한국어 임베딩 4장

4.1 NPLM(Neural Probabilistic Language Model)

모델 기본구조

- 기존 언어 모델의 단점인 존재하지 않는 n-gram이 포함된 문장이 나타날 확률 값을 0으로 부여한다. 문장의 장기 의존성 문제, 단어/문장 간 유사도를 계산할 수 있게 고안되었다.
- 자체적으로 단어 임베딩 역활 자체를 수행 가능하다.
-

NPLM : Neural Probabilistic Language Model

NPLM 기본 구조

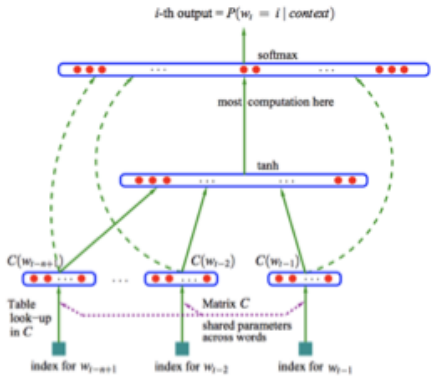


Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i-th word feature vector.

기본 아이디어:

직전까지 등장한 n-1개 단어들로 다음 단어를 맞추는 n-gram 언어 모델

"발 없는 말이 천리 간다"

↙ 4-gram

발, 없는, 말이, ??(t=4)

없는, 말이, 천리, ??(t=5)

Page 01

NPLM의 학습

- 단어 시퀀스가 주어졌을 때 다음 단어가 무엇인지 맞추는 과정에서 학습된다.
- 직전까지 등장한 n-1개의 단어들로 다음 단어를 맞추는 n-gram 언어 모델이다.
- 즉, 조건부 확률을 최대화 시킨다는 말과 그 의미가 같다.

  EX) “발”, “없는”, “말이” 세 개 단어가 주어졌다고 가정해보자

    - 그 다음 단어로는 무수히 많은 단어가 올 수 있지만 우리가 가진 말뭉치에서는 “천리”라는 단어가 가장 자주 나온다 했을시에, “천리”라는 단어가 들어가야 조건부 확률이 최대가 됨을 의미한다!
    - 또한 다음 시퀀스인 “없는”, “말이”, “천리” 로 주어진 경우 다음 단어도 비슷한 계산식으로 “간다”로 잘 맞춤을 의미한다.
- 또한 NPLM의 말단에는 소프트맥스 함수를 둠으로써 확률 벡터엣 가장 높은 요소의 인덱스에 해당하는 단어가 실제 정답 단어와 일치하도록 학습한다.
- 지금까지 NPLM의 알고리즘을 알아봤다면 어떤 입력 값들이 들어가는지 알아보자
    - C(W_t) = [0 0 0 1 0] * [11 18 25, 10 12 19, 4 6 13, 23 5 7, 17 24 1]
    - 여기서 앞의 0이 포함된 행렬은 예측행을 포함하는 원핫벡터의 내적이다.
    - 또한 만약 뒤의 특정 행 23 5 7이 있을시 각각이 입력층, 은닉층, 출력층을 의미한다.
    - 이를 y_w_t = b + W_x + Utanh(d+H_x)에 넣어 스코어 벡터를 계산한다.(딥러닝과 동일)
    - 이 후 이를 소프트맥스 함수에 적용하고 특정 인덱스와 비교후 역전파 하는 식으로 학습이 이루어지게 된다.
    -

NLPM과 의미 정보

4.2 Word2Vec

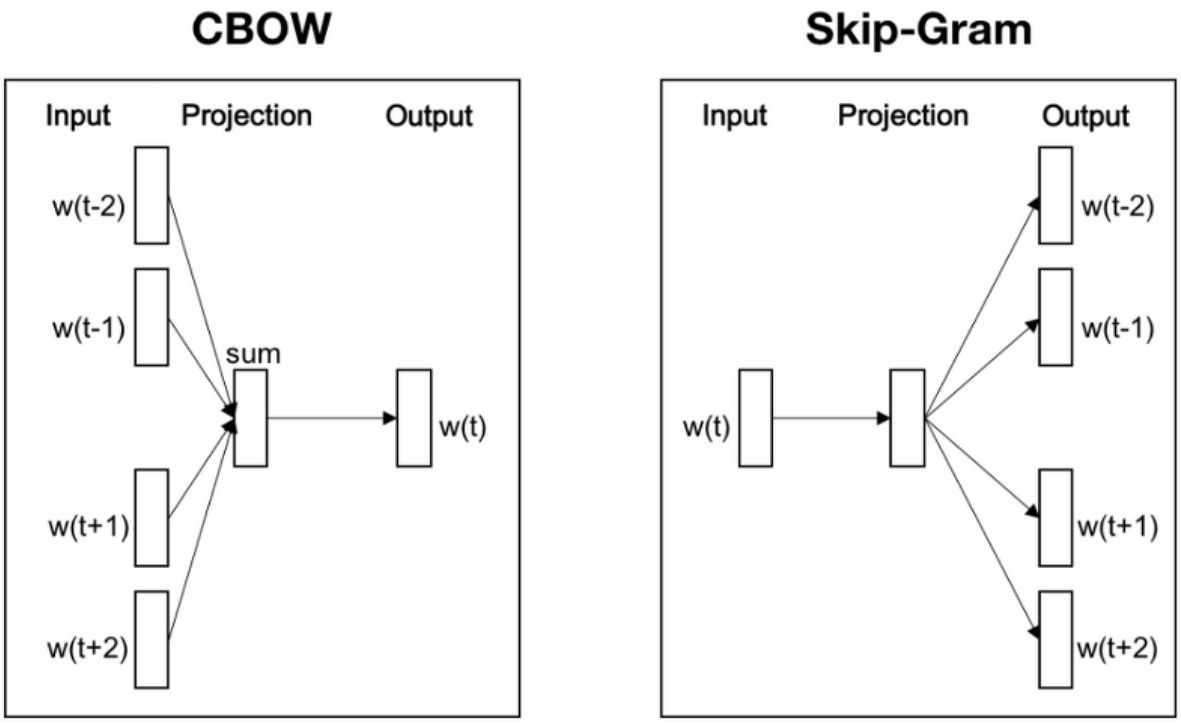
Skip-Gram과 CBOW라는 두가지 모델로 제안되었다.

이 두 모델을 근간으로 하되, 네거티브 샘플링 등 학습 최적화 기법을 제안한 내용이 논문에서 청므 제시되었다,

모델 기본 구조

CBOW : 주변에 있는 문맥 단어들을 가지고 타깃 단어 하나를 맞추는 과정에서 학습된다.

Skip-gram : 타깃 단어를 가지고 주변 문맥 단어가 무엇일지 예측하는 과정에서 학습된다.



Skip-gram이 같은 말뭉치로도 더 많은 학습 데이터를 확보할 수 있어 임베딩 품질이 CBOW보다 좋은 경향이 있다. 위의 그림만 봐도 Output 결과의 갯수가 다르다!

학습 데이터 구축

- Skip-gram 모델의 학습 데이터를 구축하는 과정
  - 포지티브 샘플 : 타깃 단어와 그 주변에 실제로 등장한 문맥 단어 쌍
  - 네거티브 샘플 : 타깃 단어와 그 주변에 등장하지 않는 단어 쌍
- ex) 개울가 (에서 속옷 빨래 를 하는) 남녀...
  - 포지티브 샘플 : (빨래, 에서), (빨래, 속옷), (빨래, 를), (빨래, 하는)
  - 네거티브 샘플 : (빨래, 책상), (빨래, 안녕), (빨래, 자동차), -무수히 많다.

이런식으로 구성시 네거티브 샘플이 많이 생길수 있고 기준이 애매하므로 → 희귀한 단어가 네거티브 샘플로 좀 더 잘 뽑힐 수 있도록 설계됨.

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n} f(w_j)^{3/4}}$$

하지만 이와는 별개로 자주 등장하는 단어는 학습에서 제외하는 서브 샘플링이란 기법도 적용이 된다. → 고빈도 단어의 경우 등장 횟수만큼 모두 학습시키는 것이 비효율적이라는 생각에서 근거!

모델 학습

- word2vec은 이제 이렇게 구성한 학습 데이터를 가지고 이진분류 학습을 한다. 즉 해당 쌍이 포지티브 샘플(+) 인지, 네거티브 샘플(-) 인지 학습한다.
- 이 때, 업데이트 되는 매트릭스는 2가지인데, 둘 다d차원의 단어 벡터의 행렬이다.(U(타깃), V(문맥))
- 이 후 아래과 같은 로그우도 함수를 최대화 하는 식으로 학습된다.
- 하나의 매트릭스 u는 각 중심 단어를 행으로 차원을 가지는 행렬이고, v는 각 주변 단어를 행으로 차원을 가지는 행렬이다.여기서 차원은 하이퍼 파라미터다. 보통 100 ~ 300 정도가 적당하다고 한다.

튜토리얼

4.3 FastText — word2vec이 안되는데 무슨..

- 페이스북에서 개발해 공개한 단어 임베딩이다.
- 각 단어를 문자 단위 n-gram으로 표현한다.

모델 기본 구조

fastText

기존 언어 모델의 비판

- **단어의 형태학적 특성을 반영하지 못했다.**

  기존의 Word2Vec과 GloVe 등은 단어 단위의 개별적 임베딩을 하기 때문에 teach, teacher, teachers 등과 같이 실제 유사한 의미를 가짐에도 불구하고 임베딩 벡터가 유사하게 구성되지 않았다.

- **희소한 단어를 임베딩하기 어렵다.**

  기존의 Word2Vec은 Distribution hypothesis를 기반으로 학습하기 때문에, 출현 횟수가 많은 단어는 임베딩을 잘하지만, 출현 횟수가 적은 단어는 제대로 임베딩을 하지 못한다.

Page 27

fastText

fastText 모델 구조

- 단어의 시작과 끝에 〈, 〉를 추가해 단어 경계를 표현.
- 각 단어를 문자(character) 단위 n-gram으로 표현.
- 그리고 〈원래 단어〉를 추가.
- 아래와 같이 5개의 문자 단위 n-gram 벡터의 합으로 원래 단어의 임베딩을 표현.

"슈퍼마켓"

↓ tri-gram ※〈슈퍼〉는 다른 단어

〈슈퍼, 슈퍼마, 퍼마켓, 마켓〉, 〈슈퍼마켓〉

$$\rightarrow u_{\text{슈퍼마켓}} = (z_{\langle\text{슈퍼}} + z_{\text{슈퍼마}} + z_{\text{퍼마켓}} + z_{\text{마켓}\rangle} + z_{\langle\text{슈퍼마켓}\rangle}) / 5$$

Page 28

fastText

fastText 모델 구조

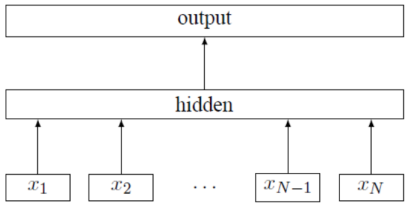


Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

1. Look-up table을 이용해 단어의 임베딩을 구함.
2. 단어 벡터들을 평균을 Input으로 사용.
3. 나머지는 Skip-gram 모델과 같음.
4. Skip-gram과 마찬가지로 Negative Sampling을 사용.

$$P(+|t,c) = \frac{1}{1 + exp(-u_t v_c)}$$ 최대가 되야 함.

※fastText의 로그우도 함수

$$L(\theta) = \underbrace{logP(+|t_p, c_p)}_{\text{1개의 포지티브 샘플}} + \underbrace{\sum_{i=1}^{k} logP(-|t_{n_i}, C_{n_i})}_{\text{k개의 네거티브 샘플}}$$

Page 29

fastText

fastText 효과

- n-gram의 문자단위 임베딩으로 Out-of-Vocabulary(OOV)를 처리할 수 있다.
- 단어의 내부 구조를 반영할 수 있다. (BPE; Byte Pair Encoding 알고리즘과 유사)
- 새로운 단어가 등장해도 기존의 n-gram vector를 찾아서 summation하면 재학습 과정 없이 대응할 수 있음.
- 어휘의 구문적(syntactic) 변화 규칙을 잘 잡아낼 수 있다.
- 한글을 자소 단위로 분해하여 한국어 임베딩에서 효과가 높다.
- 연구진들이 fastText로 학습된 언어의 임베딩 벡터와 코드를 공개했음 !

Page 30

한글 자소와 FastText

4.4 LSA(Latent Semantic Analysis) - 잠재 의미 분석

- 특이값 분해를 수행해 데이터의 차원 수를 줄여 계산 효율성을 키워 행간에 숨어 있는 잠재 의미를 이끌어내기 위한 방법론이다.

PPMI 행렬

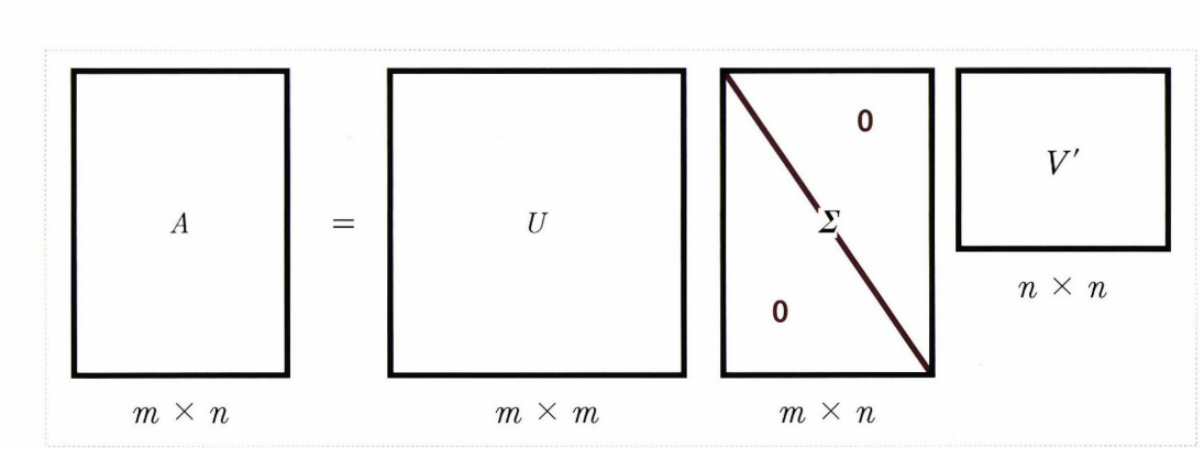
- 점별 상호 정보량(PMI) : 두 확률변수 사이의 상관성을 계량화한 지표다.

$$\mathrm{PMI}(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- 분자가 분모가 작을 경우 PMI는 음수가 된다. 하지만 이 경우는 말뭉치가 엄청 큰 경우 두 단어가 함께 등장할 확률로서 매우 작은 크기를 가진다.. 즉 잘 안일어난다. 10억개 가운데 1개
- 따라서 우리는 이러한 PMI 지표 대신 양의 값만을 사용하는 PPMI(Positive Pointwise Mutual Information)을 사용한다.
- PPMI(A,B) = max(PMI(A,B) , 0)
- SPMI : PMI에서 logk값을 빼준 값이다. - Word2Vec에서 사용된다.
- SPMI(A,B) = PMI(A,B) - logk

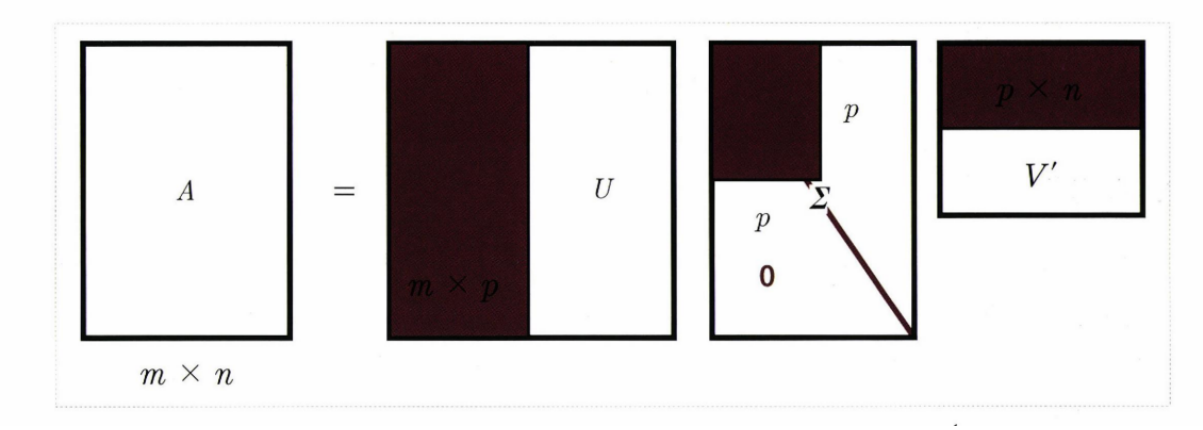
행렬 분해로 이해하는 Wrod2Vec

- 특이값 분해(SVD)



- Truncated SVD

  시그마의 대각원소 주에 상위 몇 개만 추출해서 여기에 대응하는 U와 V의 원소도 함께 제거해 더욱 차원을 줄인 형태로 분해하는 것!



- 위의 시그마가 대각행렬들이 잠재적인 의미를 가진 단어 임베딩이다.
- 즉 LSA를 적용하면 단어와 문맥 간의 내재적인 의미를 효과적으로 보존할 수 있 게 되어 문서간 유사도 측정 등 모델의 성능 향상에 도움을 줄 수 있다.
- 추가적으로 데이터의 노이즈, 희소성을 줄일 수 있다.

튜토리얼