Word Embeddings

Made by Jay Hong

Check

Word Embeddings

1. Vectorization & Word Embedding?
2. Word 2 Vector?
3. Sub-word Embedding?

1. Vectorization & Word Embdding?

1. One-Hot Encoding
2. Vectorization
3. Word Embedding

One-hot Encoding

- ✓ 하나만 1이고 나머지는 0으로 만드는 방법

- ✓ Example)
  - ✓ Cat, Kitty, Dog, Puppy
  - ✓ Cat -> [1,0,0,0]
  - ✓ Kitty -> [0,1,0,0]
  - ✓ Dog -> [0,0,1,0]
  - ✓ Puppy -> [0,0,0,1]

Term-Document Matrix

- ✓ 모든 단어를 별개의 Token으로 보고, 해당 Token이 몇 회 나왔는지 세는 방법

- ✓ 특징
  - ✓ 비슷한 문장은 비슷한 단어의 반복 횟수를 가짐
  - ✓ 그럼 그 문장들 사이의 유사도는 높아짐
  - ✓ 유사도 계산은 내적, cosine similarity 등을 사용

Matrix Factorization

- ✓ TF Matrix를 SVD 적용
- ✓ 차원이 줄고, 성능이 좋아지긴 함
- ✓ 거의 사용 X
- ✓ SVD – S
  - ✓ SVD에 Scailing을 적용 -> Scailing을 많이 하자
- ✓ SVD – L (COALS)
  - ✓ Log 변환
  - ✓ Min(x,t) = 100 으로 변환
  - ✓ 구조적 역할 (a, the, an … 등) 제거

Probabilistic Topic Modeling

- ✓ LDA를 적용한 방법
- ✓ 거의 사용 X

Distributed Vector Representation

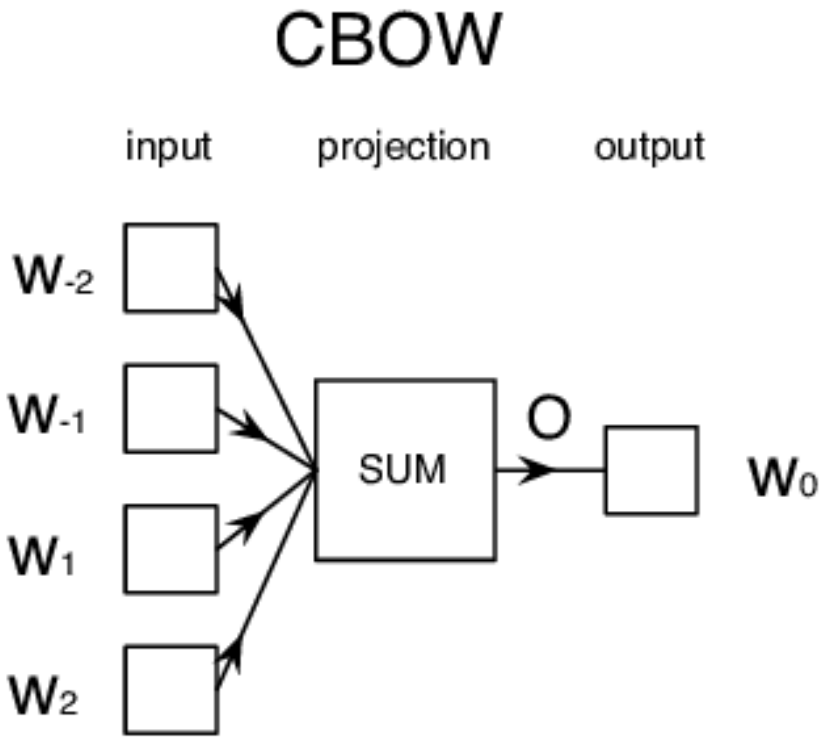
- ✓ Vector를 고차원 공간에 Non-zero 하게 분포시킨 방법
- ✓ 두 단어 사이의 의미론적 해석이 가능해짐

2. Word 2 Vector

1. Word 2 Vector
2. Skip-gram
3. GloVe
4. Doc2Vec

Idea

- ✓ 중심 단어 c는 주변 단어 o에 의해 구성됨
- ✓ 단어는 비슷한 문맥 속에서 비슷한 의미를 지닌다.



특징

- ✓ One-Hot은 해당 단어를 정확히 묘사하기에 그 의미를 갖는다면,
  W2V는 단어에 대한 추정치로, 그 의미가 희미해짐

- ✓ Vector이기에, 의미를 선형적으로 해석하기엔 좋음
  - ✓ Man : Woman :: King : Queen

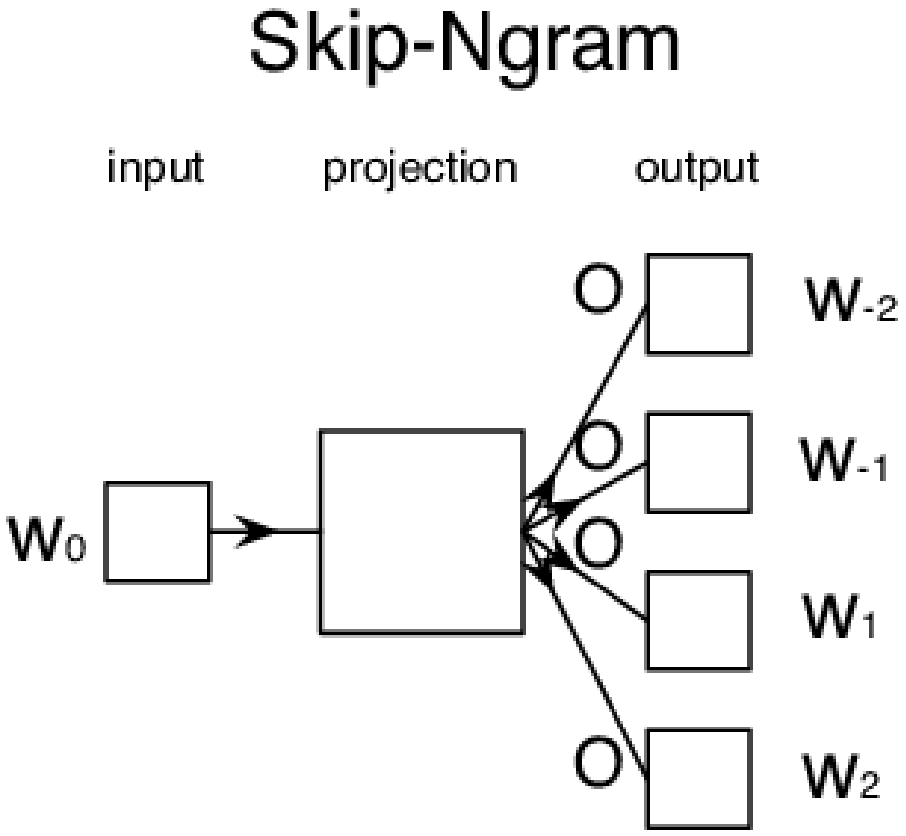
- ✓ 이를 이용해서, 새로운 단어들을 찾아나갈 수 있음

수식

- ✓ Objective Function :
  - ✓ exp(중심단어의 vector * 문맥단어의 vector) -> softmax
  - ✓ 모든 단어들에 대해서, 중심 단어가 나타날 확률에 대한 계산
- ✓ Optimize Function
  - ✓ Observed – Prediction
  - ✓ 실제값과 예측 값의 차를 Loss로 Optimization진행

Idea

- ✓ Word 2 Vector의 다른 분야
- ✓ 중심단어 c를 이용해 문맥 단어 o를 구성하자.



학습

- ✓ 모든 단어에 대해서, 중심 단어 c가 나타났을 때 문맥 단어 o가 나타날 확률 계산하자
  - ✓ 학습 속도가 매우 느림.

학습속도 보완

- ✓ Hierarchical Softmax -> Tree 형태로 학습 (거의 사용 X)
- ✓ Negative Sampling
  - ✓ 문제를 “모든 단어에 대해 문맥단어가 나타날 확률”인 Softmax에서
    “Sample 된 단어 O가 같은 문맥에서 나타날 확률”인 Binary 문제로 바꿈
  - ✓ 같은 문맥에서 나온 단어 -> True Case
    다른 문맥에서 나온 단어 -> Negative Case
    이를 Binary Cross Entropy를 이용해서 학습
  - ✓ 학습 시간을 엄청나게 낮춤
- ✓ 모든 단어 Update -> 등장하는 단어만 Update
  - ✓ 모든 단어의 W를 업데이트하는 비용이 너무 크기에, 등장한 단어들을 indexing해 Update하자.

Word 2 Vector 적용 분야

- ✓ 단어간 유사도 계산
- ✓ 기계 번역
- ✓ POS, NER
- ✓ 감성 분석
- ✓ Clustering
- ✓ Semantic Lexicon Building

Idea

- ✓ Global Vectors for Word Representation

- ✓ 문장단위의 Local로 보는 것이 아닌,
  문장을 통틀어 통계치를 이용해 Global 하게 바라보자.

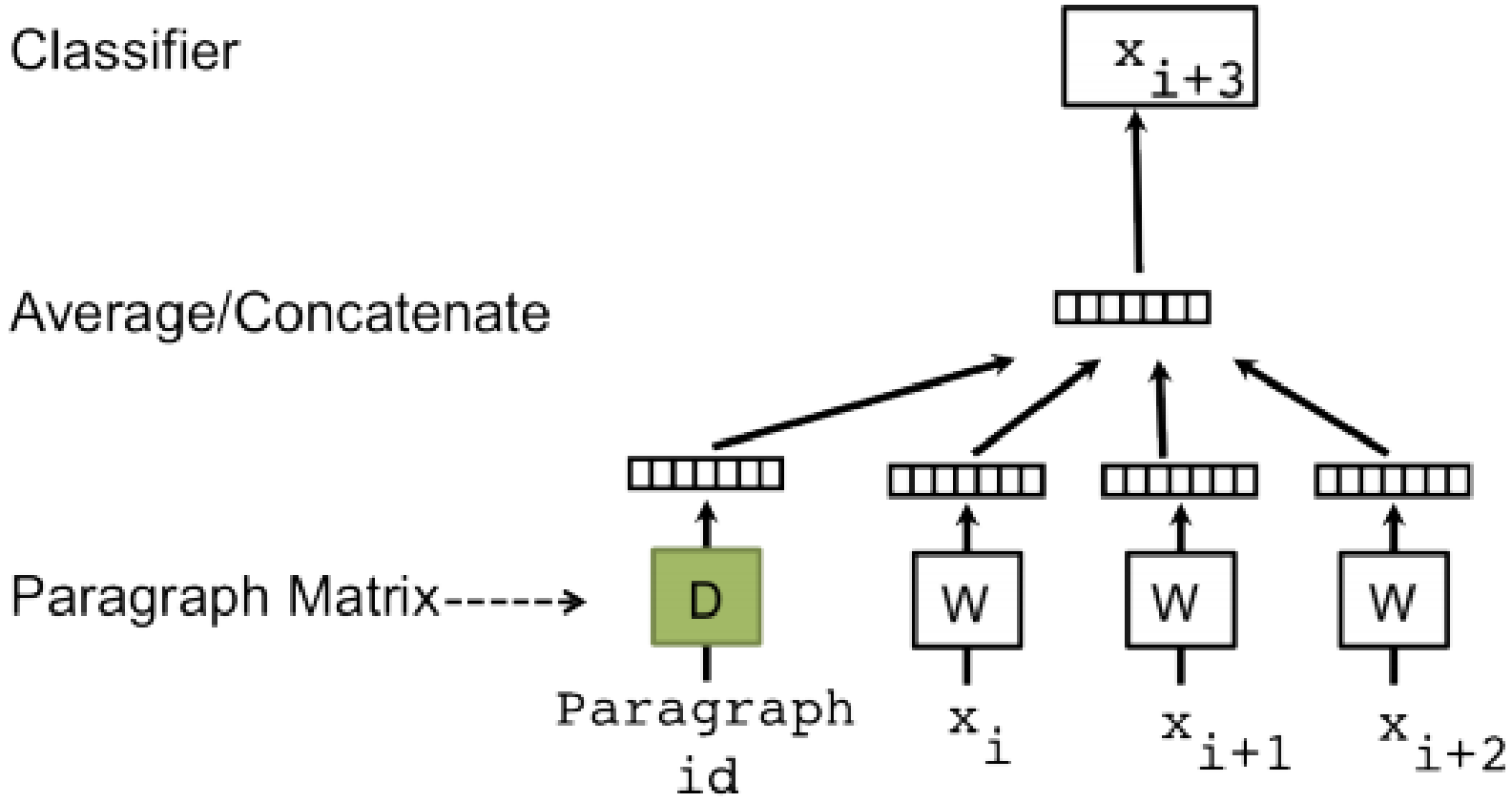
- ✓ 동시 발생 matrix를 이용한 word embedding
  - ✓ 학습 속도 향상
  - ✓ 적은 속도로도 잘 학습

- ✓ Log-bilinear model : wx(wa-wb)
  - ✓ 단어 x에 대해서, log(a / b) = log(a) – lob(b)
  - ✓ 즉, 단어 x가 주어졌을 때, 단어 a가 등장할 log probability – 단어 b가 등장할 log probability
  - ✓ 즉, a와 b의 동시 발생 확률을 알 수 있음

  - ✓ x를 다양하게 변화시키면 a와 b의 동시 발생 확률 추정치를 확인할 수 있음

Idea

- ✓ Word Embedding에 Paragraph 추가



특징

- ✓ PV-DM
  - ✓ Distributed Memory Version of Paragraph Vector
  - ✓ 기존 Word Vector에 Paragraph Matrix를 합한 벡터
- ✓ PV-DBOW
  - ✓ Distributed Bag of Words Version of Paragraph Vecotr
  - ✓ word vector를 고려하지 않고 paragraph vector만을 이용해 단어 예측
- ✓ PV-DM을 이용해서 문맥의 특성을 파악하고
  PV-DBOW를 이용해서 문맥을 통해 단어들을 예측하는 단계를 거침
- ✓ 같은 문맥이나 문장에 나타나는 단어는 유사도가 높아짐
  - ✓ 다의어의 관점에서 좋을듯

3. Sub-word Embedding

1. BPE
2. Word Piece / Sentence Piece
3. Contextual Word Embedding

BPE 등장배경

- ✓ 언어마다 가지는 문자 체계가 다름
  - ✓ 중국어, 일본어는 띄어쓰기가 없음
  - ✓ 기타 등등…
- ✓ 단어를 조합, 변형해서 새로운 단어들을 많이 만듦
  - ✓ Ham + Burger -> Hamburger
  - ✓ pre- + position -> Preposition
  - ✓ 아이스아메리카노 -> 아아

⇒ 통용될 수 있는 Algorithm이 있으면 좋을 것 같다.

Byte Pair Encoding

- ✓ Byte의 Pair 단위로 Encoding하자.

- ✓ 컴퓨터는 Byte 단위로 구분자가 존재함
  - ✓ Ascii 코드 기반으로 각 character를 나타내는 byte 코드가 다름
  - ✓ example) A : **01000001**
  - ✓ 그리고 이렇게 모아서 단어를 이루는 방식

- ✓ 즉, 단어를 Byte Level로 쪼개고, pair frequency를 바탕으로 단어를 merge하는 방법

- ✓ Example)
  - ✓ 아침, 아침운동, 아침밥, 점심밥
  - ✓ Vocab = {아,침,운,동,밥,점,심}
  - ✓ {아,침}이 3회로 가장 빈번하게 등장 => 단어장에 {아침} 이란 단어 추가
  - ✓ 이 과정을 target vocab size까지 반복 증가

특징

- ✓ 단어 단위의 Piece화
- ✓ 빈도가 아닌 혼란도를 기준으로 n-gram을 추가하는 방법
- ✓ CJK (Chinese Japanese Korean)의 공용 적용을 위해,
  - ✓ Korean의 띄어쓰기는 _로 바꾸어 띄어쓰기를 없앰
- ✓ Bert는 Word Piece를 적용

특징

- ✓ 단어의 의미는 문맥에 따라 변화
  - ✓ 맥락을 잡아주는 역할

Q) “배”의 Embedding 값은 무엇인가?
A) 어떤 “배”인지 모르겠네
Q) “배”가 많이 나와서 운동좀 해야겠어
A) 그 때의 “배“ 는 ~~야.