컨텐츠기반모델 이해(Word2Vec)

Writer	최윤서(학부학생/공과대학 도시공학)
키워드	
Reference	
주차	2주차

1주차 컨텐츠 기반 필터링 복습

해당 사용자가 **좋아하는 콘텐츠와 비슷한 콘텐츠**를 추천

- How
    1. 아이템을 벡터로 임베딩
    2. 벡터들 간 유사도 계산해서 비슷한 컨텐츠를 추천
- 임베딩 방법론
    - Count Vectorizer = 단순 카운트
    - TF-IDF = 다른 문서에는 자주 등장하지 않고 특정 문서에서만 자주 등장하는 단어들에 높은 가중치를 부여.
    - Word2Vec (오늘 다룰 것!)

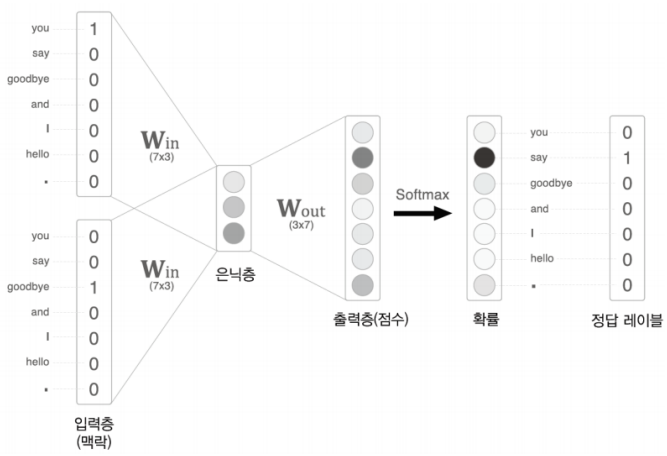
Word2Vec

- Count Vectorizer, TF-IDF(원핫인코딩)에서의 sparse matrix 단점을 해결
- 단어의 의미를 반영하여 임베딩.
- 비슷한 위치에 등장하는 단어들이 유사한 단어라는 가정.

  ex) 나는 ‘강아지’를 좋아한다. 나는 ‘고양이’를 좋아한다
- 어떻게 임베딩? 신경망 기반의 예측 모델을 통해서 학습하면서 이루어짐. (CBOW, Skipgram)

CBOW

- 주변단어(입력)을 통해서 중심단어(정답 레이블)을 예측하도록 학습하면서 임베딩 벡터를 구함.
- 하이퍼파라미터
    - window size: 해당 단어를 기준으로 앞뒤 몇개까지를 주변단어로 볼 것인지.
    - projection layer의 차원: 몇 차원의 벡터로 단어를 표현할 것인지.
- ex) You **say** goodbye and I say hello. (window size=1, projection layer 차원=3)



'say' 단어를 벡터로 임베딩하는 과정.

1. 주변단어들이 One-hot vector 형태(7차원)로 입력됨.
2. 각각 가중치행렬과 곱해져 5차원의 vector로 변환됨. 두개를 평균낸 것이 임베딩 벡터(5차원)
3. 임베딩 벡터(5차원)에 가중치행렬이 곱해져 7차원의 vector로 변환됨.
4. 출력층에 softmax 취해서 각 단어가 나올 확률로 만들어주기.
5. 정답 레이블과 비교하여 오차(cross-entropy loss)를 줄이기 위해 backpropagation

   → 정답 레이블을 잘 예측하도록 계속 weight 업데이트.

- 한 문장에서 단어의 수만큼 학습 수행.

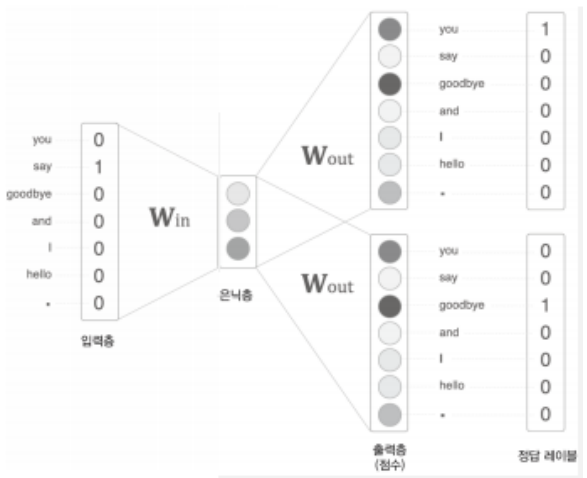
#1	you	say	goodbye	and	I	say	hello	.
#2	you	say	goodbye	and	I	say	hello	.
#3	you	say	goodbye	and	I	say	hello	.
#4	you	say	goodbye	and	I	say	hello	.
#5	you	say	goodbye	and	I	say	hello	.
#6	you	say	goodbye	and	I	say	hello	.
#7	you	say	goodbye	and	I	say	hello	.
#8	you	say	goodbye	and	I	say	hello	.

ex) 입력 you, goodbye → 출력 say

입력 say, and → 출력 goodbye

Skip-gram

- 중심단어(입력)을 통해 주변단어(출력)을 예측하도록 학습하면서 임베딩 벡터를 구함.
- ex) **You** say **goodbye** and I say hello. (window size=1, projection layer 차원=3)



'you', 'goodbye' 단어를 임베딩하는 과정

- 한 문장에서 단어의 수만큼 학습 수행
- 일반적으로 skip-gram으로 학습하는 것이 CBOW보다 더 성능이 좋음.

  같은 epoch의 학습을 한다고 하더라도 skipgram에서 각 단어들은 여러 context에 걸쳐 빈번하게 학습됨