Ch4. 단어 수준 임베딩

- 예측 기반: NPLM, Word2Vec, FastText
- 행렬 기반: LSA, GloVe, Swivel

가중 임베딩 (Weighted Embedding): 문장 수준으로 확장

예측 기반

NPLM (Neural Probabilistic Language Model)

통계 기반 전통적 언어 모델의 한계

- 학습 데이터에 존재 하지 않는 경우 (확률값 0)에 대해 back-off, smoothing이 가능하지만 완전하지 않다
- 문장의 long-term dependency 포착이 어렵다 (최대 n=5)
- 단어, 문장 간 유사도 계산을 못한다

NPLM: 직전 등장 n-1개의 단어들로 다음 단어 맞추는 n-gram LM

context 단어들이 target 단어와 관계를 지니게 된다

→ context 단어들 벡터가 벡터 공간에서 같은 방향으로 업데이트

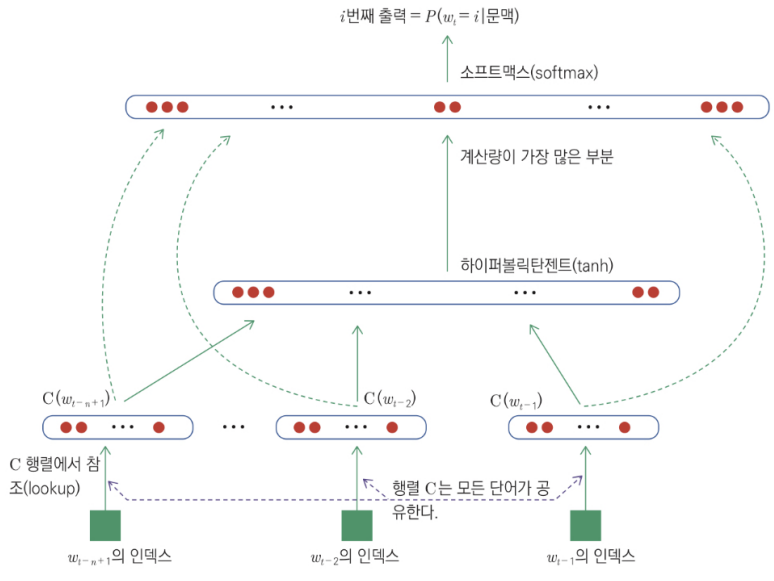

그림 4-1 NPLM(Bengio et al., 2003)

C 행렬에서 해당 단어를 lookup 하여 word vector를 구한다

lookup: 단어의 원핫 벡터 (1,|v|) * C 행렬 (|V|,dim_w)

Word2Vec

두개의 논문으로 나누어 발표

- Efficient Estimation of Word Representations in Vector Space (Mikolov et al. 2013a)

  Skip-Gram, CBOW
- Distributed Representations of Words and Phrase and their Compositionality (Mikolov et al. 2013b)

  Negative Sampling 등 학습 최적화 기법

모델 구조 & 파라미터

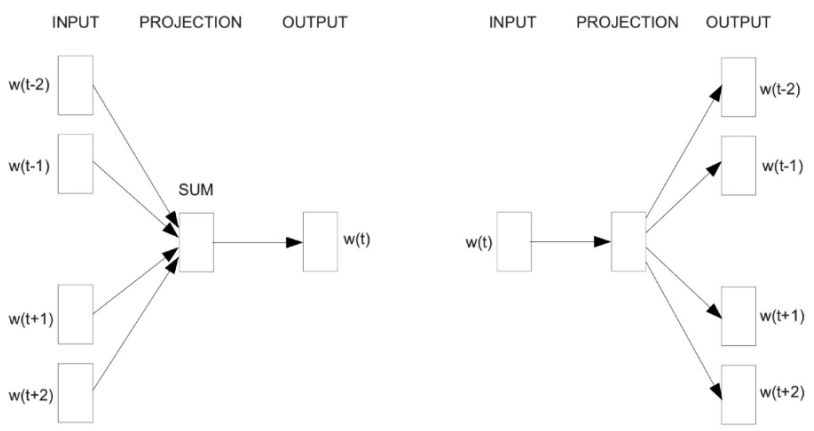

그림 4-7 CBOW와 Skip-gram 모델(Mikolov et al., 2013a)

- CBOW: 문맥(t-2~t+2)으로 타겟(t) 맞춤
- **Skip-Gram :** 타겟으로 문맥 단어(하나씩) 맞춤
  - 같은 말뭉치로 더 많은 학습 데이터 확보 가능 → CBOW보다 품질 나은 경향
  - ex) CBOW 데이터 {[t-2,t-1,t+1,t], t}
  - Skip-gram 데이터: {t,t-2}, {t,t-1}, {t,t+1},{t,t+2}

Skip-gram 학습 파라미터: U (|V|,d), V (d,|V|) 행렬

U_t*V_c t 단어 & c 단어 유사도 계산하는 것으로 생각

Skip-gram을 단어 예측으로 학습하기엔 **Softmax 때문에 연산량이 많다**

→ 주어진 context가 맞는지 아닌지 **Positive,Negative Binary CLF**로 해결 (Negative Sampling)

Positive Sample:

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

- Sigmoid로 Binary Classification, log-likelihood 최대화
- 위 식 분모를 줄여야 확률이 상승 → (-내적)을 줄여야 → 내적을 키워야 → 유사도 키워야 (cosine similarity와 비례)

학습 데이터 구축

- Positive Sample: t 단어 주변에 실제로 등장한 문맥 단어 c
- Negative Sample: 주변에 등장하지 않은 단어 (랜덤 추출)
  - 1개의 Positive에 k개의 Negative 계산
  - 시그모이드 k+1 회 계산 << 1 스텝에 Vocab 사이즈 소프트맥스 계산
  - 작은 데이터 에서는 k 5~20, 큰 데이터 에서는 2~5가 성능 좋음

SGNS: Skip-gram Negative Sampling

Negative Sample 뽑는 방법:

말뭉치 자주 등장하지 않는 희귀한 단어가 네거티브 샘플로 조금 더 잘 뽑힐 수 있도록 설계

$$P_{negative}(w_i) = \frac{U(W_i)^{\frac{3}{4}}}{\sum U(w_j)^{\frac{3}{4}}}$$

U(W): 해당 단어의 unigram 확률 (해당 단어 빈도/전체 단어 수)

3/4 처리 함으로써 0.01 이었던 희귀 단어가 0.03으로 좀더 높아짐

Subsampling: 자주 등장하는 단어는 학습에서 제외

$$P_{subsampling}(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

f(w): w의 빈도 (frequency)

빈도가 높을 수록 분모가 커짐 → 루트항이 작아짐 → subsample 확률 커짐

Subsample 확률은 **제외할** 확률 값이 클수록 학습에서 제외

Subample 값 낮을 수록 **나올때 마다 빼놓지 않고** 학습

- 전체 흐름: Negative Sample로 먼저 선정 → Subsample로 필터링?

모델 학습

Positive의 log likelihood를 최대화 한다

한번 업데이트 할 때 1쌍의 positive, k쌍의 negative를 학습

$$L(\theta) = \log P(+|t_p, c_p) + \sum \log P(-|t_{n_i}, c_{n_i})$$

학습 후

U나 U+V^T를 d차원의 임베딩, U;V^T를 2d 차원의 임베딩 등으로 활용 가능

- Stochastic Gradient Descent로 학습 (배치 1)

FastText (Facebook 17)

(TACL 17) Enriching Word Vectors with Subword Information

각 단어 문자를 **Character 단위** n-gram으로 표현

Negative Sampling, 학습 등은 Word2Vec과 동일

target 단어는 character n-gram, context 단어는 그냥 단어 단위로 - ?

U: Character n-gram Vocab

V: 단어 단위 n-gram

단어 임베딩(u) : 문자 단위 n-gram 벡터(z) 들의 **합**

$$u_t = \sum z_g$$

ex) 시나브로 → <시나브로> : <, >는 단어 시작, 끝 나타내는 토큰

G_t: t 단어에 속한 n-gram 집합 (전체 단어도 포함)

3-gram G_t ={<시나, 시나브, 나브로, 브로>, <시나브로>}

Target,Context 쌍 학습 할 때 target에 속한 n-gram 벡터 **모두 업데이트**

Character N-gram으로 분리 함으로써

- 용언 활용, 관계 어미들이 가깝게 임베딩 됨
- **오타, Unknown에 대해 robust 하다**

FastText는 조사,어미가 발달한 한국어에 좋은 성능 낼 수 있다

한글 텍스트를 자소 단위로 분리 (jamo_sentence)

나는 학교에.. → ㄴㅏㄴㅡㄴㅎㅏㄱ..

각 자소를 하나의 문자로 본다

어휘 집단에 속한 단어들 또한 자소로 분리되어 있음

쿼리 단어 또한 자소 단위로 분해한 뒤 FastText 임베딩 추출

행렬 분해 기반

Latent Semantic Analysis (LSA) - 잠재 의미 분석

단어-문서 행렬,TF-IDF 행렬, 단어-문맥 행렬등 커다란 행렬에 SVD 수행

- 데이터 차원 수 축소 → 계산 효율성 키움
- 숨어 있는 잠재 의미 이끌어냄
- 도출 되는 행 벡터들을 단어 임베딩으로 사용

LSA를 적용하면 단어와 문맥간 내재적인 의미 (latent, hidden meaning)을 효과적으로 보존 할 수 있게 된다고 주장

입력 데이터의 **노이즈, 희소성(sparsity) 줄일 수 있다**

SVD (Singular Value Decomposition)

PPMI 행렬

PPMI:

PMI 수식에서 분자<분모일 경우 음수가 된다

→ 음수가 될 경우 신뢰가 어렵 → 음수를 0으로 치환한 PPMI 이용

Shifted PMI:

PMI에서 logk를 뺀 값(임의의 양수 k)

$$PMI(A,B) - logk$$

행렬 분해로 이해하는 잠재 의미 분석

SVD 중 truncated SVD는 특이값 가운데 가장 큰 d개만 가지고 해당 특이 값에 대응하는 singular vector들로 A를 근사하는 기법

→ m개 단어, n개 문서로 표현한 행렬에 Truncated SVD 실행하면

U 행렬은 d 차원으로 **각 단어 벡터** 표현 가능

V 행렬은 d차원으로 **각 문서 벡터** 표현 가능

행렬 분해로 이해하는 Word2Vec

Word2Vec의 학습은 Shifted PMI 행렬을 U와 V로 분해하는 것과 같다

$$U_i \cdot V_j = PMI(i,j) - \log k$$

우변의 k: skip-gram 모델의 네거티브 샘플 수

k=1 이면 Skip-gram은 PMI 행렬 분해하는 것과 같다

GloVe

Global Word Vectors (Pennington et al. 14) 단어 임베딩 기법

단어-문맥 행렬을 분해

Word2Vec과 LSA 기법의 단점을 극복하고자

- Matrix Factorization 방식:
  - 말뭉치 전체 통계량 모두 활용할 수 있지만 결과물로 단어간 **유사도 측정 어렵다** (단어를 표현하는데에는 좋지 않다)
  - SVD 계산량이 많다
- Local context window 방식:
  - **Local Context만 학습 한다** → 전체 통계 정보 반영 어렵다

Matrix Factorization의 **global**한 co-occurrence 정보와 skip gram의 **벡터 표현**을 가져온 모델

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

GloVe Cost Function

- X: word-word co-occurrence counts
- X_ij: 두 단어 i,j가 동시에 나타난 **횟수** (i: target, j: context)
- f(X_ij): Weighting Function → X_ij 값이 0일 경우 log값이 무한대로 커지는 문제

- f(0) = 0
- f(x) should be non-decreasing
  - rare co-occurrences are not overweighted
- f(x) should be relatively small for large x
  - frequency co-occurrences are not overweighted
- x_max 값은 100, alpha는 3/4로 이용

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

행렬 U,V는 랜덤 초기화 → Gradient Descent로 업데이트

학습이 끝나면 U를 단어 임베딩으로 쓰거나 U+V^T, V;V^T 등을 임베딩으로 사용

정리:

벡터 내적으로 유사도 기준으로 학습 하면서 global co-occurrence 정보를 학습

Swivel

Submatrix-Wise Vector Embedding Learner (Google - Shazeer et al. 16)

PMI 행렬 분해 기반의 단어 임베딩 (Glove는 단어-문맥 행렬)

단어 i,j가 같이 자주 등장 할 수록 두 단어 벡터의 내적이 PMI와 일치하도록 강제

동시 등장 여부 유무 따라서 목적함수 분리

Table 1. Training objective and cost functions. The $\mathbf{pmi}^*$ function refers to the "smoothed" PMI function described in the text, where the actual value of 0 for x_{ij} is replaced with 1.

Count	Cost	Intuition
$x_{ij} > 0$	$\frac{1}{2} f(x_{ij})(w_i^\top \tilde{w}_j - \mathbf{pmi}(i;j))^2$	Squared error: the model must accurately reconstruct observed PMI subject to our confidence in x_{ij}.
$x_{ij} = 0$	$\log\left[1 + \exp\left(w_i^\top \tilde{w}_j - \mathbf{pmi}^*(i;j)\right)\right]$	"Soft hinge:" the model must not *over*-estimate the PMI of common features whose co-occurrence is unobserved.

- f(x_ij): 동시 등장 빈도
  - 동시 등장 빈도 높을수록 PMI값과 같도록 더 강제
- PMI*: 동시 등장 횟수를 0대신 1로 가정한 PMI
  - log 0인 경우 -inf로 발산하는 문제

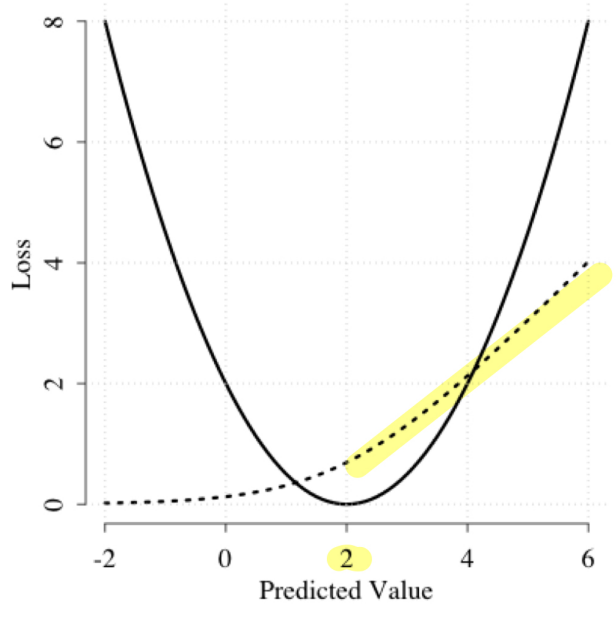

Figure 3. Loss as a function of predicted value $w_i \tilde{w}_j^\top$, evaluated for arbitrary objective value of 2.0. The solid line shows $\mathcal{L}_1$ (squared error); the dotted line shows $\mathcal{L}_0$ (the "soft hinge").

Soft Hinge Loss 쓰는 이유:

- loss penalizes the model for **over-estimating**
- 내적값을 작게 예측하여 관계가 없다고 하는것(anti-correlation)은 허용되지만
- PMI*값에 대해서
  - i,j가 **각각 고빈도**이지만 같이 등장 x → **의미상 무관할것** (각각 많이 나왔는데도 같이 등장 안한거면 좀더 확실) → 내적값 작게 유지
    - PMI*값이 작다 → 내적값이 커지면 로스가 금방 커져서 바로 Penalize
  - i,j가 **각각 저빈도** 이다 → 의미상 관계가 일부 있을 수도 있으나 말뭉치 크기가 작아 **우연히 동시 빈도가 낮을 수도** → 내적값이 커지는것 허용
    - PMI*값이 크가 → 내적값이 약간 커도 loss가 크게 늘지 않다

어떤 단어 임베딩을 사용할 것인가

단어 임베딩을 평가하는 방법

- 단어 유사도 평가 (word similarity test)
- 유추 평가 (word analogy test)

단어 유사도 평가

단어 쌍을 미리 구성한 후 벡터간 Cosine Similarity와 사람 평가 점수 **상관관계(correlation)** 계산하여 평가

- Spearman, Pearson Correlation 이용 (1에 가까울수록 상관관계 강함)
- 책 실험: Word2Vec, FastText 같은 예측 기반 임베딩들이 GloVe, Swivel등 행렬분해 방법 비해 상관관계가 상대적으로 강했음
  - 예측 기반 방법들이 의미적 관계가 잘 녹아 있다고 해석

단어 유추 평가

벡터간 계산 (+,-) 통해 의미론적 유추

- 갑 - 을 + 병 → 정
- 대한민국 - 서울 + 일본 → 도쿄

Google analogy 참고한 단어 유추 평가 데이터셋 (420개)

계산된 벡터와 코아싱ㄴ 유사도가 가장 높은 단어를 단어로 예측

- 책 실험: Word2Vec, Glove가 상대적으로 선방

단어 임베딩 시각화

임베딩 품질을 정성적, 간접적으로 확인하는 기법

t-SNE로 차원 축소하여 시각화

t-**Stochastic** Neighbor Embedding: Stochastic: 거리 정보를 확률적으로 나타내기 때문

고차원 벡터 이웃간 거리를 최대한 보존하는 저차원 벡터 학습

Bokeh 라이브러리로 시각화

가중 임베딩

단어 임베딩을 문장 수준 임베딩으로 확장하는 방법 (ICLR - Arora et al. 2016)

문서 내 단어의 등장은 저자가 생각한 주제에 의존 한다 → 주제에 따라 단어 사용 양상이 달라질 것

주제 벡터 c_s가 주어졌을 때 단어가 나타날 확률 정의

$$P(W|C_s) = \alpha P(w) + (1-\alpha)\frac{exp(\tilde{C_s} \cdot V_w)}{Z}$$

C_s*V_W: 주제 벡터와 단어 벡터가 유사할 수록 확률 올라가도록

alpha: hyperparameter

새로운 단어 벡터를 만들 때 가중치는 해당 단어가 얼마나 말뭉치에 자주 등장하는지 P(w) 감안해서 만듬

- 희귀한 단어 → 높은 가중치 P(w) → Norm 키운다
- 고빈도 단어 → 낮은 가중치 → 크기 줄임

문장 등장 확률: 문장에 속하는 단어들 확률 누적 곱

문장이 등장할 확률을 최대화하는 주제 벡터는 문장에 속한 단어들에 해당하는 단어 벡터에 가중치를 곱해 만든 새로운 벡터들의 합에 비례한다.

가중치: 해당 단어가 말물치에 얼마나 자주 등장하는지, P(w)를 감안해 만든다

Additional Reference

[1] https://aegis4048.github.io/demystifying_neural_network_in_skip_gram_language_modeling

[2] https://www.sallys.space/blog/2018/04/30/glove/

[3] https://misconstructed.tistory.com/40

[4] Pennington, Jeffrey et al. “GloVe: Global Vectors for Word Representation.” *EMNLP* (2014).