4주차 논문리뷰: Ask MeAnything: Dynamic Memory Networks for Natural Language Processing

0. Abstract

- 대부분의 자연어 처리 작업은 QA 문제로 변환 가능
- 입력된 언어 정보에 대해 질문을 처리하고 관련된 답변을 생성하는 **동적 메모리 네트워크(DMN)**라는 신경망 구조를 소개
- **DMN**
  - 입력된 문장과 질문을 처리
  - **에피소드 메모리(episodic memory)**를 형성
  - 그 정보를 바탕으로 답변을 생성
  - 질문은 반복적인 **어텐션(attention)** 과정을 통해 입력에 집중할 대상을 결정하고, 이전 반복에서 얻은 결과에 따라 조건부로 어텐션을 조정
  - end-to-end 훈련 가능
  - **질문 응답**(Facebook의 bAbI 데이터셋), **감정 분석을 위한 텍스트 분류**(Stanford Sentiment Treebank), **품사 태깅을 위한 시퀀스 모델링**(WSJ-PTB)에서 SOTA 달성

💡

1. **Attention**
   : 입력 데이터 중 어떤 부분에 집중할지를 결정하는 메커니즘

   → 질문에 따라 중요한 정보에 집중하고 답변을 생성하는 과정에서 매우 중요하게 작동

2. **episodic memory**
   : 입력 정보 중에서 반복적인 어텐션 과정을 통해 선택된 중요한 정보를 저장하고, 이를 기반으로 답을 유추하는 메모리

1. Introduction

- **QA**: 텍스트의 의미를 이해하고, 관련된 사실을 추론할 수 있는 능력이 요구되는 복잡한 자연어 처리 작업
  - **기계 번역**: "프랑스어로 번역이 무엇인가?"라는 질문
  - **개체명 인식(named entity recognition, NER):** "이 문장에서 개체명 태그는 무엇인가?"라는 질문
  - **품사 태깅(part-of-speech tagging, POS):** "품사 태그는 무엇인가?"라는 질문
  - **감정 분석(sentiment analysis)**: "이 문장의 감정은 무엇인가?"라는 질문

→ 이러한 QA를 해결할 수 있는 신경망 기반의 DMN 제안

- **DMN**
  - **입력-질문-답변 트리플렛(triplet)**을 사용한 훈련
  - 시퀀스 태깅, 분류, sequence-to-sequence 작업 및 추론이 필요한 QA 문제를 해결 가능
  - **메모리 모듈: 검색된 사실을 기반으로 추론 & 모든 관련 정보를 벡터로 표현**
  - **답변 모듈: 메모리 모듈에서 받은 벡터를 바탕으로 답변을 생성**

1. 모든 입력과 질문에 대한 **표현을 계산**

2. 질문 표현이 반복적인 **어텐션 프로세스를 유발**하며, 이를 통해 입력에서 관련된 사실을 검색

I: Jane went to the hallway.
I: Mary walked to the bathroom.
I: Sandra went to the garden.
I: Daniel went back to the garden.
I: Sandra took the milk there.
Q: Where is the milk?
A: garden
I: It started boring, but then it got interesting.
Q: What's the sentiment?
A: positive
Q: POS tags?
A: PRP VBD JJ , CC RB PRP VBD JJ .

→ 이 논문에서 평가한 작업들의 입력, 질문 및 답변 예시

- **개체명 인식(NER)**: 텍스트에서 사람, 장소, 조직 같은 고유명사를 식별하는 작업
- **품사 태깅(POS tagging)**: 문장에서 각 단어의 품사를 식별하는 작업
- **시퀀스 투 시퀀스(sequence-to-sequence)**: 입력된 시퀀스(예: 문장)를 다른 시퀀스로 변환하는 작업. (ex. 기계 번역)
- **메모리 모듈**: 어텐션을 통해 선택된 정보를 바탕으로 한층 더 깊이 있는 추론을 수행하는 부분.

2. Dynamic Memory Networks

- DMN은 **네 가지 주요 모듈**로 구성된 신경망 아키텍처
  1. **입력 모듈(Input Module)**: 주어진 텍스트 입력을 벡터 표현으로 변환
  2. **질문 모듈(Question Module)**: 입력 모듈과 유사하게, 주어진 질문을 벡터 표현으로 변환

     → 이 벡터 표현은 에피소드 메모리 모듈에 전달되며, 에피소드 메모리 모듈이 반복적으로 질문에 대한 정보를 기반으로 메모리를 갱신하는 데 사용됨

3. **에피소드 메모리 모듈(Episodic Memory Module)**: 입력된 텍스트에서 어떤 부분에 집중할지 결정하는 **어텐션 메커니즘**을 사용
   → 질문과 이전 메모리를 고려해 새로운 메모리 벡터를 생성하고, 반복을 통해 추가적인 관련 정보를 검색하여 메모리를 갱신
4. **답변 모듈(Answer Module)**: 최종 메모리 벡터를 바탕으로 답변을 생성하는 모듈

1. Input Module

- 자연어 처리 문제에서 입력은 일반적으로 단어 시퀀스
- **입력 시퀀스**: RNN을 통한 인코딩 & 각 시점에서 단어 임베딩이 RNN의 입력으로 주어짐
  - 입력 시퀀스가 단일 문장일 경우, 입력 모듈의 출력은 RNN의 마지막 hidden states
  - 입력 시퀀스가 여러 문장일 경우, 문장 사이에 문장 종료 토큰(end-of-sentence token)을 삽입하고, 문장 종료 토큰에 해당하는 hidden states 를 최종 출력으로 사용

- RNN 선택
  - 실험에서는 **GRU(Gated Recurrent Unit)** 사용
  - LSTM(Long Short-Term Memory)도 실험했지만 비슷한 성능을 보였고, 계산 비용이 더 많이 들기 때문에 GRU를 선택
    - GRU는 기울기 소실 문제를 덜 겪음

- **GRU의 내부 동작**
  1. **업데이트 게이트(zt)**

$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1} + b^{(z)}\right)$$

     : 입력과 이전의 은닉 상태를 기반으로 은닉 상태를 얼마나 업데이트할지 결정하는 게이트

2. **리셋 게이트(rt)**
   : 이전 은닉 상태를 얼마나 무시할지 결정하는 게이트

$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1} + b^{(r)}\right)$$

3. **후보 은닉 상태(h̃t)**

   : 새로운 은닉 상태를 계산하는 단계로, 리셋 게이트의 영향도 반영됨

$$\tilde{h}_t = \tanh\left(Wx_t + r_t \circ Uh_{t-1} + b^{(h)}\right)$$

4. **최종 은닉 상태(ht)**

   : 업데이트 게이트를 사용하여 이전 은닉 상태와 새로운 후보 은닉 상태 간의 가중치를 조합해 최종 은닉 상태를 결정

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

2. Question Module

- 입력 시퀀스와 마찬가지로 질문도 **RNN을 통한 인코딩**
- 각 질문 단어는 임베딩된 후, 시점별로 GRU의 은닉 상태가 업데이트됨
- **GRU 수식**

$$q_t = GRU(L[w_t^Q], q_{t-1})$$

  - L: 임베딩 행렬 - 단어를 고정 길이의 벡터로 변환
  - w_t^Q: 질문 시퀀스의 t번째 단어
  - q_t: 질문을 표현하는 벡터
  - 질문 모듈의 최종 출력은 순환 네트워크 인코더의 마지막 은닉 상태 qT_Q로 정의

3. Episodic Memory Module

- 에피소드 메모리 모듈은 입력 모듈에서 출력된 벡터 표현을 반복적으로 갱신하며 내부 메모리를 업데이트

  → 어텐션 메커니즘 사용

- **작동 원리**
  - 각 반복에서 어텐션 메커니즘은 질문 q와 이전 메모리 m^i−1을 사용해 입력된 사실 표현 c_t 중 **어디에 집중할지를 결정**
  - 메모리는 **GRU를 사용하여 업데이트**되며, 메모리 벡터 e_i는 새로운 정보를 포함하는 **에피소드**를 형성
  - 이 에피소드 벡터는 질문에 대한 답변을 생성하기 위해 **최종 메모리 m^TM로 전달**

- 반복적인 프로세스를 통해 이 모듈은 여러 번 입력을 다시 살펴보고 새로운 정보를 찾을 수 있음
  - ex) "축구공은 어디에 있습니까?"라고 물으면, 첫 번째 반복에서는 축구공에 대한 정보가 있는 문장을 검색 & 반복을 통해 축구공과 관련된 다른 정보가 있는 문장을 찾아 메모리에 추가

- **어텐션 메커니즘**
  - 각 후보 사실 c_t와 이전 메모리, 질문을 입력으로 받아 게이트 값 gt^i를 계산.

    → 이 게이트 값은 **현재 반복에서 어느 정도 해당 입력에 집중할지를 결정**

  - **스코어링 함수** G(c,m,q): 입력, 메모리, 질문 사이의 **유사성을 기반으로 게이트 값을 산출**

    → 2층 feed forward 신경망으로 정의됨

1. 입력 벡터 ct, 이전 메모리 m^i-1, 질문벡터 q를 기반으로 게이트 값 gt^i 계산

$$z(c, m, q) = \left[c, m, q, c \circ q, c \circ m, |c - q|, |c - m|, c^T W^{(b)} q, c^T W^{(b)} m\right]$$

- o: **원소별 곱셈**을 의미
- W^(b): 학습된 **행렬 파라미터**로, 입력과 질문 간의 내적을 계산

$$g_t^i = \sigma\left(W^{(2)}\tanh\left(W^{(1)}z(c_t, m, q) + b^{(1)}\right) + b^{(2)}\right)$$

- W(1), W(2): 학습된 **가중치 행렬**
- b(1), b(2): 편향값
- tanh: **하이퍼볼릭 탄젠트** 활성화 함수로, 비선형성을 추가
- σ: **시그모이드 함수**로, 출력을 [0, 1] 사이의 값으로 만듦

- **메모리 업데이트 메커니즘**
  - 각 반복에서 게이트 값을 사용해 GRU를 통해 입력 c1, .. , cT에 대한 에피소드 벡터를 계산
  - 최종적으로 답변 모듈로 전달되는 에피소드 벡터는 GRU의 마지막 상태로 정의됨

$$h_t^i = g_t^i \cdot GRU(c_t, h_{t-1}^i) + (1 - g_t^i) \cdot h_{t-1}^i$$

  - ht^i: t번째 입력에 대한 새로운 은닉 상태
  - gt^i: 어텐션 메커니즘에서 계산된 게이트 값으로, 해당 입력에 어느 정도 집중할지를 결정
  - h_t-1^i: 이전 은닉 상태(현재 메모리를 반영)
  - GRU(c_t, h_t-1^i): 새로운 입력 c_t에 따라 업데이트된 은닉 상태

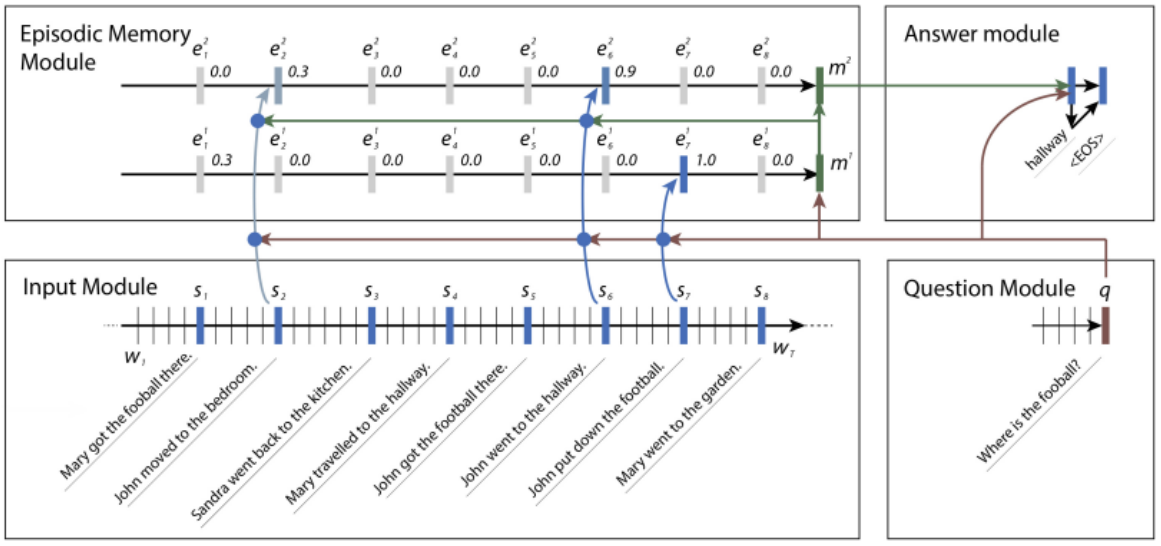


Figure 3. Real example of an input list of sentences and the attention gates that are triggered by a specific question from the bAbI tasks (Weston et al., 2015a). Gate values g_t^i are shown above the corresponding vectors. The gates change with each search over inputs. We do not draw connections for gates that are close to zero. Note that the second iteration has wrongly placed some weight in sentence 2, which makes some intuitive sense, as sentence 2 is another place John had been.

→ 실제 입력 리스트와 특정 질문에 의해 트리거된 어텐션 게이트의 예

- 게이트 값은 각 입력에 대해 계산된 벡터들로 나타나며, 반복이 거듭될수록 게이트 값이 달라짐
- 각 반복에서 **특정 입력에 할당된 게이트 값이 더 커지면 해당 입력에 더 많은 주의를 기울이게 됨**

4. Answer Module

- 답변 모듈은 벡터를 입력받아 답변을 생성함
- 작업 유형에 따라 에피소드 메모리가 끝난 후 한 번 실행되거나, 각 시점에서 반복적으로 실행될 수 있음
- **GRU 이용**
  - q: 질문
  - a_t-1: 이전 GRU 은닉 상태
  - y_t-1: 이전에 예측한 출력
  - GRU의 은닉 상태 a_t는 현재 시점에서의 답변을 생성하는 데 사용됨
  - 출력: **cross-entropy error**로 훈련됨
  - 시퀀스 끝을 나타내는 종료 토큰 추가됨

- 출력 예측

$$y_t = \text{softmax}(W^{(a)} a_t)$$

→ W(a): 학습된 가중치 행렬

→ 소프트맥스(softmax): 분류 작업에서 사용되는 함수로, 각 클래스에 속할 확률을 계산

- **GRU 업데이트**

$$a_t = \text{GRU}([y_{t-1}, q], a_{t-1})$$

→ y_t-1: 이전에 예측된 단어

→ q: 질문 벡터(이 벡터와 예측된 단어를 결합하여 GRU의 입력으로 사용)

- **시퀀스 모델링 작업**

  : 입력 시퀀스의 각 단어에 대한 레이블을 예측

  - DMN이 각 입력 단어에 대해 동일한 방식으로 실행됨
  - e_i = h_i^i로 업데이트되며, 각 단어에 대한 게이트 값이 계산됨
  - **첫 번째 반복에서는 모든 단어에 대해 동일한 게이트가 사용되지만, 이후 반복에서는 다른 게이트 값이 계산**

5. Training

- **Supervised classification problem**로 정의됨
- 답변 시퀀스의 **cross-entropy error**를 최소화하는 방식으로 진행
- **Backpropagation과 Gradient Descent**: 모든 모듈이 벡터 표현을 기반으로 상호작용하며, 각 모듈은 **미분 가능한 신경망** 구조

3. Related Work

1. Deep Learning

- **RNN(재귀 신경망):** 구문 분석, 감정 분석, 질문 응답, 논리적 추론 등의 작업에 사용
  - 메모리, 질문 모듈이 없기 때문에 여러 문장 간에 추론을 요구하는 작업 처리 X

- **chain-structured RNN:** 언어 모델링, 음성 인식 및 이미지로부터 문장 생성 등에 성공적으로 적용됨

- **Sequence-to-Sequence** 모델: 기계 번역 작업에서 많이 사용되는데, 에피소드 메모리 없이 입력 시퀀스를 바로 출력 시퀀스로 매핑하는 특별한 경우에 해당

2. Attention and Memory

- Attention 메커니즘: 이미지 분류, 자동 이미지 Captioning, 기계 번역에 유용
- **Neural Turing Machines:** 메모리를 활용해 알고리즘 문제(예: 리스트 정렬)를 해결
- 메모리 네트워크가 자연어 처리 작업에서 추가적으로 메모리와 질문 모듈을 포함하도록 확장됨

3. NLP Applications

- **DMN**: 여러 자연어 처리 작업에 적용할 수 있는 일반적인 모델로, 여러 NLP 문제에 대해 SOTA 달성
- **QA**: 대규모 지식베이스(KB)나 신경망을 사용하며, 일부는 문장만 사용하기도 함
  - 정답을 도출하지 못하면, 사실에 접근하지 못하거나 추론을 할 수 없기 때문
- **감정분석**
  - 매우 성공적인 분류 작업 중 하나
  - **Stanford Sentiment Treebank**는 표준 벤치마크 데이터셋으로 사용됨

    - **다중 필터 기반 CNN**을 사용해 감정 분석에서 SOTA 달성

- **품사태깅**
    - **WSJ-PTB** 데이터셋 사용
    - **semisupervised** 기법 사용

4. Neuroscience

- 신경과학에서, 에피소드 메모리는 인간이 공간적 및 시간적 맥락에서 특정 경험을 저장하는 것을 의미

  → DMN의 에피소드 메모리 모듈에 영향: 질문과 관련된 정보를 반복적으로 검색하여 새로운 정보를 추론할 수 있게 함

4. Experiments

- DMN은 질문 응답, 품사 태깅, 감정 분석 작업에서 독립적으로 훈련됨
- 훈련 시 모든 데이터셋에 대해 학습, 개발, 테스트 세트를 사용
    - 개발 세트가 없는 경우 학습 세트의 10%를 개발 세트로 사용
- **Adam 옵티마이저**를 사용해 훈련
- 단어 벡터는 **GloVe**로 미리 학습된 것을 사용
    - GloVe: 단어를 고정된 차원의 벡터로 표현하는 **단어 임베딩(Word Embedding)** 기법

1. Question Answering

- Facebook의 **bAbI 데이터셋:** 모델이 사실을 검색하고 그 사실을 기반으로 추론할 수 있는 능력을 테스트하기 위한 합성 데이터셋
    - **coreference resolution**, **귀납적 추론**(inductive reasoning)과 같은 여러 종류의 문제를 포함

- **coreference resolution:**
  텍스트 내에서 같은 대상을 지칭하는
  명사 또는 대명사를 식별하는 작업
  = 여러 문장에서 등장하는 대명사나 명사들이 동일한 개체를 지칭하고 있는지를 결정하는 과정

- **objective function**

$$J = \alpha E_{CE}(\text{Gates}) + \beta E_{CE}(\text{Answers})$$

→ E_CE: cross-entropy error(예측값과 실제값의 차이를 측정)

→ α, β: 하이퍼파라미터(게이트와 답변에 대한 오류를 각각 조절)

  - α = 1, β = 0에서 시작하여 게이트 오류 최소화
  - β = 1로 설정해 답변 오류를 추가로 학습
  - 각 반복에서 어텐션 게이트는 입력된 문장 중에서 어떤 문장이 중요한지 결정
  - 최종 벡터 e_i: softmax 통해 계산됨

$$\text{softmax}(g_t^i) = \frac{\exp(g_t^i)}{\sum_{i-1}^{T_C} \exp(g_i^i)}$$

→ g_t^i: 특정 문장에 대한 게이트 값

→ softmax: 모델이 한 번에 하나의 문장에 집중하도록 하는 특성(분류 문제에서 각 클래스에 속할 확률을 계산해 주는 함수)

2. Text Classification: Sentiment Analysis

- **Stanford Sentiment Treebank(SST)**: 감정 분석을 위한 대표적엔 데이터셋
  - 구(phrase) 레벨에서의 **세분화된 레이블**(fine-grained labels)과 **문장 단위**의 레이블을 포함
  - **Fine-grained 분류**: 문장이나 구문을 5단계(매우 부정적, 부정적, 중립적, 긍정적, 매우 긍정적)로 분류

- **이진 분류**: 중립적이지 않은 모든 문장을 긍정적 또는 부정적으로 분류

- DMN은 **binary classification**(이진 분류)에서 SOTA를 달성했으며, fine-grained 분류에서도 다른 모델들과 비교할 만한 성능을 보임

  →
  DMN이 이진 분류에서 88.6%, 세분화된 분류에서 52.1%의 정확도를 기록

Task	Binary	Fine-grained
MV-RNN	82.9	44.4
RNTN	85.4	45.7
DCNN	86.8	48.5
PVec	87.8	48.7
CNN-MC	88.1	47.4
DRNN	86.6	49.8
CT-LSTM	88.0	51.0
DMN	**88.6**	**52.1**

Table 2. Test accuracies for sentiment analysis on the Stanford Sentiment Treebank. MV-RNN and RNTN: Socher et al. (2013). DCNN: Kalchbrenner et al. (2014). PVec: Le & Mikolov. (2014). CNN-MC: Kim (2014). DRNN: Irsoy & Cardie (2015), 2014. CT-LSTM: Tai et al. (2015)

3. Sequence Tagging: Part-of-Speech Tagging

- 품사 태깅(POS tagging)은 문장의 각 단어를 품사로 분류하는 작업
- **WSJ-PTB**(Wall Street Journal 데이터셋)를 사용하여 실험
- **Søgaard**(2011)의 방법과 성능 비교
- **DMN은 97.56%의 정확도로 SOTA 달성, 개발 세트에서도 97.5%의 정확도 기록**

Model	Acc (%)
SVMTool	97.15
Sogaard	97.27
Suzuki et al.	97.40
Spoustova et al.	97.44
SCNN	97.50
DMN	**97.56**

Table 3. Test accuracies on WSJ-PTB

corporate tree structure in the retrieval process.

4. Quantitative Analysis of Episodic Memory Module

- NLP 작업에서 에피소드 메모리 모듈이 얼마나 중요한지, 입력에 대해 몇 번의 반복적인 처리를 수행하는 것이 정확도에 미치는 영향을 분석
- 0-pass DMN에서는 **에피소드 메모리 모듈이 사용되지 않으며**, 모든 출력이 입력 모듈에서 바로 답변 모듈로 전달됨 = 반복적인 처리가 없는 상태

Max passes	task 3 three-facts	task 7 count	task 8 lists/sets	sentiment (fine grain)
0 pass	0	48.8	33.6	50.0
1 pass	0	48.8	54.0	51.5
2 pass	16.7	49.1	55.6	**52.1**
3 pass	64.7	83.4	83.4	50.1
5 pass	**95.2**	**96.9**	**96.5**	N/A

Table 4. Effectiveness of episodic memory module across tasks. Each row shows the final accuracy in term of percentages with a different maximum limit for the number of passes the episodic memory module can take. Note that for the 0-pass DMN, the network essential reduces to the output of the attention module.

→ bAbI 작업과 Stanford Sentiment Treebank(SST)에서의 정확도

→ **여러번 반복을 수행하는 것이 어려운 추론 작업에서 높은 성능을 달성하는 데 매우 중요하다는 것을 강조**

→ **5-pass**: 최대 5번의 패스를 허용한 경우로, 가장 높은 성능을 달성

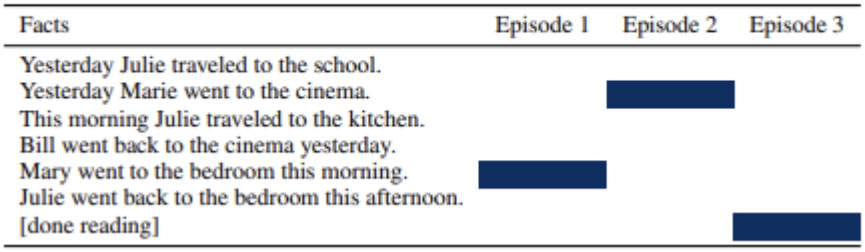
- **reasoning 작업**: 여러번 반복 필요
- **감정 분석 작업:** 차이가 크지 않지만, 두 번의 패스가 한 번의 패스보다 높은 성능

- 간단한 감정 단어만 포함된 경우에는 추가적인 패스가 큰 이점 X

5. Qualitative Analysis of Episodic Memory Module

- **bAbI 데이터셋**의 질문에 대해, **세 번의 반복**을 수행할 때 각 패스가 입력의 다른 부분에 집중하는 과정을 분석

Question: Where was Mary before the Bedroom?
Answer: Cinema.



→ 어두운 색으로 표시된 부분이 모델이 주목한 영역

- 감정 분석에 대해서도 에피소드 메모리 모듈을 평가

  → DMN이 1회 반복과 2회 반복에서 다르게 작동하는 예시를 통해, 2회 반복된 경우 더 정확하게 결과를 도출할 수 있음을 보여줌

  - **2회 반복 DMN의 Attention**:
    - 2회 반복된 DMN은 1회 반복된 DMN보다 더 **집중된 Attention**을 보임.
    - 반복 횟수가 적을수록 RNN의 은닉 상태가 인접한 시간 단계의 내용을 더 많이 포함해야 하기 때문에, 한 번의 패스에서는 중요한 부분에만 어텐션을 주기 어려움
    - 2회 반복에서는 각 패스에서 더 중요한 정보를 분명하게 추출할 수 있음
  - **2회 반복에서의 키워드 집중**:
    - 2회 반복된 DMN에서는 중요한 키워드에 더 집중하게 됨.
    - ex) "best"와 같은 긍정적인 단어가 처음에는 높은 중요성을 가졌지만, 2회 반복 후 문맥이 명확해지면서 "best"보다는 "lukewarm"과 같은 단어가 더 중요해짐.

      → 문장이 진행됨에 따라 어텐션이 적절한 단어에 맞춰 변화하는 과정

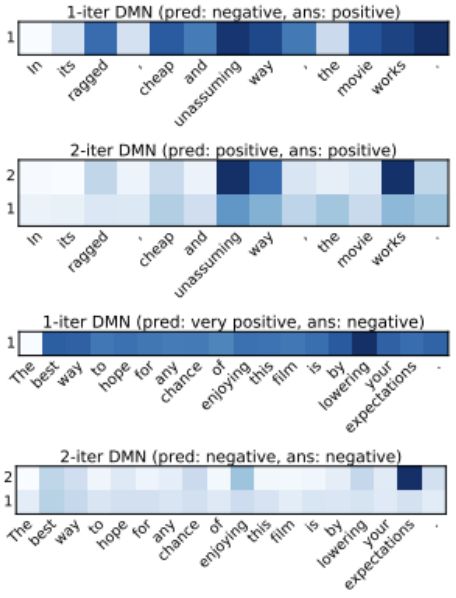


Figure 4. Attention weights for sentiment examples that were only labeled correctly by a DMN with two episodes. The y-axis shows the episode number. This sentence demonstrates a case where the ability to iterate allows the DMN to sharply focus on relevant words.

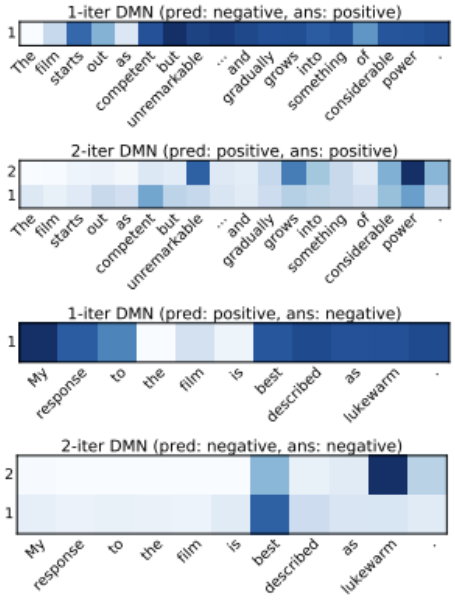


Figure 5. These sentence demonstrate cases where initially positive words lost their importance after the entire sentence context became clear either through a contrastive conjunction ("but") or a modified action "best described."

→ Figure4: DMN이 두 번의 에피소드를 통해 올바른 감정 분석을 수행한 예시

- 첫 번째 반복에서는 감정 단어에 집중하고, 두 번째 반복에서는 문맥을 고려하여 더 정확한 판단

→ Figure5: 긍정적인 단어가 처음에 중요하게 보였지만, 반복 후 문맥에서 그 중요성이 줄어들고 대조적인 단어가 더 중요해지는 상황을 설명

5. Conclusion

- **DMN 모델**은 다양한 NLP 응용(분류, 질문 응답, 시퀀스 모델링)에서 잠재적으로 일반화할 수 있는 아키텍처

- 단일 아키텍처로 여러 NLP 문제를 해결할 수 있는 가능성을 제시
- **end-to-end로 학습 가능**