단어 임베딩 활용

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

06

APPLICATION

ARTIFICIAL INTELLIGENCE

Notice

THINKING

생각해보기

✓ 단어 임베딩은 어떻게 활용할까요?

학습목표

GOALS

1 단어 임베딩이 사용되는 이유와 그 의미에 대해 설명할 수 있다.

2 단어 임베딩의 종류와 단어 임베딩을 통해 할 수 있는 일들에 대해 설명할 수 있다.

3 단어 임베딩의 유사도 평가 방식과 그 종류를 설명할 수 있다.

4 단어 임베딩의 유추 평가 방식과 시각화 방식을 설명할 수 있다.

5 단어 임베딩을 문장 수준 임베딩으로 확장하는 방법인 가중 임베딩에 대해 설명할 수 있다.

6 문장 단위로 임베딩이 이루어지는 문장 임베딩 방식에 대해 설명할 수 있다.

1 단어 임베딩의 의미와 역할

2 단어 유사도, 유추 평가

3 단어 임베딩 시각화

4 가중 임베딩, 문장 임베딩

5 실습

CONTENTS

학습내용

APPLICATION

01 단어 임베딩의 의미와 역할

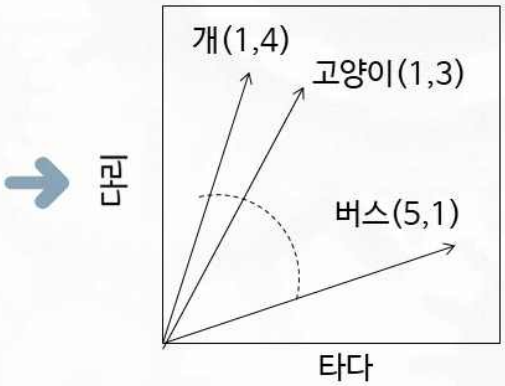
단어 임베딩

사람이 쓰는 자연어를 기계가 이해할 수 있는 숫자의 나열인 벡터로 바꾼 결과 혹은 그 일련의 과정 전체를 의미

01 단어 임베딩의 의미

> **임베딩에 자연어의 통계적 패턴 정보를 주면 자연어의 의미를 함축할 수 있다.**

	타다	다리
개	1	4
고양이	1	3
버스	5	1



02 단어 임베딩에 쓰이는 3가지 철학

BOW(bag of words) 가정 : 어떤 단어가 (많이) 쓰였는가

- TF-IDF, Deep averaging network

언어 모델 : 단어가 어떤 순서로 쓰였는가

- ELMo, GPT

분포 가정 : 어떤 단어가 같이 쓰였는가

- PMI, Word2Vec

03 단어 임베딩의 역할

1. 단어, 문장 간 관련도 계산
2. 의미적, 문법적 정보 함축
3. 전이 학습

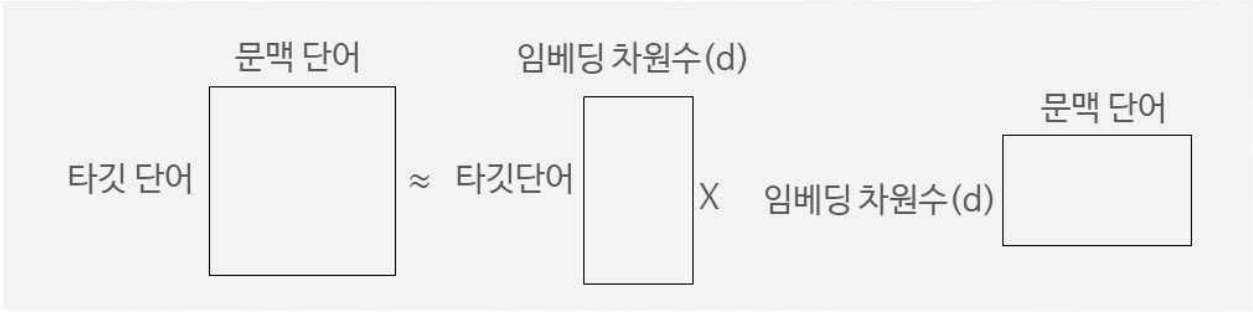
04 단어 임베딩의 종류

- 행렬 분해 기반 방법
- 예측 기반 방법
- 토픽 기반 방법

말뭉치 정보가 들어있는 원래 행렬을

두 개 이상의 작은 행렬로 쪼개는 방식의 임베딩 기법

- 분해 이후 둘 중 하나의 행렬만 쓰거나 둘을 더하거나 이어 붙여 임베딩으로 사용
- LSA, Glove, Swivel



PMI(점별 상호 정보량)

두 확률변수 사이의 상관성을 계량화하는 단위

- 두 확률변수가 완전히 독립인 경우 그 값이 0이 됨
  예를 들어 단어 A, B가 있을 때 단어 A가 나타나는 것이 단어 B의 등장할 확률에 전혀 영향을 주지 않고, 단어 B가 나타나는 것이 단어 A에 영향을 주지 않는 경우를 독립이라 함

- 두 단어의 등장이 독립일 때에 대비해 얼마나 자주 같이 등장하는지를 수치화 한 것

06 PMI

$$PMI(A,B) = log\frac{P(A,B)}{P(A)X\ P(B)}$$

PMI 행렬은 단어 - 문맥 행렬에 위 수식을 적용한 결과

카페,에서,뜨거운,커피,를,마시는,당신

	카페	에서	뜨거운	커피	를	마시는	당신	total
카페 ... 커피 ...		+1	+1		+1	+1		20
total			15					1000

$$PMI(\text{커피},\text{뜨거운}) = log\frac{P(\text{커피},\text{뜨거운})}{P(\text{커피})X\ P(\text{뜨거운})} = log\frac{\frac{10}{1000}}{\frac{20}{1000}X\ \frac{15}{1000}}$$

07 LSA

LSA

커다란 행렬에 차원 축소 방법의 일종인 특이값 분해를 수행해 데이터의 차원 수를 줄여 계산 효율성을 키우는 한편 행간에 숨어있는 잠재 의미를 이끌어내기 위한 방법론

- 단어-문서 행렬이나 단어-문맥 행렬 등에 특이값 분해를 시행한 뒤 그 결과로 도출되는 행 벡터들을 단어 임베딩으로 사용할 수 있음

PPMI 행렬

PMI는 두 단어의 등장이 독립이라 가정할 때 대비 얼마나 같이 자주 등장하는지를 수치화한 것

- 그러나 수식 상 A, B 두 단어가 동시에 등장할 확률이 두 단어가 독립일 때보다 작으면 PMI는 음수가 됨

08 LSA에 사용되는 PPMI 행렬

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

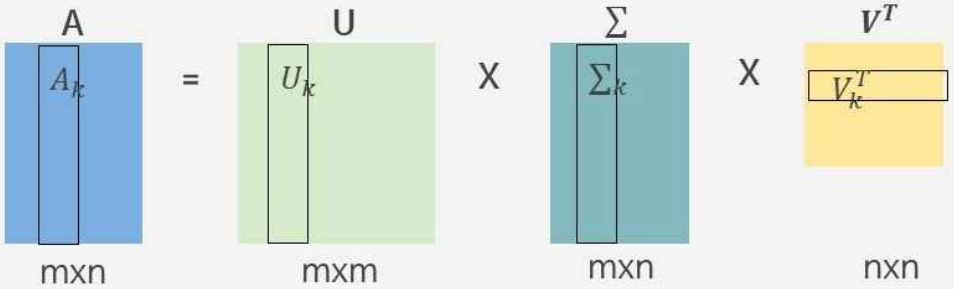
PPMI 행렬

PMI는 두 단어의 등장이 독립이라 가정할 때 대비 얼마나 같이 자주 등장하는지를 수치화한 것

- 그러나 이러한 결과는 신뢰하기 힘든데, 말뭉치가 충분히 크지 않는 한 두 단어가 동시에 나타날 확률이 두 단어가 독립일 때보다 작기가 힘들기 때문임

$$PPMI(A,B) = max(PMI(A,B), 0)$$

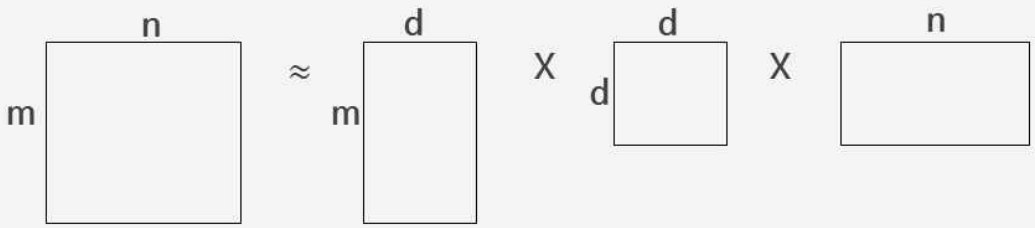
특이값 분해(SVD)는 mxn 크기의 임의의 사각행렬 A를 아래와 같이 분해하는 것을 가리킴



09 LSA의 행렬 분해

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

Truncated SVD는 특이값 가운데 가장 큰 d개만 가지고, 해당 특이값에 대응하는 특이벡터들로 원래 행렬 A를 근사하는 기법



- **Word2Vec과 잠재의미분석(LSA) 두 기법의 단점을 극복하고자 한 기법**

- **임베딩된 두 단어 벡터의 내적이 말뭉치 전체에서의 동시 등장 빈도의 로그 값이 되도록 목적함수를 정의**

- $$\mathcal{J} = \sum_{i,j=1}^{|V|} f(A_{ij})(U_i \cdot V_j + b_i + b_j - logA_{ij})^2$$

- **임베딩된 단어 벡터 간 유사도 측정을 수월하게 하면서도 말뭉치 전체의 통계정보를 좀더 잘 반영하고자 한 것이 GloVe의 핵심목표**

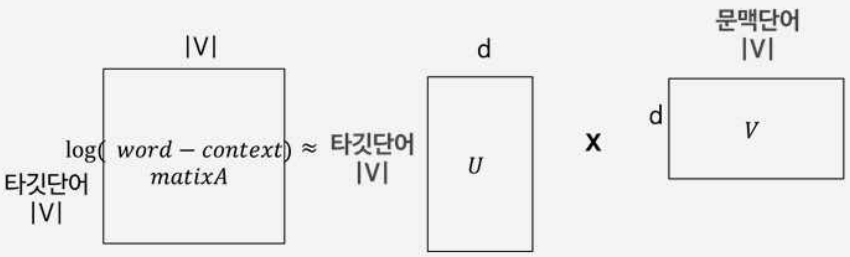
10 GloVe

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

GloVe는

우선 학습 말뭉치를 대상으로 단어-문맥행렬 A를 만드는 것에서부터 학습을 시작

이후 목적함수를 최소화하는 임베딩 벡터를 찾기 위해 행렬 분해를 수행



- 처음에 행렬 U, V를 랜덤으로 초기화한 뒤 목적함수를 최소화하는 방향으로 U,V를 조금씩 업데이트해 나가고, 학습 손실이 더 줄지 않거나 정해진 스텝 수만큼 학습했을 경우 학습을 종료. 이때 생긴 U를 단어 임베딩으로 쓸 수 있음

토픽 기반 방법

주어진 문서에 잠재된 주제를 추론하는 방식으로 임베딩을 수행하는 기법

- 각 문서가 어떤 주제 분포를 갖는지 확률 분포 형태로 반환하기 때문에 임베딩 기법의 일종으로 이해할 수 있음
- LDA

예측 기반 방법

어떤 단어 주변에 특정 단어가 나타날지 예측하거나,
이전 단어들이 주어졌을 때 다음 단어가 무엇일지 예측하거나,
문장 내 일부 단어를 지우고 해당 단어가 무엇일지 맞추는 과정에서 학습하는 방법

- Word2Vec, FastText, BERT, ELMo, GPT

Word2vec의 단점

- 학습 때 보지 못했던 단어에 대한 word vector를 분간하지 못하며, 자주 쓰이지 않는 단어들의 word vector 또한 학습을 잘 하지 못함

➡ 이러한 단점을 보완하기 위해 FastText가 제안됨

Subword embedding을 이용한 word embedding 방법으로,
등록되지 않은 단어와 자주 나오지 않는 단어에 대해 자주 쓰이는
단어와의 형태적 유사성을 고려한 word vector를 추정

각 단어를 문자 단위 n-gram으로 표현

- Ex) 해바라기 : 〈해바, 해바라, 바라기, 라기〉 , 〈해바라기〉
- 〈 , 〉 는 단어의 경계를 나타내기 위해 FastText가 사용하는 특수 기호
- 단어를 문자 단위 n-gram 벡터의 합으로 나타냄 $u_t = \sum_{g \in G_t} z_g$

$$u_{\text{해바라기}} = +z_{<\text{해바}} + z_{\text{해바라}} + z_{\text{바라기}} + z_{\text{라기}>} + z_{<\text{해바라기}>}$$

아래 수식에 정의된 조건부 확률을 최대화하는 과정에서 학습

$$P(+ \; or \; -|t.c) = \frac{1}{1+\exp(-u_t v_c)} = \frac{1}{1+\exp(-\sum_{g \in G_t} z_g^{\top} v_c}$$

입력 단어 쌍(t,c)가 실제로 포지티브/네거티브 샘플이라면 모델이 해당 입력 쌍이 포지티브/네거티브라고 맞추도록 학습하는 방식

Word2vec과 다르게 타깃단어(t), 문맥 단어(c) 쌍을 학습할 때 타깃 단어(t) 속한 문자 단위 n-gram 벡터(z)들을 모두 업데이트

15 네거티브 샘플링

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

1 실제 포지티브 샘플을 모델이 포지티브 샘플링이라 맞출 때

- 포티지브 샘플이 주어졌을 때 수식을 최대화 하려면 분모를 최소화해야 함
  - 분모를 최소화하는 것 : v_c와 z들 간 내적 값 높이는 것
  - 벡터의 내적은 코사인 유사도와 비례하기 때문에 결국 문자간 n-gram 벡터와 문맥 단어의 포지티브 샘플(c)에 해당하는 단어 벡터 간 유사도를 높여야 한다는 의미

2 실제 네거티브 샘플을 모델이 네거티브 샘플링이라 맞출 때

- 네거티브 샘플이 주어졌을 때 수식을 최대화 하려면 분자를 최대화해야 함
  - 분모를 최소화하는 것 : v_c와 z들 간 내적 값 낮추는 것
  - 벡터의 내적은 코사인 유사도와 비례하기 때문에 결국 문자간 n-gram 벡터와 문맥 단어의 포지티브 샘플(c)에 해당하는 단어 벡터 간 유사도를 낮춰야 한다는 의미

15 네거티브 샘플링

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

FastText 모델이 최대화해야 할 로그 우도 함수

$$\mathcal{L}(\theta) = logP(+|t_p, c_p) + \sum_{i=1}^{k} logP(-|(t_{n_i}, c_{n_i})$$

모델을 한번 업데이트 할 때 1개의 포지티브샘플(t_p,c_p)와 k개의 네거티브 샘플(t_{ni}, c_{ni}) 을 학습한다는 의미

APPLICATION

02 단어 유사도, 유추 평가

01 단어 임베딩의 평가

> 자연어 단어 간 통사적, 의미론적 관계가
> 얼마나 잘 녹아 있는지 정량적으로 평가

- 유사도 평가
- 유추 평가

유사도 평가란?

- 자연어와 자연어 사이의 의미적 유사성을 점수화하는 일련의 과정

활용분야

정보검색 | 문서분류 | 의미의 모호성 해결

문서의 요약 | 번역 성능의 자동평가 | 문서의 일관성 측정

02 유사도 평가

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

기대효과

1. 자연어의 비교 및 평가를 위한 기반 마련
2. 단어 관계로 구성된 네트워크 데이터(온톨로지)를 활용하는 유사도 연구의 초석

> 유사도를 측정하는 데 있어 다양한 방식이 사용됨

유사도 종류

- 코사인 유사도
- 유클리디안 유사도
- 자카드 유사도
- 맨하탄 유사도

03 유사도 종류

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

코사인 유사도

- 일반적으로 성능이 좋기 때문에 유사도 측정에 가장 대표적으로 사용되는 유사도
- 두개의 벡터값에서 코사인 각도를 구하는 방법임
- 단순히 좌표상의 거리를 구하는 다른 유사도 측정과 달리 두 벡터의 각도를 구하기 때문에 방향성 또한 측정 가능함
- 두 문장이 유사하면 같은 방향을 가리키고, 그렇지 않으면 직교로 표현함

$$\text{Similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

자카드 유사도

- 두 문장을 각각 단어의 집합으로 만든 뒤 두 집합을 통해 유사도를 측정함
- 두 집합의 교집합인 공통된 단어의 개수를 두 집합의 합집합인 전체 단어의 갯수로 나눔
- 공통의 원소의 개수에 따라 0~1 사이의 값이 나오며, 1에 가까울수록 유사도가 높음

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

03 유사도 종류

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

유클리디안 유사도

- 가장 기본적인 거리를 측정하는 방식
- L2 거리라고도 하며, N차원 공간에서 두 점 사이의 최단 거리를 구하는 접근법임
- 단순히 두 점 사이의 거리를 뜻하기 때문에 결과값에 제한이 없음
- 따라서 다른 유사도와 비교하려면 0과 1사이의 값으로 정규화가 필요함
- L1 정규화 방법 : 벡터의 모든 값을 더한 후 이 값으로 각 벡터의 값을 나눔

$$\sqrt{(p_1 - q_1)2 + (p_2 - q_2)^2 + \cdots + (pn - qn)^2} = \sqrt{\sum_{i=1}^{n} (p_i - qi)^2}$$

맨하탄 유사도

- **사각형 격자로 이뤄진 지도에서 출발점에서 도착점까지를 가로지르지 않고 갈 수 있는 최단 거리를 구하는 공식임**
- 맨하탄 유사도 역시 유클리디언 유사도와 마찬가지로 결과값에 제한이 없음
  → 따라서 다른 유사도와 비교하기 위해서는 정규화가 필요함

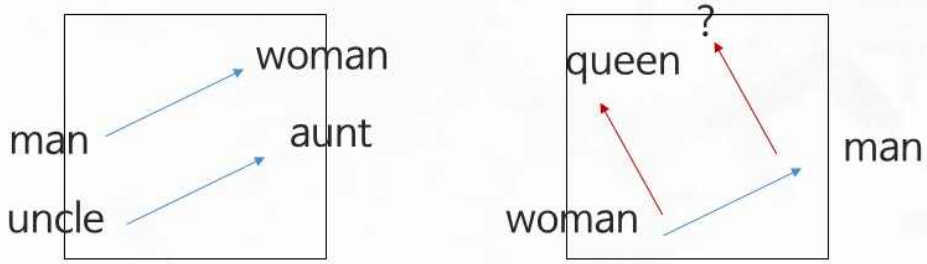
$$\sum_{i=1}^{k} | xi - yi |$$

유추 평가

유추 평가에서는 단어 임베딩이 단어 사이의 의미 관계를 얼마나 잘 학습했는지 평가

“갑과 을의 관계는 병과 정의 관계와 같다”

- “ ' 갑' – ' 을' + ‘정’ = ? ”이라는 질의에 대해 ‘병’을 도출해 낼 수 있는지를 평가



APPLICATION

03 단어 임베딩 시각화와 가중/문장 임베딩

단어 임베딩 시각화

의미가 유사한 단어를 사람이 쉽게 이해할 수 있는 형태의 그림으로 표현해 임베딩의 품질을 정성적, 간접적으로 확인하는 기법

> **단어 임베딩은 보통 고차원 벡터이기에 사람이 인식하는 2, 3차원으로 축소해 시각화를 하게 됨**

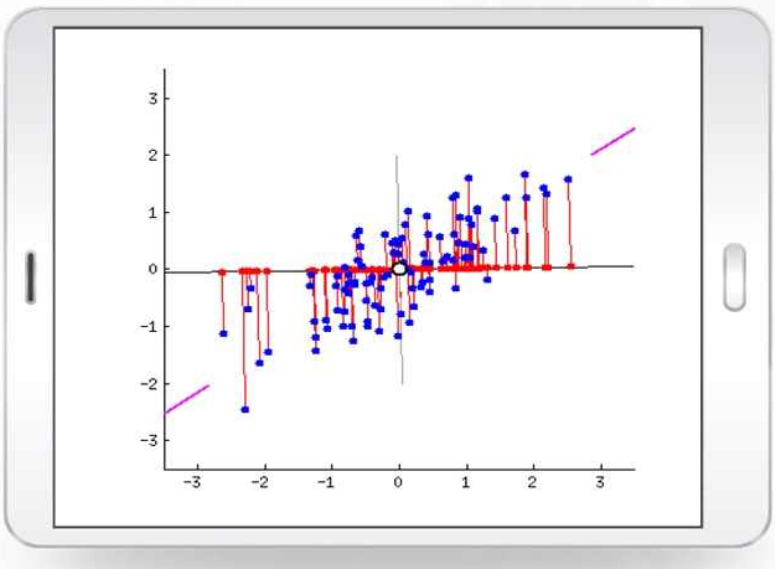
PCA t-SNE 히트맵 이용

01 단어 임베딩 시각화

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

PCA

- 데이터의 분산을 최대한 보존하면서 서로 직교하는 새 기저(축)을 찾아 고차원 공간의 표본들을 선형 연관성이 없는 저차원 공간으로 변환하는 기법



출처: https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

PCA

- 특성들이 통계적으로 상관관계가 없도록 데이터셋을 회전시키는 기술임

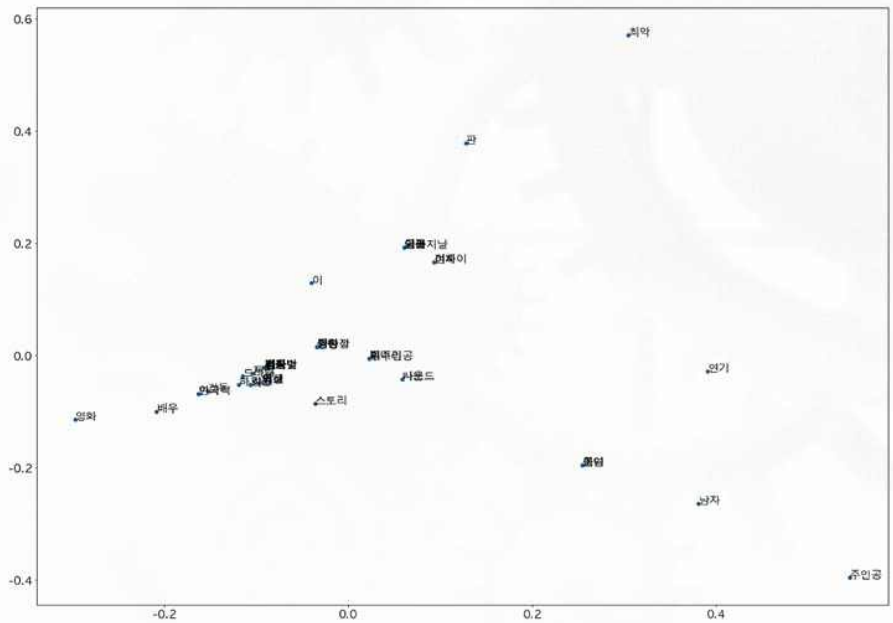
ⓥ 특성들의 상관관계가 가장 큰 방향(분산이 가장 큰 방향)을 찾고 그 방향과 직각인 방향 중에서 가장 많은 정보를 담은 방향을 찾음

ⓥ 선형변환을 이용하기 때문에 비선형 특성을 가진 데이터에 대해서는 데이터의 특성을 잘 추출하지 못하는 한계가 있음

01 단어 임베딩 시각화

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

PCA를 이용한 시각화



출처: 퍼블릭에이아이(www.publicai.co.kr)

t-SNE

t-SNE

고차원의 원 공간에 존재하는 벡터 x에 이웃 간의 거리를 최대한 보존하는
저차원 벡터 y를 학습함으로써, 고차원의 데이터를 2차원의 지도로 표현

- 다른 알고리즘들보다 안정적인 임베딩 학습 결과를 보임
- t-SNE가 데이터 간 거리를 stochastic probability로 변환하여 임베딩에 이용하기 때문, 이 값은 perplexity에 의해 조정됨

01 단어 임베딩 시각화

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

t-SNE 알고리즘

- p는 고차원 원공간에 존재하는 i 번째 개체 x_i가 주어졌을 때 j번째 이웃인 x_j 가 선택될 확률

$$p_{j|i} = \frac{e^{\frac{|x_i - x_j|^2}{2\sigma_i^2}}}{\sum_k e^{-\frac{|x_i - x_k|^2}{2\sigma_i^2}}}$$

- q는 저차원에 임베딩된 i번째 개체 y_i가 주어졌을 때 j번째 이웃인 y_j가 선택될 확률

$$q_{j|i} = \frac{e^{-|y_i - y_j|^2}}{\sum_k e^{-|y_i - y_k|^2}}$$

t-SNE 알고리즘

- t-SNE는 p와 q의 분포차이를 최대한 작게 해서 차원축소가 잘 이루어지도록 함
- 두 확률 분포가 얼마나 비슷한지 확인하기 위해 KL-divergence 지표를 사용, 두 분포가 완전히 다르면 1, 동일하면 0의 값을 가짐
- t-SNE는 아래 비용함수를 최소화 하는 방향으로 학습을 진행함

$$Cost = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

01 단어 임베딩 시각화

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

t-SNE 알고리즘

1. t-SNE는 계산 속도를 높이기 위해 p계산시 쓰이는 σ_i 계산을 생략
2. i번째 개체가 주어졌을 때 j번째 개체가 이웃으로 뽑힐 확률과 j번째 개체가 주어졌을 때 i번째 개체가 선택될 확률을 동일하다고 가정

$$p_{j|i} = \frac{p_{j|i} + p_{i|j}}{2}$$

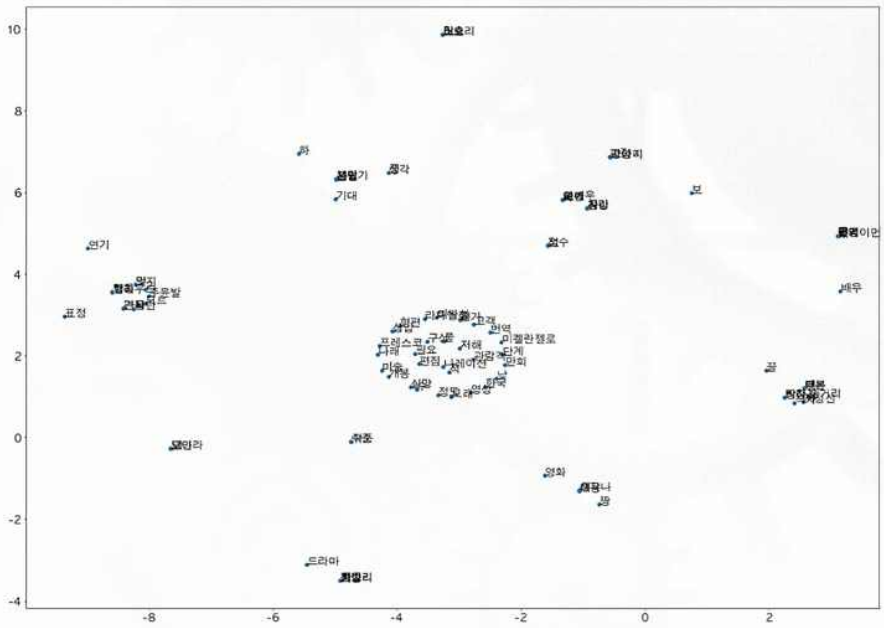
$$q_{j|i} = \frac{q_{j|i} + q_{i|j}}{2}$$

$$Cost = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$$

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (y_j - y_i)(p_{ij} - q_{ij})$$

01 단어 임베딩 시각화

t-SNE를 이용한 시각화



출처: 퍼블릭에이아이(www.publicai.co.kr)

02 단어 임베딩의 한계

단어 임베딩 방식은

벡터에 해당 단어의 문맥적 의미를 함축은 하지만 동음이의어를 분간하기 어려움

> **단어의 형태가 같을 때 동일한 단어로 보고,**
> **모든 문맥 정보를 해당 단어 벡터에 투영하기 때문**

가중 임베딩

- 단어 임베딩을 문장 수준 임베딩으로 확장하는 방법
- 단어의 등장은 저자가 생각한 주제에 의존한다고 가정,
  즉 주제에 따라 단어의 사용 양상이 달라질 것으로 예상

- 따라서 단어의 등장 확률을 주제벡터가 주어졌을 때
  해당 단어가 나타날 확률로 정의

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

가중 임베딩

- 문장 등장 확률을 문장에 속한 모든 단어들이 등장할 확률의
  가중누적 곱(로그를 취하여 덧셈)으로 나타냄
  ex) CBoWModel

단어가 아닌 문장 단위로 내용을 분류, 비교

동음이의어 처리가 어려웠던 단어 임베딩의 한계 극복

문장들을 각 문장을 표현하는 고정된 길이의 벡터로 변환 시 벡터 간 비교로 문장을 비교 가능

- 문장벡터를 얻는 가장 간단한 방법은 문장에 존재하는 단어 벡터들의 평균을 구하는 것
- Doc2Vec, LDA, ELMo, GPT, BERT

04 문장 임베딩

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

행렬 분해 모델

- LSA

확률 모형

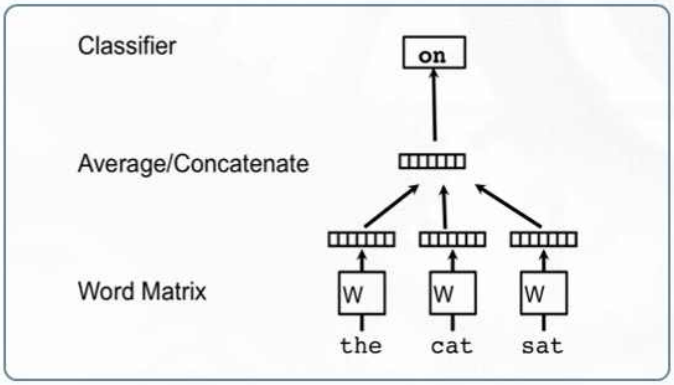
- LDA

뉴럴 네트워크 기반 모델

- Doc2Vec, ELMo, GPT, BERT

언어모델

- 우선 이전 단어 시퀀스 k개가 주어졌을 때 그 다음 단어를 맞추는 언어모델을 만듦
- 예시 문장 : The cat sat on the mat



출처: Quoc V. Le, Tomas Mikolov,Distributed Representations of Sentences and Documents (ICML, 2014)

- 이전 단어가 주어진 상태에서 다음 단어를 맞추는 것을 상정, 이 모델은 문장 전체를 처음부터 끝까지 이같이 한 단어씩 슬라이딩해 가면서 다음 단어가 무엇일지 예측함

05 Doc2Vec

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

기본 수식

- $\mathcal{L} = \frac{1}{T}\sum_{t=k}^{T-k} logp(w_t|w_{t-k}\dots,w_{t+k})$

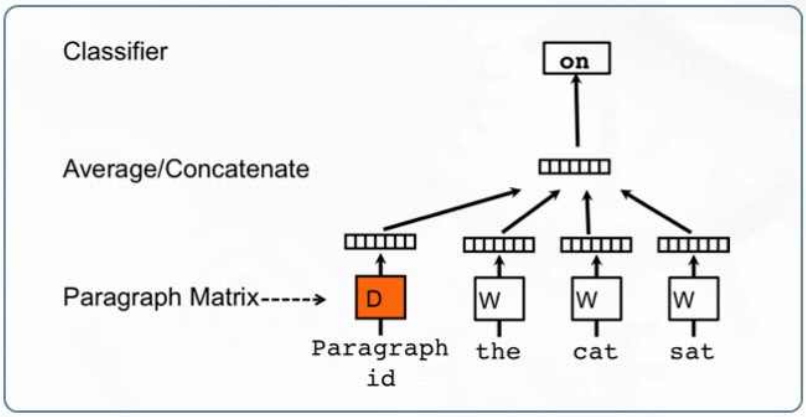
- 이 값이 커질수록 모델에 이전 k개 단어를 입력했을 때 모델이 다음 단어를 잘 맞추게 됨

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

- $y = b + U \cdot h(w_{t-k}\dots,w_{t-1};W)$

PV-DM

- PV-DM : 아까와 같은 모델에 문서 ID를 추가해 아래와 같은 구조를 만들었는데, 즉 이전 k개 단어들과 문서 ID를 넣어서 다음 단어를 예측한다는 뜻
- 이전 모델과 다른 점은 y를 계산할 때 D라는 문서 행렬에서 해당 문서 ID에 해당하는 벡터를 참조해 h함수에 다른 단어 벡터들과 함께 입력하는 것이고 이외의 과정은 동일함



출처: Quoc V. Le, Tomas Mikolov,Distributed Representations of Sentences and Documents (ICML, 2014)

05 Doc2Vec

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

PV-DM

- PV-DM 방식으로 만들어진 문서 임베딩은 해당 문서의 주제정보를 함축한다고 함

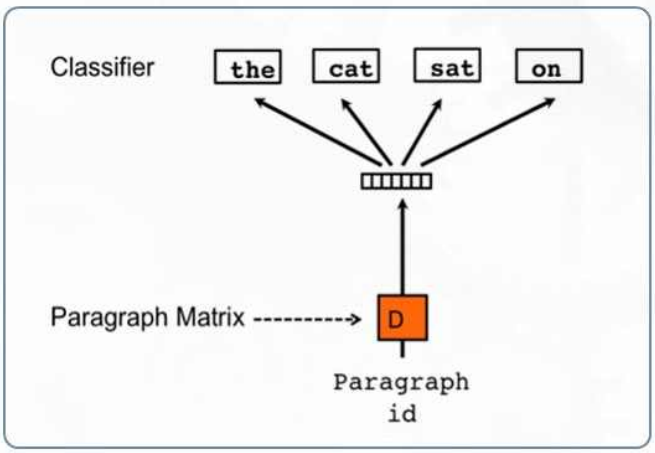
Why?

문서 임베딩은 동일한 문서 내 존재하는 모든 단어와 함께 학습될 기회를 갖기 때문

- PV-DM은 단어 등장 순서를 고려하는 방식으로 학습하기 때문에 순서 정보를 무시하는 백오브워즈 기법 대비 강점이 있음

PV-DBOW

- PV-DBOW 모델은 Word2Vec의 Skip-gram 모델을 본떠 만들었는데 Skip-gram 모델이 타깃 단어를 가지고 문맥 단어를 예측하는 과정에서 학습하는 것처럼 문서 ID를 가지고 문맥 단어를 맞추는 모델



출처: Quoc V. Le, Tomas Mikolov, Distributed Representations of Sentences and Documents (ICML, 2014)

잠재의미 분석은 단어-문서 행렬이나 TF-IDF 행렬, 단어-문맥 행렬 또는 PMI 행렬에 특이값 분해로 차원 축소를 시행하고, 여기에서 단어에 해당하는 벡터를 취해 임베딩을 만드는 방법

단어-문서 행렬이나 TF-IDF 행렬에 SVD를 시행, 축소된 이 행렬에서 문서에 대응하는 벡터를 취해 문서 임베딩을 만드는 방식

LDA란?

잠재 디리클레 할당(LDA)

주어진 문서에 대하여 각 문서에 어떤 토픽이 존재하는지에 대한 확률 모형

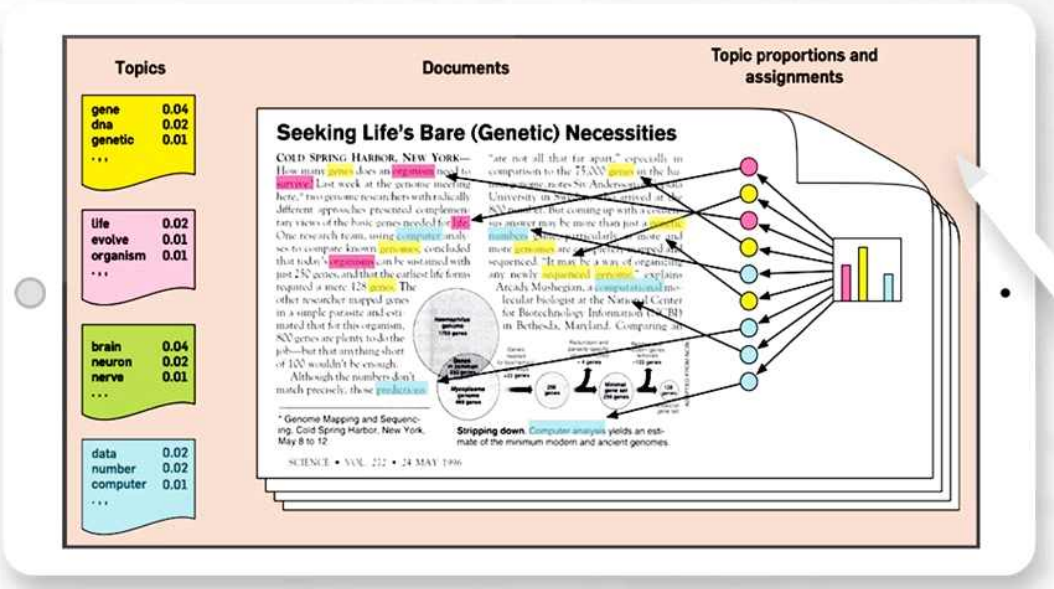
- 말뭉치 이면에 잠재된 토픽을 추출한다는 의미에서 토픽 모델링이라고도 불림
- 문서를 토픽 확률 분포로 나타내 각각을 벡터화한다는 점에서 LDA를 임베딩 기법의 일종으로 이해할 수도 있음

07 LDA

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

LDA 개요

- 토픽별 단어의 분포, 문서별 토픽의 분포를 모두 추정하는 모델



출처: blei, Introduction to Probabilistic Topic Models (2011)

LDA의 가정

- LDA는 LDA가 가정하는 문서 생성과정에 따른 아키텍처를 갖고 있음

글감/주제 선정 > 주제 내 단어 선정

LDA – 이 과정의 역방향인, 현재 문서에 등장한 단어가 어떤 토픽에서 뽑혔는지 추론하는 과정

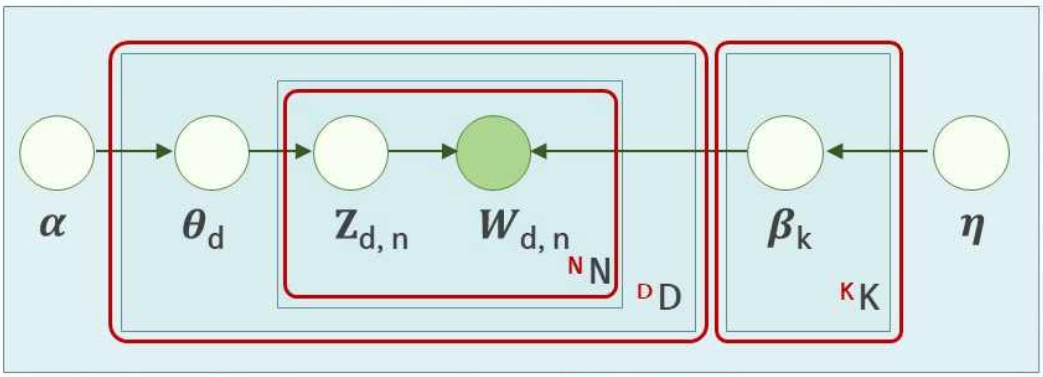
– 명시적으로 라벨링이 되어있지 않은 말뭉치에 등장하는 단어 이면에 존재하는 정보(토픽)를 추론하는 것

07 LDA

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

LDA의 구조

- D : 말뭉치 전체 문서 개수
- K : 전체 토픽수(하이퍼파라미터)
- N : d번째 문서의 단어 수
- 네모칸은 해당 횟수만큼 반복하라는 의미, 동그라미는 변수

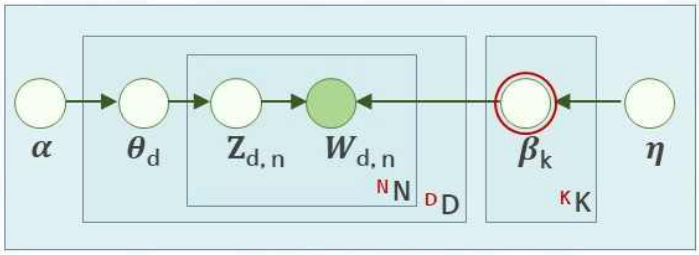


출처: blei, Introduction to Probabilistic Topic Models(2011)

LDA의 구조

- β_k : k번째 토픽에 해당하는 벡터

단어	토픽1	토픽2	토픽3
개	0.269	0.000	0.267
빌딩	0.115	0.000	0.133
구름	0.231	0.313	0.400
아파트	0.000	0.312	0.400
하늘	0.000	0.312	0.000
고양이	0.192	0.063	0.000
버스	0.192	0.000	0.200



출처: blei, Introduction to Probabilistic Topic Models (2011)

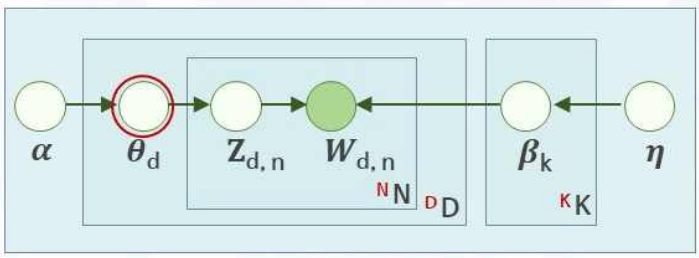
07 LDA

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

LDA의 구조

- θ_d : d번째 문서가 가진 토픽비중

단어	토픽1	토픽2	토픽3
문서1	0.400	0.000	0.600
문서2	0.000	0.600	0.400
문서3	0.375	0.625	0.000
문서4	0.000	0.375	0.625
문서5	0.500	0.000	0.500
문서6	0.500	0.500	0.000

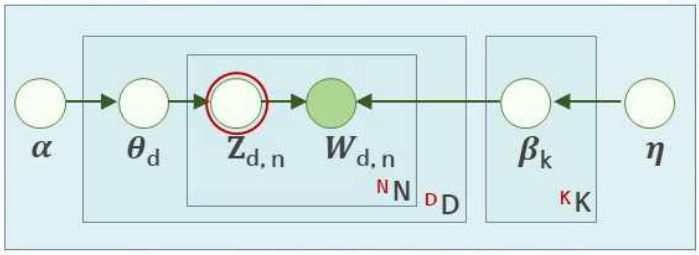


출처: blei, Introduction to Probabilistic Topic Models (2011)

LDA의 구조

- $z_{d,n}$: d번째 문서 n번째 단어가 어떤 토픽인지 나타내는 변수

단어	토픽1	토픽2	토픽3
문서1	0.400	0.000	0.600
문서2	0.000	0.600	0.400
문서3	0.375	0.625	0.000
문서4	0.000	0.375	0.625
문서5	0.500	0.000	0.500
문서6	0.500	0.500	0.000



출처: blei, Introduction to Probabilistic Topic Models (2011)

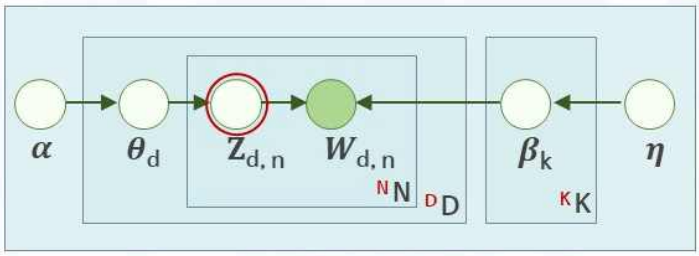
07 LDA

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

LDA의 구조

- $w_{d,n}$: d번째 문서 내에 n번째로 등장하는 단어

단어	토픽1	토픽2	토픽3
문서1	0.400	0.000	0.600
문서2	0.000	0.600	0.400
문서3	0.375	0.625	0.000
문서4	0.000	0.375	0.625
문서5	0.500	0.000	0.500
문서6	0.500	0.500	0.000



출처: blei, Introduction to Probabilistic Topic Models (2011)

LDA 수식

- LDA는 토픽의 단어 분포(β)와 문서의 토픽 분포(θ)의 결합으로 문서 내 단어들이 생성된다고 가정, 즉 토픽의 단어 분포와 문서의 토픽 분포의 결합확률이 커질수록 LDA가 가정하는 문서 생성 과정이 합리적이라는 것을 의미

- $$p(\beta_{1:K}, \theta_{1:D}, z_{1:D} w_{1:D}) = \prod_{i=1}^{K} p(\beta_i|\eta) \prod_{d=1}^{D} p(\theta_d|\alpha) \left\{ \prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n}) \right\}$$

- 색이 다른 변수 제외 모두 미지수, $p(z,\beta,\theta|w)$ 를 최대로 하는 z,β,θ 을 찾아야하며 즉 구해야 할 사후확률분포가 $p(z,\beta,\theta|w) = p(z,\beta,\theta,w) / p(w)$

깁스샘플링

나머지 변수는 고정시킨 채 하나의 랜덤변수만을 대상으로 표본을 뽑는 기법

- LDA에서는 사후확률분포 $p(z,\beta,\theta|w)$를 구할 때 토픽의 단어분포(β)와 문서의 토픽 분포(θ)를 계산에서 생략하고 토픽(z)만을 추론

- z만 알 수 있으면 나머지 변수(β,θ)를 계산할 수 있도록 모델을 설계

d번째 문서 i번째 단어($z_{d,i}$)가 실제 j번째 토픽이 될 확률을 깁스샘플링을 적용해 도출

$$p(z_{d,i} = j \,|\, z_{-i}, w) = \frac{n_{d,k} + \alpha_j}{\sum_{i=1}^{K}(n_{d,i}+\alpha_i)} X \frac{v_{k,w_{d,n}} \beta_{w_{d,n}}}{\sum_{j=1}^{V}(v_{k,j}+\beta_j)} = AB$$

변수	내용
$n_{d,k}$	k번째 토픽에 할당된 d번째 문서의 단어 빈도
$v_{k,w_{d,n}}$	전체 말뭉치에서 k번째 토픽에 할당된 단어$w_{d,n}$의 빈도
$w_{d,n}$	d번째 문서에 n번째로 등장한 단어
α	문서의 토픽 분포 생성을 위한 디리클레 분포 파라미터
β	토픽의 단어 분포 생성을 위한 디리클레 분포 파라미터
K	사용자가 지정하는 토픽 수
V	말뭉치에 등장하는 전체 단어 수
A	d번째 문서가 k번째 토픽과 맺고 있는 연관성 정도
B	d번째 문서와 n번째 단어가 k번째 토픽과 맺고 있는 연관성 정도

1 말뭉치 전체 문서 모든 단어에 토픽이 이미 할당 되었다고 가정

- LDA는 초기에 문서 전체 내 모든 단어의 주제를 랜덤하게 할당하고 학습을 시작
  - 문서의 단어별 토픽 분포 : 단어 5개로 구성된 문서1의 모든 단어에 주제 할당

z_{1j}	3	2	1	3	1
$w_{1,n}$	고양이	구름	나무	별	겨울

↓

단어	토픽1	토픽2	토픽3
고양이	1	0	35
구름	50	0	1
…	…	…	…

2 깁스 샘플링으로 잠재된 토픽이 무엇인지 추론

- 깁스 샘플링으로 문서1 두번째 단어의 잠재된 토픽이 무엇인지 추론
- 깁스 샘플링 적용을 위해 문서1의 두번째 단어의 토픽 정보를 지움
  - 문서1의 단어별 토픽 분포

z_{1j}	3	?	1	3	1
$w_{1,n}$	고양이	구름	나무	별	겨울

단어	토픽1	토픽2	토픽3
고양이	1	0	35
…	…	…	…
구름	10	8-1	1

09 깁스 샘플링의 적용

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

3 토픽 추론 과정 그림으로 확인- 문서 내 단어들의 토픽 분포(θ)

- 깁스샘플링 수식의 A값은 같은 문서 내 단어들의 토픽 분포(θ)에 영향을 받는다.

z_{1j}	3	?	1	3	1
$w_{1,n}$	고양이	구름	나무	별	겨울

토픽1 토픽2 토픽3

4 토픽 추론 과정 그림으로 확인 - 토픽 내 단어 분포(β)

- 깁스샘플링 수식의 B값은 토픽 내 단어의 분포(β)에 영향을 받는다.

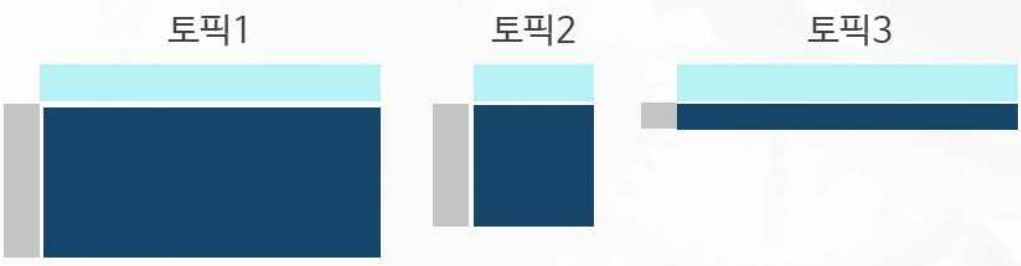
단어	토픽1	토픽2	토픽3
고양이	1	0	35
…	…	…	…
구름	10	8-1	1

09 깁스 샘플링의 적용

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

5 토픽 추론 과정 그림으로 확인 - 토픽의 도출 : A와 B의 곱으로 도출

- 단어의 주제가 어떤 토픽이 될지 그 확률은 각각 넓이로 이해할 수 있다.



SUMMARY

학습정리

- 단어 임베딩의 의미와 필요성
- 단어 임베딩 종류와 알고리즘
- 단어 임베딩 평가 방식
- 단어 임베딩의 한계
- 단어 임베딩의 시각화
- 가중 임베딩, 문중 임베딩의 이해

EXPANSION

확장하기

1. 단어 임베딩을 사용하는 이유는 무엇이고 그 종류에는 어떤 것들이 있을까요?
2. 단어 임베딩의 유사도 평가와 유추 평가는 무엇일까요?
3. 단어 임베딩의 유사도 평가 방법에는 어떤 것들이 있을까요?
4. 문장 임베딩은 무엇이고 어떤 모델들이 있을까요?

참고 문헌

REFERENCE

◆ 참고 사이트
- 용어들에 대한 정의 : https://ko.wikipedia.org/wiki.
- 한국어 임베딩 저자 이기창님 블로그 : https://ratsgo.github.io
- 딥러닝을 이용한 자연어 처리 입문 : https://wikidocs.net/book/2155
- 퍼블릭에이아이(www.publicai.co.kr)

◆ 참고 서적
- 이기창, 『한국어 임베딩』, 에이콘, 2019

서체 출처 : 에스코어드림체-㈜에스코어, 나눔글꼴체-㈜네이버