에 위치하는 벡터 생성  
E.g. [2, 1, 2, 2, 1, 1, 1, 1, 1]

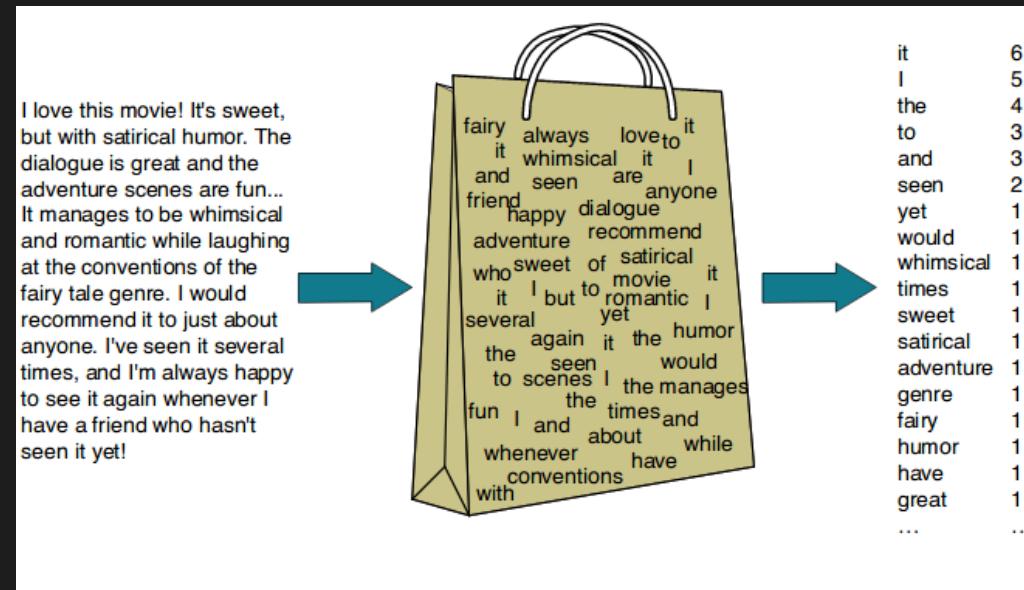
이미지 출처: <https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/#h-topic-modelling-for-feature-selection>

# BoW, TF-IDF

- Bag of Words (BoW)

- 여러 문서가 존재하는 데이터일 경우, 여러 문서에 대해 하나의 vocabulary를 만들고 이를 기반으로 BoW를 진행

- Document-Term Matrix (DTM) 이라고도 부름



이미지 출처: <https://dudeperf3ct.github.io/lstm/gru/nlp/2019/01/28/Force-of-LSTM-and-GRU/>

# BoW, TF-IDF

- Bag of Words (BoW)

- E.g. 아래와 같은 문서가 있다고 하자.
  - 문서1 : 먹고 싶은 사과
  - 문서2 : 먹고 싶은 바나나
  - 문서3 : 길고 노란 바나나 바나나
  - 문서4 : 저는 과일이 좋아요
- 그러면 여기에 대한 DTM은 다음과 같다.

	과일이	길고	노란	먹고	바나나	사과	싫은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

# BoW, TF-IDF

---

- Bag of Words (BoW)의 한계
  - Sparse representation
    - 앞의 예시에서 봤듯이, 필연적으로 0이 많을 수 밖에 없다 → 효율적 X
  - 단어들의 순서를 고려하지 X
  - 단순 등장 횟수 기반 접근
    - 불용어(stopwords): 큰 의미를 가지지 않는 단어. 한국어의 조사, 영어의 관사 등
    - 불용어가 큰 값을 가짐 → 문서의 의미 있는 부분이 반영되지 X

# BoW, TF-IDF

- TF-IDF (Term Frequency, Inverse Document Frequency)

- BoW 한계 중 세 번째 문제 (불용어 문제)를 개선한 문서 표현 방법
- $tf(t, d)$ 
  - 문서 d 내에서 단어 t의 빈도수 (혹은 상대 빈도수 =  $f_{t,d} / \sum_{t' \in d} f_{t',d}$ )
  - BoW를 계산할 때 이미 계산함
- $df(t)$ 
  - 특정 단어 t가 등장하는 문서의 수
- $idf(t, d)$ 
  - Df에 역수를 취한 것
  - $idf(t, d) = \log(\frac{n}{1+df(t)})$  ( $n$ : 총 문서의 수)
- $tf - idf = tf(t, d) * idf(t, d)$

# BoW, TF-IDF

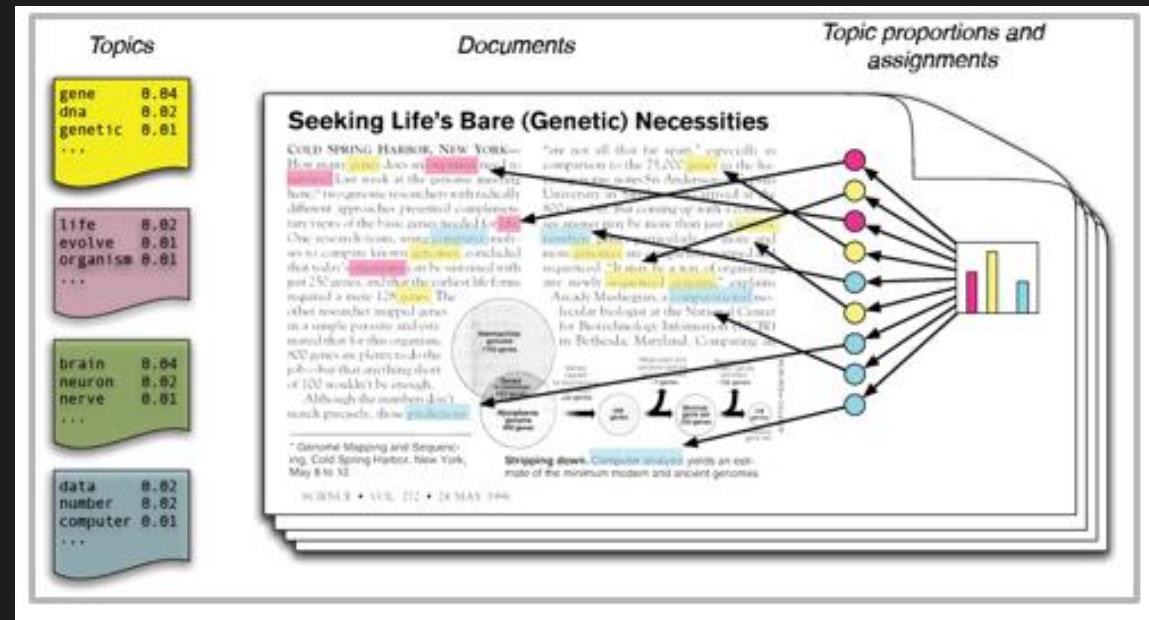
- TF-IDF (Term Frequency, Inverse Document Frequency)
  - 예시

	과일이	길고	노란	먹고	바나나	사과	싫은	저는	좋아요
문서1	0	0	0	0.288	0	0.693	0.288	0	0
문서2	0	0	0	0.288	0.288	0	0.288	0	0
문서3	0	0.693	0.693	0	0.575	0	0	0	0
문서4	0.693	0	0	0	0	0	0	0.693	0.693

# Topic Modeling

# Topic Modeling

- 문서로부터 주제(Topic)을 찾아내는 task
- Word2Vec 이전의 통계적인 방법론
- 데이터 마이닝 – 텍스트 데이터 상의 숨겨진 의미구조 파악!



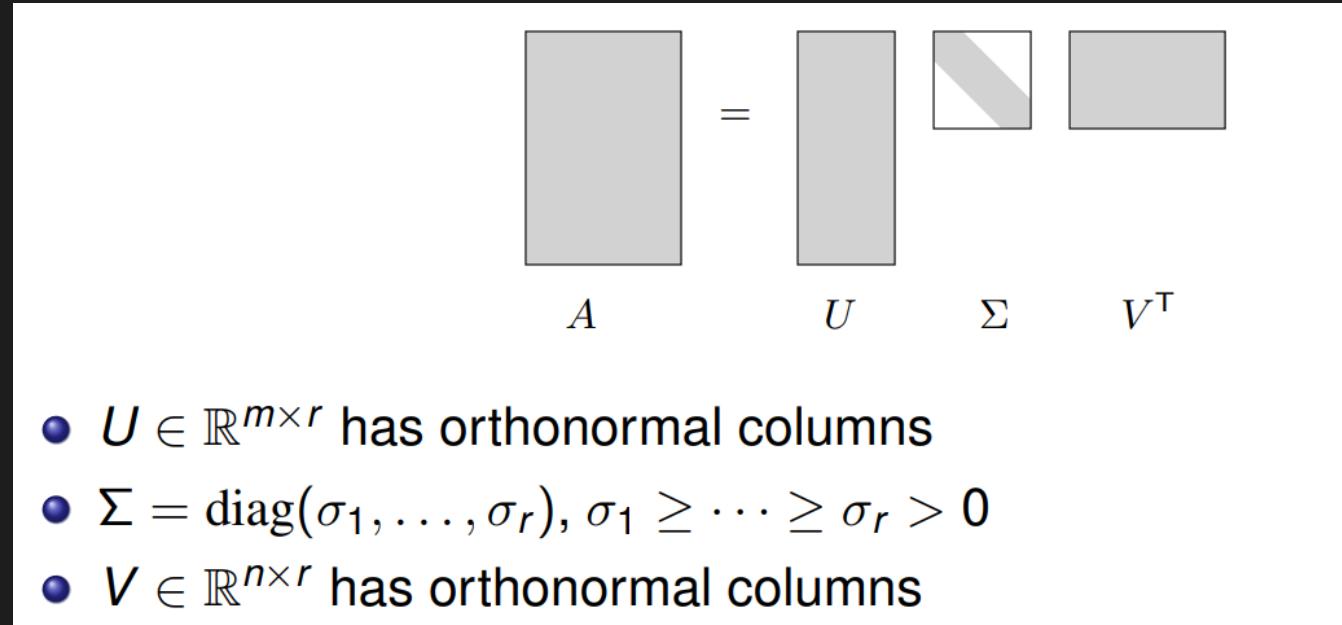
이미지 출처: <https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/#h-topic-modelling-for-feature-selection>

# Topic Modeling

- 잠재 의미 분석 (Latent Semantic Analysis – LSA)

- TF-IDF도 여전히 한계가 있었는데…
    - 단어의 의미를 고려하지 못함!
    - 문서의 잠재된 의미를 끌어내려면 어떻게 해야 할까?
  - 선형 대수학의 특이값 분해 (SVD) 이용

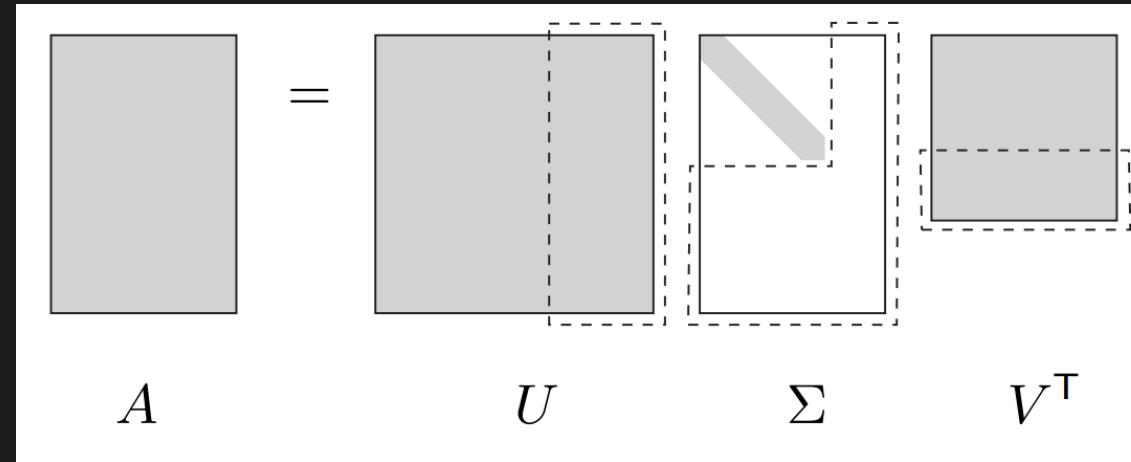
- $A = U\Sigma V^T$



이미지 출처:  
백승준 교수님 전산수학1 강의자료

# Topic Modeling

- 잠재 의미 분석 (Latent Semantic Analysis – LSA)
  - Full SVD vs Truncated SVD

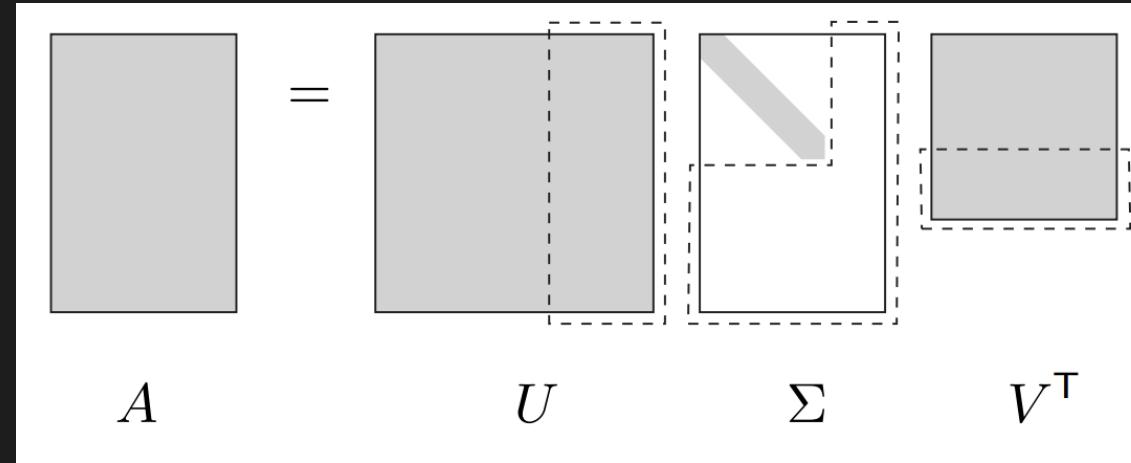


이미지 출처:  
백승준 교수님 전산수학1 강의자료

- **Truncated SVD**는 full SVD를 수행해서 나온 컴포넌트 중 일부 행/열을 절단 시킨 것
- $\Sigma$ 의 원소 (eigenvalue)는 0행 0열이 가장 크고, 밑으로 갈수록 작아짐
- → matrix의 왼쪽/위쪽에 있는 feature가 가장 중요한 정보. 갈 수록 중요도가 낮아짐

# Topic Modeling

- 잠재 의미 분석 (Latent Semantic Analysis – LSA)
  - Full SVD vs Truncated SVD

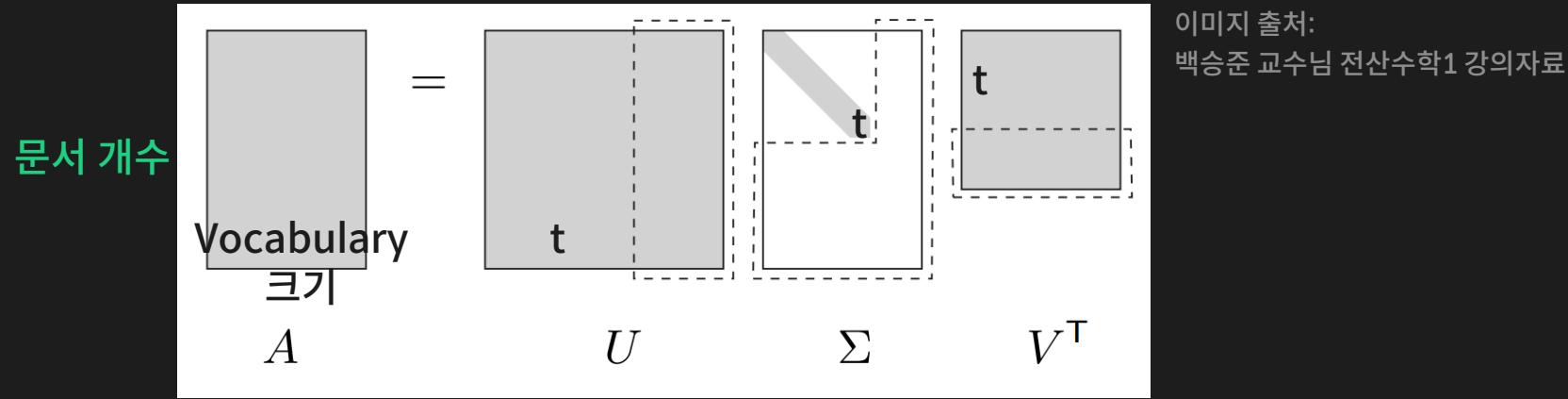


이미지 출처:  
백승준 교수님 전산수학1 강의자료

- Truncated SVD를 수행하면 별로 중요하지 않은 정보들이 잘려 나감
- → 노이즈 제거. 설명력이 높은 정보만 포함됨!
- → 심층적인 의미 확인 가능

# Topic Modeling

- 잠재 의미 분석 (Latent Semantic Analysis – LSA)



- LSA에서는 추출하고 싶은 topic 수 만큼  $t$ 를 설정
- Truncated SVD 수행 후,  $V^T$ 의 차원은  $t$  (topic 수)  $\times$  vocabulary 크기  
→ 각 topic을 구성하는 단어 집합 추출 가능!

# Topic Modeling

- 잠재 의미 분석 (Latent Semantic Analysis – LSA)

- 예시
  - 야구 커뮤니티 데이터 크롤링 후 분석
  - 뉴스 기사처럼 topic이 설명하게 나뉘는 것이 아니라 다 야구 이야기를 하기 때문에 명확하게 추출이 되지 않음…

```
Topic 1: [('오늘', 0.98334), ('경기', 0.14692), ('새끼', 0.06009), ('진짜', 0.02428), ('이유', 0.02334)]  
Topic 2: [('새끼', 0.89661), ('진짜', 0.28383), ('시발', 0.26993), ('개추', 0.12763), ('요즘', 0.05484)]  
Topic 3: [('시발', 0.84309), ('진짜', 0.36193), ('내일', 0.08098), ('경기', 0.04741), ('선발', 0.03099)]  
Topic 4: [('진짜', 0.81852), ('내일', 0.25548), ('경기', 0.08824), ('선발', 0.07179), ('투수', 0.06144)]  
Topic 5: [('내일', 0.89595), ('경기', 0.22458), ('선발', 0.18835), ('투수', 0.05542), ('야구', 0.0326)]  
Topic 6: [('경기', 0.94598), ('시즌', 0.06334), ('작년', 0.06192), ('올해', 0.05015), ('어제', 0.03439)]  
Topic 7: [('투수', 0.81163), ('야구', 0.38733), ('선발', 0.21934), ('선수', 0.20601), ('올해', 0.1621)]  
Topic 8: [('야구', 0.77232), ('선수', 0.39408), ('요즘', 0.07512), ('올해', 0.05247), ('사람', 0.03195)]  
Topic 9: [('올해', 0.94176), ('시즌', 0.11572), ('작년', 0.1109), ('이유', 0.07996), ('홈런', 0.06268)]  
Topic 10: [('선발', 0.89285), ('시즌', 0.11726), ('선수', 0.0935), ('다음', 0.05337), ('의리', 0.04311)]
```

# Topic Modeling

---

- 잠재 디리클레 할당(Latent Dirichlet Allocation - LDA)
  - LSA에서 좀 더 개선된 토픽 모델링 알고리즘
  - 통계학적인 방법 사용
  - 자세한 설명은 어려우므로 생략합니다. 궁금하신 분들은 추가 자료에 있는 논문을 읽어보세요!

# Topic Modeling

---

- Topic Modeling의 한계
  - 새로운 단어가 추가되었을 경우 전체적인 값을 다시 계산해주어야 함  
→ 새롭게 데이터가 쌓여 가는 현실 세계와 맞지 않음
  - 그러므로 요즈음에는 Word2Vec이나 GloVe와 같은 Neural Network 기반 방법이 쓰임

# Tagging

# Tagging

---

- What is Tagging?
  - 각 단어가 어떤 유형에 속해 있는지 분류하는 task
  - 대표적인 task
    - 품사 태깅 (Part-of-speech tagging)
    - 개체명 인식(NER)

# Tagging: POS tagging

---

- POS (Part-of-Speech)
  - 품사
  - 품사 태깅을 위해, 영어나 한국어에는 각 품사를 새로운 용어로 나타낸다.

# Tagging: POS tagging

- POS (Part-of-Speech)

- 영어

이미지 출처: <https://excelsior-cjh.tistory.com/entry/Chap041-Partofspeech-Tagging>

Number	Tag	Description
1	CC	Coordinating conjunction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential there
5	FW	Foreign word
6	IN	Preposition or subordinating conjunction
7	JJ	Adjective
8	JJR	Adjective, comparative
9	JJS	Adjective, superlative
10	LS	List item marker
11	MD	Modal
12	NN	Noun, singular or mass
13	NNS	Noun, plural
14	NNP	Proper noun, singular
15	NNPS	Proper noun, plural
16	PDT	Predeterminer
17	POS	Possessive ending
18	PRP	Personal pronoun

19	PRP\$	Possessive pronoun
20	RB	Adverb
21	RBR	Adverb, comparative
22	RBS	Adverb, superlative
23	RP	Particle
24	SYM	Symbol
25	TO	to
26	UH	Interjection
27	VB	Verb, base form
28	VBD	Verb, past tense
29	VBG	Verb, gerund or present participle
30	VBN	Verb, past participle
31	VBP	Verb, non-3rd person singular present
32	VBZ	Verb, 3rd person singular present
33	WDT	Wh-determiner
34	WP	Wh-pronoun
35	WP\$	Possessive wh-pronoun
36	WRB	Wh-adverb

# Tagging: POS tagging

- POS (Part-of-Speech)

- 한국어

- (세종 품사 태그)

대분류	세종 품사 태그	
	태그	설명
체언	NNG	일반 명사
	NNP	고유 명사
	NNB	의존 명사
	NR	수사
	NP	대명사
용언	VV	동사
	VA	형용사
	VX	보조 용언
	VCP	긍정 지정사
	VCN	부정 지정사
관형사	MM	관형사
부사	MAG	일반 부사
	MAJ	접속 부사
감탄사	IC	감탄사
조사	JKS	주격 조사
	JKC	보격 조사
	JKG	관형격 조사
	JKO	목적격 조사
	JKB	부사격 조사
	JKV	호격 조사
	JKQ	인용격 조사
	JX	보조사
	JC	접속 조사

이미지 출처:

<https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=aramjo&logNo=221404488280>

선어말 어미	EP	선어말 어미
어말 어미	EF	종결 어미
	EC	연결 어미
	ETN	명사형 전성 어미
	ETM	관형형 전성 어미
접두사	XPN	체언 접두사
접미사	XSN	명사 파생 접미사
	XSV	동사 파생 접미사
	XSA	형용사 파생 접미사
어근	XR	어근
부호	SF	마침표, 물음표, 느낌표
	SE	줄임표
	SS	따옴표, 괄호표, 줄표
	SP	쉼표, 가운뎃점, 콜론, 빗금
한글 이외	SO	불임표(물결, 숨김, 빠짐)
	SW	기타기호 (논리수학기호, 화폐기호)
	SL	외국어
	SH	한자
	SN	숫자

# Tagging: POS tagging

- POS (Part-of-Speech)

- 한국어

(세종 품사 태그)

```
print('꼬꼬마 형태소 분석 :',kkma.morphs("열심히 코딩한 당신, 연휴에는 여행을 가봐요"))
print('꼬꼬마 품사 태깅 :',kkma.pos("열심히 코딩한 당신, 연휴에는 여행을 가봐요"))
print('꼬꼬마 명사 추출 :',kkma.nouns("열심히 코딩한 당신, 연휴에는 여행을 가봐요"))
```

꼬꼬마 형태소 분석 : ['열심히', '코딩', '하', 'ㄴ', '당신', ',', '연휴', '에', '는', '여행', '을', '가보', '아요']

꼬꼬마 품사 태깅 : [('열심히', 'MAG'), ('코딩', 'NNG'), ('하', 'XSV'), ('ㄴ', 'ETD'), ('당신', 'NP'), (',', 'SP'), ('연휴', 'NNG'), ('에', 'JKM'), ('는', 'JX'), ('여행', 'NNG'), ('을', 'JKO'), ('가보', 'VV'), ('아요', 'EFN')]

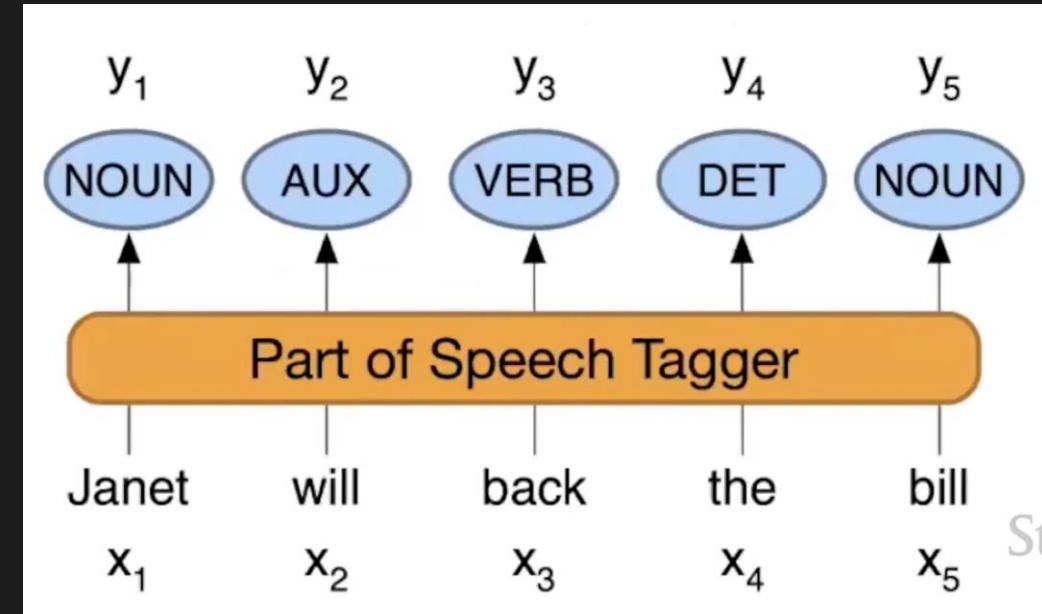
꼬꼬마 명사 추출 : ['코딩', '당신', '연휴', '여행']

# Tagging: POS tagging

- POS Tagging?
  - 문장에 있는 단어(word)들의 품사를 식별해서 품사 태그를 붙여주는 task
- Methods
  - 규칙 기반
    - 손으로 만든 규칙을 기반으로 품사를 태깅 하는 것
    - 규칙은 주로 어휘, 형태 및 구문 정보와 같은 언어의 언어적 특성에 영향 받음
    - 한계
      - 전문가(언어학자 등)이 필요
      - 언어에 따라 완벽한 규칙이 없을 수도 있음 → 오류가 날 가능성 o

# Tagging: POS tagging

- Methods
  - 통계적 방법론
    - Hidden Markov Model
    - Maximum Entropy Markov Model
  - ML 방법론
    - 베이지안, SVM 등
  - Deep Learning 방법론
    - Sequence Labeling



이미지 출처: Stanford CS124

# Tagging: NER

- **개체명 인식 (Named Entity Recognition) 이란?**
  - 문장에서 개체명 (Named Entity – NE)를 인식하는 task
- **개체명 (Named Entity) 이란?**
  - 이름을 가진 개체
  - 미리 정의해 둔 사람, 회사, 장소, 시간, 단위 등에 해당하는 단어
- **NER task는 어떤 식으로 수행해야 할까?**
  - 기존 품사 태깅 task처럼, 각 토큰(품사 태깅에서는 단여였지만, 여기서는 토큰입니다)을 태깅
  - 이 때 여러 개의 토큰이 하나의 개체명이 될 수도 있음!
    - **BIO 시스템**을 사용하여 여러 토큰을 하나의 개체명으로 묶어줌

# Tagging: NER

- BIO System

- 기존 개체명 태그(사람, 장소 등) 옆에 **추가적인 태그 (B, I, O)**를 붙여 준다!
- B: Begin. 토큰이 개체명의 시작 지점일 때 붙음
- I: Inside. 토큰이 개체명의 중간이나 끝일 때 붙음
- O: Outside. 토큰이 개체명이 아닐 때 붙음

B-PER	I-PER	I-PER	O	O	O	B-LOC	OB-LOC	B-LOC	O
Michael	Jeffery	Jordan	was	born	in	Brooklyn	, New	York	.

# Tagging: NER

---

- How?
  - POS tagging처럼,
    - 규칙 기반
    - Unsupervised learning 기반: Clustering (문맥적 유사도 기반)
    - Supervised learning 기반: 딥 러닝 모델을 사용하여 Sequence Labeling
  - 등의 방법을 사용한다!

# Coreference Resolution

# Coreference Resolution

- 단어에서 동일한 대상(entity)를 가리키는 지칭(mentions)을 찾는 것

Barack Obama nominated Hillary Rodham Clinton as his secretary of state on Monday. He chose her because she had foreign affairs experience as a former First Lady.

# Coreference Resolution

- 단어에서 동일한 대상(entity)를 가리키는 지칭(mentions)을 찾는 것

Barack Obama nominated Hillary Rodham Clinton as his secretary of state on Monday. He chose her because she had foreign affairs experience as a former First Lady.

# Coreference Resolution

- 단어에서 동일한 대상(entity)를 가리키는 지칭(mentions)을 찾는 것



Barack Obama nominated Hillary Rodham Clinton as **his** secretary of state on Monday. **He** chose her because she had foreign affairs experience as a former First Lady.

# Coreference Resolution

- 단어에서 동일한 대상(entity)를 가리키는 지칭(mentions)을 찾는 것



Barack Obama nominated Hillary Rodham Clinton as his secretary of state on Monday. He chose her because she had foreign affairs experience as a former First Lady.

# Coreference Resolution

---

- Applications

- Text에 대한 전반적인 이해
- Machine translation
- Dialogue Systems
  - “Book tickets to see **James Bond**”
  - “**Spectre** is playing near you at 2:00 and **3:00** today. **How many tickets would you like?**”
  - “**Two** tickets for the showing at **three**”

# Coreference Resolution

---

- How? 두 단계에 걸쳐 이루어짐
  1. 지칭어(mentions)을 detect (easy)  
“[I] voted for [Nader] because [he] was most aligned with [[my] values],” [she] said
  2. Detected된 지칭어를 동일한 대상끼리 분류 (hard)  
“[I] voted for [Nader] because [he] was most aligned with [[my] values],” [she] said

# Coreference Resolution

---

- 1. Mention Detection
  - Mention(지칭어)의 종류로는 세 가지가 있다!
    - Pronouns (대명사)
      - I, your, it, she, him, …
    - Named entities (개체명)
      - People, places, Paris, Joe Biden, Nike
    - Noun phrases (명사구)
      - A dog, the big fluffy cat stuck in the tree

# Coreference Resolution

---

- 1. Mention Detection
  - Mention의 종류별로 detect하는 방법이 다르다!
    - Pronouns (대명사)
      - POS tagger 사용
    - Named entities (개체명)
      - NER tagger 사용
    - Noun phrases (명사구)
      - Parser 사용

# Coreference Resolution

---

- 1. Mention Detection

- Bad mentions: mention처럼 보이지만 mention이라고 하기엔 애매한 것들…

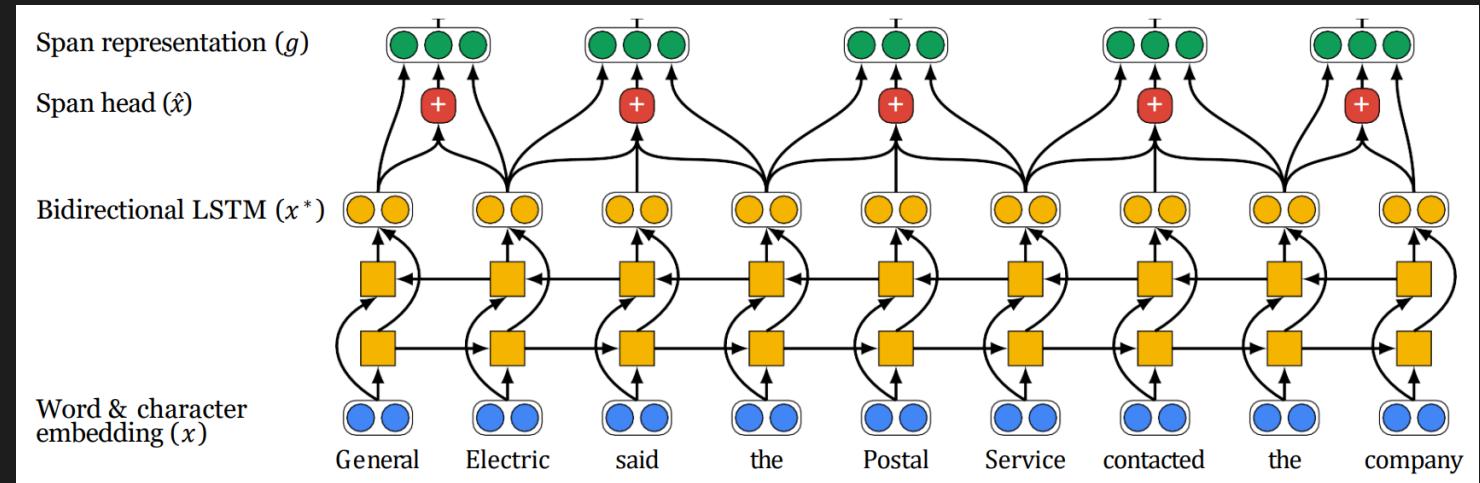
- It is sunny
  - Every student
  - No student
  - The best donut in the world
  - 100 miles

→ 어떻게 처리해야 할까?

- Bad mention을 걸러내는 classifier를 학습시킨다
  - 모든 mention을 “candidate mention”으로 취급하고, detection이 끝난 후 모든 singleton mention들을 버린다!

# Coreference Resolution

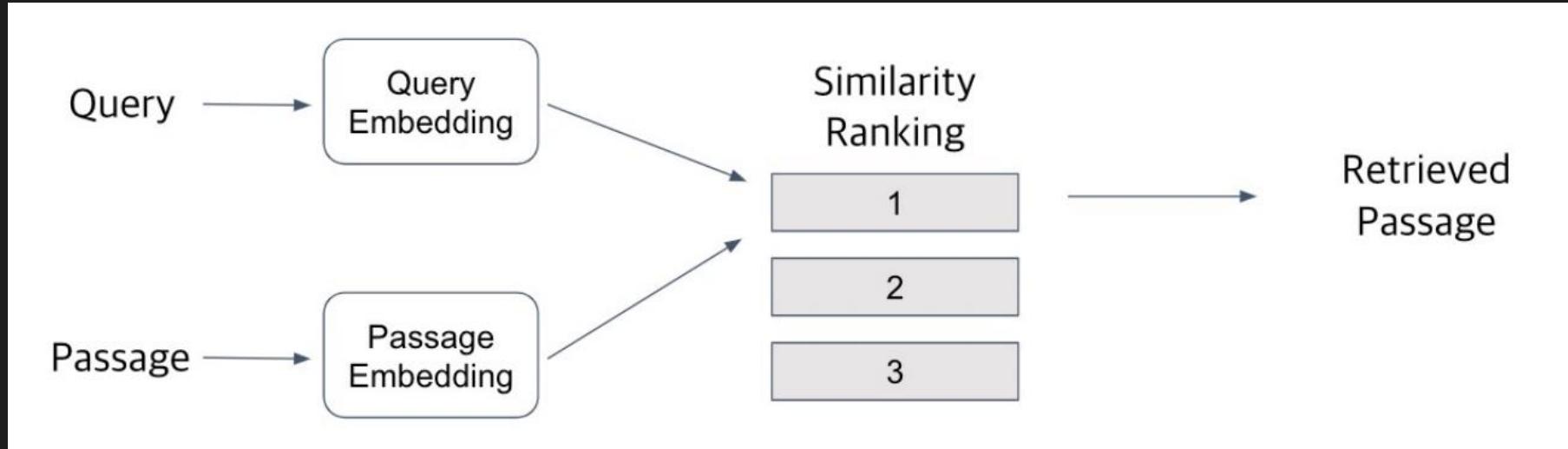
- 2. Coreference Models
  - Rule-based
  - Mention-Pair
  - Mention Ranking
  - Clustering
  - End-to-End Neural Model



# Retrieval

# Retrieval

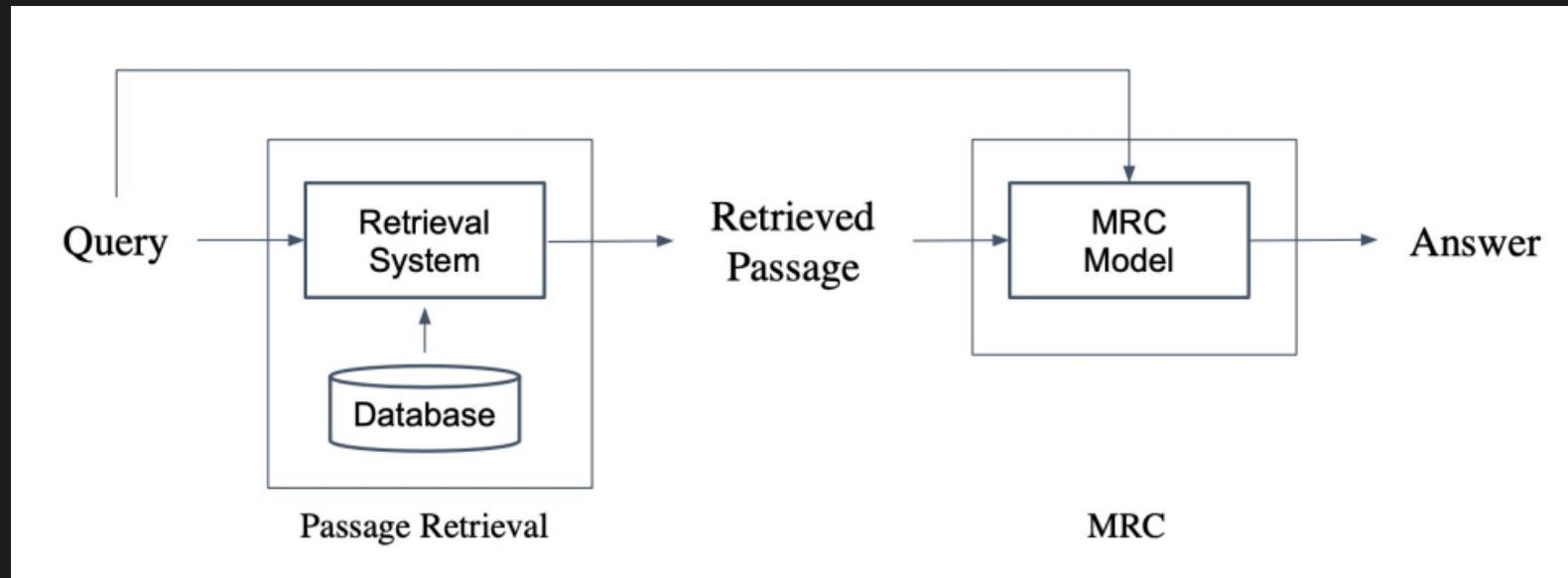
- Passage Retrieval이라고도 하며,  
Query와 가장 관련 있는 passage를 찾아 주는 task



- Query와 Passage Embedding의 유사도를 기준으로 순위를 매기고 가장 높은 것을 채택

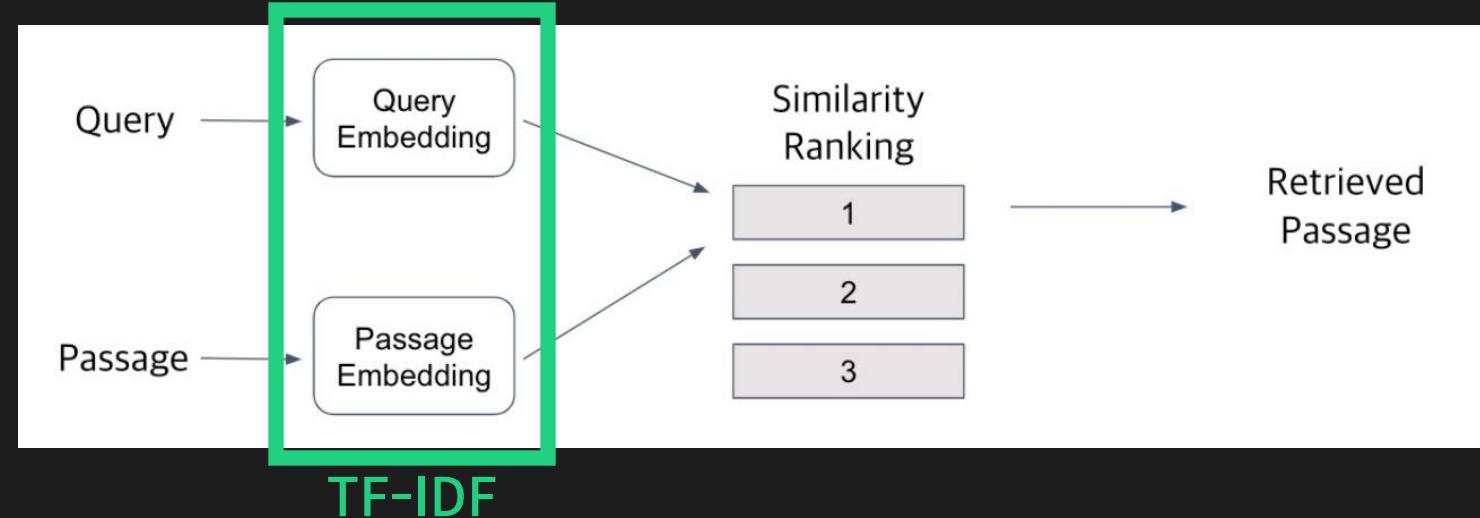
# Retrieval

- Task에 사용되는 embedding의 종류에 따라 두 가지로 구분할 수 있다.
  - Sparse retrieval
  - Dense retrieval
- 뒤에서 배울 **Open-domain Question Answering**에 유용하게 사용된다!



# Retrieval: Sparse Retrieval

- **Sparse embedding**을 이용해서 retrieval을 수행하는 방법
- Neural network보다는 통계적인 방법론 위주
- **TF-IDF** 알고리즘
  - Query의 TF-IDF vector에 대해 각 Passage의 TF-IDF vector를 내적
  - 가장 값이 큰 passage를 채택



# Retrieval: Sparse Retrieval

- BM25 알고리즘
    - TF-IDF 알고리즘에서, 문서의 길이까지 고려하여 embedding 생성
    - TF-IDF 계열 알고리즘 중 SOTA 성능
    - 실제 검색엔진에서도 쓰이는 알고리즘

$$score(D, Q) = \sum_{i=1}^n IDF(q_i) * \frac{f(q_i, D) * (k_1 + 1)}{f(q_i, D) + k_1 * (1 - b + b * \frac{|D|}{avgdl})}$$

문서  $D$ 에서  $q_i$ 의 term frequency  
 $f(q_i, D) * (k_1 + 1)$  문서  $D$ 의 길이  
 $f(q_i, D) + k_1 * (1 - b + b * \frac{|D|}{avgdl})$   
파라메터      문서 집합의 평균 문서 길이

이미지 출처: <https://velog.io/@jkl133/BM25>

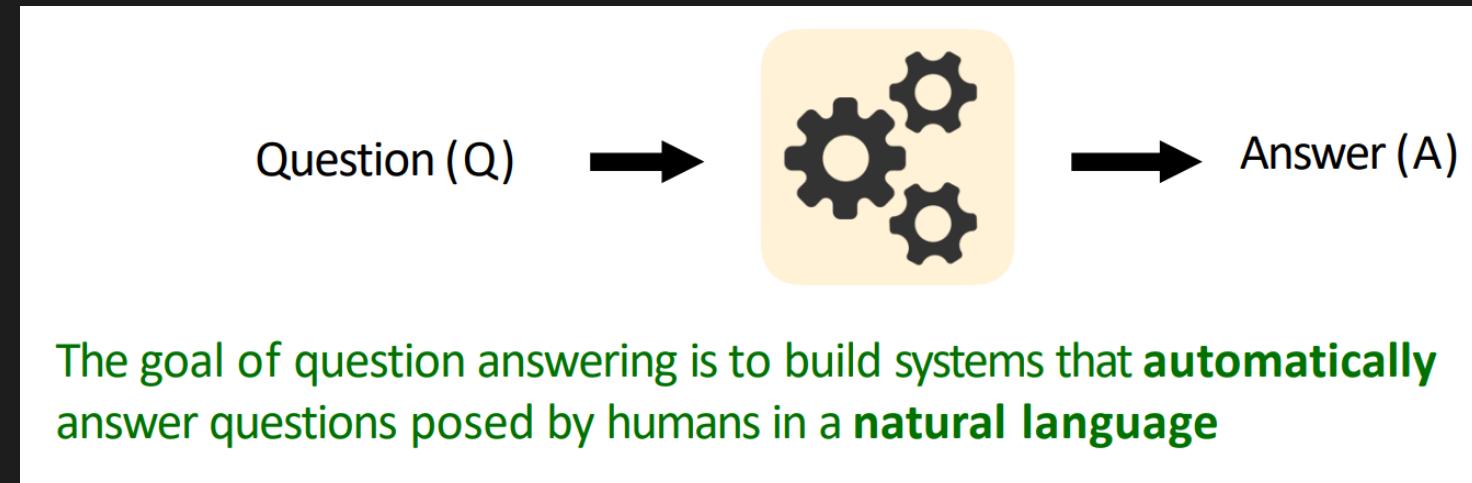
# Retrieval: Dense Retrieval

- 바로 다음 장인 Question Answering task와 연관되므로, 거기서 설명드릴게요!

# Question Answering

# Question Answering

- Question Answering?
  - 주어진 질문에 대해 답변을 내는 task



# Question Answering

---

- Task는 기준에 따라 다양하게 분류될 수 있다.
  - 어떤 information source를 기준으로 답을 내는가?
    - Text passage, web docs, knowledge bases, 표, 이미지, …
  - Question type
    - 진실 vs 거짓, open-domain vs closed-domain, simple vs compositional, …
  - Answer type
    - Text의 짧은 부분, paragraph, list, 예/아니오, …

# Question Answering

- 무궁구진한 활용 가능성이 있다!

A screenshot of a Google search results page. The search query "Where is the deepest lake in the world?" is entered in the search bar. Below the search bar, there are several navigation links: All (selected), Maps, Images, News, Videos, More, Settings, and Tools. A message indicates "About 21,100,000 results (0.71 seconds)". Below this, there are four images: a landscape photo of a deep blue lake, a 3D cross-section diagram of Lake Baikal showing its depth, another landscape photo of the lake, and a satellite map highlighting the lake's location. Below the images, the text "Siberia" is displayed, followed by a paragraph about Lake Baikal.

Where is the deepest lake in the world?

All Maps Images News Videos More Settings Tools

About 21,100,000 results (0.71 seconds)

## Siberia

Lake **Baikal**, in Siberia, holds the distinction of being both the deepest lake in the world and the largest freshwater lake, holding more than 20% of the unfrozen fresh water on the surface of Earth.

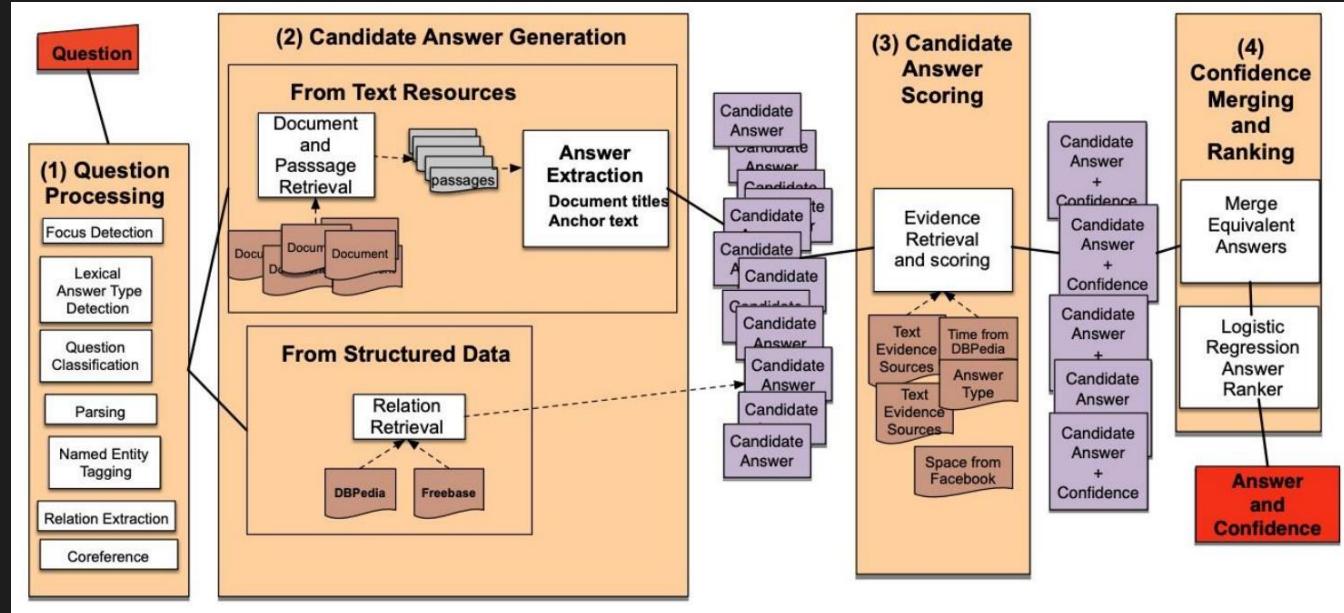
# Question Answering

- 무궁구진한 활용 가능성이 있다!



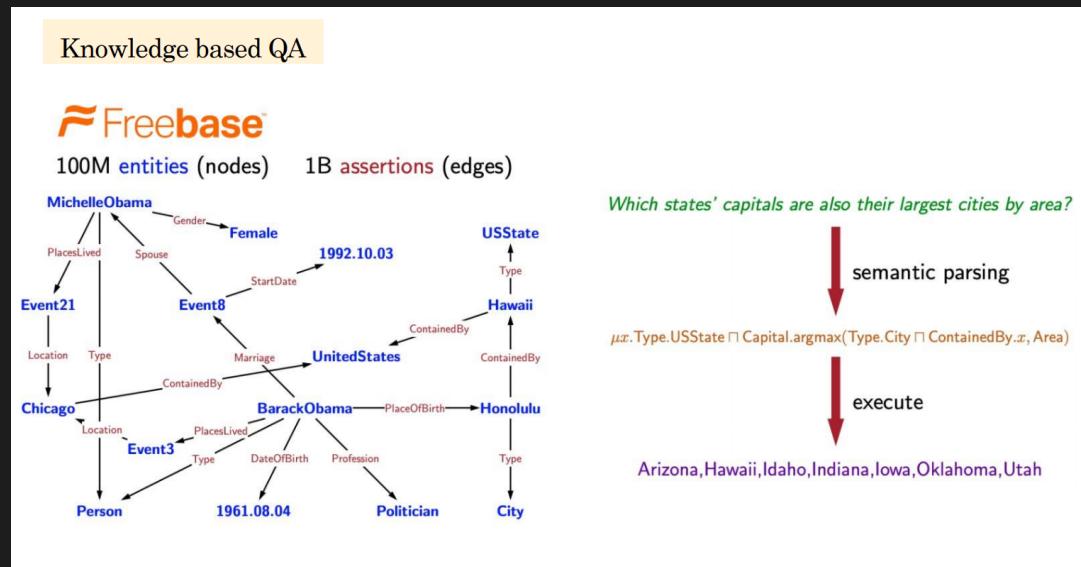
# Question Answering

- 딥러닝 이전 QA 모델: IBM Watson



# Question Answering

- Textual QA task 외에도 unstructured text 기반 QA task 및 image 기반 visual QA task 등 여러 형태의 QA task가 존재한다!
- 하지만 이 시간에는 textual QA task에만 집중할 것



# Question Answering

- Reading comprehension
  - 주어진 passage를 읽고 관련된 문제에 답변하는 것
  - $(P, Q) \rightarrow A$
  - E.g.
    - 태풍 또는 열대폭풍은 열대 해상에서 발생한 열대저기압이 발달하여, 중심 부근의 최대 풍속이 17.2m/s 이상의 강한 폭풍우를 동반한 국지적 기상 현상을 말한다. 태풍과 같은 열대폭풍은 발생 지역에 따라 명칭이 다르다. 인도양과 남태평양에서 발생하면 사이클론(cyclone)이라고 하며, 북태평양 중부와 동부, 북대서양 서부에서는 최대 풍속 32.7 m/s 이상의 열대저기압 폭풍은 허리케인(Hurricane)이라고 한다. 브라질 동쪽 남대서양에서는 거의 발생하지 않아 명칭이 정의되어 있지 않지만, 브라질에서는 사이클론, 미국에서는 허리케인으로 부른다. 과거 호주에서는 원주민의 언어로 콩포, 우울을 뜻하는 윌리윌리(willy-willy)로 불렸지만 현재는 사이클론으로 불린다. 각 지역마다 발생 기준에 차이가 있으며, 코리올리 힘의 영향으로 북반구에서는 반시계 방향으로 남반구에서는 시계 방향으로 회전한다. (후략)
    - Q. 호주에서 한때 태풍을 무엇이라고 불렀는가?

# Question Answering

- Reading comprehension이 왜 중요한가?
  - Computer system이 어느 정도로 인간의 언어를 이해하는지 측정할 수 있다!
  - 많은 다른 NLP task는 reading comprehension problem으로 환원될 수 있다!

**Information extraction**  
(Barack Obama, educated\_at, ?)

Question: Where did Barack Obama graduate from?

Passage: Obama was born in Honolulu, Hawaii.  
After graduating from Columbia University in 1983,  
he worked as a community organizer in Chicago.

(Levy et al., 2017)

**Semantic role labeling**

UCD **finished** the 2006 championship as Dublin champions ,  
by **beating** St Vincents in the final .

**finished**

Who finished something? - UCD  
What did someone finish? - the 2006 championship  
What did someone finish something as? - Dublin champions  
How did someone finish something? - by beating St Vincents in the final

**beating**

Who beat someone? - UCD  
When did someone beat someone? - in the final  
Who did someone beat? - St Vincents

(He et al., 2015)

# Question Answering

- Stanford question answering dataset (SQuAD)
  - 프로그래밍 과제 4에서, 저희는 KorQuAD라는 데이터셋을 이용해 Question answering과 Question generation task를 수행해 볼 겁니다.
  - KorQuAD는 SQuAD 데이터셋을 벤치마킹하여 한국어로 된 질의응답 데이터셋으로, 해당 데이터를 이용하려면 SQuAD에 대해서도 알아야겠죠?

# Question Answering

---

- Stanford question answering dataset (SQuAD)
  - 스탠포드 대학교에서 제작한 QA dataset
  - 100k개의 (passage, question, answer) 데이터셋
  - Passage는 English Wikipedia에서 긁어옴
    - 주로 100~150개의 단어로 구성
  - Question은 크라우드소싱으로 수집
  - 각 answer는 짧은 text segment(span)로 구성
    - 해당 데이터로 학습된 모델은 긴 답변은 생성하지 못한다. 한계.

# Question Answering

- Stanford question answering dataset (SQuAD)
  - Evaluation
    - Exact match
      - prediction이 GT와 정확히 같으면 1, 아니면 0
    - F1 score
      - Prediction과 GT가 일부라도 겹치면 점수 부여

The definition of true positive (TP), true negative (TN), false positive (FP), false negative (FN)			
		Tokens in Reference	Tokens Not in Reference
tokens in candidate	TP	FP	
tokens not in candidate	FN	TN	

$Precision = \frac{\text{num}(\text{same\_token})}{\text{num}(\text{pred\_tokens})}$

$Recall = \frac{\text{num}(\text{same\_token})}{\text{num}(\text{groud\_tokens})}$

$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$

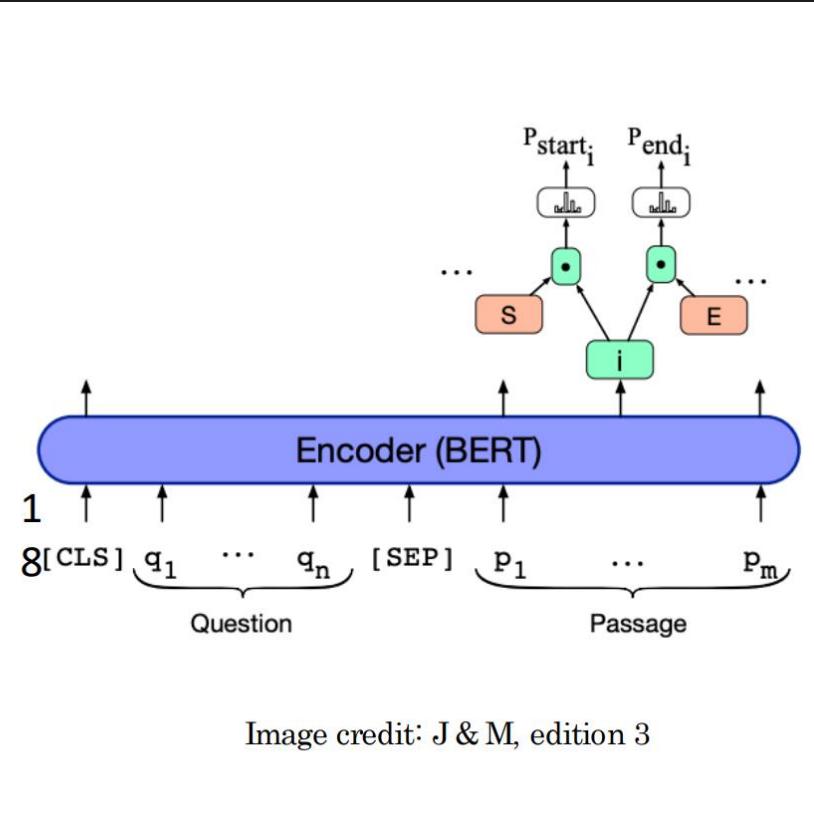
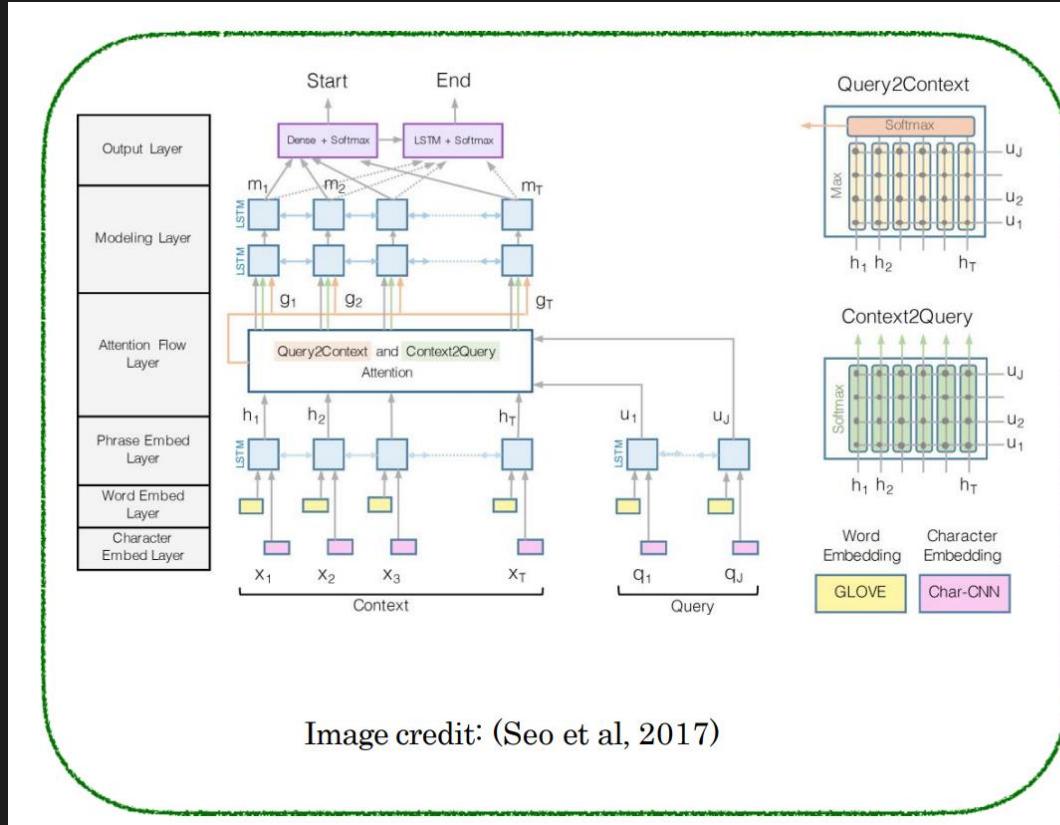
이미지 출처 <https://www.ohsuz.dev/mrc-cookbook/mrc-lecture-01>

# Question Answering

- SQuAD 데이터셋을 이용해 어떻게 모델을 학습시킬까?
  - Problem formulation
    - Input:  $C = (c_1, c_2, \dots, c_N), Q = (q_1, q_2, \dots, q_M), c_i, q_i \in V$
    - Output:  $1 \leq start \leq end \leq N$
  - C: context(=passage), Q: question
  - $N \sim 100, M \sim 15$
- LSTM-based model(~2018)과 BERT-like model(2019~)로 학습

# Question Answering

- SQuAD 데이터셋을 이용해 어떻게 모델을 학습시킬까?

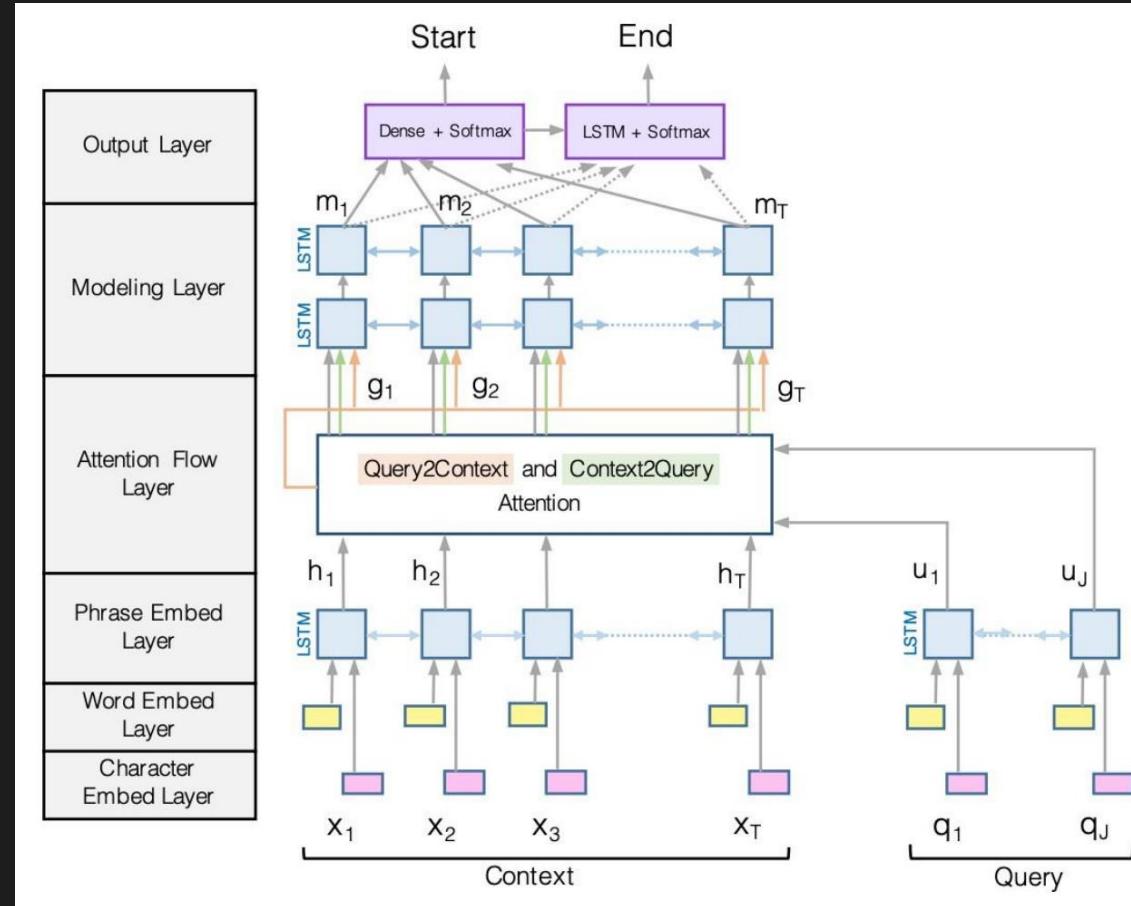


# Question Answering

- Recap: Seq2seq model w/ attention
  - Source와 target sequence 대신 **passage**와 **question**을 사용
  - Attention → passage에 있는 단어 중 question과 가장 관련 있는 단어 알아내기 (그 역도)
  - 문장을 생성하는 것이 아니라, start와 end token을 예측하는 것이므로 autoregressive decoder는 필요 없다!

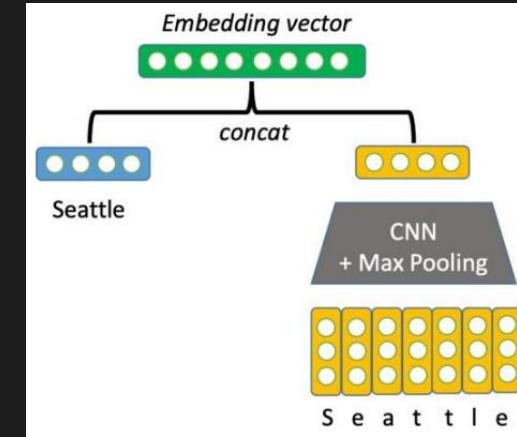
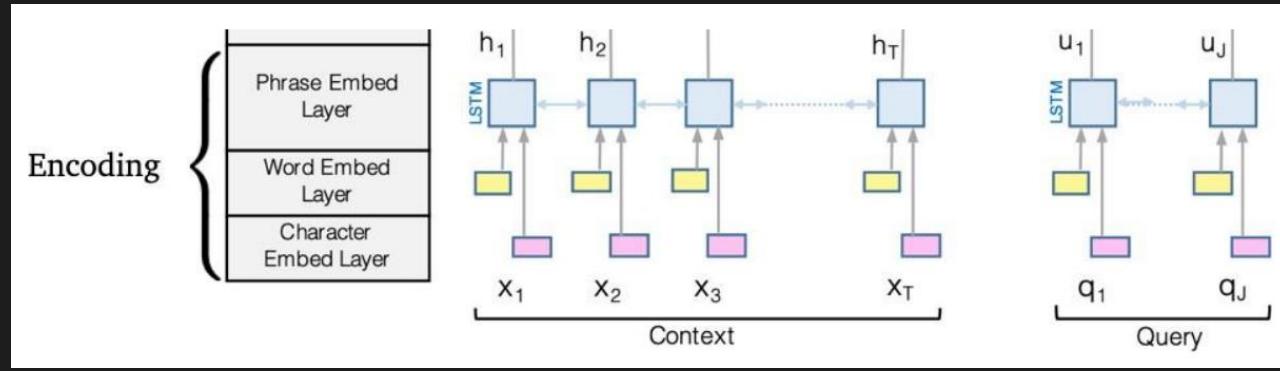
# Question Answering

- BiDAF: the Bidirectional Attention Flow model



# Question Answering

- BiDAF: Encoding



- Context와 query에 대해 word embedding(GloVe)와 character embedding(CNN 적용) 수행

$$e(c_i) = f([GloVe(c_i); charEmb(c_i)]), \quad e(q_i) = f([GloVe(q_i); charEmb(q_i)])$$

- 그 후 각각에 bidirectional LSTM을 적용하여 contextual embedding 구하기

$$\vec{c}_i = LSTM(\vec{c}_{i-1}, e(c_i)) \in \mathbb{R}^H$$

$$\hat{c}_i = LSTM(\hat{c}_{i+1}, e(c_i)) \in \mathbb{R}^H$$

$$c_i = [\vec{c}_i; \hat{c}_i] \in \mathbb{R}^{2H}$$

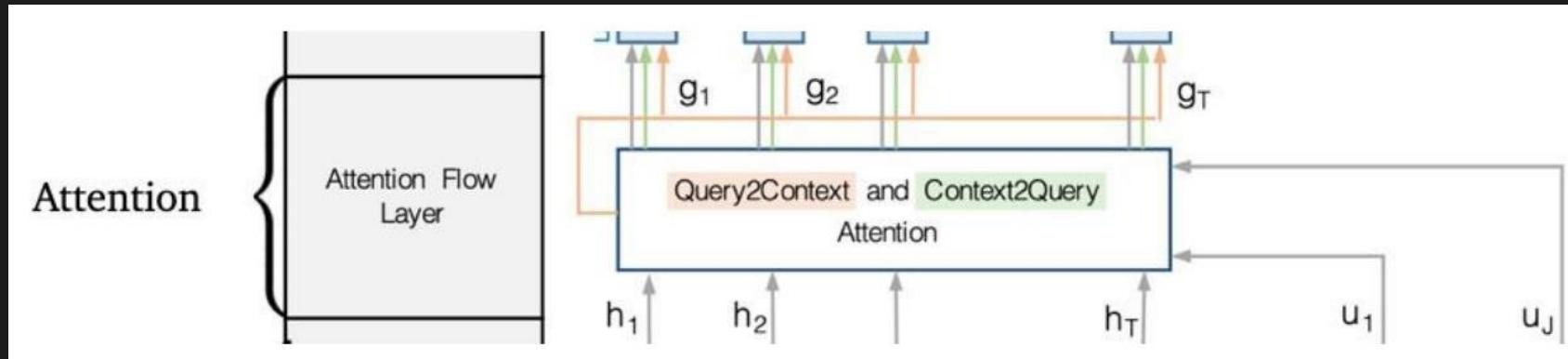
$$\vec{q}_i = LSTM(\vec{q}_{i-1}, e(q_i)) \in \mathbb{R}^H$$

$$\hat{q}_i = LSTM(\hat{q}_{i+1}, e(q_i)) \in \mathbb{R}^H$$

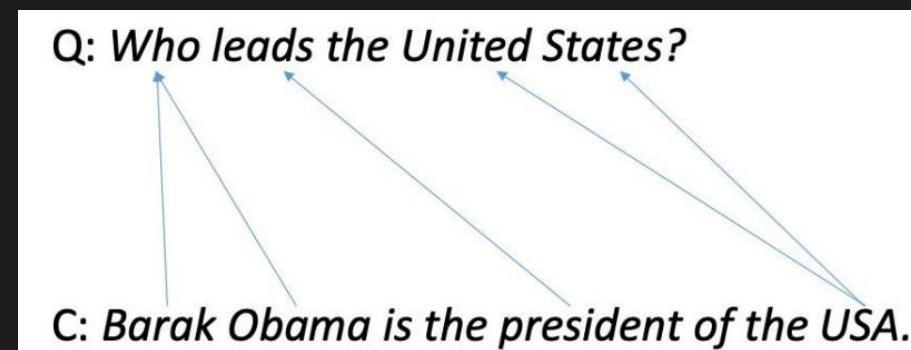
$$q_i = [\vec{q}_i; \hat{q}_i] \in \mathbb{R}^{2H}$$

# Question Answering

- BiDAF: Attention

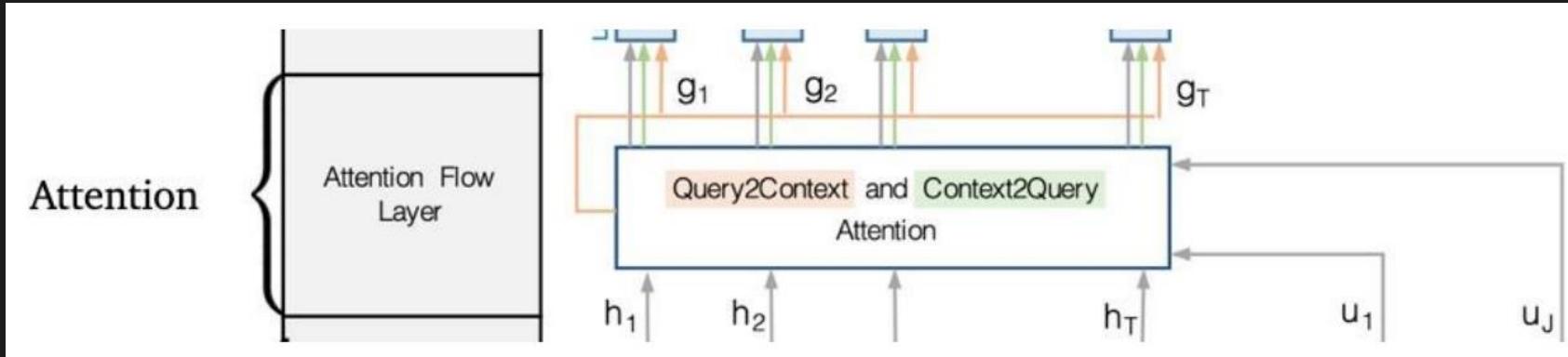


- Context-to-query attention
  - 각 context word에 대해, 해당 word와 가장 관련 있는 query word 구하기



# Question Answering

- BiDAF: Attention



- Query-to-context attention
  - 주요 query word에 대해, 해당 word와 가장 관련 있는 context word 구하기

While **Seattle's** weather is very nice in summer, its weather is very rainy  
in winter, making it one of the most **gloomy cities** in the U.S. LA is ...

*Q: Which city is gloomy in winter?*

# Question Answering

- BiDAF: Attention

- 이 과정을 식으로 더 자세히 알아보자!
- 우선, 모든  $(c_i, q_i)$ 에 대해 **similarity score**를 구한다.

$$S_{i,j} = w_{sim}^\top [c_i; q_j; c_i \odot q_j] \in \mathbb{R}, \quad w_{sim} \in \mathbb{R}^{6H}$$

- 해당 similarity score에 대해 **Context-to-query attention**을 수행한다.

$$\alpha_{i,j} = \text{softmax}_j(S_{i,j}) \in \mathbb{R}, \quad a_i = \sum_{j=1}^M \alpha_{i,j} q_j \in \mathbb{R}^{2H}$$

- 똑같이 S에 대해 **Query-to-context attention**을 수행한다.

$$\beta_i = \text{softmax}_i(\max_{j=1}^M (S_{i,j})) \in \mathbb{R}^N, \quad b = \sum_{i=1}^N \beta_i c_i \in \mathbb{R}^{2H}$$

- 최종 결과:  $g_i = [c_i; a_i; c_i \odot a_i; c_i \odot b] \in \mathbb{R}^{8H}$

# Question Answering

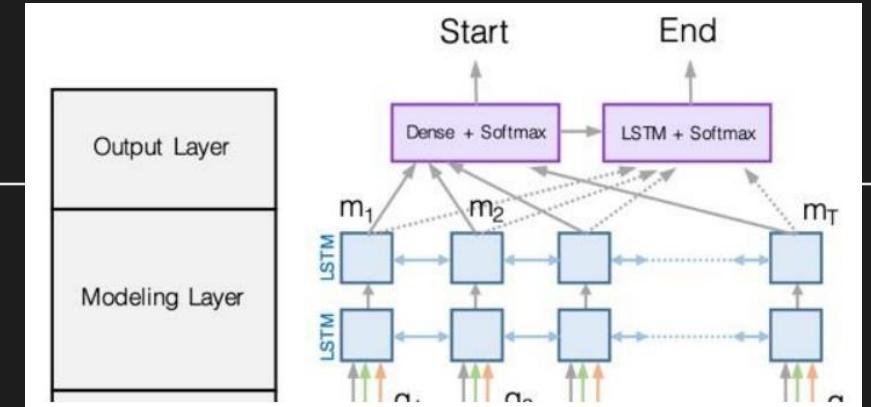
- BiDAF: Modeling and output layers

- Modeling layer

- 아까 전에 나온  $g_i$ 를 2-layers bi-directional LSTM에 넣는다.

→ context word 사이 상호작용 효과

$$m_i = BiLSTM(g_i) \in \mathbb{R}^{2H}, \quad m'_i = BiLSTM(m_i) \in \mathbb{R}^{2H}$$



- Output layer

- Start와 end token을 prediction하는 두 classifier( $w$ ) 도입

$$p_{start} = \text{softmax}(w_{start}^\top [g_i; m'_i]), \quad p_{end} = \text{softmax}(w_{end}^\top [g_i; m'_i])$$
$$w_{start}, w_{end} \in \mathbb{R}^{10H}$$

- 최종 training loss:  $\mathcal{L} = -\log p_{start}(s^*) - \log p_{end}(e^*)$

# Question Answering

- BiDAF
  - Performance: **77.3 F1 score**
  - W/o context-to-query attention 67.7 F1
  - W/o query-to-context attention 73.7 F1
  - W/o character embeddings 75.4 F1

	Published <sup>[12]</sup>	LeaderBoard <sup>[1]</sup>
Single Model	EM / F1	EM / F1
LR Baseline (Rajpurkar et al., 2016)	40.4 / 51.0	40.4 / 51.0
Dynamic Chunk Reader (Yu et al., 2016)	62.5 / 71.0	62.5 / 71.0
Match-LSTM with Ans-Ptr (Wang & Jiang, 2016)	64.7 / 73.7	64.7 / 73.7
Multi-Perspective Matching (Wang et al., 2016)	65.5 / 75.1	70.4 / 78.8
Dynamic Coattention Networks (Xiong et al., 2016)	66.2 / 75.9	66.2 / 75.9
FastQA (Weissenborn et al., 2017)	68.4 / 77.1	68.4 / 77.1
BiDAF (Seo et al., 2016)	68.0 / 77.3	68.0 / 77.3
SEDT (Liu et al., 2017a)	68.1 / 77.5	68.5 / 78.0
RaSoR (Lee et al., 2016)	70.8 / 78.7	69.6 / 77.7
FastQAExt (Weissenborn et al., 2017)	70.8 / 78.9	70.8 / 78.9
ReasoNet (Shen et al., 2017b)	69.1 / 78.9	70.6 / 79.4
Document Reader (Chen et al., 2017)	70.0 / 79.0	70.7 / 79.4
Ruminating Reader (Gong & Bowman, 2017)	70.6 / 79.5	70.6 / 79.5
jNet (Zhang et al., 2017)	70.6 / 79.8	70.6 / 79.8
Conductor-net	N/A	72.6 / 81.4
Interactive AoA Reader (Cui et al., 2017)	N/A	73.6 / 81.9
Reg-RaSoR	N/A	75.8 / 83.3
DCN+	N/A	74.9 / 82.8
AIR-FusionNet	N/A	76.0 / 83.9
R-Net (Wang et al., 2017)	72.3 / 80.7	76.5 / 84.3
BiDAF + Self Attention + ELMo	N/A	<b>77.9 / 85.3</b>
Reinforced Mnemonic Reader (Hu et al., 2017)	73.2 / 81.8	73.2 / 81.8

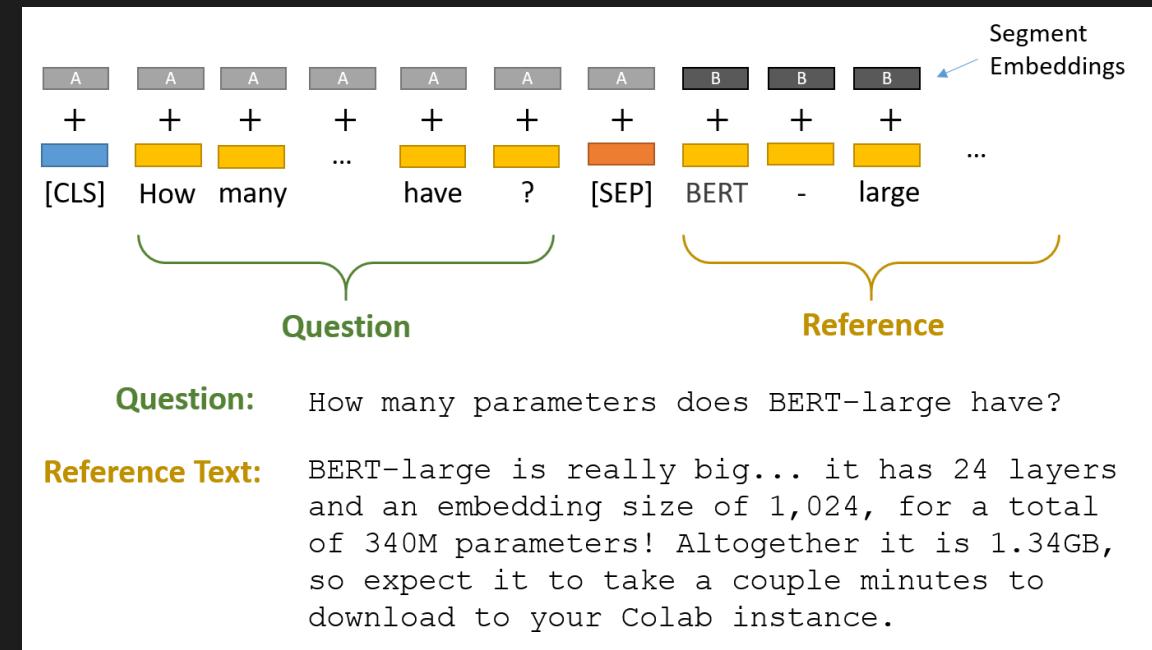
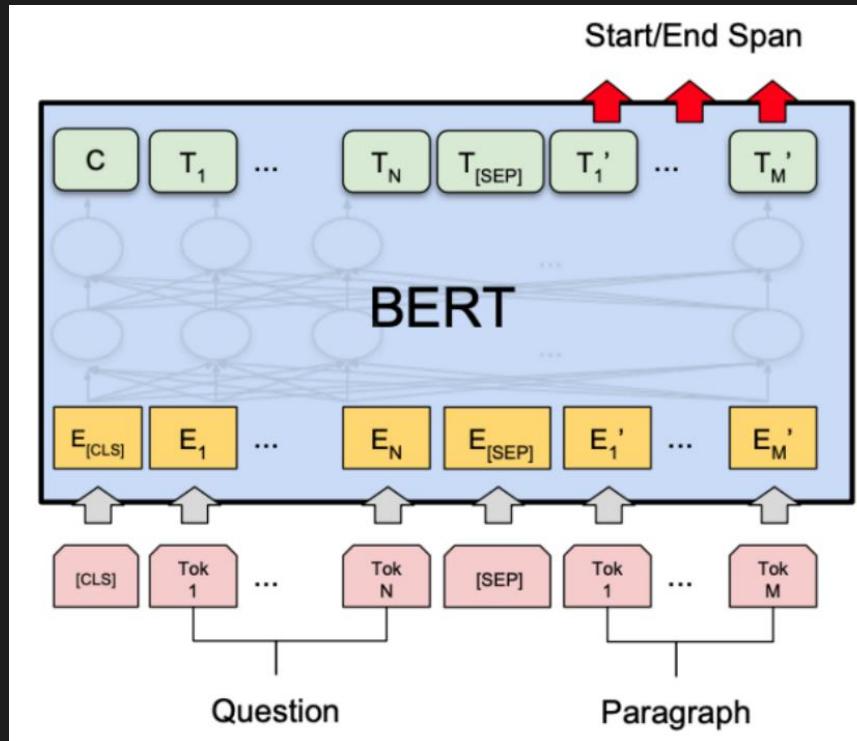
# Question Answering

- BiDAF
  - performance

	Published <sup>[12]</sup>	LeaderBoard <sup>[1]</sup>
Single Model	EM / F1	EM / F1
LR Baseline (Rajpurkar et al., 2016)	40.4 / 51.0	40.4 / 51.0
Dynamic Chunk Reader (Yu et al., 2016)	62.5 / 71.0	62.5 / 71.0
Match-LSTM with Ans-Ptr (Wang & Jiang, 2016)	64.7 / 73.7	64.7 / 73.7
Multi-Perspective Matching (Wang et al., 2016)	65.5 / 75.1	70.4 / 78.8
Dynamic Coattention Networks (Xiong et al., 2016)	66.2 / 75.9	66.2 / 75.9
FastQA (Weissenborn et al., 2017)	68.4 / 77.1	68.4 / 77.1
BiDAF (Seo et al., 2016)	68.0 / 77.3	68.0 / 77.3
SEDT (Liu et al., 2017a)	68.1 / 77.5	68.5 / 78.0
RaSoR (Lee et al., 2016)	70.8 / 78.7	69.6 / 77.7
FastQAExt (Weissenborn et al., 2017)	70.8 / 78.9	70.8 / 78.9
ReasoNet (Shen et al., 2017b)	69.1 / 78.9	70.6 / 79.4
Document Reader (Chen et al., 2017)	70.0 / 79.0	70.7 / 79.4
Ruminating Reader (Gong & Bowman, 2017)	70.6 / 79.5	70.6 / 79.5
jNet (Zhang et al., 2017)	70.6 / 79.8	70.6 / 79.8
Conductor-net	N/A	72.6 / 81.4
Interactive AoA Reader (Cui et al., 2017)	N/A	73.6 / 81.9
Reg-RaSoR	N/A	75.8 / 83.3
DCN+	N/A	74.9 / 82.8
AIR-FusionNet	N/A	76.0 / 83.9
R-Net (Wang et al., 2017)	72.3 / 80.7	76.5 / 84.3
BiDAF + Self Attention + ELMo	N/A	<b>77.9 / 85.3</b>
Reinforced Mnemonic Reader (Hu et al., 2017)	73.2 / 81.8	73.2 / 81.8

# Question Answering

- BERT for reading comprehension



- 자세한 내용은 프로그래밍 과제 4에서..!

# Question Answering

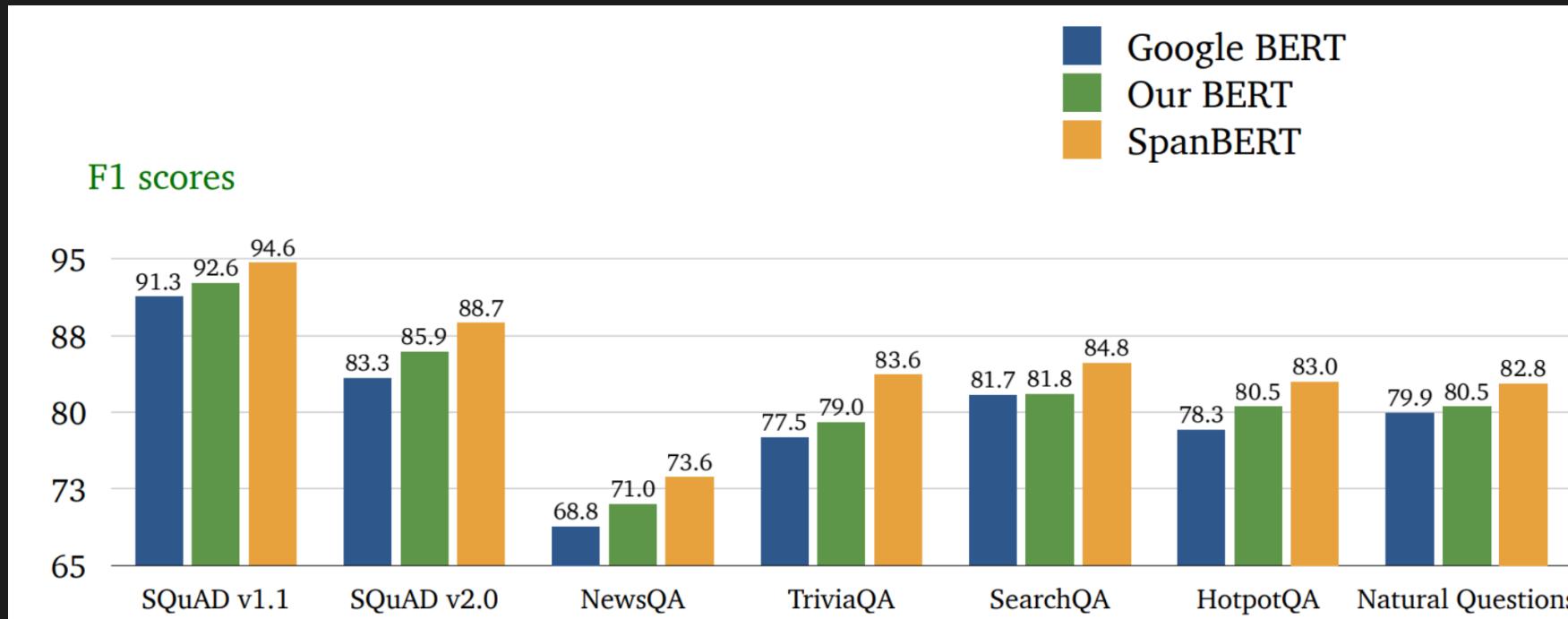
- BERT for reading comprehension
  - evaluation

	F1	EM
Human performance	91.2*	82.3*
BiDAF	77.3	67.7
BERT-base	88.5	80.8
BERT-large	90.9	84.1
XLNet	94.5	89.0
RoBERTa	94.6	88.9
ALBERT	94.8	89.3

# Question Answering

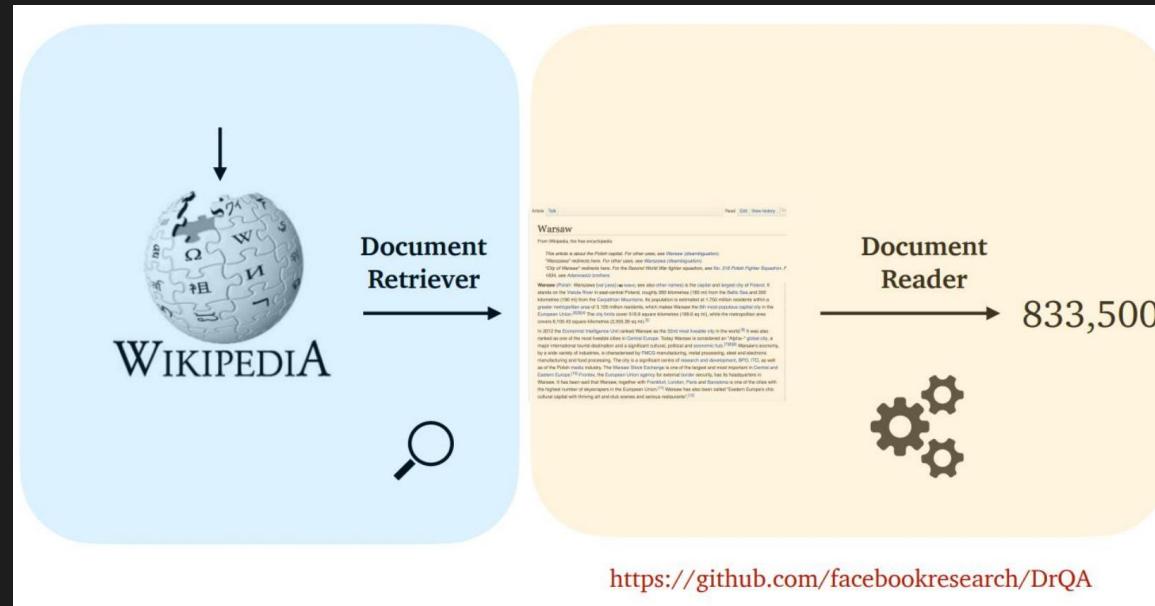
- BERT for reading comprehension

- BERT의 pre-training 방법을 조금만 바꾸어도 QA task에서 성능을 더 올릴 수 있다고 합니다.
- QA task for BERT의 특성 상 **span**을 예측하는 task를 pretraining task로 했을 때 더 성능이 잘 나왔다고 해요. (SpanBERT 기억나시죠?)



# Question Answering

- Open-domain question answering
    - 기존 QA task와 동일. 단 **passage**가 주어지지 않음!
    - 대신 Wikipedia 같은 거대 document 저장소에 접근 가능! 하지만 정답이 있는 문서가 무엇인지는 몰라요…
- **Retrieval** 필요!



# Question Answering

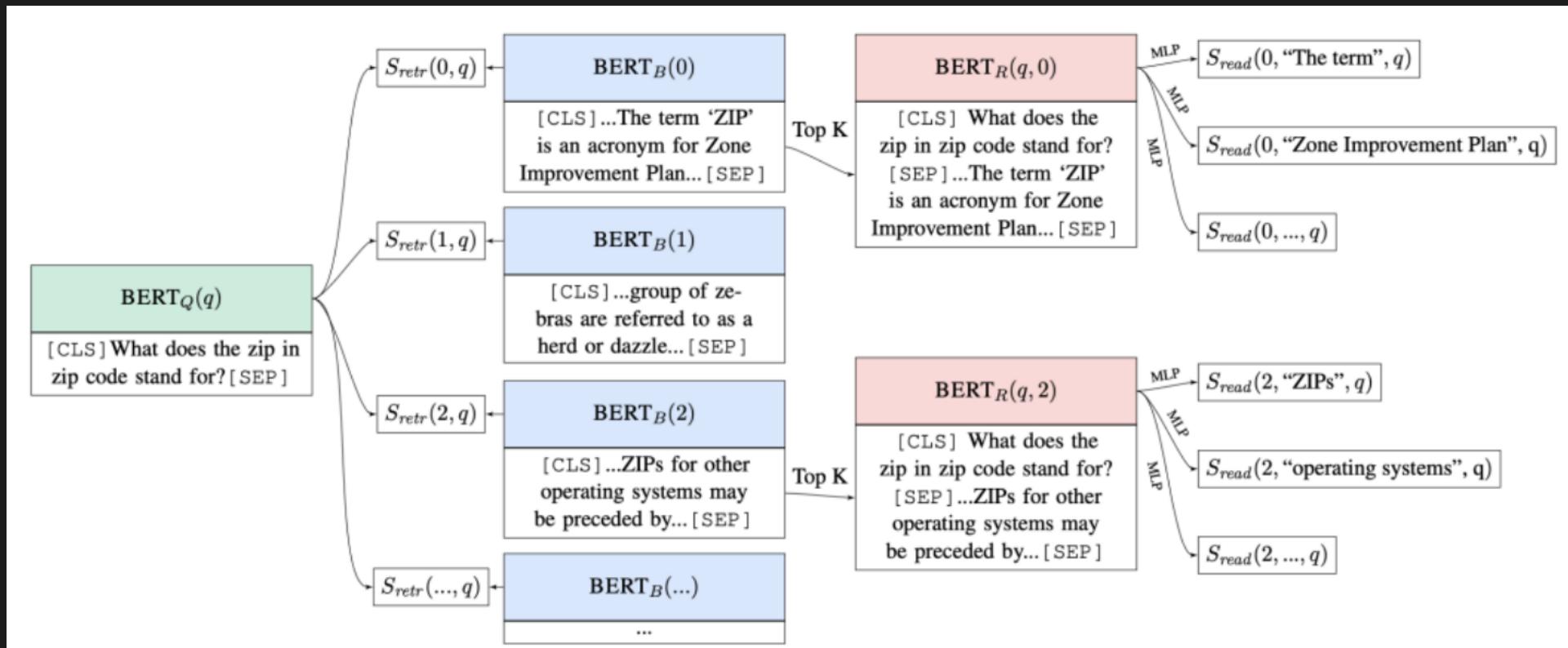
- Retriever-reader framework

- Input: a large collection of documents  $\mathcal{D} = D_1, D_2, \dots, D_N$  and  $Q$
- Output: an answer string  $A$
- Retriever:  $f(\mathcal{D}, Q) \rightarrow P_1, \dots, P_K$       K is pre-defined (e.g., 100)
- Reader:  $g(Q, \{P_1, \dots, P_K\}) \rightarrow A$       A reading comprehension problem!

- DrQA라는 모델에서는
  - Retriever로 standard TF-IDF 모델 사용
  - Reader로 Neural Network 모델 사용

# Question Answering

- Retriever-reader framework
  - Latent Retrieval for Weakly Supervised Open Domain Question Answering (2019)
    - End-to-end model

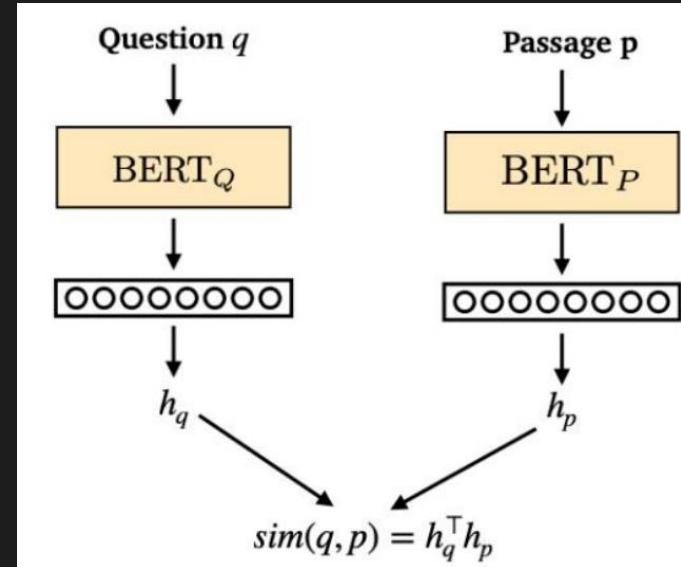


# Question Answering

- Retriever-reader framework
  - Latent Retrieval for Weakly Supervised Open Domain Question Answering (2019)
    - End-to-end model
    - BERT를 이용해 question과 passage의 embedding을 구하고, dot product를 이용해 similarity score를 구함 → ranking을 매겨 상위 K개의 passage를 input으로 넣어 QA 수행
    - Passage의 개수가 많으므로 쉽지 않음…

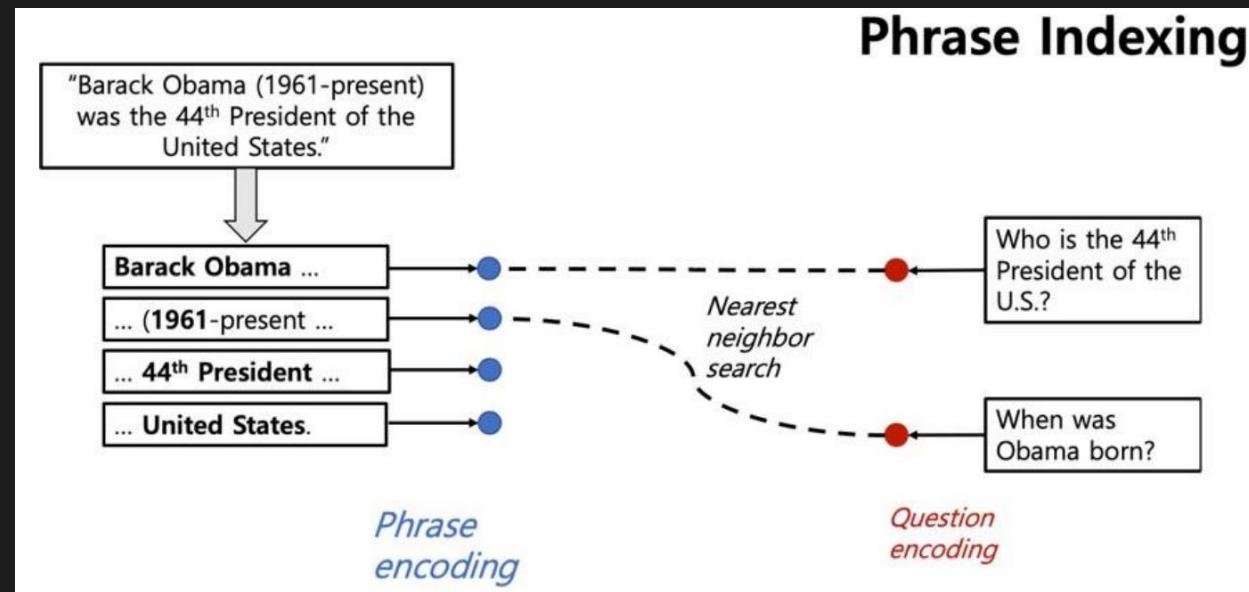
# Question Answering

- Retriever-reader framework
  - Dense passage retrieval (DPR) (2020)
    - Question, passage pair data를 이용해 retrieval을 finetuning
    - → 기존 dense retrieval의 단점 보완



# Question Answering

- Retriever-reader framework
  - Learning Dense Representations of Phrases at Scale (2020)
    - 모든 phrase와 question을 dense vector로 encode한 후 nearest neighbor search로 알맞은 쌍을 찾음
    - 매우 빠름. CPU로도 동작



# Summary

# Summary

---

- 저희는 오늘
  - NN 이전 통계학적 방법론인
    - BoW와 TF-IDF에 대해 살펴보았습니다.
    - Topic modeling에 대해서도 살펴보았습니다.
  - Tagging task과
  - Coreference Recognition에 대해서도 살펴보았습니다.
  - Retrieval 중
    - Sparse retrieval
    - Dense retrieval에 대해 살펴보았습니다.
  - Question answering task에 대해 살펴보고, 대표적인 모델
    - BiDAF와
    - BERT (finetuned for QA) 에 대해서도 살펴보았습니다.

# Discussion (~ 이번 주 목요일 수업 전까지)

- Q1. 수업 내용에서, “모든 task는 *Reading comprehension*으로 환원된다.”고 하였습니다. 그렇다면, 어떤 downstream task를 수행할 때 그에 맞게 fine-tuning 된 모델을 사용하는 것과 해당 task를 reading comprehension으로 치환하여 푸는 경우 중 어떤 것이 더 나은 성능을 보일까요? 의견을 자유롭게 적어주세요!
- Q2. Question Answering 파트에서는 LSTM 기반 모델인 BiDAF와 transformer 기반 모델인 BERT를 배웠습니다. 이 두 가지의 공통점과 차이점을 생각해 보고, 왜 BERT 모델이 성능이 더 좋게 나왔는지에 대해 자신의 의견을 써 주세요!
  - HINT: BiDAF는 Context-Query attention, Query-Context attention을 사용하였습니다. 그럼 BERT는…?

# 프로그래밍 과제

(~8월 20일까지)

- 놀랍게도,, 프로그래밍 과제는 아직도 제작 중입니다. (그래도 90% 이상 완성됨)
- 오늘 안에 AIKU 공지방에 과제 링크를 공지하겠습니다 😊
- 늦어져서 죄송합니다 ㅠㅠ

# 참고자료

## 자료 출처

- Standford Univ. cs224n
- 고려대 이병준 교수님 강의자료
- [위키독스 딥 러닝을 이용한 자연어처리 입문](#)
- [POS tagging survey paper](#)
- [개체명 인식](#)
- [Retrieval 정리 자료](#)

## 추가 자료

- [LDA 논문](#)

본 PPT는 고려대학교 딥러닝학회 AIKU의 정기세미나 및 기타 활동 내용을 바탕으로 하고 있습니다.

무단 도용 및 활용을 금합니다.

관련한 문의는 @aiku.\_.official로 DM부탁드립니다.

감사합니다.