**GloVe**

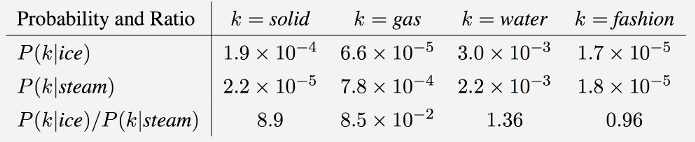
<https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/04/09/glove/>

Word2vec에서는 내적을 해서 분자의 likelihood를 높였는데 이는 코사인 유사도와 관련이 있다. = 임베딩된 단어간 유사도도 내적을 통해 한다.  
GloVe에서는 단어-문맥 행렬에 특이값분해(SVD) 실시 => 데이터 차원 축소  
노이즈를 줄여 내재적 의미 이끄는 LSA의 단점 또한 언급 => GloVe가 LSA, Word2Vec 비판

LSA는 말뭉치 전체의 통계적인 정보 모두 활용 => but 단어/문서 간 유사도 측정 어려움  
W2V은 지정한 윈도우에서만 학습/분석 => 말뭉치 전체의 co-occurrence(공기정보) 정보 반영 어려움

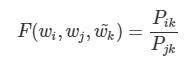
임베딩된 두 단어벡터의 내적이 말뭉치 전체에서 동시 등장 확률 로그값이 되도록 objective function 설정 = “단어벡터간 유사도 측정 수월, 말뭉치 전체의 통계 정보를 좀 더 잘 반영해보자”

동시에 같이 등장한 단어의 빈도를 각각 세어서 말뭉치의 단어 개수로 나눠줌 = 동시등장확률 = the words’ probability of co-occurrence)



Steam 등장 시 solid 나올 확률 > ice 등장 시 solid 나올 확률 (비율= 8.9)

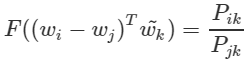
K 자리에 ice, steam이 올 수 있고, 전제 자리에 solid가 올 수도 있다. => 이거를 전체 단어에 대해 = 논문 아이디어

이 식을 만족하는 임의의 함수 F를 찾고자 함.

P-ik = P(k|i) = i번째 단어 주변에 k번째 단어가 등장할 조건부 확률 = i번째 단어와 k번째 단어가 동시에 등장한 빈도수 (X-ik)를 로 나눈 값 <위에서 여태 이야기한 것>

 이게 최종 목적임 = 임베딩된 두 단어벡터의 내적은 전체 말뭉치의 동시등장확률이 되도록 하는 것

3개 단어(중심 단어 1개, 주변 단어 2개 일때는) 주변 단어 끼리 뺀것과 중심단어를 내적한다. 그렇게 되면 이 값은 아까 그 ratio가 되게 하는 것이 목표이다.

이것도 최종 목표라 할 수 있다. 이 형태 만족하는게 준동형인가??? 또한 이 준동형은 단어가 바뀌어도 되는 것인가? 이것은 바로 대칭행렬 X로부터 오는 것인가? => 따라서 본격적인 학습을 하기전에 X를 만들고 시작해야함

최종은 중심단어와 주변단어 1개만을 고려하면 된다. 또한 조건 3가지가 존재

* Wi가 중심단어가 될수도 wk가 중심단어가 될수도 있다. => wi와 wk를 바꾸어도 같은 값을 반환
* Co-occurrence matrix X는 대칭 행렬
* Homomorphism!

<https://wikidocs.net/22885>

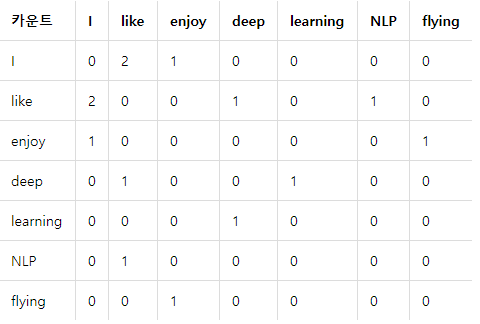
Word2vec, GloVe 중 어떤 것이 뛰어나다고 할 수는 없고 두 가지를 전부 사용해보고 성능이 더 좋은 것을 사용해야 한다.

LSA는 DTM이나 TF-IDF 행렬 처럼 문서에서의 단어 빈도수를 카운트한 전체적인 통계 정보를 입력으로 차원 축소(truncated SVD) / but 유사도 측정 불가

Word2Vect은 실제값과 예측값에 대한 오차를 손실 함수를 통해 줄여나감. / but window 내에서만 관계 고려

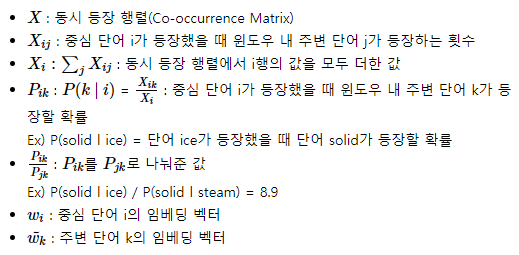
GloVe는 카운트 기반 방법과 예측 기반 방법 모두 사용

윈도우 기반 동시 등장 행렬



P(like|I) = 2/3 = 그 성분/그 행의 전체 성분 합 = I가 등장했을 때(I가 중심단어) like가 윈도우 내에 등장할(like가 주변에) 확률

이 개념을 가지고 loss function 작성 (window 개념만 word2vec에서 가져옴, 딥러닝 안씀)

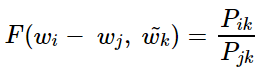
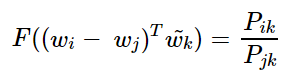


이것이 GloVe의 목표

= 임베딩된 중심 단어와 임베딩된 주변 단어의 내적이 전체 코퍼스(모든 문장)에서의 동시 등장 확률이 되도록 만드는 것이다.

사실 더 정확히는

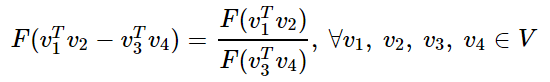
이 맞다.

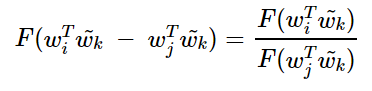
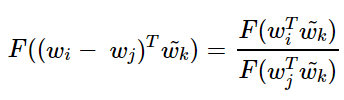
* 일단 두 비교 단어의 동시 등장 확률의 ratio를 벡터 공간에 인코딩이 목표 => wi – wj 의 차이를 임의의 함수 F의 입력으로 사용라는 전제하에 아래 제안을 실행
* 그러면 좌변은 벡터 값, 우변은 스칼라 값이니 좌변을 내적해서 형태를 맞춰 주도록 한다. -> 선형공간에서 단어의 의미 관계를 뺄셈과 내적으로 가능하게 된다.
* 하지만 어떤 단어가 중심 단어가 될지, 주변 단어가 될지 모르니 이 둘의 관계는 자유로워야 한다. F가 실수의 덧셈, 양수의 곱셈에서 Homomorphism(준동형) 이여야함 이 부분 다시 ,, 왜 준동형을 만족하는지. / 준동형이 뭔지

준동형: Group G, H가 있을 때 (G,+), (H, \*) 라고 했을 때, f : G->H 일 때 f(x+y) = f(x)\*f(y) // x,y는 G에 속함

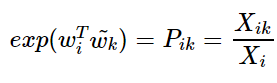
F는 결과값으로 스칼라 값이 나와야함. P-ik/P-jk => a,b 가 벡터라면 스칼라 값이 나올 수 없음. a,b가 내적 값이라면 스칼라 값 가능

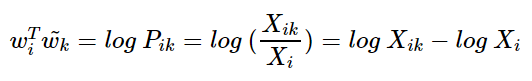
하지만 앞서 우리는 임베딩의 차이를 입력으로! => 뺄셈에 대한 준동형식으로 변경 => 곱셈도 나눗셈이 된다.



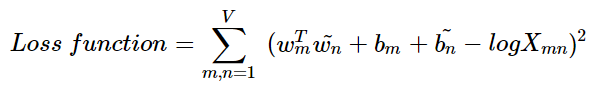


* 뺄셈에 대한 준동형식의 형태! 그럼 함수 F를 어떻게 찾지?
* 지수 함수가 바로 정확하게 이 형태를 만족

저 식이 결국에는 P-ik/P-jk 라는 전제와 같아야함. 따라서 을 만족한다.

가 된다.

하지만 우리가 가정한 것은 wi와 wk의 위치가 바뀌어도 식이 성립해야 하니 logXi라는 중심단어가 i일때만 해당하는 식을 wi에 대한 편향 bi와 wk에대한 편향 bk를 추가한다.

라는 식이 나오고 wmwn+bm+bn과 logXmn의 차이를 최소화 해야한다.(V는 단어 집합의 크기)

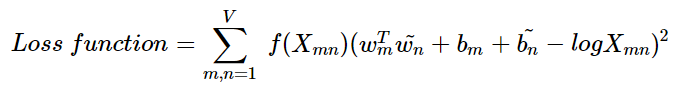
하지만 logXmn이 0이 될 수 있어서 그것이 문제가 될 수 있고 log(1+Xmn)으로 변경했지만 해결이 되지 않는다.

동시 등장 행렬에서 0이 많을 수 있다 = sparse matrix / 사실 X-ik가 굉장히 낮을 경우 정보에 거의 도움이 되지 않는다.

X-ik의 값에 따라 영향을 받는 가중치 함수 f(X-ik)를 loss function에 도입

X-ik 작다 => 작은 가중 치 줘서 작게 / X-ik 크다 => 큰 가중치 줘서 크게(하지만 max 가중치는 정해져 있다. ‘it is’ 같은 불용어는 높은 가중치를 받아서는 안된다.





* 점차 wm,wn,bm,bn 업데이트해서 loss 최소화!

**Lecture**

SVD로 Embedding 하는 작업도 있다.

Evaluation word vectors! (intrinsic vs extrinsic evaluation)

**paper**

zz