NLP 2주차 : 워드 임베딩

태그

▼ **목차**

인코딩(Encoding)

데이터를 기계가 이해할 수 있도록 숫자 등으로 변환해주는 작업을 “인코딩”이라고 한다. 텍스트 처리 분야에서는 정수 인코딩과 원-핫 인코딩이 쓰인다.

1. 정수 인코딩(Integer Encoding)

정수 인코딩은 단어에 정수를 부여하는 방법이다.

1. 하나의 단어를 빈도수 순으로 정렬한 **단어 집합(vocabulary)**을 만들고, 빈도수가 높은 순서부터 차례로 낮은 숫자부터 정수를 부여하는 방법
2. 각 단어와 정수 인덱스를 연결하고, 토큰을 변환해주는 방법

   > `{'you' : 0, 'say' : 1, 'goodbye' : 2, ....}`

   이런 식으로 정수값으로 라벨링하여 “단어 사전”을 만든다.

2. 원-핫 인코딩(One-Hot Encoding)

원-핫 인코딩은 단어 사전(=정수 인코딩한 결과)의 크기를 벡터의 차원으로 하는 방법이다. 이렇게 표현된 벡터를 **원-핫 벡터(One-Hot vector)**라고 한다.

벡터 표현 방식

표현하고 싶은 단어의 인덱스에 1의 값을 부여하고, 다른 인덱스에는 전부 0을 부여한다.

만약 단어 집합의 크기가 10이고, 사과가 0번 인덱스의 단어라면,

> “1 0 0 0 0 0 0 0 0 0”

이런 식으로 표현된다.

▼ 원-핫 인코딩의 한계

1. 단어의 개수가 늘어날수록 벡터의 차원이 늘어나 필요한 공간이 계속 늘어난다.
2. 단어 간 유사도를 표현하지 못한다.

> 강아지, 개, 고양이 3가지 단어가 있을 때 세 단어 간의 유사 정도를 파악할 수 없다.

워드 임베딩 (Word Embedding)

텍스트 → 숫자(컴퓨터가 처리할 수 있는)

컴퓨터는 일반적으로 텍스트 보다 숫자를 더 잘 처리한다. 따라서 컴퓨터가 텍스트를 계산할 수 있도록 숫자로 변환해야한다. 그 과정에서 단어의 의미를 얼마나 잘 수치화하는지에 따라 자연어 처리의 성능도 달라진다.

텍스트를 숫자로 바꾸는 방법은 여러가지가 있다. 그중 “워드 임베딩”이란, 각 단어를 신경망 학습을 통해 벡터화하는 방법이다.

1. 희소 표현 (Sparse Representation)

앞서 본 원-핫 인코딩과 같이, 벡터 또는 행렬(matrix)의 값이 **대부분이 0으로 표현**되는 방법이다. 그러므로 0과1로 되어있는 원-핫 벡터는 희소 벡터(sparse vector)이다.

벡터의 차원은 단어의 개수에 따라 결정된다. 단어가 10,000개 있다면 벡터의 차원은 10,000이어야 한다.

▼ 희소 표현의 한계

1. 희소 벡터는 단어의 개수가 늘어나면 벡터의 차원이 한없이 커진다. → 공간적 낭비
2. 원-핫 벡터는 단어의 의미 또는 단어 간 유사도를 표현하지 못한다.

2. 밀집 표현 (Dense Representation)

희소 표현과 반대되게, 벡터의 차원을 단어 집합의 크기로 상정하지 않는다. **사용자가 설정한 값**으로 모든 단어의 벡터 표현의 차원을 맞춘다. 이 과정에서 더이상 0이나 1만 가진 값이 아니라 실수값을 가지게 된다.

희소벡터의 예

> Ex) 강아지 = [0 0 0 0 1 0 0 0 0 0 0 0 ... 중략 ... 0] # 이때 1 뒤의 0의 수는 9995개. 차원은 10,000

밀집벡터의 예

> Ex) 강아지 = [0.2 1.8 1.1 -2.1 1.1 2.8 ... 중략 ...] # 이 벡터의 차원은 128

이렇게 벡터의 차원이 조밀해져 “밀집 벡터(dense vector)”라고 부른다.

과정

단어를 랜덤한 값을 가지는 밀집 벡터로 변환한 뒤에, 인공 신경망의 가중치를 학습하는 것과 같은 방식으로 단어 벡터를 학습한다.

3. 분산 표현 (Distributed Representation)

분산 표현 방법은 **’비슷한 문맥에서 등장하는 단어들은 비슷한 의미를 가진다’** 라는 가정 하에 만들어진 표현방법이다.

분포 가설 (distributional hypothesis)

> 예를 들어, 강아지란 단어는 귀엽다, 예쁘다, 애교 등의 단어가 주로 함께 등장하는데,
> 분포 가설에 따라서 저런 내용을 가진 텍스트를 벡터화한다면, 저 단어들은 의미적으로 가까운 단어가 된다.

희소 표현 방법에서는 각 단어 벡터 간 의미나 유사성을 표현할 수 없다는 단점이 있었기에, 대안으로 단어의 의미를 다차원 공간에 벡터화하는 방법을 사용하기로 했다.

과정

분포 가설을 이용하여 단어들의 셋을 학습하고, 벡터에 단어의 의미를 여러 차원에 분산하여 표현한다.

이렇게 분산 표현을 이용하여 단어 간 의미적 유사성을 벡터화하는 작업을 워드 임베딩이라고 하고, 그 결과로 나온 벡터를 “임베딩 벡터(embedding vector)”라고 한다.

정리

텍스트→숫자	희소 표현	밀집 표현	분산 표현
표현 방식	여러 고차원에 분리된 표현	사용자가 설정한 값으로 단어의 차원을 맞춘 표현	여러 저차원에 단어의 의미를 분산하여 표현
벡터	원-핫 벡터, 희소 벡터	밀집 벡터	임베딩 벡터
단점	공간적 낭비 발생, 단어 간 의미 표현 불가		
워드 임베딩	x	o	o

워드투벡터 (Word2Vec)

분산표현 방법의 대표적인 모형이 2013년 Google에서 발표한 **Word2vec**이다. 단어 벡터 간 유의미한 유사도를 반영할 수 있도록 단어의 의미를 수치화하는 방법을 “워드투벡터”라고 한다.

워드투벡터가 하는 일

http://w.elnn.kr/search/

한국어 단어로 벡터 연산을 해볼 수 있는 사이트이다. 여기서는 단어들로 더하기, 빼기 연산을 할 수 있다(=실제로는 Word2Vec 벡터). 예를 들어 아래의 식에서 좌변을 집어 넣으면, 우

변의 답들이 나온다.

> 한국 - 서울 + 도쿄 = 일본
> 박찬호 - 야구 + 축구 = 호나우두

단어의 의미를 가지고 연산을 하고 있는 것처럼 보인다. 이런 연산을 통해 분산표현을 이해할 수 있다.

Word2Vec에는 CBOW(Continuous Bag of Words)와 Skip-Gram 두 가지 방식이 있다.

주변 단어와 중심단어

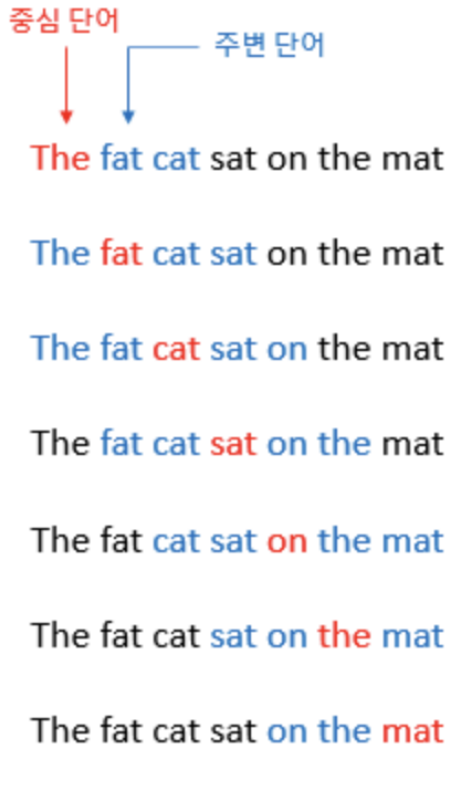
> 코퍼스(말뭉치) 예문 : "The fat cat sat on the mat"
>
> `{"The", "fat", "cat", "on", "the", "mat"}` 으로부터 `sat`을 예측하는 상황

이 때 예측해야하는 단어 `sat`을 “중심 단어(center word)”라고 하고, 예측에 사용되는 단어들을 “주변 단어(context word)”라고 한다.
주변 단어로 사용할 앞, 뒤로 단어들의 범위를 “윈도우(window)”라고 한다. 예를 들어, 윈도우 크기가 2이고, 예측하고자 하는 중심 단어가 `sat`이라고 한다면 앞의 두 단어인 `"fat"`와 `"cat"`, 그리고 뒤의 두 단어인 `"on"`, `"the"`를 참고한다.

윈도우 크기를 정했다면, 윈도우를 계속 움직여서 주변 단어와 중심 단어 선택을 바꿔가며 학습을 위한 데이터 셋을 만들 수 있는데, 이 방법을 “슬라이딩 윈도우(sliding window)”라고 한다.

1. CBOW(Continuous Bag of Words)

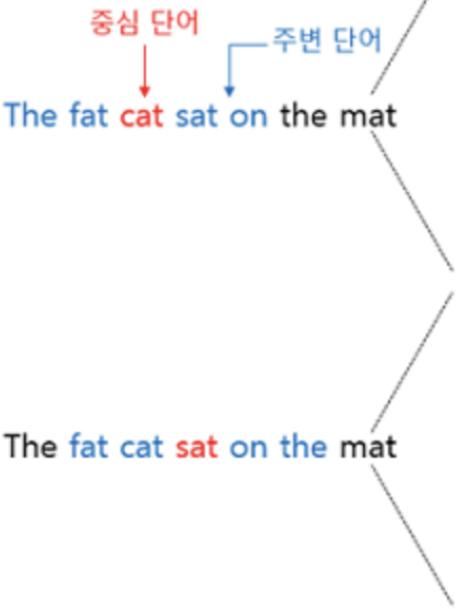
- 주변 단어들을 가지고, 중심 단어들을 예측하는 방법



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]
[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, 1]	[0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]

2. Skip-gram

- 중심 단어로 주변 단어들을 예측하는 방법



중심 단어	주변 단어
cat	The
cat	Fat
cat	sat
cat	on
sat	fat
sat	cat
sat	on
sat	the

여러 논문에서 성능 비교를 진행했을 때, 전반적으로 Skip-gram이 CBOW보다 성능이 좋다고 알려져 있다. 왜냐하면, 모델이 학습할 때 Skip-gram이 CBOW에 비해 비교적 여러 문맥을 고려하기 때문이다.

관련 설명 블로그 : https://heytech.tistory.com/353

워드 투 벡터 실습 :: 깃허브 주소

https://github.com/Kyubyong/wordvectors