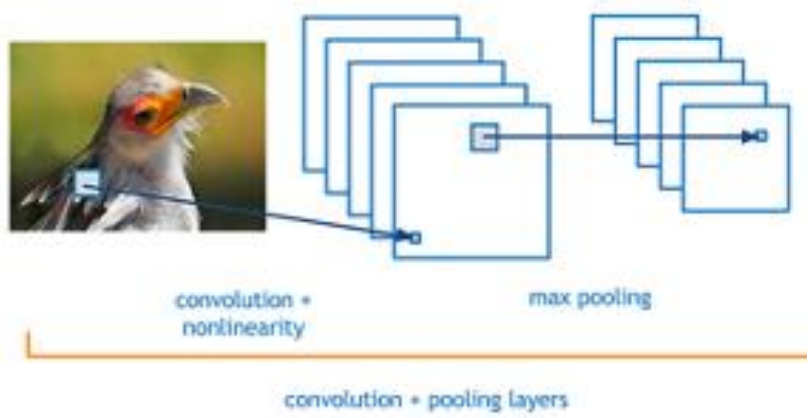

TRAIN AND TEST

1. ConvNets for NLP

Why CNN?

- Computer Vision 의 관점

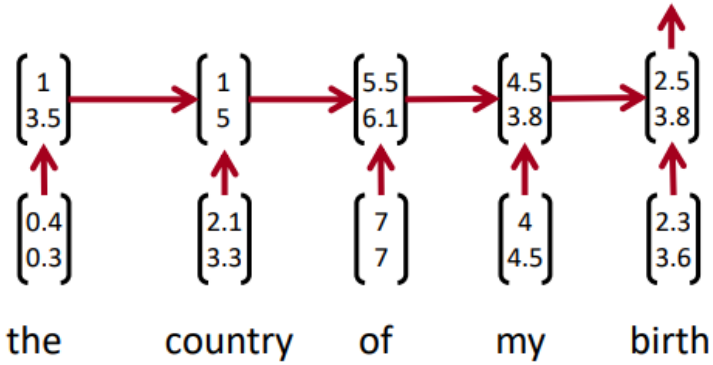


사람은 이미지를 인식할 때, 한 부분만 보는 것이 아니라 여러 부분을 함께 보고 인식!

마찬가지로 Neural Networks 도 여러 부분을 함께 보고, 해당 정보를 취합해서
이미지 인식을 학습하면 좋지 않을까?

Why CNN?

- NLP 의 관점

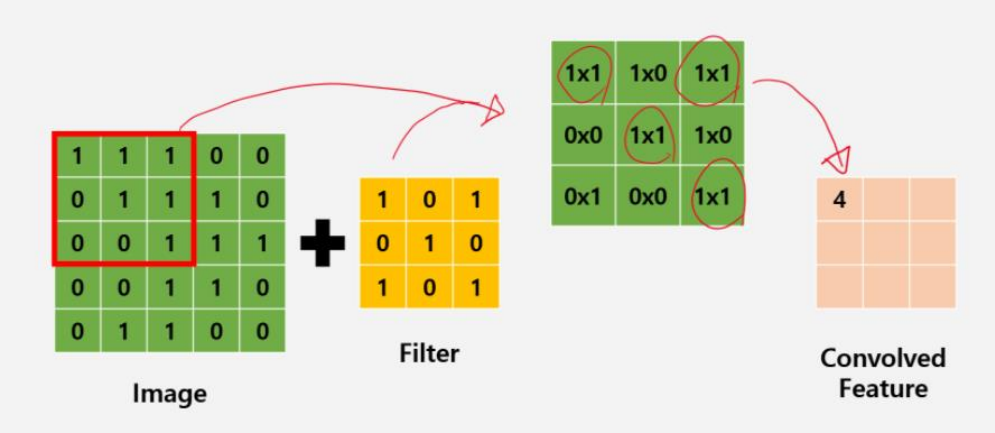


기존의 RNN을 이용한다면 아무리 LSTM을 사용한다고 해도, 마지막 단어에 너무 많은 정보가 들어가고 마지막 단어의 정보가 앞으로 전파되는 것이 어려움!

CNN을 활용하면 만들어진 word embedding을 바탕으로 구(n-gram) 단위 학습을 할 수 있어서 새로운 방식으로 사용 가능

CNN

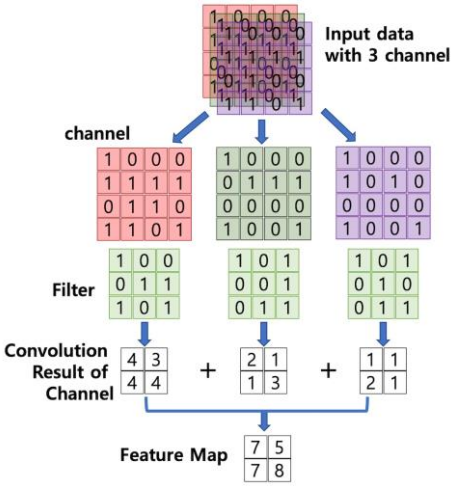
1. Convolution 과 Filter(weight, parameter...)



Convolution : Filter를 이동해 가면서 input feature의 각 자리의 값과 filter의 각 자리 값을 곱해서 더함 -> 새로운 feature 생성

Filter : CNN을 활용할 때, 학습되는 것!

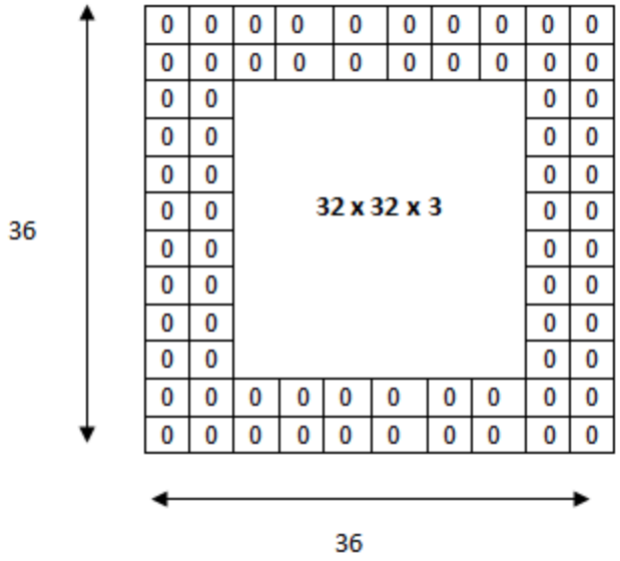
2. Channel



Channel : Feature map의 차원(?) 이라고 생각하면 편하다

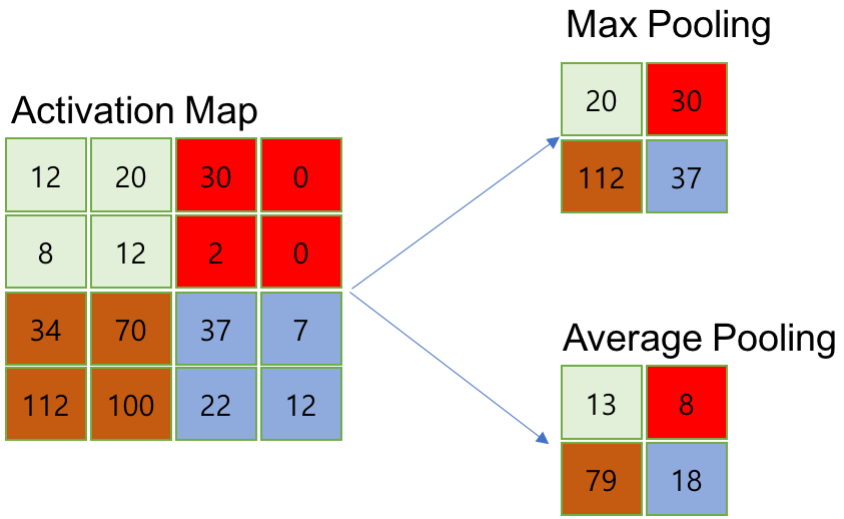
Channel의 수가 커질수록 더 복잡한 CNN 이 된다.

3. Padding



Convolution을 할수록 filter의 작용으로 인해 feature의 사이즈가 작아지는데,
이를 방지하기 위해서 임의의 값(보통 0) 으로 채워 넣어서 convolution을 하는 것.

4. Pooling



Feature map에서 중요한 부분만 남기는 방식을 통해 사이즈를 줄이는 것.

CNN with NLP

- 일반적으로 NLP 에서는 구 단위의 정보를 얻기 위해 1D convolution을 활용
- 특정 n개의 단어를 함께 convolution 할 수 있는 filter를 통해 CNN을 학습한다.
- 나머지 부분은 이미지처리의 CNN과 비슷한 프로세스로 진행됨.

CNN with NLP

Example

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

CNN with NLP

- CNN 응용

  1. Pooling over time
  2. Stride 활용
  3. Local Pooling
  4. k-max pooling
  5. Dilation

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

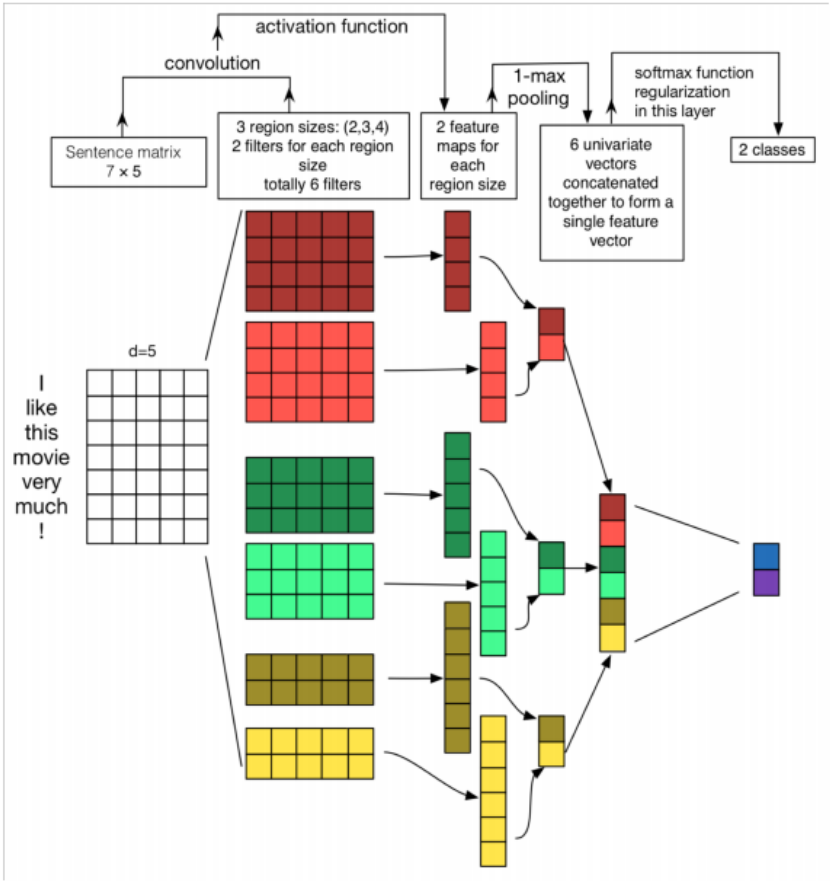
1,3,5	0.3	0.0
2,4,6		
3,5,7		

2	3	1
1	−1	−1
3	1	0

1	3	1
1	−1	−1
3	1	−1

Dilation

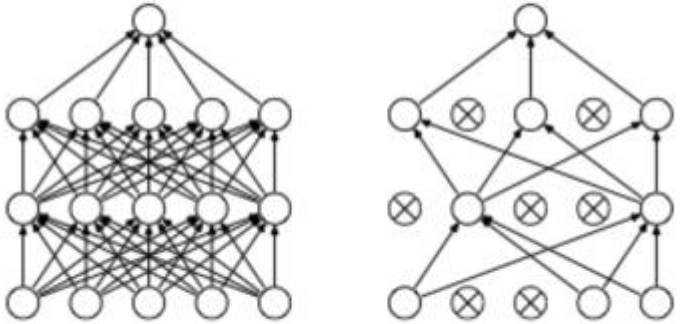
CNN for Sentence Classification



Techniques

Dropout

Dropout: A Simple Way to Prevent Neural Networks from Overfitting [Srivastava et al. 2014]

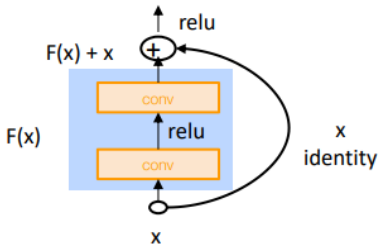


(a) Standard Neural Net (b) After applying dropout.

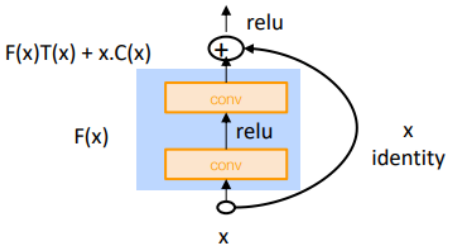
- Neural Networks 의 학습 과정에서 각 batch 마다 randomly perceptron을 선택해서 학습.
- 최종 과정에서 모든 perceptron을 이용하여 예측
- 이 과정을 통해 overfitting을 막을 수 있다.

Techniques

Gated units used vertically



Residual block
(He et al. ECCV 2016)



Highway block
(Srivistava et al. NeurIPS 2015)

- 특정 layer를 학습할 때 output 과정에서 input 값을 사용한다
- Input의 정보를 잃지 않기 위해서 사용!
- 일반적으로는 skip connection이 주로 사용되고, gate를 활용한 방식도 종종 사용됨.

Techniques

Batch Normalization

- 각 batch 마다 layer에서 나오는 output을 정규화 해주는 기법.
- 모델의 학습 결과가 feature에 너무 민감해지는 것을 막아준다.
- Parameter Initialization에 민감해지는 것을 막아준다.

2. Information of Part of Words

Character-level Models

- 지금까지는 대부분 word-level model을 배움
  ex) NMT : 나는 학생이다 -> I am a [?]

- Word-level model은 형태소에 대한 이해가 필요하고, 엄청 대용량의 vocab 가 필요하다.

- 만약 대용량의 vocab가 없다면 OOV(out of vocabulary) 문제가 발생함.
  ex) Covid-19 is virus. -> [unk] 는 바이러스다.

- 아무리 큰 vocab을 사용한다고 해도 신조어 등의 문제는 해결하지 못함.

Character-level Models

- 이런 character-level model의 문제를 해결하기 위해 나온 것이 character-level model!
- 첫번째 방법

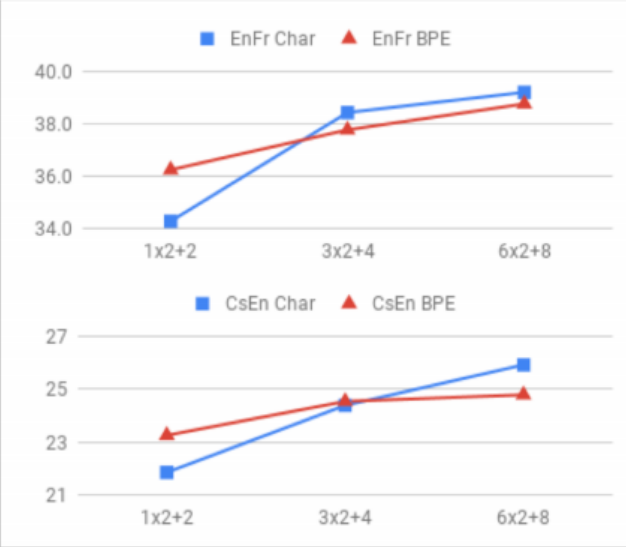
  각 character 마다 embedding을 두고 character embedding을 바탕으로
  word embedding을 만듬.
  ex) b : [0.3, 0.2], o : [0.7, 0.1], y : [0.1, 1.3] -> boy : [0.3, 0.2, 0.7, 0.1, 0.1, 1.3]

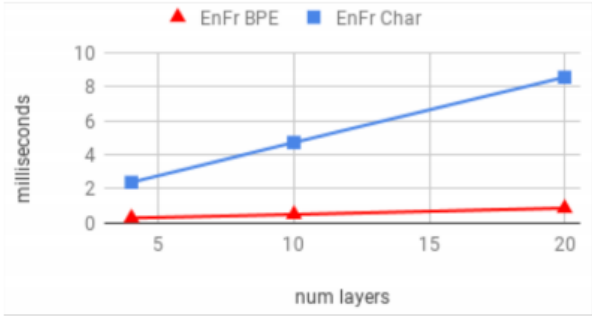
  이 방식을 이용하면 oov 문제를 해결할 수 있음.
  ex) covid : c + o + v + I + d

- 두번째 방법

  그냥 character embedding을 사용하여 모든 task 해결
  ex) 나는 학생이다 -> I am a studen[?]

Character-level Models





- 언어에 따라 성능이 많이 좋을 수도, 그냥저냥 일수도 있음
- 대신 시간이 엄~청 오래 걸림!

Subword Models

BPE (Byte Pair Encoding)

- Character-level encoding을 개선시킨 기법

Dictionary

5 l o w
2 l o w e r
6 n e w **est**
3 w i d **est**

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, est, **lo**

Ex) lowest 의 경우...

기존의 dictionary를 사용하면 : OOV!
Character level encoding을 사용하면 : l + o + w + e + s + t

BPE를 사용하면 : lo + w + est --→ 비교적 간단하다!

Subword Models

Wordpiece / Sentencepiece

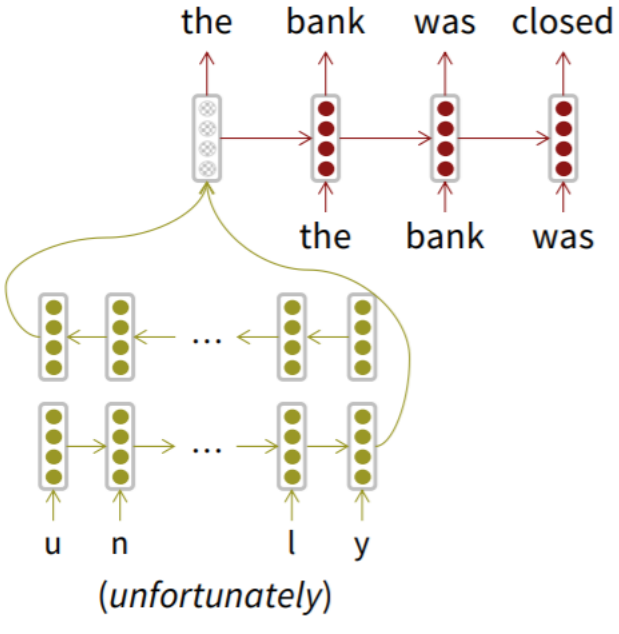
- Wordpiece : BPE 기반 + 단순 등장횟수로 병합하는 것이 아니라, 확률기반으로 병합.
  BPE 처럼 character 단위로 합치는 것이 아니라 단어 안에서 tokenize가 실시
- Sentencepice : Wordpiece와 비슷하나, 공백을 처리하는 기술이 다름.

Wordpiece
https://arxiv.org/pdf/1609.08144.pdf

Sentencepiece
https://arxiv.org/pdf/1808.06226.pdf

Subword Models

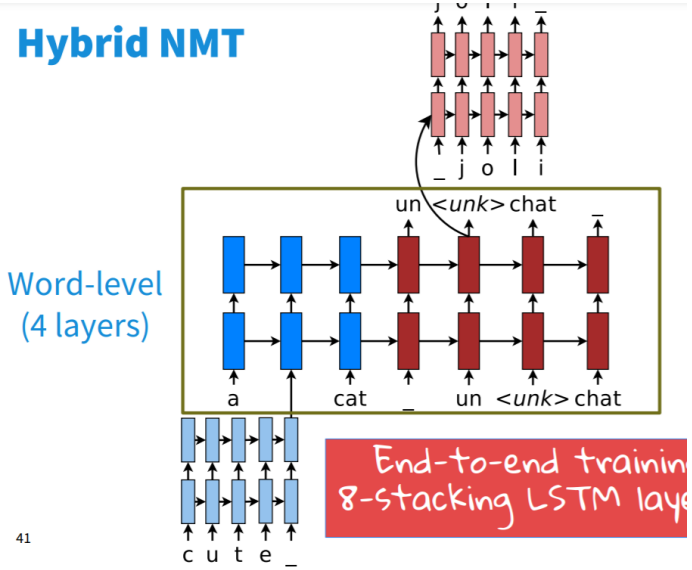
Character-based LSTM



- Word embedding을 만들 때 character 단위 bi-LSTM를 돌아서 만들어냄!
- 만들어낸 word embedding을 통해 RNN Language Model 생성

Subword Models

Hybrid NMT



- 일반적으로는 word-level NMT
- 필요시에 character-level bi-LSTM 사용!
  -> 불필요한 시간 낭비를 줄이면서 <UNK>를 처리할 수 있음!

Subword Models

Fasttext

- Word2vec을 subword로 나눠서 진행한 것.
- 같이 등장한 subword를 보고 각 subword embedding을 만들어 냄
- 만들어낸 subword embedding을 통해 word embedding을 생성!