딥러닝 기반 자연어처리 기법의 최근 연구 동향 정리 1

ratsgo's blog 글을 정리하였습니다.

1. 서론

NLP 는 인간 언어 분석과 표현을 자동화하기 위한 계산 기법.

컴퓨터로 하여금 파싱, 품사태깅에서부터 기계번역, 대화시스템에 이르기까지 모든 과업 수행 가능.

머신러닝에서의 접근: 고차원이면서 sparse 한 feature 를 학습한 얕은 모델

최근 수 년간 dense vector rerpesenation 에 기반한 뉴런 네트워크가 우수한 성능 보임

딥러닝은 자동화된 피처 추출 및 표현 가능한데 비해 머신러닝 기반 nlp 는 사람이 추출한 피처에 강하게 의존하여 시간 많이 소요되고 불완전함

간단한 딥러닝 프레임워크로는 개체명 인식(NER), 의미역결정(SRL), 품사태깅(POS tagging)

복잡한 딥러닝 기반 알고리즘: CNN, RNN, Recursive Neural Network 등이 있음

Attention, reinforcement learning, deep generative model 등

2. 분산표상 (distributed representation)

언어모델은 결합확률 함수를 학습해야 했기에 통계 기반 자연어처리 기법은 차원의 저주로 어려움을 겪음. -> 저차원 벡터공간에 존재하는 단어의 분산표상을 학습하는 연구 동기가 됨

A. 단어 임베딩

벡터 또는 단어 임베딩은 distributional hypothesis 를 기반으로 함, 즉 비슷한 의미를 지닌 단어는 비슷한 문맥에 등장하는 경향이 있을 것.

따라서 이 벡터들은 이웃한 단어의 특징을 잡아내고자 함.

분산표상 벡터의 주된 장점: 이 벡터들이 단어 간 유사성을 내포하고 있음. 코사인 유사도와 같은 지표를 사용함으로써 벡터 간 유사성 측정 가능.

일반적으로 단어 임베딩은 레이블이 없는 방대한 말뭉치에서 보조적인 목적함수를 최적화. 문맥 유사도 잡아내는데 효율적. 차원이 작은 덕분에 계산 빠르고 효율적.

워드 임베딩은 주로 문맥을 통해 학습됨.

Continuous-bag-of-words(CBOW)와 skip-gram 모델 제안

B. Word2Vec

Word2Vec 은 CBOW 와 Skip-gram 두 방식이 있음.

CBOW: k 개 만큼의 주변 단어가 주어졌을 때 중심 단어의 조건부 확률 계산

Skip-gram: 중심단어가 주어졌을 때 주변단어 예측

개별단어 임베딩의 한계

1. 두 개 이상의 단어의 조합이 개별 단어 벡터의 조합으로 표현될 수 없음

해결책: 동시등장단어(word coocurrence)에 기반한 구문을 식별하고 이들을 별도로 학습

최근 기법: 레이블이 없는 데이터로부터 n-gram 임베딩을 직접 학습시키는 것

2. 주변 단어의 작은 window(중심 단어를 예측하기 위해서 앞, 뒤로 몇 개의 단어를 볼지 범위) 내에만 기반한 임베딩을 학습

때로는 이러한 임베딩은 상반된 극성을 갖는 단어가 의미상 유사한 단어로 클러스터링됨

이는 SSWE(Sentiment specific word embedding)으로 해결 가능

단어 임베딩 주의사항: 임베딩이 사용된 어플리케이션에 크게 의존

단어 벡터를 재학습하여 현재 task space 에 맞추기 위해 task specific embedding 을 제안

딥러닝 모델에 자주 사용되는 임베딩 프레임워크들

Gensim, Pydsm, Dissect

(+) Word2Vec 은 딥 러닝 모델(Deep Learning Model)은 아님.

딥러닝은 입력층과 출력층 사이의 은닉층의 개수가 충분히 쌓인 신경망을 학습할 때를 말하는데 Word2Vec 는 입력층과 출력층 사이에 하나의 은닉층만이 존재함. 그래서 얕은신경망(Shallow Neural Network)이라고 부름.

출처: https://wikidocs.net/22660

C. 문자 임베딩

단어 임베딩은 문법적, 의미적 정보 잡아냄

품사태깅이나 개체명인식 같은 테스크에서는 단어 내부의 형태 정보 매우 유용

3. 콘볼루션 신경망

워드 임베딩이 인기를 끈 이후, 단어 결합이나 n-gram 으로부터 높은 수준의 피처를 추출해내는 효율적인 함수의 필요성 증대

추상화된 피처들은 감성분석, 요약, 기계번역, 질의응답 같은 다양한 NLP 문제에 사용될 수 있음

문장 모델링에서 CNN 을 학습하는 경우 학습 과정에서 단어 벡터(가중치)가 학습되는 초기 단어 임베딩 기법의 아이디어로 생각할 수 있음

CNN 은 문장의 잠재적인 semantic representation 을 만들어내기 위해 입력 문장으로부터 핵심적인 n-gram 피처 추출 능력을 갖고 있음

A. CNN 기본구조

1) 문장 모델링

문장의 i 번째 단어에 해당하는 임베딩 벡터를 wi, 임베딩 벡터의 차원수를 d 라고 함.

n 개의 단어로 이뤄진 문장이 주어졌을 때, 문장은 n x d 크기의 임베딩 행렬로 표현할 수 있음

wi, wi+1, …, 저의 결합(concatenation)을 wi:i+j

콘볼루션 필터 k 는 차원수가 hd 인 벡터

이 필터는 h 개 단어벡터에 적용됨

콘볼루션 필터에 의해 새로 추출된 피처 ci 는 wi:i+h-1 을 활용해 생성됨

Ci = f(wi:i+h-1 * k transpose + b)

CNN 에서 콘볼루션 필터의 수는 전형적으로 수 백개, 각 필터는 n-gram 의 특정 패턴을 추출함

콘볼루션 레이어는 대개 맥스풀링 계층이 후행함.

이유: 분류에 필요한 고정 길이의 출력을 제공하여 필터의 크기가 각기 달라도 입력값은 항상 고정된 차원의 출력으로 매핑함. 또한 가장 핵심적인 n-gram 피처를 유지하면서 출력의 파원을 줄임.

단어 임베딩은 랜덤 초기화나 레이블 없는 방대한 말뭉치에서 사전학습될 수 있음. 정답 데이터의 양이 적을 때 성능 향상에 때로는 유용함.

콘볼루션 계층과 맥스풀링의 조합은 보다 깊은 CNN 네트워크를 위해 종종 겹쳐 쌓게 됨.

Sequential convolution 은 고도로 추상화된 표현을 잡아내 문장의 분석을 개선할 수 있게 해줌.

필터는 문장 피처의 전체 요약을 만들기까지.

3) 윈도우 접근법

CNN 아키텍처는 완전한 자연어 문장을 벡터로 표현.

개체명인식, 품사태깅, SRL 같은 많은 NLP 문제는 단어 단위의 예측이 필요함

윈도우 접근법은 *단어의 범주(tag)가 기본적으로 이웃 단어에 의존할* 것이라고 가정함

각 단어에 대해 고정된 크기의 윈도우가 가정되고, 윈도우 내에 있는 하위 문장들이 고려됨

독립형(standalone) CNN 은 앞서 설명한 바와 같이 이러한 하위 문장에 적용되며 예측은 윈도우 중심에 있는 단어에 기인함

Poria et al.: 문장 각 단어에 태그를 붙이기 위해 multi-level deep CNN 제안

언어적 패턴의 셋과 결합되어 이들의 앙상블 분류기는 aspect detection 에 있어 좋은 성능을 냄

단어 수준 분류의 주요 목표는 일반적으로 전체 문장에 레이블 시퀀스를 할당하는 것

Condiitonal Random Field(CFR) 같은 구조화된 예측 기법은 때때로 인접한 클래스 레이블 간의 의존성을 더 잘 포착함

이 기법은 결국 전체 문장에 최대 스코어를 내는 결합된(cohesive) 레이블 시퀀스를 생성함

문맥적 범위를 넓히기 위해, 전통적인 윈도우 접근법은 종종 time-dealy neural network(TDNN)곽 ufgkqehla

콘볼루션은 시퀀스 전체의 모든 윈도우에서 수행되는데, 이런 콘볼루션들은 일반적으로 필터의 폭이 사전에 정의된다는 점에서 제약을 받음

전통적인 윈도우 접근법은 레이블이 달린 단어 주변의 윈도우에 있는 단어들만 고려하는 반면, TDNN 은 문장 내 모든 윈도우들을 동시에 고려.

때때로 TDNN 레이어는 CNN 아키텍처처럼 stack 되어 하위 계층에서 로컬 피처, 상위 계층에서 글로벌 피처를 추출함.

B. CNN 어플리케이션

NLP 문제에 CNN 을 적용한 주요 연구

Kim (2014) : 감성, 주관성, 질문유형 분류를 포함한 다양한 문장 분류 문제에 CNN 실험

특정 과업 학습 후 랜덤하게 초기화된 콘볼루션 필터는 목적하는 태스크에 유용한 특정한 n-gram 피처 감지기(deterctor)가 됨

BUT 장거리 의존성(long distance dependencies)를 모델링할 수 없음

Kalchbrenner et al. (2014): 문장의 의미를 모델링하기 위한 dynamic convolutional neural network(DCNN) 제안 for 문장의 의미 모델링

Dynamic k max pooling 전략 제안: 시퀀스 p 가 주어졌을 때 가장 활동적인 k 개의 피처를 뽑는 방법. 선택은 피처의 순서를 보존하지만, 특정 위치에는 민감하지 않음.

TDNN 개념 기반으로 dynamic k max pooling 전략 추가. 작은 폭의 필터가 입력 문자의 긴 범위를 커버할 수 있게 함,

➔ 문맥적 의미를 모델링하는 데 있어 개별 필터의 범위에 대해 언급했고, 필터의 도달 범위를 확장하는 방법론을 제안함

Mukerjee and Liu (2012) : 감정분석 문제는 극성(polarity)에 관계된 aspect 의 효과적인 추출을 필요로 함.

Ruder et al. (2016) : 좋은 결과를 위해 워드 임베딩과 aspect vector 를 결합한(concatenate)값을 입력하는 CNN 을 적용함

CNN 은 텍스트 길이에 따라 성능이 달라짐. 장문의 텍스트에 대한 CNN 모델의 성능은 좋았던 반면 짧은 텍스트에선 반대였음.

Wang et al. (2015b) : 짧은 텍스트의 표현(representation)을 모델링하는 데 CNN 제안. 단문의 외부적 지식이 사용된 multi-scale semantic units 를 도입한 의미론적 클러스터링(semantic clustering)을 제안함. CNN 은 이러한 유닛들을 결합하고 전체적인 표현(representation)을 만들어내는데 쓰임.

Poria et al. (2016) : CNN 네트워크를 활용해 트위터 텍스트에서 빈정되는 부분을 찾아내는 기법(sarcasm detection)을 제안함. 감정, 감성, 개성 데이터셋으로 사전 학습된 형태의 보조적인 지원이 state-of-the-art 성능을 이끌어내는 데 사용됐음.

CNN 은 또한 다른 task 에서도 널리 쓰임.

Denil et al. (2014): **요약**을 위해 문장을 구성하는 단어의 의미를 문서의 의미로 매핑시키는 데 DCNN 을 적용

학습과정 뿐 아니라 텍스트의 자동요약을 위한 인사이트를 제공하였음

QA 분야

Yih et al. (2014): 질의에 답할 때 지식 베이스(Knowledge Base)에서 어떤 사실을 찾아봐야 하는지 결정하기 위해 질의와 KB 사이의 의미적 유사성을 측정하는 방안 제시

의미론적 표현(semantic representation)을 생성하기 위해 CNN 사용.

분류 문제를 풀기 위한 세팅과 달리, 정답셋으로는 클래스 레이블 대신 긍정 혹은 부정적인 텍스트 쌍을 씀(예: 쿼리-문서)

Dong et al. (2015): 다중 aspect 로부터 질의를 분석하고 이해하는 한 편 질의의 표현을 만들기 위한 multi-column CNN(MCCNN)을 소개함

입력 질의에서 응답 유형과 문맥을 구성하는 aspect 로부터 정보를 추출하기 위해 다수의 column network 를 씀

KB 의 개체(entities)와 관계를 벡터로 표현함으로써, 그들은 후보가 되는 응답에 순위를 매기기 위하여 질의-응답 쌍을 사용하여 CNN 모델 학습시킴

Severyn and Moschitti (2016): 축약 네트워크를 질의와 응답 문장의 최적 표현(representation)을 모델링하는 데 사용함. 질의와 응답 쌍 사이에 단어를 매칭시킴으로써 임베딩에서 추가적인 피처를 제안함.

이러한 파라미터들은 네트워크에 의해 튜닝됨.

CNN 모델은 *분류를 넘어서는 의미론적 일치(semantic matching)*가 필요한 특정 NLP 태스크에도 적합함.

CNN 은 질의와 문서를 *고정된 차원의 의미 공간에 투영(project)*하는 데 사용됨.

의미 공간에서는 *쿼리와 문서 간 코사인 유사도*가 특정 쿼리에 해당하는 *문서의 랭킹*을 매기는 데 쓰임.

모델은 단어 시퀀스에서의 일시적인 context window 를 고려함으로써 쿼리나 문서에서의 풍부한 문맥 구조 추출을 시도함. 이 모델은 n-gram 수준에서의 문맥 피처를 잡아냄. 그 후 핵심 n-gram 은 전체를 아우르는 문장 벡터를 형성하기 위한 결합(aggregate)되는 콘볼루션과 맥스풀링 계층에 의해 발견됨.

CNN 은 문장에서 가장 중요한 정보를 뽑기 위한 방법과 연결되어 있음. 하지만 *기존의 맥스풀링 전략은 문장에서 가치 있는 정보를 종종 잃어버림*. 다중 이벤트 모델링(multiple-event modeling)에서 이러한 정보 손실 문제를 극복하기 위해, Chen et al. (2015b)는 수정된 풀링 전략 *Dynamic multi-pooling CNN DMCNN* 을 제안함

CNN 은 본질적으로 로컬 연결성(local connectivity), 가중치 공유, 풀링 등의 특징이 있음. 이러한 속성은 많은 task 에서 고도로 요구되는 *불변성(invariance)*를 어느 정도 보장함. 음성인식도 불변성을 필요로 함.

Abdel-Hamid et al. (2014): 하이브리드 CNN-HMM 모델을 사용함. 변동성은 종종 화자 차이에 기인한 음성 신호에서 종종 발견됨.

Palaz et al. (2015): CNN 기반의 음성 인식 시스템에 대한 집중적인 분석을 수행함. 원시 입력(raw input)과 음성(phone) 사이의 관계를 직접적으로 모델링하는 CNN 의 능력을 입증했고, 강건한(robust) 자동 스피치 인식 시스템을 만들어 냄.

기계번역

순차적인 정보와 장기 의존성에 대한 인내(perseverance)가 필요. 구조적으로 이러한 작업들은 이러한 피처들을 잃어버리는 CNN 네트워크에 적합하지 않음/

CNN 은 전반적으로 contextual window 내에 있는 의미적 단서를 추출하는 데 고도로 효율적. 그러나 CNN 은 매우 많은 데이터를 필요로 하여 데이터가 부족할 때는 문제가 됨.

CNN 의 다른 이슈는 먼 거리의 문맥 정보를 모델링하기가 불가능하고 그들의 표현(representation)에서 시퀀셜한 순서를 보존할 수 없다는 것.

문장 모델링, 기계 번역 등에는 Recursive NN 이 적합.

4. Recurrent Neural Networks

RNN 은 순차적인 정보를 처리하는 네트워크

전통적인 NN 과 달리 모든 입력값이 독립적이라고 가정함

'recurrent'라는 용어는 모델이 입력 시퀀스의 각 인스턴스에 대해 같은 작업을 수행하고 아웃풋은 이전 연산 및 결과에 의존적이라는 데에서 붙었음

일반적으로 recurrent unit 에 토큰을 하나씩 입력함으로써 고정 크기의 벡터가 시퀀스를 표현하기 위해 생성됨

즉, RNN 은 이전 연산결과를 '기억'하고, 현재 연산 과정에서 이 정보를 활용.

랭귀지 모델링, 기계번역, 음성인식, 이미지 캡셔닝 등과 같은 많은 NLP 작업에 적합

A. RNN 의 필요성

데이터를 순차적으로 처리 -> 언어에서 고유한 순차적인 성격을 포착할 수 있는 능력이 있음. 단어는 이전 단어를 바탕으로 의미를 갖게 됨. 이와 관련해서는 'dog'와 'hot dog'간의 의미 차이일 것.

RNN 은 이러한 *문맥 의존성을 모델링하기* 위하여 만들어짐

RNN 이 시쿠너스 모델링에 적합한 다른 요인은 매우 긴 문장, 단락, 심지어 문서를 포함해 다양한 텍스트 길이를 모델링할 수 있는 능력. CNN 과 달리 RNN 은 *무제한 컨텍스트를 포착할 수 있는 유연한 계산 step* 을 가짐. 임의의 길이의 입력값을 처리할 수 있음.

RNN 은 또한 시간 분산 조인트 처리를 위한 네트워크 지원을 제공. 품사태깅 같은 시퀀스 레이블링 작업.

CNN 과 RNN 은 문장을 모델링할 때 다른 목적(함수)를 가짐.

RNN 이 경계없는 긴 문장을 생성하려고 하는 반면, CNN 은 가장 중요한 n-gram 을 추출하려고 함.

둘다 n-gram 피처를 잡아내는 데 효율적이지만, 단어 순서에 대한 민감도가 지역적으로(locally) 제한되며 장기(long-term) 의존성은 보통 무시됨.

B. 네트워크 구조

1) RNN 기본 구조

St = f(Uxt + Wst-1)

St: 현재 입력 및 이전 단계의 히든 스테이트를 기반으로 계싼됨

F: 비선형 변환

U, V, W 는 모든 시점에서 공유됨

NLP 맥락에서 xt 는 일반적으로 one-hot 벡터/임베딩 벡터.

RNN 의 히든 스테이트: 일반적으로 가장 중요한 요소.

히든 스테이트는 다른 time step 의 정보를 누적한 네트워크의 메모리 요소로 간주될 수 있음. 그러나 실제에선 이러한 간단한 RNN 네트워크는 악명 높은 베니싱 그래디언트 문제가 있어 LSTM, GRU, ResNet 에 의해 극복됨

2) Long Short-Term Memory

LSTM: 간단한 RNN 에 forget gate 를 추가하여, 배니싱 그래디언트 문제, 익스플로딩 그래디언트 문제를 모두 극복할 수 있었음

오차가 무제한적인 time step 에 역전파될 수 있음.

3 개의 게이트, 즉 input/forget/output gate 로 구성됨.

3) Gated Recurrent Units

LSTM 과 유사한 성능이면서 구조적으로는 더 간단함.

GRU 는 reset gate 와 update gate 의 두 개 gate 로 구성되며 LSTM 처럼 메모리를 보호함.

GRU 는 LSTM 보다 효율적인 RNN 이 될 수 있음.

C. 어플리케이션

1) 단어 수준 분류를 위한 RNN

Lample et al. (2016): 개체명 인식을 위한 bidirectional LSTM 모델

타겟 단어 주변의 임의의 긴 맥락 정보를 포착해 두 개의 고정된 크기의 벡터를 반환. 그 위에 또 하나의 fully-connected 레이어가 있음. 최종 개체명 태깅에 CRF 레이어 사용.

언어 모델링(Language modeling)면에서도 빈도 기반의 기존 방법론보다 상당한 개선을 보여줌.

Graves(2013): 긴 범위의 문맥 구조에서 복잡한 시퀀스를 모델링하는 데 RNN 이 효율적임

은닉층이 여러 개인 deep RNN 을 제안함.

Kim et al. (2016): 형태적(morphological) 정보를 포함하는 문자 기반의 표현(representation) 사용을 탐색함. 베이스라인인 단어 기반의 LSTM 모델보다 나은 성능을 나타낸 다는 점에서, 문자 기반의 모델이 문자로부터 의미, 철자(orthographic) 정보 모두를 추출할 수 있다는 것을 보여줌.

2) 문장 수준 분류를 위한 RNN

Wang et al. (2015a): LSTM 으로 트위터 문장을 인코딩하는 모델 제안. 히든 스테이트는 감성 극성을 예측하는 데 사용됨.

이러한 간단한 전략은 좀 더 복잡한 DCNN 구조와 비교해도 좋은 성능을 나타냄. DCNN 은 CNN 모델이 장기 의존성을 포착할 수 있도록 디자인 됨. 저자들은 LSTM 게이트가 'not'이라는 단어가 감성의 극성을 반전시키는 걸 포착할 수 있음을 보임.

CNN 과 유사하기, RNN 의 히든 스테이트는 또한 텍스트 사이의 의미적 일치(semantic matching)에 쓰일 수 있음.

대화시스템에서 Low et al.(2015): Dual-LSTM 으로 메시지를 후보 응답과 매칭시키는 모델을 제안. 메시지와 응답을 고정된 크기의 벡터로 인코딩하고, 그리고 나서 후보 응답의 순위를 매기기 위해 둘을 내적함.

3) 문장 생성을 위한 RNN

텍스트 혹은 시각적 데이터가 주어졌을 때, deep LSTM 은 기계번역, 이미지 캡셔닝 등 태스크에서 합리적인 성능을 보여줌.

이러한 사례에서 RNN 은 디코더(decoder)라고 불림.

Sutskever et al. (2014): 시퀀스와 다른 시퀀스를 매핑시키는 일반적인 deep LSTM encoder-decoder 프레임워크 제안. LSTM 으로 인코딩된 고정된 크기의 벡터는 데코더로 불리는 또다른 LSTM 의 초기 상태(initial state)로 쓰임

추론 과정에서 디코더는 토큰을 하나씩 생성하고, 직전에 생성된 토큰으로 히든 스테이트를 업데이트.

+) end-to-end 기계번역을 위한 4 개 계층으로 이뤄진 LSTM 을 실험했고, 좋은 성능을 보여줌.

Vinyals and Le(2015): 동일한 프레임워크를 인간 대화를 모델링하는 데 제공함

1 억개 이상의 메시지-응답 쌍을 학습했을 때, LSTM 디코더는 오픈 도메인에서 아주 흥미로운 응답을 생성할 수 있음.

Li et al. (2016a): 개별 화자의 인물정보를 내포하는 constant persona vector 를 디코더 입력값으로 사용할 것으로 제안.

언어는 텍스트 입력값을 표현하는 의미 벡터들에 주로 기반해 생성됨. 유사한 프레임워크가 이미지 기반의 언어 생성에서도 성공적으로 사용되고 있음. 이미지 기반의 언어 생성에서는 가시적 피처(visual feature)가 LSTM 디코더를 조건부화하는 데 쓰임. 이미지 기반의 언어 생성에서는 가시적 피처(visual feature)가 LSTM 디코더를 조건부화하는데 쓰임.

비주얼 QA 는 텍스트와 이미지 둘 모두에 기반한 언어 생성을 요구하는 또 다른 태스크임.

Malinowski et al. (2015): CNN 에 의해 모델링된 입력 이미지와 LSTM 으로 모델링된 텍스트를 조건으로 이미지 캡션의 시퀀스를 예측하는 end-to-end 딥러닝 모델을 최초로 제시.

D. 어텐션 매커니즘

기존의 인코더-디코더 프레임워크 문제: 인코더가 때로는 해당 작업과 완전히 관련되지 않은 정보까지도 인코딩해야 됨.

어텐션 매커니즘: 디코더가 입력 시퀀스를 다시 참조할 수 있게 하여 위의 문제를 완화하려고 시도함.

디코더는 마지막 히든 스테이트와 생성된 토큰에 더하여 'context' 벡터에 대해 조건부화함

특히 긴 시퀀스에 대해 모델의 성능을 향상시킴.

이미지캡셔닝. Xu et al. (2015): 입력 이미지의 다른 부분으로 LSTM 디코더를 조건부화

어텐션 시그널은 이전의 히든 스테이트 및 CNN 이 뽑아낸 피처에 의해 결정됨

Vinyals et al. (2015b): 파싱 트리를 선형화(linearing)함으로써 구문 파싱 문제를 시퀀스 간 학습 문제로 풀었음

어텐션 매커니즘은 이 작업에서도 효율적(data-efficient)임이 입증됨

입력 시퀀스를 참조하는 또 다른 단계는 대화 생성과 텍스트 요약과 같은 작업에 유용한 특정 조건 하에서 입력 단어 또는 하위 시퀀스를 출력 시퀀스에 복사하는 것.

Aspect 기반의 감정분석. *Wang et al. (2016):* aspect 임베딩을 사용한 어텐션 기법의 해법 제안. 어텐션 모듈은 분류할 aspect 에 영향을 주는 문자의 선택 영역에 초점을 둠.

Tang et al. (2016): 멀티플-홉 어텐션을 사용하는 메모리 네트워크에 기반한 해법 채택. 계산계층은 메모리의 정보 영역 대부분에 대한 검색을 향상시키고 결과적으로 분류를 도움.