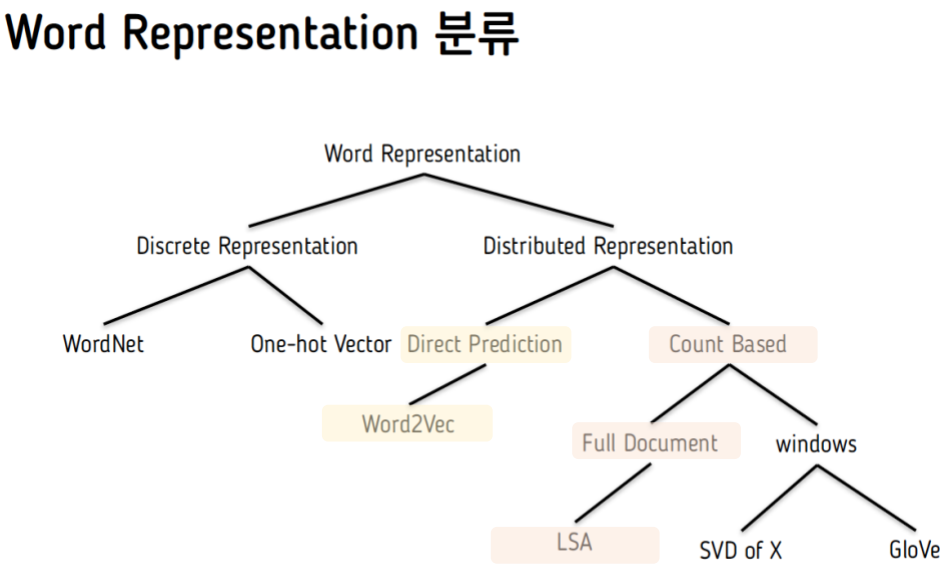
<Introduction>

단어의 벡터를 학습하는 모델은 대표적으로 두 가지가 있다.
1) **global matrix factorization methods** (ex. LSA)
2) **local context window methods** (ex. skip-gram)

하지만 두 가지 모델 모두 단점이 있다. LSA와 같이 global matrix factorization methods 같은 경우에는 통계적 정보를 효율적으로 이용한다는 장점이 있지만 **상대적으로 word analogy task에 약하다.** skip-gram과 같은 local context window 방식은 word analogy에 더 강하지만 global co-occurrence counts 대신 각각의 local context에서 훈련하기 때문에 **통계적 정보를 잘 이용하지 못한다는 단점**이 있다.
따라서 이 논문에서는 통계적 정보를 잘 이용하면서 analogy task에도 괜찮은 성능을 보이는 모델을 제시하고자 한다.

<Related Work>




출처 : 곽근봉(2017)

-Matrix Factorization Methods
직역하면 행렬 분해 방식인데 쉽게 말해서 통계 정보를 담고 있는 **큰 행렬을 분해해서 작은 차원의 워드 벡터를 생성**하는 것이다. 이 행렬 분해 방식에는 문제가 있는데 가장 자주 출현하는 단어들이 **유사도 측정에 불균형적으로 영향을 준다**는 것이다. 예를 들어 "the"와 "and"의 동시 출현 횟수는 이 단어들 자체가 자주 등장하므로 클 것인데 이 때문에 이 두 단어 사이에 의미적 연관성이 적음에도 불구하고 유사도에 크게 영향을 줄 수 있다.

-Shallow Window-Based Methods

Word2vec에 활용된 방법론으로서 인공신경망 기반의 임베딩 방식이고, **local context window 내에서 대상 단어와 근접한 단어들을 학습하여 예측**하는 방식이다. 결과 부분에서 Word2vec과 성능 비교를 위해 GloVe모델을 skip-gram, CBOW 모델과 비교하는 부분이 나오는데 skip-gram은 단어가 주어졌을 때 문맥을 예측하는 방식이고, CBOW는 문맥이 주어졌을 때 단어를 예측하는 방식이다. 앞서 언급한 Matrix Factorization 방식과는 다르게 corpus의 동시 발생 **통계 정보를 직접적으로 활용하지 않는다는 단점**이 있다.

# 2021 P-SAT 논문 스터디

2021-01-18 화요일
김원구

---

## GloVe 3장

**단어 출현 빈도에 대한 통계량**은 단어 표현을 학습하기 위한 비지도 방법의 주요한 기본 정보다.
하지만,
1) 이 통계량으로부터 어떻게 의미가 생성되고,
2) 단어 벡터가 어떻게 의미를 표현하는지는 여전히 의문 사항이다.

본 논문에서는 Global Vector를 직접 끌어내는 모델인 GloVe라는 모델을 소개한다.

## 관련 Notation

$X$ : 단어-단어 동시출현 빈도에 대한 행렬. 행렬 $X$의 원소 $X_{ij}$는 단어 $i$의 문맥에서 단어 $j$의 출현 횟수

$\sum_k X_{ik}$ : 단어 $i$의 문맥에서 나타나는 모든 단어의 출현 횟수

$P_{ij} = P(j|i) = X_{ij}/X_i$ : 단어 $i$의 문맥에서 단어 $j$가 나타날 확률

**Big data may mean more information, but it also means more false information.**
**=> big data may mean more information but it also mean more false information**

| | big | data | may | mean | more | information | but | it | also | false |
|---|---|---|---|---|---|---|---|---|---|---|
| big | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| data | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| may | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| mean | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 1 | 0 |
| more | 0 | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
| information | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| but | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| it | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| also | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| false | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |

$$X_i = \sum_k X_{ik}$$

$$X_{ij}\ ex)\ X_{mean,more} = 2$$

ex) $X_{mean} = 4$

$P_{ij} = P(j|i) = X_{ij}/X_i$

$ex)\ P_{mean,more} = \frac{2}{4} = \frac{1}{2}$

## 동시 출현 행렬 (Co-occurrence matrix)

이제 간단한 예시를 통해 어떻게 특정한 의미가 단어 동시 출현 확률로부터 도출되는지를 살펴보자.

Table 1: Co-occurrence probabilities for target words *ice* and *steam* with selected context words from a 6 billion token corpus. Only in the ratio does noise from non-discriminative words like *water* and *fashion* cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam.

| Probability and Ratio | $k = solid$ | $k = gas$ | $k = water$ | $k = fashion$ |
|---|---|---|---|---|
| $P(k\|ice)$ | $1.9 \times 10^{-4}$ | $6.6 \times 10^{-5}$ | $3.0 \times 10^{-3}$ | $1.7 \times 10^{-5}$ |
| $P(k\|steam)$ | $2.2 \times 10^{-5}$ | $7.8 \times 10^{-4}$ | $2.2 \times 10^{-3}$ | $1.8 \times 10^{-5}$ |
| $P(k\|ice)/P(k\|steam)$ | 8.9 | $8.5 \times 10^{-2}$ | 1.36 | 0.96 |

위 도표는 GloVe 논문의 Table1이다. 코퍼스는 열역학에 관한 주제이며, 우리는 $i$=ice와 $j$=steam의 관계에 대해 알고 싶다. 이를 여러 단어 $k$에 대한 동시 출현 행렬 상의 확률을 이용해 구해보자.
$k$가 solid처럼 ice와 더욱 연관 있을 때, $P_{ik}/P_{jk}$ 값은 높을 것이며, $k$가 gas처럼 steam과 더욱 연관 있을 때, $P_{ik}/p_{jk}$ 값은 낮을 것이다.
$k$가 water나 fashion처럼 ice와 steam 모두에 대해 연관이 깊거나 없을 경우 해당 값은 1로 수렴할 것이다.

이를 통해 알 수 있는 것은 동시 출현 행렬로부터 구한 $P_{ik}/P_{jk}$ 비율로
1) 연관 있는 단어(solid, steam)와 연관 없는 단어(water, fashion)를 구분지을 수 있다는 점
2) 두 연관 단어(solid, steam)를 더 잘 구별할 수 있다는 점
이다.
따라서 두 단어 $i, j$에 대한 직접적인 확률이 아닌 다른 단어 $k$에 대한 각각의 확률로 objective function을 만들어보자.

## Objective function

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \qquad (1)$$

$w \in \mathrm{R}^d$ : 단어 벡터 (target)*

$\tilde{w}_k \in \mathrm{R}_d$ : 별도 문맥(seperate)의 단어 벡터 (문맥)**

* 단어 벡터는 초깃값으로부터 학습을 통해 최적화된다.

** 각 단어 벡터는 고유의 target 단어를 가지지만 다른 단어들의 컨텍스트 역할도 한다. 결국 모든 계산이 대칭이기 때문에( $X$가 symmeric matrix) 두 벡터는 동일해야 하지만, 근사 계산(approximate computation)을 사용하기에 두 벡터는 약간 다를 수 있다. 따라서, 해당 단어의 최종 벡터로 두 개의 합을 사용한다. 이렇게 하면 오버피팅을 막고 noise도 줄어들어 결과가 더 좋아진다. like boosting

(1)의 식으로 확률비 정보를 벡터 스페이스에 임베딩하고, 함수 $F$의 정체는 점차 밝혀갈 예정이다.

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \qquad (2)$$

벡터 공간은 본질적으로 선형 구조이기에(vector spaces are inherently linear structures, 선형 연산에 대해서 닫혀있기 때문인가?)

두 타겟 단어 $i$와 $j$의 차(distance)를 확률비 $\frac{P_{ik}}{P_{jk}}$를 구하는 데에 사용할 수 있다. 따라서, 함수 $F$ 를 두 타겟 단어 $i$와 $j$의 차이에만 의존하도록 제한할 수 있게 된다.

하지만 이렇게 되면 식 (2)의 좌변은 벡터지만, 우변은 스칼라이다. 그러므로 $w_i - w_j$와 $\tilde{w}_k$를 내적하여 이를 해결한다.

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \quad (3)$$

한편, 동시출현 행렬에서 타겟 단어와 문맥 단어의 구별은 임의적(arbitrary)이기에 두 단어의 역할을 바꿀 수도 있어야 한다. 따라서, $w \leftrightarrow \tilde{w}$도 성립해야하고 당연히 $X \leftrightarrow X^T$도 성립해야 한다(symmetry). GloVe의 최종 모델은 이러한 교환(대칭)에 변함이 없어야 하지만 식 (3)그렇지 않다.
다행히, 대칭성은 2단계어 걸쳐 유지될 수 있다.
step 1) 함수 $F$가 군 $(\mathrm{R}, +)$와 $(\mathrm{R}_{>0}, \times)$에서 준동형사상이다. (homomorphism between the groups $(\mathrm{R}, +)$ and $(\mathrm{R}_{>0}, \times)$)

cf) 준동형사상

$$\phi: G \to G' \ \& \ a \in G, b \in G \phi(ab) = \phi(a)\phi(b)$$

이를 통해 식 (3)은 아래와 같이 변형된다.

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} \quad (4)$$

한편, 식 (4)는 식 (3)을 사용하면 단어 $i$만을 고려해 간단히 풀리는데,

$$F(w_i^T \tilde{w}_k) = P_ik = \frac{X_ik}{X_i} \quad (5)$$

처럼 풀리게 된다.

식 (4)의 해는 함수 $F$가 준동형사상임을 고려하면 $F = exp$이므로 $F = exp$를 취한 후 $log$를 씌우면

$$F(w_i^T \tilde{w}_k) = exp(w_i^T \tilde{w}_k) \Leftrightarrow log(exp(w_i^T \tilde{w}_k))$$

$$= w_i^T \tilde{w}_k = log(P_{ik}) = log(\frac{X_{ik}}{X_i}) = log(X_{ik}) - log(X_i) \quad (6)$$

이 된다. 식 (6)에서, 우변의 $log(X_i)$가 없다면 $w_i^T \tilde{w}_k = log(X_{ik})$로 $X$가 대칭 행렬임을 보일 수 있다.

step 2) 그런데, $log(X_i)$는 단어 $k$에 대해 독립이므로 $w_i$의 편향 $b_i$에 포함시킬 수 있다. 여기에 $\tilde{w}_k$에 대한 편향 $\tilde{b}_k$를 추가하면 결과적으로 대칭성을 유지할 수 있게 된다.

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = log(X_{ik}) \quad (7)$$

위 식은 처음 loss function인 (1)에 비해 매우 간결하지만, 로그는 0을 진수로 갖지 못하기에 1을 더해 이를 해결한다.
$log(X_{ik}) \to log(1 + X_{ik})$

이렇게 문제를 다 해결한 듯 했지만 이 모델에는 주요 결함이 있는데 단어 간 동시 발생이 거의 없거나 아예 없어도 모든 동시 발생에 대한 가중치가 동일하다는 것이다.
이러한 요소는 noise를 주며 빈번한 요소들보다 정보가 없다.
한편, 위 식은 $log(X_{ik})$를 $y$로 하는 Least Squares Estimation의 식으로 볼 수 있다.*
그러므로 위 결함을 WLS로 해결할 수 있다.
* word2vec은 단어 임베딩을 분류 문제로 해결했지만 GloVe는 회귀 문제로 해결함 (X=임베딩 벡터, y=동시출현 행렬)

$$J = \sum_{i,j=1}^{V} f(X_{ij})(W_i^T \tilde{W}_j + b_i + \tilde{b}_j - logX_{ij})^2 \qquad (8)$$

$f(X_{ij})$는 해당 WLS의 가중치 함수이며, $V$는 어휘 개수이다.
가중치 함수는 아래와 같은 특성을 따라야 한다.
1) $f(0) = 0$. 만일 $X_{ij}$값이 0이라면, log(X_ij)는 음수로 발산하는데 이를 잡으려 내적값이 이상해지는 것을 방지하는 장치.
2) Non=decreasing 함수. 거의 발생하지 않는 동시 발생에 과대 가중하지 않기 위함.
3) 큰 값에 대해서는 가중치가 상대적으로 작아야 함. 2)와 비슷한 맥락으로, 자주 발생하는 동시 발생에 대해서 과대 가중하지 않기 위함.
이를 만족하는 여러 함수가 있지만, 그 중 잘 학습하는 가중치 함수는 아래와 같다.

$$f(x) = \begin{cases} (x/x_{max})^{\alpha} & if\ x < x_{max} \\ 1 & otherwise \end{cases}$$

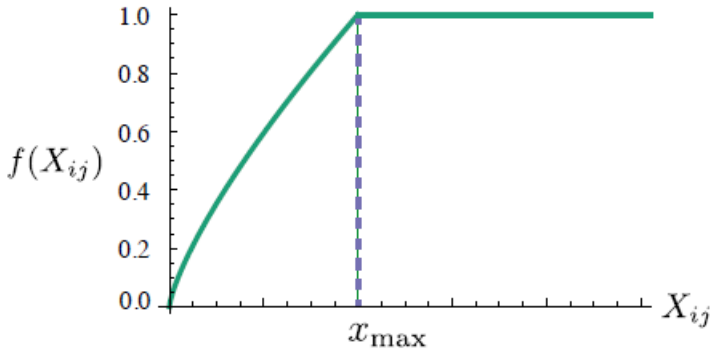



Figure 1: Weighting function $f$ with $\alpha = 3/4$.

모델 성능은 컷 오프에 따라 조금씩 달라지는데, 실험을 통해 $X_{max} = 100$으로 정했다. 또한 $\alpha = 3/4$가 모든 선형 버전에서 성능이 괜찮음을 발견했다.

3.1 타 모델과의 관계

이번 장에서 살펴볼 것은 Glove 모델과 다른 모델(구체적으론 주로 Skip-Gram)과의 관계이다. 대표적인 window 기반 모델인 Skip-gram과 glove의 차이점을 살펴보도록 하자. 우선 설명에 앞서 Skip-gram 모델의 기본적인 아이디어는 중심단어 $w_i$가 등장했을 때, 주변단어 $w_j$가 등장할 것을 예측하는 모델을 만든 것이다. 그래서 두 단어 벡터의 내적 값을 최대화(1)이 되도록 한다. 이를 위해 소프트 맥스 계층을 이용해 중심단어 $w_i$가 등장했을 때, 주변단어 $w_j$가 등장할 확률을 예측한다. 이를 수식으로 표현하면 다음과 같다.

$$Q_{ij} = \frac{\exp(w_i^T \tilde{w}_j)}{\sum_{k=1}^{V} \exp(w_i^T \tilde{w}_k)}$$

그리고 Skip-gram 모델의 손실 함수는 다음과 같다.

$$J = -\sum_{\substack{i \in \text{corpus} \\ j \in \text{context}(i)}} \log Q_{ij} \, .$$

단순히 negative log possibilty이다. 식을 이렇게 꾸리는 이유는 크로스엔트로피에서 y가 1인 항만 떼어다 놓은 것이다. 우린 어차피 $Q_{ij}$, 즉, 중심단어가 주어졌을 때, 주변단어의 확률이 커지기만 하면 그만이기 때문이다. 하지만 $Q_{ij}$를 계산하는 과정에서 문제가 있다. 소프트맥스 계층은 모든 클래스에 대해 지수함수를 계산 해야한다. 즉, 모든 단어에 대해 확률을 계산해야 하는 것이다. 불필요한 연산이 너무 많아진다. 이를 해결하기 위해 Glove처럼 생각해보자면, X행렬이 있다면, $Q_{ij}$를 한번만 계산하고, 이를 $X_{ij}$ 곱해주면 될 것이다. 식으로 나타내면 다음과 같다.

$$J = -\sum_{i=1}^{V} \sum_{j=1}^{V} X_{ij} \log Q_{ij} \, , \qquad (12)$$

앞에서 $X_i = sum(X_{ik})$ $P_{ij} = \frac{X_{ij}}{X_i}$ 라고 했다. 이를 위 식에 대입한다면 식은 또 다음과 같이 바뀔 수 있다. $P_{ij}$는 중심단어 $word_i$가 등장했을 때 주변단어 $word_j$가 등장한 확률이다. 즉, 우리가 실제로 알고 있는 정답(Y)이다. 그리고 $Q_{ij}$우리가 예측한 확률이다. 결국 정답과 예측 사이의 비교가 되는 것이고, 이로 인해 크로스 엔트로피의 형태로 나타낼 수 있게 된다.

$$J = -\sum_{i=1}^{V} X_i \sum_{j=1}^{V} P_{ij} \log Q_{ij} = \sum_{i=1}^{V} X_i H(P_i, Q_i),$$

하지만 문제가 있다. 위에서도 언급했듯이 $Q_{ij}$ *를 계산하는 것은 연산량이 많이 필요하다.* 이를 해결할 근본적인 방법은, 확률의 형태를 포기하는 것이다. 확률은 (경우의 수)/(전체 경우)이다. 즉, 분모는 일반화를 위해 존재할 뿐이다. 분모를 제거해도, 두 경우가 똑 같은 사건에서 나왔다면 비교가 가능하다. 그래서 우리는 식을 또 변형한다.

$$\hat{J} = \sum_{i,j} X_i (\hat{P}_{ij} - \hat{Q}_{ij})^2$$

이런 식으로 바꾸면 우리는 결국 확률 분포의 비교가 아니기 때문에 최소 제곱법을 손실함수로 사용하게 된다. 여기서 새로 등장한 노테이션은 다음과 같다.

$$\hat{P}_{ij} = X_{ij}$$

$$\hat{Q}_{ij} = \exp(w_i^T \tilde{w}_j)$$

하지만 여전히 문제는 존재한다. $X_{ij}$는 종종 너무 큰 값을 가진다. 그러니까, 단어의 빈도 분포는 포아송 분포와 같이 long tail한 형태이다. 이때, 소수의 무척 자주 등장하는 단어들은, 손실함수에서 매우 큰 영향을 끼치게 되고, 최적화에 방해가 된다. 이를 스케일링하기 위해 로그를 씌워주자.

$$\begin{aligned} \hat{J} &= \sum_{i,j} X_i (\log \hat{P}_{ij} - \log \hat{Q}_{ij})^2 \\ &= \sum_{i,j} X_i (w_i^T \tilde{w}_j - \log X_{ij})^2 . \end{aligned}$$

Skip-Gram의 논문을 살펴보면 학습을 더욱 효과적으로 시키기 위해, 한 번 더 스케일링 해주게 된다. 우리도 이 개념을 이용해보자. 그리고 우린, Glove에서 사용한 f(x) 함수를 활용하자. Glove의 관점에서 본 skip-gram 모델이 최종적으로 완성되었다.

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

그런데 식 16과 식 8을 보면 거의 동일하게 이루어져있다. 두 식 모두 weight*loss 형태의 weighted least square 형태이다. 즉, Glove랑 Skip-Gram의 차이는 X 행렬의 존재여부가 아닐까?!

(사족임 뇌피셜임)

3.2 모델 복잡도.

모든 알고리즘은 시간을 소비한다. 그리고 어느 정도의 시간을 소비하는지는 중요한 요소이다. 하지만, 시간은 결국 어떤 컴퓨터를 쓰느냐에 따라 달라지게 된다. 그래서 알고리즘 분야에서 모델의 복잡함을 평가할 때는, 연산할 양을 가지고 평가하게 된다. 1+1이나 2+2나 결국 한번만 계산하니까 똑 같은 복잡도를 가진 식이라는 말이다. 하지만 1+1+1 은 2+2보다 복잡한 모델일 것이다. 두 번 계산하니까!

Glove 모델의 경우 결국 X 행렬의 크기만큼만 계산이 이루어지게 된다. 즉 vocab의 $|V|^2$만큼의 복잡도를 가지게 된다. 정확한 계산량은 모른다. 그런데 어차피 V가 1000이 넘어갈 것이고 그럼 2차식인 것이 중요하지 2차식의 1차항은 계산량에 거의 미치는 영향이 없다. 즉, 최고 차항만 계산량을 결정하는 요소라고 볼 수 있다. 대애애애애충 이 정도 계산량이다 하는 것이다. 이것을 기호로 $O(|V|^2)$로 나타낸다. Skip-Gram의 경우엔 전체 문서를 순서대로 훑는다. 그래서 Skip-Gram의 경우엔 ₩C₩에 비례할 것이다. 문서가 커지면 그에 정비례하는 것이다.

하지만 이러한 생각엔 문제가 있다. 행렬 X는 sparse matrix일 수 밖에 없다. 동시에 등장하지 않는 단어가 더 많겠지. 그리고 이러한 단어 조합은 계산이 이루어지지 않는다. 즉, $O(|V|^2)$이것과 차이가 커질 수 있다는 말이다. 보다 정밀한 측정이 필요하다. 3.2 장에선 이 계산을 다루고 있다.

이를 계산하기 위해 Glove 논문에선 다음과 같은 가정을 한다.

$$X_{ij} = \frac{k}{(r_{ij})^{\alpha}}.$$

여기서 $r_{ij}$는 $w_{ij}$가 빈도수 기준 상위 몇번째 랭킹인지이다. 그리고, k와 양수 알파는 모두 하이퍼파라미터다.  즉, 자주 등장하는 단어일수록 큰 값을 가지도록 만들어주었다. 그리고 전체 문서에 등장하는 단어의 수는 동시 등장 행렬 X의 모든 원소의 합과 같고, 이를 다시 나타내면 다음 식

과 같다.

$$|C| \sim \sum_{ij} X_{ij} = \sum_{r=1}^{|X|} \frac{k}{r^{\alpha}} = kH_{|X|,\alpha},$$

여기서 $H_{n,m}$은 $H_{n,m} = \sum_{r=1}^{n} \frac{1}{r^m}$로 표현되는 조화수이다. (사실 여기부터 잘 모르겠다.) 그리고 위의 식은 다음과 같이 다시 표현할 수 있다.

$$|X| = \max_{i,j}\{r_{ij}|X_{ij} > 0\} = \max_{i,j}\{r_{ij}|X_{ij} = 1\} \sim k^{1/\alpha}$$

그리고 이를 종합하면 다음 식이 나오게 된다.

$$|C| \sim |X|^{\alpha} H_{|X|,\alpha}$$

조화수의 경우 리만 제타 함수를 통해 근사할 수 있다. 그래 나도 무슨 소리인지 모르겠다... 근사값은 다음과 같다.

$$H_{x,s} = \frac{x^{1-s}}{1-s} + \zeta(s) + \mathcal{O}(x^{-s}) \quad \text{if } s > 0, s \neq 1$$

이를 대입하면 다음과 같은 식이 된다.

$$|C| \sim \frac{|X|}{1-\alpha} + \zeta(\alpha)|X|^{\alpha} + \mathcal{O}(1)$$

그리고 최종적으로 계산된 복잡도는 다음과 같다.

$$|X| \begin{cases} \mathcal{O}(|C|) & \text{if } \alpha < 1 \\ \mathcal{O}(|C|^{1/\alpha}) & \text{if } \alpha > 1 \end{cases}$$

또한, Glove는 알파를 1.25로 했을 때 처음 했던 가정이 잘 지켜졌다고 한다. 결과적으로 최종적인 모델의 복잡도는 $|X| = \mathcal{O}(|C|^{0.8})$ 이며, 이는 Skip-Gram의 복잡도 $\mathcal{O}(|C|)$보다 우수하다고 할 수 있다. 사실 도찐 개찐같다.

<Experiments>

-Evaluation methods

세 가지 experiment를 진행하였다. 이 논문에서 특히 강조하고 있는 부분은 Word analogies이다.

1) **Word analogies**

유추 작업은 "a is to b as c is to ___?"과 같은 질문들로 구성되어 있으며 semantic 부분과 syntactic 부분으로 나누어진다. semantic 부분은 "Athens is to Greece as Berlin is to ___?"와 같이 사람이나 장소에 대한 유추 질문이고, syntactic 부분은 "dance is to dancing as fly is to ___?"와 같이 동사의 시제나 형태에 관련된 질문이다. "a is to b as c is to ___?" 에서 정답 d를 찾기 위해서 코사인 유사도를 이용해 $w_b - w_a + w_c$와 가장 가까운 d의 워드 벡터 $w_d$ 를 구한다.

2) **Word similarity**

3) **Named entity recognition**

-Corpora and training details

이 논문에서는 모델을 각각 사이즈가 다른 다음 5개의 corpora에 대해 훈련시켰다.

1. a 2010 Wikipedia dump with 1 billion tokens
2. a 2014 Wikipedia dump with 1.6 billion tokens
3. Gigaword5 which has 4.3 billion tokens
4. Gigaword5 + Wikipedia2014 with 6 billion tokens
5. 42 billion tokens of web data, from Common Crawl

모델의 experimental setup은 다음과 같다.

$x_{max} = 100$
$\alpha = 3/4$
optimizer = AdaGrad
learning rate = 0.05
300차원 미만 : 50 iterations, 300차원 이상 : 100 iterations
window: 특별한 언급 없으면 왼쪽으로 10개, 오른쪽으로 10개

모델이 $W$ 와 $\widetilde{W}$ 두개의 단어 벡터를 가지게 되는데 instance들을 training한 후에 그 결과들을 더해주면 과적합과 노이즈를 줄일 수 있으므로 여기서는 단어 벡터로 $W + \widetilde{W}$를 사용하기로 한다.

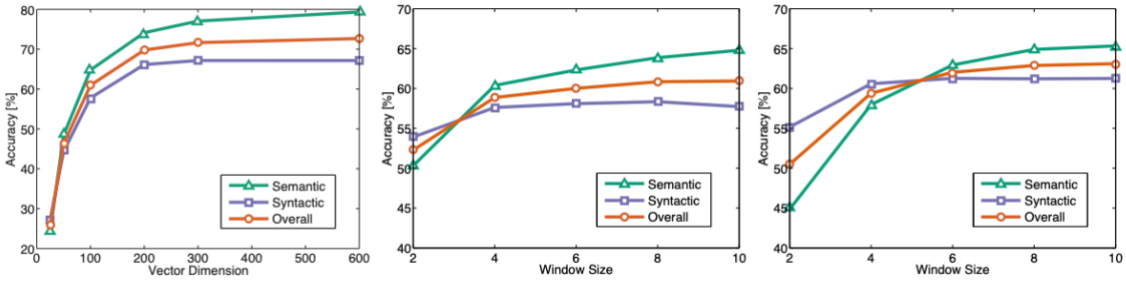
위와 같은 조건으로 실험한 결과를 **SVD**(Singular Value Decomposition), **SG**(Skip-Gram), **CBOW**(Continuous Bag-Of-Words)과 **비교**한다.

<Results>

| Model | Dim. | Size | Sem. | Syn. | Tot. |
|---|---|---|---|---|---|
| ivLBL | 100 | 1.5B | 55.9 | 50.1 | 53.2 |
| HPCA | 100 | 1.6B | 4.2 | 16.4 | 10.8 |
| GloVe | 100 | 1.6B | 67.5 | 54.3 | 60.3 |
| SG | 300 | 1B | 61 | 61 | 61 |
| CBOW | 300 | 1.6B | 16.1 | 52.6 | 36.1 |
| vLBL | 300 | 1.5B | 54.2 | 64.8 | 60.0 |
| ivLBL | 300 | 1.5B | 65.2 | 63.0 | 64.0 |
| GloVe | 300 | 1.6B | 80.8 | 61.5 | 70.3 |
| SVD | 300 | 6B | 6.3 | 8.1 | 7.3 |
| SVD-S | 300 | 6B | 36.7 | 46.6 | 42.1 |
| SVD-L | 300 | 6B | 56.6 | 63.0 | 60.1 |
| CBOW† | 300 | 6B | 63.6 | 67.4 | 65.7 |
| SG† | 300 | 6B | 73.0 | 66.0 | 69.1 |
| GloVe | 300 | 6B | 77.4 | 67.0 | 71.7 |
| CBOW | 1000 | 6B | 57.3 | 68.9 | 63.7 |
| SG | 1000 | 6B | 66.1 | 65.1 | 65.6 |
| SVD-L | 300 | 42B | 38.4 | 58.2 | 49.2 |
| GloVe | 300 | 42B | **81.9** | **69.3** | **75.0** |

Analogy task에 대한 결과이다. **GloVe모델이 다른 모델들과 비교해 상당히 성능이 좋다는 것을 알 수 있다.** 밑줄 친 부분은 비슷한 크기의 모델 그룹 내에서 가장 높은 accuracy인데 몇 가지 경우를 제외하고는 대부분 GloVe 모델에 밑줄이 쳐져 있다. 또한 GloVe모델이 420억 개의 큰 corpus에서도 쉽게 훈련된다는 것을 알 수 있다. 하지만 더 큰 corpus에서 성능이 감소한 SVD-L의 결과를 통해 **corpus의 크기가 커진다고 해서 다른 모델들 또한 성능 향상이 보장되는 것은 아니라**는 것을 알 수 있다.
Analogy task뿐만 아니라 Similarity task, NER task에서도 GloVe의 성능이 좋다는 결과가 나왔다.

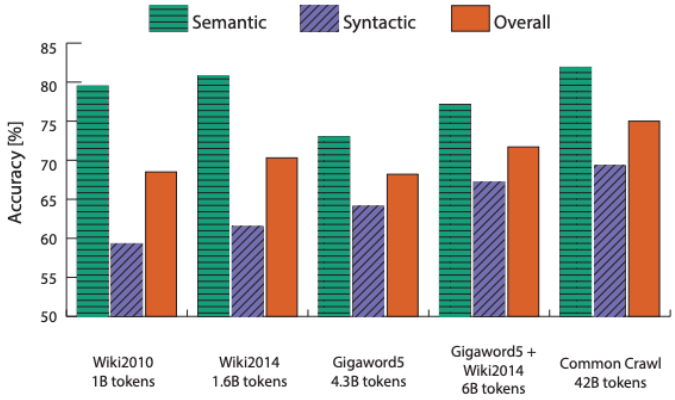
다음으로 GloVe 모델에 대해서 다양한 측면에서 분석해 보고자 한다.
-Context Size




(a) Symmetric context (b) Symmetric context (c) Asymmetric context

위 그림은 다양한 vector length와 context window에 따른 결과들을 보여준다. 여기서 'Symmetric'은 대상 단어를 중심으로 오른쪽과 왼쪽 모두 대칭적으로 확장한 window를 가리킨다. 반대로 'Asymmetric'은 대상 단어를 중심으로 왼쪽으로만 확장한 window를 가리킨다. (a)를 통해 **차원이200보다 커지면 accuracy의 증가율이 감소**한다는 것을 알 수 있으며, (b)와 (c)를 통해 **문법 task에서는 비대칭이면서 크기가 작은 window가 성능이 좋다**는 것을 볼 수 있고, **의미 task에 대해서는 non-local 할 때 성능이 좋기 때문에 window 크기가 커야 한다**는 것을 알 수 있다. 아무래도 직관적으로 생각했을 때 문법 문제는 가까이에 있는 문맥을 통해 알 수 있으며 단어의 순서에 크게 의존하기 때문에 window 크기가 작아야 하고 의미적인 부분은 국한된 부분만 보는 것보다 window size를 크게 보는 것이 낫기 때문이라는 것을 알 수 있다.

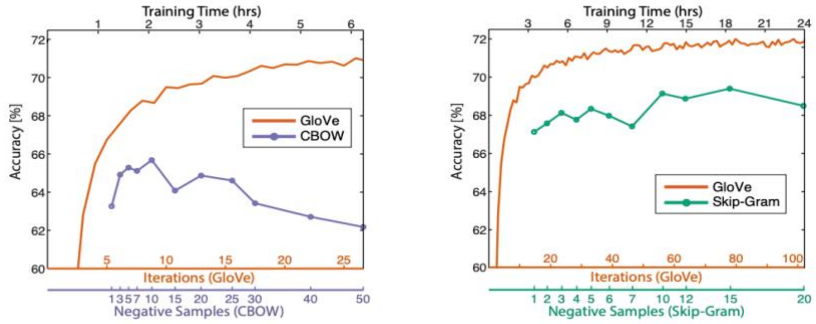
-Corpus Size




위 그래프는 Word analogy task에 대해서 300차원의 벡터로 각각 다른 크기의 corpus에 대해 학습한 결과이다. **문법 task에서는 corpus의 크기가 커질수록 일관되게 성능이 높아진다**는 것을 알 수 있다. 더 큰 corpus가 일반적으로 더 나은 통계자료를 생성하기 때문일 것이다. 반면 **의미 task에서는 단순히 corpus의 크기보다는 어떤 corpus에 훈련했느냐 하는 것이 더 중요**하다. 그래프에서 볼 수 있듯이 Wikipedia corpus가 Gigaword corpus보다 크기는 작지만 semantic subtask에서의 성능은 더 좋게 나타나는데 그 이유는 Wikipedia가 지역에 대해 포괄적인 글들을 포함하고 있기 때문에 city-and

country based analogy에서 Wikipedia가 더 우세한 것으로 보인다. 또한 Wikipedia는 고정된 뉴스 리포지터리인 Gigaword와 다르게 새로운 지식을 이해하도록 입력 값이 업데이트 된다.

-Run-time




(a) GloVe vs CBOW (b) GloVe vs Skip-Gram

총 run-time은 X채우기와 모델 훈련 시간으로 나뉘는데 X채우기는 window size, vocabulary size, corpus size 등 여러가지 요인에 영향을 받는 반면 **모델 훈련 시간은 vector size와 iteration에 영향을 받는다.**

-Comparison with

GloVe와 word2vec의 철저한 양적인 비교는 영향을 줄 수 있는 파라미터들이 여러가지가 있기 때문에 복잡하다. 따라서 vector length, context window size, corpus, vocabulary size 와 같이 변동에 주요하게 작용하는 요인들을 제어한다. 남아있는 가장 중요한 변수는 training time 이고, GloVe모델에서 관련된 파라미터는 training iteration이다. word2vec에서는 training epoch지만 최근에는 코드가 하나의 epoch로 되어있기 때문에 the number of negative sample을 컨트롤하는데 이 negative sample을 추가함으로써 training word의 개수를 증가시키고 이는 epoch를 늘리는 것과 비슷한 효과를 낸다. 따라서 GloVe와 CBOW, 그리고 GloVe와 SG를 비교하는 위의 그래프는 각각 2개의 x축을 가지게 된다. 그래프를 보면 word2vec모델의 성능은 negative sample의 개수가 10을 넘어갈 경우 떨어지게 된다.

종합해보면 같은 corpus, vocabulary, window size, training time에 대해 **GloVe는 꾸준히 word2vec과 비교해 더 나은 결과를 낸다**. 더 빠른 시간 안에 더 나은 결과를 내고, 속도를 무시하고도 최고의 결과를 낸다.

<Conclusion>

최근에 distributional word representation의 학습에 대해 count-based methods가 나은지 prediction-based method가 나은지에 대한 질문이 화두가 되고 있다. (+word2vec은 prediction-based method이고 GloVe는 count-based method이다.) 이 논문에서는 두 가

지 방법에 큰 차이가 없다고 보지만 count-based 방식의 global statistic을 담고 있다는 효율성이 이점이 될 수 있다. **결론적으로 GloVe는 count data를 잘 활용하는 동시에 word2vec모델의 장점인 의미 있는 선형 하위 구조를 담고 있는 모델이며, word analogy, word similarity, named entity recognition task 모두에서 다른 모델들과 비교해 우수한 성능을 보여준다.**