


Ku - Big 자연어 처리 실전반

kubig 11기 강호석



커리큘럼

1. Text Preprocessing

- Tokenize
- Cleaning and normalization
- encoding

✓ 2. Word Representation

2-1 Local Representation

- Bag of words
- Document term Matrix
- TF-IDF
- * Document Similarity

2-2 Continuous Representation

- Word2Vec
- Glove, Elmo

3. Text Classification with RNN

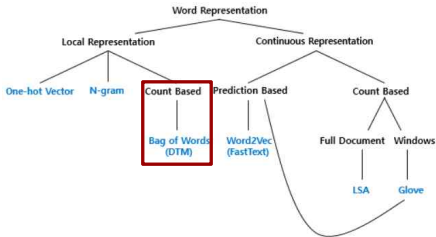
- Naive bayes
- BiLSTM

4. 심화과정

- Tagging
- Encoder & Decoder / Transformer

Local Representation

Local Representation



Local Representation

- 매핑하려는 단어만 고려하여 값을 매핑
- One-hot Vector, Bow, DTM, TF-IDF

Continuous Representation

- 주변단어들을 고려하여 값을 매핑
- Word2Vec, FastText, Glove

Local Representation

1. Bag of Words(Bow)

단어의 출현빈도에만 기반하여 텍스트 데이터를 수치화하는 방법

단어의 순서는 고려하지 않음

Bow 생성과정

- 1) 각 단어 고유한 인덱스를 매핑
- 2) 각 인덱스 위치에 단어가 등장한 횟수 기록

Local Representation

Review 1: This movie is very scary and long

Review 2: This movie is not scary and is slow

Review 3: This movie is spooky and good

단어 빈도에 기반한 표현방법

⇒ 분류 및 유사도 문제에 활용

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

Local Representation

Document To Matrix(DTM)

Bow들을 쌓아 행렬 형태로 만든 것

-	과일이	길고	노란	먹고	바나나	사과	싫은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

Local Representation

Bow와 DTM의 한계

1) Sparse Representation(희소 표현)

문서가 많아 단어 사전이 커질수록 행렬안의 0값이 기하급수적으로 증가

공간낭비 및 계산량 증가

2) 단순 빈도 접근

단어간 유사도 정보가 없음

단어 별 중요도를 판단할 수 없음

Local Representation

Tf-IDF(Term Frequency-Inverse Document Frequency)

DTM에 단어 사용빈도와 문서 내 단어 빈도를 가중치로 부여

(1) $tf(d,t)$: 특정 문서 d 에서의 특정 단어 t 의 등장 횟수.

(2) $df(t)$: 특정 단어 t 가 등장한 문서의 수.

(3) $idf(d, t)$: $df(t)$ 에 반비례하는 수.

$$idf(d, t) = \log\left(\frac{n}{1 + df(t)}\right) \quad * n \text{은 전체 문서의 개수}$$

Local Representation

Tf-idf 적용

-	과일이	길고	노란	먹고	바나나	사과	싫은	저는	좋아요
문서1	0	0	0	0.287682	0	0.693147	0.287682	0	0
문서2	0	0	0	0.287682	0.287682	0	0.287682	0	0
문서3	0	0.693147	0.693147	0	0.575364	0	0	0	0
문서4	0.693147	0	0	0	0	0	0	0.693147	0.693147

문서 내에서 반복되는 단어는 가중치

많은 문서에서 자주 등장하는 단어는 역가중치

Local Representation

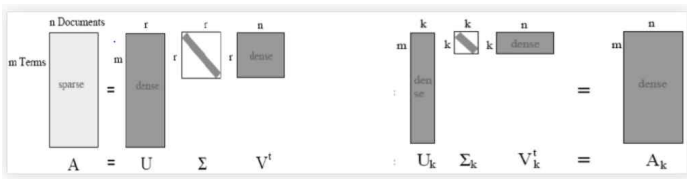
LSA (Latent Semantic Analysis)

- 빈도 기반 표현은 자체적으로 단어에 의미를 비교할 수 없음
- 문서내에 잠재적인 의미를 찾아내는 방법
- FA, PCA

Local Representation

LSA (Latent Semantic Analysis)

- k 를 설정하여 중요한 잠재의미만을 추출 ($k < r$)



Local Representation

LDA (Latent Dirichlet Allocation)

- LSA : DTM을 차원 축소 하여 축소 차원에서 근접 단어들을 토픽으로 묶는다.
- LDA : 단어가 특정 토픽에 존재할 확률과 문서에 특정 토픽이 존재할 확률을 결합확률로 추정하여 토픽을 추출한다.

Local Representation

LDA (Latent Dirichlet Allocation)

문서1 : 저는 사과랑 바나나를 먹어요

문서2 : 우리는 귀여운 강아지가 좋아요

문서3 : 저의 깜찍하고 귀여운 강아지가 바나나를 먹어요

<각 문서의 토픽 분포>

문서1 : 토픽 A 100%

문서2 : 토픽 B 100%

문서3 : 토픽 B 60%, 토픽 A 40%

<각 토픽의 단어 분포>

토픽A : 사과 20%, 바나나 40%, 먹어요 40%, 귀여운 0%, 강아지 0%, 깜찍하고 0%, 좋아요 0%

토픽B : 사과 0%, 바나나 0%, 먹어요 0%, 귀여운 33%, 강아지 33%, 깜찍하고 16%, 좋아요 16%

Local Representation

LDA (Latent Dirichlet Allocation)

- 문서가 이와 같은 순서로 작성되었음을 가정

1) 문서에 사용할 단어의 개수 N 을 정합니다.

- Ex) 5개의 단어를 정하였습니다.

2) 문서에 사용할 토픽의 혼합을 확률 분포에 기반하여 결정합니다.

- Ex) 위 예제와 같이 토픽이 2개라고 하였을 때 강아지 토픽을 60%, 과일 토픽을 40%와 같이 선택할 수 있습니다.

3) 문서에 사용할 각 단어를 (아래와 같이) 정합니다.

3-1) 토픽 분포에서 토픽 T 를 확률적으로 고릅니다.

- Ex) 60% 확률로 강아지 토픽을 선택하고, 40% 확률로 과일 토픽을 선택할 수 있습니다.

3-2) 선택한 토픽 T 에서 단어의 출현 확률 분포에 기반해 문서에 사용할 단어를 고릅니다.

- Ex) 강아지 토픽을 선택하였다면, 33% 확률로 강아지란 단어를 선택할 수 있습니다. 이제 3)을 반복하면서 문서를 완성합니다.

Local Representation

LDA (Latent Dirichlet Allocation)

1) 사용자는 알고리즘에게 토픽의 개수 k 를 알려줍니다.

앞서 말하였듯이 LDA에게 토픽의 개수를 알려주는 역할은 사용자의 역할입니다. LDA는 토픽의 개수 k 를 입력받으면, k 개의 토픽이 M 개의 전체 문서에 걸쳐 분포되어 있다고 가정합니다.

2) 모든 단어를 k 개 중 하나의 토픽에 할당합니다.

이제 LDA는 모든 문서의 모든 단어에 대해서 k 개 중 하나의 토픽을 랜덤으로 할당합니다. 이 작업이 끝나면 각 문서는 토픽을 가지며, 토픽은 단어 분포를 가지는 상태입니다. 물론 랜덤으로 할당하였기 때문에 사실 이 결과는 전부 틀린 상태입니다. 만약 한 단어가 한 문서에서 2회 이상 등장하였다면, 각 단어는 서로 다른 토픽에 할당되었을 수도 있습니다.

3) 이제 모든 문서의 모든 단어에 대해서 아래의 사항을 반복 진행합니다. (iterative)

3-1) 어떤 문서의 각 단어 w 는 자신은 잘못된 토픽에 할당되어져 있지만, 다른 단어들은 전부 올바른 토픽에 할당되어져 있는 상태라고 가정합니다. 이에 따라 단어 w 는 아래의 두 가지 기준에 따라서 토픽이 재할당됩니다.

- $p(\text{topic } t \mid \text{document } d)$: 문서 d 의 단어들 중 토픽 t 에 해당하는 단어들의 비율
- $p(\text{word } w \mid \text{topic } t)$: 단어 w 를 갖고 있는 모든 문서들 중 토픽 t 가 할당된 비율

Local Representation

LDA (Latent Dirichlet Allocation)

doc1

word	apple	banana	apple	dog	dog
topic	B	B	???	A	A

doc2

word	cute	book	king	apple	apple
topic	B	B	B	B	B

doc1

word	apple	banana	apple	dog	dog
topic	B	B	???	A	A

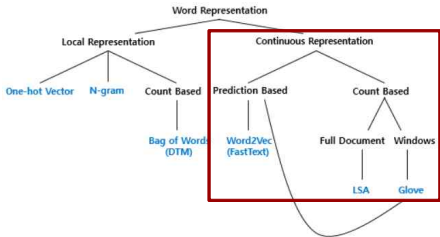
doc2

word	cute	book	king	apple	apple
topic	B	B	B	B	B

- apple을 어떤 topic에 할당할지 결정
- 해당 문서 => 전체 문서 순으로 확률이 높은 토픽에 할당
- 모든 단어에 대해 반복

Continuous Representation

Continuous Representation



Local Representation

- 매핑하려는 단어만 고려하여 값을 매핑
- One-hot Vector, Bow, DTM, TF-IDF

Continuous Representation

- 주변단어들을 고려하여 값을 매핑
- Word2Vec, FastText, Glove

Continuous Representation

워드 임베딩

밀집 표현(Dense Representation)

단어 사전의 크기가 벡터의 크기인 희소 표현인 것에 비해

사용자가 지정한 크기로 조정이 가능한 밀집 표현

강아지 = [0 0 0 0 1 0 0 0 0 0 0 0 ... 0] # 이 때 1 뒤의 0의 수는 9995개. 차원은 10,000

강아지 = [0.2 1.8 1.1 -2.1 1.1 2.8 ...] # 이 벡터의 차원은 128로 지정

Continuous Representation

워드 임베딩

워드 임베딩은 단어를 밀집 형태로 표현하는 것

-	원-핫 벡터	임베딩 벡터
차원	고차원(단어 집합의 크기)	저차원
다른 표현	희소 벡터의 일종	밀집 벡터의 일종
표현 방법	수동	훈련 데이터로부터 학습함
값의 타입	1과 0	실수

Continuous Representation

Word2Vec

분산 표현 : 단어의 '의미'를 다차원 공간에 벡터화하는 방법

'비슷한 위치의 단어는 비슷한 의미를 가진다'를 가정

ex) 한국 - 서울 + 도쿄 = 일본, 고양이 + 애교 = 강아지

C-Bow와 skip-gram 두 가지 방식

Continuous Representation

Word2Vec

C-bow : 주변 단어를 통해 주위 단어를 예측

중심 단어 주변 단어

↓ ↓

The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

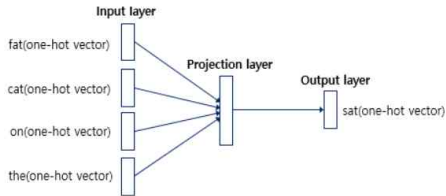
The fat cat sat on the mat

중심 단어	주변 단어
[1, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 1, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]

* 주변 단어 개수를 윈도우라고 부름

Continuous Representation

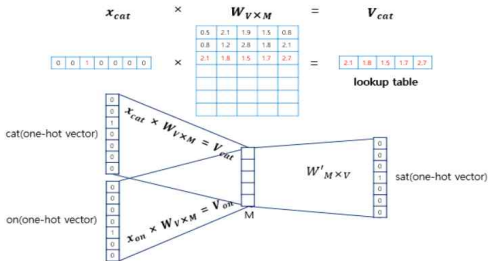
Word2Vec



주변 단어를 통해 중심 단어를 밀집 벡터로 표현

Continuous Representation

Word2Vec



W와 W'을 학습(훈련 전에는 작은 랜덤 값)

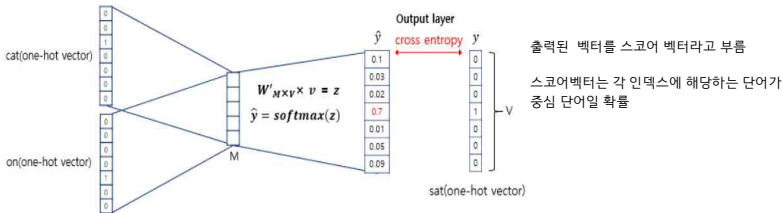
학습 후 출력되는 밀집 벡터의 차원은 M

생성되는 V들의 평균이 곧 중심단어의 분산표현

즉, W 일부분의 열평균이 중심단어의 분산표현

Continuous Representation

Word2Vec



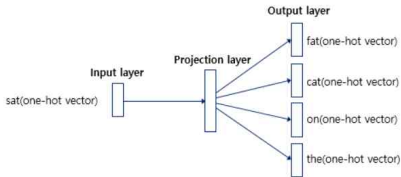
출력된 벡터를 스코어 벡터라고 부름

스코어벡터는 각 인덱스에 해당하는 단어가
중심 단어일 확률

Continuous Representation

Word2Vec

skip-gram



Continuous Representation

Word2Vec

Subsampling frequent words

- W, W'가 단어 수가 증가할수록 계산량이 급격히 증가
- 자주 등장하는 단어는 학습량을 줄이도록 설정
- 아래의 식은 i 번째 단어를 학습에서 제외시키기 위한 확률

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \quad f(w_i) \text{는 해당 단어가 말뭉치에 등장한 비율, } t \text{는 } 0.00001 \text{을 권장}$$

Continuous Representation

Word2Vec

Skip-Gram with Negative Sampling

- 전체 단어가 많아질수록 softmax 계산을 위한 비용이 증가
- 윈도우 사이즈 내에 등장하지 않는 단어(negative sample)를 5~20개 정도 선정
이를 정답단어와 합쳐 전체 단어처럼 소프트맥스 확률을 구하는 것
- 윈도우에 등장하지 않은 단어가 negative sample로 뽑힐 확률

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^n f(w_j)^{3/4}}$$

Continuous Representation

Glove

- DTM과 Tf-idf는 전체적인 통계정보를 활용
- word2vec은 의미 유추에 뛰어남
- 두 가지 방법의 절충안이 Glove

Continuous Representation

Glove

카운트	I	like	enjoy	deep	learning	NLP	flying
I	0	2	1	0	0	0	0
like	2	0	0	1	0	1	0
enjoy	1	0	0	0	0	0	1
deep	0	1	0	0	1	0	0
learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
flying	0	0	1	0	0	0	0

Continuous Representation

Glove

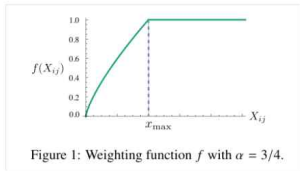
동시 등장 확률과 크기 관계 비(ratio)	k=solid	k=gas	k=water	k=fasion
$P(k \mid \text{ice})$	0.00019	0.000066	0.003	0.000017
$P(k \mid \text{steam})$	0.000022	0.00078	0.0022	0.000018
$P(k \mid \text{ice}) / P(k \mid \text{steam})$	8.9	0.085	1.36	0.96

Continuous Representation

Glove

임베딩 된 중심 단어와 주변 단어 벡터의 내적이 전체 코퍼스에서의 동시 등장 확률이 되도록 만드는 것

$$Loss = \sum f(x_{ij}) \times (w_i^t w_j + b_i + b_j - \log(x_{ij}))^2$$



w : 각 단어의 임베딩 벡터

b : 각 단어들의 기본 빈도수(Bias)

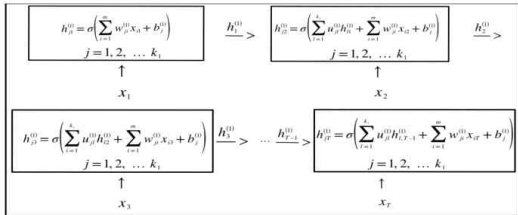
x_{ij} : 두 단어가 동시 등장할 확률

$F(X_{ij})$: 단어 등장 빈도에 따른 가중치 함수

Continuous Representation

Elmo(Embeddings from Language Model)

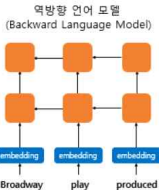
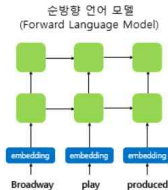
- word2vec, Glove 방법은 문맥을 고려하지 못한다는 단점 \Rightarrow RNN모델 도입



Continuous Representation

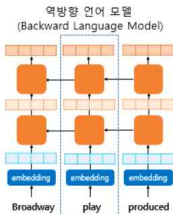
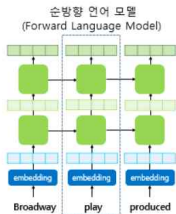
Elmo(Embeddings from Language Model)

- 양방향 맥락을 고려하는 모델



Continuous Representation

Elmo(Embeddings from Language Model)



1) 각 층의 출력값을 연결(concatenate)한다.



Continuous Representation

Elmo(Embeddings from Language Model)

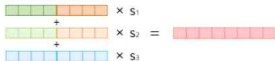
1) 각 층의 출력값을 연결(concatenate)한다.



2) 각 층의 출력값 별로 가중치를 준다.




3) 각 층의 출력값을 모두 더한다.




4) 벡터의 크기를 결정하는 스칼라 매개변수를 곱한다.





제목을 입력하십시오

부제목을 입력하십시오



제목을 입력하십시오

제목을 입력하십시오

내용을 입력하십시오

제목을 입력하십시오

내용을 입력하십시오

내용을 입력하십시오

xx%

내용을 입력하십시오

제목을 입력하십시오

제목을 입력하십시오

내용을 입력하십시오

제목을 입력하십시오

제목을 입력하십시오

부제목을 입력하십시오

내용을 입력하십시오

내용을 입력하십시오

xx%

내용을 입력하십시오

