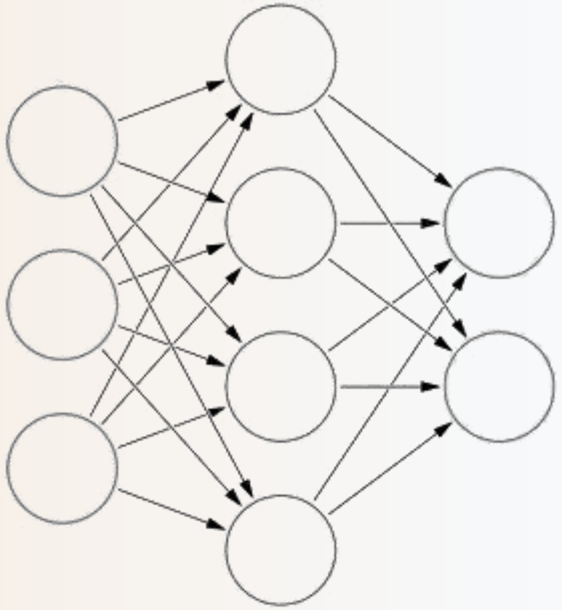
Lexicon Integrated CNN Models with Attention for Sentiment Analysis

- Bonggun Shin, Timothy Lee, Jinho D. Choi



Korea University,
2016010662
JooSung Yoon

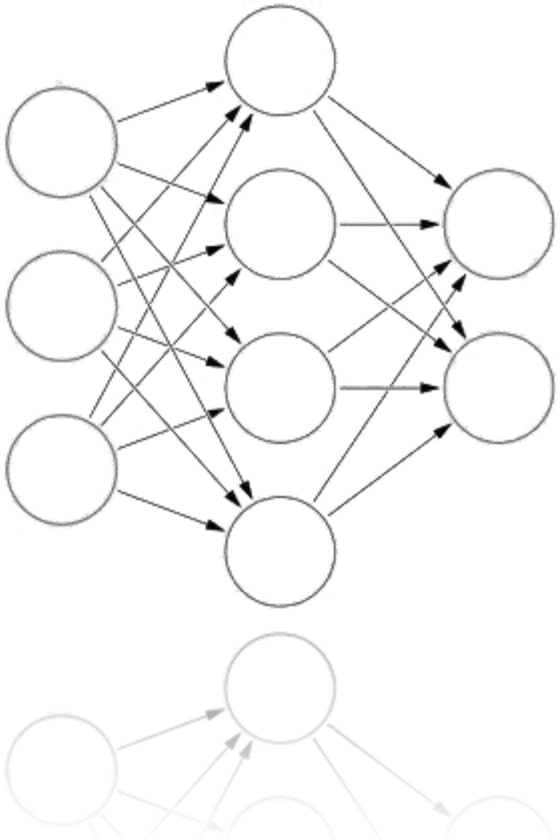
Deep Learning
professor **Dongsuk Yook**

Abstract

본 연구는 **Sentiment Analysis**를 하기 위해
Lexicon embedding과 Attention mechanism을
CNN에 적용한 Novel Approach에 관한 것이다

Keywords

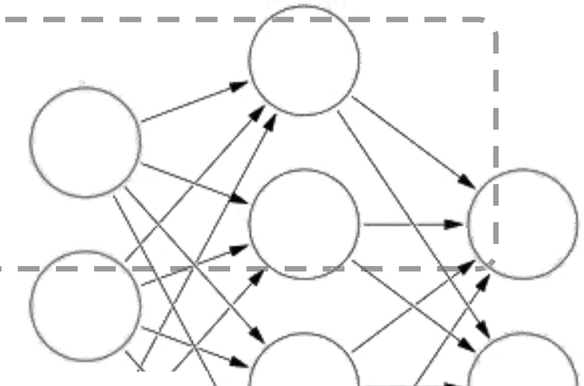
Sentiment analysis, Word embedding,
Lexicon, Attention, CNN, SemEval'16 Task 4



Introduction

Sentiment Analysis

Sentiment Analysis is a task of **identifying sentiment polarities expressed in documents**, typically positive, neutral, or negative.



Statistical models based on sparse features

Lexicons' sentiment scores are shown to be highly effective

Word embedding

The use of lexicons is getting faded away.
(WE are believed to capture the sentiment aspects of those words)

Introduction

Research Question

1. Can lexicons be **still useful** for sentiment analysis when coupled with WE?
2. If yes, what is the most effective way of **incorporating lexicons with WE**?

3가지 Approach

1. Naïve concatenation
2. Multichannel
3. Separate convolution

Embedding Attention

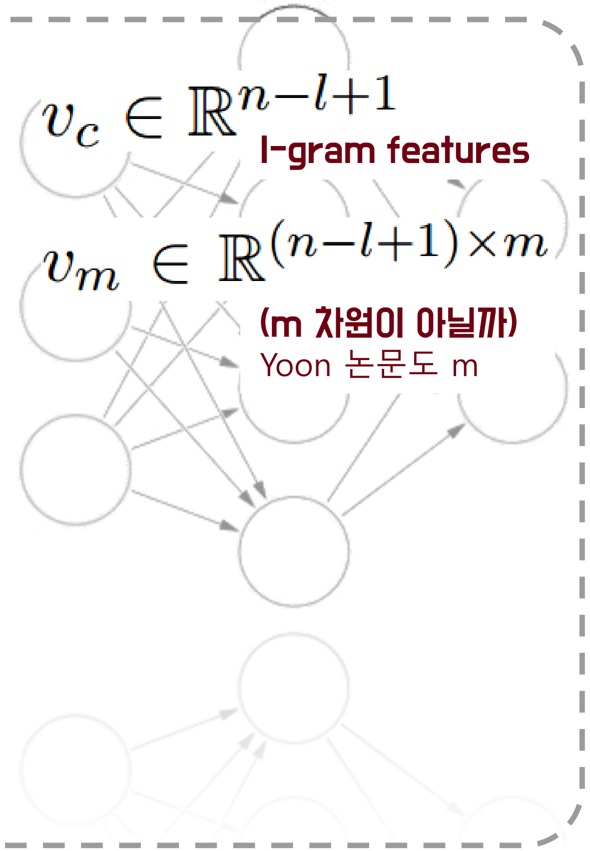
Word embedding으로 이루어진
Document matrix를 Length=1의
Filter로 Attention Matrix를 생성 후
Max pooling을 통해 Attention vector 계산

Approach

Notation

Input document matrix	$s \in \mathbb{R}^{n \times d}$
Num of words	n
Dim of word embedding	d
i'th word in document	$w_i \in \mathbb{R}^d$
Weights of filter	$c \in \mathbb{R}^{l \times d}$
Length of the filter	l
Num of the filter	m

Activation map of Conv (filter 마다 생기는)	$v_c \in \mathbb{R}^{n-l+1}$ l-gram features
Output of max pooling	$v_m \in \mathbb{R}^{(n-l+1) \times m}$ (m 차원이 아닐까) Yoon 논문도 m



Approach

Lexicon Integration

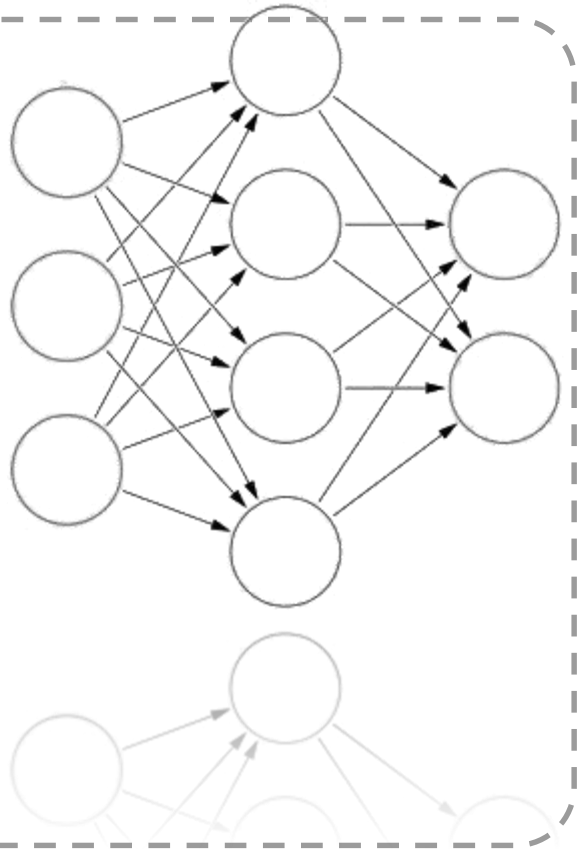
Lexicon Embedding

Multiple sources of lexicon datasets으로 부터 **score를 얻어서 embedding**

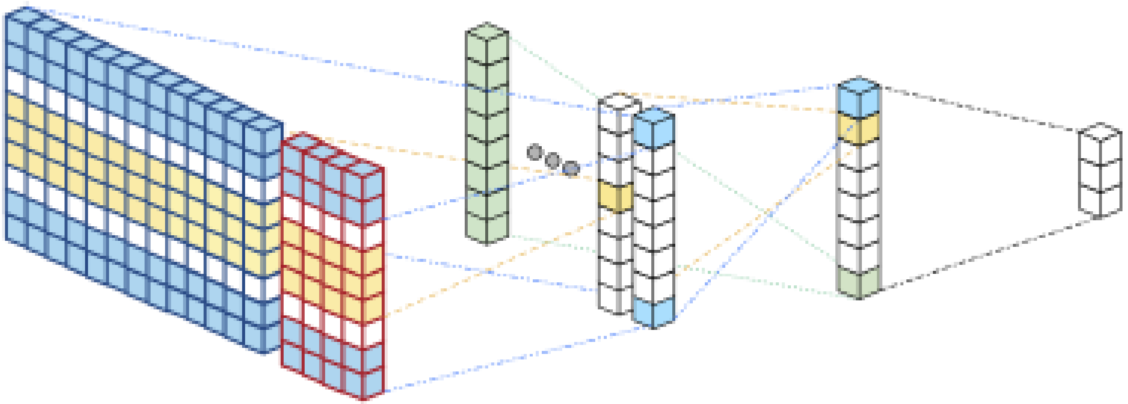
각 dataset은 key-value 형태 (**{ Joy : 0.98 }**) (-1: Negative, +1:Positive)

0.9
0.7
0.9

→ **Word "Joy"에 대한 sentiment score들을 embedding**



Approach



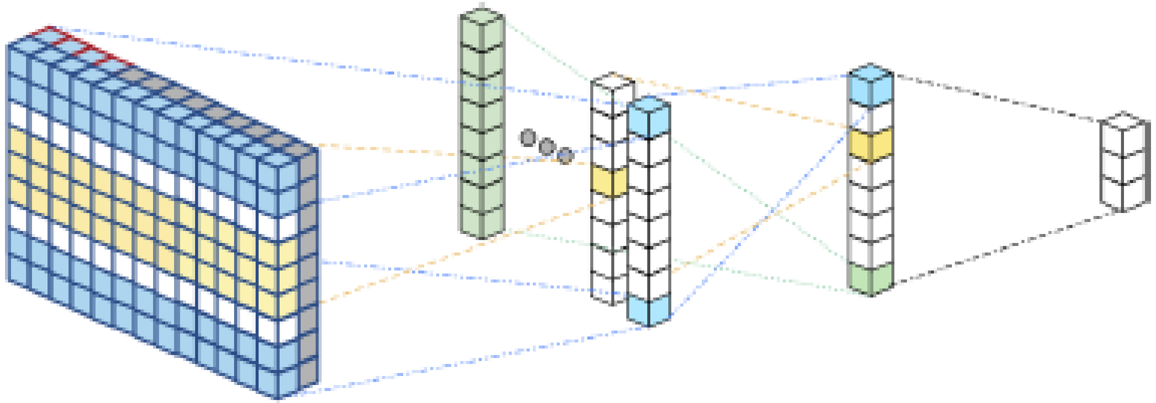
(a) Naive concatenation (Section 3.2.1). The lexicon embeddings (on the right) are concatenated to the word embeddings (on the left).

Naïve Concatenation

가장 간단한 방법으로,
Word Embedding과 Lexicon Embedding
을 **단순하게 합친다**

$s \in \mathbb{R}^{n \times d}$ → $s \in \mathbb{R}^{n \times (d+e)}$

Approach



(b) Multichannel (Section 3.2.2). The lexicon embeddings are added to the second channel whereas the word embeddings are added to the first channel.

Multichannel

Conv Layer에서 함께 Convolved 됨

$d \gg e$ 인 경우 나머지

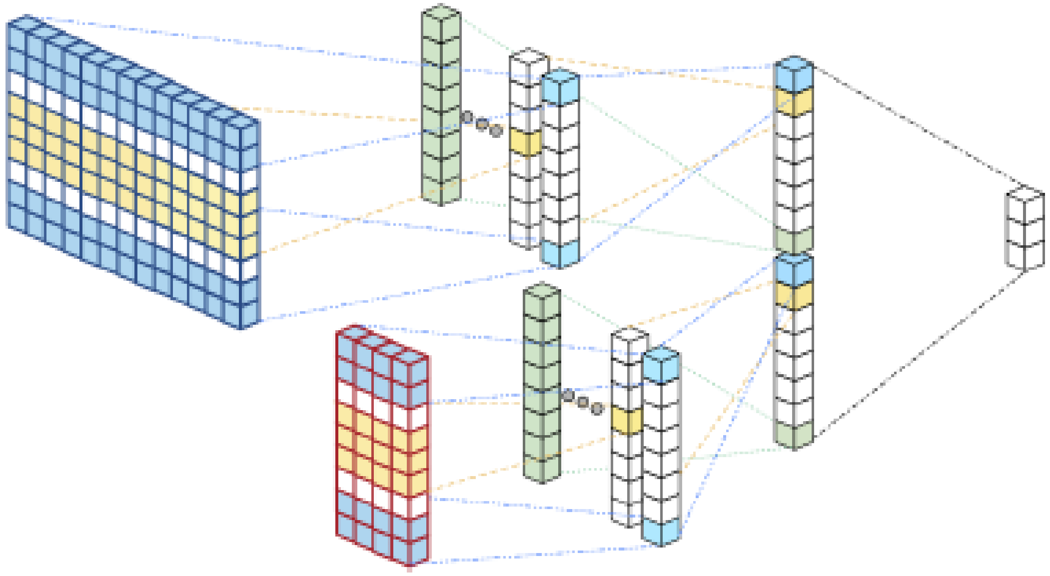
Lexicon embedding은 Zero padding

$s \in \mathbb{R}^{n \times d}$ → $s \in \mathbb{R}^{n \times d}$

\+ Channel추가

Approach

Size of 2nd layer: $[(n - l_w + 1) \times m_w] + [(n - l_x + 1) \times m_x]$



(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).

Separate Convolution

Word embedding, Lexicon embedding을 각각 따로 Convolution, max pooling 한 후 합쳐서 Softmax layer에 넣음

l_w, l_x 각각의 filter 길이

$m_w\ m_x$ 각각의 filter 개수

Approach

왜 Attention을 도입했나?

현재는 I-gram feature를 잡아내는 형태의 구조

즉, "account only for **local views, not the global view** of the document" 라 할 수 있음

현 구조의 단점

Negation이 있는 경우 local view만으로는

정확한 의미를 판별할 수 없음

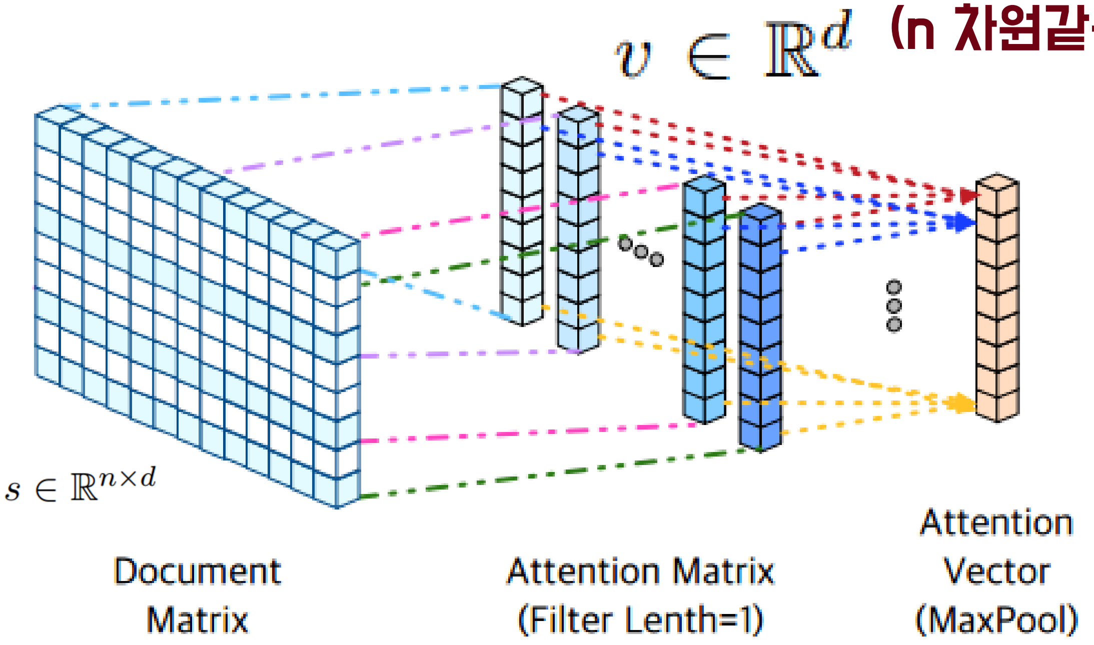
EAV로 해결해보자

Embedding Attention Vector

(EAV)

Approach

(n 차원같음)



(a) Give a document matrix, the attention matrix is first created by performing multiple convolutions. The attention vector is then created by performing max pooling on each row of the attention matrix.

Embedding Attention Vector

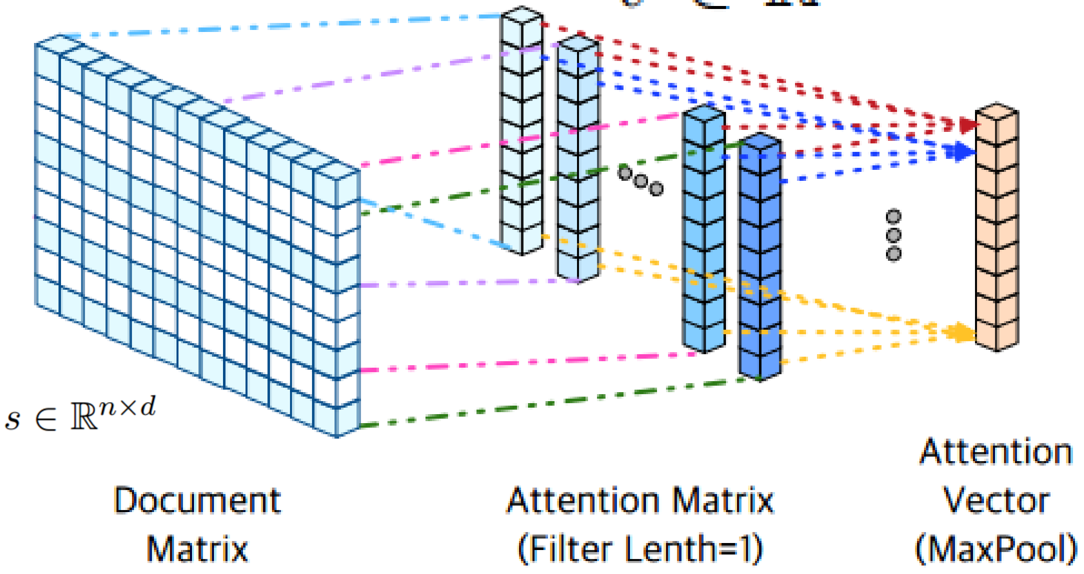
the EAV in the word embedding space is calculated as a weighted sum of each column in the document matrix

$$s \in \mathbb{R}^{n \times d}$$

Lexicon Integration 방법에 맞게 EAV도 WE, LE 각각에 대해서 두 개씩 생성

Approach

$v \in \mathbb{R}^d$ (n 차원같음)



(a) Give a document matrix, the attention matrix is first created by performing multiple convolutions. The attention vector is then created by performing max pooling on each row of the attention matrix.

Embedding Attention Vector

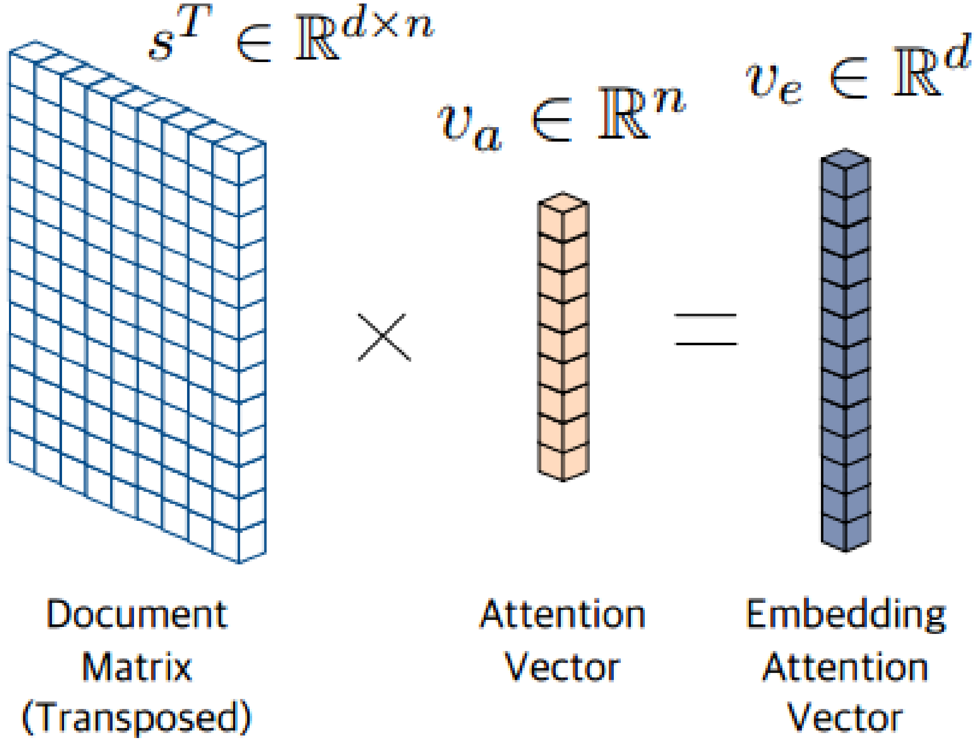
1. Filter length = 1 로 Convolve
2. Aggregate all conv outputs

-> Attention matrix $s_a \in \mathbb{R}^{n \times m}$

3. Execute max pooling for each row of attention matrix $s_a \in \mathbb{R}^{n \times m}$

-> Attention vector $v_a \in \mathbb{R}^n$

Approach



(b) The embedding attention vector is created by multiplying the transposed document matrix to the attention vector.

Embedding Attention Vector

4. Transpose the document matrix s

$s^T \in \mathbb{R}^{d \times n}$

and multiply it with $v_a \in \mathbb{R}^n$

= Embedding attention vector $v_e \in \mathbb{R}^d$

-> Penultimate layer(끝에서 두번째)에

Additional information으로 추가

Experiments

	+	0	-	All
TRN	6,480	6,577	2,328	15,385
DEV	786	548	254	1,588
TST	7,059	10,342	3,231	20,632

Table 1: Statistics of the SemEval'16 Task 4 dataset. +/0/-: positive/neutral/negative.

Dataset - SemEval-2016 Task 4a

Tweets with annotation of

Three sentiment classes

: positive, neutral, and negative

Evaluated

by **Averaging the F1-scores**

Experiments

	++	+	0	-	–	All
TRN	1288	2322	1624	2218	1092	8,544
DEV	165	279	229	289	139	1,101
TST	399	510	389	633	279	2,210

Table 2: Statistics of the Stanford Sentiment Treebank dataset. ++/+/0/-/–: very positive/positive/neutral/negative/very negative.

Dataset - Stanford Sentiment Treebank

Movie reviews from Rotten Tomatoes

-> Evaluating the robustness across different genres

Five classes: very positive, positive, neutral, negative, very negative.

Experiments

Word Embeddings

Word2vec (skip-gram, negative sampling) 사용

WE은 Domain에 영향을 받음

그렇기 때문에 pretrained model이 아닌

SemEval'16, SST Datasets으로 training

(3.67M word types),(2.67M word types)

Pre-tokenize는 NLP4J (Open source) 사용

Lexicon Embeddings

6 types of sentiment lexicons 사용

- National Research Council Canada (NRC)
- NRC Hashtag Sentiment Lexicon
- NRC Sentiment140 Lexicon
- Sentiment140 Lexicon
- MaxDiff Twitter Sentiment Lexicon
- Bing Liu Opinion Lexicon

(missing words는 neutral score로 0 부여)

Experiments

Word Embeddings

Word2vec (skip-gram

WE은 Domain에 영

그렇기 때문에 pretr

SemEval'16, SST I

Pre-tokenize는 NLP4J

Lexicon Embeddings

ent lexicons 사용

Council Canada (NRC)

iment Lexicon

0 Lexicon

icon

entiment Lexicon

Lexicon

(missing words는 neutral score로 0 부여)

	Word Emb		Lexicon Emb	
	S16	SST	S16	SST
TRN	70.12	97.66	11.53	9.20
DEV	81.90	98.91	3.29	3.32
TST	68.57	98.58	12.40	4.98

Table 3: The percentage of word types covered by our word and lexicon embeddings for each dataset.

Evaluation

7 models

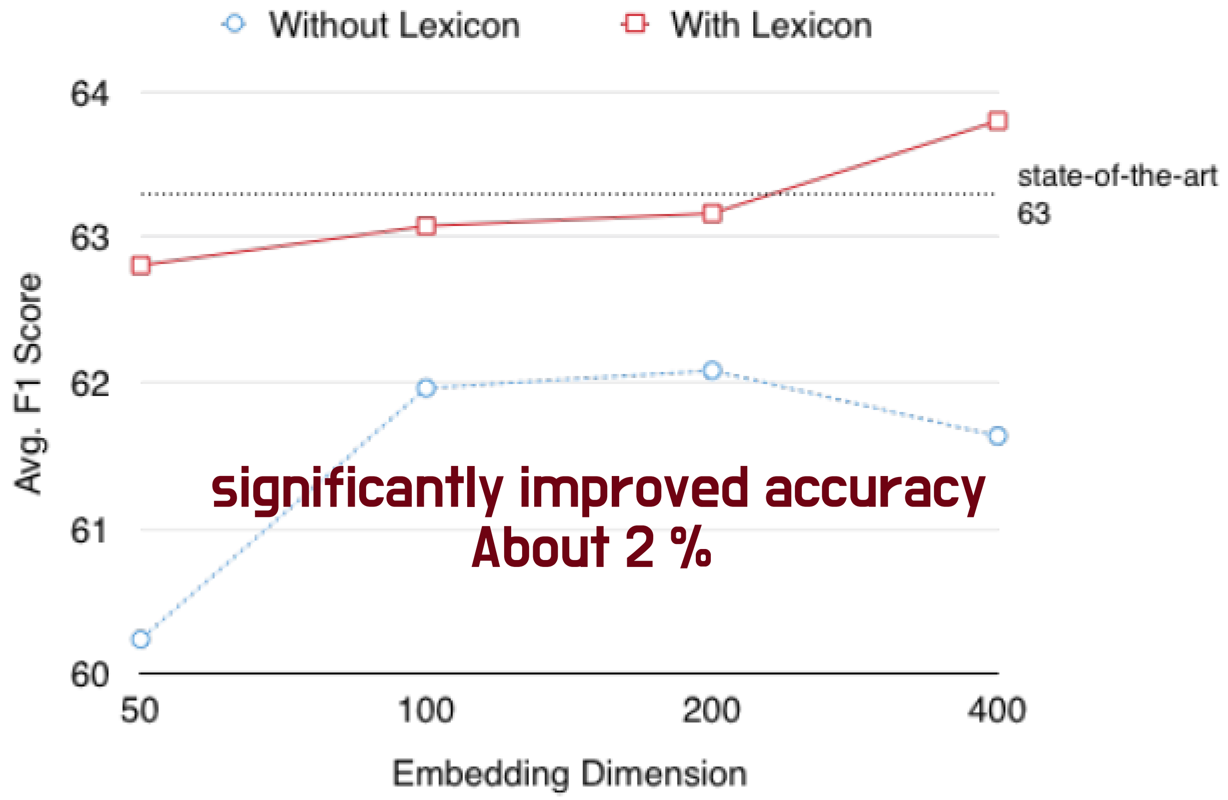
1. Naïve concatenation(NC)
2. Multichannel(MC)
3. Separate(SC)
4. EAV
5. NC+EAV
6. MC+EAV
7. SE+EAV

Configuration

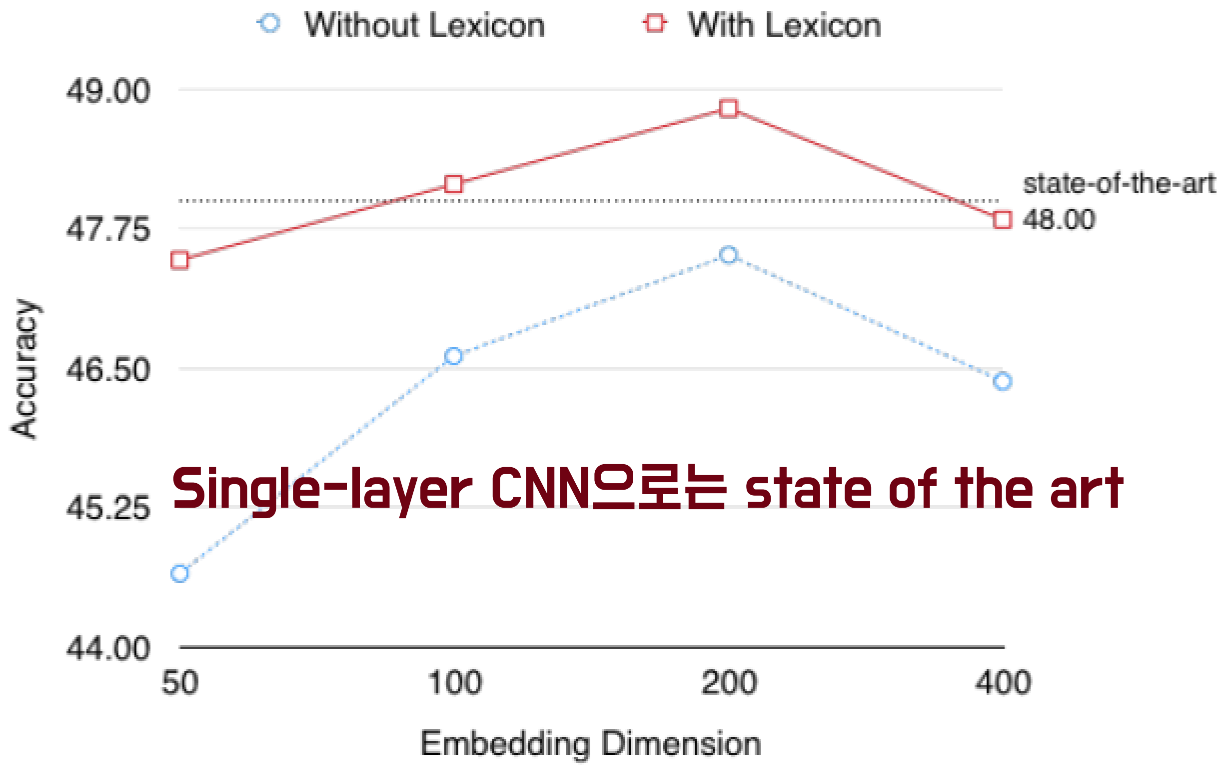
- Filter size = (2, 3, 4 ,5)
  for both word and lexicon
- Num of filters = (64 and 9)
  for both word and lexicon, respectively
- Num of filters = (50 and 20)
  for EAV in word and lexicon embedding space, respectively

Evaluation

F1 measure & Accuracy



SC case (a) SemEval Task

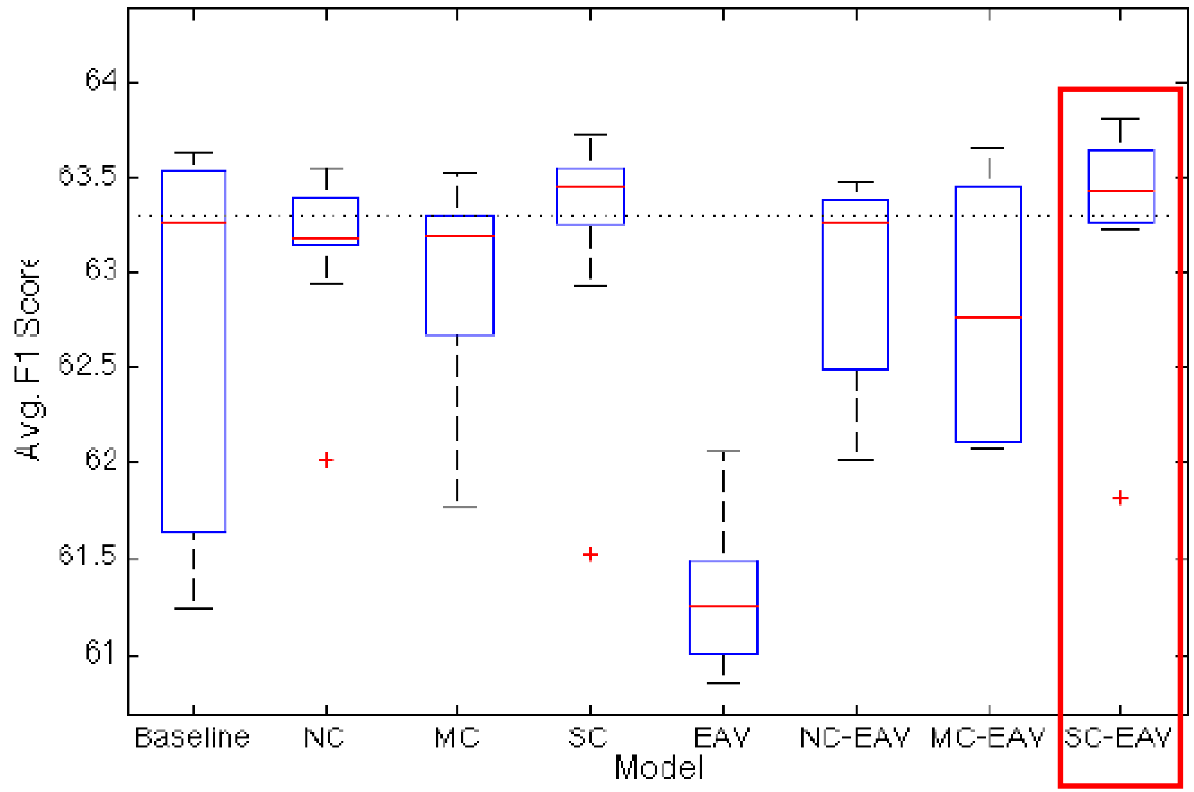


(b) SST Task

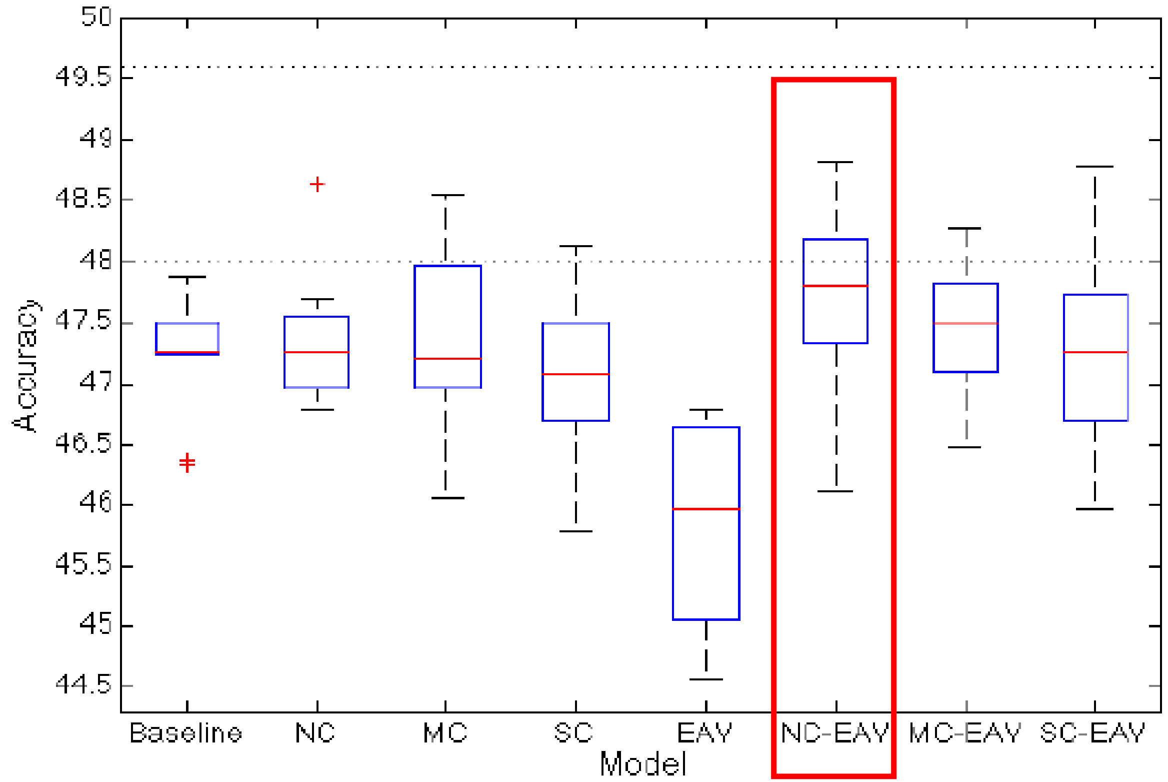
Lexicon integration은 WE dim에 robust

Evaluation

Box plot



SemEval'16 Task



SST Task

Evaluation

Negative / Positive

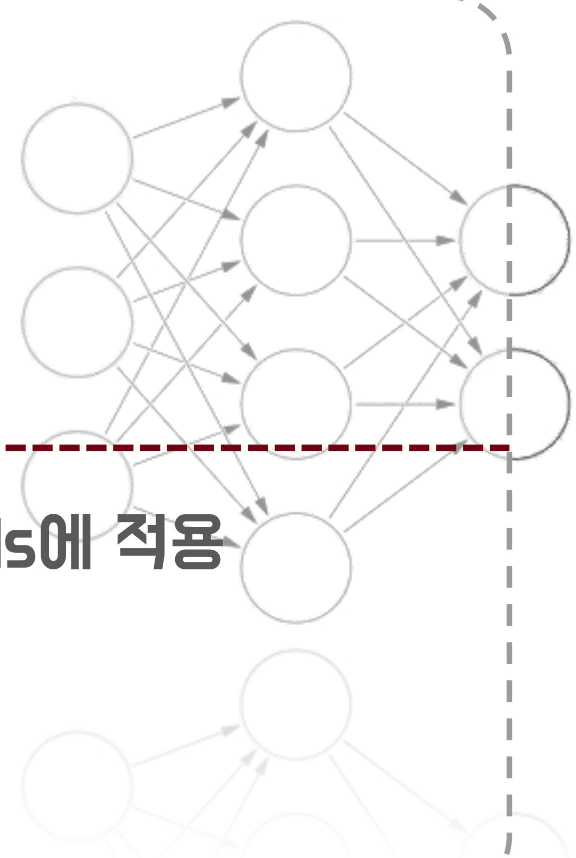
Figure 5: Five selected negative tweets with the attention heatmap. Examples are from the set where the baseline gives wrong answers but SC-EAV predicts correctly. Intensity of each word roughly ranges from -1 to 1. This weights (intensities) are the values of the attention vector of the word embeddings in the SC-EAV model. While negative words get more attention (dark reds), non-sentimental words such as stop words get less attention (greens and light blues).

Attention vector가 death, attack, sick등 부정적인 단어 잘 잡아냄

Conclusion & Future work

- Lexicon integration은 accuracy, stability, 관점에서 유용
- Attention mechanism을 통한 Attention heatmap 은 Explanatory feature 제공하며 accuracy 향상에 도움을 줌

- Attention model을 each single word 외에 multiple words에 적용
- More lexicon dataset 사용해서 coverage 개선
- Ensemble of multi layer CNN models



Q&A

Thank you ☺