A Neural Probabilistic Language Model

paper review

9기 김은혜

발표 순서입니당 휴

- 1. Vectors, Conditional Probability
- 2. Language Models
- 3. Before NLPM
- 4. Distributed Representation
- 5. Model Architecture
- 6. Results

Vectors

an n-dimensional vector (-vector) x over the field F (R or C from now on) is an ordered list of scalars, written as

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \; x_i \in \mathbf{F}, \; i = 1, \ldots, n.$$

- n is the *dimension* or *size* of x (in x_i, i is the *index*)
- x_i : ith *element*, *component*, *entry*, or *coefficient*

inner product

Inner product (or dot product) of two n-vectors X, Y is

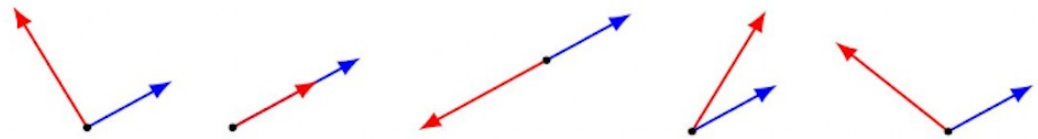
$$x \cdot y = x^T y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n = \sum_{i=1}^{n} x_i y_i$$

Angle of Vectors

orthogonal, aligned, anti-aligned, acute, obtuse

the angle between two nonzero n-vectros x, y defined as

$$\theta = \angle(x, y) = \arccos(\frac{x^T y}{\|x\|\|y\|}), \quad \cos\theta = \frac{x^T y}{\|x\|\|y\|}$$



application of angles

- measure the similarity (or dissimilarity) by angle of vectors
- cosine similarity
-example: document similarity

	Veterans Day	Memorial Day	Academy Awards	Golden Globe Awards	Super Bowl
Veterans Day	0	60.6	85.7	87.0	87.7
Memorial Day	60.6	0	85.6	87.5	87.5
Academy A.	85.7	85.6	0	58.7	85.7
Golden Globe A.	87.0	87.5	58.7	0	86.0
Super Bowl	87.7	87.5	86.1	86.0	0

Conditional Probability
조건부 확률

사건 A 가 주어졌을 때, 사건 B의 조건부 확률

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

$$P(w_i = i|context) = \frac{\exp(y_{w_t})}{\sum_i \exp(y_i)}$$

$$P(w_t|w_1, w_2, \cdots, w_{t-1}) \approx P(w_t|t_{t-(n-1)}, \cdots t_{t-1})$$

$$P(\theta|\mathcal{D}) = P(\theta)\frac{P(\mathcal{D}|\theta)}{P(\mathcal{D})}$$

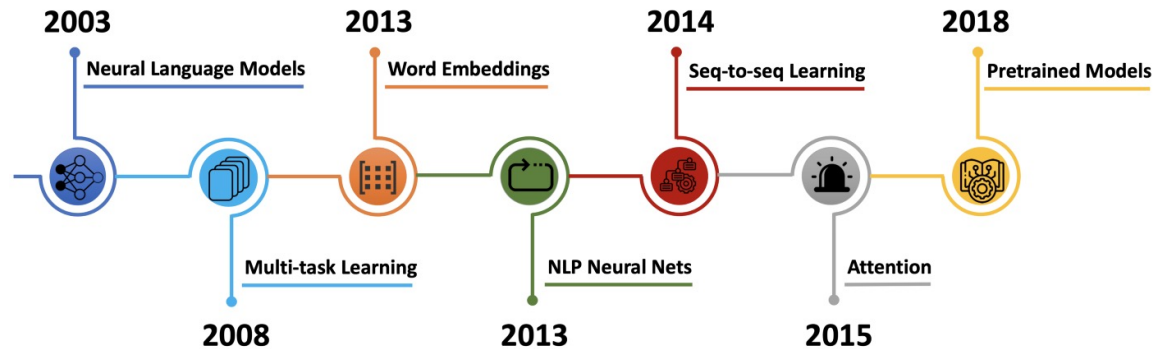
- $P(\theta|\mathcal{D})$: 사후확률 (posterior)
- $P(\theta)$: 사전확률 (prior)
- $P(\mathcal{D}|\theta)$: 가능도 (likelihood)
- $P(\mathcal{D})$: Evidence

Language Models

-언어 모델이란, 단어 시퀀스 (Sequence) 에 확률을 할당하는 모델

-단어 Sequence 는 단어들의 배열을 말함.

-다음 단어의 등장 확률을 예측하는 모델이다.



언어 모델의 활용 사례

Google

Google 검색창에 다음 예측어가 나오는 것.

3. Before NPLM
Statistical Language Modeling

- 목적: 단어들의 배열Sequence에 대한 결합 확률 함수joint probability function 을 학습하는 것
- 논문이 제안하는 모델은 단어의 분산 표상distributed representation 을 배우는 것과 더불어, 단어들의 배열에 대한 확률 함수를 배움.
- Count 기반의 조건부 확률적 접근

$$P(w_1, w_2, w_3, w_4, w_5, \ldots w_n) = \prod_{n=1}^{n} P(w_n | w_1, \ldots, w_{n-1})$$

P (오늘은 날씨가 매우 좋군요)

P (오늘은)

P (날씨가 | 오늘은)

P (매우 | 오늘은 날씨가)

P (좋군요 | 오늘은 날씨가 매우)

3. Before NLPM
Curse of Dimensionality in Statistical Language Modeling

- **Curse of Dimensionality 차원의 저주: 고차원 공간에서 데이터를 다룰 때 나타나는 성능 저하 현상**
- The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces that do not occur in low-dimensional settings such as the three-dimensional physical space of everyday experience (Wiki)
- **고차원 공간에서 발생하는 이유 → Bag Of Words 모델의 문제점**
- **Bag of Words: 단어들의 출현 빈도(frequency, count)에만 집중하는 텍스트 데이터의 수치화 표현 방법**
- **Ex: 498, 867개의 한국어 단어를 Bag of Words 화 하면 약 50만개의 차원으로 문장이 모델링됨. (고차원)**
- **One-Hot Encoding 과 유사하지만 문장 혹은 단락과 같은 단어 Sequence 의 표현 방법**

```
(1) 각 단어에 고유한 정수 인덱스를 부여합니다.  # 단어 집합 생성.
(2) 각 인덱스의 위치에 단어 토큰의 등장 횟수를 기록한 벡터를 만듭니다.
```

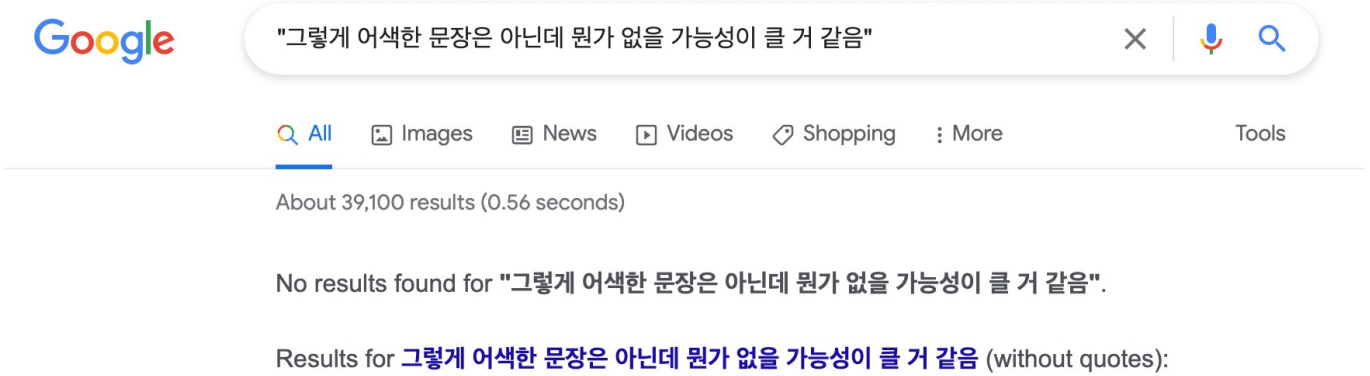
```
doc2 = '소비자는 주로 소비하는 상품을 기준으로 물가상승률을 느낀다.'

vocab, bow = build_bag_of_words(doc2)
print('vocabulary :', vocab)
print('bag of words vector :', bow)
```

```
vocabulary : {'소비자': 0, '는': 1, '주로': 2, '소비': 3, '하는': 4, '상품': 5, '을': 6, '기준': 7, '으로': 8, '물가상승
률': 9, '느낀다': 10}
bag of words vector : [1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1]
```

3. Before NLPM
Curse of Dimensionality in Statistical Language Modeling

Problems Occuring with Bag of Words

- 1. Data Scarcity (Sparsity Problem) : 관찰한 데이터와 입력 받는 데이터 간 괴리가 발생함.



정확하게 일치하는 검색어가 나오지 않는다! Google 이 Training 한 데이터에서 저 문장과 정확히 일치하는 문장이 없었기 때문.

한국어가 50만개의 단어라면, 그 단어로 생성할 수 있는 문장의 개수는 무한대임.

즉, Data Scarcity (데이터 부족) 문제가 발생하는 것.

= Sparsity Problem : 새로운 단어 조합에 대해서 확률값이 0이 되는 문제

3. Before NLPM
Curse of Dimensionality in Statistical Language Modeling

Problems Occuring with Bag of Worrds

- 1. Data Scarcity (Sparsity Problem

Sparsity Problem : 새로운 단어 조합에 대해서 확률값이 0이 되는 문제

P (오늘은 날씨가 야호 좋군요)

P (오늘은) | P (날씨가 | 오늘은) | P (야호 | 오늘은 날씨가) | P (좋군요 | 오늘은 날씨가 야호)

$$\frac{count(\text{오늘은 날씨가 야호})}{count(\text{오늘은 날씨가})}$$

'오늘은 날씨가 야호'가 corpus 에 등장할 확률이 거의 0이다!

A corpus is **a collection of machine-readable texts that have been produced in a natural communicative setting**

3. Before NLPM
Curse of Dimensionality in Statistical Language Modeling

Problems Occuring with Bag of Words

- 2. 파라미터 낭비 문제가 발생함

Take the example of One-Hot Encoding

Ex: 50만개의 단어를 기반으로 한 벡터가 있다고 하고, 이 중에서 우리가 문장 속에서 10개의 단어만 쓴다면,

10개를 제외한 50만-10개의 파라미터 값들은 0이 될 것이고, 그러한 차원들은 놀고 있게 되는 것!

즉, 파라미터는 많아서 계산량이 늘지만 값이 0인 파라미터들인 것이 문제.

단어의 수에 비례하여 차원이 증가한다!

```
doc2 = '소비자는 주로 소비하는 상품을 기준으로 물가상승률을 느낀다.'

vocab, bow = build_bag_of_words(doc2)
print('vocabulary :', vocab)
print('bag of words vector :', bow)
```

```
vocabulary : {'소비자': 0, '는': 1, '주로': 2, '소비': 3, '하는': 4, '상품': 5, '을': 6, '기준': 7, '으로': 8, '물가상승
률': 9, '느낀다': 10}
bag of words vector : [1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1]
```

3. Before NLPM
Curse of Dimensionality in Statistical Language Modeling

Problems Occuring with Bag of Words

- 3. Discrete (이산형) 표현 상의 문제

Bag-of-Words 는 Count-Based 벡터화를 함.

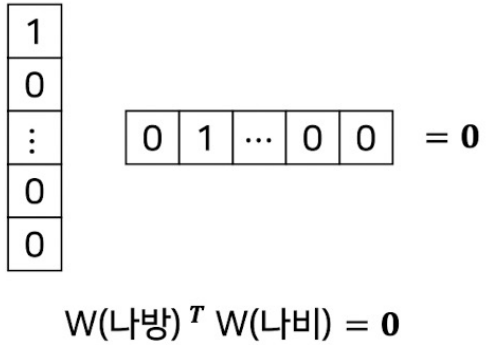
(Count-Based: 단어가 얼마나 사용되었는지 횟수에 따라 측정하는 것!)

즉, 모든 차원이 가지는 값은 정수값이 된다.

논문에서 discrete 한 표현 법은 일반화generalization 능력이 떨어진다는 지적을 함.

즉, 한 차원의 값이 조금만 바뀌어도 모델은 비교적 큰 영향을 받게 된다.

벡터간의 유사도는 두 단어 벡터를 내적해서 알게 되는데, BoW 는 관계 유사성을 알 수 없으며, 약간의 값 변화도 의미상 차이가 굉장히 클 수 있음.



3. Before NLPM
Attempts to Solve The Curse of Dimensionality

N-gram Model

- N-gram model 은 P(Context 문맥) 식을 근사한다. $\hat{P}(w_t|w_1^{t-1}) \approx \hat{P}(w_t|w_{t-n+1}^{t-1}).$
- 문맥 Context 이란?: 구하고자 하는 단어보다 앞선 n 개 단어 Sequence
- N=1 이면 unigram, N=2 이면 bigram, N=3 이면 trigram.....
- 논문이 나온 2003년 기점, GPGPU 가 존재하지 않았고 컴퓨터 성능이 부족해 작은 N 의 N-gram 사용.

P (오늘은 날씨가 야호 좋군요)

$$\text{Unigram LM} : p(w_1^N) = \prod_{n=1}^{N} p(w_n)$$

$$\text{Bigram LM} : p(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-1})$$

$$\text{Trigram LM} : p(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-2}, w_{n-1})$$

Unigram 의 예시

P (오늘은) P (날씨가) P (야호) P (좋군요)

Bigram 의 예시

P (오늘은) P (날씨가|오늘은) P (야호|날씨가) P (좋군요|야호)

여전히 Sparsity Problem 이 나타남!

3. Before NLPM
Attempts to Solve The Curse of Dimensionality

N-gram 의 Sparsity Problem 에 대한 대안

1. Additive (Laplace) Smoothing: 단어의 count 을 1씩 추가하여, 등장할 경우의 수 (count)가 0인 경우 없도록하는 것

P (오늘은 날씨가 야호 좋군요) 의 Unigram Example:
오늘은 +1, 날씨가 +1, 야호 +1, 좋군요 +1

2. Backoff Model

P(오늘은 날씨가 야호 좋군요) 의 Trigram Example:

P(좋군요 | 날씨가 야호) → count is 0

Then change to P(좋군요|야호) → count 가 그래도 높아짐!

P(좋군요 | 날씨가 야호) → α P(좋군요|야호) (α : trigram 에서 bigram 되는 통계량을 보정하는 상수)

3. Smoothed (Interpolated) Model

결측치 Interpolation (보간법)과 비슷함! Unigram, Bigram, Trigram 의 Linear Combination 이용

λ 1 * P(좋군요) + λ 2 * P(좋군요|야호) + λ 3* P(좋군요|날씨가 야호)

3. Before NLPM
Curse of Dimensionality in Statistical Language Modeling

그래도 문제는....

단어의 유사성에 대해 고려하지 않는다! (Semantically Similar 한 것을 인식하지 못한다)

전통적인 모델들 (Statistical Language Modeling, N-grams) 은 훈련 데이터에서 문장 또는 단어가 발견될 확률을 통계적으로 추정함.

하지만, 이 과정에서 단어의 유사성이 전혀 고려되지 않음.

Take the Paper's Examples

S1: A cat is walking in the bedroom.

S2: A dog was running in a room.

S3: A soldier shouted ”Fire in the hole!”

S1	S2	비교
The	A	관사
cat	dog	비슷한 크기의 애완 동물
is	was	시제가 다른 be동사
walking	running	동명사 - 이동을 묘사
in	in	같은 전치사
the	a	관사
bedroom	room	room이 bedroom을 포함함

Sentence	a	bedroom	cat	dog	everyone	fire	hole	in	is	room	running	soldier	the	walking	was	shouted
S1	1	1	1	0	0	0	0	1	1	0	0	0	1	1	0	0
S2	2	0	0	1	0	0	0	1	0	1	1	0	0	0	1	0
S3	1	0	0	0	0	1	1	1	0	0	0	1	1	0	0	1

- 벡터화 하면, Cosine Similarity 는 S1 & S3 가 높음.
- (The, Dog) =0 (완전 다름)
- (Dog, Cat) =0 (완전 다름)
- (Is, Was) = 0 (완전 다름)
- (Is, Is) = 1 (완전 동일) 하지만 우리는 직관적으로 dog, cat 이 비슷한 존재임을 알 수 있음!

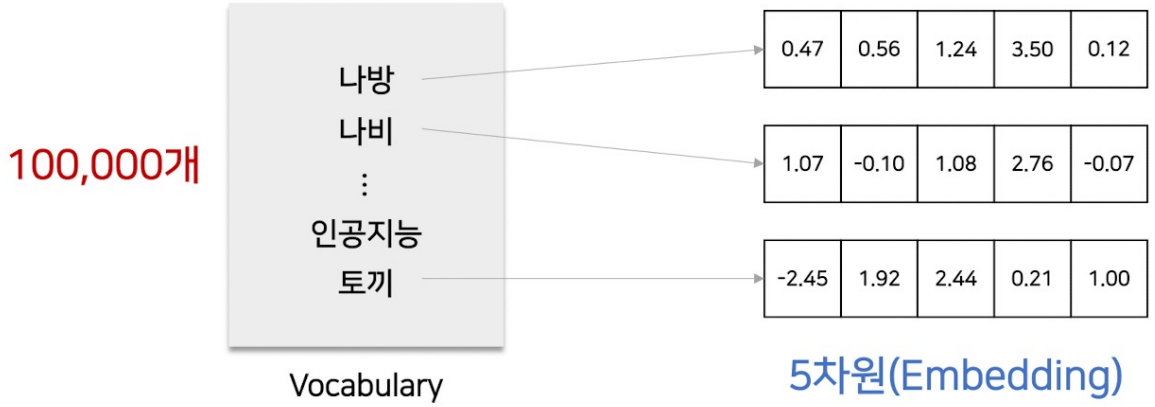
- `S1` 과 `S2` 의 유사도는 `0.38`
- `S1` 과 `S3` 의 유사도는 `0.43`
- `S2` 과 `S3` 의 유사도는 `0.38` 로

4. Distributed Representation

- Curse of Dimensionality: 단어 간 유사도 알 수 없음, 노는 파라미터가 많음, 값의 변화에 민감함.
- Solution: 분산 표상 (Distributed Representation)

단어를 0과 1이라는 이산형 정수로 표시하는 것이 아니라, 연속형 실수로 표현하는 것.

- 분산 표상 기법에는 word embeddings, word2vec, character embeddings 이 존재함.

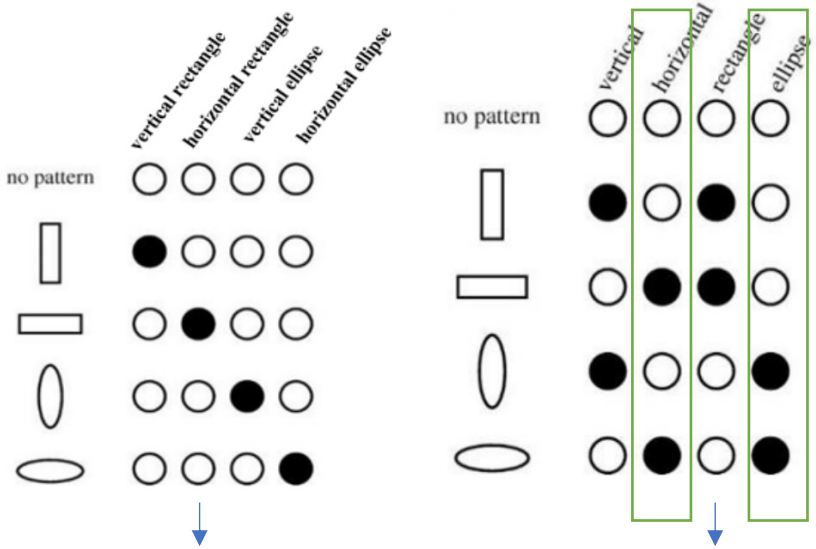


장점: 차원이 적기 때문에 필요 공간이 적음, 연속형 (Continuous) 이라서 내적하여 단어간 관계를 알 수 없음

실수값들이 조금 변화해도 의미적인 차이가 덜함 (Ex: 인공지능, 다람쥐)

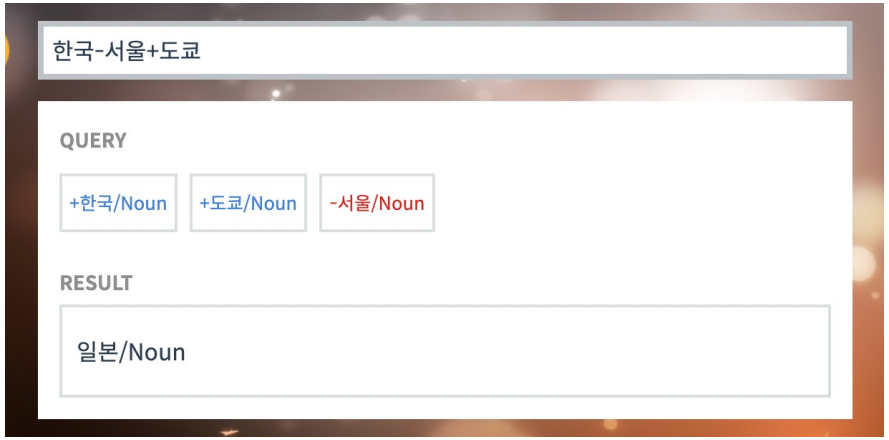
4. Distributed Representation

- **Distributed Representation 의 원리**



- One-Hot Encoding
- Not Distributed Representation!
- Need 4 digits

- Distributed Representation
- 초록색 박스는 not necessary
- Need only 2 digits



- **단어 벡터화에 대한 더 직관적인 이해가 필요하다면? : word2vec.kr**

4. Distributed Representation
Word Embeddings

- Word Embeddings 은 '단어' 를 vector 으로 embedding 하는 기법
- 어떤 규칙성으로 embedding 해야 함. (Bag of Words 도 벡터화 하지만, 이것은 Word Embeddings 이 아님)

- Word Embedding 의 전제: <u>Distributional Hypothesis</u>

같은 문맥에서 사용되거나 나타나는 (occur) 하는 단어들은 유사한 의미를 가지는 경향이 있다.

Words that are used and occur in the same contexts tend to purport similar meanings.

Thus, these vectors try to capture the characteristics of the neighbors of a word.

1) 같은 문맥에서 사용되는 단어들은 유사한 의미를 가지는 경향이 있음.
2) 문맥을 관찰 가능한 단어의 sequence 으로 볼 수 있음.
3) 서로 다른 두 단어가 사용되는 텍스트를 조사했을 때, 주변에서 나타나는 단어들의 패턴이 비슷하다면, 두 단어의 뜻도 유사할 것이라는 가정

즉, 한 단어를 vector 로 임베딩하기 위해서는 그 단어 자체만으로 불가능하고,

주변 단어들의 특징을 조사해야 하지만, distributional hypothesis를 따라는 embedding 이 가능해진다!

4. Distributed Representation

그래도 문제는....

그래서 논문이 제안하는 문제점에 대한 해결 방안!

1. 단어들을 distributed (분산된) word feature vector 로 표현하고 (=단어를 m 차원 벡터와 연관 짓고)

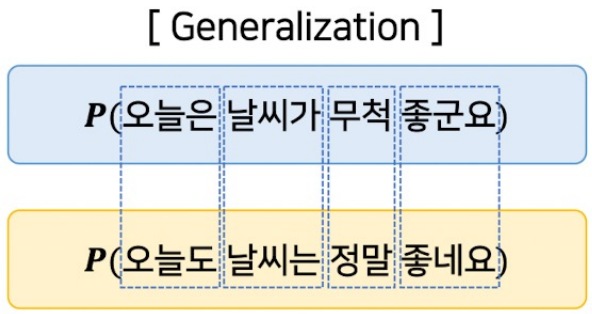
2. 단어들의 feature vector 들의 결합 확률 함수 (joint probability) 로 단어의 sequence 을 표현하고

(= m차원 벡터로 표현된 단어들의 조건부 확률을 표현하고)

3. Feature Vector 와 확률 함수의 파라미터들을 동시에 학습함 (=조건부 확률과 m 차원의 벡터를 동시에 학습함)

즉, Language Model 의 차원에서는 단어 Sequence 에 대한 joint probability function 을 나타내는 것이 목적이며,
Word Representation 차원에서는 Vocabulary 내 각 단어의 Embedded Feature 을 학습하는 것이 목적이다.

장점?: 유사 구조 및 유사 단어를 가진 문장의 generalization 이 가능해짐!



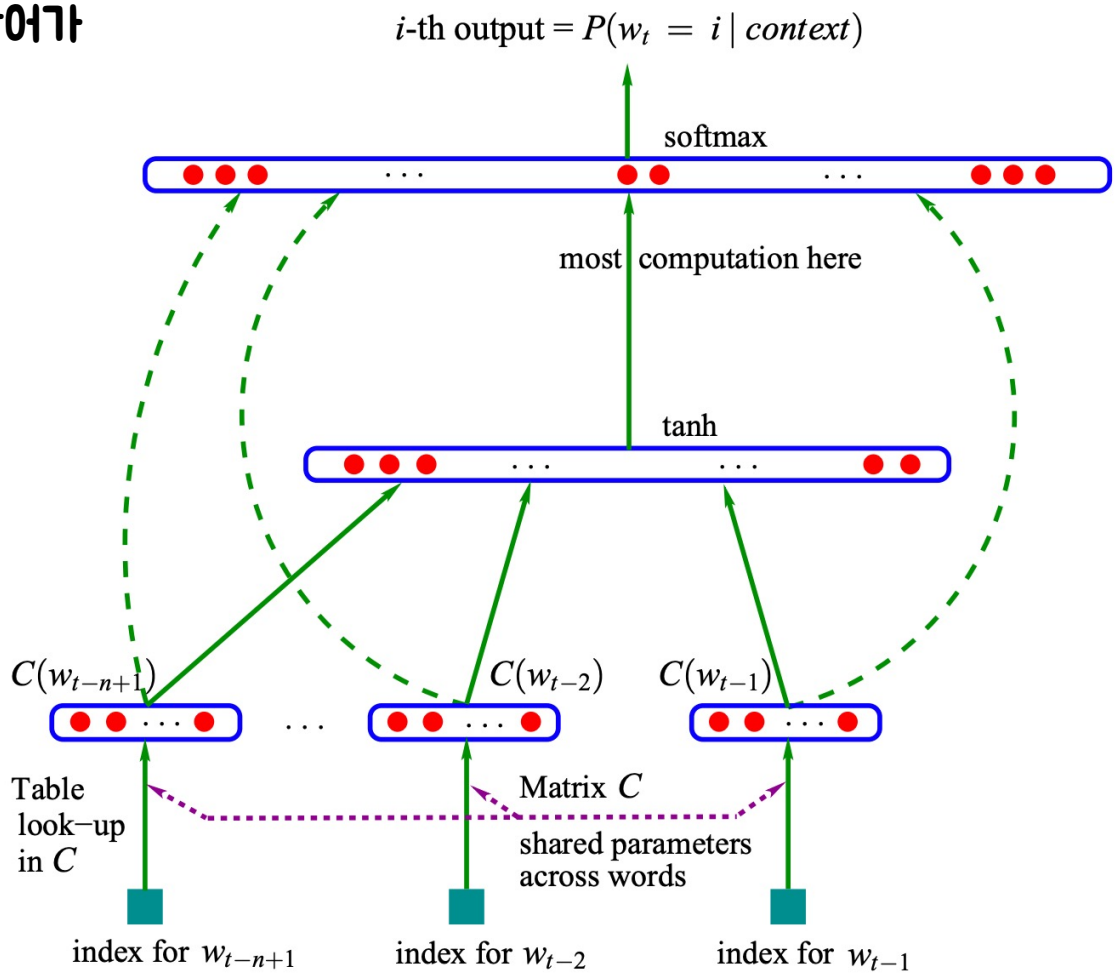
Semantically & Syntactically similar 한 단어들의 확률이 증가함.
Ex) Feature space 내 '무척' 이 등장하면 문법적/의미적 유사한 단어
'정말'이 등장할 확률도 높아짐.

5. Model Architecture

Training Set 은 Vocabulary Set V (corpus) 에 있는 단어 Sequence

모델의 목적: t 부터 t-n+1 까지의 단어들을 입력으로 넣었을 때, t 번째 단어가 나올 조건부 확률이 높게 나오도록 만드는 것!

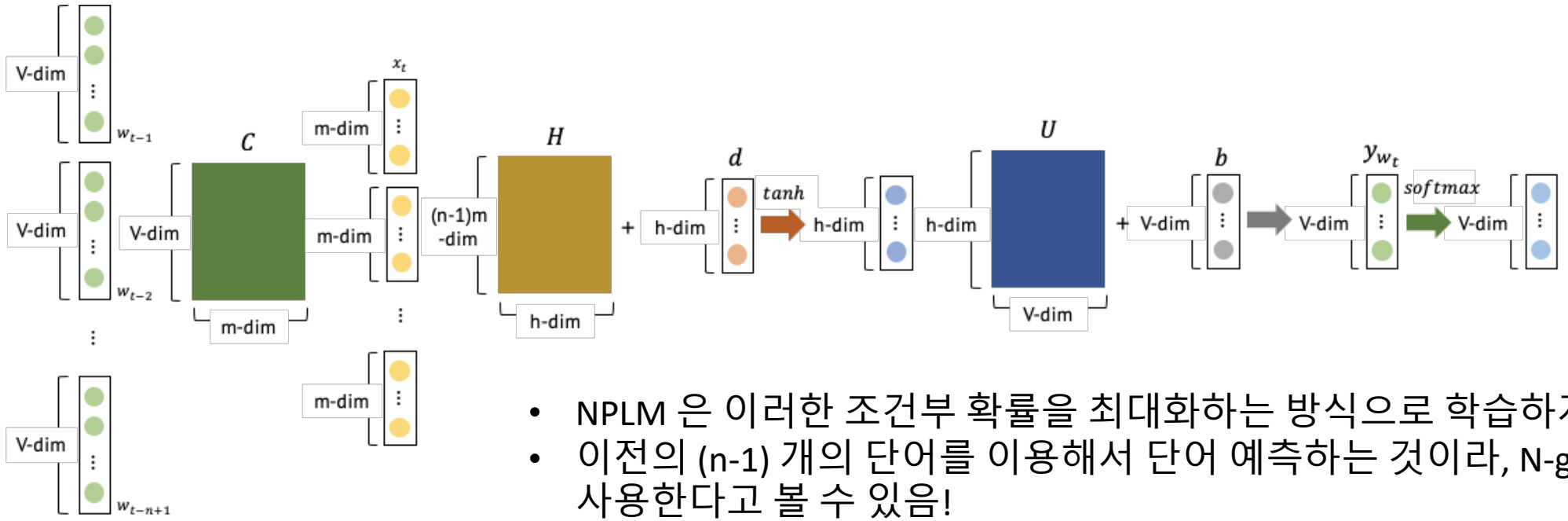
A **Neural Probablistic Language Model** is an early language modelling architecture. It involves a feedforward architecture that takes in input vector representations (i.e. word embeddings) of the previous n words, which are looked up in a table C.
The word embeddings are concatenated and fed into a hidden layer which then feeds into a softmax layer to estimate the probability of the word given the context.



NPLM의 출력층에서는 *V*-차원의 점수 벡터(score vector) 값을 *softmax* 함수를 통과시켜 나온 확률벡터를 제공한다.

확률 벡터에서 높은 확률값이 제공되는 인덱스의 단어가 (one-hot-vector를 사용하기 때문에) 실제 정답에 해당하는 단어와 일치하도록 학습을 진행한다.

5. Model Architecture

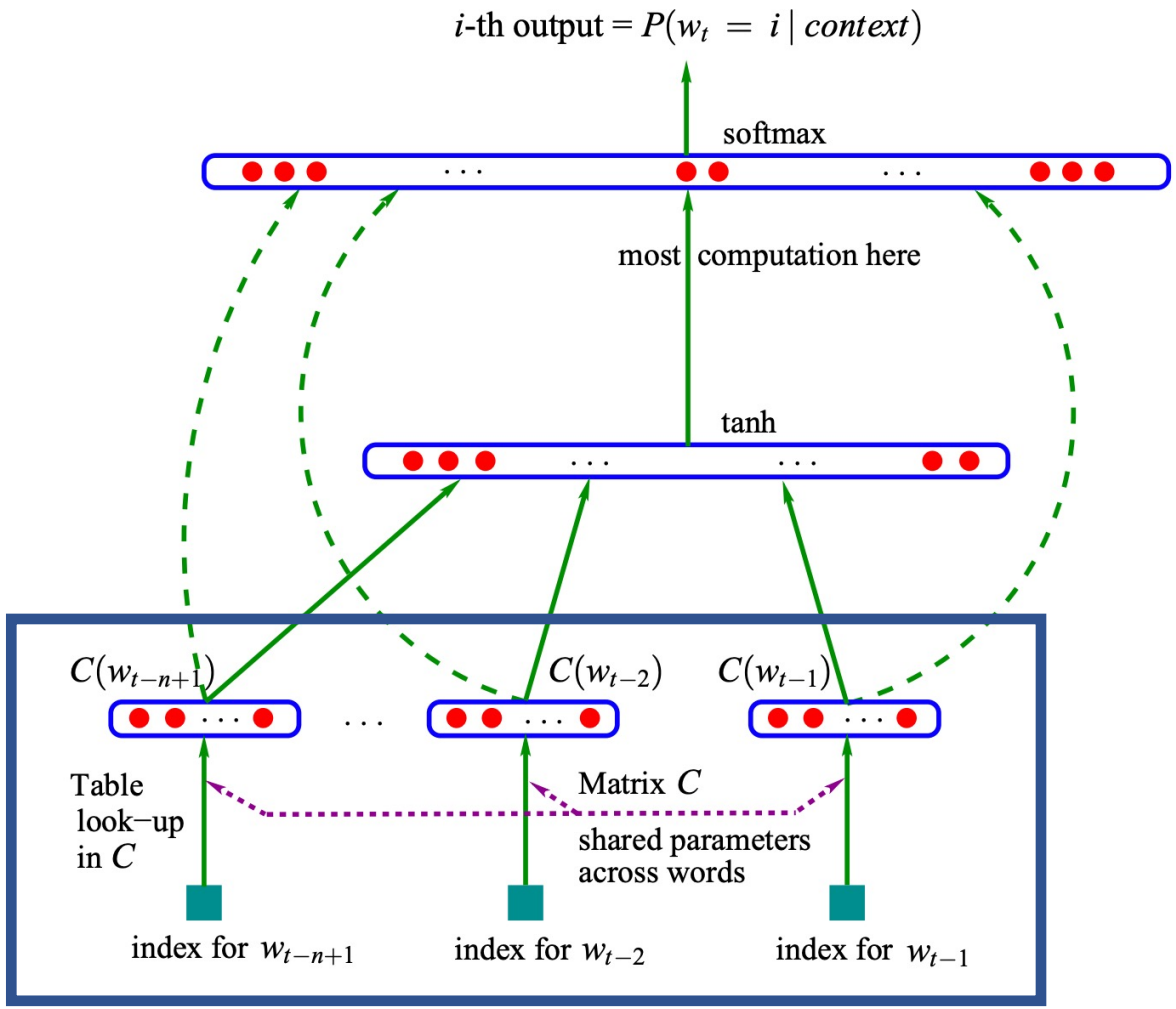


- NPLM 은 이러한 조건부 확률을 최대화하는 방식으로 학습하게 됨.
- 이전의 (n-1) 개의 단어를 이용해서 단어 예측하는 것이라, N-gram 모델을 사용한다고 볼 수 있음!
- 조건부 확률을 최대화 하려면, 분자를 최대화 및 분모 최소화 해야함.

$$P(w_t | w_{t-1}, \ldots, w_{t-n+1}) = \frac{exp(y_{w_t})}{\sum_i exp(y_i)}$$

w_i : i 번째 오는 단어 (word)
y_{w_i} : w_i 에 해당하는 점수 벡터 (Score Vector)

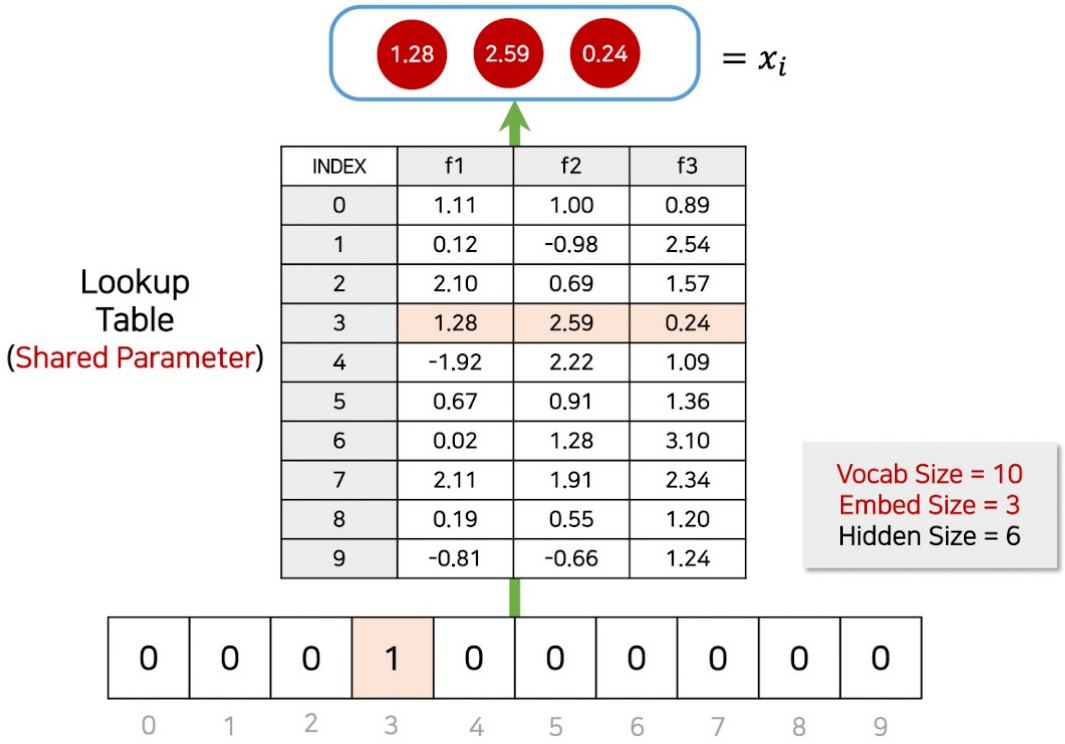
5. Model Architecture
Step 1. Input



5. Model Architecture
Step 1. Data Input

각 단어를 C 행렬을 통해 m 차원 벡터로 표현해야함.
단어를 m 차원 실수 벡터로 연관지어야 하는데, 논문에서는 이것을 Distributed Feature Vectors 이라고 함.

Look up table: An initialized array that contains precalculated information.



1. 행렬 C의 초기 값은 랜덤.

$$C = \begin{bmatrix} 11 & 18 & 25 \\ 10 & 12 & 19 \\ 4 & 6 & 13 \\ 23 & 5 & 7 \\ 17 & 24 & 1 \end{bmatrix}$$

발 없는 말이 천리

2. 각 단어에 해당하는 원-핫 벡터를 행렬 C와 내적
(Look-up table)

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 11 & 18 & 25 \\ 10 & 12 & 19 \\ 4 & 6 & 13 \\ 23 & 5 & 7 \\ 17 & 24 & 1 \end{bmatrix} = \begin{bmatrix} 23 & 5 & 7 \end{bmatrix}$$

5. Model Architecture
Step 1. Data Input

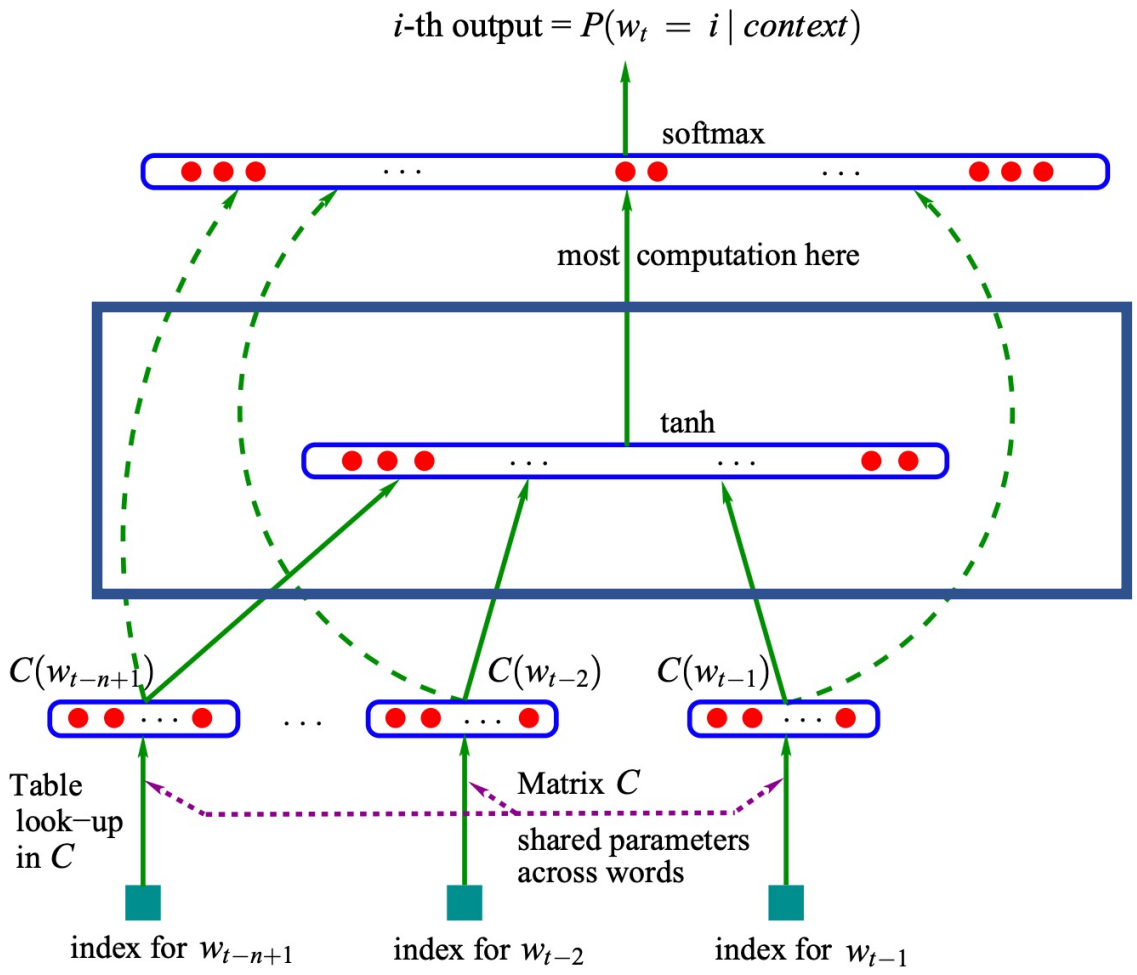
1.1 Mapping C
: distributed feature vectors
A |V| by m matrix of free-parameters
입력층에는 선행하는 단어들이 one-hot-vector 입력으로 들어옴. 각 벡터는 V 차원일 것인데,
C와의 연산을 통해 지정된 크기인 m 차원 벡터로 변환! $x_t = C \cdot w_t$ $C(i) \in \mathbb{R}^m$
이것은 C행렬에서 t 번째 열만 참조 (look up) 하는 것과 동일한 연산임. C 행렬의 i 번째 행을, i 번째 단어의 벡터라고 규정 지었으며, C 의 형태는 $|V| \times m$ 다.
이 값들이 training 과정에서 업데이트 되는, distributed probability vector

1.2 Mapping g
: input sequence of feature vectors to a conditional probability distribution with parameter w
G 는 feature vector 들을 t 번째 다음 단어를 위한 조건부 확률 분포와 mapping 시켜주는 함수이며, FC layer 이나 RNN 으로 구현

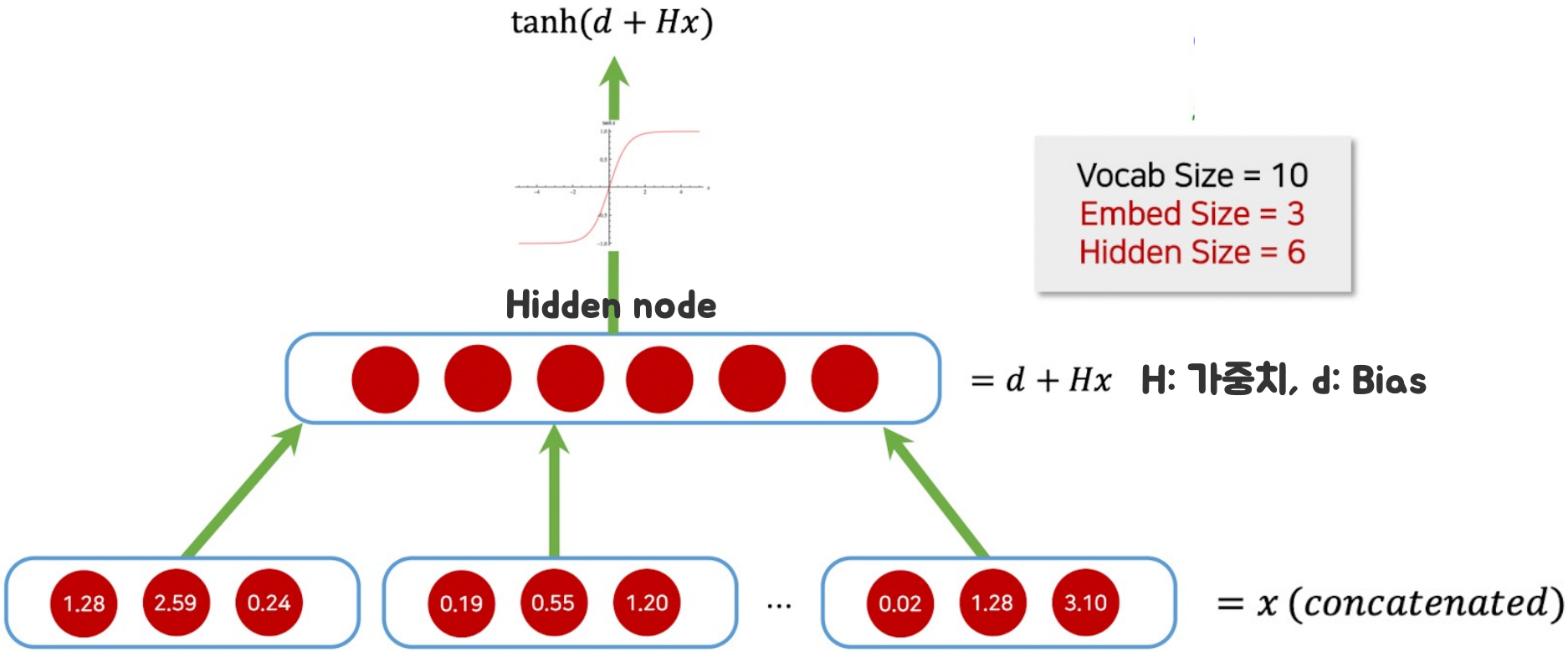
1.3 F function
: composition of (C & g) with parameters
(C,w)

$$f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$$

5. Model Architecture
Step 2. Hidden Layers & Probability Functions



5. Model Architecture
Step 2. Hidden Layers



Look-up table 을 거친 단어들은 임베드된 피처로 변환됨, Feature 을 concatenate 하여 모델에 입력됨!

5. Model Architecture
Step 2. Hidden Layers

M 차원으로 표현된 벡터를 2층의 신경을 사용하여 조건부 확률을 구성함.

1. **임베딩된 m 차원의 벡터들을 concatenate 하여 하나로 벡터로 만듬 → context 이 됨!**

$$x = \left(C(w_{t-n+1}), \cdots, C(w_{t-2}), C(w_{t-1})\right)$$

2. **Hidden Layer 에 통과시킴.** $y = U\tanh(d + Hx)$

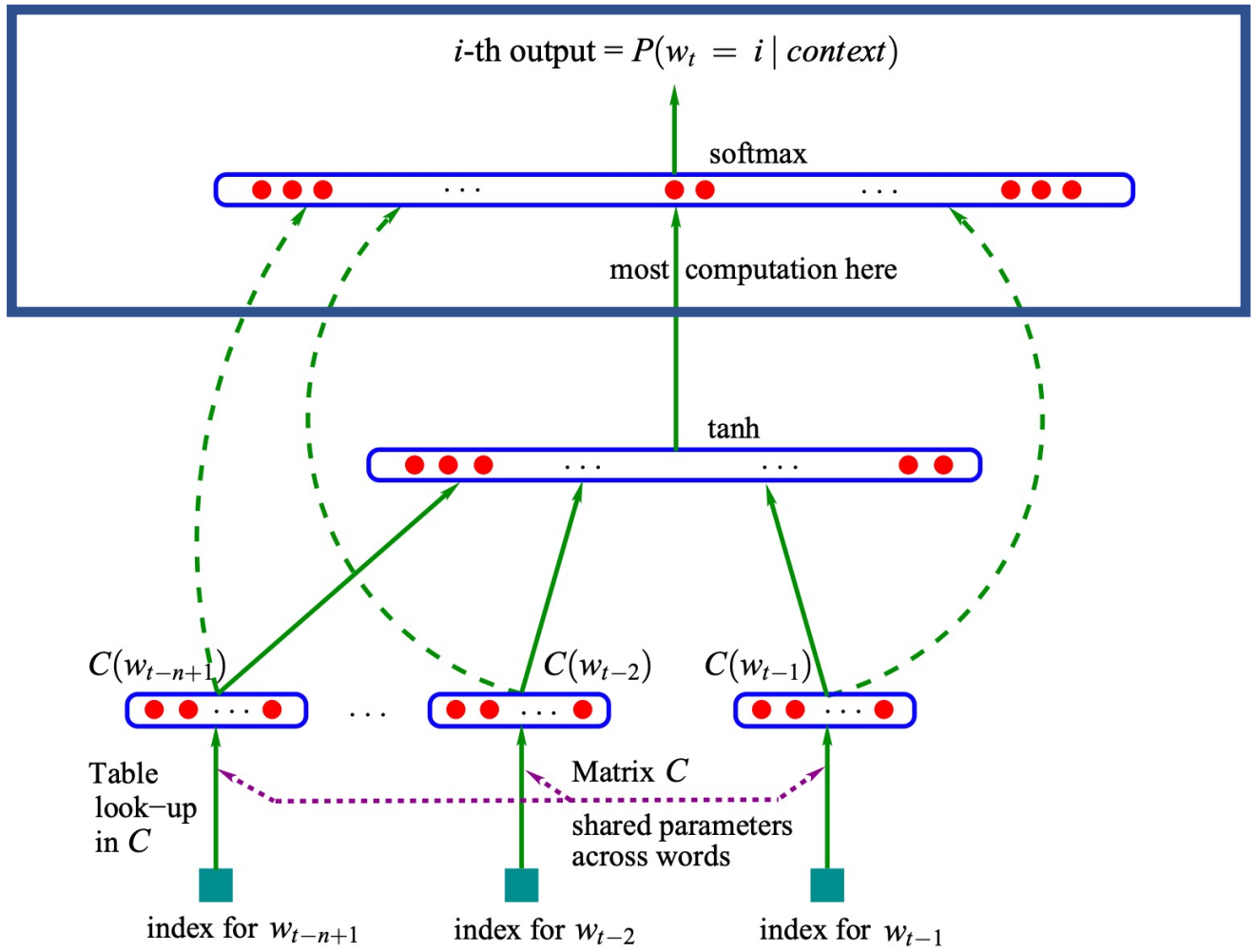
$$x = [x_{t-1}, x_{t-2}, \ldots, x_{t-n+1}]$$

$$x = [\underbrace{10, 12, 19}_{\text{없는}}, \underbrace{4, 6, 13}_{\text{말}}, \underbrace{23, 5, 7}_{\text{이}}]$$

위에서 계산된 x가 모델의 입력으로 input layer,
hidden layer, output layer를 거쳐 스코어 벡터 계산

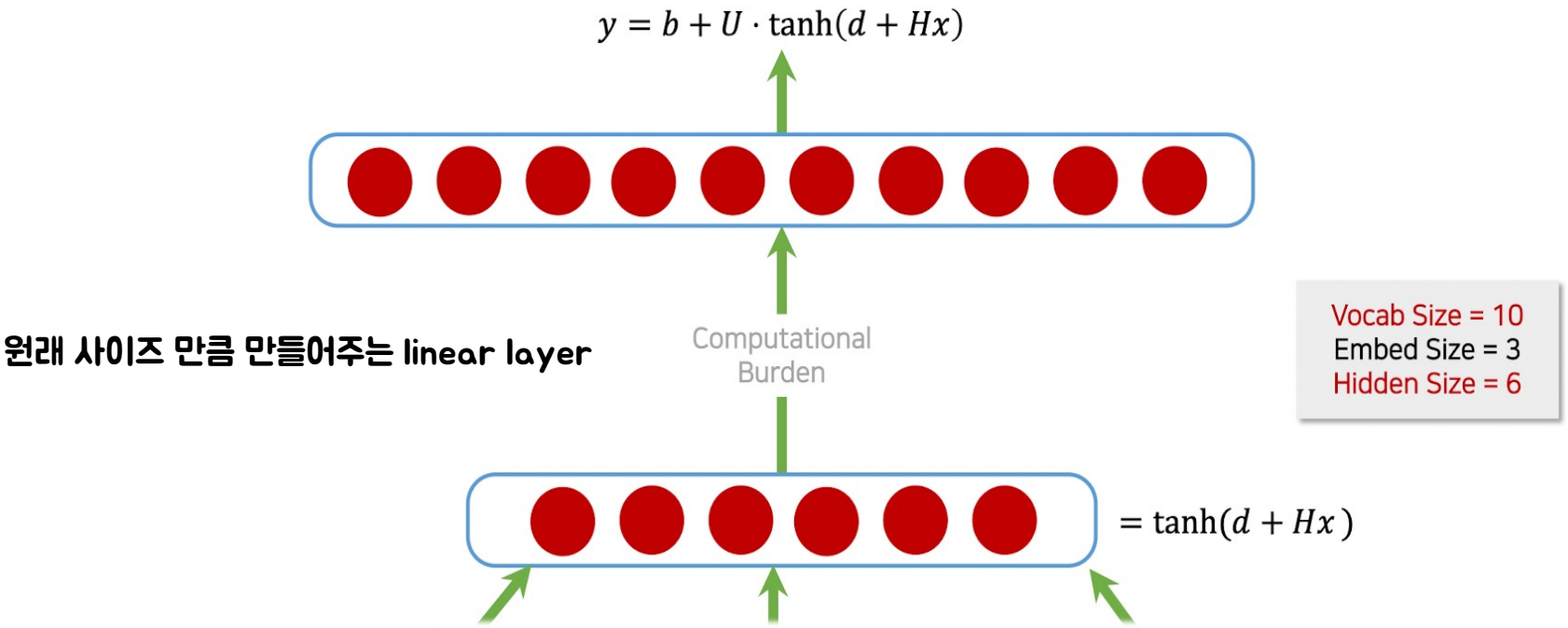
$$y_{w_t} = b + Wx + Utanh(d + Hx)$$

5. Model Architecture
Step 3. Output & Probability Functions



5. Model Architecture
Step 3. Output & Probability Functions

[logit] Softmax 함수 통과하여 확률로 변환해줌



입력 벡터는 hidden layer *H*를 통과하고 *d*를 더해줘서 *h*-차원의 벡터를 생성한다.

결과로 출력된 벡터를 *U*와 연산을 하고, *b*를 더해주면 *V*-차원의 점수 벡터 ywt 가 출력된다. 점수 벡터는 *softmax* 함수를 통해서 확률분포를 제공한다.

5. Model Architecture
Step 3. Output & Probability Functions

계산된 스코어 벡터(y)에 소프트맥스 함수를 적용하여
원래 단어의 인덱스와 비교해 loss를 계산하여 역전파

$$y_{w_t} = b + Wx + Utanh(d + Hx)$$

↓ softmax

$$P(w_t|w_{t-1}, ..., w_{t-n+1}) = \frac{exp(y_{w_t})}{\sum_i exp(y_t)}$$

The cat is **walking** in the bedroom.
A dog was running in a room.
The cat is running in a room.
A dog is **walking** in a bedroom.
The dog was **walking** in the room.

n=4 일 때,
walking을 예측하려면

The, cat, is, A, dog, was

C =

0.1 0.2 0.1
0.2 0.1 0
.
.
0.2 0.1 0.1
0.4 0.6 0.1

walking을 맞추는 과정에서
발생한 손실(train loss)을 최소화하는
그래디언트(gradient)를 받아
동일하게 업데이트
· 벡터 공간에서 같은 방향으로 움직임 !

5. Model Architecture
Parameters in the Model & etc

- Number of Free Parameters : |V| (1+nm+h)+h(1+(n-1)m)
- |V|: number of words in the corpus (계산량에 가장 많은 영향을 끼침)
- n: size of context (2,3, bigger than this can be less effective!)
- m: size of the vector (for example, it s 4 in the 'ellipse-rectangle' example)
- H: hidden layer 의 수

$$H \in R^{h\times(n-1)m}, \quad x_t \in R^{(n-1)\times m}, \quad d \in R^{h\times 1}$$
$$U \in R^{|V|\times h}, \quad b \in R^{|V|}, \quad y \in R^{|V|} \quad , C \in R^{m\times|V|}$$

Parallel Implementation
- 논문이 나온 2003년...General Purpose GPU 가 없었고, 컴퓨터 성능이 낮아서 여러 대 컴퓨터 연결해서 모델 돌림.

Time
- 시간을 재기도 하였는데, 그것은 파라미터가 많기 때문

  θ=(b,d,W,U,H,C)

$$b = |V|$$
$$d = h$$
$$U = |V| \times h$$
$$W = |V| \times (n-1)m$$
$$H = h \times (n-1)m$$
$$C = |V| \times m$$

5. Model Architecture
모델 성능 평가 Perplexity

논문은 total measurement 로 perplexity 를 선택함.

Perplexity?: A measurement of how well a probability distribution or probability model (q) predicts a sample.
Perplexity : 언어 모델이 주어진 문장을 헷갈리는 정도

문장이 헷갈린다? → 문장에 대한 확률이 낮다 → Perplexity 가 높다 → 언어 모델의 성능이 낮다.

$$PP = \exp(-\frac{1}{N}\sum_{i=1}^{N}\log_e q(x_i))$$

모든 테스트 세트에서 확률 모델 q의 불확실 정도가 어떻게 되는지를 측정함.
즉, 이 값이 높을수록 모델이 예측을 잘 못하며, 낮을수록 해당 테스트 토큰을 확실하게 측정한다는 것!

$$Perplexity(W) = P(w_1, w_2, \dots, w_n)^{-\frac{1}{n}}$$

$$= \sqrt[n]{\frac{1}{P(w_1, w_2, \dots, w_n)}}$$

$$= \sqrt[n]{\frac{1}{\prod_{i=1}^{n} P(w_i|w_1, \dots, w_{i-1})}}$$

⇒ N-gram에 따라 변동

6. Results

- Rare words (<= 3 into single symbol) : <UNK>
- Datasets: Brown Corpus, AP News (95-96)

영어 단어 Brown corpus
(약 16000개의 단어)에 대해서, 은닉층 유닛수를 100, 임베딩 차원을 30, direct connection이 없고, 5-Gram 을 사용했을 때 결과는 Perplexity 가 제일 낮았다.

	n	c	h	m	direct	mix	train.	valid.	test.
MLP1	5		50	60	yes	no	182	284	268
MLP2	5		50	60	yes	yes		275	257
MLP3	5		0	60	yes	no	201	327	310
MLP4	5		0	60	yes	yes		286	272
MLP5	5		50	30	yes	no	209	296	279
MLP6	5		50	30	yes	yes		273	259
MLP7	3		50	30	yes	no	210	309	293
MLP8	3		50	30	yes	yes		284	270
MLP9	5		100	30	no	no	175	280	276
MLP10	5		100	30	no	yes		265	**252**
Del. Int.	3						31	352	336
Kneser-Ney back-off	3							334	323
Kneser-Ney back-off	4							332	321
Kneser-Ney back-off	5							332	321
class-based back-off	3	150						348	334
class-based back-off	3	200						354	340
class-based back-off	3	500						326	**312**
class-based back-off	3	1000						335	319
class-based back-off	3	2000						343	326
class-based back-off	4	500						327	312
class-based back-off	5	500						327	312

Table 1: Comparative results on the Brown corpus. The deleted interpolation trigram has a test perplexity that is 33% above that of the neural network with the lowest validation perplexity. The difference is 24% in the case of the best n-gram (a class-based model with 500 word classes). *n* : order of the model. *c* : number of word classes in class-based n-grams. *h* : number of hidden units. *m* : number of word features for MLPs, number of classes for class-based n-grams. *direct*: whether there are direct connections from word features to outputs. *mix*: whether the output probabilities of the neural network are mixed with the output of the trigram (with a weight of 0.5 on each). The last three columns give perplexity on the training, validation and test sets.

	n	h	m	direct	mix	train.	valid.	test.
MLP10	6	60	100	yes	yes		104	**109**
Del. Int.	3						126	132
Back-off KN	3						121	127
Back-off KN	4						113	119
Back-off KN	5						112	**117**

Table 2: Comparative results on the AP News corpus. See the previous table for the column labels.