Text Classification & Sentence Representation, 강의 내용 정리

Index

Overview

Text Classification

Text classification이란 사람의 언어로된 문장이나 문단을 input으로 받아 그 input이 어느 카테고리에 해당되는지를 output으로 출력하는 작업이다(카테고리의 개수는 정해져있다고 생각한다).

예시)

- Sentiment analysis, 감정분석: 주어진 리뷰가 긍정적인 리뷰인지 부정적인 리뷰인지 판단한다.
- Text categorization, 문서분류: 주어진 게시글이 어느 카테고리에 속해있는지 판단한다.
- Intent classification, 의도분석: 주어진 질문이 어떤 것과 관련이 있는지 판단한다.

How to represent sentence & token?

문장이란 토큰으로 이루어진 가변길이의 sequence이다. $X = (x_1, x_2, \cdots, x_T)$. 각각의 토큰들은 사전에 속해있는 단어 중 하나이다. $x_t \in V$. 한번 사전이 구축되고 index encoding된다면 문장은 integer index로 이루어진 sequence일 뿐이다.

How to represent a token

Dog, wolf, cat, tiger처럼 각 단어가 왜 저런 알파벳의 나열인지 알 수 없다. 이 단어들의 생김새가 서로에게 영향이 없다는 것을 반영하여 one-hot encoding을 할 수 있다. 즉, token은 하나의 index를 부여받는다. 여기서 token의 encoding이 token의 의미를 반영하게 하는 방법 알아보자.

1. 처음에 우리는 우리의 말뭉치 안 단어에 대해 아는 것이 없다고 가정한다. 따라서 one-hot encoding을 이용한다.
    - $x = [0, 0, \cdots, 0, 1, 0, \cdots, 0] \in 0, 1^{|V|}$
        - |V|: the size of vocabulary
        - Only one of the elements is 1: $\sum_{i=1}^{|V|} x_i = 1$
    - Every token is equally distant away from all the others.
        - $||x - y|| = c \geq 0, if x \neq y$
2. 이후에 뉴럴 네트워크가 token의 의미를 벡터로 만들 수 있게 한다.
    1. 이는 행렬곱으로 쉽게 만들 수 있다.
        - $Wx = W[\hat{x}]$, if x is one-hot, where $\hat{x} = argmax_j x_j$ is the token's index is the vocabulary.

2. 이 작업을 'table-lookup layer'라고 한다.
3. table-lookup 연산 후에, input 문장은 continuous, high-dimensional 벡터의 sequence가 된다.
    - $X = (e_1, e_2, \cdots, e_T),\ \text{where } e_t \in \mathbb{R}^d$
    - The sentence length T differs from one sentence to another.

분류기는 input 문장이 변형된 vector의 sequence를 하나의 벡터로 압축하는 일을 한다.

How to represent sentence? CBoW & RN & CNN

문장의 의미를 representation하는 방법에 대표적인 방법이 있지는 않다. 풀고 싶은 문제에 따라 적합한 representation 방법이 있고 이를 적용해야한다.

CBoW, Continuous bag-of-words

구성이 아주 간단하고 계산 비용도 싸며 어느 정도 정확도를 가지고 있다. 하지만 order를 전혀 고려하지 못한다.

- token의 순서를 없앤다. $(x_1, x_2, \cdots, x_T) \rightarrow \{x_1, x_2, \cdots, x_T\}$
- 간단히 token의 평균을 구한다.:
    - Averaging is a differentiable operator. $\frac{1}{T}\sum_{t=1}^{T} e_t$
- bag-of-n-gram으로 확장할 수 있다.
    - N-gram: a phrase of N tokens
- text classification에서 아주 효율적이다.
    - 예를 들어, 만약 긍정적인 단어가 많은 문장이라면 긍정적일 가능성이 높다.

RN, Relation Network: Skip Bigrams

계산 비용은 CBoW보다 비싸지만, 그래도 쌍을 이루는 단어에 대한 뜻은 잘 파악할 수 있다.

- 모든 token들의 pair를 생각한다. $(x_i, x_j) \forall i \neq j$
- 한 pair의 두 token 벡터를 뉴럴 네트워크를 이용하여 합친다.
    - $f(x_i, x_j) = W\phi(U_{left}e_i + U_{right}e_j)$
    - ϕ is a element-wise nonlinear function, such as tanh or ReLU
- 이를 pair의 relationship vector라 하고 이를 합해서 평균내면 RN의 출력 vector가 된다.
    - $RN(X) = \frac{2}{N(N-1)}\sum_{i=1}^{T-1}\sum_{j=i+1}^{T} f(x_i, x_j)$
- 계산 비용이 많이 들겠지만 pair 뿐만 아니라 3개 혹은 4개 이상의 묶음도 생각할 수 있다.

CNN, Convolutional Neural Network

K-grams의 특성을 계층적으로 잡아낸다.

- One 1-D convolutional layer: considers all k-grams
    - $h_t = \phi(\sum_{\gamma=-k/2}^{k/2} W_\gamma e_{t+\gamma}),\ \text{resulting in } H = (h_1, h_2, \cdots, h_T).$
- 하나의 convolutional layer을 쌓으므로서 점진적으로 윈도우를 증가시킨다.
- CNN은 우리 사고랑 잘 맡는다. tokens -> mulit-word expressions -> phrases -> sentence
- 실용적 관점에서는 DAG에 그저 다른 operation일 뿐이다.
- Recent advances
    - Multi-width convolutional layers [Kim, 2014; Lee et al., 2017]
    - Dilated convolutional layers [Kalchbrenner et al., 2016]

- Gated convolutional layers [Gehring et al, 2017]

How to represent a sentence? Self Attention & RNN

Self-Attention

CNN의 단점: 아주 긴 sequence의 첫번째와 마지막 단어가 dependency가 있다면 CNN이 캡처하기 어렵다.

RN의 단점: 문장에 대한 모든 단어의 순서쌍이 가지고 있는 가중치가 모두 1이다.

- 먼저 t번째 단어가 CNN처럼 주변단어만 보는 것이 아니라 모든 단어를 다 본다.
  - $h_t = \sum_{t'=1}^{T} \alpha(x_t, x_{t'}) f(x_t, x_{t'})$
  - weighting function α가 neural network가 될 수 있다.
    - 쉬운 방법중 하나는 다음과 같다.
      - $\alpha(x_t, x_{t'}) = \sigma(f(x_t, x_{t'}))$ where f is in RN
    - normalization된 weights는 다음과 같다.
      - $\alpha(x_t, x_{t'}) = \frac{exp(f(x_t, x_{t'}))}{\sum_{t''=1}^{T} exp(f(x_t, x_{t''}))}$, where f is in RN
- Self-attention는 CNN과 RN을 확장한 것이라 생각할 수 있다.
- single layer를 이용해 long-range depenencies를 잡아낼 수 있다.
- 또, irrelevent long-range depenencies를 무시할 수 있다.
- multi-head와 multi-head attention을 통해 더 확장할 수 있다.
- self-attention의 약점:
  - Quadratic computational complexity, $O(T^2)$

RNN

- Linear compuational complexity, $O(T)$
  - $h_t = RNN(h_{t-1}, x_t)$, where $h_0 = 0$.
- 문장이 길어지면 하나의 vector로 compress하기 어렵다.
- Bidirectional RNN을 이용하여 양쪽 방향에서 확인할 수는 있다.
- 상속적이고 연속적인 프로세싱이다.
  - 문장의 한 단어 씩 봐야하기 때문에 분산 계산을 할 수 없다.
- 대표적으로 LSTM과 GRU가 있다.

Summary

How to represent a sentence?

- 이 강의에서 5개의 sentence representation 방법을 학습했다.
  - CBoW, RN을 제외하고 나머지 방법들은 sentence representation이 vector로 표현되었다. $H = \{h_1, \cdots, h_T\}$
    - 이러한 방법들은 성능을 향상 시키기 위해 임의의 방법으로 stacked될 수 있다.
    - 이러한 벡터들은 평균을 내어 분류에 사용될 수 있다.