<https://jeongukjae.github.io/posts/cs224n-lecture-17-the-natural-language-decathlon/>

<https://drive.google.com/file/d/19fLxjW5OurVs8NuBiagW0EOexiAiMNzL/view>

single task learning의 한계

1. 대개 pre-train 없이 from scratch(처음부터) 학습이 진행되며 각 task에 대한 많은 데이터셋을 필요로 한다.
2. Local optima를 넘기 위해서는 잘 정돈된 데이터셋 > 1000xC가 필요하다.

따라서 핵심인 pre-training이 Word2Vec, GloVe, CoVe, ELMo, BERT 등에서 성공을 이루었다.

그러면 왜 pre-training을 안했었나?

1. 영상과 다르게 언어는 여러 특성을 포함한다. Logical, linguistic, emotional, visual ++ 따라서 pre-training을 막 쓰면은 그 특성을 캐치 못할 수도 있다..?(도메인이 너무 다른 느낌인가?)
2. 또한 이전 단어에 대한 short & long term 을 잘 기억하면서 task를 해결해야 한다. 이 부분 또한 pre-training으로 캐치하기 힘든가?
3. NLP는 중간 or 분리된 task로 많이 나누어져 있다. (각각에 따른 벤치마크가 필요하다.)
4. 하나의 비지도 학습 task가 전체 문제를 해결할 수도 없다.
5. 언어는 현실적으로 분명한 supervision이 필요

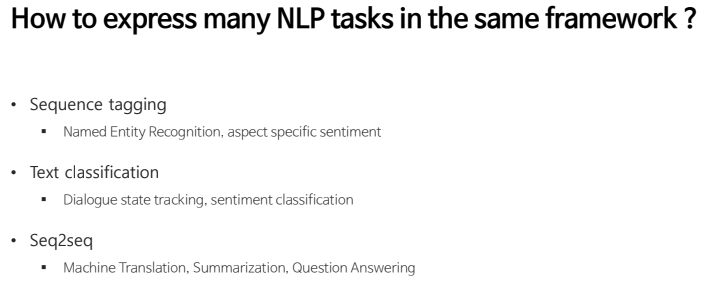
그러면 pre-training을 효과적으로 하기 위해 multi-task에 대해 pre-training 하면 이 문제들을 해결할 수 있지 않을까?

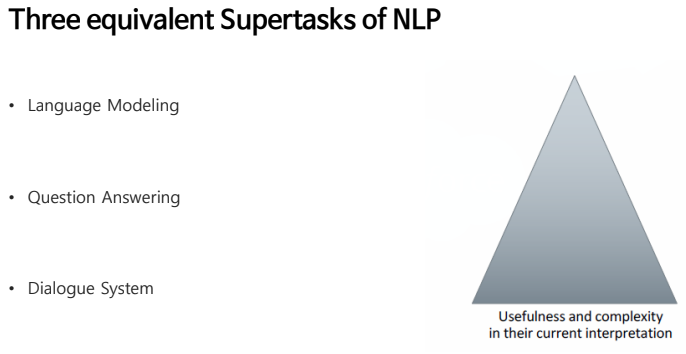
1. 하나의 통합된 모델은 지식을 어떻게 전달할지 결정할 수 있다.(Domain adaptation, Weight sharing, Transfer and zero shot learning)
2. 새로운 task가 주어졌을 때 쉽게 적응 가능
3. 실제 production을 위해 deploy 하는 것 매우 간단해짐.
4. 잠재적으로 continual learning 나아갈 수 있다.

아래처럼 여러가지 NLP Tasks가 있다.

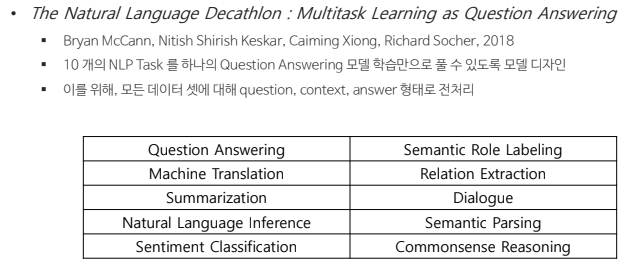
이 Task들을 어떤 방식으로 한번에 통합해서 pre-train 할까? 대개 아래처럼 3가지가 존재한다.

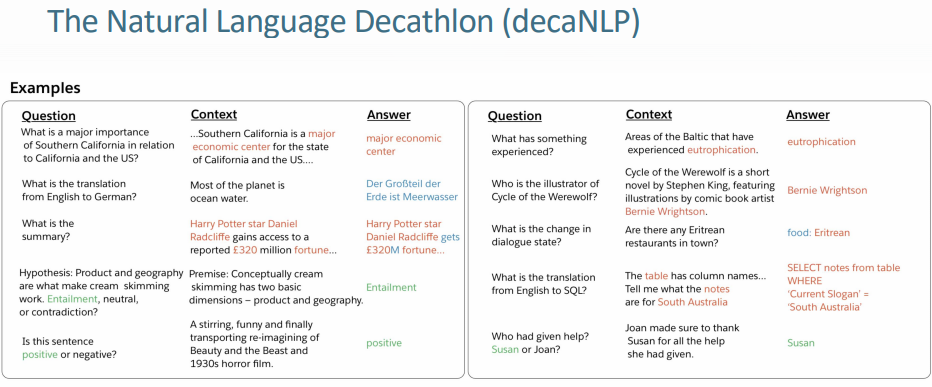
(BERT는 LM 을 썼다.)

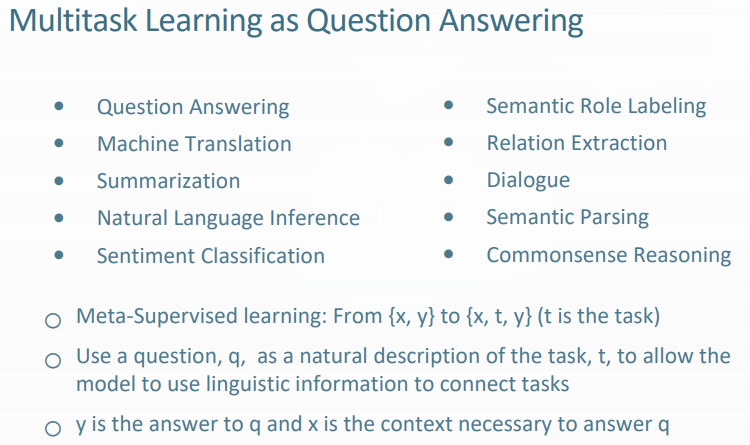




따라서 McCann(2018)은 10개의 NP Task를 하나의 QA 데이터셋으로 바꾸어(question, context, answer) 하나의 QA 모델 디자인 설계를 하여 pre-train 하였다. (아래처럼)





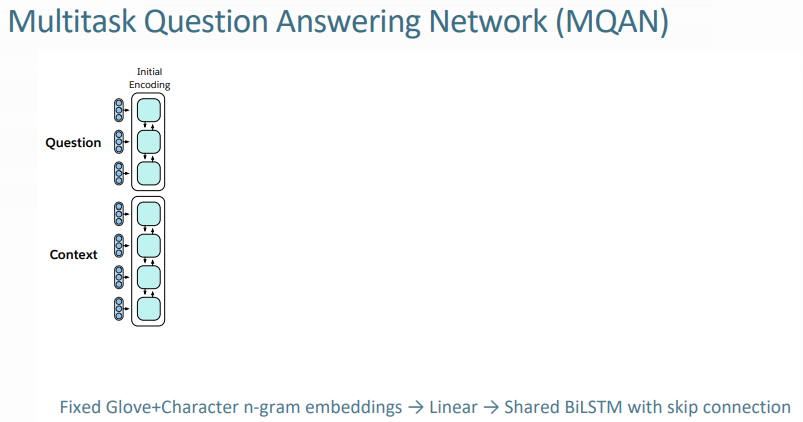


t, x, y는 각각 question, context, answer로 보면 되고 이런 형태로 10가지 task에 대한 데이터셋을 구성한다는 것이다.

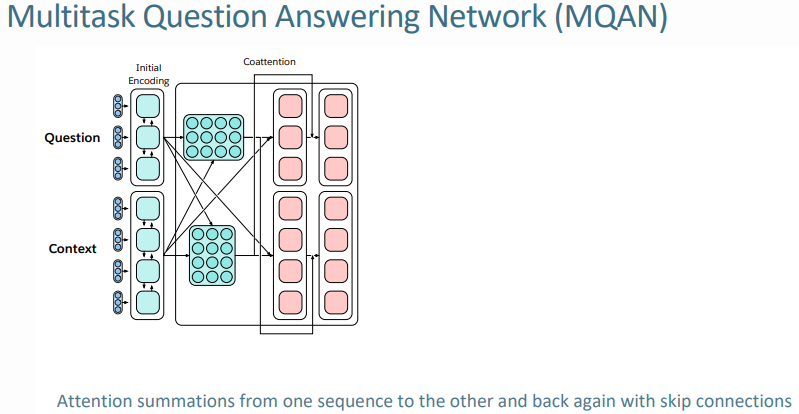
decaNLP에서는 no task-specific modules, adjust to perform disparate tasks, open the zero-shot inference for unseen tasks 의 특성을 가진다.

대략의 플로우는 다음과 같다.

1. Start with a context & Ask a question
2. Generate the answer one word at a time  
   \* pointing to context  
   \* pointing to question  
   \* choosing a word from an external vocab
3. Pointer switch is choosing between 3 options(가중치 게이트) for each output word



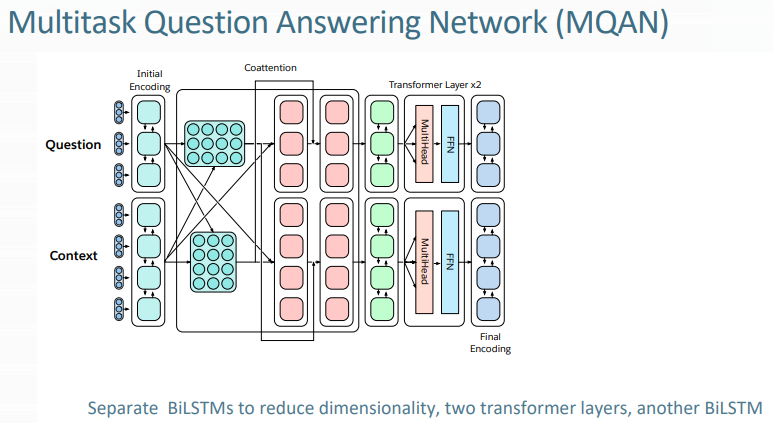
처음에는 GloVe + Character n-gram 으로 임베딩 했지만 후에 CoVe -> BERT 임베딩 까지 하였다.



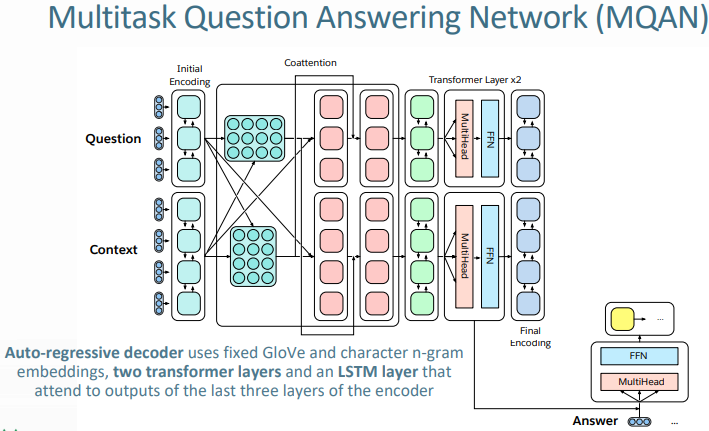
Q와 C에서 서로에 대해 Query를 바꿔가며 Co-Attention을 하는듯 싶다.

그 후에 아래 그림처럼 dimensionality를 줄이기 위해 양 끝에 BiLSTMs를 이용하면서 중간에 Transformer도 이용한다.

2개의 인코더 layers를 이용하는듯?



아래처럼 이제 encoding 정보를 decoder(트랜스포머 인코더 2개인가?와 1 LSTM 디코더)에 전달해 준다. (처음은 <Start> 토큰부터 시작하나??)그리고 1개의 결과가 나올 때마다 어텐션이 계산된다.



아래처럼 Q & C에 대한 어텐션 pointer를 구하고 Q or C 어디에서 아웃풋을 낼지에 대해 반영하는지에 대한 lambda, 그리고 외부 vocab에서 아웃풋을 낼지에 대한 gamma를 배우게 되면서 최종 아웃풋을 내게 된다.

(어텐션은 Query(현재 상태의 state 한 개)와 Key(어떤 tokens 전체)가 기본이라 생각하자)

