Lec 2. Word Vectors, Word Senses, and Neural Classifiers

- **Optimization : Gradient Descent**

  : algorithm to minimize $J(\theta)$ by changing θ iteratively

  [when learning step size is too small → waste of time & computation
  [" " " " too big → diverge or bounce back and forth

  - $\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$ * α : step size / learning rate

    $\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$ for single param.

  - problem : $\nabla_\theta J(\theta)$ is very expensive to compute (모든 window 에 대한 function 이므로..)

    ⇒ **SGD (Stochastic Gradient Descent)**

    : window를 sample, 각 mini batch 마다 업데이트 반복

    → sparsity 문제 (sample된 벡터만 업데이트하므로)

- **Negative Sampling**

  - word2vec의 2가지 model [skip-grams (SG)
    [continuous bag of words (CBOW)

  - skip-gram 모델의 $p(o|c) = \frac{\exp(u_o^T v_c)}{\underbrace{\sum_{w \in V} \exp(u_w^T v_c)}_{\text{computationally expensive (계산량 多)}}}$

  ⇒ negative sampling : true pair (center & its context words) vs noise pairs (center word paired w/ random word)
  에 대해 이진 로지스틱 회귀 방식으로 train.

  $$J_t(\theta) = \log \sigma(u_o^T v_c) + \underbrace{\sum_{t=1}^{K} E_{j \sim P(w)} [\log \sigma(-u_j^T v_c)]}_{\text{K개의 negative sample } (P(w) = U(w)^{3/4}/Z)}$$

  outside word 등장 확률 maximize, random word 등장 확률 minimize

- **co-occurence matrix**

  - window based / full document 2가지 방식
  - 단어의 개수 ↑ 일수록 dimension ↑
    → sparse & less robust

  ⇒ SVD (Singular Value Decomposition) 을 이용하여 차원축소
  $X = U\Sigma V^T$

  자주 등장하는 function word 를 scale 하면 도움이 됨.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

- **word prediction methods**

  - count based : 학습속도 빠름 / 통계 정보로 학습 수행
    단어간 유사도 사용 / 빈도 높은 단어에 대해 과도하게 높은 중요도 부여
    ex) LSA, HAL, COALS, Hellinger-PCA
  - direct prediction : corpus size로 scale / 전체에 대한 통계정보 사용 X
    다양한 용도로 사용 가능 / 단어간 유사도 이상의 패턴정보

- **GloVe**

  - count 기반 & prediction 기반을 합침
  - capture ratios of co-occurrence probabilities as linear meaning components

    using log-bilinear model with vector differences.

    $w_i \cdot w_j \approx \log P(i|j)$

    두 단어벡터의 내적 ≈ 두 단어의 동시 등장 확률

    $w_x(w_a - w_b) = \log \frac{P(x|a)}{P(x|b)}$

  - $J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$

    https://wikidocs.net/22885 참고하기..

    * $f(X_{ij})$ →  ⇒ scalable to huge corpora

- **How to evaluate word vectors?**

  - Intrinsic : specific / intermediate task
    - 계산 속도 빠름
    - analogy evaluation
  - Extrinsic : real task
    - 계산속도 느림
    - NER (named entity recognition)

- **word senses and word sense ambiguity**

  - 단어가 가진 다양한 뜻을 벡터 하나만으로 표현하기 어려움

    → 단어의 의미별로 다르게 clustering https://aclanthology.org/P12-1092/

    → weighted sum 활용 https://aclanthology.org/Q18-1034/