

Skip-Thought Vectors

2015년 논문

Abstract

- generic, distributed sentence 인코더의 비지도 학습 방법 제시
- BookCorpus 데이터셋 사용
- 인코더-디코더 모델
- 주변 문장을 예측하는 방식 (skip-gram의 sentence-level 버전)
- Vocabulary expansion (for 학습 시 나오지 않은 단어 encode)
- 8 tasks를 통해 평가
- off-the-shelf encoder (highly generic, robust, perform well)

Introduction

- word vec 에서 sentence vec 발전
recursive, recurrent, conv, recursive-conv, ...
⇒ 특정 task에 특화되어 학습
- 특정 task 없이 high-quality representation 학습할 수 있는 loss function 고려
⇒ skip-gram을 sentence-level로 생각
- BookCorpus 데이터셋 사용
 - 16 장르 (로맨스-2865개, 판타지-1479개, SF-786개, Teen-430개, 등등)
 - 특정 도메인이나 application에 치우쳐있지 않음

# of books	# of sentences	# of words	# of unique words	mean # of words per sentence
11,038	74,004,228	984,846,357	1,316,420	13

Table 1: Summary statistics of the **BookCorpus dataset** [9]. We use this corpus to training our model.

- 평가

- 모델을 학습시키고 인코더를 task의 feature extractor로 사용
- skip-thought vectors를 추출하고, **linear** model 학습
- 추가적인 **fine-tuning X**

⇒ representation 직접적인 평가 위해

- 어려움
 - 충분히 큰 단어 사전 구성하기 ex. # of Wikipedia words > # of our book words
 - ⇒ linear mapping : pre-trained w2v → 인코더의 vocabulary 학습

Approach

1. Inducing skip-thought vectors

- 인코더-디코더 형태
- RNN encoder with GRU, RNN decoder with a conditional GRU 사용
- 표기

$$\begin{aligned}
 & \text{sentence tuple} : (s_{i-1}, s_i, s_{i+1}) \\
 & w_i^t : t - th \text{ word for sentence } s_i \\
 & x_i^t : its \text{ word embedding}
 \end{aligned}$$

- 3 부분 : 인코더, 디코더, 목적 함수

인코더

- 역할 : words → sentence vector로 map

$$\begin{aligned}
\mathbf{r}^t &= \sigma(\mathbf{W}_r \mathbf{x}^t + \mathbf{U}_r \mathbf{h}^{t-1}) \\
\mathbf{z}^t &= \sigma(\mathbf{W}_z \mathbf{x}^t + \mathbf{U}_z \mathbf{h}^{t-1}) \\
\bar{\mathbf{h}}^t &= \tanh(\mathbf{W} \mathbf{x}^t + \mathbf{U}(\mathbf{r}^t \odot \mathbf{h}^{t-1})) \\
\mathbf{h}^t &= (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \bar{\mathbf{h}}^t
\end{aligned}$$

디코더

- 역할 : 언어 모델 \Rightarrow 인코더의 output으로 주변 문장들 예측(생성)
- 2개 디코더 사용 : 이전 문장, 다음 문장 예측

$$\begin{aligned}
\mathbf{r}^t &= \sigma(\mathbf{W}_r^d \mathbf{x}^{t-1} + \mathbf{U}_r^d \mathbf{h}^{t-1} + \mathbf{C}_r \mathbf{h}_i) \\
\mathbf{z}^t &= \sigma(\mathbf{W}_z^d \mathbf{x}^{t-1} + \mathbf{U}_z^d \mathbf{h}^{t-1} + \mathbf{C}_z \mathbf{h}_i) \\
\bar{\mathbf{h}}^t &= \tanh(\mathbf{W}^d \mathbf{x}^{t-1} + \mathbf{U}^d(\mathbf{r}^t \odot \mathbf{h}^{t-1}) + \mathbf{C} \mathbf{h}_i) \\
\mathbf{h}_{i+1}^t &= (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \bar{\mathbf{h}}^t
\end{aligned}$$

- 언어모델

$$P(w_{i+1}^t | w_{i+1}^{<t}, h_i) \propto \exp(v_{w_{i+1}^t} h_{i+1}^t)$$

where,

V : 사전(voca) 행렬(weight matrix)

$v_{w_{i+1}^t}$: w_{i+1}^t 단어에 해당되는 V 의 row

목적 함수

- (s_{i-1}, s_i, s_{i+1}) 에 대한 objective

$$\sum_t \log P(w_{i+1}^t | w_{i+1}^{<t}, h_i) + \sum_t \log P(w_{i-1}^t | w_{i-1}^{<t}, h_i)$$

- total objective : 모든 training tuple에 대해 위의 값을 sum

2. Vocabulary expansion

- 논문 방법 : linear model 사용

$$\begin{aligned} V_{w2v} &: \text{pre-trained } w2v \text{ space} \\ V_{rnn} &: \text{model's word embedding space} \\ \text{mapping } f &: V_{w2v} \rightarrow V_{rnn} \\ \text{s.t. } f(v) &= Wv \\ \text{, where } v &\in V_{w2v} \\ f(v) &\in V_{rnn} \end{aligned}$$

- 학습

$$\text{minimize } \|v_{rnn} - Wv\|^2$$

- 다른 방법
 - pre-trained word vector로 initialize \Rightarrow decoding 계산 복잡, clipping the voca 필요
 - character-level로 학습

Experiments

- 활용
 - 학습된 인코더를 feature extractor로 사용
 - 문장 간의 점수 계산 필요하면 component-wise로 계산 (product, absolute difference)
 - 분류 task는 추가적 fine-tuning하지 않고 linear classifier로 이어서 학습

Details of training

- 모델 : uni-skip, bi-skip, combine-skip

- Uni-skip
 - 인코더 with 2400 dim
- Bi-skip
 - forward 인코더 with 1200 dim, backward 인코더 with 1200 dim \Rightarrow concat해서 사용
- combine-skip
 - 학습된 uni-skip + 학습된 bi-skip \Rightarrow 4800 dim
- Initialization
 - recurrent matrices : orthogonal initialization
 - None-recurrent weights : uniform distribution in $[-0.1, 0.1]$
- 미니 배치 사이즈 : 128
- gradient clipping when norm of the parameter vector exceeds 10
- optimizer : Adam
- vocab size : 20,000 words \Rightarrow vocabulary expansion 후 930,311 words 가능
- 새로운 문장의 전처리 최소화 (토큰화) for robust 확인

코드

- TF 버전

[tensorflow/models](https://github.com/tensorflow/models/tree/master/research/skip_thoughts)

https://github.com/tensorflow/models/tree/master/research/skip_thoughts