8장. 텍스트 분류

2022-11-16

8.1 들어가며

NLP Lab
Natural Language Processing

- Text Classification : 텍스트로 이루어진 문서를 입력으로 받아 사전에 정의된 클래스 중에 어디에 속하는지 분류하는 것

문제	클래스 예
감정 분석	긍정, 중립, 부정
스팸 메일 탐지	정상, 스팸
사용자 의도 분류	명령, 질문, 잡담 등
주제 분류	각 주제
카테고리 분류	각 카테고리

- 텍스트를 입력으로 받아 불연속적인 값으로 출력
  - 주식의 오름세를 예측하는 문제와 비슷

한림대학교 HALLYM UNIVERSITY

8.4 RNN 활용하기

- **RNN**은 각 time-step의 단어를 입력으로 받아 은닉상태 업데이트

$$h_t = f_\theta(x_t, h_{t-1})$$

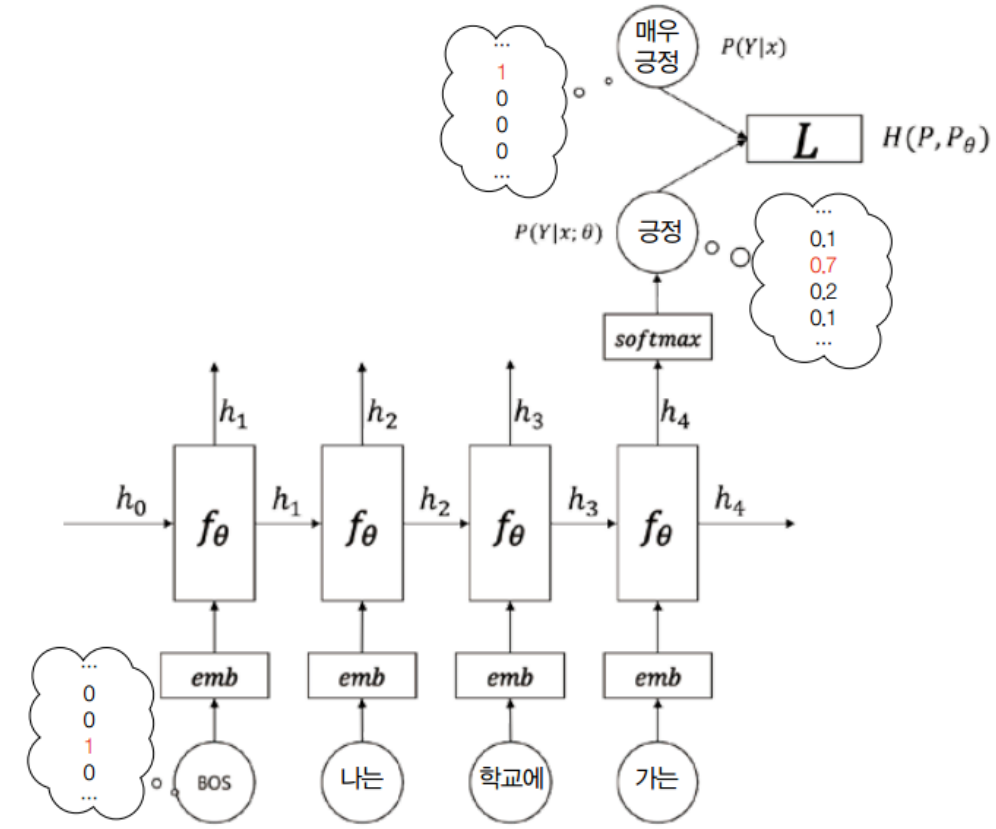
- n개의 **단어**로 이루어진 문장 x가 있을 때 RNN에 피드 포워드 하면 **n개의 은닉상태**

- **마지막 은닉 상태**로 텍스트의 클래스 분류

$$\hat{y} = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \; P(\mathrm{y} \mid x; \theta)$$

$$\text{where } P(\mathrm{y} \mid x; \theta) = h_n = f_\theta(x_n, h_{n-1}) \text{ and } x = \{w_1, w_2, \cdots, w_n\}$$

8.4.1 아키텍처 내부 살펴보기



▶ RNN의 마지막 time-step의 출력을 사용할 경우

$$x \sim P(\mathbf{x})$$

$$\text{where } x = \{w_1, w_2, \cdots, w_m\} \text{ and } w_i \in \{0,1\}^{|V|} \text{ and } |w_i| = 1$$

$$\text{Thus, } |x_{1n}| = (n, m, |V|)$$

$$\text{where } x_{1n} = [x_1, x_2, \cdots, x_n] \text{ and } n = \text{batch\_size}$$

RNN 활용

- 입력의 **원-핫 벡터**를 위치 인덱스로 기억, **Embedding layer** 통과

$$|x_{1:n}| = (n, m, 1) = (n, m)$$

$$\tilde{x}_{1:n} = \text{emb}_\theta(x_{1:n})$$
$$|\tilde{x}_{1:n}| = (n, m, d) \text{ where } d = \text{word\_vec\_dim}$$

- **초기 은닉 상태**와 단어 Embedding Tensor를 **RNN**에 통과

$$h_t = \text{RNN}_\theta(x_t, h_{t-1})$$
$$\text{where } |x_t| = (n, 1, d),\ |h_t| = (n, 1, h) \text{ and } h = \text{hidden\_size}$$

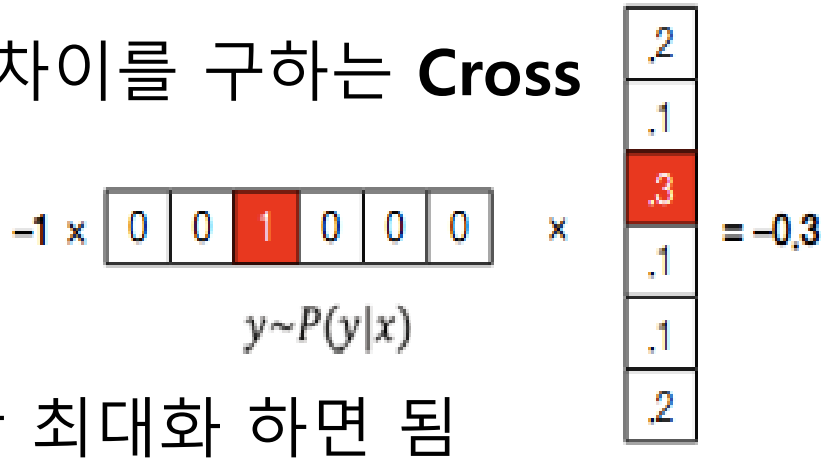
$$H = RNN_\theta(x_{1:n}, h_0)$$
$$\text{where } H = [h_1; h_2; \cdots; h_m] \text{ and } |H| = (n, m, h)$$

- 그러면, 모든 time-step에 대한 출력과 **마지막 은닉 상태** 반환 $h_m = H[:, -1]$
- 마지막 time-step으로 **softmax**를 통과하여 **이산확률 분포** 표현

$$\hat{y} = \text{softmax}(h_m \cdot W + b)$$
$$\text{where } |\hat{y}| = (n, |\mathcal{C}|),\ |h_m| = (n, 1, h),\ W \in \mathbb{R}^{h \times |\mathcal{C}|} \text{ and } b \in \mathbb{R}^{|\mathcal{C}|}$$

RNN 활용

- 이렇게 구한 확률 분포 예측 답 y^과 실제 답 y의 차이를 구하는 **Cross Entropy loss 최소화**



- y_i는 원-핫 벡터이므로 1인 인덱스의 log 확률 값만 최대화 하면 됨
- softmax 수식에 따라, 다른 인덱스의 확률 값 감소

- 즉, **Negative Log Likelihood (NLL)를 최소화**하는 것과 같음

- Softmax + CE loss == LogSoftmax + NLL loss

$$\mathcal{L}(\hat{y}, y) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \hat{y}_i$$

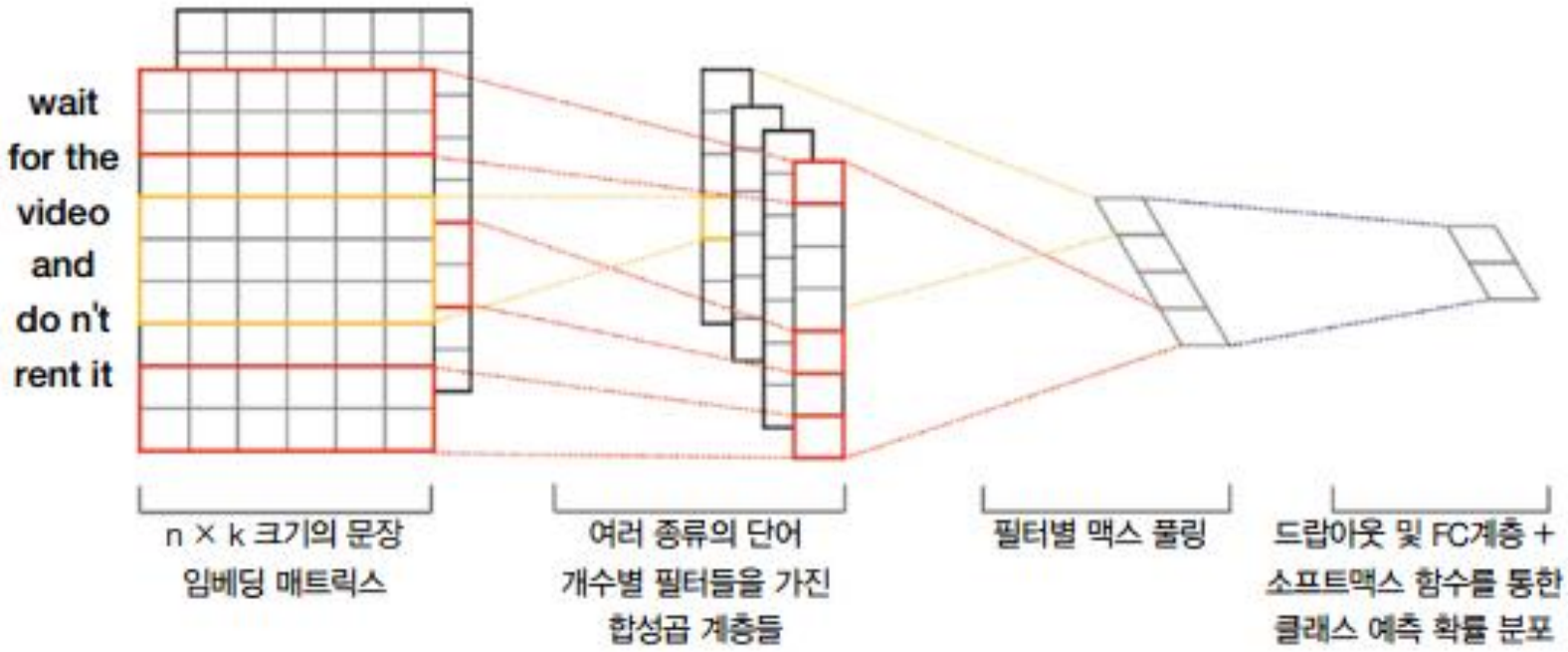
$$\theta \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}(\hat{y}, y)$$

8.5 CNN 활용하기

- RNN에 국한된 자연어처리
  - 텍스트 문장은 여러 단어로 구성
  - 문장의 길이가 상이
  - 문장 내 단어들은 같은 문장 내의 단어에 따라서 영향을 받음
  - w_t 는 이전 단어들 w_1, …, w_{t-1} 에 의존
  - 시간 개념이 도입되므로 RNN 사용이 불가피하다고 여겨짐

8.5 CNN 활용하기

- 자연어 처리에서는 주변 단어에 따라 서로 영향을 받기 때문에 RNN에 국한되어 있었지만, 합성곱 연산을 이용한 CNN으로도 구현



▶ CNN을 활용한 텍스트 분류 아키텍처[30]

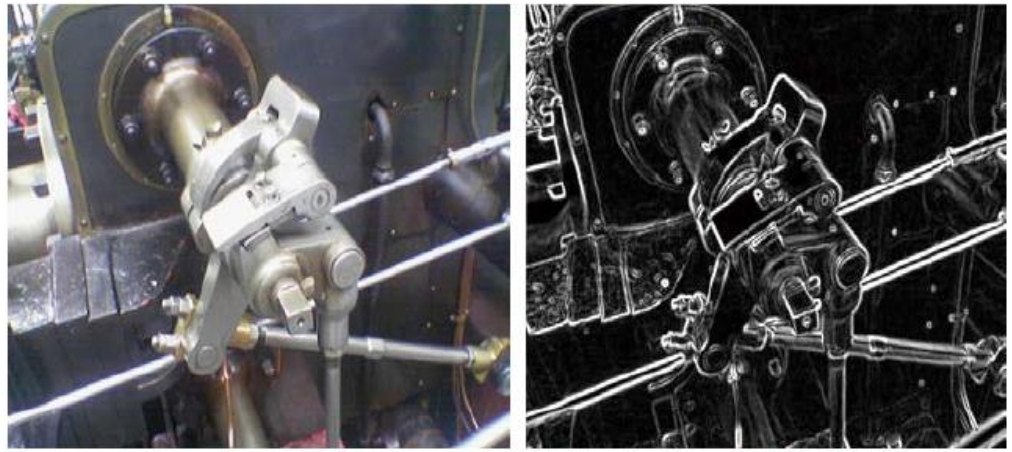
8.5.1 합성곱 연산

- 합성곱 필터 (convolution filter)
  - CNN 의 목적이 합성곱 필터의 자동 구성을 위한 학습임
  - 주어진 이미지에서 윤곽선을 찾기 위한 합성곱 필터

−1	0	+1
−2	0	+2
−1	0	+1

+1	+2	+1
0	0	0
−1	−2	−1

▶ 수직, 수평 윤곽선을 검출하기 위한 소벨 필터Sobel filter



▶ 소벨 필터 적용 전(左)과 후(右)[4]

8.5.2 합성곱 계층

- **CNN**은 주어진 이미지에 kernel이 차례로 **합성곱 연산** 수행
- 연산 후에는 입력보다 차원이 감소 ex. 6x6(input) 3x3(kernel) 4x4(output)

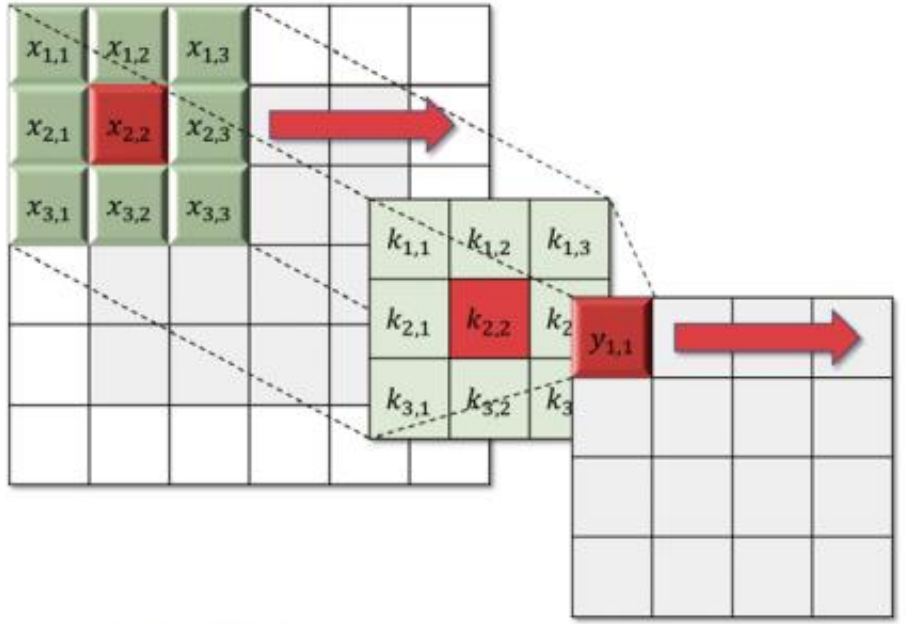
$$
\begin{aligned}
y_{1,1} &= \text{Convolution}(x_{1,1}\cdots,x_{3,3},\theta) \text{ where } \theta=\{k_{1,1},\cdots,k_{3,3}\} \\
&= x_{1,1}*k_{1,1}+\cdots+x_{3,3}*k_{3,3} \\
&= \sum_{i=1}^{3}\sum_{j=1}^{3} x_{i,j}*k_{i,j}
\end{aligned}
$$

- **출력 차원**의 크기 계산법
- 패딩을 통해 입출력 차원 유지 가능

$$\text{output\_size} = \text{input\_size} - \text{filter\_size} + 1$$

For example, if $y=\text{CNN}(x,k)$,

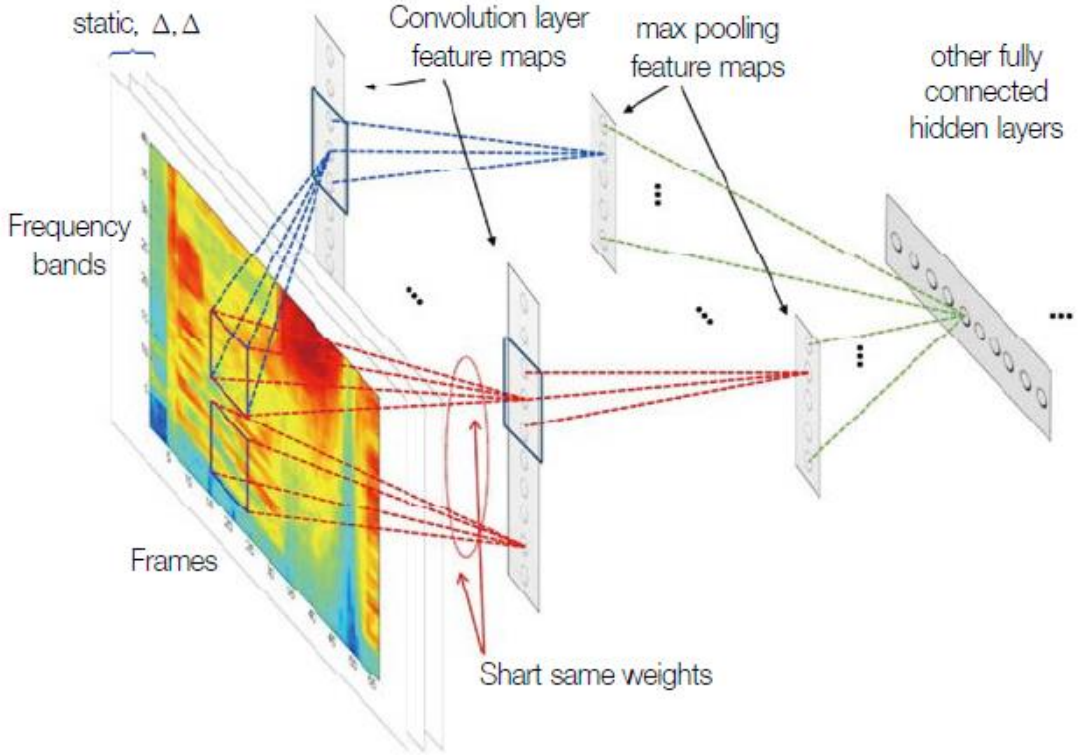
$y \in \mathbb{R}^{4\times4}$ where $x \in \mathbb{R}^{6\times6}$ and $k \in \mathbb{R}^{3\times3}$



▶ 합성곱 연산을 적용하는 과정

8.5.2 합성곱 계층

- 음성 분야에서도 효과
  - 푸리에 변환 (Fourier transform) 으로 2차원 시계열 데이터 추출
  - 합성곱 연산 사용



▶ 음성인식에 CNN이 적용된 예[1]

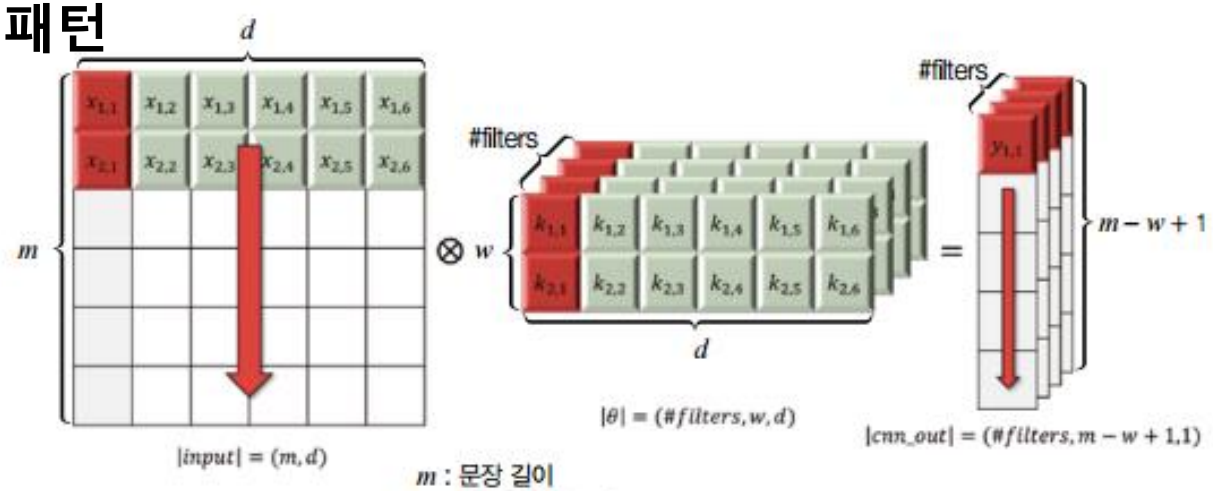
8.5.3 텍스트 분류에 CNN 적용하기

- 원-핫 벡터의 **인덱스**를 표현하는 단어 임베딩은 1차원 벡터
- 문장 내의 **모든 time-step**의 단어 임베딩 벡터를 합치면 2차원 행렬
- 그 후, CNN 합성곱 연산 수행
  - θ : 가중치 파라미터(filter)
  - **Input** : n(batch문장), m(time-step), d(벡터차원)
  - **filter**: **w개의 단어 조합에 대한 d차원의 패턴**

$$cnn\_out = CNN(input, \theta)$$

$$|input| = (n, m, d) = (n, 1, m, d)$$
$$|\theta| = (\#filters, w, d)$$
$$|cnn\_out| = (n, \#filters, m-w+1, \underbrace{1}_{d-d+1}),$$
$$\text{where } n = batch\_size.$$

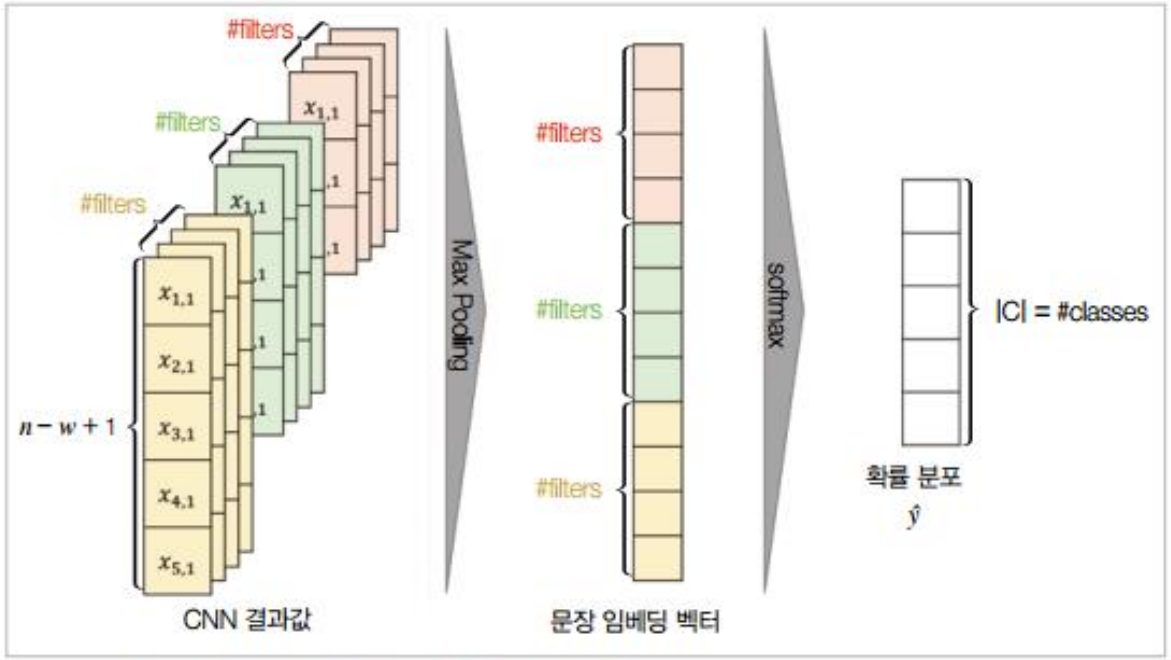
$$w \in \mathcal{W} = \{3, 4, 5\}$$



▶ 두 단어(토큰)의 패턴을 찾는 CNN

8.5.3 텍스트 분류에 CNN 적용하기

- CNN 계층의 결과값 : filter별 점수 (**각 특징 별 score**)
- 결과값을 **max pooling**하여 각 문장의 특징에 대한 최고 점수 추출
- 가변길이의 CNN 결과값을 **고정길이**로 변환 -> **문장의 임베딩 벡터**



▶ cnn_out에서 맥스 풀링을 통해 특징별 최고 점수를 뽑아내는 과정

- Softmax 함수를 통해 클래스별 확률분포 반환
- Cross Entropy 손실함수 사용

8.5.3 텍스트 분류에 CNN 적용하기

$$cnn\_out_i = CNN(input, \theta_i)$$

where $|\theta_i| = (\#filters, w_i, d)$ and $w_i \in \{w_1, w_2, \cdots, w_h\}$

$$pool\_out_i = max\_pooling(cnn\_out_i)$$

$$|pool\_out_i| = (n, \#filters)$$

$$pool\_out = [pool\_out_1; pool\_out_2; \cdots; pool\_out_h]$$

$$|pool\_out| = (n, h \times \#filters).$$

We can consider pool_out as sentence embedding vectors.