1. The course

- technics for NLP
- A big picture understanding of human languages + 구현이 어려운 이유
- PyTorch를 이용한 NLP의 주요문제 이해 + 구현

2. Human language and word meanining

언어의 사회성 — 컴퓨터에게 어려운 이유

인간 언어는 굉장히 최근에 생김 부족의 다른 구성원과의 소통을 원활히 해줌

글쓰기 발달 — 거리를 넘어서 소통

GPT-3가 하는 일은 다음에 올 단어를 찾는 것. SQL같은 언어도 유창

언어를 컴퓨터에게 이해시키기

- Common NLP solution ; WordNet (유의어와 상위어가 한 set에 있는 사전)같은 thesaurus (유의어사전)를 만듦
  - ↳ 한계 : 상황에 따른 뉘앙스를 알 수 없음
  - 단어의 새로운 의미는 〃 → 추복하기 위해선 노동이 많이 필요

traditional NLP의 문제

- 단어가 별개의 symbol로 여겨짐

  one-hot encoding을 사용하면 단어 하나 당 하나의 벡터로 바꿔주기 때문에 Vector dimension이 너무 커짐
  = 단어의 수

  게다가 vector 끼리의 similarity를 표현할 방법X

  → (과거의 시도) WordNet의 유의어 리스트를 이용해보려 했으나 실패

  → vector 자체에 similarity를 담는 방법을 배워보자

  <Distributional Semantics>

  '단어의 의미는 자주 문장 내에서 가까이 쓰이는 단어들로 결정된다'는 아이디어에서 출발

  $banking = \begin{bmatrix} 1.2 \\ 0.3 \\ 2.5 \end{bmatrix}$ 처럼 n-dimensional vector로 나타냄
  ↳ 실제로는 약 300

  이런 Word Vectors는 Word Embeddings, (neural) word representations 라고도 부름

  유의어들끼리 가까운 거리에 있도록 설정

  이게 바로 word2vec!

3. Word2vec Introduction

center word c 가 주어졌을 때 context ("outside") words o 가 문장 내에 나올 확률을 알아냄

계산된 p를 이용해 c의 vector 값에 맞춰 o의 vector를 정함

그리고 o를 center word로 바꾸고 그 옆 단어를 context word로 정함

4. Word2vec objective function gradients

* objective function = cost f = loss f

Likelihood $= L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P(W_{t+j} \mid W_t ; \theta)$ ↱ conditional probability

objective function $= J(\theta) = -\frac{1}{T} \log L(\theta)$

goal : maximize the likelihood (= minimize the objective function)

$* \; P(o \mid c) = \dfrac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$

음수를 피하기 위해 제곱

dot product ∝ similarity
$u^T v = u \cdot v = \sum_{i=1}^{n} u_i \cdot v_i$

$R^n \rightarrow (0,1)^n$ 인 softmax function가 바로 이것! (scaling해준다는 뜻)
제곱하니까 큰 수를 강조하게 됨. max의 일종

* max는 최댓값을 반환하지만 softmax는 정규분포 그 자체를 반환

5. Optimization basics

$\theta = \begin{bmatrix} v_{aa} \\ \vdots \\ u_z \end{bmatrix} \in \mathbb{R}^{2dV}$ (V: 단어 수) 는 모델의 모든 parameter가 담긴 하나의 긴 vector

우리가 배우는 모델에서 θ는 word vector 하나 뿐

결국 수학적으로 모든 vector gradients를 계산해 objective f 을 최소화해야 함

* 모든 단어는 center vector와 context vector 2개를 가진다

이유 : 하나만 있으면 한 단어를 center vector와 context vector로 동시에 쓸 때 미분이 못 생겨짐

계속 계산하다 보면 두 벡터는 비슷해진다 (동일 x)

<Gradient>

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m \\ j\ne 0}} \log P(w_{t+j} \mid w_t)$$

$$P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

* $\exp(n) = e^n$

inverses

$$\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V}\exp(u_w^T v_c)} = \frac{\partial}{\partial v_c}\log \exp(u_o^T v_c) - \frac{\partial}{\partial v_c}\log\sum_{w=1}^{V}\exp(u_w^T v_c)$$

① ②

① $$\frac{\partial}{\partial v_c} u_o^T v_c = \frac{\partial}{\partial v_c}(u_{o_1}v_{c_1} + u_{o_2}v_{c_2} + \cdots + u_{o_d}v_{c_d})$$

↑ vector

$$= u_{o_1} + u_{o_2} + \cdots + v_{o_d} = u_o$$

② $$\frac{\partial}{\partial v_c}\log\sum_{w=1}^{V}\exp(u_w^T v_c)$$ $f(g(x))$에 chain rule을 이용할 것

f, $z = g(v_c)$

$h(x) = f(g(x))$

$h'(x) = f'(g(x))g'(x)$

f, $z = g(v_c)$ 다시 이용

$$= \frac{1}{\sum_{w=1}^{V}\exp(u_w^T v_c)} \times \frac{\partial}{\partial v_c}\sum_{x=1}^{V}\exp(u_x^T v_c)$$

여기서 index를 바꾸지 않으면 곤란해짐

$$= \quad '' \quad \times \sum_{x=1}^{V}\frac{\partial}{\partial v_c}\exp(u_x^T v_c)$$

$$= \quad '' \quad \times \sum_{x=1}^{V}\exp(u_x^T v_c)\cdot u_x$$

$$\therefore \frac{\partial}{\partial v_c}\log p(o \mid c) = u_o - \frac{\sum_{x=1}^{V}\exp(u_x^T v_c)\cdot u_x}{\sum_{w=1}^{V}\exp(u_w^T v_c)}$$

$$= u_o - \sum_{x=1}^{V}\frac{\exp(u_x^T v_c)}{\sum_{w=1}^{V}\exp(u_w^T v_c)}u_x$$ softmax function

$$= u_o - \sum_{x=1}^{V} p(x \mid c)u_x$$ expectation $E[X] = p(x)\cdot X$

= observed − expected (즉, $y - y_{pred}$)