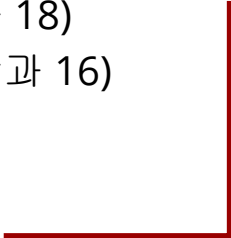




# 실전 NLP 분반

조민제(11기/경제학과 18)  
최정윤(12기/영어영문학과 16)



# Contents

---






- 1. Goals**
- 2. To whom?**
- 3. Curriculum**

# Goals

NLP 주요 모델과 워크플로우를 모두 이해하고,  
공모전/프로젝트를 진행해보자!







SOTA 논문 읽고 코드를 구현해보자!

Natural Language Processing

 Machine Translation  56 benchmarks 974 papers with code	 Language Modelling  19 benchmarks 956 papers with code	 Question Answering  63 benchmarks 854 papers with code	 Sentiment Analysis  50 benchmarks 583 papers with code	 Text Generation  47 benchmarks 413 papers with code
--	---	---	---	--

▶ See all 361 tasks

Transformers

METHOD	YEAR	PAPERS
 BERT	2018	1658
 Transformer	2017	1472
 RoBERTa	2019	169
 GPT-2	2019	137
 XLNet	2019	75
 GPT	2018	46



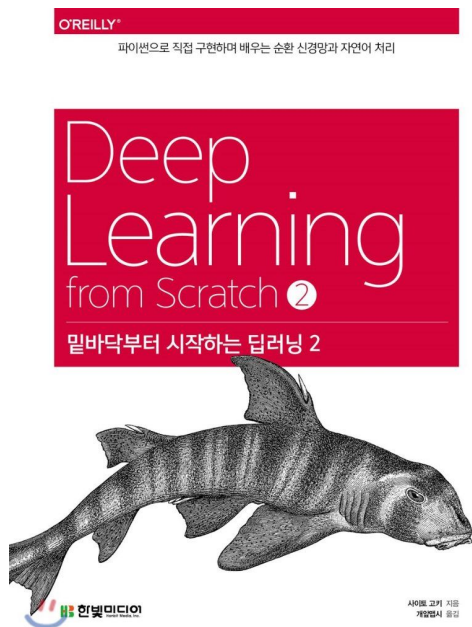
# To whom?

---

1. 함께 실력을 쌓은 후 **NLP 프로젝트**/공모전 및 **SOTA 논문**을 공부해보고 싶으신 분
2. 기초 딥러닝 지식이 있으신 분 (cnn,rnn..등 딥러닝 모델 대략 이해한다)
3. 프로젝트로 간단하게 모델은 쌓아봤는데 아직도 잘 모르겠다...  
  
⇒ 원리를 제대로 이해하여 **직접 구현**해보고 싶으신 분

# Curriculum

교재(예정) : 밑바닥부터 시작하는 딥러닝2



병행하면 좋은 참고 자료들

- Kaggle project 코드 리뷰
- Github에 올라온 관련 코드 리뷰
- Stanford CS224n 강의자료 (논문, 과제)
- paperswithcode 주요 논문

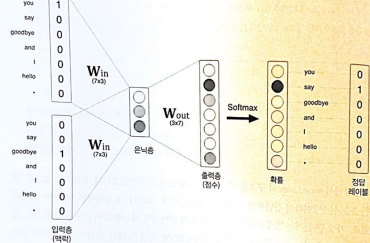
# Curriculum

## 스터디 진행 방식

### 4.1 word2vec 개선 ①

그럼 알 장의 보습부터 시작합니다. 알 장에서는 [그림 4-1]과 같은 CBOW 모델을 구현했습니

그림 4-1 알 장에서 구현한 CBOW 모델



[그림 4-1]과 같이 알 장의 CBOW 모델은 단어 2개를 벡터로 사용해, 이를 바탕으로 하나의 단어 (타깃)를 예측합니다. 이때 입력 속 가중치( $W_{in}$ )와 행렬 곱으로 은닉층이 계산되고, 다시 출력 속 가중치( $W_{out}$ )와 행렬 곱으로 각 단어의 점수를 구합니다. 그리고 이 점수에 소프트맥스 함수를 적용해 각 단어의 출현 확률을 얻고, 이 확률을 정답 레이블과 비교하여 (정확

하는 교차 엔트로피 오차를 적용하여) 손실을 구합니다.

**WARNING.** 알 장에서는 맥락의 윈도우 크기를 1로 한정했습니다. 다시 말해 타깃 앞·뒤 한 단어만 사용

한 것입니다. 이번 장에서는 나중에 이런 크기의 맥락도 다룰 수 있도록 기능을 추가할 것입니다.

이 단락을 읽고 CBOW 모델의 구조를 이해하는 데 도움이 될 것입니다. 알 장에서는

### 4.3.1 CBOW 모델 구현

이번에도 CBOW 모델을 구현해볼 텐데, 알 장의 단순한 SimpleCBOW 클래스를 개선할 것

니다. 개선점은 Embedding 계층과 Negative Sampling Loss 계층을 적용하는 것이고, 나

아가 맥락의 윈도우 크기를 임의로 조절할 수 있도록 확장합니다.

개선된 CBOW 클래스의 모습은 다음과 같습니다. 우선 초기와 테스트부터 보시죠 (← ch04/

chow.py).

```
import sys
sys.path.append('.')
import numpy as np
from common.layers import Embedding
from ch04.negative_sampling_layer import NegativeSamplingLoss

class CBOW:
    def __init__(self, vocab_size, hidden_size, output_size, corpus):
        V, H = vocab_size, hidden_size

        # 가중치 초기화
        W_in = 0.01 * np.random.randn(V, H).astype('f')
        W_out = 0.01 * np.random.randn(V, H).astype('f')

        # 계층 생성
        self.in_layers = []
        for i in range(2 + window_size):
            layer = Embedding(W_in)  # Embedding 계층 사용
            self.in_layers.append(layer)

        self.loss = NegativeSamplingLoss(W_out, corpus, power=0.75, sample_size=5)

        # 모든 가중치와 기울기를 배열에 모은다.
        layers = self.in_layers + [self.loss]
        self.params, self.grads = [], []
        for layer in layers:
            self.params += layer.params
            self.grads += layer.grads

        # 인스턴스 변수에 단어의 분산 표현을 저장한다.
        self.word_vecs = W_in
```

Scanned with CamScanner

필요 시  
참고자료  
병행



### A Detailed Explanation of Keras Embedding Layer

Python notebook using data from [multiple data sources](#) · 80,654 views · 2y ago

beginner, deep learning, nlp, +1 more

In [14]:

```
embeddings=embeddings.reshape(-1,maxlen,8)
print("Shape of embeddings : ",embeddings.shape)
print(embeddings)
```

```
Shape of embeddings : (3, 12, 8)
[[[ 0.00501078 -0.02695217  0.00602702  0.00266289  0.03473682
      0.0097171  0.01049457  0.04075214]
 [ -0.01465576 -0.03764923  0.02566164 -0.03778331  0.00713614
      0.04891029 -0.04066588  0.04367645]
 [ 0.02802004  0.03197568 -0.03032377 -0.04005457  0.04845554
```

캐글 코드리뷰

### 1-2.Word2Vec

### Word2Vec-Skipgram(Softmax).py

1-2.Word2Vec

```
18 return random_inputs, random_labels
19
20 # Model
21 class Word2Vec(nn.Module):
22     def __init__(self):
23         super(Word2Vec, self).__init__()
24         # W and Wt is not Transpose relationship
25         self.W = nn.Linear(vocab_size, embedding_size, bias=False) # vocab_size > embedding_size Weight
26         self.Wt = nn.Linear(embedding_size, vocab_size, bias=False) # embedding_size > vocab_size Weight
27
28     def forward(self, x):
29         # X : [batch_size, vocab_size]
30         hidden_layer = self.W(x) # hidden_layer : [batch_size, embedding_size]
31         output_layer = self.Wt(hidden_layer) # output_layer : [batch_size, vocab_size]
32         return output_layer
33
```

깃헙 참고자료

### Efficient Estimation of Word Representations in Vector Space

word2vec  
논문발체

Tomas Mikolov  
Google Inc., Mountain View, CA  
tmikolov@google.com

Kai Chen  
Google Inc., Mountain View, CA  
kaichen@google.com

Greg Corrado  
Google Inc., Mountain View, CA  
gcorrado@google.com

Jeffrey Dean  
Google Inc., Mountain View, CA  
jeff@google.com

### Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We

밑바닥부터 시작하는 딥러닝2 발체

# Curriculum

	스터디 내용	
1주차	신경망 복습, 자연어와 단어와 분산 표현	↕ 밑바닥 딥러닝2 + 코드
2주차	word2vec, word2vec 속도 개선	
3주차	순환 신경망(RNN), 게이트와 추가된 RNN	
4주차	RNN을 사용한 문장 생성, 어텐션	
5주차	Transformer Variations(Transformer, Bert, Transformer-XL 등)	↕ sota 논문
6주차	GPT(Generative Pre-trained Transformer)	
7주차	XL-Net: Generalized Autoregressive Pretraining for Language Understanding	

\*대략적인 계획, 참여하시는 학회원들의 의견과 실력을 참고하여 효율적인 스터디 방향으로 조정 예정



# Q&A







# 감사합니다

겨울방학 동안 같이 NLP 공부해요>.<

