Content Attention Model for Aspect Based Sentiment Analysis

Abstract

- 논문의 저자들은 기존의 content attention 기반 ABSA 연구에 대해서 문제 의식을 가지고 있다.
- 기존의 연구들은 대부분 문장 내에 주어진 aspects 들에 대한 단어들과의 연관성을 고려하지 않고 sentiment words 과 shift(??) 를 식별하는데 초점을 둔다.
- 따라서 기존의 연구들은 복잡한 문장 구조를 가지는 문장이다 multi aspects를 가지고 있는 문장에 대해서는 충분히 잘 다루지 못한다.
- 논문의 저자들은 이를 타파하기 위해서 aspect 기반 감성 분류 모델에 기반한 새로운 content attiention 모델을 제안한다. 두 개의 attention 메커니즘과 함께한다.
    - sentence-level content attention mechanism
        - 전역적인 관점(global perspective)으로부터 주어진 aspect 에 대한 중요한 정보를 포착하는게 가능하다.
    - context attention mechanism
        - 단어들의 순서와 그들은 상관 관계를 동시에 고려한다.
        - `customized memories` 시리즈에 그것들(단어들?을 의미하는듯)을 임베딩하면서...

Introduction

- ABSA 의 주요 목적은 주어진 개체들에서 aspects 들을 추출하는 것과 각각의 aspect 에 대한 감성 표현을 결정하는 것이다.
- 논문의 저자들은 aspect 기반의 감성 분류 문제에 대해서 집중한다.
    - 주 목적은 특정 측면에 대한 트윗, 코멘트로 전달되어지는 사용자의 의견이 긍정적이냐 부정적이냐 중립적이냐를 결정하는 것이다.
    - semeval 데이터에서 가져온 하나의 예시
        - Looking around, I saw a room full of New Yorkers enjoying a real meal in a real restaurant, not a clubhouse of the fabulous trying to be seen
        - 이 문장의 aspect 들은 `room`, `meal`, `clubhouse` 등등이라고 할 수 있는데, 이 aspect 들에 대해 긍정인지 부정인지 중립인지에 대한 분류 작업을 시행한다.
- context words들이 주어진 aspect 에 대한 감성 극성과 관련 있어 보이지만, 많은 경우에는 contexst words 들의 일부들만이 주어진 aspect 에 대한 감성 극성과 관련이 있다.
    - 위의 리뷰의 예시를 사용하자면, 위의 리뷰에서 aspect 는 room, meal, clubhouse 등등이 있었다.
    - aspect meal 에 대한 감성 극성을 나타내는 관련있는 단어는 enjoying 이며, 이는 context word 가 분석하려고 하

는 aspect 에 대한 감성을 잘 나타내주는 것처럼 보인다.
    - 그러나 context word 중에는 fabulous 가 있는데, 이는 meal aspect 에 대한 감성 분석을 나타내고 있는 것처럼 보이지 않는다.
    - 만일 감성 분류 모델이 aspect words 들을 구분하지 못한다면 특정 사용에 문제가 된다.
- 이러한 문제점을 해결하기 위해서 attention based model 이 많이 나타났는데, 공통적으로 문제점이 있다고 저자는 이야기한다.
- **첫 번째 문제**
    - 대부분의 이 분야의 attention modeling은 주어진 aspect 에 대한 각각의 context word의 공헌도와 관련성을 고려하지 않고 문장에 있는 context 정보의 일부분만을 고려한다.
    - 심각한 문제가 될 수 있는데, sentiment words 들과 shifters 들이 초점이 맞춰질 수도 있지만 전부다 주어진 aspect 와 관련이 있는건 아니기 때문.
    - 즉 context word 들이 주어진 aspect 에 대한 감성을 잘 잡아낼 수도 있지만 모든 context word 들이 잘 잡아내는 건 아니기 때문이다.
    - 예시
        - The mini's body hasn't changed since late 2010 - and for a good reason.
        - 주어진 aspect 는 `body` 인데, attention 모델을 통해서 `n't`, `good`, `late` 와 같은 단어들이 `focused words` (aspect의 감성을 설명해주는 단어를 이와 같이 표현하는 거 같다.) 포착될 수 있다.
        - 포착된 focused word 들은 부정(negative) 감정을 담고 있다.
        - 하지만 이에 대해서 이 단어들 중에 `good` 만이 진정 `body` aspect 의 감성 극성과 관련 있다고 이야기 할 수 있고, 올바르게 전달된 감성은 긍정(positive)이 되어야 한다고 말할 수 있다.
        - 본 논문의 연구에서는 `short-sighted behavior`이 분류기의 예측 정확도에 중대한 손실을 일으킨다고 주장한다.
- **두 번째 문제**
    - 현존하는 attention 모델들은 문장에 의해서 전달되어진 전반적인 의미를 계산하지 않고 `words-level` level 만을 고려한다.
    - 특정 사용자들에서 보이는 풍자적이거나 아이로닉한 주장들과 같은 복잡한 문장들을 위해서, 전체 문장에 관한 더 정확한 정보가 올바른 results 를 예측하기 위해서 필요하다.
    - 예시
        - Maybe the mac os improvement were not the product they want to offer
        - 이는 명백하게 `mac os` 제품에 대한 부정적인 리뷰를 담고 있는 아이로닉한 문장이지만, 대부분의 attention model 은 `improvement` 에 대해서 높은 가중치를 주게 되고 긍정(positive) 로 예측하는 경향을 보인다.
- **세 번째 문제**
    - 문장(sentence) 는 주어진 topic에 대해서 multiple aspect를 포함하고 있을 것이다.
    - 따라서 각 단어(나는 focused word, 또는 context word들로 이해했다.) 는 주어진 aspect에 따라서 다른 중요도를 문장 내에서 가질 수 있다.
    - 이전의 작업들도 물론 이 작업에 대해서 고려했는데 널리 펴져 있는 hidden state 기반, memory 기반의 모델을 이용해서 해결하려고 했다.

        > 1. Attention-based LSTM for aspect-level sentiment classification.

- **해결 방안**
  - **sentence level content attention mechanism**
    - 첫 번째 문제와 두 번째 문제를 해결하기 위해서 `sentence level content attention mechanism` 을 사용했다.
    - attention weight를 계산하기 위해서 문장 내에서 aspect 와 각각의 단어들에 의해서 전달된 정보들만 고려하지 않고 full sentence의 전체 의미를 고려할 것이다.
    - > Aspect 와 그 주변 단어의 의미만을 고려하지 않을 것이다. 문장의 의미도 고려할 것이다.
    - 이 모델의 이름을 sentence-level content attention moduls(SAM) 이라고 부른다.
    - SAM 모델은 전역적인 관점으로부터 주어진 aspect 에 대한 중요한 정보를 포착할 수 있다. 그리고 전체 문장을 output embedding vecor 로 임베드 할 수 있다.
    - 이 output embedding vector 는 aspect에 특화된 sentence representation 으로 여겨질 수 있다.
  - **context attention mechanism**
    - 세 번째 문제점을 해결하기 위해서 context attention mechanism 을 사용했다.
    - 단어 시퀀스의 순서만을 고려하는 게 아니라 단어들과 aspect간의 상관관계를 계산한다.
    - 이를 context attention based memory mudule의 약자인 CAM 으로 표현한다.
    - CAM 은 `customized memory` 를 각각의 aspect에 제공한다. 이는 시퀀셜한 방법에서 업데이트될 수 있다.
- Contribution
  - Content attention based aspect based sentiment classification(cabasc) 를 발전
  - 위에서 설명한 의미론적 mismatch 문제를 해결하기 위해서 두 개의 attention model mechanism 을 사용. 그리고 성능 평가
  - 역시나 SemEval 데이터 셋, twitter 데이터 셋 사용
  - 성능이 높아졌다.

> model frame work

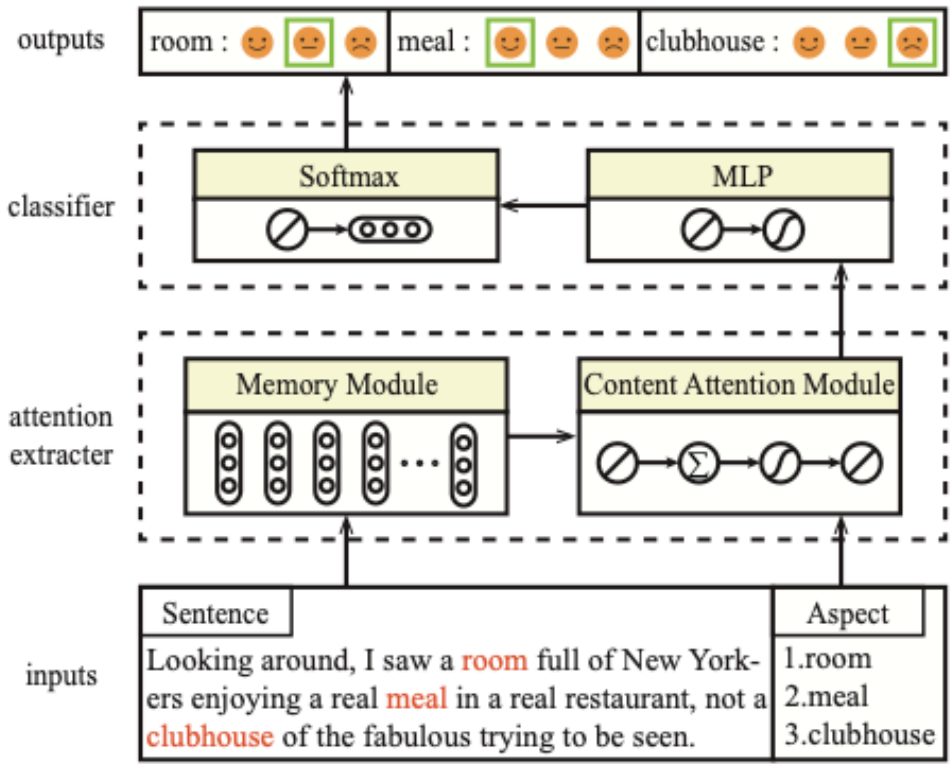


Figure 1: The framework of content attention based aspect based sentiment classification model.

Related work

ABSA 는 감성 분석의 기초적인 작업으로 몇 가지 subtask 들을 포함한다.

- aspect extraction
- opinion identification
- aspect based sentiment classification

> 논문의 저자들은 attention modeling을 aspect based sentiment classification 문제를 해결하기 위해서 발전시키는 것에 초점을 둔다.

Neural Network Models for ABSC Task

ABSC(Aspect based sentiment classification) 은 주어진 aspect 에 대한 감성 극성을 결정하는데 목적을 둔다.

요즘은 신경망 기반 작업을 시도한다.

신경망

- Recursive neural networks(RecNNs) 는 받아들일만한 회귀 뉴럴 네트워크를 주장한다. 이 모델은 타겟 단어에 대한 문맥 단어들의 감정을 선전할 수 있다.
    - RecNNs는 효과적인 방법이라고 설명되어져 왔다.
    - 그러나 syntax parsing 문제가 발생한다.
- RNNs(Recrrent neural networks)
    - 다양한 sequence learning task 에 효과적으로 작용한다고 알려져왔다.
    - 좀 괜찮은 것들은 대부분 RNN 을 기반으로 한다.
- TD-LSTM(target_dependent long short-term memory network)
    - Left and right 로부터의 표현을 학습할 수 있다.
- attention mechanism
    - Tang et al 이 사용을 했는데, 이 모델이 지금 논문에서 주장하는 것과 가장 유사하다.

Neural Attention Models for ABSC Task

- 위에서 언급한 것과 같아. ABSC 문제를 해결하기 위한 대부분의 뉴럴 네트워크는 문맥 단어들(context words)과 주어진 aspect 사이의 상관관계를 고려하지 않는다.
    - context words 들과 aspect 사이의 상관관계를 고려하기 위해서 다양한 attention mechanism 을 나타나고 있다.
- LSTM based single-hop attention model(Wang et al)
    - aspect 와 word embedding 들의 합침을 input 으로 간주한다.
    - 그리고 LSTM hidden states을 attention 을 계산하는데 사용한다.
- Interactive attention mechanism(Ma et al)(hidden state based attention models)
    - context 와 aspect 로부터 attention을 학습하는 상호적인 attention mechanism 을 제안한다.
    - Hidden state based attention models 로 부른다.
        - 논문의 저자들은 이러한 유형의 attention model 을 `memory based position-aware attention model` 이라고 칭한다.
- Tang et al
    - words 들과 주어진 aspect 의 관련 거리에 기반해서 attention 을 계산한다.
    - aspect 와 가깝게 위치한 word는 더 신뢰할만하다고 암묵적으로 주장할 수 있는데, 이는 논쟁이 있을만 하다.
    - ex)The two waitress's looked like they had been sucking on lemons
    - target aspect 인 `waitress` 에 대한 감성 극성을 얻고 싶을 때, `sucking on lemons` 가 aspect 와 더 가깝게 언급된 `like` 보다 더 중요하다.
    - 논문이 제안하는 context attention mechanism 은 context word 와 aspect 사이의 상관관계를 명시적으로 고려하면서 이 논란을 완화시키는 걸 돕는다.
- Chen et al
    - recurrent attention mechanism based on customized momory 를 주장한다.

- 멀리 떨어져 있어서 분리된 감성 특징들을 포착하기 위한 모델
- 이 부분은 현 논문이 제안하는 memory mechanism 과 유사하다.
- **하지만 LSTM 의 본래의 특성 때문에, 이 모델은 attention weight을 계산할 때 오직 문장의 부분 정보만을 고려한다.**
- 현 논문에서 제안하는 모델에서, 제안된 sentence-level content attnetion mechanism 은 이 memory model 의 `short-sight` 문제의 일반적이고 효과적인 해결책을 제공한다.

Model

> 먼저 세 개의 base line model 을 소개(위에서 언급된 메커니즘들의 정당성을 증명하는데 사용되어지는 base line model 들)
>
> training process 또한 이 section 에서 다룬다.

Baseline Model A (BaseA)

The baseline model A는 기초적인 모델이고, 그 다음 모델들은 이 모델로부터 도래된다.

- S 는 input sentence (N words들의 구성으로 이루어진)
- S_a {si, ……,s_(i+L)}는 input sentence 내에서 나타난 주어진 aspect 를 나타낸다.
  그리고 이는 L 개의 words로 구성된다.
- **우리 모델의 목표는 asepct S_a 에 대한 문장 S의 감성 극성을 예측하는 것.**

- **Inputs(논문에서 제안한 모델 아키텍처의 input 데이터에 대한 가공 방법 및 의미)**
  - L ∈ Rd×|V | => word embedding matrix
    - d = word vector 의 dimension,
    - v = vocabulary
  - matrix 자체는 비지도 학습으로 탄생 가능.
  - 문장 S내의 단어 s_i는 낮은 차원으로 그리고 실제 값 embedding e_i로 매핑된다.
  - 실제값 임베딩 ei ∈ Rd 는 아래와 같이 정의된다.
    - e_i = L_oi (matrix 의 i번째 단어의 원핫 인코딩을 의미하는 듯)
    - 여기서 o_i는 s_i의 원핫인코딩이다. 길이는 |V| 이다.

- 다음으로 문장 S에 해당하는 embedding representation 의 집합인 E를 얻게 된다.
- 만일 aspect S_a가 단일 단어이면, aspect representation v_a는 aspect word의 embedding이다.
- 만일 S_a가 문구라면, v_a는 aspect word embedding의 평균값을 가진다.

- **Memory Module**
  - E들의 vectors들은 장기 기억 메모리(long term memory), M = {m1,m2,....m_N} 를 구성하기 위해서 하나씩 쌓인다.
  - 여기서 M은 R(dxN) 에 속한다.
  - M은 input sentence의 정보를 저장하고 있고, 여기서 memory의 content들은 학습가능한 변수들로 여겨지고, 모델은 어떻게 예측을 위해서 memory를 사용할지 배운다.

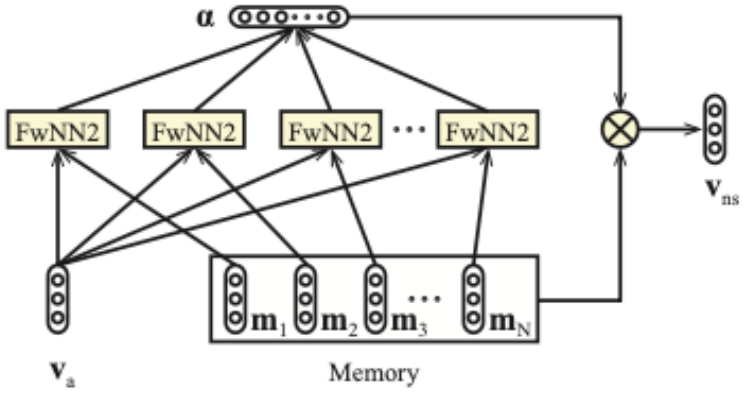
- **Content Attention Module**
  - 직관적으로 문장 내에서 일어난 주어진 aspect의 감성 극성에 대해 문장 내에 있는 단어들은 각각 다르게 기여를 한다.
  - 가장 관련성 있는 정보를 파악하기 위해서 문장 내의 가장 중요한 위치의 집합을 찾기 위해서 Bahdanau et al.은 일렬로 attention model 을 늘어놓은 모델을 사용한다.
  - 논문의 저자는 여기에 영감을 받아서 그림과 같은 구성의 content attnetion modul을 사용한다.
    - 

Figure 2: The content attention module of BaseA model.

    - 위에서 구성한 M 메모리로부터 주어진 aspect S_a와 가장 관련있는 감성 극성을 가진 정보를 검색하기 위해서
  - M matrix 의 메모리 slice m_i의 attention weight을 계산하기 위해서, 2개의 inputs을 사용한 피드 포워드 네트워크(FwNN2)를 aspect S_a의 감성 극성에 단어 s_i 가 얼마나 중요한지 점수 짓기 위해서 사용한다.
  - 점수는 m_i와 aspect representation v_a를 기반으로 하여 계산되어지고 공식적으로 다음과 같이 계산이 된다.
    - $$c_i = \mathbf{W}_1 \tanh(\mathbf{W}_2\mathbf{m}_i + \mathbf{W}_3\mathbf{v}_a + \mathbf{b}_1)$$
    - 여기서 모델 파라미터의 shape들은 다음과 같다.
      - W1 ∈ R1×d , W2 ∈ Rd×d, W3 ∈ Rd×d, vector b1 ∈ Rd ()
    - attention weight s_i 의 a_i는 다음과 같이 계산된다.

        - $\alpha_i = \frac{\exp(c_i)}{\sum_{j=1}^{N} \exp(c_j)}$
        - 모든 계산된 attention weights들은 주어진 aspect S_a와 함께한 memory M의 attention weight vector를 형성한다.
        - attention weight vector a = (a1,a2....a_N)
    - 최종적으로 aspect 특화적인 sentence representation v_ns는 아래와 같이 계산되어진다.
        - $\mathbf{v}_{ns} = \mathbf{M}\boldsymbol{\alpha}$
        - 여기서 v_ns 의 shape 은 vns ∈ Rd

- **MLP**
    - 깊이는 딥러닝 방법의 중요한 요소이다.
    - 멀티플 비선형 layers 들은 더 추상적이고 유용한 표현들을 얻게끔 돕는다.
    - 논문의 저자들은 모델의 깊이를 간단하고 효과적이게 증가시킬 수 있는 one hidden layer 를 가진 MLP 를 사용한다.
    - MLP 는 aspect 특화적인 문장 representation 인 V_ns를 input으로 이용한다.
    - 아래와 같이 정의된다.
    - $\mathbf{v}_{ms} = g(\mathbf{W}_4\mathbf{v}_{ns} + \mathbf{b}_2)$
    - 여기서 v_ms 의 shape 은 vms ∈ Rd 이며, MLP 모듈의 output이다.
    - W4 ∈ Rd×d 은 weight matrix
    - b2 ∈ Rd bias vector 이다.
    - g(.) 는 비선형 활성화 함수이고 여기에 tanh 함수를 사용한다.
- **Softmax**
    - MLP 의 output v_ms는 선형 함수에 의해서 길이 |C|의 실제 값 벡터로 바뀐다.
    - 여기서 C는 가능한 감성 카테고리들을 의미한다.
    - aspect S_a의 감성 극성을 예측하기 위해서 softmax layer에 vector가 입력 된다.
    - $$pred = \text{softmax}(\mathbf{W}_5\mathbf{v}_{ms} + \mathbf{b}_3)$$
    - pred ∈ R|C | 는 C 전체에 대한 조건 확률 분포다.
        - 선형 layer 에 대한 parameters 들.
        - W5 ∈ R|C |×d
        - b3 ∈ R|C |

Baseline Model B(Base B)

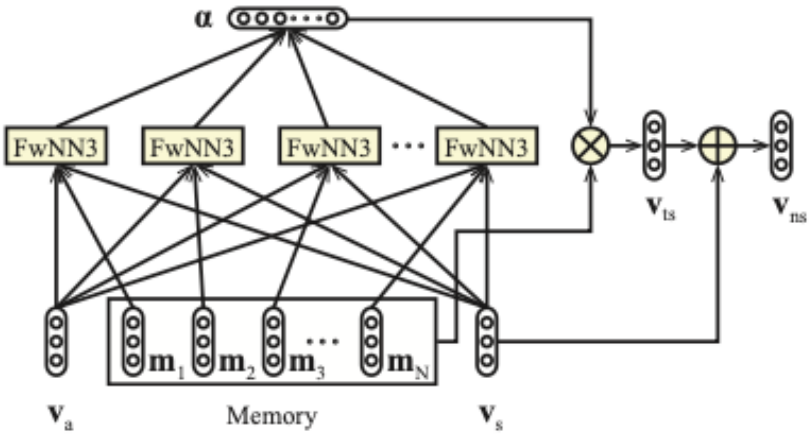


Figure 3: The sentence-level content attention module of BaseB model.

- BaseA의 Content attention module 은 attnetion weight 을 문장 전체에 대한 전체 의미를 고려하지 않고 계산한다.
- 오직 문장 정보의 부분만을 고려하는 것은 주어진 aspect 에 대해서 관련이 없는 몇몇의 focused words로 향할 수 있다.
  => 이는 모델의 감성 극성 예측에 영향을 끼칠 것이다.
- 복잡한 문장들을 위해서, 문장에 의해서 전달된 전체적인 의미를 가지지 않은 content attention module에 의해서 생산되어진 aspect 특화적인 sentence representation은 감성 분류에 적합하지 않을 수 있다.
- 이러한 문제를 해결하기 위해서 BaseA 를 발전시킨 sentence information의 사용을 제안한다.
- content attention module에 두 개를 확장한 새로운 모델 BaseB 는 fig3 에 보인다.

> sentence-level content attention mechanism 을 사용한

- Sentence-level Content Attention Module(SAM)
  - **첫 번째 연장**은 Attention weight의 계산에 문장 representation을 추가한다.
  - 문장 representation 은 전역적인 관점으로부터의 aspect 의 감성 극성을 위한 단어의 중요도 점수를 smoothing 한다. (중요한 감성 특징들을 더 정확하게 포착하는 능력을 증진시키기 위해서)
  - 세 개의 inputs 을 가지고 있는 피드 포워드 뉴럴 네트워크(FwNN3)는 단어 s_i의 점수 c_i를 계산하는데 사용되어진다.

$$c_i = \mathbf{W}_6 \tanh(\mathbf{W}_7 \mathbf{m}_i + \mathbf{W}_8 \mathbf{v}_a + \mathbf{W}_9 \mathbf{v}_s + \mathbf{b}_4)$$

  - 여기서 m_i는 m_i ∈ Rd 이고 이건, s_i(단어)의 memory slice다.
  - v_a ∈ Rd 는 aspect representation 이다.
  - v_s ∈ Rd 는 sentence representation 이다.
  - W6 ∈ R1×d, W7 ∈ Rd×d, W8 ∈ Rd×d, W9 ∈ Rd×d 는 weight matrices
  - b4 ∈ Rd 는 bias vector

        - **논문의 저자들은 문장의 정보를 보존하기 위해서 문장 내에서 문장의 표현으로서 참여하고 있는 단어 임베딩의 평균을 취한다. => 그리고 이것은 놀라울 정도로 효과가 좋은 것으로 증명된다.**
        - 모든 계산된 문장에 대한 점수들은 {c1, c2 …., c_N} 으로 표현된다.
        - 그리고 문장 S의 attention weight vector a(a1,a2…..,a_n) 원소를 계산하기 위해서 Eq.(3) 이 사용된다.
        - $$\alpha_i = \frac{\exp(c_i)}{\sum_{j=1}^{N} \exp(c_j)} \qquad (3)$$
        - output embedding 벡터 v_ts는 아래와 같이 계산되어진다.
        - $$\mathbf{v}_{ts} = \mathbf{M}\boldsymbol{\alpha}$$
        - 여기서 v_ts는 shape을 vts ∈Rd 가진다.
        - M은 memory module에서 만들어진 memory 다.
    - **두 번째 연장**은 전체 문장을 모델이 복잡한 문장들을 다루는 능력을 향상시켜줄 수 있는 output vector로 임베딩하는 것이다.
    - 이를 문장 표현 v_s를 output 벡터 v_ts에 추가하면서 성취했다.
    - $$\mathbf{v}_{ns} = \mathbf{v}_{ts} + \mathbf{v}_s$$
    -
        - 여기서 v_ns 는 aspect 특화적인 문장 표현이다. Sentence level content attention module의 output으로서, 이건 알아차려진 감성 특징들은 강조되어져 있고 문장 정보들은 embedded 되어 있다.

Baselin e Model C (BaseC)

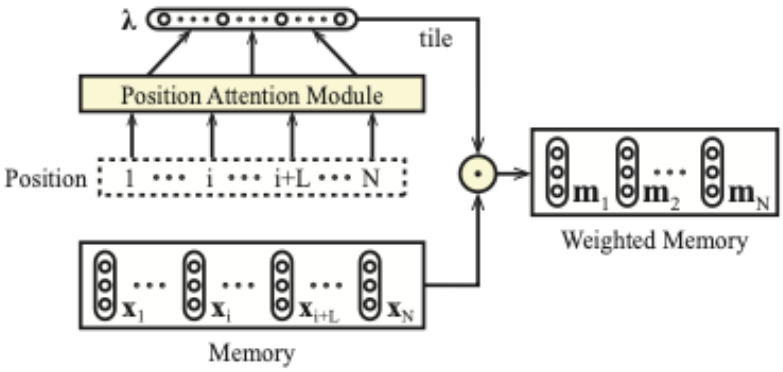


Figure 4: The position attention based memory module of BaseC model.

- 똑같은 단어는 memory module 에 의해서 생성된 똑같은 memory slice를 가진다.

- 하지만, 단어들의 의미의 다양성을 고려하는 것 때문에, 다른 맥락 안에서는 다른 감성 극성을 가리킬 것이고 문장은 주어진 토픽의 다중 aspects들을 포함할 것이다.
- 따라서 문장 내에서 논의되고 있는 aspect 에 따라서 각각의 단어들은 다른 중요도를 가질 것이다.
- > 결국 context 에 따라서 각각의 word들의 의미 및 중요도는 달라진다.

- 위와 같은 문제들을 해결하기 위해서 논문의 저자들은 BaseB의 memory module 을 확장시키기 위한 position attention mechanism 을 사용한다.
  - 이를 통해서 position attention based memory module 이 되려고 하고, 아래 그림에서 보여지는 것과 같이 문장의 주어진 aspect에 대한 `customized memory` 를 만든다.
  - > 단순히 memory module 을 사용해서 단어의 표현을 얻는 게 아니라 주어진 aspect 에 따른 customized memory 를 만든다는 의미인듯
  - position attention mechanism 의 숨겨진 직관은 **aspect 주변의 단어들이 aspect의 감성 극성에서 더 중요한 영향력**을 가진다는 것이다. => BaseC

- Position Attention Based Memory Module
  - 문맥 단어(context word) 의 position은 문장 내에서 그것과 aspect 와의 절대적인 거리로 정의된다.
  - 그리고 aspect word의 위치는 0 으로 여겨진다.
  - 만일 aspect 가 구(phrase) 인 경우, 좌측 문맥 단어(left context words)의 positions 들은 aspect의 첫 번째 단어로 계산된다. 반면에 우측 문맥 단어(right context words)들은 마지막 단어로 계산된다.
  - $\lambda \in RN$ 을 문장 S에 대한 **position weight 벡터**로 여기자.
    - i-th 째 벡터의 요소는 다음과 같이 계산되어진다.
    - $$\lambda_i = 1 - p_i/N$$
    - 여기서 p_i는 sentence S에 속해있는 단어 s_i의 위치라고 할 수 있다.
    - 그리고 N은 문장의 길이(length)다.
  - memory M은 weighted memory M_w = (m_w1, m_w2, ...m_wN) 을 만들기 위해 position attention weights에 의해서 가중된다.
    - memory slice $m_{wi} \in R_d$ 아래와 같이 계산된다.
    - $$\mathbf{m}_{wi} = \mathbf{q}_i \odot \mathbf{m}_i$$
    - 여기서 m_i의 shape 은 $m_i \in R_d$ 이고 M안에 있는 s_i의 memory slice이다.
    - q_i 의 shape 은 $q_i \in R_d$ 이고, λi 를 타일링하여 얻어진 벡터이다. 이것은 q_i벡터를 만들기 위해서 λi를 d 번 복사한다
    - $\odot$ 은 요소별 곱셉을 의미한다.
    - M_w은 sentence level attention module 에 넣어진다.

Content Attention Based Aspect Based Sentiment Classification Model(Cabasc)

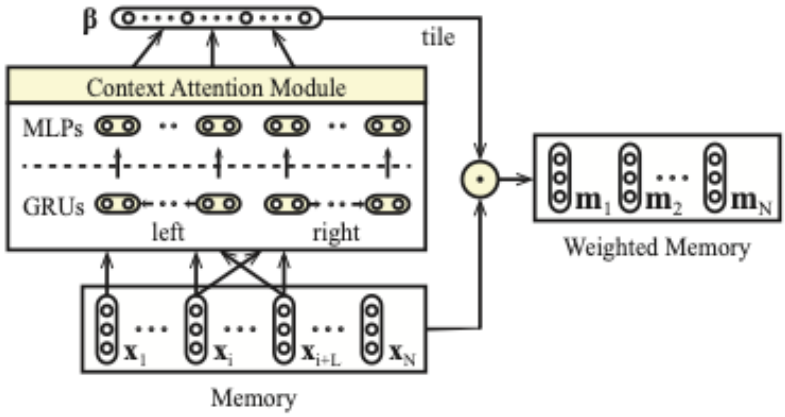


Figure 5: The context attention based memory module of Cabasc model.

- 앞서 설명한 position attention weight 은 문맥 단어(context word) 와 aspect 단어와의 포지션만을 고려해서 계산하고 문맥 단어와 aspect 사이의 **상관관계**는 무시한다. 이 상관관계는 가까운 position 에 대한 정보보다 더 중요한 정보다.
- 이러한 문제점 때문에, 논문의 저자들은 context attention mechanism 을 제안한다. 이 모델은 단어 순서 정보, aspect 정보 그리고 문맥 단어와 aspect 사이의 상관관계들이 계산된 attention weight 에 모델링 된다.
- Cabasc 은 메모리 모듈을 제외하고 BaseC 모델과 동일하다.
  - Cabasc 의 메모리 모듈은 context attention based momory module 이다. (위의 그림과 같은)
  - 이는 BaseC에서 사용된 position attention based momoery module 과는 다르다
  - > Cabasc 모듈이 position attention mechanism 을 사용하지 않고 context attention 메커니즘을 사용한다는 점에서 다르다.

- Context Attention Based Memory Module(CAM)
  -

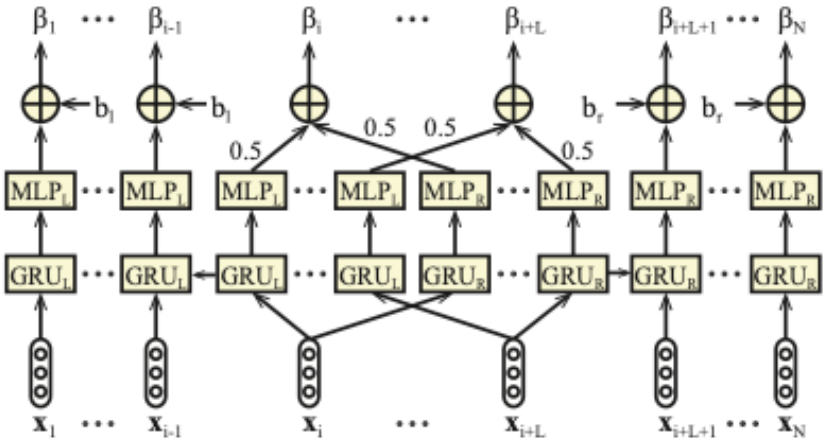


Figure 6: The context attention mechanism.

- 첫 째로 input 문장 S에 대해서 aspect part S_ls = {s1, . . . , s_i−1,s_i,...,s_i+L}를 가지고 있는 left context 와 aspect part S_rs ={s_i, ...,s_i+L,s_i+L+1,...,s_N } 를 가지고 있는 right context 로 나눈다.
- 이 두 부분의 단어들의 임베딩들은 matrix L과 연관이 있고, S_ls 와 S_r s 에 해당하는 두 개의 벡터 리스트들 Els = {e1,...,ei−1,ei,...,ei+L} and Ers = {ei,...,ei+L,ei+L+1, . . . , eN } 은 각각 얻어진다.
- 문맥 단어(context word) 와 aspect 간의 문맥 정보를 모델링하기 위해서, 우리는 두 개의 GRU neural 을 사용한다.
  - left = GRU_L, Right = GRU_R
  - E_ls와 E_rs를 각각 모델링한다.
  - GRU_L의 input은 E_ls이고, GRU_L을 오른쪽에서 왼쪽으로 운영한다.
  - time step t에서, GRU_L은 E_ls의 요소 e_t를 관찰한다.
  - hidden state 를 아래와 같이 업데이터한다.

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{e}_t + \mathbf{U}_r \mathbf{h}_{t-1})$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{e}_t + \mathbf{U}_z \mathbf{h}_{t-1})$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h \mathbf{e}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (\mathbf{1} - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t$$

  - Wr ∈ Rd×d
  - Ur ∈ Rd×d
  - Wz ∈ Rd×d
  - Uz ∈ Rd×d
  - Wh ∈Rd ×d
  - Uh ∈ Rd ×d
  - 이 모두 weighted matrix
  - σ 는 logistic sigmoid function 을 나타낸다.
- gate r_t의 업데이트는 새로운 hidden state h_t로부터의 output의 업데이트 범위를 조정해준다.
- gate z_t를 갱신하는 것은 이전 hidden state로부터 정보를 얼마나 허용할지를 제어한다.
- E_ls 를 다 읽게 되면, GRU_L 은 hidden state list H_ls를 생산한다.
  - H_ls = {hi +Ll , ...,hi ,hi−1,...,h1}

- GRU_R 도 같은 일을 하지만 E_rs를 input으로 필요로한다.(GRU_L 은 E_ls를 필요로 했다.)
  - H_rs ={hir,...,hi+Lr,hi+L+1,...,hN}.

- MLP 는 h_l의 attention weight β_l 을 계산한다.
  - 여기서 h_l은 H_ls 의 원소이다.
  - $$\beta_l = \sigma(\mathbf{W}_{10}\mathbf{h}_l + b_5) + b_l$$
  - 여기서 W_10 의 shape 은 W10 ∈ R1×d 이고 이는 weight matrix이다.
  - b5 ∈R
  - bl ∈R, 기초 attention weight 이다.
  - H_ls의 원소를 input으로 취하는 MLP는 MLP_L 로 불린다.
    - MLP_L은 파라미터들을 공유한다.
    - Reverse attention weight list for H_ls 는 β_ls ={β1,...,βi−1,βil,...,βi+Ll} 이다.

      > reverse 를 하는 이유를 GRU_L은 오른쪽에서 왼쪽으로 학습을 하면서 hidden state 를 만들어내니까 그런가 싶다.

- MLP_RS 에 의해서 H_rs 의 βrs = {βir ,...,βi+Lr ,βi+L+1,..., βN } 가 계산되어진다. 계산 식은 다음과 같다.
  - $\beta_r = \sigma(\mathbf{W}_{11}\mathbf{h}_r + b_6) + b_r$
  - W11 ∈R1×d 이고,
  - b6 ∈R
  - hr ∈Hrs , and br ∈R 은 기초적인 attention weight
- > 정리하자면 input 데이터에 대해서 context 를 왼쪽 부분 오른쪽 부분으로 나누었으며, 각각 GRU_L 과 GRU_R 을 통해서 hidden state를 얻게 되고 이를 각각 단어에 대한 표현으로 사용한다.
  >
  > 사용된 hidden state 에 가중치를 두는 작업이 필요한데 이때 MLP 를 사용한다.
  >
  > 왼쪽 맥락에 대해서 가중치를 두는 식이 존재하고, 이를 통해서 왼쪽 맥락의 각각의 hidden state에 대한 가중치를 얻는다.(β_ls)
  >
  > β_rs 도 마찬가지다.

- β_ls를 왼쪽 context 에 해당하는 β__lc를 만드는데 사용한다.

  β_rs를 오른쪽 context 에 해당하는 β__rc를 만드는데 사용한다.
- β_a 는 aspect 에 해당하는 attention weights들이다.
  - 이 중의 하나의 원소는 다음과 같이 계산된다.
  - $\beta_k = (\beta_{k_l} + \beta_{k_r}) \times 0.5$
  - i≤k≤i+L,βkl ∈βls and βkr ∈βrs.

- β lc + β rc + β a를 concatenate 해서 context attention weight β = {β1, β2, . . . , βN } 를 얻는다.

- weighted memory Mw = (mw1,mw2,...,mwN) 는 β 에 기반해서 계산되어진다.

$$\mathbf{m}_{wi} = \mathbf{y}_i \odot \mathbf{m}_i$$

    - 여기서 m_i는 기존 M의 slice이고 y_i는 shape이 yi ∈ Rd 이고 β_i를 tiling하여 얻어진 벡터이다. (β i ∈β)
    - 논문의 저자들은 문장의 표현을 M_w의 평균으로 취한다.

Model Training

$$loss = -\sum_i \log p_{t_i}$$

논문의 모델은 교차 손실 함수(cross entropy loss) ㅁ를 지도학습 방식으로 줄이면서 학습을 한다.

(loss function 은 위와 같다.)

Experiments and Discussion

이후부터는 그냥 읽음