
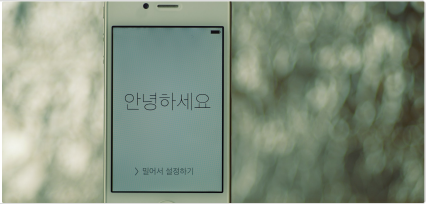


위클리 NLP week 1 ~ week 12

위클리 NLP - jiho-ml

구글 컴퓨터 언어학자가 쓰는 자연어 처리 (natural language processing)에 대한 튜토리얼! 매주 구독해서 받아보세요.

 <https://jiho-ml.com/tag/weekly-nlp/>



Week 1 - 컴퓨터에게 언어란?

corpus : 단어를 모아 vocabulary 구성
one-hot vector : 단어를 column vector로 표현 → 단어의 관계를 고려하지 않음

	hello	world	processing
“unk”: 0	0	0	0
“hello”: 1	1	0	0
“world”: 2	0	1	0
“natural”: 3	0	0	0
“language”: 4	0	0	0
“processing”: 5	0	0	1

Week 2 - 단어를 가방에 때려 넣으면 문장이 된다

여러 개의 단어 vector를 합한 것을 **bag-of-words(BoW) vector** : 단어의 순서를 고려하지 않음

“hello natural world”

0

1

0

0

0

0

+

0

0

0

1

0

0

+

0

0

1

0

0

0

=

0

1

1

0

0

0

N-gram : 연속된 n개의 단어 뭉치

ex) "I love studying machine learning"에서 n=2는 I love, love studying, studying machine, machine learning으로 총 4개

n-gram을 이용해 BoW를 만들면 vocabulary가 커지게 됨

term frequency - inverse document frequency(tf-idf) : 단어 간 빈도수에 따라 중요도를 계산

term frequency (tf) : 현재 문서에서의 단어 빈도수

document frequency (df) : 이 단어가 나오는 문서의 총 개수

tf-idf score = tf x log(N / df) (N은 전체 문서 수)

ex) 관사나 대명사는 자주 나오지만 중요하지 않은 단어(stopword) - tf, df 가 높음

인공지능, 자연어처리 같은 단어는 자주 등장하지는 않지만 중요한 단어 - tf는 높을 수 있지만, df는 낮음

⇒ 다른 문서에도 자주 나오는건 중요도를 낮춰 계산 - df가 분모로

BoW, Tf-idf vector의 단점 : 순서가 중요한 문제에는 쓰기 어려움, vocabulary가 커질수록 쓰기 어려움, 단어 간의 관계를 표현하지 못함

Week 3 - vector 기초

vector? 크기값인 scalar + 방향 / N차원 공간에 존재

Week 4 - vector

word embedding : 단어 간의 관계를 학습해 vector에 저장

one-hot vector를 차원을 줄여 표현 (sparse vector → dense vector)

- **Glove** - 같은 문장에 한 단어가 어떤 근처 단어들과 몇번 같이 나오는지 counting
but, 전체 corpus에 40000개 단어가 있으면 matrix는 40000 x 40000
⇒ dimensionality reduction을 통해 차원 축소 40000 x 40000을 300 x 40000으로 → Singular Value Decomposition (SVD) 알고리즘 사용
→ 각 열이 하나의 단어를 대표하게 됨
ex) "I enjoy fling", "I like NLP", "I like deep learning"

X =

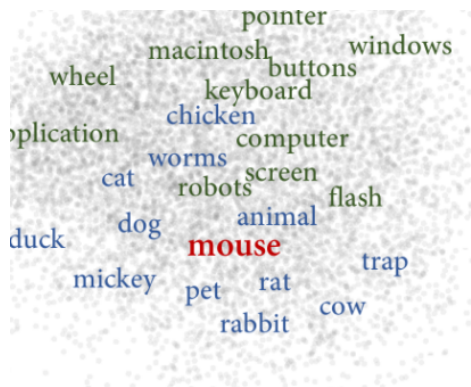
	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

co-occurrence matrix

- **word2vec** - neural network를 사용해 주변 단어와 타깃 단어의 관계를 예측(classification) → CBOW, skipgram 알고리즘을 통해 학습(skipgram으로 학습된 embedding이 결과가 더 좋음)
input은 N x 1 word embedding, 학습 될 수록 word embedding에 주변 단어와의 관계 정보가 encoding 됨
⇒ **continuous bag-of-words (CBOW)** 알고리즘 : 주변 단어들을 모두 합쳐서 본 후 타깃 단어 맞추기
⇒ **skipgram** 알고리즘 : 타깃 단어를 보고 주변 단어 맞추기

Week 5 - vector

문장, 단어들을 vector로 만들어 N차원의 공간의 한 점으로 표현



2차원으로 표현한 word embedding space

두 vector 사이의 거리 : 유클리디안 거리
두 vector 사이의 각 : cosine similarity (같은 방향이면 1, 반대 방향이면 -1, 서로 수직이면 0)
vector를 단어 빈도 수로 계산하기 때문에 NLP에서는 cosine similarity가 쓰임
<https://projector.tensorflow.org> 로 word2vec 3차원으로 볼 수 있음

Week 6 - 문서 분류

주제 별 문서 분류 : **document classification**
linear regression은 예측하는 회귀, linear regression을 sigmoid와 함께 쓰면 분류모델로 만들 수 있음 → linear regression + sigmoid = logistic regression classifier(이분법)

logistic regression 모델의 input은 vector로 만든 문장 → output은 probability score
긍/부정 분석에서는 단어들에게 가중치를 주는 feature importance를 사용해 단어들의 중요도 부여

Week 7 - 머신러닝과 NLP는 왜 함께 갈까

언어는 고차원 함수이기 때문에 완벽한 답을 찾는 것은 불가능 ⇒ 근사치를 구하는 방법으로 답을 찾음 : function approximation

딥러닝은 non-linear function의 근사치를 구하는데 유용하기 때문에 NLP와 딥러닝은 함께 해야함

Week 8 - 스팸 메일 분류기 만들기

스팸 메일 분류기 - binary classification로 **Navive Bayes classifier** 모델 사용

스팸 메일 데이터 분류 방법 1. 직접 라벨링 / 2. anomaly detection (변칙 데이터를 찾아내기)

Navive Bayes classifier = $P(d|c) P(c) \rightarrow c$: 확률, d : 문서

ex) $P(\text{이메일}|\text{spam}) = P(\text{"보험"}|\text{spam}) \times P(\text{"판매"}|\text{spam}) \dots \rightarrow \text{joint probability}$

Week 9 - 글을 그림처럼 보는 CNN

글 → 이미지로 : **CNN** (Convolutional Neural Network)

단어 = $N \times 1$ matrix (column vector) , 문장 = $N \times m$ matrix (2차원 matrix)

embedding 사이즈가 300이고, 문장에 단어가 10개라면, 10×300 matrix → 2차원 heatmap 이미지로 표현 가능

wordCNN은 1차원 필터 사용 ← depth 고려 x

pretrained word embedding

- word2vec~skipgram <https://code.google.com/archive/p/word2vec/>

- glove <https://nlp.stanford.edu/projects/glove/>

- fasttext~skipgram <https://fasttext.cc/>

Week 10 - RNN, 단어를 컨베이어 벨트처럼 한 개씩

문장들을 순차적으로 처리하는 방식의 모델 : **RNN** (Recurrent Neural Network)

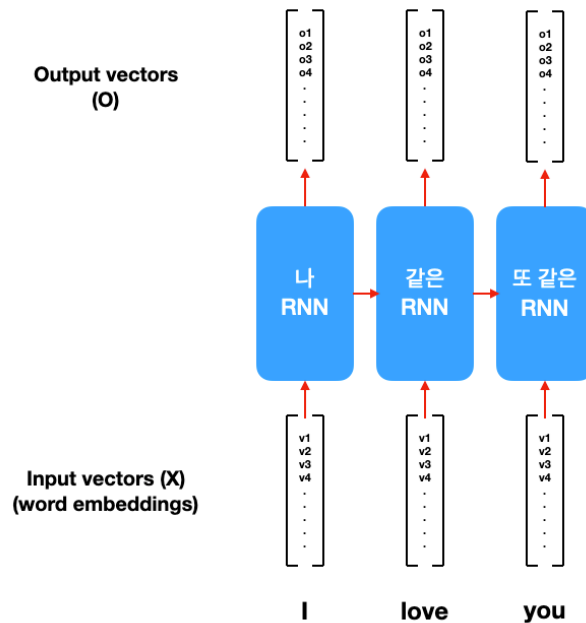
RNN → time series modeling에 많이 쓰이는 모델 (time series : 시간에 따라 변하는 변수)

input → hidden state → output

$N \times 1$ column vector로 된 단어들이 순차적으로 input으로 들어가고, output이 하나씩 생성됨

hidden state는 각 timestep마다 이어져 input들 간의 상관관계를 저장할 수 있는 메모리

⇒ 컨베이어 벨트에 넣는 것 처럼 보임



RNN 응용

- [Long Short Term Memory \(LSTM \)](#)
 - 모델 안에 여러 gate를 추가해 input, output, memory 간에 더 섬세한 제어가 가능
 - input, output, forget, update gate
 - 현재 RNN은 대부분 LSTM, 연산량이 多
- [Gated Recurrent Unit \(GRU \)](#)
 - 과거의 정보와 연계성을 더 잘 배우는 모델
 - LSTM과 비슷하지만 더 적은 gate로 연산량 ↓

Week 11 - 머신러닝 모델이 데이터를 공부하는 방법

머신러닝 모델은 cost function(=loss function)을 최소화하는 가중치를 찾아 학습
ex) stochastic gradient descent(SGD)

Week 12 - AI모델

데이터는 train data와 evaluation data로 나뉘야 함

evaluation data는 validation과 test data로 나눌 수 있음

validation → 하이퍼파라미터를 결정하는 용도

모델 학습을 잘 시키는 법

1. 데이터를 무작위로 나눔
2. validation과 test set의 데이터가 train set과 중복이 안되게
3. 세 데이터 set의 분포가 비슷하게 만든다
4. 데이터 수가 적으면 k-fold cross validation을 이용 → 데이터를 k만큼 나눠 k만큼 학습한 평균