# Lecture 10. (Textual) Question Answering

## Ⅰ. Motivation & History

정보 검색을 하면 정답을 포함하고 있는 문서를 찾아주거나 문서에서 정답을 찾아준다. 문서에서 정답을 찾아주는 것을 MRC(Machine Reading Compehension)이라고 한다.

## Ⅱ. SQuAD(Stanford Question Answering Dataset)

1. 질문의 종류

   1) Factoid type questions (what, which, when, who, how) => SQuAD

      Ex) 한국의 수도는?

   2) List type questions

      Ex) 방탄소년단 멤버에는 누가 있는가?

   3) Causual questions (why, how)

      Ex) 왜 늦었니?

   4) Hypothetical Questions

      Ex) 남한과 북한이 전쟁이 나면 어떻게 될까?

   5) Complex Questions

      Ex) 세계2차대전의 원인은 무엇일까?

2. SQuAD(1.1)

   QA dataset은 Passage(P)와 Question(Q), Answer(A)로 이루어져 있다.


   Answer은 항상 Passage 속 하위 sequence로 구성되어있다. SQuAD dataset은 위키피디아 문서를 바탕으로 크라우드 소싱 인력들이 해당하는 질문과 답변을 생성한 것이고 질문에 대해 3가지 답변을 sampling한다. SQuAD dataset을 이용한 평가 방법으로는 exact match 방법과 f1 score를 이용하는 방법이 있다.

1) exact match : 예측한 답과 실제 답이 일치하면 1, 아니면 0

2) f1 score : 예측된 답과 실제 답의 중첩토큰을 계산. 더 안정적인 평가방법.

3. SQuAD(2.0)

1.1에서 2.0으로 dataset이 발전하면서 added unanswerable question이 생겼다. 즉, 답이 없는 질문들이 추가되었다. 현존하는 reading comprehension dataset들은 주로 answerable한 질문들에 초점을 맞추거나 쉽게 판별이 가능한 가짜 unanswerable(대답 불가능한) 질문들을 자동 생성해왔다. .SQuAD 2.0에서는 이런 약점을 보완하게 된다.

- 기존 dataset(SQuAD 1.1)에 새로운 5만 개 이상의 unanswerable questions(응답 불가능한 질문)를 병합

- unanswerable question은 온라인의 crowd worker들이 직접 생성(즉, 기계적으로 생성된 것이 아니라 진짜 인간이 생성했으므로 질이 더 높음)

- worker들에 의해 생성된 unanswerable question은 응답 가능한 질문들과 유사하여 기계적으로 판별이 어려움.

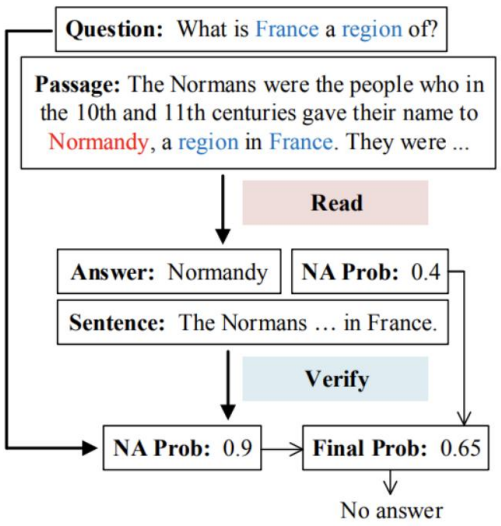
4. No answer 모델을 처리하는 방법

1) 임계값을 사용한다.

임계값 이상의 결과에 대해서만 answer을 예측하고 임계값 이하의 결과에 대해서는 no answer로 예측

2) Machine Reading Comprehension with Unanswerable Questions(2019 AAAI)

Read then verify 시스템을 제안 -> Passage에 대한 question에 대한 정답과 no answer 확률을 각각 도출함 -> 이렇게 도출한 정답이 적절한지 확인한다.



5. SQuAD Limitations

1) Span-based answers만 존재

2) Passage 내에서만 정답을 찾도록 하는 질문 구성. 여러 문서들을 비교하여 진짜 정답을 찾아낼 필요는 없음. 실제 마주하게 될 질문(답변 보다 잘 정립된 문법구조)

3) 동일 지시어 문제를 제외하고는 multi-fact 문제, 문장 추론문제가 거의 없음.

그럼에도 불구하고 SQuAD는 well-targeted, well-structed, clean dataset이다.

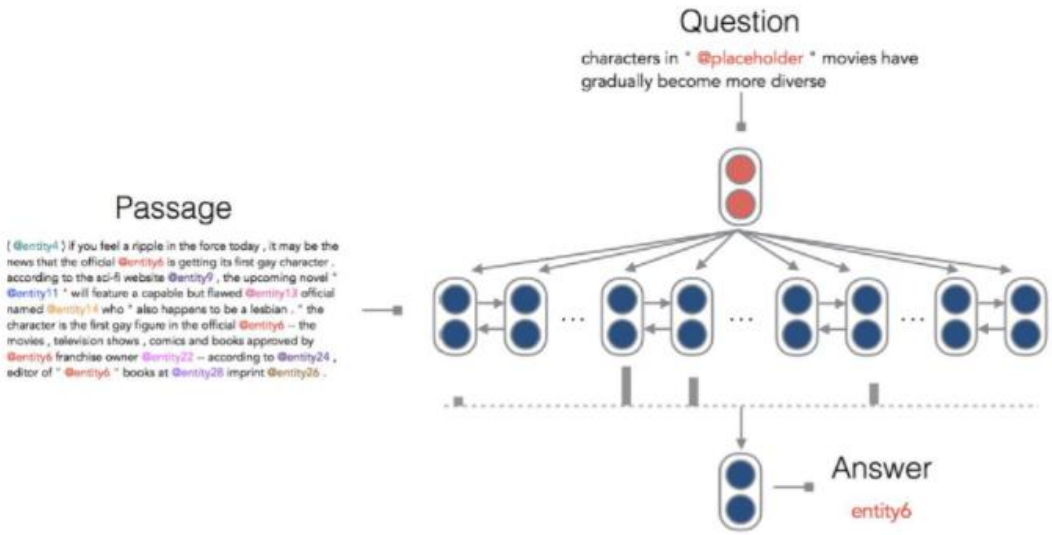
- QA 문제를 푸는데 있어 가장 많이 사용되고 경쟁하는 dataset
- 실제 시스템을 개발하기 위한 유용한 start point

KorQuAD(2.0) : 한국어 위키백과를 대상으로 대규모 MRC 데이터를 구축한 것

## Ⅲ. QA model

1. Stanford Attentive Reader

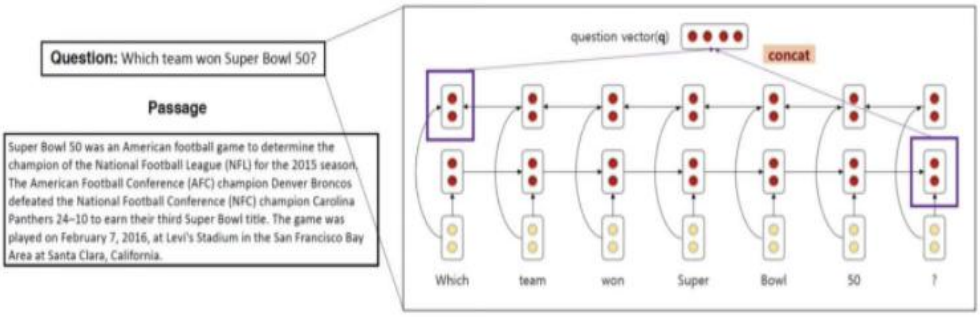
질의에 대한 응답을 찾는 모델을 구축하는데 Bi LSTM with attention을 적용함.




Stanford attentive model은 아래와 같이 세 부분으로 나누어져 있다.

1) Question vector 생성

주어진 문장에 있는 단어들의 embedding을 사전에 학습된 glove에서 가져와 one-layer Bidirectional LSTM에 넣는다. Bidirectional LSTM의 각 방향 마지막 hidden state를 concat하여 question vector를 얻는다.




2) Passage vector 생성

주어진 문장에 있는 단어들의 embedding을 사전에 학습된 glove에서 가져와 one-layer bidirectional LSTM에 넣는다. 그리고 각각의 포지션에서 bidirectional LSTM의 hidden state를 concat하여 문장 내 단어 개수 만큼의 passage vector를 생성한다.

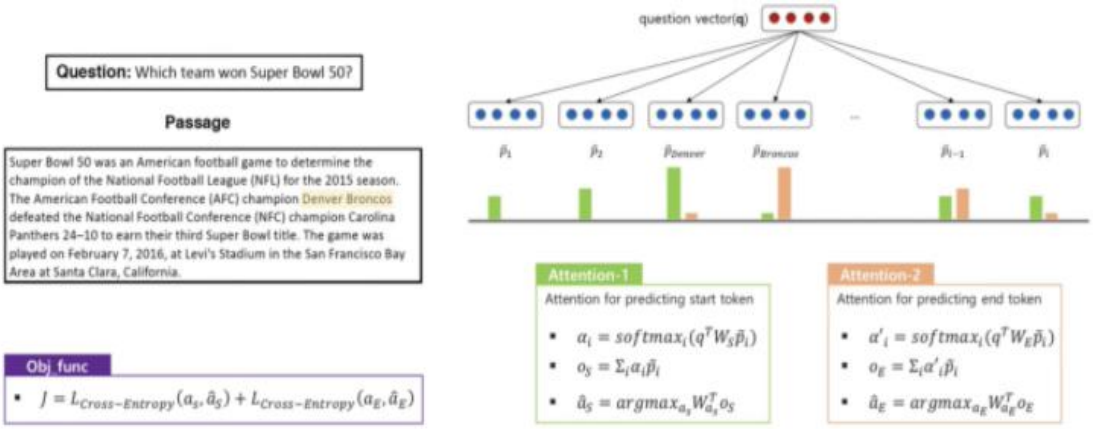
3) Attention

-$a_i$ : i개의 p벡터와 한 개의 q벡터를 이용하여 attention을 적용한 후 softmax를 취함.

-$o_s$ : $a_i$와 $p_i$ 벡터를 곱하여 모두 더함.

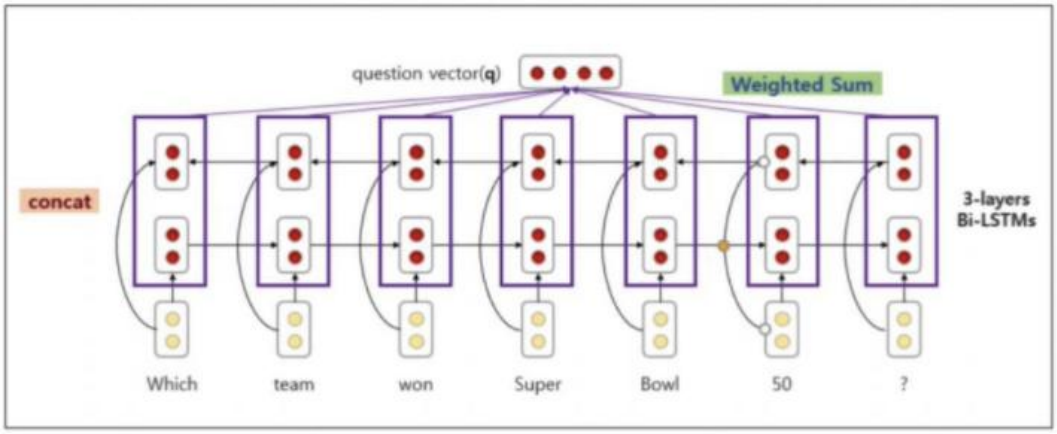
-$a_s$ : $o_s$에 linear transform을 취함.

=> 이렇게 start token과 end token을 구함.

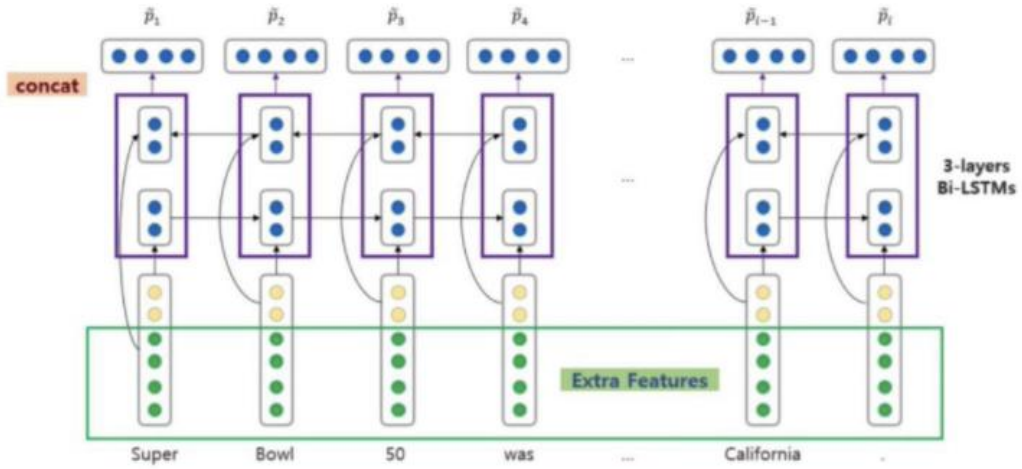



2. Stanford Attentive Reader++

Stanford attentive reader과 차이점을 살펴보면, Stanford Attentive Reader++에서는 one layer bidirectional LSTM이 아닌 3 layer bidirectional LSTM을 사용하게 되었다. 또한 question vector를 구성할 때, 각 방향 마지막 hidden state를 concat이 아닌, bidirectional LSTM state를 포지션별로 concat한 후 weighted sum을 하여 구성한다.




Passage vector를 구성할 때도 단어의 embedding 값만 사용하지 않고 extra feature들을 더 추가하여 passage vector를 구성한다. extra feature들에는 word embedding(GloVe 300d), linguistic features(POS&NER tags, one-hot encoded), Term frequency(unigram probability), exact match(질문에 단어가 등장하는지, 3 binary features(exact, uncased, lemma)), Aligned question embedding이 있다.




3. BiDAF : Bi-Directional Attention Flow for Machine Comprehensive

   Question에서 Passage로 한 방향으로만 진행되는 Stanford Attentive Reader와 달리 attention이 양방향으로 적용된 모델. (Question <-> Passage) 이 모델은 총 6단계로 이루어져 있다.

   1. Character Embedding Layer : charCNN을 사용하여 각 단어를 vector space에 mapping.
   2. Word Embedding Layer : pre-trained word embedding model을 사용하여 각 단어를 vector space에 mapping
   3. Contextual Embedding Layer : 위의 두 결과를 concat하고 bidirectional LSTM에 넣기 전에 two-layer highway-network를 거쳐 d-dimension vector로 만들어준다.

4. Attention Flow Layer : Query to Context(Context의 어떤 정보가 Query와 관련이 있는지), Context to Query(Query의 어떤 정보가 Context와 관련이 있는지) 이렇게 양방향으로 attention이 일어남. 즉, Query와 Context를 쌍으로 묶어서 Attention을 학습하는 곳이다. Query와 context(=passage)를 single feature vector로 요약하지 않고 query와 context를 연결시킨다. 양방향 attention을 위해 share matrix S를 이용한다.

$Stj = \alpha(H:t, U:j)$ -> similarity matrix

→ $H:t$= context의 t번째 word vector, $U:j$ = query의 j번째 word vector

→ $\alpha$ = similarity를 encoding하는 역할

$Stj$ 는 t-th context word와 j-th query word의 similarity 를 의미

-Context2Query Attention : $S_t$에 대해 softmax를 취하고 이렇게 나온 $a_t$와 U를 이용해 t번째 context word 입장에서 유사성이 높은 query의 특성을 알게 된다.

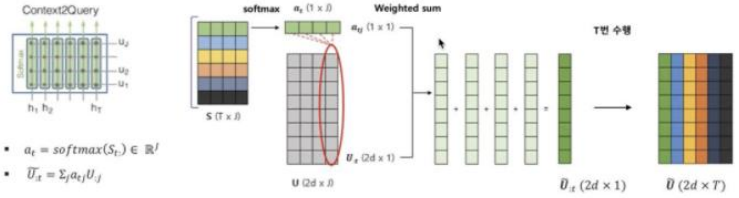
$a_t = softmax(St:) \in RJ$

→ $St:$ = t-th context word와 query word들의 similarity를 의미

→ similarity $Stj$에 $softmax$를 취해줬으므로 유사성이 큰 weight만 값이 커지게 됨

$\widetilde{U_{:t}} = \sum a_{tj} U_{:j}$ (j-th query word에 t-th context와의 attention weight를 곱함) $H:t$= context의 t번째 word vector, $U:j$ = query의 j번째 word vector

t-th context word 입장에서 유사성이 높은 query의 특성을 알게 됨




-Query2Context Attention : S에서 maxcol을 통해 가장 큰 값을 뽑아 softmax를 취하고 이렇게 나온 b와 H를 이용하여 query 입장에서 중요한 word를 알게 된다.

b = $softmax(maxcol(S))$ $H:t$= context의 t번째 word vector, $U:j$ = query의 j번째 word vector

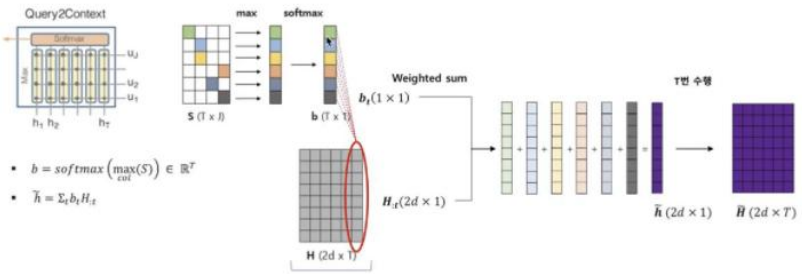
→ $maxcol$ : 각 row에서 가장 큰 값을 뽑아냄

→ 각 context word마다 가장 query words중에서 similarity가 큰 항만 뽑히게 됨.

→ 여기에 softmax를 취해주므로 상대적으로 더 query와 상관 있는 context word만 남게 된다.

$\tilde{h} = \sum b_t H_{:t}$ (weight bt를 곱해서 값이 더해지면, query와 similarity가 큰 t-th context word만 값이 더해짐)

similarity가 큰 context word만 살아남게 되므로 query 입장에서 중요한 word만 살아남게 됨



5. Modeling Layer : 앞서 구한 G(H와 U를 이용하여 구함)를 2-layer bidirectional LSTM을 거쳐 M을 만들고 M과 G를 concat하여 linear transform과 softmax를 취해 start, end token을 예측한다.

6. Output Layer : 최종적으로 sum of negative log probabilities의 average를 최소화하는 방향으로 학습이 이루어진다.