# WORD EMBEDDING

\* WORD: Discrete, not Continuous

- 단어는 분리되어 개별적으로 존재하기 때문에 이산적이고 categorical value(범주 값 형태)형태이지만, 우리 머리속에서는 다르게 동작한다.

우리 머리속에서는 어휘는 계층적 의미구조를 가지고, 단어 사이에는 유사성을 가지고 있다. 하지만 컴퓨터는 유사성에 대해서 알까??

- One-hot 인코딩으로 표현된 값은 유사도나 모호성을 표현할 수 없다. One-hot 인코딩은 Sparse vector이기 때문이다. 표현된 값을 유사도나 모호성을 표현하려면 Dense vector로 표현하는게 유리하다.

\* Feature Vector

각각 특징들을 모아서 하나의 vector로 만드는 것. 단어의 feature vector은 무엇이 될까? 신경망은 x와 y사이의 관계를 학습하는 과정에서 자연스럽게 x의 feature를 추출하는 방법을 학습한다. (원래 입력을 복원하는 방법)

- 레이어 중간의 hidden representation은 y의 값을 구하기 위해 x에서 필요한 정보를 더 작은 차원에 압축 표현한 것이라 할 수 있다. (오토인코더와 유사)

\* Word Embedding

딥러닝의 시대에 들어와 신경망의 이러한 특성을 활용하여 단어를 연속적인 값으로 표현하자 하였다. Non-Linear 함수를 이용한 차원축소를 통해 Representation Learning을 잘하는 특성을 통해 Embedding을 잘할 수 있게 되었다. 이것은 단어의 유사도등의 연산에 유리하다.

\* Word Sence

- Homonym and Polysemy (동의어(동형어)와 다의어)

다의어: 의미는 다르지만 근본은 같음

동의어: 의미가 완전 다름

|  |  |  |  |
| --- | --- | --- | --- |
| 종류 | 단어형태 | 의미1 | 의미2 |
| 다의어(Polysemy) | 강남역 | 지하철 역사 | 역 주변 상권 |
| 동형어(Homonym) | 차 | 마시는 차(tea) | 달리는 차(car) |

We need Word Sense Disambiguation (WSD)

- Synonyms (동의어)

동의어: 비슷한 의미를 갖는 두 단어나 구를 가리키며, 서로 대체하여 사용할 수 있는 말

- Antonyms (반의어)

반의어: 반대적 의미를 가지는 언어

- Hypernyms & Hyponyms (상위어, 하위어)

단어는 개념적 의미를 가지며, 개념을 포괄하는 상위 개념을 가지며 계층적 구조를 가진다.

우리는 One-hot 인코딩을 통해 단어를 표현한다. (sparse vector) 이것을 워드 임베딩을 통해서 dense vector로 표현해서 위의 내용들을 표현할 수 있다.

# WordNet

: Thesaurus어휘의 유의어 사전이다. NLTK를 통해서 영어 사전을 사용할 수 있으며, 한국어 사전도 존재한다. 동의어 집합, 상위어, 하위등에 대한 정보가 잘 구축되어있다.

- 단어의 계층적 구조를 잘 파악

- 동사의 집합을 구할 수 있으며, 단어아시의 유사도를 계산할 수 있다.

이 모든게 코퍼스 없이 가능하나, 신조어나 틀린부분이(Hand craft방식이므로) 있다면???

Thesaurus는 단어기반의 사전이고, 신조어나 도메인에 따라서 활용도가 달라진다.

\* TF-IDF (Term Frequency Inverse Document Frequency)

: 텍스트 마이닝에서 중요하며 자연어 처리와 겹치는 부분이 많다. 다른점은 테스트 마이닝은 텍스트에서 정보추출을 중요시한다.

- 어떤 단어 w가 문서 d내에서 얼마나 중요한지 나타내는 수치 (문서내의 단어의 중요도)

- TF(Term Frequency): 문서내에서 단어의 출현 횟수. 숫자가 클수록 매우 중요하다 하지만 불용어는 TF가 값이 매우 크기 때문에 패널티가 필요하다.

- IDF (Inverse Document Frequency): 문서에서 단어가 출현한 숫자의 역수. 값이 클수록 ‘the’와 같이 일반적으로 많이 쓰이는 단어다.

이 TF-IDF를 feature로 쓰일 순 없을까?

\* Based on Context Window (Co-occurrence)

- 함께 나타내는 단어들을 사용

- 가정

의미가 비슷하면 쓰임새가 비슷할 것, 비슷한 문장에서 비슷한 역할로 사용될 것이다. 따라서 함께 나타나는 단어들은 유사할 것이다.

Context Window를 사용하여 windowing을 실행한다. window크기라는 hyper-parameter를 추가해야 한다. 이것은 hand-craft feature

텍스트, 스크린샷, 번호, 달력이(가) 표시된 사진

자동 생성된 설명

Thesaurus시소러스 기반 방식에 비해 코퍼스(or 도메인) 특화된 표현 가능

하지만 여전히 sparse한 vector로 표현된다.

Worndnet과 달리 데이터 기반(데이터의 출현 횟수)지만, traditional인 방법은 데이터 기반이라 특화된 표현이 가능하다. 하지만 sparse vector라는 한계가 발생한다.

# Word2vec

- 주변(Context window)에 같은 단어가 나타나는 단어일수록 비슷한 벡터를 가져야 한다. Co-occurrence에서 같은 방식의 가정이다. 문장의 문맥에 따라서 정해지는 것이 아니다. Context window의 사이즈에 따라서 embedding의 성격이 바뀔 수 있다.

도표, 라인, 텍스트, 평면도이(가) 표시된 사진

자동 생성된 설명

Cbow: 주변에 단어가 주어졌을 때 현재 빈칸 단어를 예측한다.

Skip-gram: 현재 단어가 주어졌을 때 주변 단어를 예측한다. (널리 알려지고 더 좋다고한다.)

\* Skip gram

- 주변 단어를 예측하는 과정에서 적절한 단어의 임베딩(정보의 압축)을 할 수 있다.

Non-linear activation functions이 없다. 두개의 linear layer로 구성되어 있다. Word2vec은 딥러닝이지만, 2개의 linear layer로 구성되어 있다. (non-linear layer가 아니다)

- 기본적인 개념은 오토인코더와 비슷하다. y를 성공적으로 예측하기 위해 필요한 정보를선택 미 압축을 한다.

도표, 라인, 텍스트, 그래프이(가) 표시된 사진

자동 생성된 설명

Corpus가 있으면 window가 움직이면서 윈도우 내에서 주변단어를 맞춘다. Window가 문서를 훑을 때마다 1epoch이다. 중심단어를 기준으로 주변을 맞췄다. 이 과정은 주변 예측을 이용하기 때문에 classification이므로 softmax 함수를 이용한다. 이것을 통해 word embedding을 수행한다. Linear layer지만 딥러닝이고, 단일선형 layer 모양이다. 예전에는 쉽고 빠르고 비교적 정확도가 높았다고 평가를 받았지만 현재는 아니다. Skip-gram은 classification task에 많이 사용된다.

Skip-gram은 자연스럽게 정보를 압축하는 과정에서 one-hot vector를 dense한 vector를 구하는 과정이다. 즉, 주변 단어를 예측하는 과정에서 필요한 정보를 압축해서 중간의 hidden vector가 embedding vector가 된다. 이것으로 유사도를 측정할 수 있다. 이전 방식보다 dense하기 때문에 유사도, 모호성, 유의성을 계산하는데 유리한 방식이다. Skip-gram은 중심단어가 주어졌을 때 주변 단어를 맞추는 과정에서 필요한 정보를 선택, 압축하는 word embedding이며 classification에서 많이 사용되는 수행방식이다.

# Glove

- 단어 x와 윈도우 내에 함께 출현한 단어들의 출현 빈도를 맞추도록 훈련한다. (이전에는 각각의 단어를 맞췄다. 이번에는 전체 단어에 대해서 출현빈도를 맞춘다.)

- Word2vec은 classification이에 사용되지만, Glove는 Regression task에 많이 사용된다.

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

Glove는 corpus 내에서 window가 한번 훑고, co-occurrence기반(빈도수 기반) matrix를 만들고, feature vector를 예측한다. 현재 단어가 주어졌을 때, 코퍼스 내에서 주변 단어 빈도에 따르는 과정에서 정보의 선택과 압축을 할 때 Embedding vector가 구해진다.

- 출현 빈도가 적은 단어에 대해서는 loss의 기여도를 낮추는 패널티를 부과한다. 따라서 출현 빈도가 적은 단어에 대해서 부정확해지는 단점을 보완한다.

- Skip-gram보다 더 빠르며, 전체 코퍼스에 대해서 co-occurrence를 구한 후 regression을 수행한다. (skip-gram처럼 )

- 출현 빈도가 적은 단어도 벡터를 비교적 정확하게 구할 수 있다.

- 현재 단어가 주어졌을 때, 주변 나타나는 단어들의 빈도를 맞추는 과정에서, 정보의 선택과 압축이 나타나는 것이 Word Embedding이다. 여기서는 Glove는 regression에 많이 사용되는 방식이다.

# Fasttext : Upgrade version of Skip gram

\* Skip gram Review: linear activation function의 neural network를 사용하여 현재 time-step의 단어가 주어졌을 때 window내에 있는 주변 단어들을 soft-max를 통해서 하나씩 예측한다. 특히 classification에서 많이 사용되는 기법이다.

기존의 word2vec은 저빈도 단어에 대학 학습과 OoV에 대한 대처가 어려웠다.

Fastext는!

1) 단어를 학습시 subword로 나누고(OoV 줄이기)

2) skip-gram을 활용하여 subword에 대한 embedding vector에 주변 단어의 context vector를 곱하여 더한다.

3) 이 값이 최대가 되도록 한다.

최종적으로 subword에 대한 embbeding vector의 합이 word embedding vector가 된다.

텍스트, 도표, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

Subword로 나뉘어지는 것을 볼 수 있다.

Skip-gram은 one-het vector를 입력을 받고, fasttext는 subword를 입력을 받는다.

결론적으로는 word2vec의 skip-gram, glove, fastetext 어떤 알고리즘이 더 뛰어나다고 할 수는없다. 구현이 쉽고 빠른 오픈 소스를 사용하자. 두개의의 다른 알고리즘 결과물을 concat하기도 한다.

Fasttext 홈페이지에서 간단하게 text classification 할수 도 있다.

Fasttext를 아나콘다 주피터에서 설치할떄는 오류가 뜨더라. fasttext알집을 lib/site-package로 옮겨서 사용하면 import fasttext가 되더라.

텍스트, 폰트, 친필, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 영수증이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 스크린샷, 친필이(가) 표시된 사진

자동 생성된 설명

도표, 라인, 스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 영수증, 화이트이(가) 표시된 사진

자동 생성된 설명

# Dimension Reduction Perspective

\* AutoEncoders: 원래의 입력을 복원하기 위한 Architecture

- 인코더와 디코더를 통해 압축(축소)과 해제(복원)을을 실행한다.

1) 인코더는 입력의 정보를 최대한 보존하고 손실과 압축을 진행한다 – z(bottleneck)에다가 최대한 많은 정보를 저장하며, 차원은 줄어든다. 압출을 함에 따라 중요한 정보를 선택해서 버리고 압축한다

2) 디코더는 중간 결과물 (z)의 정보가 입력x와 같아지도록 압축을 해제(복원) –

- 복원을 성공적으로 하기위해, 오토인코더는 특징을 추출하는 방법을 자동으로 학습한다.

\* Word2Vec: skip-gram, cbow

- Objective주어진 단어로 주변 단어를 예측하자

- y를 예측하기 위해 필요한 정보가 z(bottleneck)에 있어야 한다. 주변 단어를 잘 예측하기위해 x를 잘 압축하자.

텍스트, 폰트, 포스트잇 노트, 직사각형이(가) 표시된 사진

자동 생성된 설명

Word2vec 과정: z는 정보를 담고 있어야 한다. Encoder은 필요한 정보를 잘 담고 압축해야 한다. 이것은 back-propagation으로 학습되어야 한다. Loss를 minimize하기위해 이 잘 예측되어야 하는데, 이때 decoder가 잘 학습되어야 한다. 이때 학습 입력 데이터 z가 중요하다. 여기서 z가 잘 출력이 되려면 encoder가 잘 학습되어야 한다.

Word Embedding: one-got encoding인 단어를 벡터로 표현하는데 sparse vector를 dense vector화 한다. Word2vec은 벡터로 표현하는 방법 중 하나다. 다른 워드 임베딩 방법으로는 glove, fasttext가 있다.

# Embedding layer

One-hot vector를 neural network에 넣어준다. 그러면 vocabulary차원의 vector가 네트워크의 입력이 된다. 자연스럽게 embedding layer를 거치면 embedding vector가 return된다. Word embedding 된것을 neural network에 넣는 것이 아니다.

-임베딩 레이어의 구조

도표, 스크린샷, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

-그냥 linear layer다. One-hot vector를 embedding layer에 넣는다. 계산의 효율성이 올라간다. Embedding layer weight없이 바로 embedding vector를 deep learning에 넣으면 task에 따른 objective와 다르게 학습될 수 있다.

- one-hot vector를 linear layer에 넣었을 때 작동하며, one-hot vector를 embedding layer weight parameter은 word embedding vector들로 구성되어 있다.

텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Deep neural network에서 워드 임베딩 레이어는 계산의 효율성 때문에 사용된다.

Embedding layer를 통해 one-got vector를 각 task에 맞는 dense representation으로 바꿀 수 있다. 단순히 fix된 word embedding vector 자체를 넣어줄 경우, 해당 알고리즘의 objective에 최적화된 representation을 넣어주는 것이다. 특수한 경우에 한하여, fixed word representation을 사용하거나, word embedding vector들로 구성된 weight parameter를 seed로 사용하여 최적화한다.

하지만 baseline은 neural network에 one-hot vector를 넣어주는 것이다.

그렇다면, word embedding algorithm은 언제 쓰나요?

거이 사실 안쓴다… nlp를 상용화 할떄는 잘 안쓴다.

어떤 단어가 유사해요? 할때 사용한다. – 유사도나 모호성을 비교할 수 있으니까

딥러닝은 end-to-end 방식을 추구한다. 따라서 embedding layer를 통해 사용된다. 임베딩 벡터는 중간 산출물 같은 느낌이다. 개념은 알아야한다!