어휘 강화 임베딩을 위한 멀티채널 CNN과 결합된 계층형 집중 네트워크 기반 감성분류기

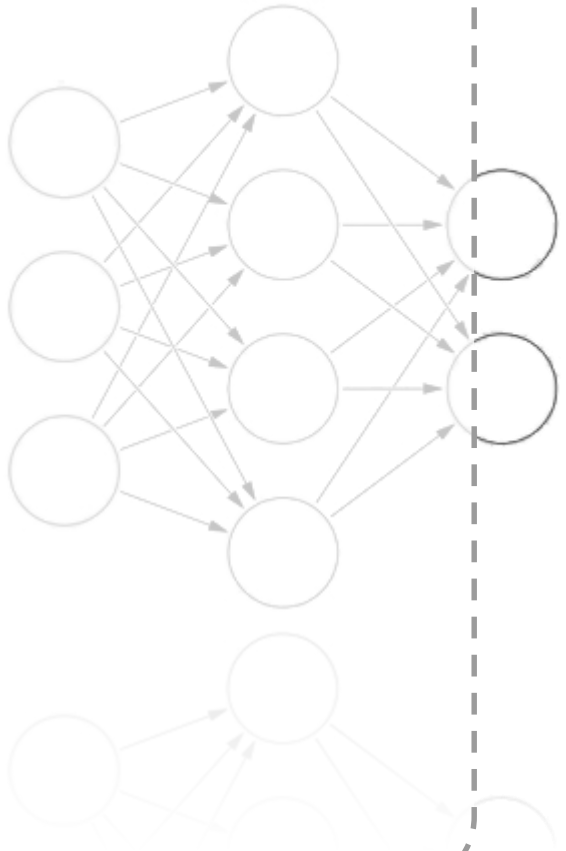
Sentiment Classifier based on Hierarchical Attention Networks Combined with Multichannel CNNs for Lexicon-Enhanced Embedding

Korea University
INI Lab
2016010662 윤주성

MS Thesis Defense

Index

- Overview
- Introduction
- Related work
- Model
- Experiments
- Conclusion & Future work
- QnA



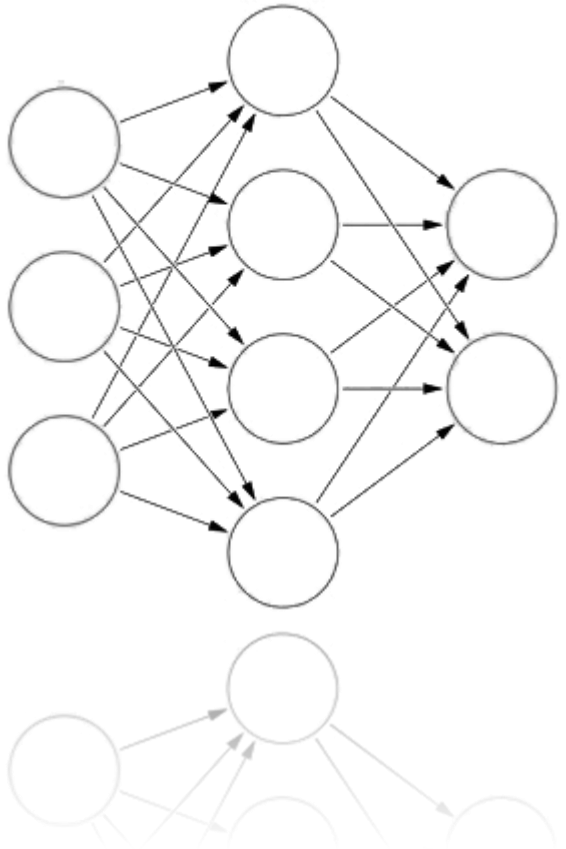
Overview

딥러닝 & 도메인 어떤 분야가 좋을까?

감성분석을 잘 할 수 있는 방법이 뭘까?

성능도 좋으면서 결과를 유추할 수 있는 방법이 뭘까?

SOTA를 넘으려면 어떻게 해야할까?



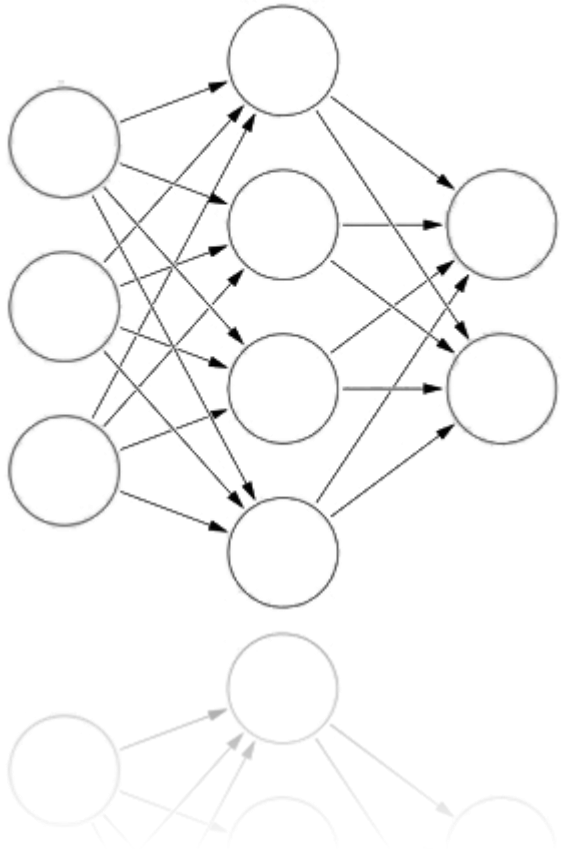
Overview

DL for NLP, 감성분석 (SemEval 2017, 트위터)

감성분석을 잘 할 수 있는 방법이 뭘까?

성능도 좋으면서 결과를 유추할 수 있는 방법이 뭘까?

SOTA를 넘으려면 어떻게 해야할까?



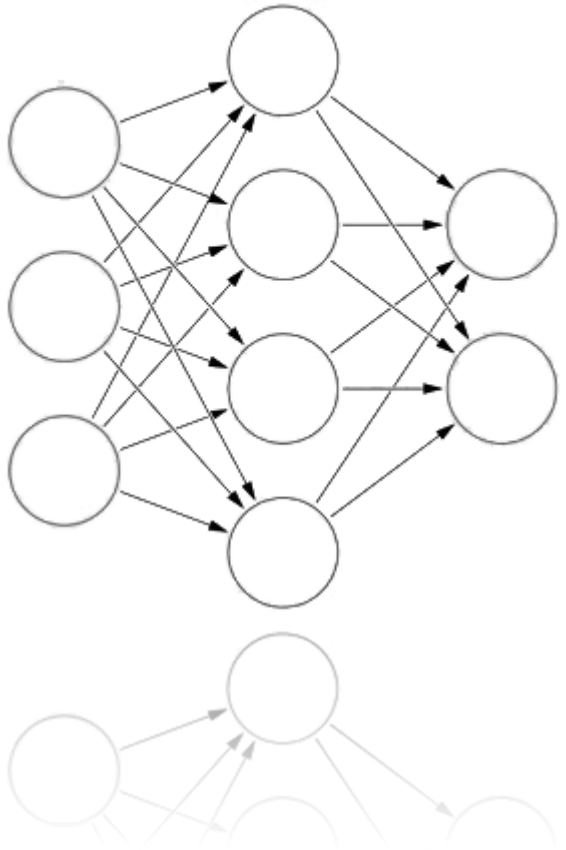
Overview

DL for NLP, 감성분석 (SemEval 2017, 트위터)

Lexicon Integrated CNN에 관한 연구

성능도 좋으면서 결과를 유추할 수 있는 방법이 뭘까?

SOTA를 넘으려면 어떻게 해야할까?



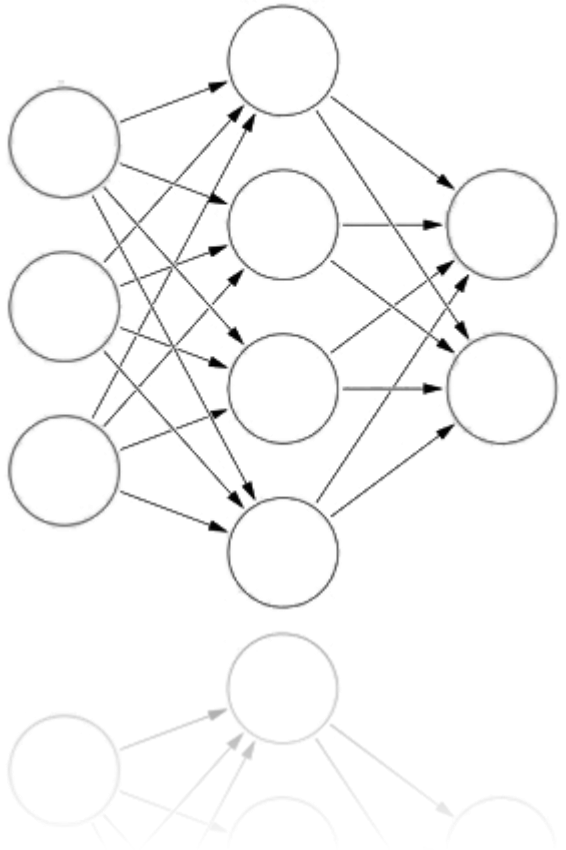
Overview

DL for NLP, 감성분석 (SemEval 2017, 트위터)

Lexicon Integrated CNN에 관한 연구

Hierarchical Attention Network에 관한 연구

SOTA를 넘으려면 어떻게 해야할까?



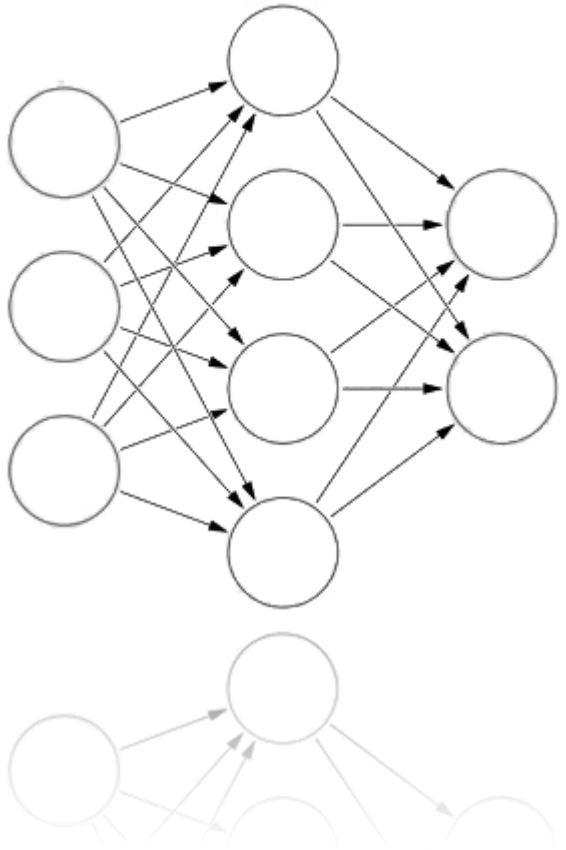
Overview

DL for NLP, 감성분석 (SemEval 2017, 트위터)

Lexicon Integrated CNN에 관한 연구

Hierarchical Attention Network에 관한 연구

두 모델의 장점을 합친 모델 제안, SOTA 성능 기록



Introduction

연구의 필요성

소셜 미디어의 발전

의견을 담고 있는 문서가 많아짐

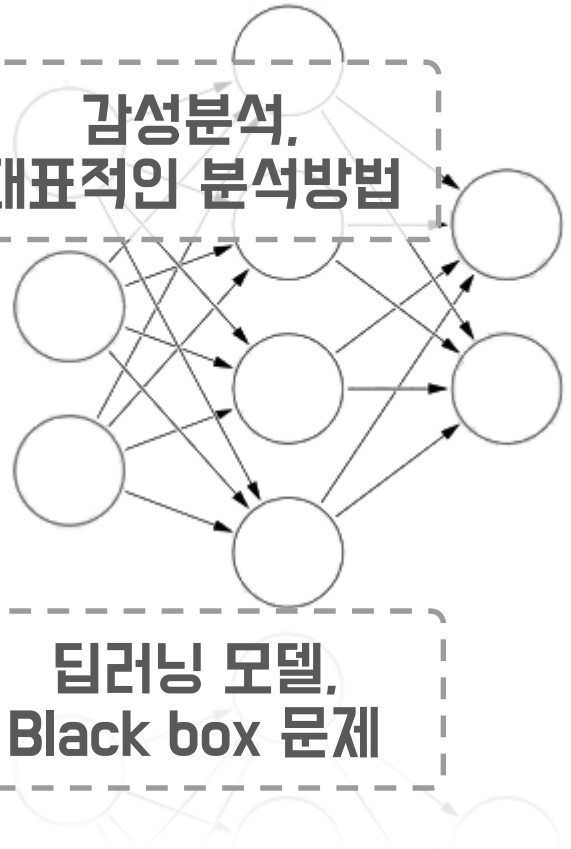
분석을 통한 트렌드 이해

감성분석, 대표적인 분석방법

기존 ML 기법의 한계

Deep Learning (e.g. textCNN)

CNN 기반 딥러닝 기법의 한계



Word2vec같은 임베딩에 의존적

문서의 구조적정보, 장기 의존성 고려X

중요도가 다른 단어 도 동일하게 처리

딥러닝 모델, Black box 문제

Introduction

연구의 필요성

소셜 미디어의 발전

의견을 담고 있는 문서가 많아짐

분석을 통한 트렌드 이해

감성분석, 대표적인 분석방법

기존 ML 기법의 한계

Deep Learning (e.g. textCNN)

CNN 기반 딥러닝 기법의 한계

Word2vec같은 임베딩에 의존적

문서의 구조적정보, 장기 의존성 고려X

중요도가 다른 단어 도 동일하게 처리

딥러닝 모델, Black box 문제

Introduction

연구의 필요성

소셜 미디어의 발전

의견을 담고 있는 문서가 많아짐

분석을 통한 트렌드 이해

감성분석, 대표적인 분석방법

기존 ML 기법의 한계

Deep Learning (e.g. textCNN)

CNN 기반 딥러닝 기법의 한계

Word2vec같은 임베딩에 의존적

문서의 구조적정보, 장기 의존성 고려X

중요도가 다른 단어 도 동일하게 처리

딥러닝 모델, Black box 문제

Introduction

연구의 필요성

소셜 미디어의 발전

의견을 담고 있는 문서가 많아짐

분석을 통한 트렌드 이해

감성분석, 대표적인 분석방법

기존 ML 기법의 한계

Deep Learning (e.g. textCNN)

CNN 기반 딥러닝 기법의 한계

Word2vec같은 임베딩에 의존적

문서의 구조적정보, 장기 의존성 고려X

중요도가 다른 단어 도 동일하게 처리

딥러닝 모델, Black box 문제

Introduction

연구의 필요성

소셜 미디어의 발전

의견을 담고 있는 문서가 많아짐

분석을 통한 트렌드 이해

감성분석, 대표적인 분석방법

기존 ML 기법의 한계

Deep Learning (e.g. textCNN)

CNN 기반 딥러닝 기법의 한계

Word2vec같은 임베딩에 의존적

문서의 구조적정보, 장기 의존성 고려X

중요도가 다른 단어 도 동일하게 처리

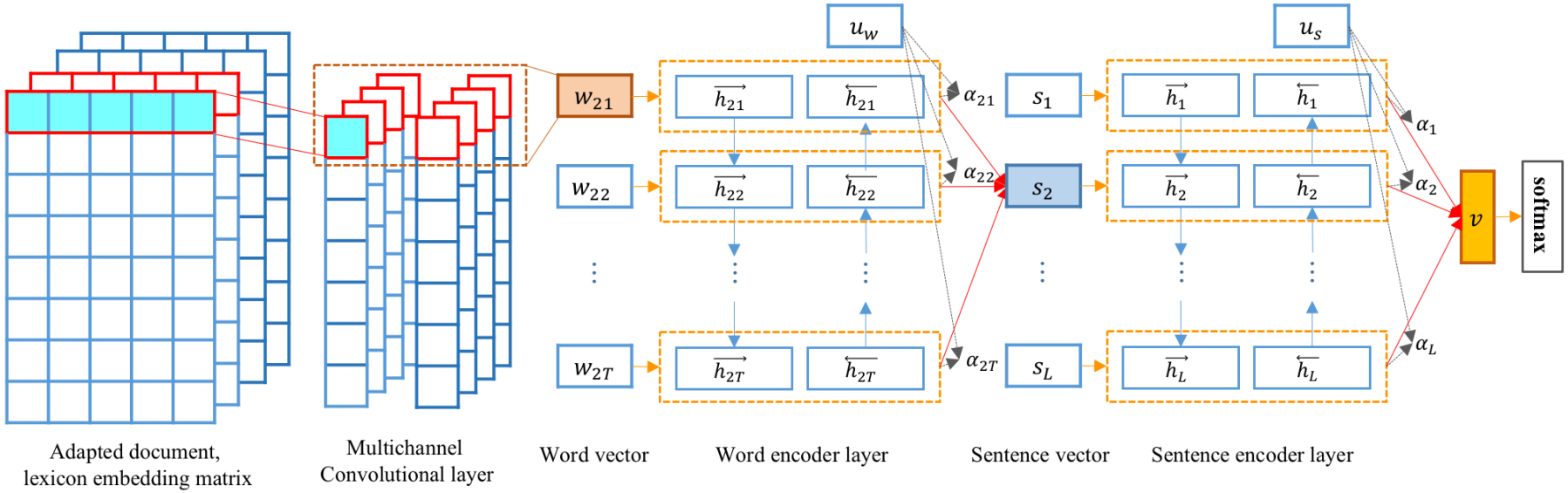
딥러닝 모델, Black box 문제

이러한 문제들, 어떻게 해결하면 좋을까?

Introduction

연구의 내용

트위터 데이터를 대상으로 기존 문제를 해결하기 위한 딥러닝 기반 감성분석 모델 제안

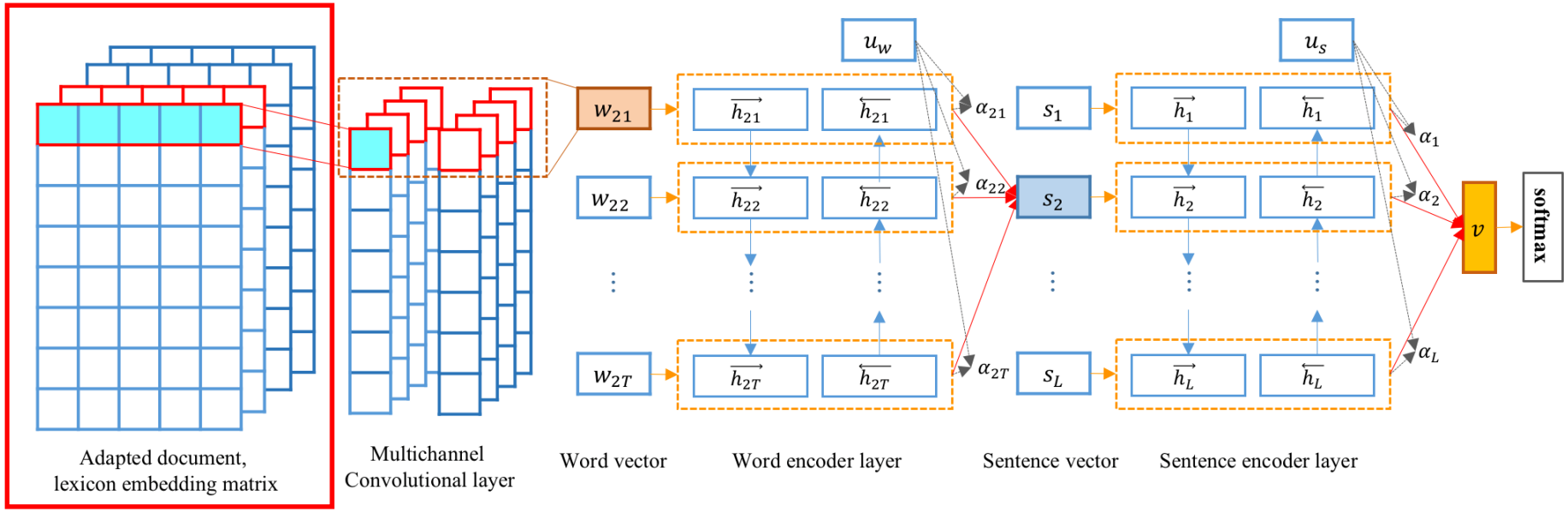


어휘 강화 임베딩을 위한 멀티채널 CNN과 결합된 계층형 집중 네트워크 기반 감성분류기

Introduction

연구의 내용

트위터 데이터를 대상으로 기존 문제를 해결하기 위한 딥러닝 기반 감성분석 모델 제안

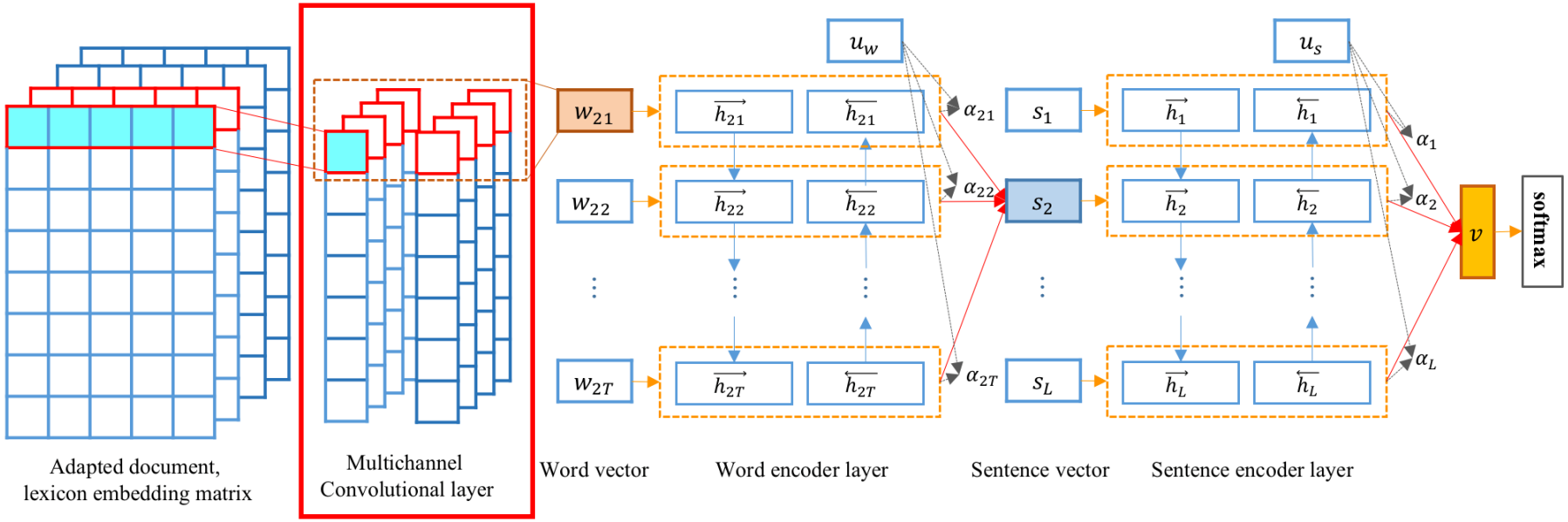


어휘 강화 임베딩을 위한 멀티채널 CNN과 결합된 계층형 집중 네트워크 기반 감성분류기

Introduction

연구의 내용

트위터 데이터를 대상으로 기존 문제를 해결하기 위한 딥러닝 기반 감성분석 모델 제안

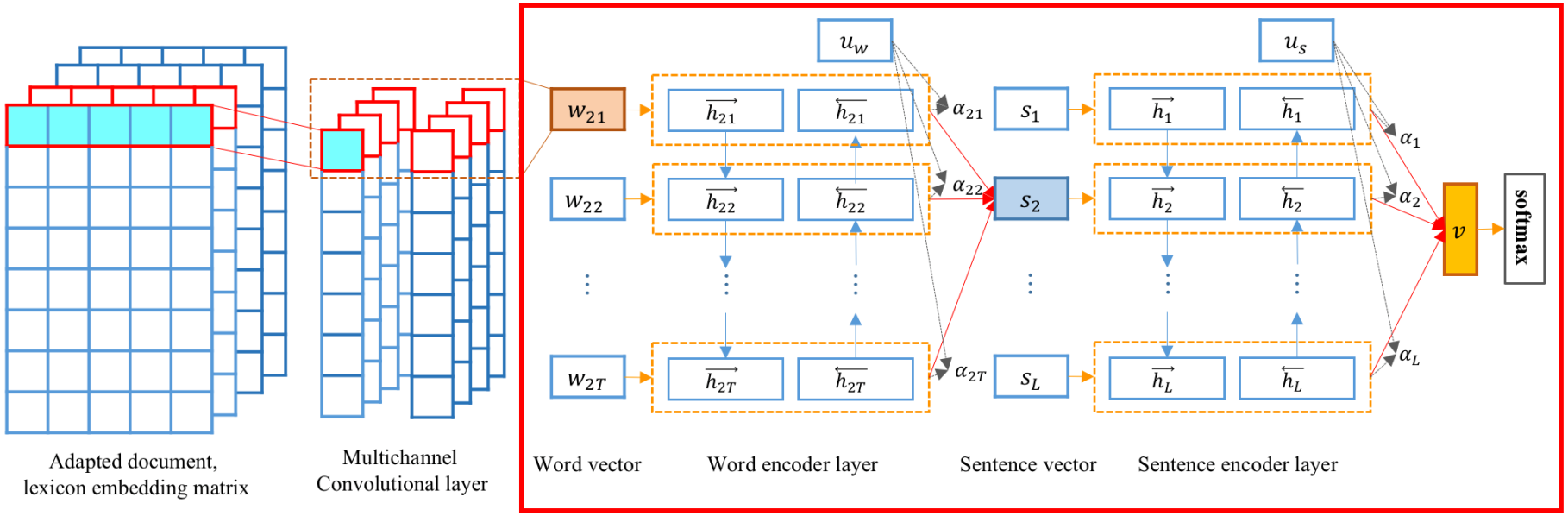


어휘 강화 임베딩을 위한 **멀티채널 CNN**과 결합된 계층형 집중 네트워크 기반 감성분류기

Introduction

연구의 내용

트위터 데이터를 대상으로 기존 문제를 해결하기 위한 딥러닝 기반 감성분석 모델 제안



어휘 강화 임베딩을 위한 멀티채널 CNN과 결합된 **계층형 집중 네트워크** 기반 감성분류기

Introduction

연구의 내용

트위터 데이터를 대상으로 기존 문제를 해결하기 위한 딥러닝 기반 감성분석 모델 제안

Pre-training +) lexicon emb	단어 레벨 GRU 문장 레벨 GRU	Attention mechanism	Attention 시각화
Word2vec같은 임베딩에 의존적	문서의 구조적정보, 장기 의존성 고려X	중요도가 다른 단어 도 동일하게 처리	딥러닝 모델, Black box 문제

Introduction

연구의 내용

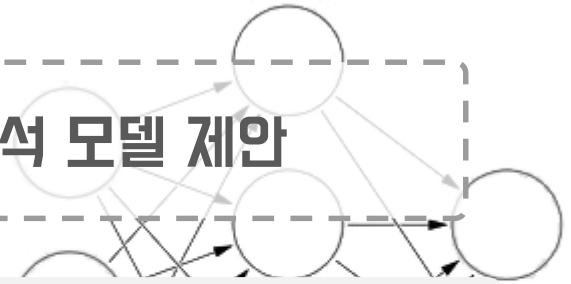
트위터 데이터를 대상으로 기존 문제를 해결하기 위한 딥러닝 기반 감성분석 모델 제안

Pre-training +) lexicon emb	단어 레벨 GRU 문장 레벨 GRU	Attention mechanism	Attention 시각화
Word2vec같은 임베딩에 의존적	문서의 구조적정보, 장기 의존성 고려X	중요도가 다른 단어 도 동일하게 처리	딥러닝 모델, Black box 문제

Introduction

연구의 내용

트위터 데이터를 대상으로 기존 문제를 해결하기 위한 딥러닝 기반 감성분석 모델 제안



Pre-training +) lexicon emb	단어 레벨 GRU 문장 레벨 GRU	Attention mechanism	Attention 시각화
Word2vec같은 임베딩에 의존적	문서의 구조적정보, 장기 의존성 고려X	중요도가 다른 단어 도 동일하게 처리	딥러닝 모델, Black box 문제

Introduction

연구의 내용

트위터 데이터를 대상으로 기존 문제를 해결하기 위한 딥러닝 기반 감성분석 모델 제안

Pre-training +) lexicon emb	단어 레벨 GRU 문장 레벨 GRU	Attention mechanism	Attention 시각화
Word2vec같은 임베딩에 의존적	문서의 구조적정보, 장기 의존성 고려X	중요도가 다른 단어 도 동일하게 처리	딥러닝 모델, Black box 문제

Introduction

연구의 내용

트위터 데이터를 대상으로 기존 문제를 해결하기 위한 딥러닝 기반 감성분석 모델 제안

Pre-training +) lexicon emb	단어 레벨 GRU 문장 레벨 GRU	Attention mechanism	Attention 시각화
Word2vec같은 임베딩에 의존적	문서의 구조적정보, 장기 의존성 고려X	중요도가 다른 단어 도 동일하게 처리	딥러닝 모델, Black box 문제

Related Work

Convolutional Neural Network

CNN은 간단하면서도 효과적인 모델 구조를 가짐
Filter의 window size에 따라 n-gram 자질 추출 가능
Text Classification 분야에서 성공적인 성과를 보임(Kim, 2014)

Convolutional neural networks for sentence classification

Y Kim - arXiv preprint arXiv:1408.5882, 2014 - arxiv.org

Abstract: We report on a series of experiments with convolutional neural networks (CNN) trained on top of pre-trained word vectors for sentence-level classification tasks. We show that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks. Learning task-specific vectors through fine-tuning offers further gains in performance. We additionally propose a simple modification to the …

☆ 99 1383회 인용 관련 학술자료 전체 16개의 버전

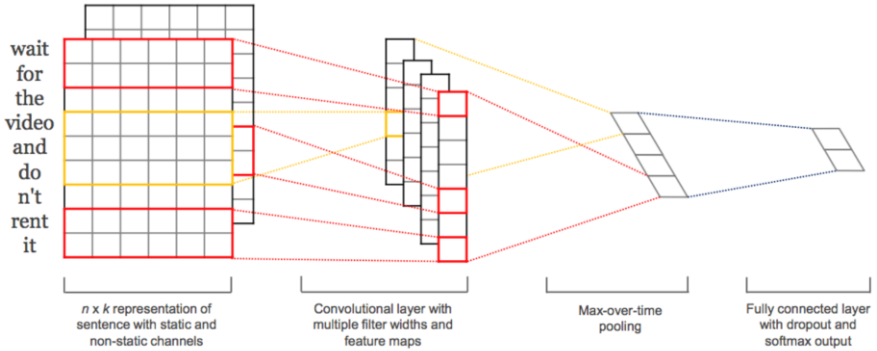


Figure 1: Model architecture with two channels for an example sentence.

Related Work

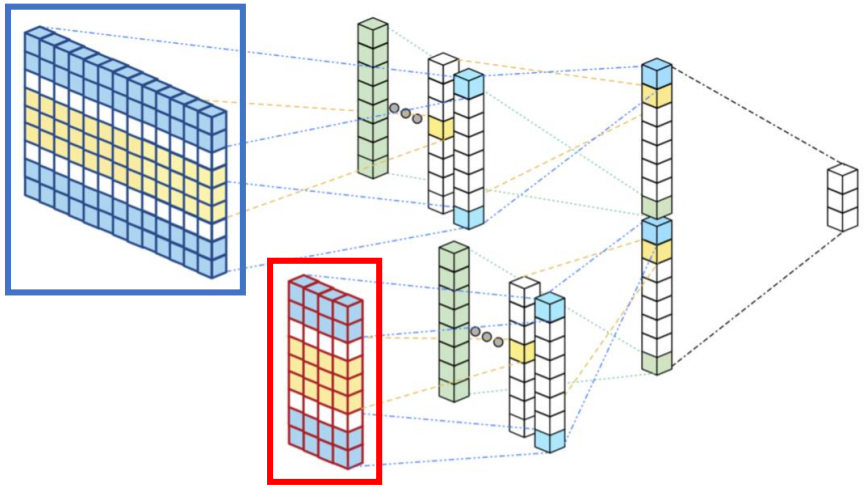
Lexicon Integrated Convolutional Neural Network

Yoon Kim이 제안한 textCNN에서 어휘(Lexicon) 자질을 추가
문서에 대한 word embedding과 lexicon embedding에 대해

[1] 각각 convolution 연산(separate convolution),
[2] 결과 벡터를 서로 concatenation 해서 최종 벡터를 생성

SemEval 2016 Testset에서 SOTA 성능 기록(F1 63.8)

Lexicon integrated cnn models with attention for sentiment analysis
B Shin, T Lee, JD Choi - arXiv preprint arXiv:1610.06272, 2016 - arxiv.org
Page 1. **Lexicon Integrated** CNN Models with Attention for Sentiment Analysis … However, some **lexicons** contain non-probabilistic scores (eg, frequency counts of the word in senti- mental contexts), which are … (a) Naive concatenation (Section 3.2.1). The **lexicon** embeddings (on …
☆ 99 7회 인용 관련 학술자료 전체 3개의 버전



(c) Separate convolution (Section 3.2.3). The lexicon embeddings are processed by a separate convolution (on the right) from the word embeddings (on the left).

Related Work

Hierarchical Attention Networks (HAN)

모델의 아이디어는 단어 단위의 GRU와, 문장 단위의 GRU로 나눠서
[1] 단어와
[2] 문장 각각에
Attention을 줘서 Hierarchical한 형태로 Feature를 잡아보자

[PDF] **Hierarchical Attention** Networks for Document Classification.

Z Yang, D Yang, C Dyer, X He, AJ Smola, EH Hovy - HLT-NAACL, 2016 - aclweb.org

… Abstract We propose a **hierarchical attention network** for document classification … Our primary contribution is a new neural archi- tecture (§2), the **Hierarchical Attention Network** (HAN) that is designed to capture two basic insights about document structure …

☆ 99 132회 인용 관련 학술자료 전체 10개의 버전 »

Zichao Yang[1], Diyi Yang[1], Chris Dyer[1], Xiaodong He[2], Alex Smola[1], Eduard Hovy[1]

[1]Carnegie Mellon University, [2]Microsoft Research, Redmond

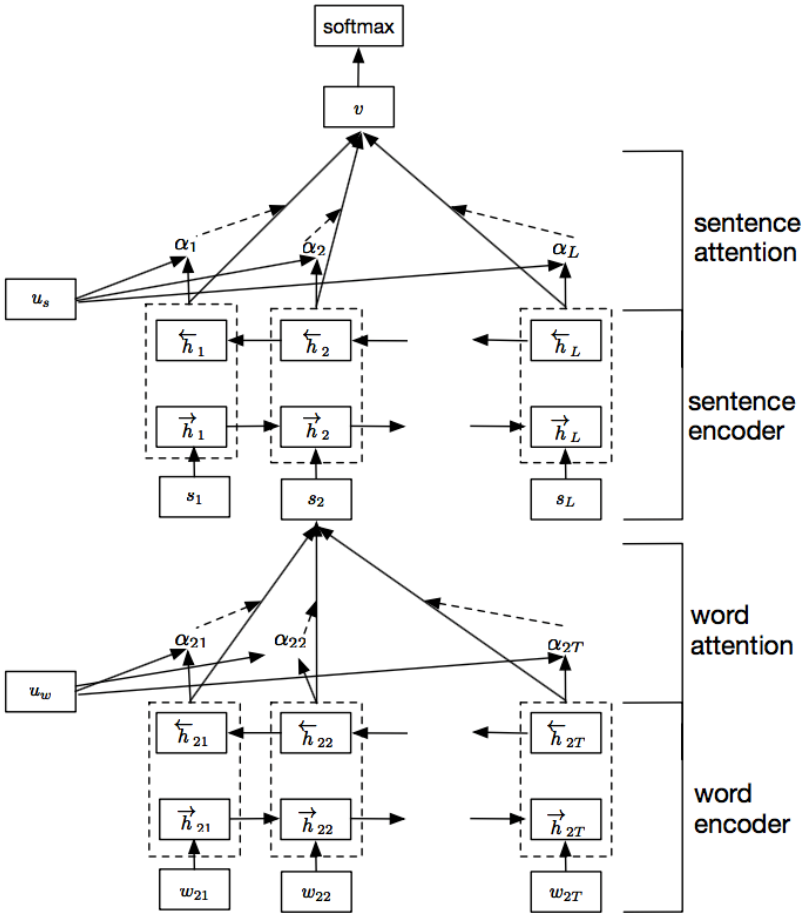


Figure 2: Hierarchical Attention Network.

Related Work

Hierarchical Attention Networks (HAN)

모델의 아이디어는 단어 단위의 GRU와, 문장 단위의 GRU로 나눠서
[1] 단어와
[2] 문장 각각에
Attention을 줘서 Hierarchical한 형태로 Feature를 잡아보자

$$u_{it} = \tanh(W_w h_{it} + b_w)$$

$$\alpha_{it} = \frac{\exp(u_{it}^\top u_w)}{\sum_t \exp(u_{it}^\top u_w)}$$

$$s_i = \sum_t \alpha_{it} h_{it}.$$

tions based on the weights. The context vector u_w can be seen as a high level representation of a fixed query "what is the informative word" over the words like that used in memory networks (Sukhbaatar et al., 2015; Kumar et al., 2015). The word context vector u_w is randomly initialized and jointly learned during the training process.

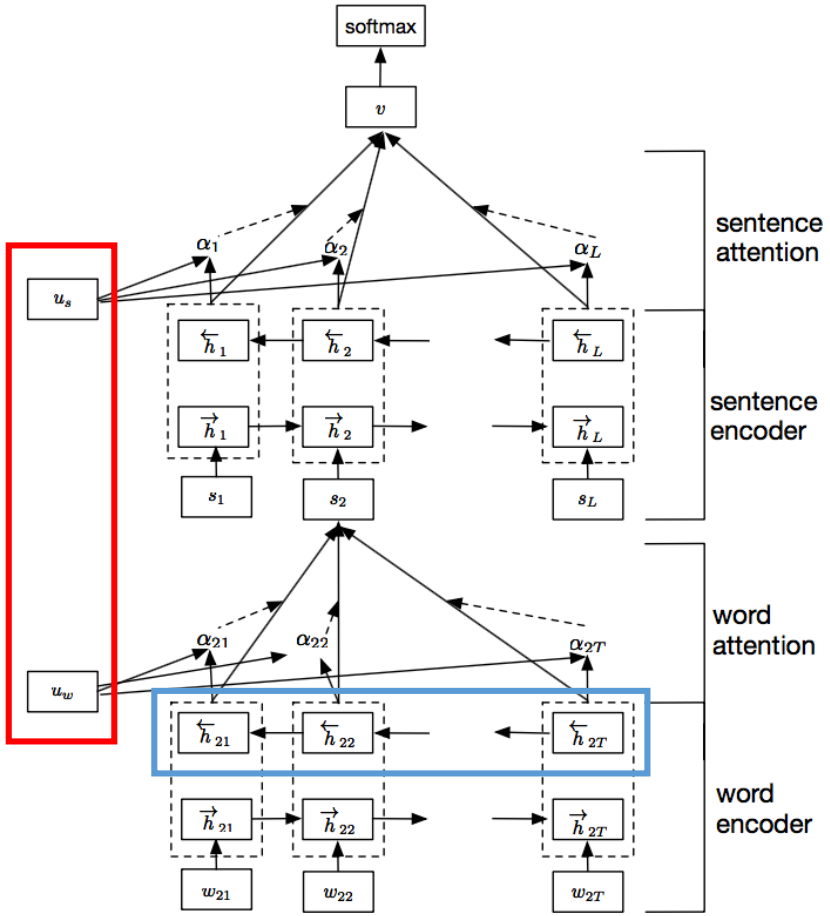


Figure 2: Hierarchical Attention Network.

Related Work

Hierarchical Attention Networks (HAN)

모델의 아이디어는 단어 단위의 GRU와, 문장 단위의 GRU로 나눠서
[1] 단어와
[2] 문장 각각에
Attention을 줘서 Hierarchical한 형태로 Feature를 잡아보자

GT: 1 Prediction: 1

why does zebras have stripes ?
what is the purpose or those stripes ?
who do they serve the zebras in the wild life ?
this provides camouflage - predator vision is such that it is usually difficult for them to see complex patterns

GT: 4 Prediction: 4

how do i get rid of all the old web searches i have on my web browser ?
i want to clean up my web browser go to tools > options .
then click " delete history " and " clean up temporary internet files . "

Figure 6: Documents from Yahoo Answers. Label 1 denotes Science and Mathematics and label 4 denotes Computers and Internet.

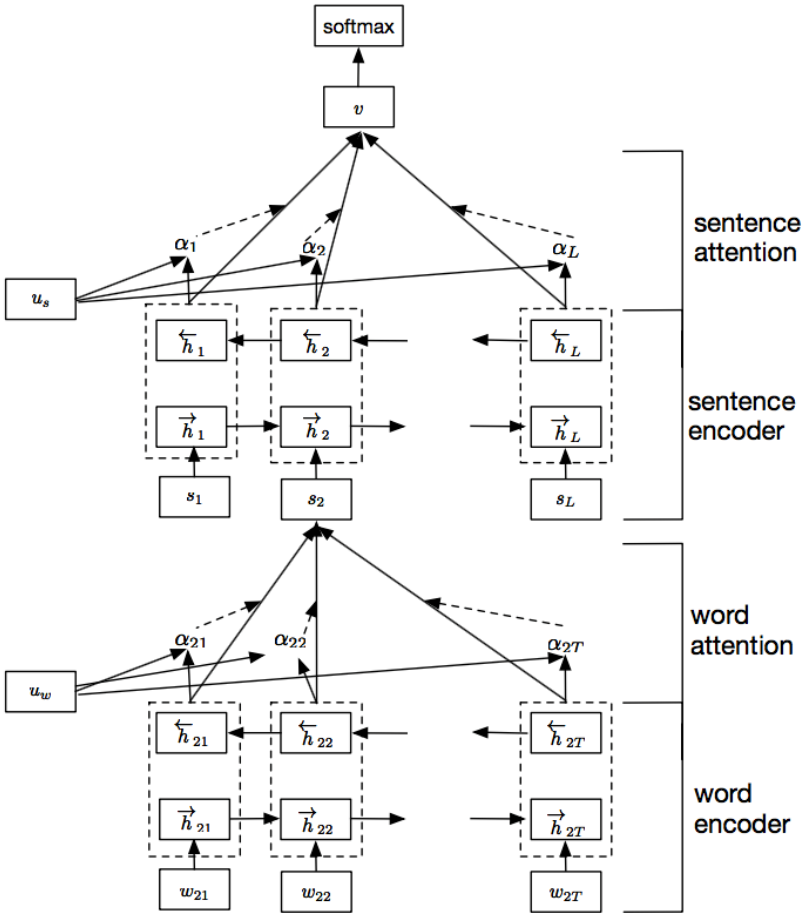


Figure 2: Hierarchical Attention Network.

Model

모델의 성능 향상을 위한 가설	모델 구조
Embedding 표현을 개선하면 성능이 높아질 것	Lexicon-enhanced embedding layer
멀티채널은 embedding 표현 개선 할 것	Multichannel convolutional layer
계층형 GRU는 문서 구조, 장기의존성 고려할 것	GRU-based word/sentence encoder
Attention은 중요 요소를 고려, 성능 높일 것	Attention word/sentence layer

Model

모델의 성능 향상을 위한 가설	모델 구조
Embedding 표현을 개선하면 성능이 높아질 것	Lexicon-enhanced embedding layer
멀티채널은 embedding 표현 개선 할 것	Multichannel convolutional layer
계층형 GRU는 문서 구조, 장기의존성 고려할 것	GRU-based word/sentence encoder
Attention은 중요 요소를 고려, 성능 높일 것	Attention word/sentence layer

Model

모델의 성능 향상을 위한 가설	모델 구조
Embedding 표현을 개선하면 성능이 높아질 것	Lexicon-enhanced embedding layer
멀티채널은 embedding 표현 개선 할 것	Multichannel convolutional layer
계층형 GRU는 문서 구조, 장기의존성 고려할 것	GRU-based word/sentence encoder
Attention은 중요 요소를 고려, 성능 높일 것	Attention word/sentence layer

Model

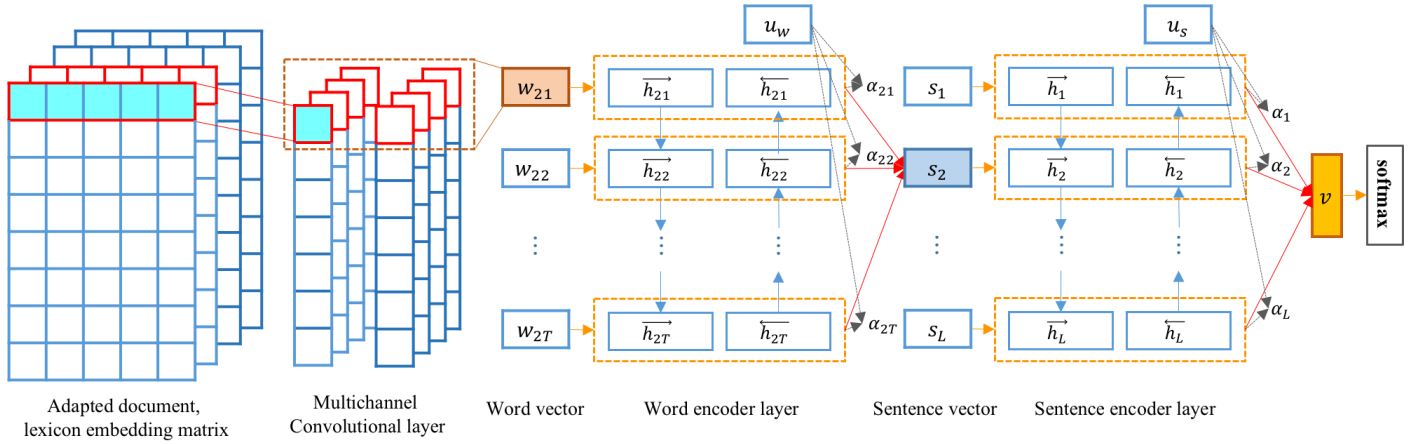
모델의 성능 향상을 위한 가설	모델 구조
Embedding 표현을 개선하면 성능이 높아질 것	Lexicon-enhanced embedding layer
멀티채널은 embedding 표현 개선 할 것	Multichannel convolutional layer
계층형 GRU는 문서 구조, 장기의존성 고려할 것	GRU-based word/sentence encoder
Attention은 중요 요소를 고려, 성능 높일 것	Attention word/sentence layer

Model

모델의 성능 향상을 위한 가설	모델 구조
Embedding 표현을 개선하면 성능이 높아질 것	Lexicon-enhanced embedding layer
멀티채널은 embedding 표현 개선 할 것	Multichannel convolutional layer
계층형 GRU는 문서 구조, 장기의존성 고려할 것	GRU-based word/sentence encoder
Attention은 중요 요소를 고려, 성능 높일 것	Attention word/sentence layer

Model

제안 모델

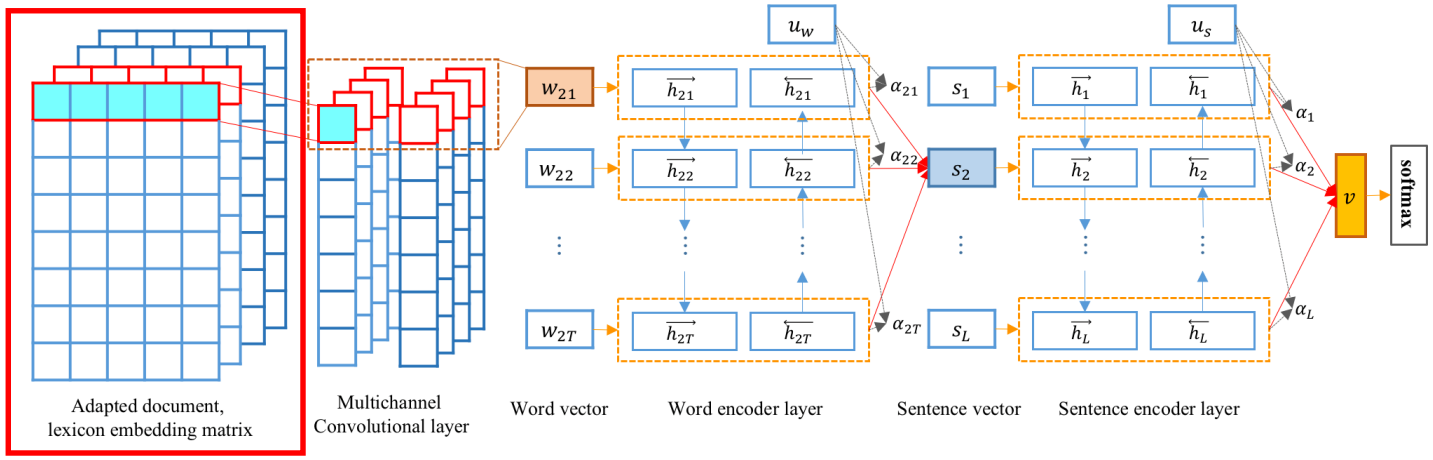


모델 구조

- Lexicon-enhanced embedding layer
- Multichannel convolutional layer
- GRU-based word/sentence encoder
- Attention word/sentence layer

Model

제안 모델



모델 구조

Lexicon-enhanced embedding layer

Multichannel convolutional layer

GRU-based word/sentence encoder

Attention word/sentence layer

Model

Embedding 개선을 위한 모델 구조

모델의 입력은 두 가지로 구성됨

[1] Word embedding
[2] Lexicon embedding

모델 구조

Lexicon-enhanced embedding layer

Multichannel convolutional layer

GRU-based word/sentence encoder

Attention word/sentence layer

Model

Embedding 개선을 위한 모델 구조

[1] Word embedding

Sentiment 140로 모델을 pre-training 할 때 학습한 word embedding을 사용함
-> adapted word embedding

이때 word2vec(skip-gram)으로 학습한 word embedding을 초기값으로 사용함

모델 구조

Lexicon-enhanced embedding layer

Multichannel convolutional layer

GRU-based word/sentence encoder

Attention word/sentence layer

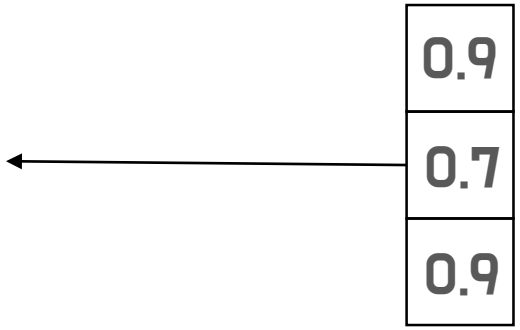
Model

Embedding 개선을 위한 모델 구조

[2] Lexicon embedding (Shin이 제안한 방법 사용)

어휘 데이터셋으로부터 감성점수를 얻어 embedding함
각 데이터셋은 key-value 형태 ({ Joy : 0.98 })
(-1: Negative, +1:Positive)

Word "Joy"에 대한
sentiment score들을 embedding



모델 구조

Lexicon-enhanced embedding layer

Multichannel convolutional layer

GRU-based word/sentence encoder

Attention word/sentence layer

Model

Embedding 개선을 위한 모델 구조

한 문장 내에 단어 T개가 있다고 한다면
한 문장은 다음과 같이 표현 가능

$$x_{1:T} = x_1 \oplus x_2 \oplus \ldots \oplus x_T$$

$x_i \in R^k$ 는 k차원을 갖는 i번째 단어벡터를 의미함

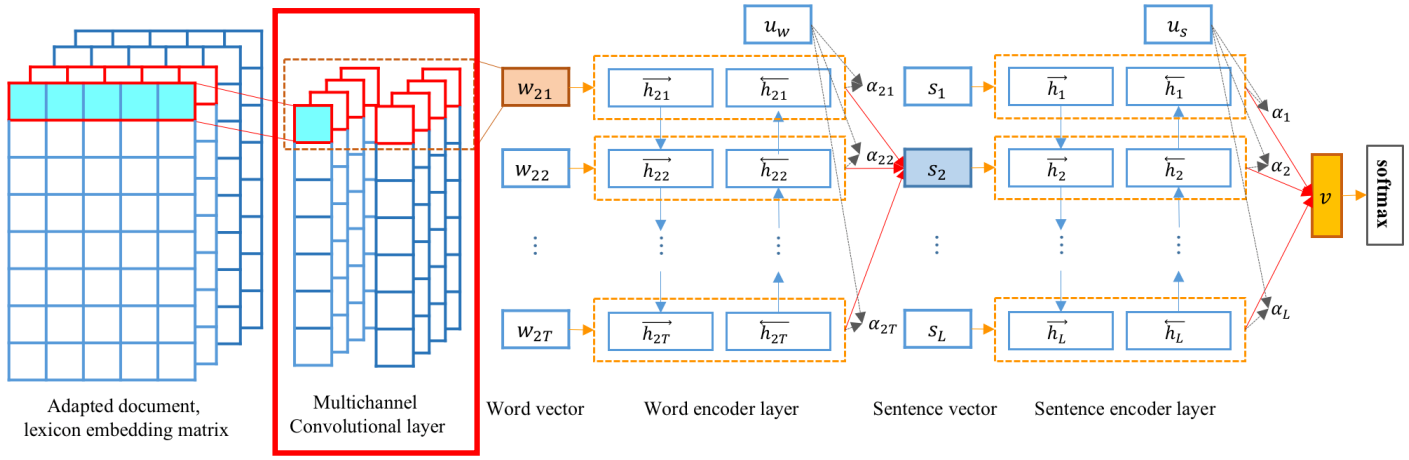
$\oplus$ 는 concatenation 연산자를 의미함

모델 구조

- Lexicon-enhanced embedding layer
- Multichannel convolutional layer
- GRU-based word/sentence encoder
- Attention word/sentence layer

Model

Embedding 개선을 위한 모델 구조



모델 구조

- Lexicon-enhanced embedding layer
- **Multichannel convolutional layer**
- GRU-based word/sentence encoder
- Attention word/sentence layer

Model

Embedding 개선을 위한 모델 구조

Adapted word embedding과
Lexicon embedding에서 자질을 추출하기 위해 사용

CNN에 사용된 filter $\mathrm{w} \in R^{hk}$ 의 개수는 200개
Window size h는 Attention으로부터 얻을 수 있는
정보 자질(informative feature)를 고려하여 1로 설정

Convolution 연산을 통해 나온 자질 $c_i = f(\mathrm{w} \cdot \mathrm{x}_{i:i+h-1} + \mathrm{b})$

모델 구조

Lexicon-enhanced embedding layer

Multichannel convolutional layer

GRU-based word/sentence encoder

Attention word/sentence layer

Model

Embedding 개선을 위한 모델 구조

멀티채널은 Kim이 사용한 방식과 동일하게 사용됨
[1] 정적 채널(static channel)
[2] 비정적 채널(non-static channel)

Lexicon embedding의 경우 Shin이 제안한 Separate convolution 기법을 적용하여 word embedding과 lexicon embedding에 각각의 다른 CNN을 적용 후 합침

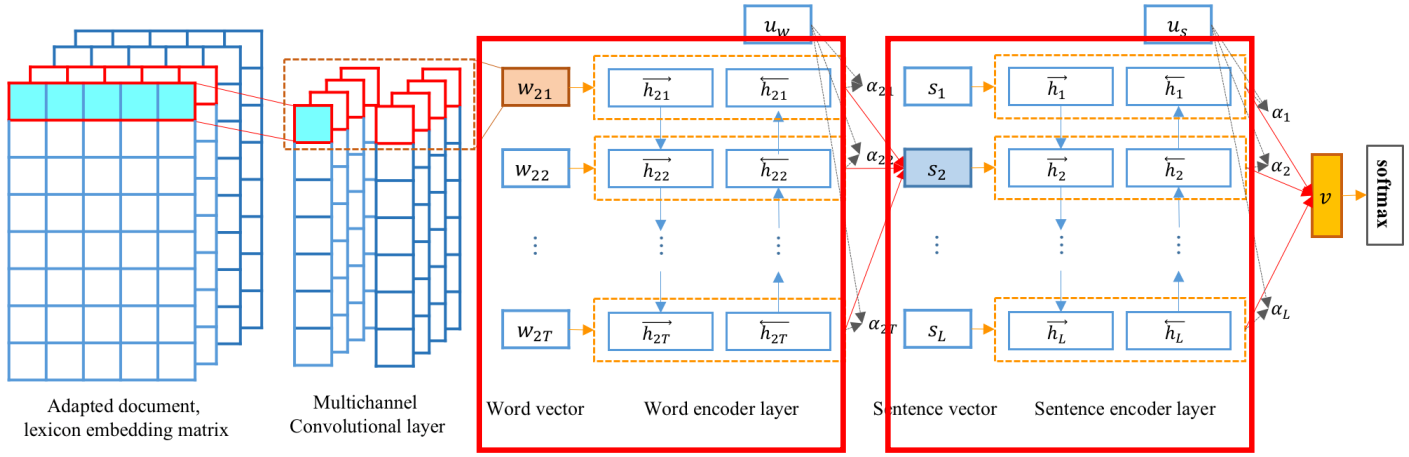
Separate convolution 자질 $c_i^{sc} = c_i^{word} \oplus c_i^{lexicon}$

모델 구조

- Lexicon-enhanced embedding layer
- **Multichannel convolutional layer**
- GRU-based word/sentence encoder
- Attention word/sentence layer

Model

문서의 구조를 고려한 HAN



모델 구조

Lexicon-enhanced embedding layer

Multichannel convolutional layer

GRU-based word/sentence encoder

Attention word/sentence layer

Model

문서의 구조를 고려한 HAN

Yang이 제안한 HAN 구조에 기초함
Word encoder와 Sentence encoder로 구성됨

$$w = c^{sc}, t \in [1, T]$$

word encoder

$$\overrightarrow{h_{it}} = \overrightarrow{\mathrm{GRU}}(w_{it}), t \in [1, T]$$

word attention α_{it}

$$\overrightarrow{h_i} = \overrightarrow{\mathrm{GRU}}(s_i), i \in [1, L]$$

sentence encoder

모델 구조

- Lexicon-enhanced embedding layer
- Multichannel convolutional layer
- **GRU-based word/sentence encoder**
- **Attention word/sentence layer**

Model

문서의 구조를 고려한 HAN

이때 사용한 GRU는 모두 Bidirectional GRU (dim=100)

$$w = c^{sc}, t \in [1, T]$$

word encoder

$$\overrightarrow{h_{it}} = \overrightarrow{\mathrm{GRU}}(w_{it}), t \in [1, T]$$

word attention α_{it}

$$\overrightarrow{h_i} = \overrightarrow{\mathrm{GRU}}(s_i), i \in [1, L]$$

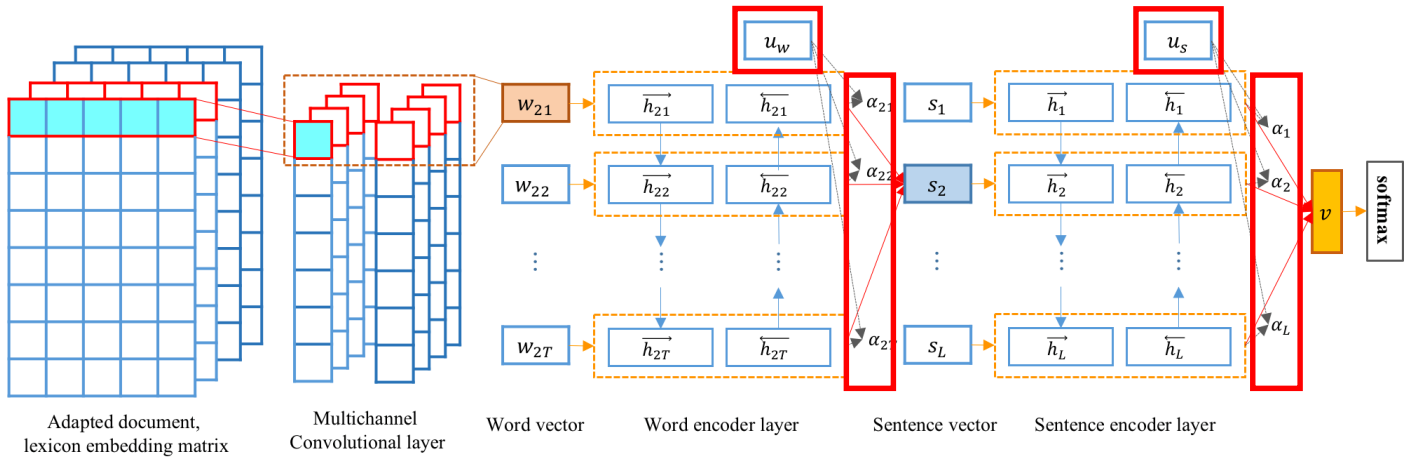
sentence encoder

모델 구조

- Lexicon-enhanced embedding layer
- Multichannel convolutional layer
- GRU-based word/sentence encoder
- Attention word/sentence layer

Model

문서의 구조를 고려한 HAN



모델 구조

- Lexicon-enhanced embedding layer
- Multichannel convolutional layer
- GRU-based word/sentence encoder
- **Attention word/sentence layer**

Model

문서의 구조를 고려한 HAN

단어 레벨에서 Attention α_{it}가 적용하여 s_i를 얻음

$$w = c^{sc}, t \in [1, T]$$

word encoder

$$\overrightarrow{h_{it}} = \overrightarrow{\mathrm{GRU}}(w_{it}), t \in [1, T]$$

word attention α_{it}

$$\overrightarrow{h_i} = \overrightarrow{\mathrm{GRU}}(s_i), i \in [1, L]$$

sentence encoder

모델 구조

- Lexicon-enhanced embedding layer
- Multichannel convolutional layer
- GRU-based word/sentence encoder
- **Attention word/sentence layer**

Model

문서의 구조를 고려한 HAN

word attention α_{it}는 hidden state h_{it} 를 통해 얻은 u_{it} 와 단어에 대한 context vector u_w을 내적 한 값에 대한 지수함수의 softmax로 얻음

$$u_{it} = \tanh(W_w h_{it} + b_w)$$

이후 word attention과 Hidden state의 가중합으로 문장 벡터 s_i 계산

$$\alpha_{it} = \frac{\exp(u_{it}^\top u_w)}{\sum_t \exp(u_{it}^\top u_w)}$$

$$s_i = \sum_t a_{it} h_{it}$$

모델 구조

Lexicon-enhanced embedding layer

Multichannel convolutional layer

GRU-based word/sentence encoder

Attention word/sentence layer

Model

문서의 구조를 고려한 HAN

문장 벡터도 같은 과정으로 계산됨

$$u_i = \tanh(W_s h_i + b_s)$$

$$\alpha_i = \frac{\exp(u_i^\top u_s)}{\sum_i \exp(u_i^\top u_s)}$$

$$v = \sum_i \alpha_i h_i$$

$$u_{it} = \tanh(W_w h_{it} + b_w)$$

$$\alpha_{it} = \frac{\exp(u_{it}^\top u_w)}{\sum_t \exp(u_{it}^\top u_w)}$$

$$s_i = \sum_t a_{it} h_{it}$$

모델 구조

- Lexicon-enhanced embedding layer
- Multichannel convolutional layer
- GRU-based word/sentence encoder
- **Attention word/sentence layer**

Model

문서의 구조를 고려한 HAN

Attention을 학습시킬 때 사용한

단어에 대한 context vector u_w 와
문장에 대한 context vector u_s 모두 100차원으로 설정

두 벡터는 랜덤으로 초기화되고
학습 단계에서 다른 parameter와 학습됨

HAN모델의 결과 벡터 v에 대해서 softmax로 최종 분류

모델 구조

Lexicon-enhanced embedding layer

Multichannel convolutional layer

GRU-based word/sentence encoder

Attention word/sentence layer

Experiments

Implementation

PYTORCH

Training

Experiments

데이터 - 트위터

SemEval 2017 Task 4 subtask A에서 제공한 데이터 셋을 사용함 (2013~2016년 데이터 제공)

<표 4-2> Train 2016 데이터 셋 예시

Class	Tweet
positive	Sunday morning, quiet day so time to welcome in #Windows10 @Microsoft @Windows http://t.co/7VtvAzhWmV
neutral	OK this is my pure speculation. @Microsoft owns the cloud compute tech. @Cloudgine is utilizing the tech. 3rd party devs is open to use.
negative	dear @Microsoft the newOoffice for Mac is great and all, but no Lync update? C'mon.

모든 트윗은 3가지 클래스를 가짐

$$C = \{\text{긍정}, \text{중립}, \text{부정}\}$$

Experiments

데이터 - 트위터

<표 4-1> 데이터 셋 개요

말뭉치	계	긍정	중립	부정	평균 단어 수	평균 문장 수	단어 사전 수
Train 2013	9,684	3,640	4,586	1,458	21.1	1.5	4,458
Dev 2013	1,654	575	739	340	21.0	1.5	1,188
Train 2015	489	170	253	6	21.4	1.6	448
Train 2016	6,000	3,094	2,043	863	20.9	1.3	2,728
Dev 2016	1,999	843	765	391	20.7	1.3	1,217
DevTest 2016	2,000	994	681	325	21.3	1.3	1,162
Test 2013	3,547	1,475	1,513	559	21.4	1.5	2,275
Test 2014	1,853	982	669	202	21.2	1.5	1,172
Test 2015	2,390	1,038	987	365	20.8	1.3	1,516
Test 2016	20,632	7,059	10,324	3,231	21.1	1.3	6,277
TwtSarc 2014	86	33	13	40	15.5	1.6	75
SMS 2013	2,094	492	1,208	394	17.2	1.4	971
LiveJournal 2014	1,142	427	411	304	12.8	1.1	489

Experiments

데이터 - 어휘 말뭉치(Lexicon corpus)

총 8종류의 어휘 말뭉치 사용
어휘에 대응되는 감성 점수는 -1 ~ +1 로 정규화
긍정, 부정으로 되어있는 경우 +1, -1로 변환
어휘 말뭉치에 등장하지 않는 단어는 0의 점수를 부여

- SemEval-2015 English Twitter Sentiment Lexicon (2015). **어휘 말뭉치 리스트**
- NRC Hashtag Affirmative and Negated Context Sentiment Lexicon (2014).
- NRC Sentiment140 Lexicon (2014).
- Yelp Restaurant Sentiment Lexicons (2014).
- NRC Hashtag Sentiment Lexicon (2013).
- Bing Liu Opinion Lexicon (2004).
- Macquarie Semantic Orientation Lexicon (2009).
- NRC Word-Emotion Association Lexicon (2010).

<표 4-3> 전처리 된 어휘 말뭉치 예시

SemEval-2015		Bing Liu Opinion Lexicon	
lexicon	sentiment score	lexicon	sentiment score
loves	+0.984	a+	+1
inspirational	+0.984	abound	+1
amazing	+0.969	meaningful	+1
post	+0.016	nicely	+1
know	0.000	winnable	+1
rare	-0.016	scarcely	-1
kill	-0.982	ruthless	-1
bitches	-0.984	zealously	-1
disappointment	-0.984	zombie	-1

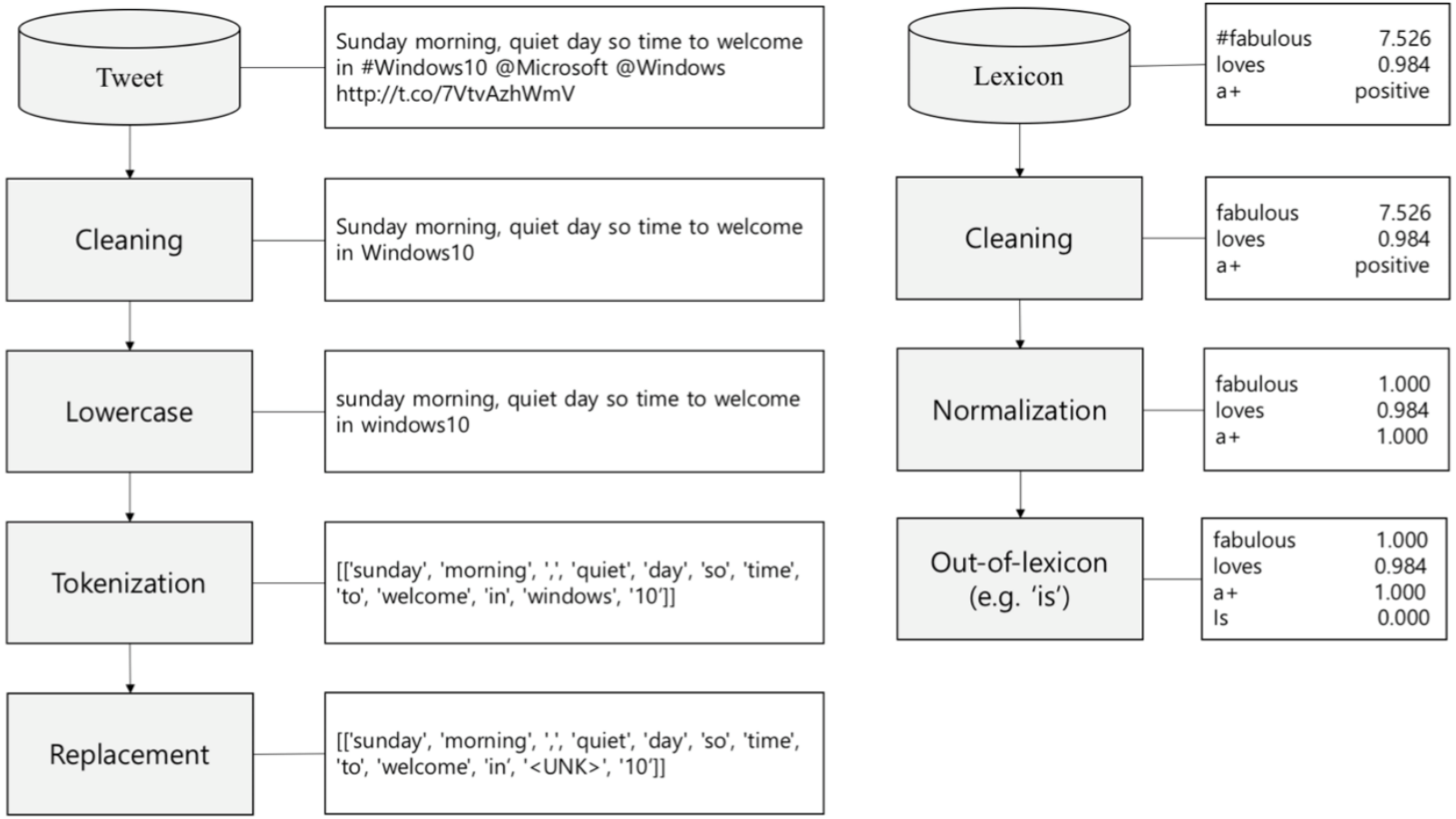
Experiments

전처리

Cleaning	URLs, hashtag에 있는 '#' 토큰 제거, 트윗 내 아이디 제거
Lowercase	트윗과 어휘에 있는 단어 모두 소문자 변환
Tokenization	트윗은 NLTK sentence tokenizer로 문장단위로 나눈 후 twitter tokenizer를 통해 단어 단위로 토큰화
Replacement	out-of-vocabulary (OOV) 단어를 〈UNK〉 토큰으로 대체

Experiments

전처리



<그림 4-1> 데이터 전처리 과정

Experiments

학습

Pre-training 단계

Sentiment 140 데이터로
word2vec -> adapted word embedding 변경

Training 단계

SemEval 2017에서 제공한 데이터로
모델 전체를 다시 학습

$$L = -\sum_{d} \log p_{dj}$$

Optimizer & Loss & Hyperparameter

Optim: SGD + Momentum 사용

Loss: Negative log likelihood 사용

Hyperparameter:

Word embedding dim = (100)
Lexicon embedding dim = (8)
GRU hidden dim = (100)
Number of filters in CNNs = (200)
Batch size = (64)
Learning rate = (0.1)
Momentum = (0.6)
Number of epochs = (20)
Dropout[10] rate in CNN, GRU = (0.5, 0.7)

Experiments

성능평가 및 분석

Evaluation metric으로 macro-average F1 score사용 $\text{Macro-average } F_1 = \frac{F_1^P + F_1^U + F_1^N}{3}$

SemEval 2017에서 제공한 Testset에 대한 결과를 모델별 비교분석 (CNN, GRU, HAN..)

객관적인 비교를 위해 SemEval 1st, 2nd Team 및 Shin의 SC-EAV 모델과 비교

Experiments

성능평가 및 분석

<표 4-5> 모델의 F1 score에 대한 실험 결과

	Model	Test 2013	Test 2014	Test 2015	Test 2016	TwtSarc 2014	SMS 2013	LiveJournal 2014
본 논문	1CNN (Baseline)	63.22	60.43	61.04	60.41	43.46	65.05	65.18
	1CNN+lex	62.70	61.37	61.76	62.19	46.39	67.07	68.04
	2CNN	61.71	61.84	61.17	60.16	51.20	64.35	66.96
	2CNN+lex	62.63	63.74	61.65	61.91	49.82	67.17	68.99
	HAN	65.39	62.64	59.30	57.13	53.01	65.58	64.50
	HAN+lex	68.03	66.95	63.98	62.25	54.11	70.32	70.78
	1CNN+BiLSTM+lex	68.08	68.45	62.60	61.92	57.23	69.82	68.62
	1CNN+BiGRU+lex	69.81	68.80	62.09	61.95	53.71	68.86	69.31
	Our model w/o att, adapted emb	69.58	67.10	62.88	62.24	55.44	68.92	70.31
	Our model w/o adapted emb	69.90	69.45	65.03	62.71	57.18	70.85	72.06
	Our model	**71.47**	70.16	66.83	**64.11**	**62.26**	**74.16**	72.84
Shin et al., 2016 (state-of-the-art)	SC-EAV	-	-	-	63.80	-	-	-
Deriu et al., 2016	SemEval 2016. 1st Team	70.01	71.55	**67.05**	63.30	56.63	-	69.51
Rouvier et al., 2016	SemEval 2016. 2nd Team	70.60	**74.40**	66.20	63.00	46.7	63.4	**74.1**

SOTA보다 0.31% 높은 64.11%의 F1 score를 기록

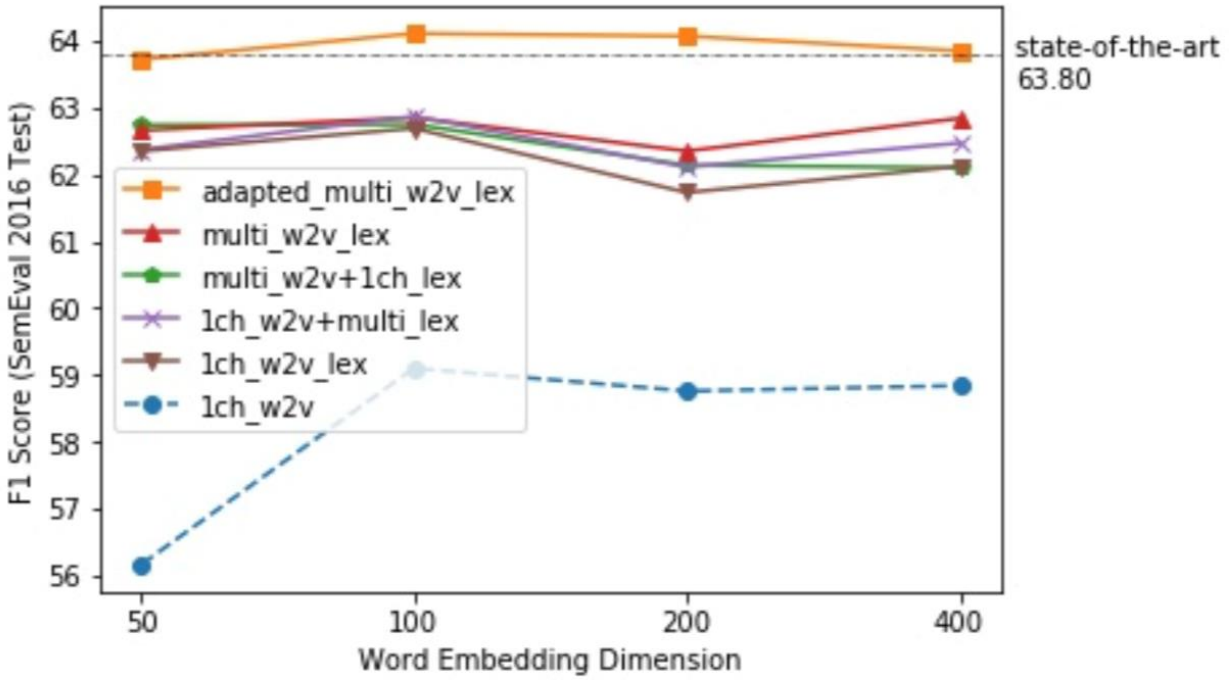
Experiments

성능평가 및 분석 - Embedding 방법에 따른 모델 성능 분석

Embedding 방법에 따라 성능이 달라짐

word embedding의 차원이 100, 200, 400일 때 SOTA성능 보다 높은 점수를 기록함

1ch_w2v_lex 단일 채널을 적용한 모델과 멀티채널을 적용한 모델을 비교 할 때, 멀티채널 모델의 성능이 대체적으로 높음



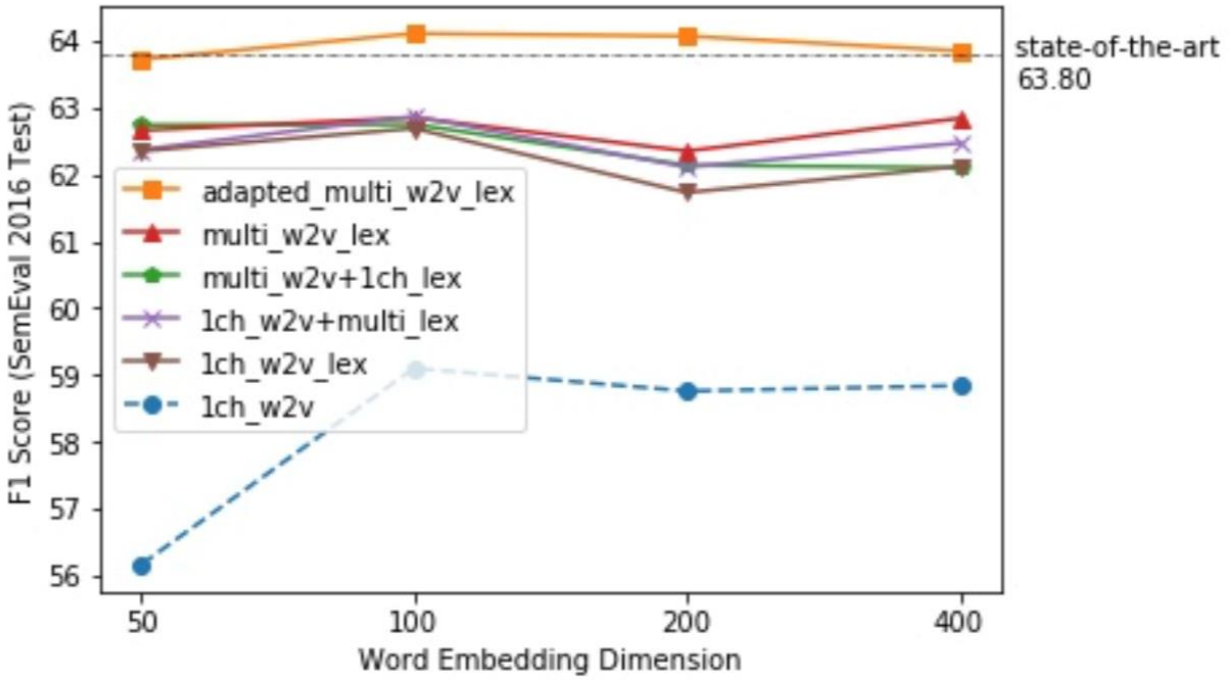
<그림 4-2> embedding 방법 및 차원 크기에 따른 모델 성능

Experiments

성능평가 및 분석 - Embedding 방법에 따른 모델 성능 분석

Word2vec의 경우 단어의 의미가 반대일지라도 문법적으로 비슷한 단어의 경우 비슷하게 위치하게 되는데, 멀티채널은 이러한 표현을 개선함

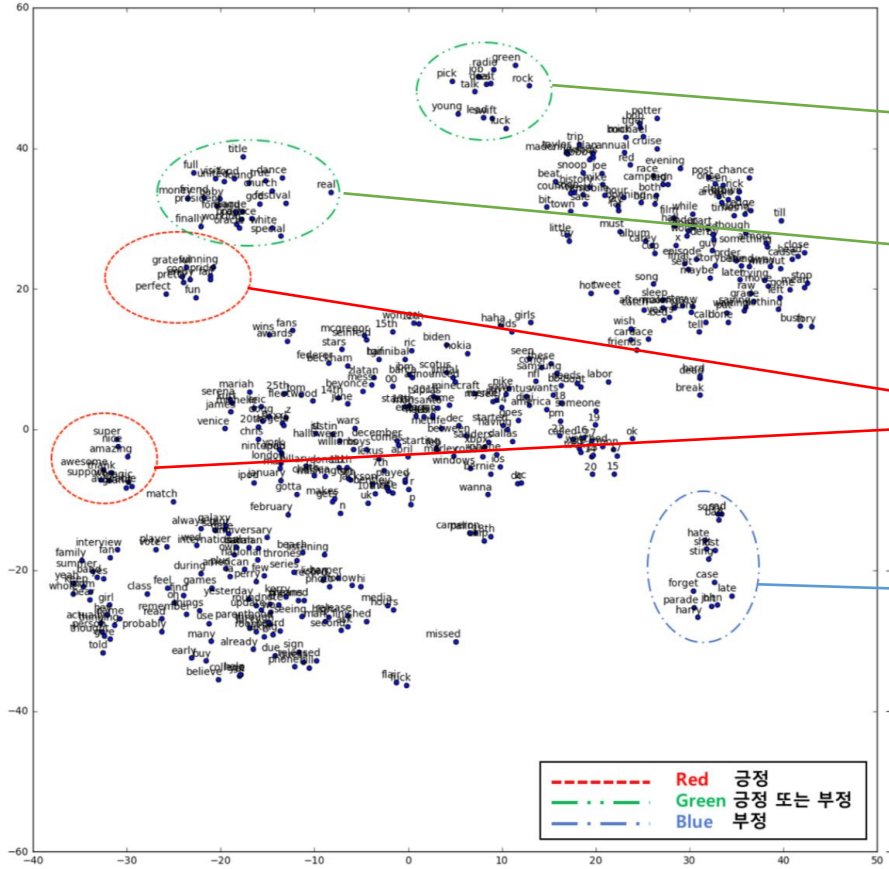
Lexicon의 경우 어휘 데이터의 특성상 커버리지가 낮기 때문에 lexicon embedding이 희소(sparse)한 특징을 갖는데, 멀티채널은 이러한 표현을 개선하는 것으로 보임



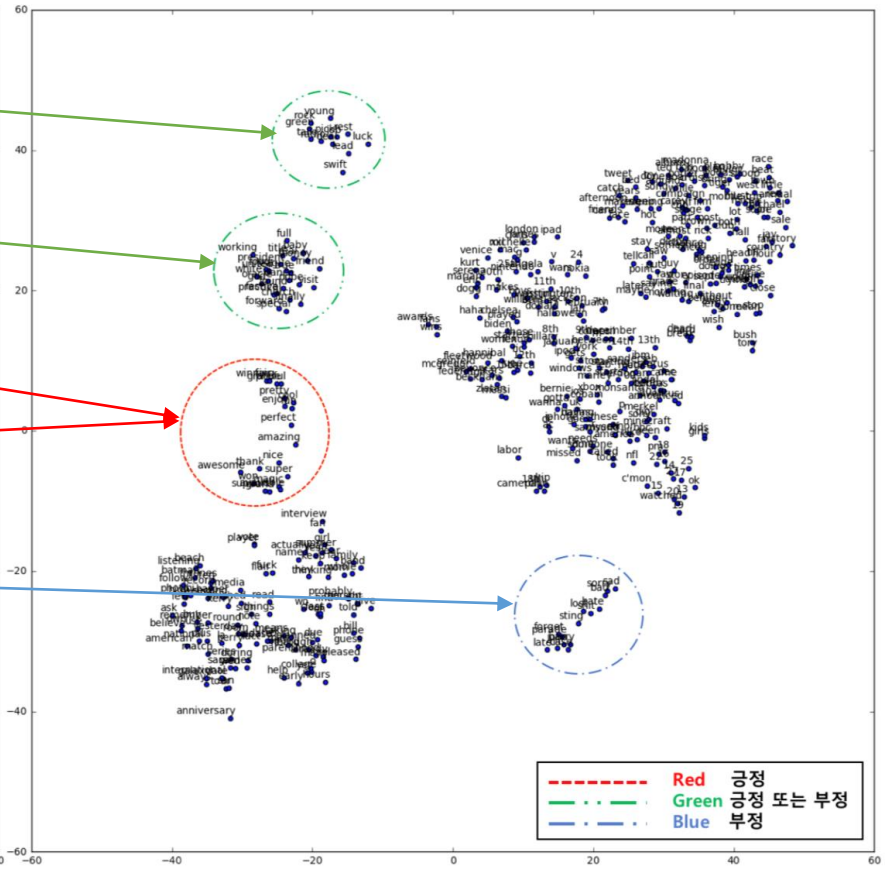
<그림 4-2> embedding 방법 및 차원 크기에 따른 모델 성능

Experiments

성능평가 및 분석 - Multichannel이 적용된 Lexicon embedding 분석



<그림 4-3> static lexicon embedding의 t-SNE



<그림 4-4> non-static lexicon embedding의 t-SNE

t-SNE Perplexity: 20

긍정적인 단어(빨간색)와 부정적인 단어(파란색)가 더 잘 군집화 됨

문맥에 따라 긍정/부정 의미를 나타내는 단어(초록색)도 더 잘 군집화 됨

Experiments

성능평가 및 분석 - 계층형 구조 및 Attention에 따른 모델의 성능 분석

- 계층형 구조를 적용한 2번 모델이 전반적으로 성능이 좋음
- Attention을 적용한 경우 1, 2번 모델보다 F1 score가 대체적으로 높음
- 결과적으로 계층형 구조+Attention 가 성능이 가장 높음

<표 4-6> 계층형 구조 및 Attention에 대한 실험 결과

	No	Model	Test 2013 (21.4/1.5)	Test 2014 (21.2/1.5)	Test 2015 (20.8/1.3)	Test 2016 (21.1/1.3)	TwtSarc 2014 (15.5/1.6)	SMS 2013 (17.2/1.4)	LiveJournal 2014 (12.8/1.1)
계층형 구조 X	1	1CNN+BiGRU+lex	69.81	68.80	62.09	61.95	53.71	68.86	69.31
계층형 구조 O	2	Our model w/o att, adapted emb	69.58	67.10	62.88	62.24	55.44	68.92	70.31
계층형 구조 O + Attention	3	Our model w/o adapted emb	69.90	69.45	65.03	62.71	57.18	70.85	72.06
제안 모델	4	Our model	71.47	70.16	66.83	64.11	62.26	73.61	72.84

(평균 단어/평균 문장수)

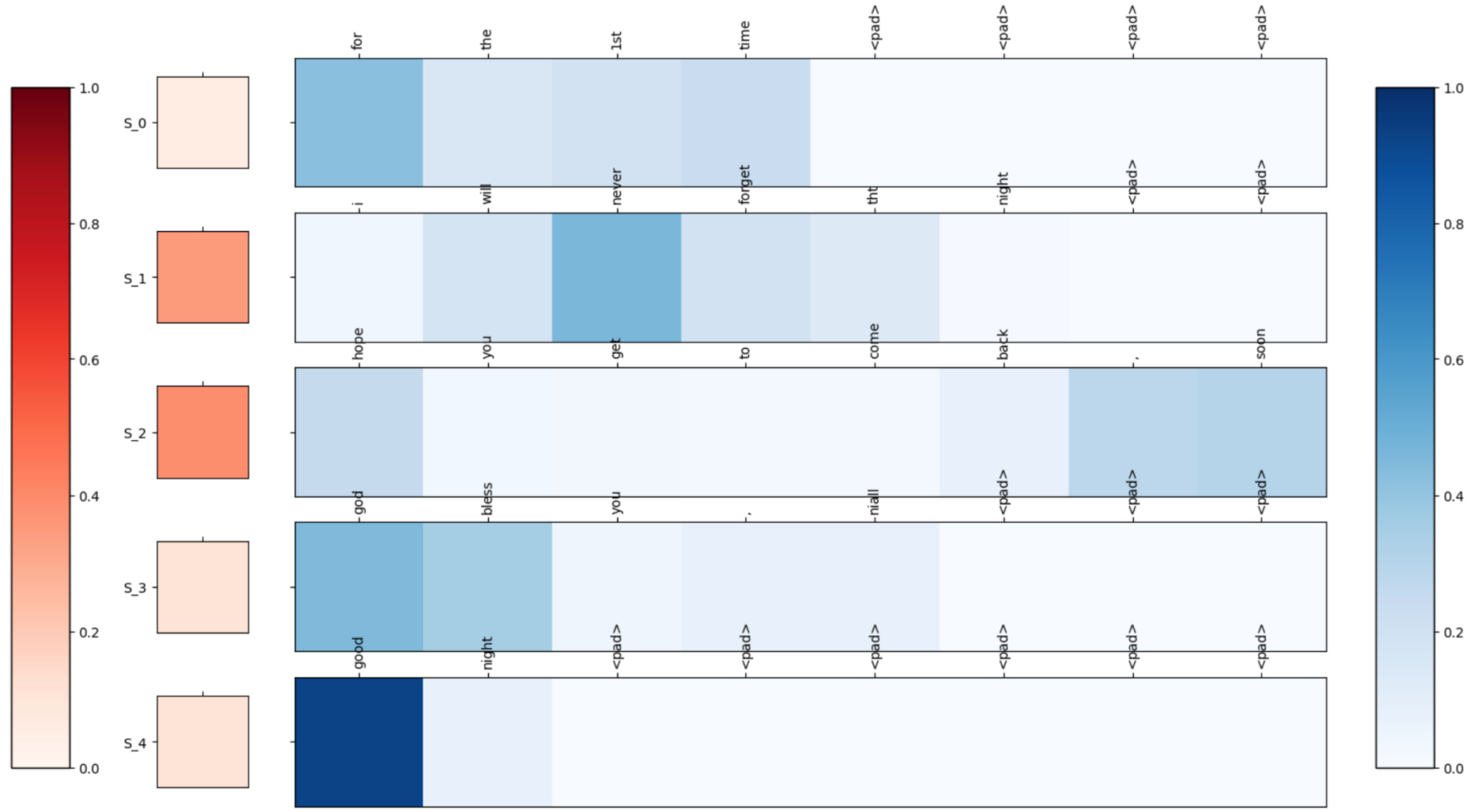
Experiments

성능평가 및 분석 - Attention 통합 결정 요인분석

긍정적인 의미를 가진 트윗:
"for the 1st time. I will NEVER forget tht night. Hope you get to come back, soon. God bless you, Niall. Good night."

Experiments

For, the, , 등의 단어에 Attention이 분산됨

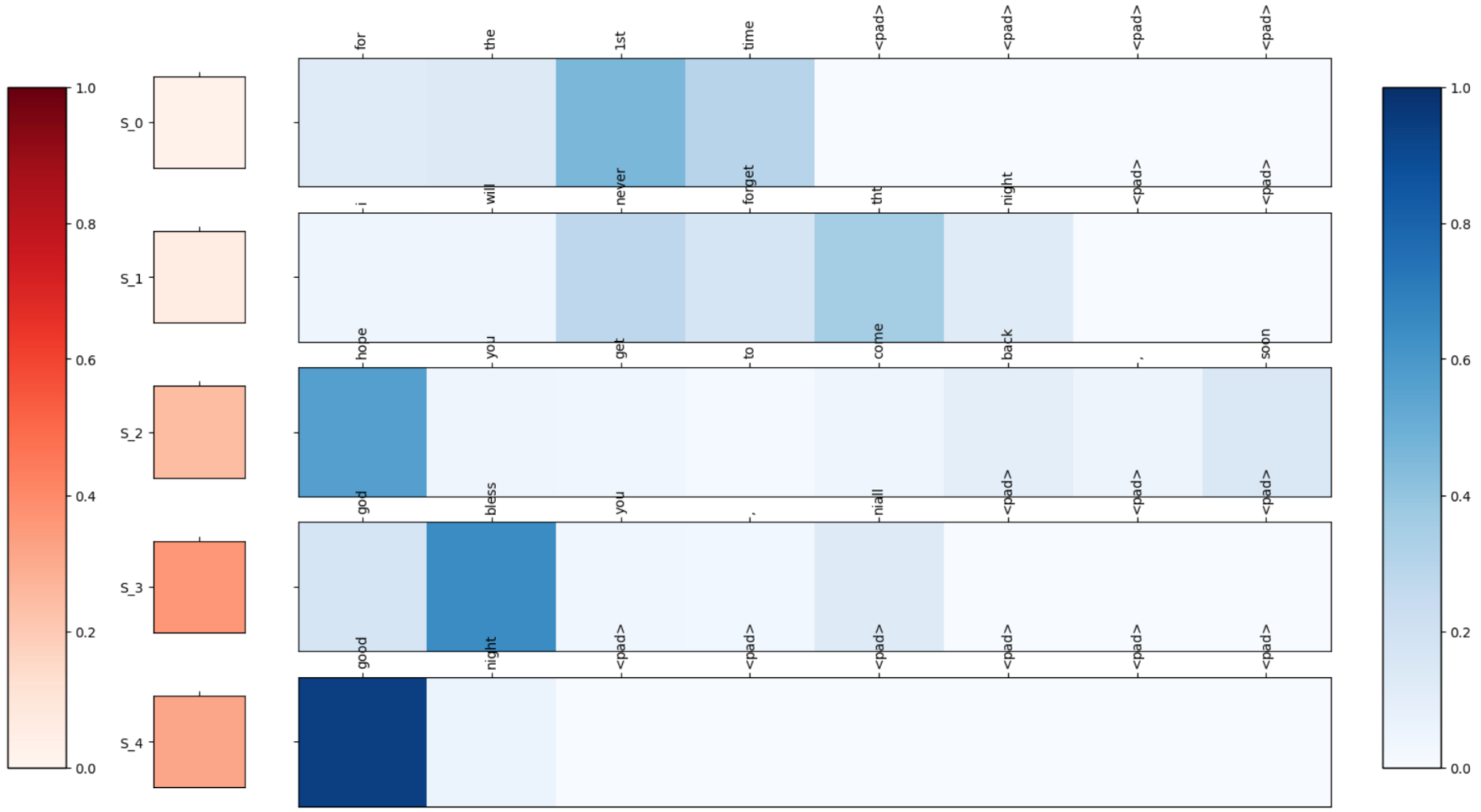


<그림 4-5> 긍정 트윗에 대한 Attention heatmap (w/o lex)

Experiments

Hope, bless, good 에
더 Attention이 잡힘

문장에 대한
Attention변함



<그림 4-6> 긍정 트윗에 대한 Attention heatmap (w/ lex)

Experiments

Lexicon을 사용할 때, 특정단어가 더 상위에 랭크됨

성능평가 및 분석 - Attention 통한 결정 요인분석

각 클래스 당 Attention이 높은 단어의 순위

어휘 자질을 추가했을 때 어휘 말 뭉치에 있는 'love', 'good', 'happy', 'not', 'iran' 등의 단어가 더 상위에 랭크 됨

'!', '<unk>', '\', ')' 토큰 같은 경우 학습된 word context vector 와 값이 비슷하기 때문에 모든 클래스에서 상위에 랭크(rank)된 보임

<표 4-7> Attention이 높은 단어 순위

No	w/o lex			w/ lex		
	긍정	중립	부정	긍정	중립	부정
1	!	<unk>	!	!	<unk>	<unk>
2	<unk>	\\	<unk>	<unk>	!	!
3	\\	!	\\	love	\\	n't
4	'	'	n't	)	?	not
5	,	,	'	good	)	\\
6	good	the	not	happy	n't	?
7	the	s	,	best	not	the
8	love	n't	the	\\	the	iran
9	happy	it	s	?	no	no
10	great	not	charlie	great	you	,
11	it	?	it	the	,	israel
12	best	murray	may	.	(	you
13	n't	a	why	excited	murray	it
14	s	)	?	wait	it	shit
15	)	may	is	'	'	)
16	wars	no	no	twilight	iran	fuck
17	i	i	fuck	a	brady	but
18	day	is	a	i	bieber	'
19	excited	sox	i	thank	of	a
20	may	(	shit	to	a	is

Conclusion & Future work

- 본 논문에서는 소셜미디어의 종류 중 하나인 트위터 데이터를 대상으로 문서 내에 표현된 감성을 긍정, 중립, 부정 클래스로 분류하고자 함
- 제안 모델의 성능이 SemEval 2016 Testset에 대해서 기존 SOTA 보다 높은 성능을 기록함

실험 결과에 기초한 결론

첫째로,
embedding의 표현을 개선하는 것은 성능을 높이는데 도움이 됨

adapted word embedding
-> 1~3% 정도 높임
lexicon embedding
-> 1~4% 정도 높임

둘째로,
멀티채널은 embedding을 개선하는데 도움이 됨

최대 1%까지 향상
Lexicon embedding 군집화

셋째로,
Attention mechanism은 성능을 높이는데 도움을 줌
모델의 결정요인에 대한 정보 제공

1~2% 정도 성능 향상
Lexicon 적용시 Heatmap 및 Attention 받은 단어 순위 변화

Conclusion & Future work

향후연구

- 트위터 데이터보다 문서의 길이가 더 긴 데이터에 대해서 연구
- fastText, Swivel등 다른 word embedding을 적용
- character embedding을 통해 OOV 문제 를 해결
- 강화학습(reinforcement learning)을 이용한 Negation scope detection에 관한 연구

QnA

Thank you :)