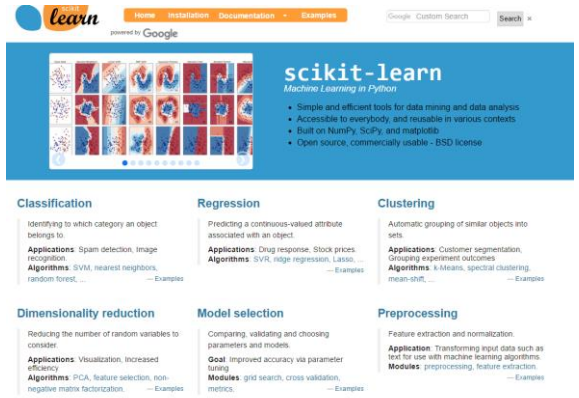
DAY4
기계학습 기반 자연어처리

실습준비

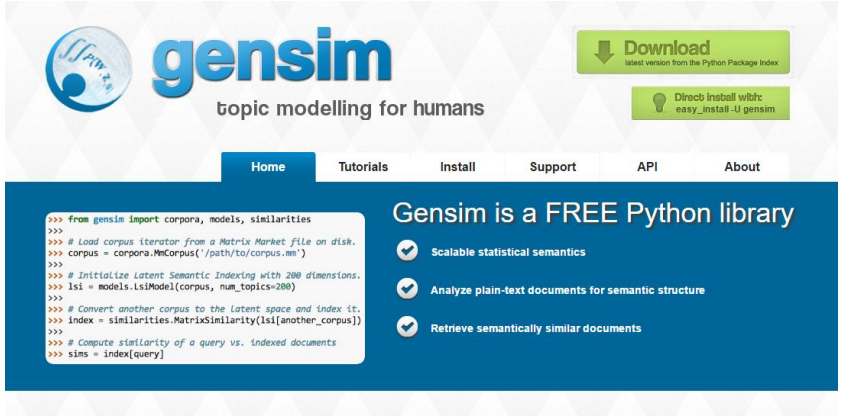
기계학습 툴/라이브러리

- Traditional Machine Learning toolkit
  - Support Vector Machines
    - SVM light (C++), libSVM (C++, Java, Matlab), …
  - **Scikit-learn (Python)**
    - 기계학습 라이브러리
  - **Gensim (Python)**
    - Topic Modeling, word2vec 라이브러리
- Natural Language toolkit
  - NLTK (Python), spaCy (Python), CoreNLP (Java), LinePipe (Java), …
- Deep Learning toolkit
  - Tensorflow (C++, Python), **PyTorch (Python)**, Caffee (C++), MXNet (C++), Keras, Chainer, …

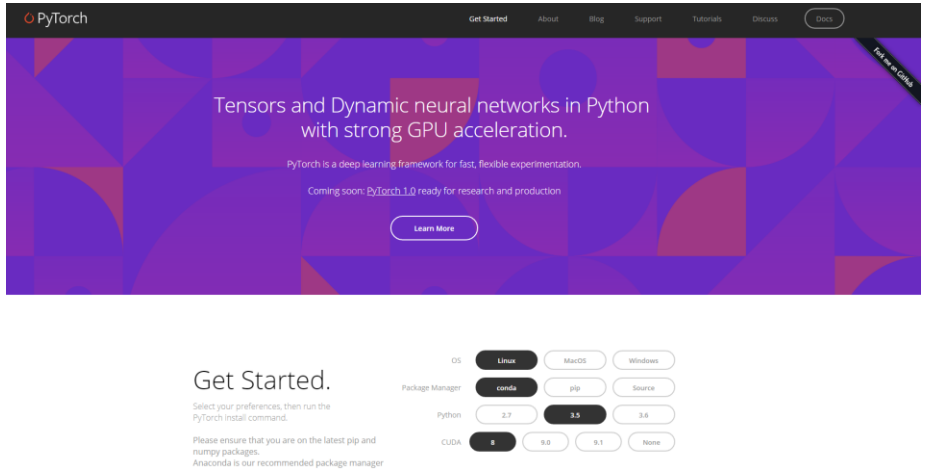
사용할 라이브러리



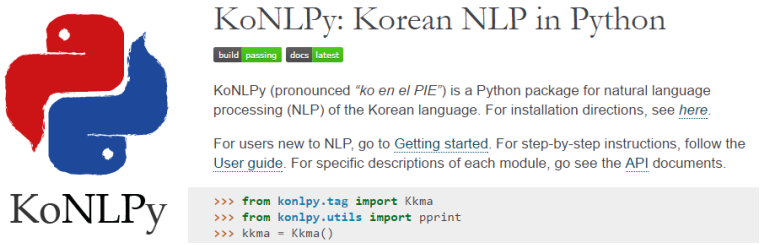
scikit-learn (scikit-learn.org)



gensim (radimrehurek.com/gensim/)



PyTorch (pytorch.org)



KoNLPy (http://konlpy.org)

실습 환경 및 코드

- Anaconda (Python 3.6 버전) 설치
- 실습 코드 및 리소스 다운로드

라이브러리 설치

- Anaconda(Python3.6) 설치

ANACONDA
Powered by Continuum Analytics®

simplify package management and deployment. Package versions are managed by the package management system conda.

- Anaconda prompt 실행 후 아래 라이브러리들을 설치
  - PyTorch
  - Scikit-learn
  - gensim
  - KoNLPy(http://konlpy.org/en/v0.4.4/install/)

설치 순서

#Python 3.6 버전 기준

```
pip install sklearn-crfsuite
pip install scikit-learn
pip install gensim
pip install six
pip install tqdm
pip install future

#PyTorch
conda install -c peterjc123 pytorch-cpu
pip install torchtext==0.2.3

#KonIpy 사용할 경우
pip install JPype1-0.6.3-cp36-cp36m-win_amd64.whl
pip install konlpy
```

테스트

- Anaconda Prompt 실행
- 실습 코드 디렉토리로 이동
- jupyter notebook 실행

```
import torch
import sklearn
import gensim
from konlpy.tag import Okt
okt = Okt()
print (okt.pos('이것은 책이다'))
```

강의 순서

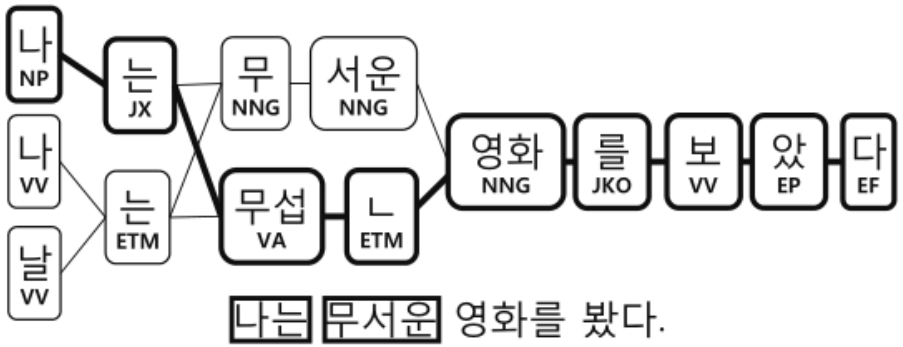
- DAY3 리뷰
- 단어 벡터화
- 문서 유사성과 분류
- 기계학습 기반 자연어처리
  - 단어 임베딩 (Word Embedding)
  - 감성분석
  - 의도분석, 개체명인식
  - 기계번역

DAY3 과정 리뷰

형태소 분석

- 한 어절 내에 있는 모든 형태소(사전 표제어)를 분리
  - 사무실에서부터였다고는 사무실+에서부터+이+었+다고는
  - 영장전담판사실 영장+전담+판사+실
  - **메시발롱도르몇번탔어? 메시+발롱도르+몇+번+타+았+어+?**

- 형태소들 간의 결합관계가 적합한지 검사
  - 체언과 조사의 결합 제약
  - 용언과 어미의 결합 제약
  - 조사, 어미 사이의 결합 제약

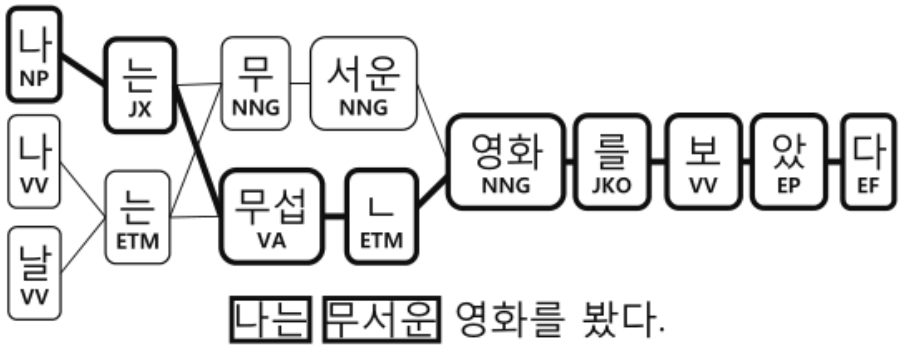


- 용언(동사, 형용사)의 원형 복원

- 복합어 분석, 미등록어 추정 (신조어, 외국어 등)

품사 태깅

- 형태소 분석의 모호성을 해결
- 주어진 문맥에 맞는 품사 태그를 선택하는 문제
- 이후 단계 모듈의 정확도를 좌우
- 통계적 기법에 의한 해결
  - 대량의 품사 태깅된 말뭉치를 구축
  - 통계적으로 확률이 높은 품사 태그를 선택



개체명 인식기 (Named Entity Recognition)

- 인명, 지명, 기관명, 제품명 등과 같은 고유명사나 명사구를 인식
  - 프리미어 리그에서 뛰고 있는 토트넘의 손흥민 선수는 ...
  - [프리미어 리그/축구리그]에서 뛰고 있는 [토트넘/축구팀]의 [손흥민/사람] 선수는 ...

감성 분석 (Sentiment analysis)

너무너무 좋아요 디자인도 좋고 처음에 랜탈할려고 했는데 그것보다도 훨 좋은 조건인거 같아요 필터만 일년에 한개씩 바꾸면 된다네요 생각보다 소음도 적어요 ^^

긍정

사전 기반 감성 분석 →

- 감성 표현 추출

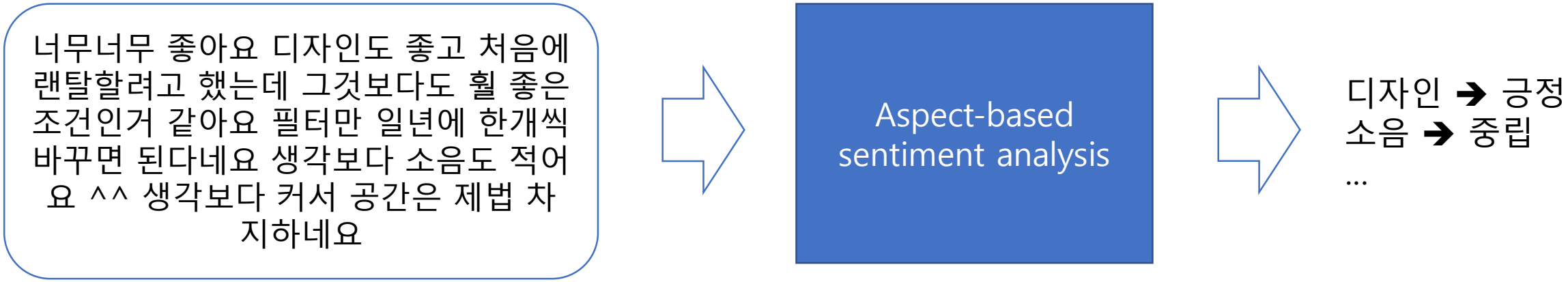
  너무너무 좋다(2), 디자인이 좋다(1), 훨씬 좋은 조건(2), 소음이 적다(1)

- 감성어휘 예

감성어휘	가중치
좋다	1
짱 좋다	2
너무 좋다	2
만족하다	1
가격이 사악하다	-1
불편하다	-1

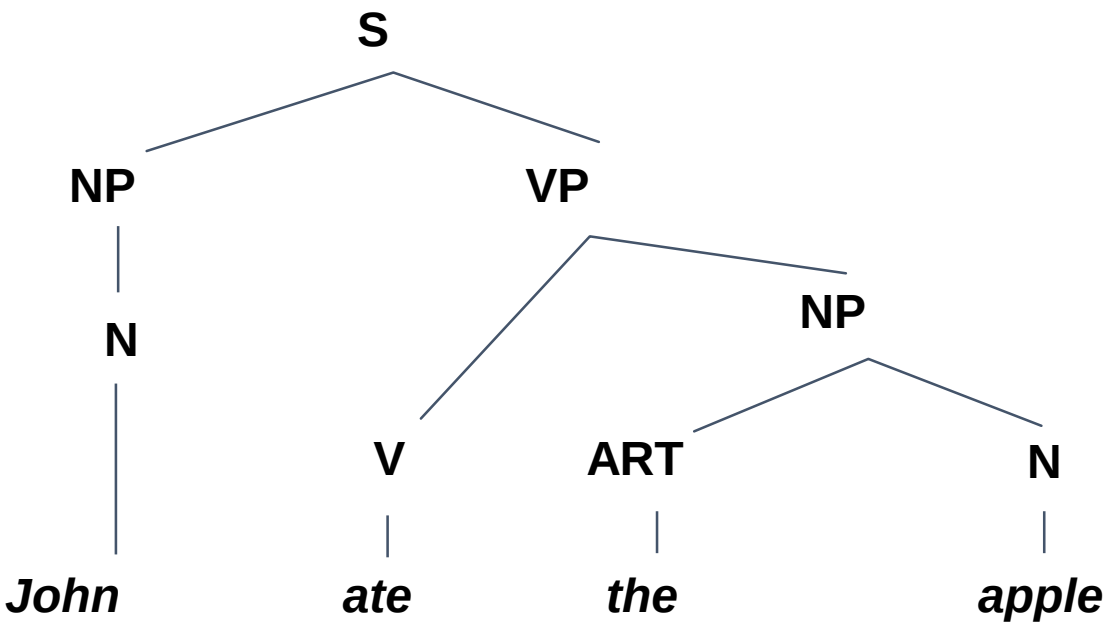
감성 분석의 향후 방향

- Aspect-based sentiment analysis

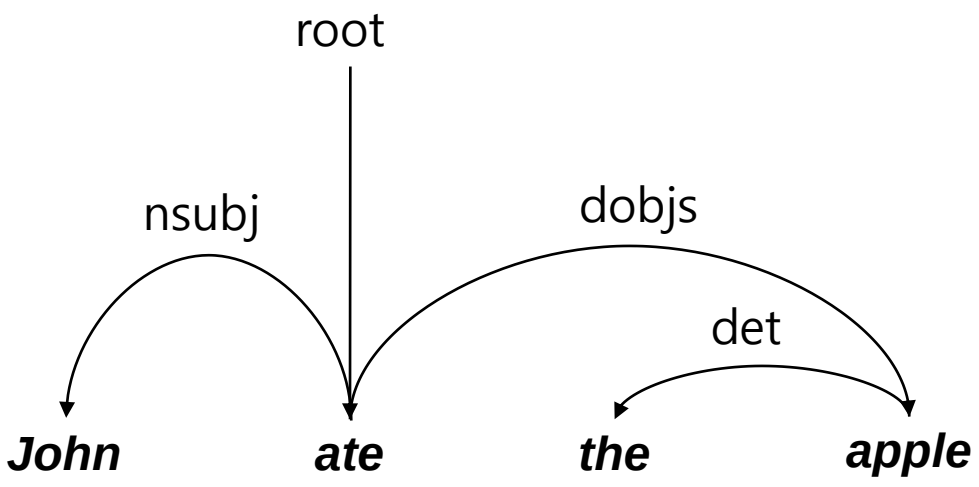


구문 분석 (Syntax Analysis)

- **문장의 구조를 분석**하여 계산 가능한 내부 형태로 표현하는 것



Constituency



Dependency

화용 분석 (Pragmatic Analysis)

- 문장이 실세계와 가지는 연관관계 분석
  - 실세계의 지식과 상식의 표현이 필요함
  - 지시, 화법, 간접화법 등의 분석

- 대명사의 지시 대상
  - "그것 좀 주문해 줘"
    ➔ "<치킨, 피자, 햄버거> 좀 주문해 줘"
  - "지난번에 주문했던 걸로 할게"
    ➔ "<2018_09_25>에 주문했던 걸로 할게"

- 상대방에게 행동을 요구하는 언어 행위, 간접적인 의사표현
  "거실이 좀 덥네~" (에어컨을 틀어줘)

기존 자연어처리 ≈ 언어자원 + 패턴기반

- 언어자원

형태소 사전

기분석 사전 (Pre-analyzed Dictionary)

감성어휘 사전

개체명 사전

+

- 패턴 (규칙 기반)

- **청킹 패턴**
  - **{명사}+ 명사**
  - **대명사 {명사}+**
- **형태소 결합 규칙**
  - **명사 + 격조사**
  - **명사 + 격조사 + 보조사**
  - **…**
- **IF 구문**

패턴기반 자연어처리

- 장점
  - 좋은 성능을 보여줌
    - **구축한 패턴이 정확하면**
    - 언어자원 정말 많으면
  - 즉각 반영
    - 특정 카테고리를 '추가/제거' 해야 할 경우
      - 지방선거 카테고리
- 단점
  - **리소스 구축 비용**
    - **시간 + 돈**
  - **새로운 도메인에 적용이 힘듦**
    - **도메인1 – 패턴1**
    - **도메인2 – 패턴2**
    - **…**
  - **패턴 유지관리 이슈**
    - **패턴 충돌**

문서 유사성과 분류

Document Similarity and Classification

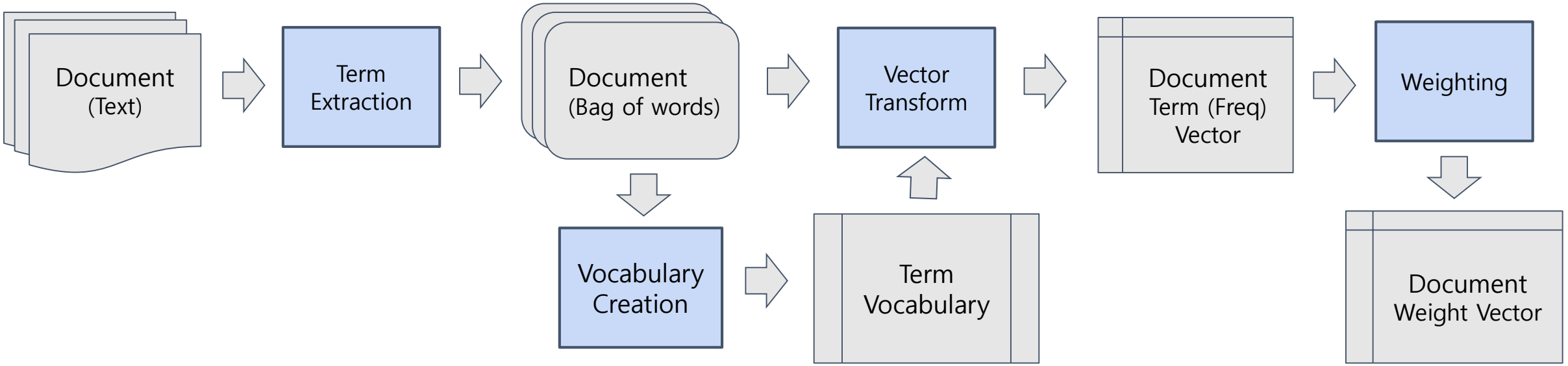
문서의 표현

- Bag of Words
  - 문서를 단어의 집합으로 간주한다.
  - 문서에 나타나는 각 단어는 feature로 간주되고
    단어의 출현 빈도에 따른 가중치를 얻는다.
- Feature Selection
  - 학습 문서에 출현한 용어(term)의 부분 집합을 선택하는 것
  - 사전의 크기를 줄여서 학습에 더 효율적인 분류기를 만든다.
  - Noise feature를 제거하여 분류의 정확도를 높인다.
  - WordNet 등 어휘 리소스를 활용하여 동의어, 상위어로 단어를 확장

문서의 표현

- 사람이 이해 하는 표현 → 기계가 이용할 수 있는 표현
- 의미적 표현→ 수학적 표현
- From Text To Weight Vector (가중치 벡터)
- Semantic Concept Space → Mathematical Space
- 의미 손실 가능성 vs 도구화

문서의 표현


Document
(Text)
Term
Extraction
Document
(Bag of words)
Vocabulary
Creation
Term
Vocabulary
Vector
Transform
Document
Term (Freq)
Vector
Weighting
Document
Weight Vector

문서의 표현

- 문서 (Document)
  - 문서는 단어로(Term, Word) 이루어져 있다.
  - 문서는 Term들의 의미 있는 나열이다.
- 예시

인공지능은 기계로부터 만들어진 지능을 말한다 ... 이 용어는 또한 그와 같은 지능을 만들 수 있는 방법론이나 실현 가능성 등을 연구하는 과학 분야를 지칭하기도 한다.

문서의 표현

Document :

인공지능은 기계로부터 만들어진 지능을 말한다. 이 용어는 또한 그와 같은 지능을 만들 수 있는 방법론이나 실현 가능성 등을 연구하는 과학 분야를 지칭하기도 한다.

Term Extraction

인공지능 기계 만들 지능 말하 용어 같 지능 만들 방법론 실현 가능성 등 연구하 과학 분야 지칭하

Term Ordering

가능성 같 과학 기계 등 만들 만들 방법론 분야 실현 연구하 **인공지능** 지능 지능 지칭하

문서의 표현

Document :

가가멜	1
…	
가능성	21
…	
등	1245
…	
지능	7229

가능성 같 과학 기계 등 만들 만들 방법론 분야
실현 연구하 **인공지능** 지능 지능 지칭하

Term ID

21 43 152 294 432 1245 2316 3214 3317 4234 4571 60
01 6025 7123 7229 7381 10231 10233 11033

문서의 표현

- Term Extraction
  - 문서를 Term 단위로 분해하는 것
  - 문서에서 Term을 추출하는 것
- 단위
  - 어절 (Token) : 띄어쓰기
  - 형태소 : 형태소 분석, Stemmer
  - 구문 : 명사구, 패턴
  - N-gram
    - Bigram, trigram
    - 음절, 형태소, 어절

Vocabulary Generation

- Document 집합에 있는 Term들을 사전화
- Filtering
  - Stop words : 조사, 어미, 관사, 접속사 등 고빈도 기능어 제거
  - 제외어 : 욕설, 비적합 단어 제거
- Document Frequency Count (DF)
  - 단어가 포함된 문서의 개수
- Ordering
- Term ID 부여

Stop Word List

- 너무 자주 출현하기 때문에 문서를 변별하는 feature로서 쓸모 없는 단어를 제외
- 대부분의 문장에 자주 등장하는 고빈도 단어들
- 대상 문서 집합에 출현한 용어를 빈도별로 리스팅하여 stop word list 생성
  예) a, an, and, are, as, has, in, is, it, was, were, will, …
- 해당 분야와 의미상 관련된 용어는 포함시켜야 함

Term Vocabulary

Term	ID	DF
가가멜	1	25
…		
가능성	21	2914
…		
등	1245	191
…		
지능	7229	1127
$Term_N$	N	D_k

Document Transformation

- Term Frequency Vector
  - Term -> ID
  - Out of Vocabulary Term 제거 : Vocabulary에 없는 단어들
  - 각 Term의 Frequency를 Count

가능성 같 과학 기계 등 만들 만들 방법론 분야 실현 연구하 인공지능 지능 지능 지칭하

Transformation

23:1, 29:1, 38:1, 52:1, 112:1, 231:2, 370:1, 391:1, 451:1, 530:1, 579:1, 689:1, 7229:2

Document Weighting

- Weight Vector
  - Weighting Scheme
    - TF, TF x IDF,
    - Probability
  - 모델에 따라 term에 부여되는 가중치가 달라짐

23:1, 29:1, 38:1, 52:1, 112:1, 231:2, 370:1, 391:1, 451:1, 530:1, 579:1, 689:1, 7229:2

Transform ation

Dense Vector < 0.0, 0.0, 0.23, 0.001. 0.6, ... 0.01, 0.72 , 0.324, 0.49>
Sparse Vector [ID113ee, 1:0.123, 235:0.643, ... 18932:0.0001, 23847:0.82]

IDF (Inverse Document Frequency)

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}$$

N : 문서집합 (collection)의 총 문서수
df : 문서 빈도 (document frequency)
tf : 용어 빈도 (term frequency)

term	df_t	idf_t
car	18,165	1.65
auto	6,723	2.08
insurance	19,241	1.62
best	25,235	1.50

Reuters 806,791 컬렉션에서 추출한 용어의 빈도와 IDF
(*최신 정보 검색론,* 안동언 외 공역, 2008, 교보문고)

TF-IDF 값

- TF와 IDF를 결합하여 각 용어의 가중치를 계산
- 문서 D에 나타난 용어 t에 부여되는 가중치

  TF-IDF (t, D) = TF x IDF
- 적은 수의 문서에 용어 t가 많이 나타날 때 가장 높은 값을 가진다 (이 문서들에 높은 식별력을 준다)
- 한 문서나 많은 문서들에 그 용어가 적게 나타나면 낮은 값을 가진다 (따라서 적합성이 뚜렷하지 않음)
- 사실상 모든 문서 안에 그 용어가 나타날 경우 가장 낮은 값을 가진다

TF-IDF 계산 문제

다음 표는 Doc1, Doc2, Doc3로 표기된 문서와 그 문서 내의 용어 빈도를 나타낸 것이다. 각 문서에 대해서, 용어들 car, auto, insurance, best의 TF-IDF를 계산하세요.
(IDF 값은 이전 슬라이드의 값을 사용)

Term	Doc1	TF-IDF	Doc2	TF-IDF	Doc3	TF-IDF
car	27		4		24	
auto	3		33		0	
insurance	0		33		29	
best	14		0		17	

(*최신 정보 검색론,* 안동언 외 공역, 2008, 교보문고)

TF-IDF 계산 문제 (답)

다음 표는 Doc1, Doc2, Doc3로 표기된 문서와 그 문서 내의 용어 빈도를 나타낸 것이다. 각 문서에 대해서, 용어들 car, auto, insurance, best의 TF-IDF를 계산하세요.
(IDF 값은 이전 슬라이드의 값을 사용)

term	Doc1	TF-IDF	Doc2	TF-IDF	Doc3	TF-IDF
car	27	44.55	4	6.6	24	39.6
auto	3	6.24	33	68.64	0	0
insurance	0	0	33	53.46	29	46.98
best	14	21	0	0	17	25.5

(*최신 정보 검색론*, 안동언 외 공역, 2008, 교보문고)

문서 유사성 - Vector Space 모델

- Term Vector Model
  - Document를 Term Space의 Vector로 가정
  - Document Vector간 유사도를 계산하여 유사성 비교

Salton, 1983

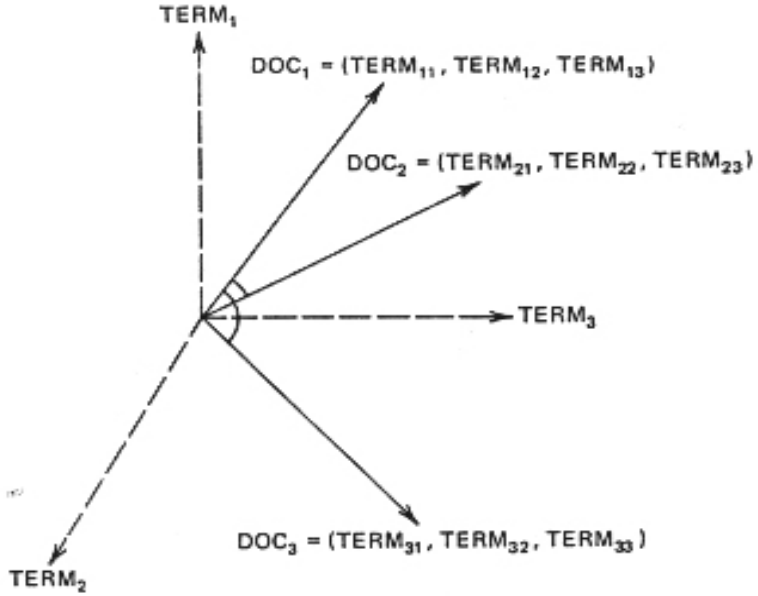
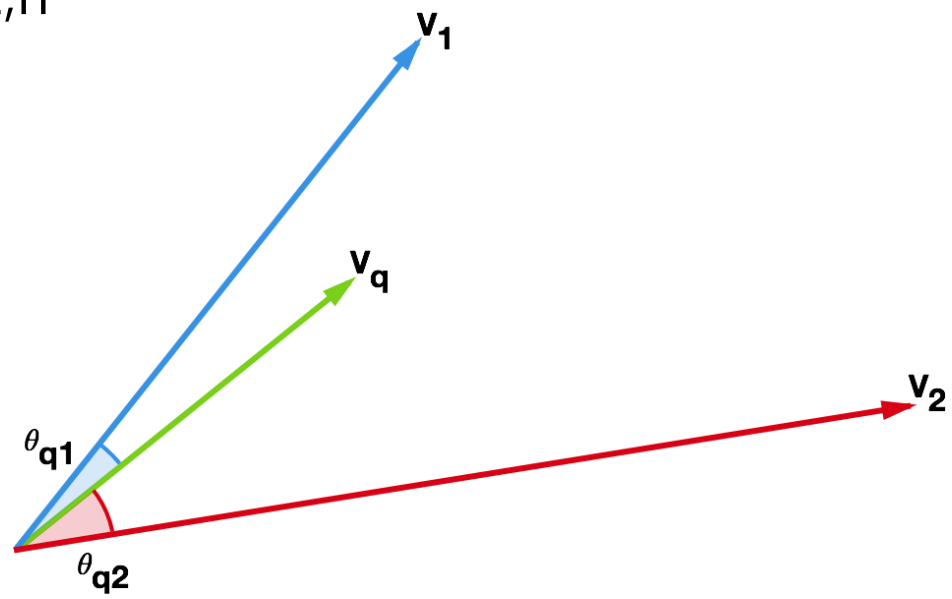


Figure 4-2 Vector representation of document space.

Cosine Similarity

- $D_1 = <t_{11}, t_{12}, t_{13}, \ldots t_{1,n-2}, t_{1,n-1}, t_{1,n}>$
- $D_2 = <t_{21}, t_{22}, t_{23}, \ldots t_{2,n-2}, t_{2,n-1}, t_{2,n}>$

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$



Cosine Similarity 계산 예

- 3 단어로 이루어진 문서 Doc1, Doc2가 있다. 각 문서 벡터의 숫자들은 단어 가중치를 의미한다. 질의문서 Q가 주어졌을 때, 각 문서에 대한 코사인 유사도를 계산해 보자.

$$Cosine(D_1, Q) = \frac{(0.5 \times 1.5) + (0.8 \times 1.0)}{\sqrt{(0.5^2 + 0.8^2 + 0.3^2)(1.5^2 + 1.0^2)}} = \frac{1.55}{\sqrt{(0.98 \times 3.25)}} = 0.87$$

$$Cosine(D_2, Q) = \frac{(0.9 \times 1.5) + (0.4 \times 1.0)}{\sqrt{(0.9^2 + 0.4^2 + 0.2^2)(1.5^2 + 1.0^2)}} = \frac{1.75}{\sqrt{(1.01 \times 3.25)}} = 0.97$$

Doc1 = [0.5, 0.8, 0.3]
Doc2 = [0.9, 0.4, 0.2]
질의문서 Q = [1.5, 1.0, 0]

질의에서 높은 가중치를 가지는 첫번째 단어에 대해 Doc2가 높은 가중치를 갖고 있으므로 -> 코사인 유사도가 더 높음

(최신정보검색론, 임해창 외 공역, 2012, 휴먼싸이언스)

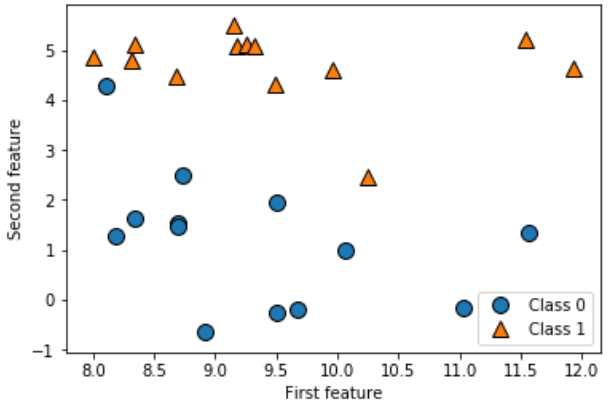
문서 분류 알고리즘

- Naïve Bayes Classifier
- *k*-Nearest Neighbors
- Support Vector Machine
- 최근 CNN, RNN이 더 좋은 성능 보임

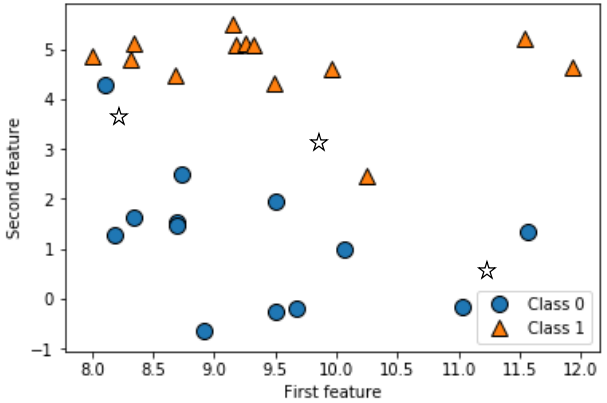
문서 분류의 활용

- 대량의 문서를 자동 분류
- 컨텐츠 필터링 (spam, adult content)
- 감성 분류
  - 영화 리뷰, 상품 리뷰
- 이메일 분류
  - 예) Gmail의 기본 메일함, 스팸, 소셜, 프로모션(광고)
  - 유저가 분류한 하위 분류함에 메일을 자동 분류

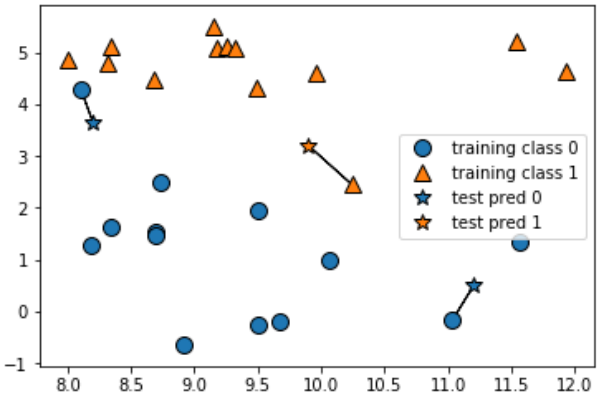
k-최근접 이웃 알고리즘 (k-Nearest Neighbors)



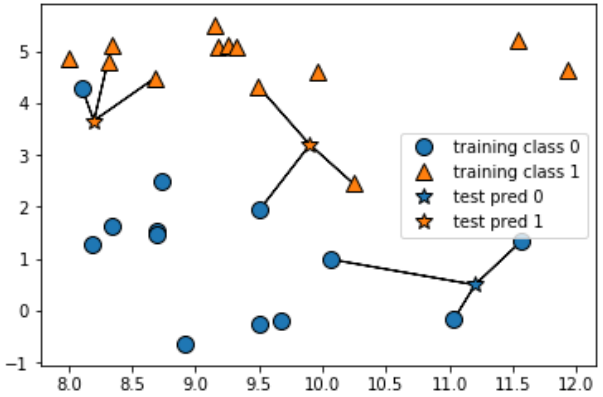
학습 데이터



테스트 데이터



$k = 1$

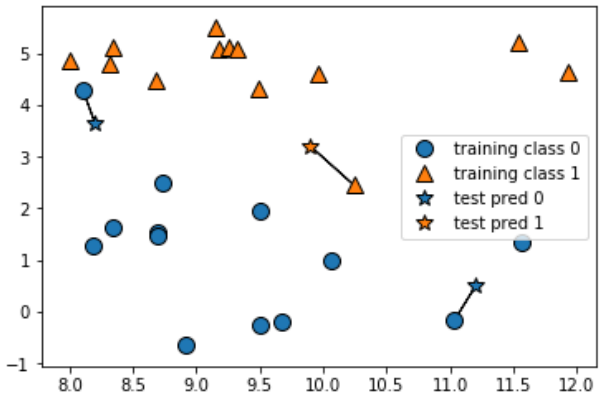


$k = 3$

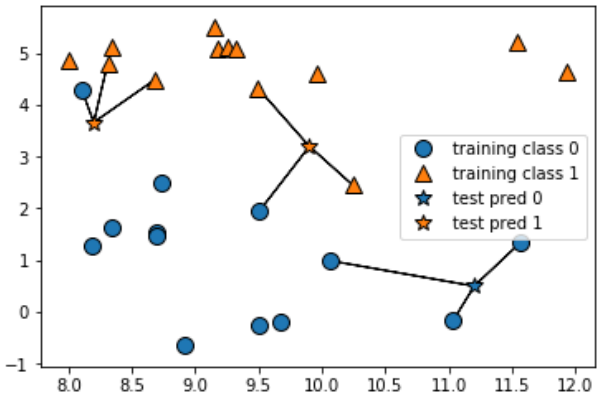
k-최근접 이웃 알고리즘 (*k*-Nearest Neighbors)

- 학습
  - 주어진 학습 데이터들을 **저장** ➔ Memory-based Learning
- 분류
  - 분류하고자 하는 데이터와 가장 가까운 k개의 데이터와 비교

- Distance-Weighted *k*-NN
  - 가까운 데이터를 더 고려
  - Weight = 1 / distance



k = 1



k = 3

나이브 베이즈 (Naïve Bayes Classifier)

- Simple, but relatively strong in text classification
- **Given a document *x*, assign the category *c* to *x***

$$P(c \mid x)$$

- Assumptions
  - Each instance **x** is a **conjunction of attribute values** a_1, a_2, …, a_n.
  - Attributes are **conditionally independent** of target class.

나이브 베이즈 (Naïve Bayes Classifier)

$$c^* = \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid x)$$

Assumption 1

$$= \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid a_1, a_2, \ldots, a_n)$$

Bayes rule

$$= \underset{c_j \in C}{\operatorname{argmax}} \frac{P(a_1, a_2, \ldots, a_n \mid c_j) P(c_j)}{P(a_1, a_2, \ldots, a_n)}$$

분모는 항상 같은 값

$$= \underset{c_j \in C}{\operatorname{argmax}} P(a_1, a_2, \ldots, a_n \mid c_j) P(c_j)$$

Assumption 2

$$\boxed{= \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_i P(a_i \mid c_j)}$$

나이브 베이즈 예제

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

- 사전 확률 (prior) :
전체 5건 문서 중 부정 3건, 긍정 2건
P(C*negative*)= 3/5 ; P(C*positive*) = 2/5
- 우도 (Likelihood) :
1) 부정 범주 문서에 나타난 전체 단어수 14개. 이 가운데 각 단어가 몇번 나타났는지 카운트해서 우도 계산
P(predictable | Cn) = 1/14
P(no | Cn) = 1/14
P(fun | Cn) = 0/14 → 스무딩 필요
2) 긍정 범주 문서에 나타난 전체 단어수 9개
P(predictable | Cp) = 0/9
P(no | Cp) = 0/9
P(fun | Cp) = 1/9

그렇다면 Test 문서의 분류는? (사후 확률 Posterior)
두 값 중 큰 쪽으로 분류
부정일 확률 :
P(Cn) * P(predictable | Cn) * P(no | Cn) * P(fun | Cn)

긍정일 확률 :
P(Cp) * P(predictable | Cp) | P(no | Cp) * P(fun | Cp)

https://ratsgo.github.io/machine%20learning/2017/05/18/naive/

나이브 베이즈 (Naïve Bayes Classifier)

$$P(a_i \mid c_j) = \frac{P(a_i, c_j)}{P(c_j)} \Rightarrow P(a_i \mid c_j) = \frac{n_c}{n}$$

- Count of c_j: n (c_j 가 학습 데이터에서 출현한 횟수)
- Count of co-occurrence a_i and c_j : n_c (c_j 와 a_i 가 같이 나타난 횟수)
- c_j 와 a_i 가 같이 나타난 횟수가 0이면 어떻게 할 것인가? ➔ Smoothing

$$P(a_i \mid c_j) = \frac{n_c + 1}{n + |A|}$$

기계학습 기반 자연어처리

기계학습의 종류 (지도 학습)

지도 학습
(Supervised Learning)

비지도 학습
(Unsupervised Learning)

강화 학습
(Reinforcement Learning)

자동차

고양이

자동차

고양이

???

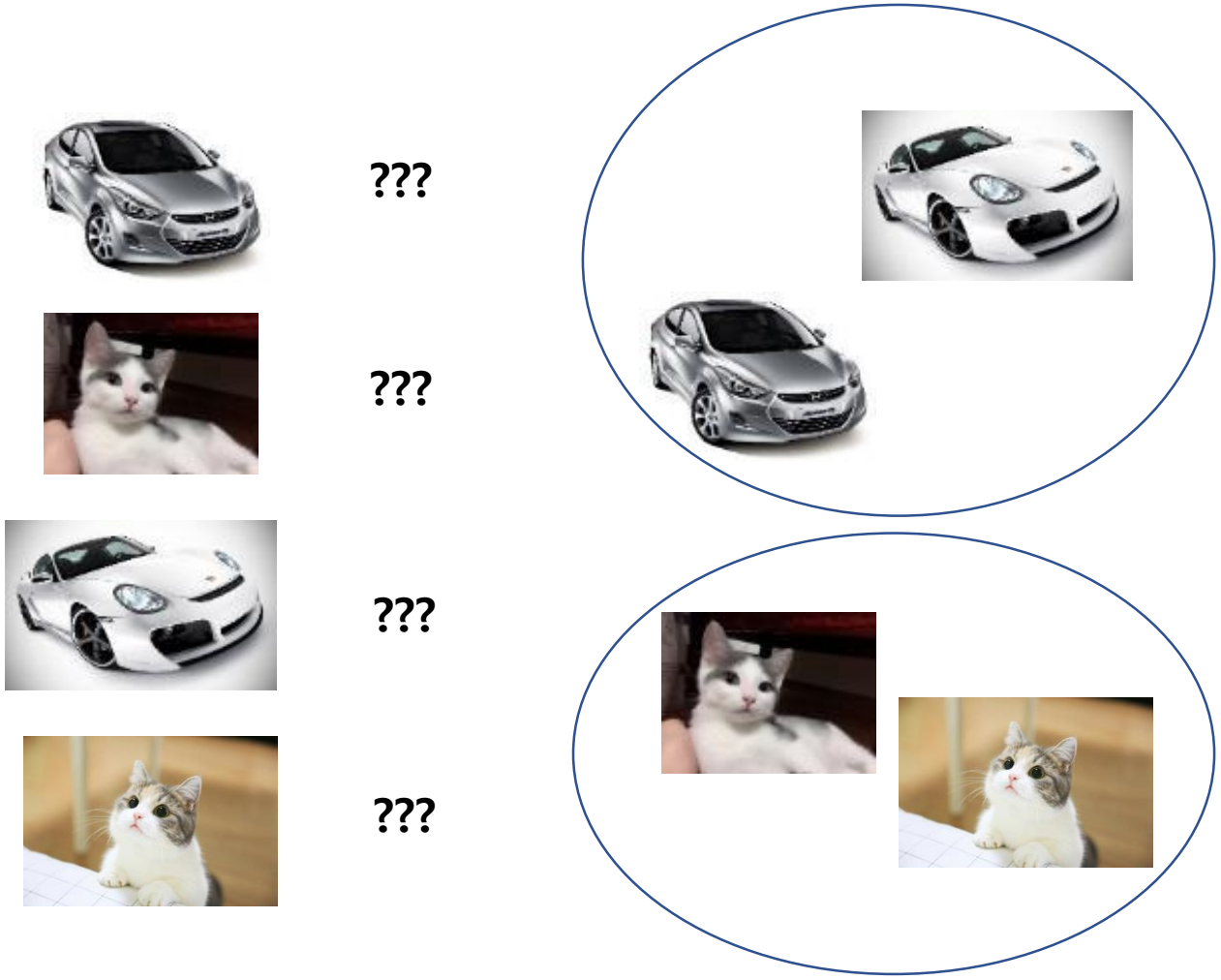
???

기계학습의 종류 (비지도 학습)

지도 학습
(Supervised Learning)

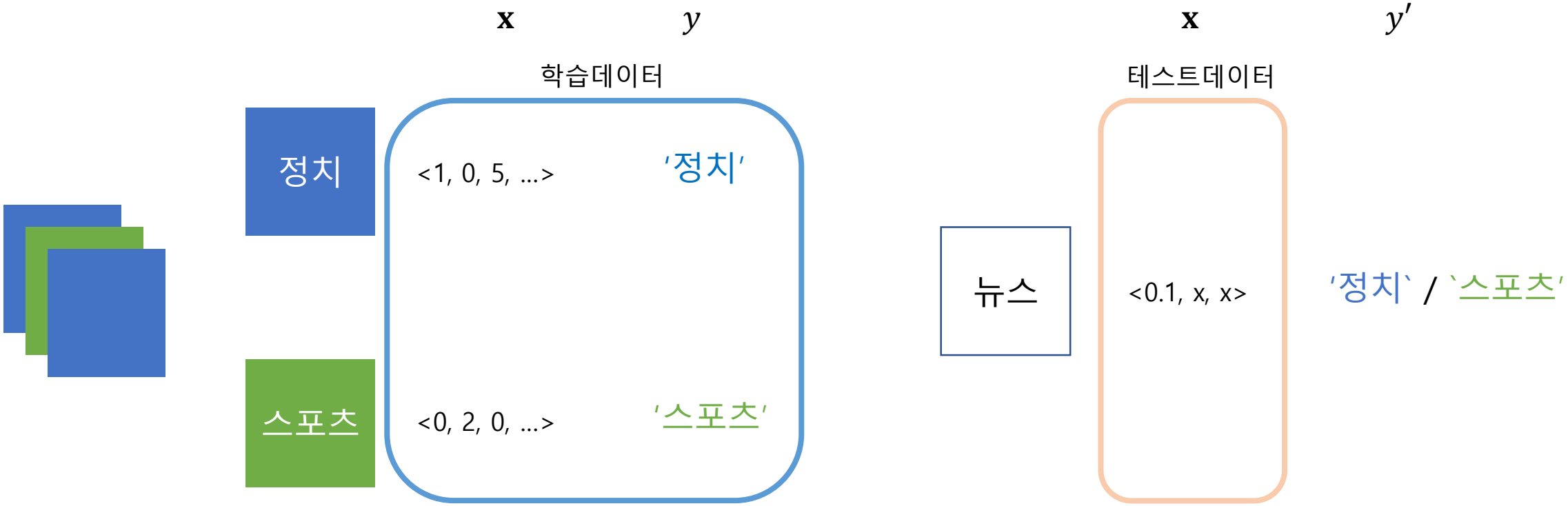
비지도 학습
(**Un**supervised Learning)

강화 학습
(Reinforcement Learning)



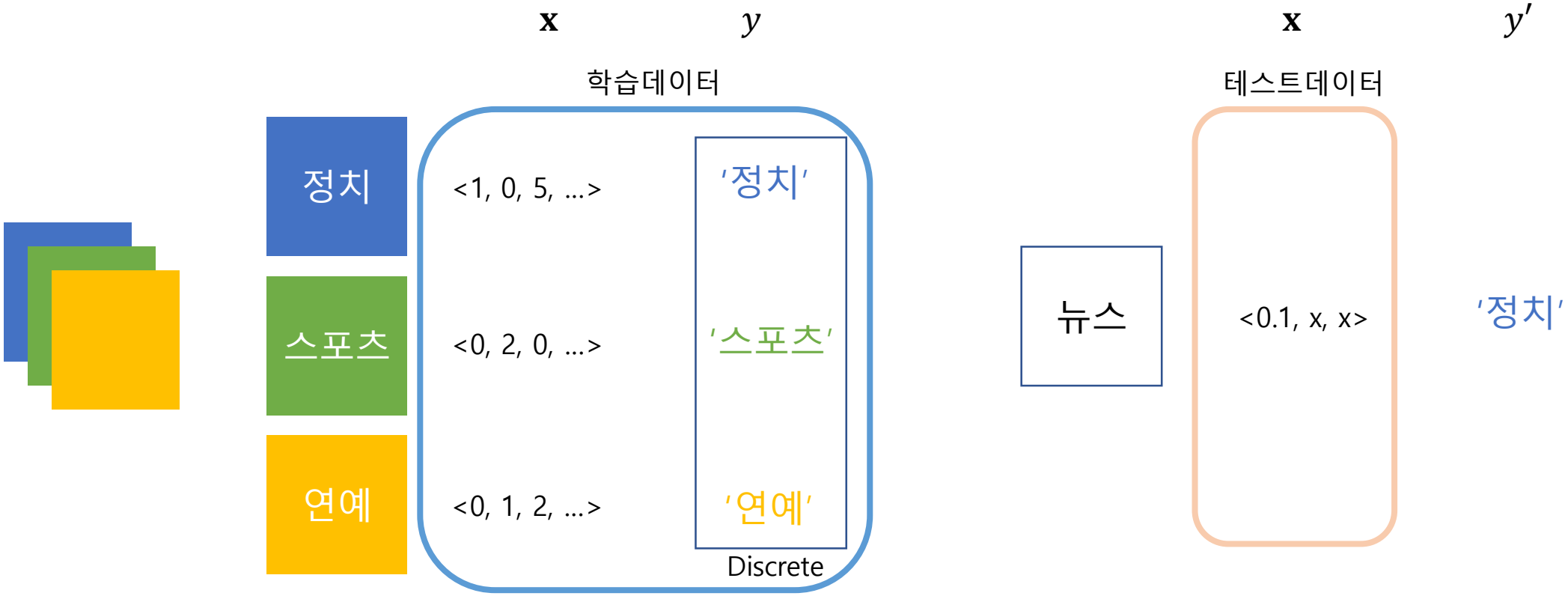
지도학습 (이진분류, Binary classification)

- 뉴스가 주어질 때, 뉴스의 카테고리를 맞춰보자



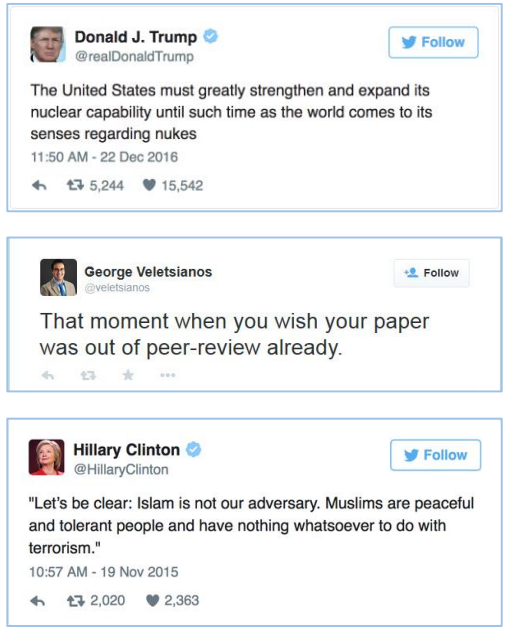
지도학습 (다중분류, Multi-class classification)

- 뉴스가 주어질 때, 뉴스의 카테고리를 맞춰보자



비지도학습

- Tweet이 주어질 때, 주제별로 비슷한 Tweet을 찾아보자



$\mathbf{x}$

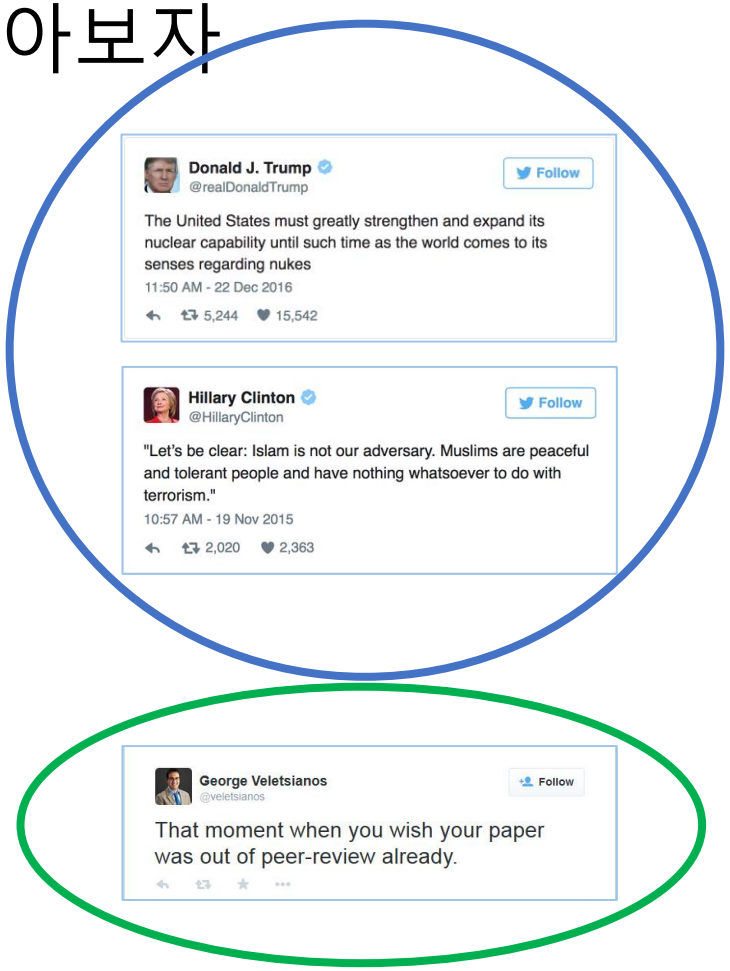
<0.1, x, x>

<x, 1.2, x>

<x, x, 1.5>

y

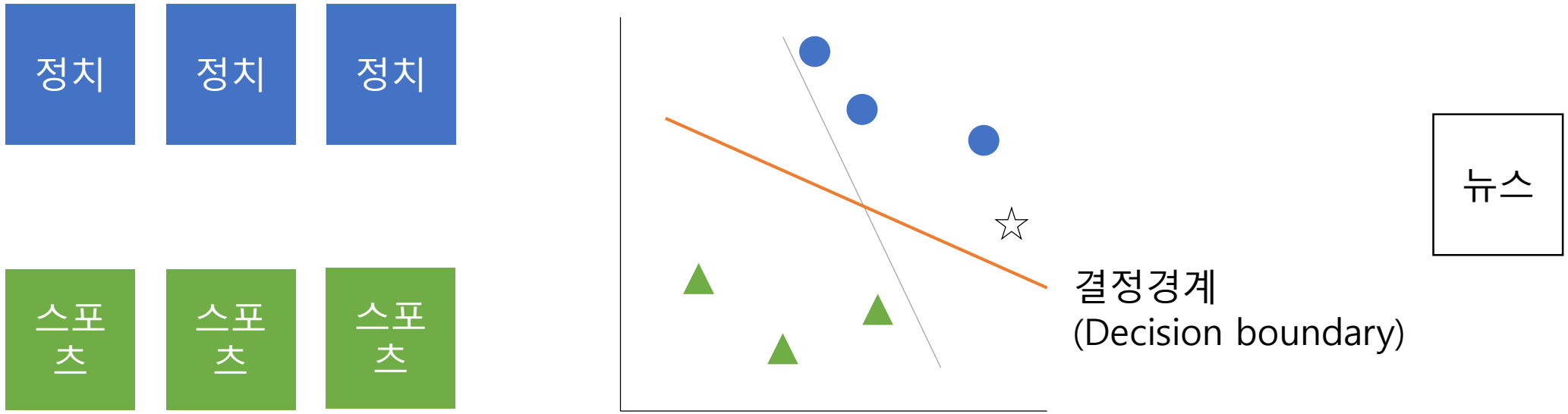
No info



군집화 (Clustering)

지도학습 (이진분류)

- 뉴스가 주어질 때, 뉴스의 카테고리를 맞춰보자



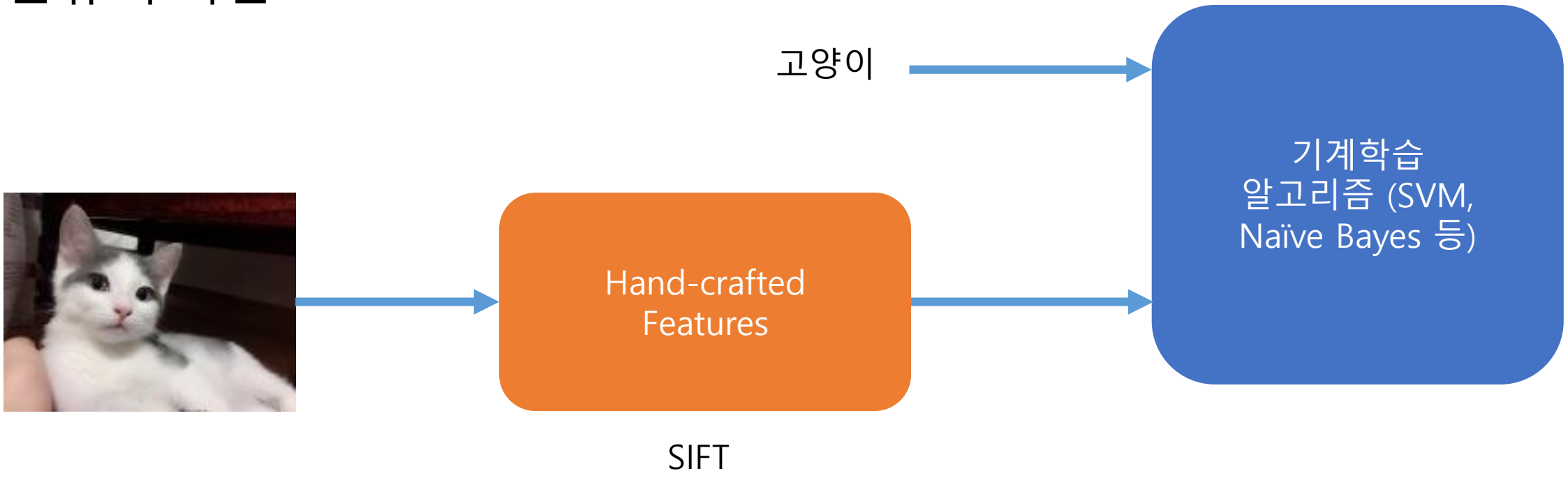
자질 표현
(Feature representation)

학습
(Learning)

평가
(Evaluation)

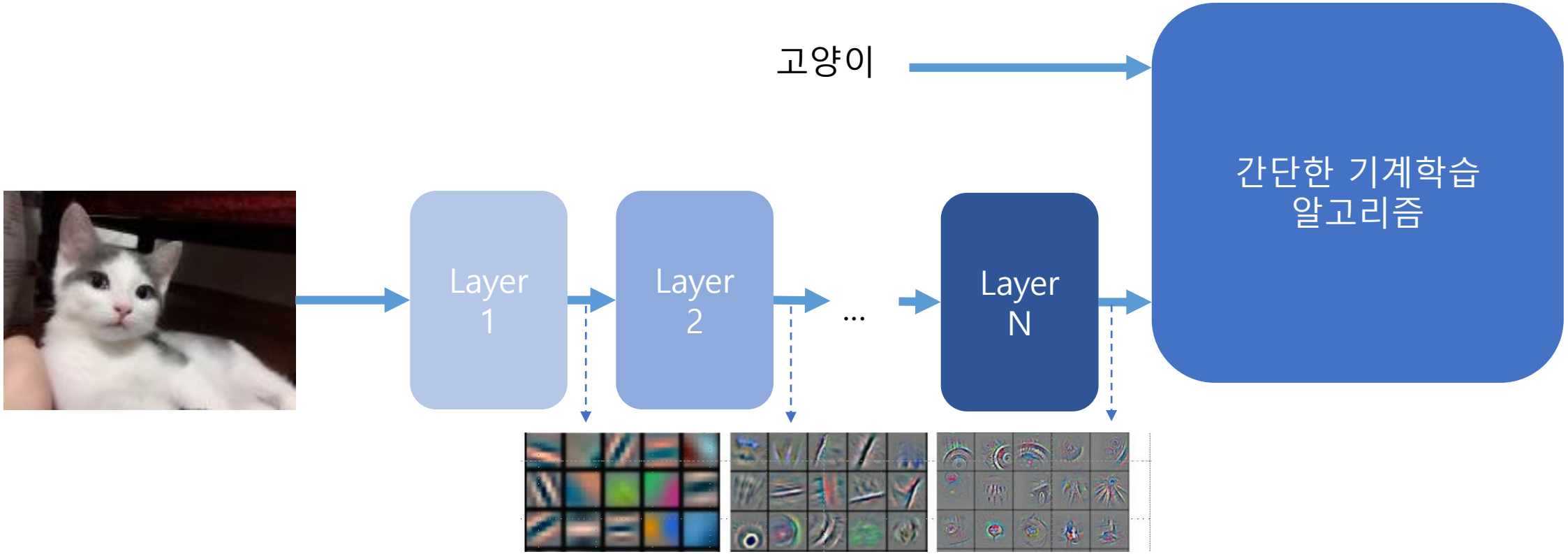
전통적인 기계학습 (분류)

- 전문가들이 디자인한 자질로 표현
- 분류기 학습

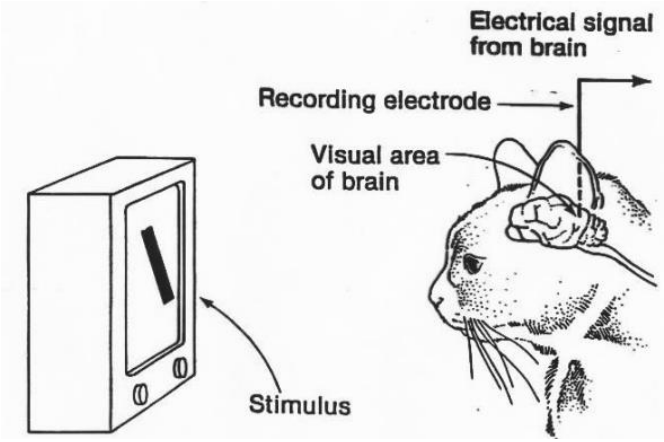


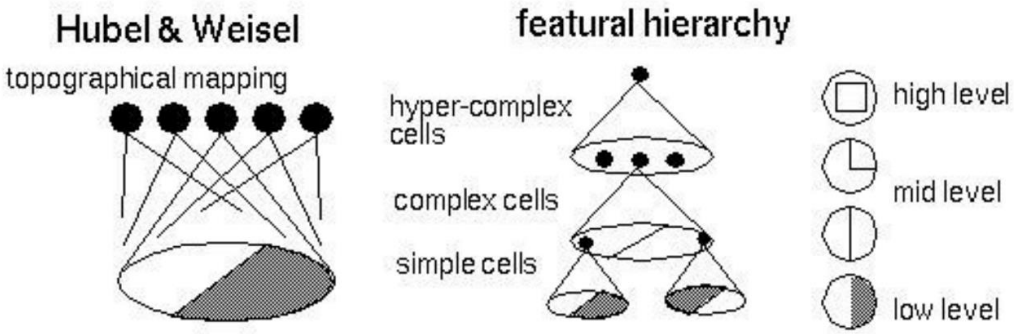
딥러닝 (Deep Learning)

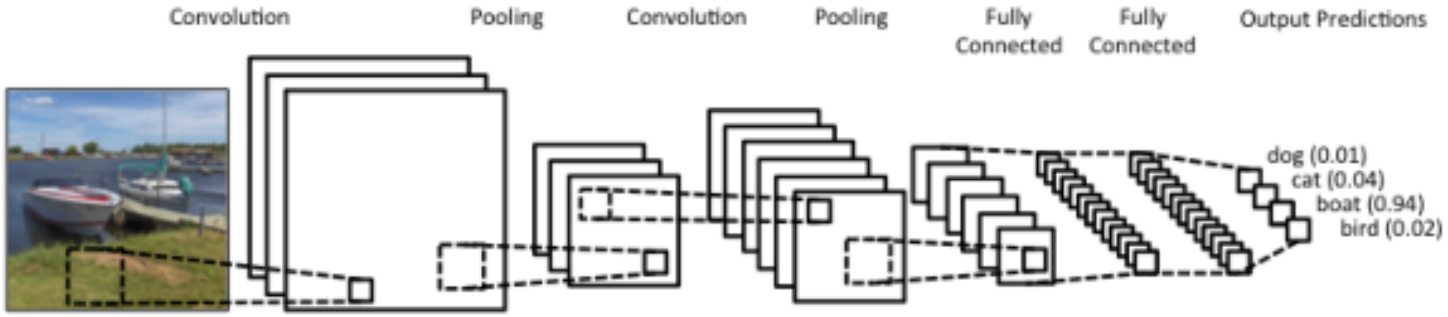
- **“Learn” features as a layer-wise hierarchical representation**



Convolutional Neural Networks (CNN)

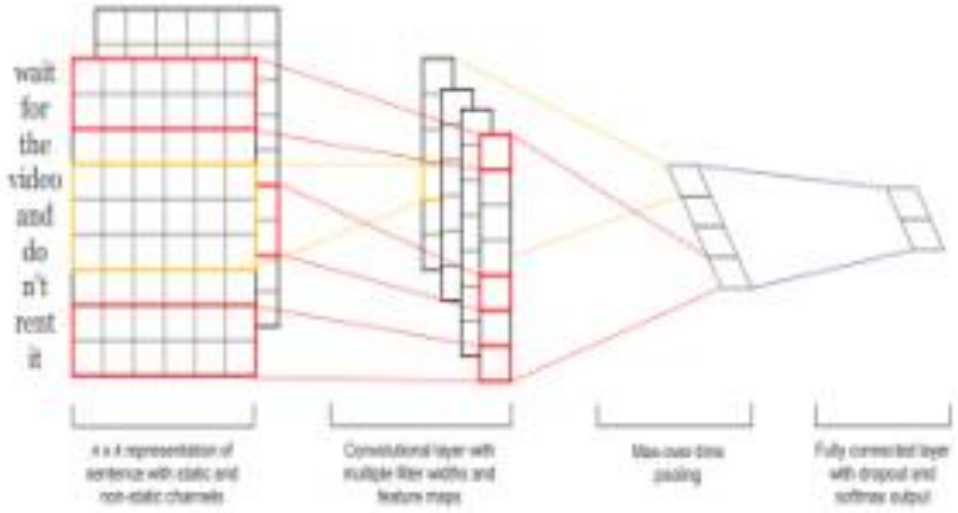




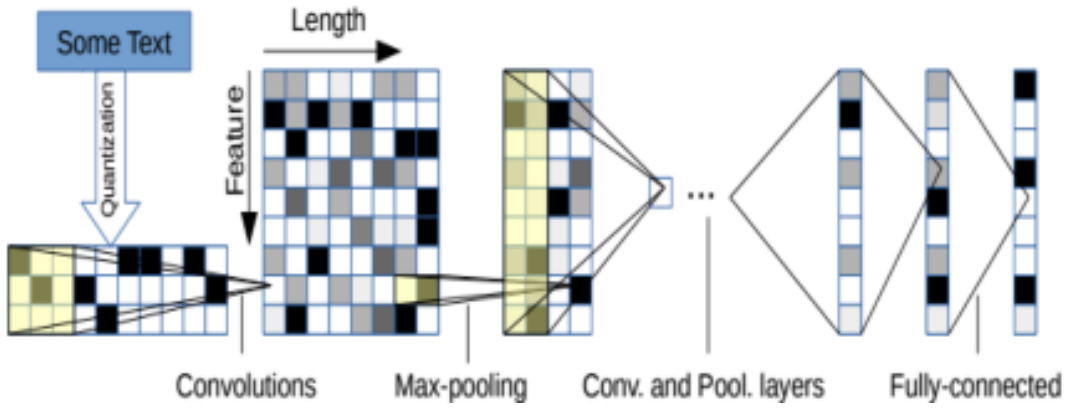


자연어처리에서의 CNN

- CNN for sentence classification



- Character-level CNN



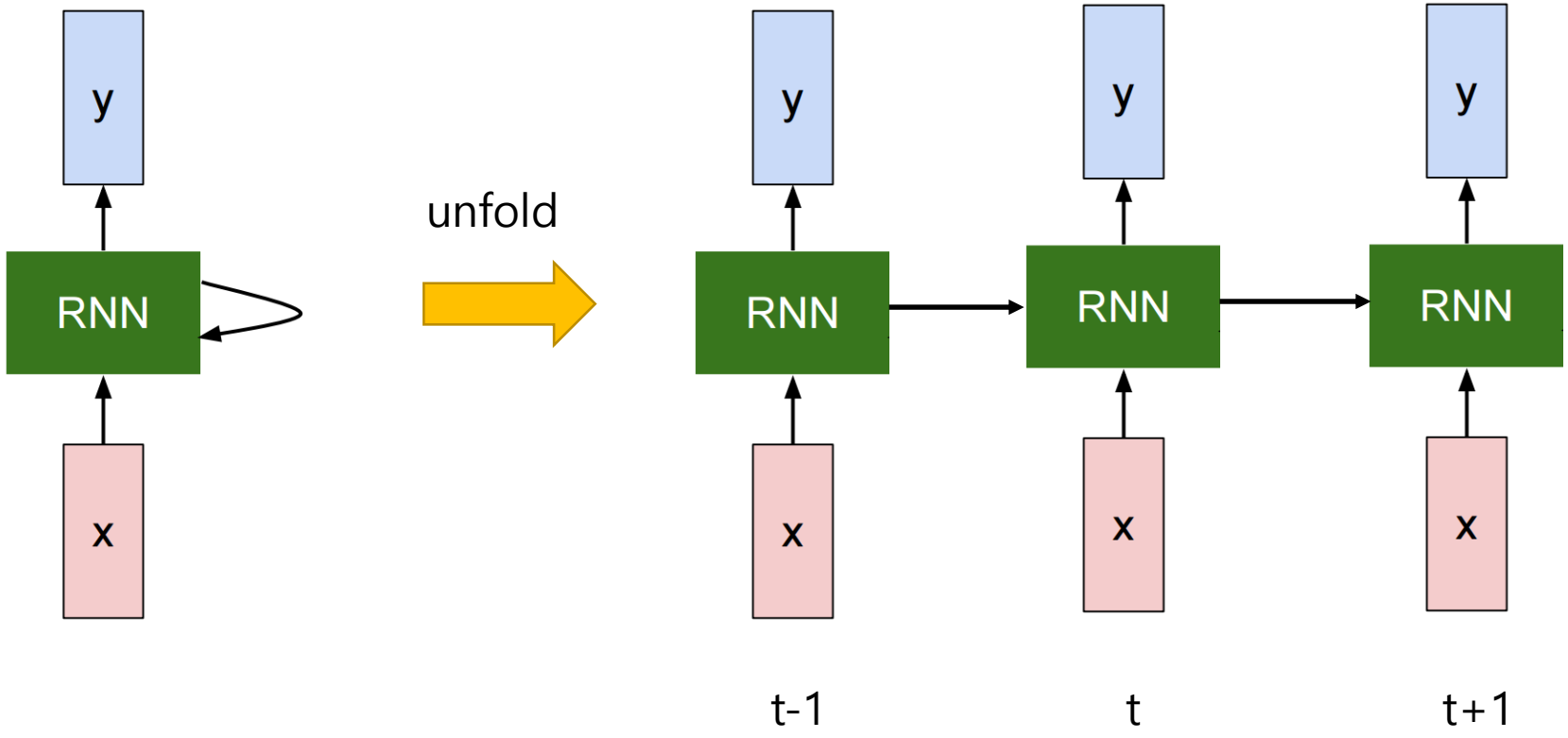
Convolution filter ➔ N-gram으로 해석

Y. Kim, Convolutional neural networks for sentence classification, EMNLP 2014

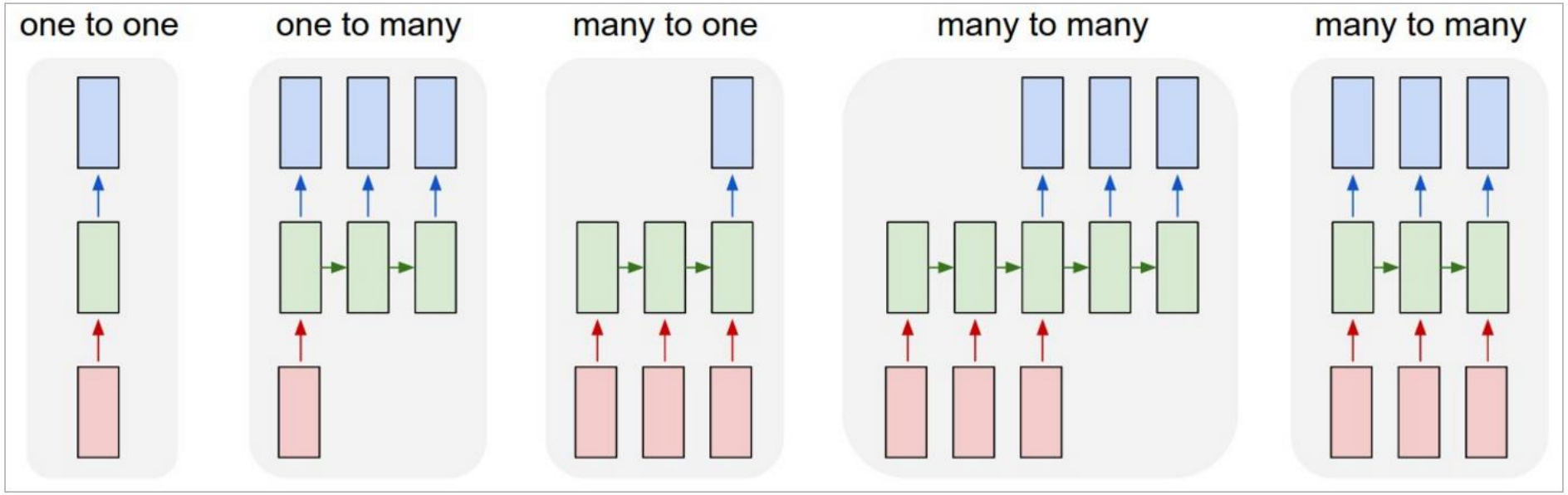
Zhang et al. Character-level convolutional networks for text classification, NIPS 2015

Recurrent Neural Network (RNN)

- Sequence 형태의 데이터 학습



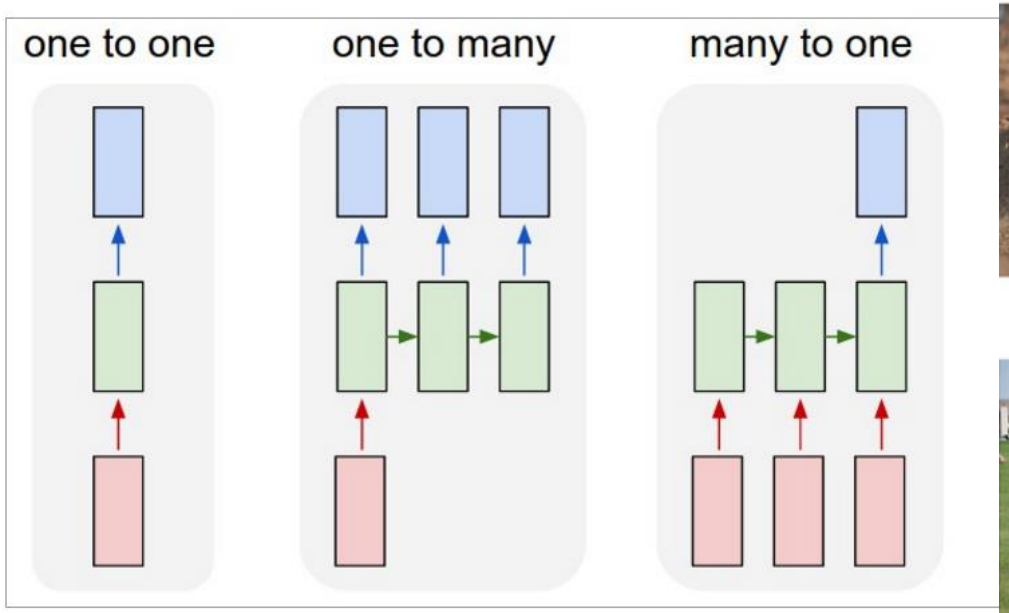
Recurrent Neural Networks



Vanilla Neural Networks

From CS231n

Recurrent Neural Networks



e.g. **Image Captioning**
image -> sequence of words

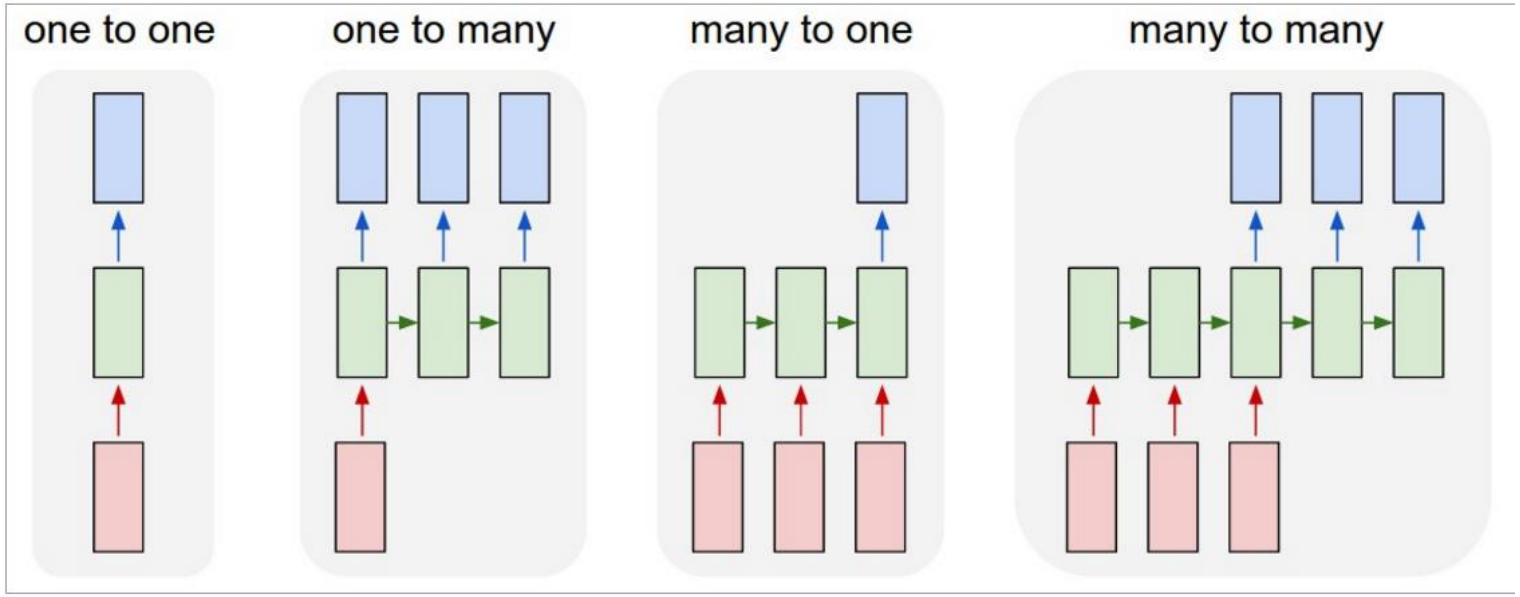
From CS231n

Recurrent Neural Networks

e.g. **Sentiment Classification**
sequence of words -> sentiment

From CS231n

Recurrent Neural Networks



e.g. **Machine Translation**
seq of words -> seq of words

From CS231n

자연어처리 및 응용 문제

- 자동 띄어쓰기 (automatic word spacing)
- 형태소 분석기 (morphological analyzer)
- 청킹 (Chunking)
- 개체명 인식기 (Named Entity Recognition)
- 구문분석기 (Dependency parsing)
- 의미역 결정 (Semantic role labeling)
- 담화 분석 (discourse analysis)
- …

- 문서분류기 (Document classification)
- 감정분석기 (Sentiment analysis)
- 언어모델 (Language modeling)
- 키워드 추출 (Keyword extraction)
- 요약 (Summarization)
- 기계번역 (Machine translation)
- 질의응답 (Question answering)
- 챗봇 (Chatbot)
- …

자연어처리와 기계학습

- 대부분의 **자연어처리 문제들은 분류 문제**로 해결가능
- 자동 띄어쓰기 (Automatic word spacing)
  - “아버지가방에들어가신다.” ➔ 아버지가 방에 들어가신다
  - 이진분류 (**T**rue – 띄어쓰기, **F**alse – 띄어쓰지 않기)

학습 데이터	**아**	**F**
	버	**F**
	지	**F**
	가	**T**
	방	**F**
	에	**T**
	들	**F**
	어	**F**
	…	

테스트 데이터	**나**	**F**
	는	**T**
	밥	**F**
	을	**T**
	…	**…**

자연어처리와 기계학습

- 대부분의 **자연어처리 문제들은 분류 문제**로 해결가능
- 개체명 인식 (Named entity recognition)
  - 프리미어 리그에서 뛰고 있는 토트넘의 손흥민 선수는 …
  - 다중분류 (사람, 리그, 축구팀, …)

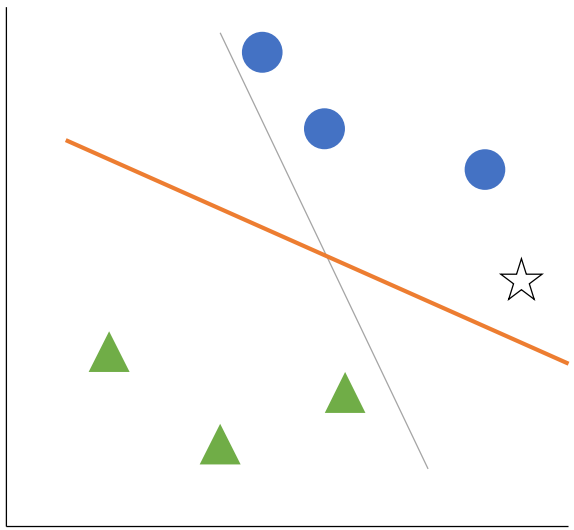
학습 데이터

프리미어	B-LEAGUE
리그	I-LEAGUE
에서	O
뛰고	O
있는	O
토트넘	B-SOCCER_TEAM
의	O
손흥민	B-PERSON
…	

```
; 한 가지 크게 달라진 점이 있다면, 89년 여름에는 탈퇴각서를 쓰고 학교
다시 시작해 보자는 해직동료들의 염려와 안타까움이 있을 따름이었다.
$한 가지 크게 달라진 점이 있다면, <89년:DT> <여름:DT>에는 탈퇴각서
학교현장으로 들어가 새롭게 다시 시작해 보자는 해직동료들의 염려와 안타
1	한	MM	O
2	가지	NNB	O
3	크	VA	O
3	게	EC	O
4	달라지	VV	O
4	ㄴ	ETM	O
5	점	NNG	O
5	이	JKS	O
6	있	VV	O
6	다면	EC	O
6	,	SP	O
7	89	SN	B_DT
7	년	NNB	I
8	여름	NNG	B_DT
8	에	JKB	O
8	는	JX	O
9	탈퇴각서	NNG	O
```

자연어처리와 기계학습

- (학습) 데이터 수집
- 데이터 표현
  - 형태소 분석기, 개체명 인식, 구문 분석 수행 (기존 기계학습)
  - 기본 분석만 수행하여 표현
- 기계학습 방법으로 모델 구축



결정경계
(Decision boundary)

자질 표현
(Feature representation)

단어 임베딩 (Word Embedding)

‘고양이’와 ‘개’는 비슷할까?

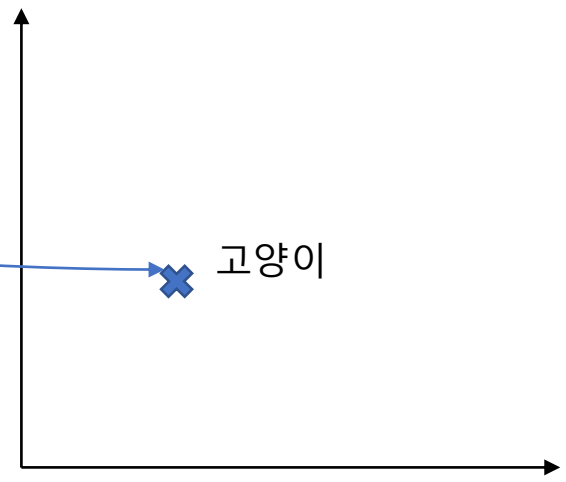
단어 표현 (Word representation)

고양이(cat)는 포유류에 속하는 동물이다. 일반적으로 "고양이"라 함은 인간에게 길들여진 **집고양이**(domestic cat)를 말한다. **야생고양이**(wild cat)는 약 10만 년에서 7만 년 전부터 존재했다. 2007년 기준으로 최근의 연구에 따르면 길들여진 고양이의 기원은 약 1만 년 전 근동지방에서 스스로 숲 속을 나와 사람들이 모여사는 마을에 대담하게 정착하여 길들여진 5마리 정도의 아프리카들고양이(*Felis silvestris lybica*)로 추측된다.[3]

고양이는 인류로부터 오랫동안 반려동물로 사랑받아 왔다. 실례로 고대 이집트의 벽화에는 고양이를 새 사냥에 이용하는 그림이 있다. 동아시아의 십이지에는 포함되어 있지 않지만, 타이와 베트남에서는 토끼 대신 고양이가 십이지 중 하나이다.

스핑크스처럼 털이 거의 없거나 맹크스처럼 꼬리가 없는 품종도 있다. 품종은 장모종, 중모종, 단모종으로 나뉘며. 단모종의 대표종은 아비니시안 고양이, 장모종의 대표종은 페르시안 고양이가 있다.

애완견과는 달리, 옛 습성이 살아 있고 발톱을 숨길 수 있어서 쥐나 작은 새를 사냥할 수 있는데, 혀로 가시가 있는 뼈에 붙은 고기를 핥아서 먹을 수 있다.



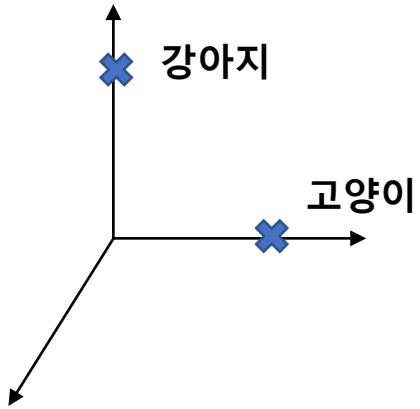
전통적인 단어 표현방법 (Word Representation)

- 단어백 (Bag-of-Word)
  - A vector with one 1 and a lot of zeroes
  - “one-hot” representation / 1-of-V coding / Discrete representation

$$\text{“고양이”} = \overbrace{(0,0,0,1,\ldots,0,0,0)}^{|V|}$$

- Simple and easy to implement
  - 벡터의 차원 = |전체 사전의 크기|

- 단어간의 유사도
  - Do two words (roughly) mean the same?
    - 고양이 = (0,0,0,1,….,0,0,0)
    - 강아지 = (0,0,1,0,…,0,0,0)
  - *Similarity*(“고양이”, “강아지”) = 0 (Wrong!!!)



Distributed Semantics

- 분포가설 (Distributional hypothesis)
  - “You shall know **a word by the company it keeps**” (J.R. Firth, 1957)

```
he curtains open and the stars shining in on the barely
ars and the cold , close stars " . And neither of the w
rough the night with the stars shining so brightly , it
made in the light of the stars . It all boils down , wr
 surely under the bright stars , thrilled by ice-white
sun , the seasons of the stars ? Home , alone , Jay pla
m is dazzling snow , the stars have risen full and cold
un and the temple of the stars , driving out of the hug
 in the dark and now the stars rise , full and amber a
```

	shining	bright	trees	dark	look
stars	38	45	2	27	12

New Directions in Vector Space Models of Meaning , Tutorial in ACL 2014

문맥정보(Context)를 사용하여 단어를 표현

- 문맥정보 어디서부터 어디까지?
  = How to make **co-occurrence matrix**
- 단어-문서 행렬
  - Term-document matrix
- 단어-단어 행렬
  - Term-term matrix
    (or word-word co-occurrence matrix)

Term-document matrix

문서

단어

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

벡터가 비슷하면 두 단어가 유사하다

벡터가 비슷하면 두 문서가 유사하다

Word-word co-occurrence matrix

- 문맥정보를 문서 대신 주변 단어로 줄여보자
- Sliding window (symmetric)

- I like deep learning .

- I like NLP .

- I enjoy flying .

|V|

counts	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

단어 임베딩 (Word embedding)

- Word-word co-occurrence matrix의 문제점
  - 단어가 늘어날 수록 차원이 커진다
  - 저장 공간이 많이 필요
  - 희소한 벡터 (sparse representation)

- 단어 임베딩 (Word embedding)
  - 중요한 정보만 남기고 **적은 차원**에서 단어를 표현하고 싶다!
    - 기계학습에서 자질로 사용할 때 처리하기 쉽다는 장점
  - Singular Value Decomposition (SVD)
  - **Skip-gram, CBOW**
  - Brown clustering

단어 임베딩 (Word embedding)

- 단어 ➔ *d*-차원 **실수 벡터 (real-value dense** vector)
  - *d* typically in the range 50 to 300
    - Note that ***d*** **<< |Vocabulary|**

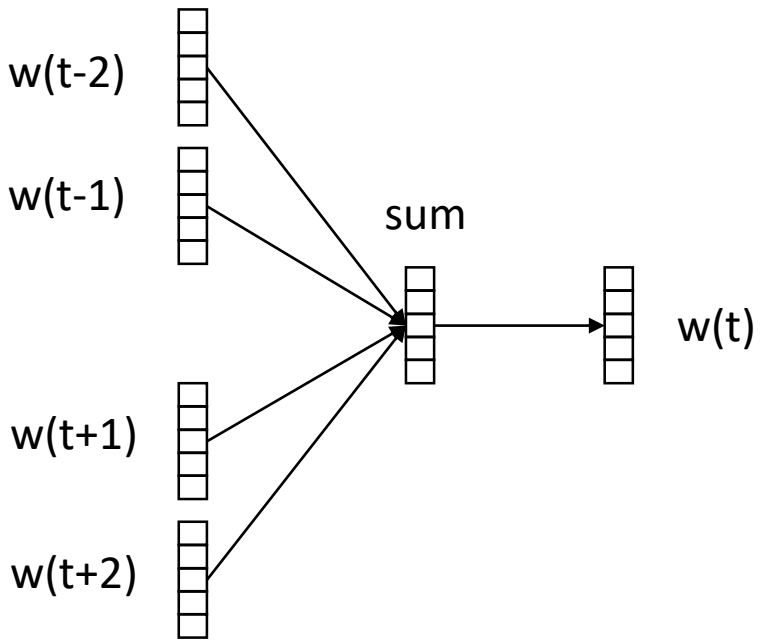
$$\text{“고양이”} = \overbrace{(0.343, -0.981, 0.753, 0.232, \ldots.)}^{\boldsymbol{d}}$$

  - **Predictive-Based**
    - 단어에서 문맥을 **예측**
    - 문맥에서 단어를 **예측**
    - word2vec (**Skip-gram**, **CBOW**), …

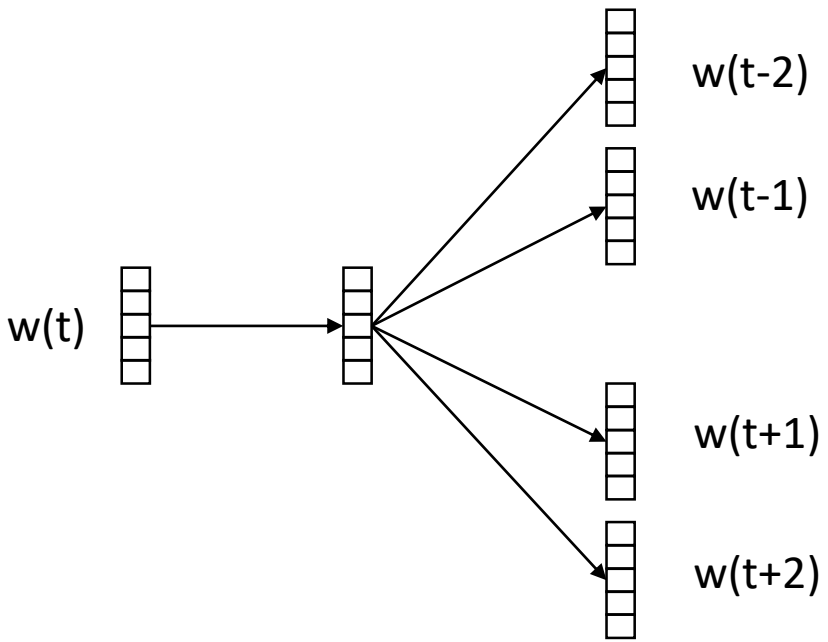
Don’t count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors, ACL 2014

Continuous bag-of-word / Skip-gram

- **Predict** a word given its bag-of-words context (CBOW)



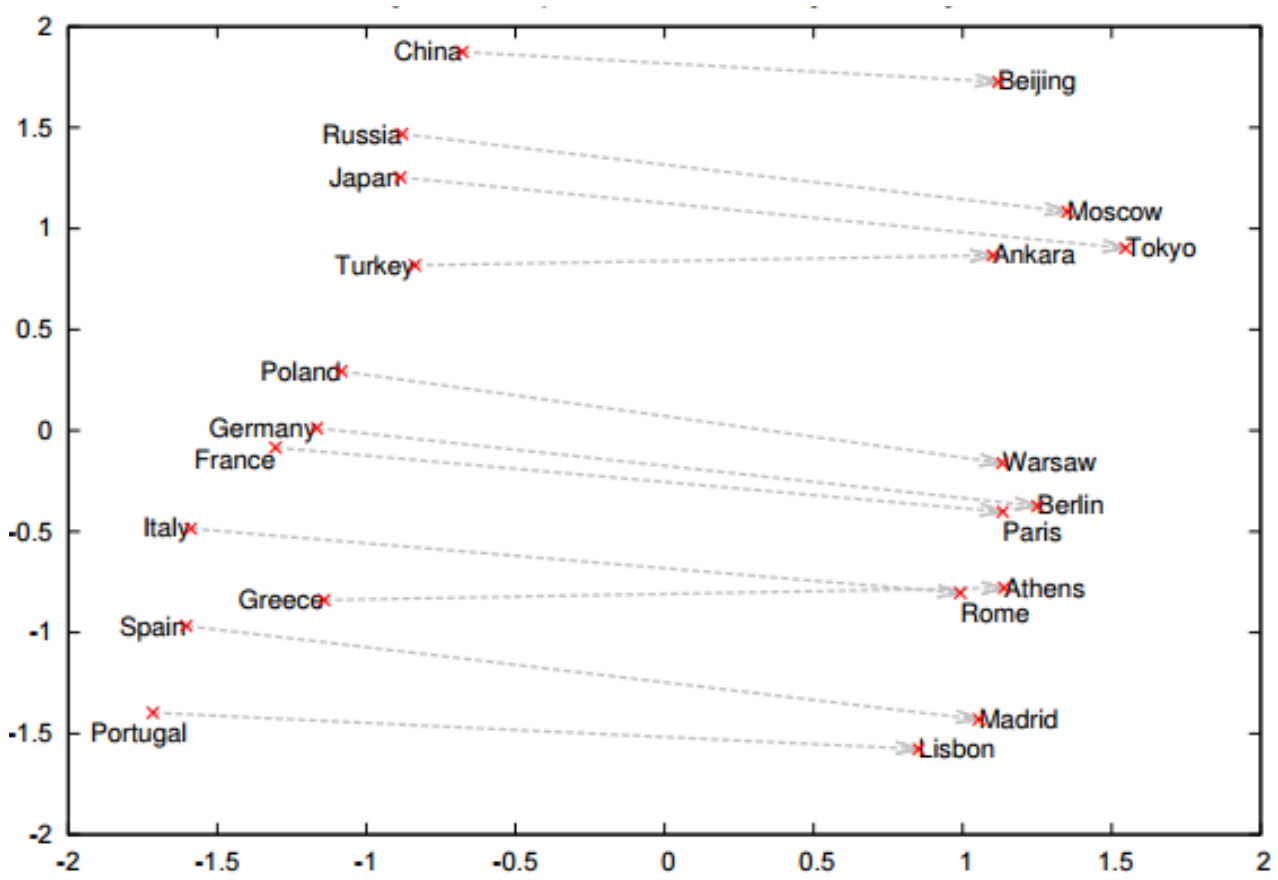
- **Predict** a context word (position-independent) from the center word



비슷한 문맥을 가진 단어들은 서로 비슷한 의미를 가짐

Distributed Representations of Words and Phrases and their Compositionality , NIPS 2013

Linear relationships in Word2vec

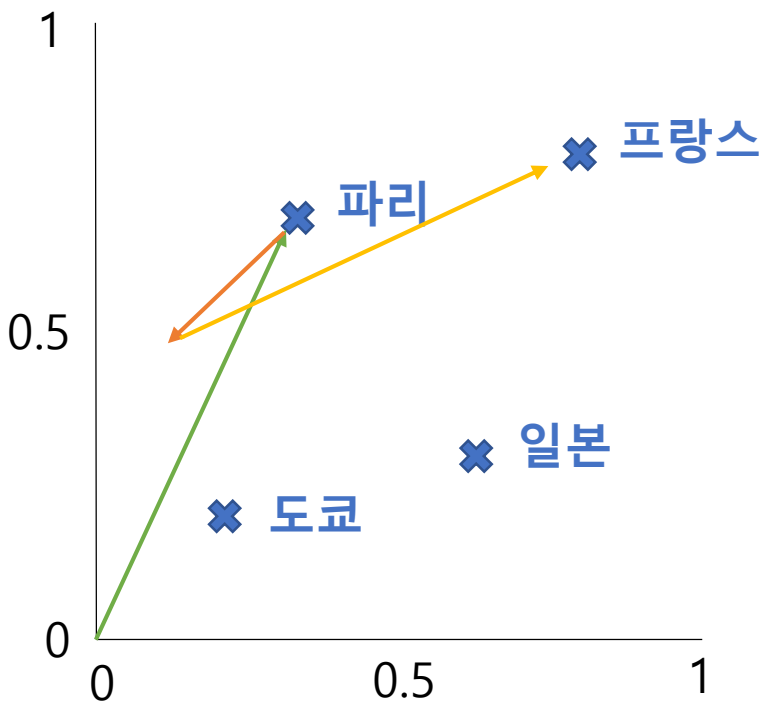


Word analogies test

- 벡터들의 연산으로 테스트할 수 있음

도쿄 : 일본 = 파리 : ?

+	파리	[0.3, 0.7]
-	도쿄	[0.2, 0.2]
+	일본	[0.6, 0.3]
	프랑스	[0.7, 0.8]



Word2vec 평가

- Task-based evaluation
  - 오타 교정
  - 질의 응답 시스템
  - …

- Intrinsic evaluation
  - Word vector distances and their correlation with human judgments
    - WordSim353: **353 noun pairs rated 0-10**

  - TOEFL multiple-choice vocabulary tests

WordSim353

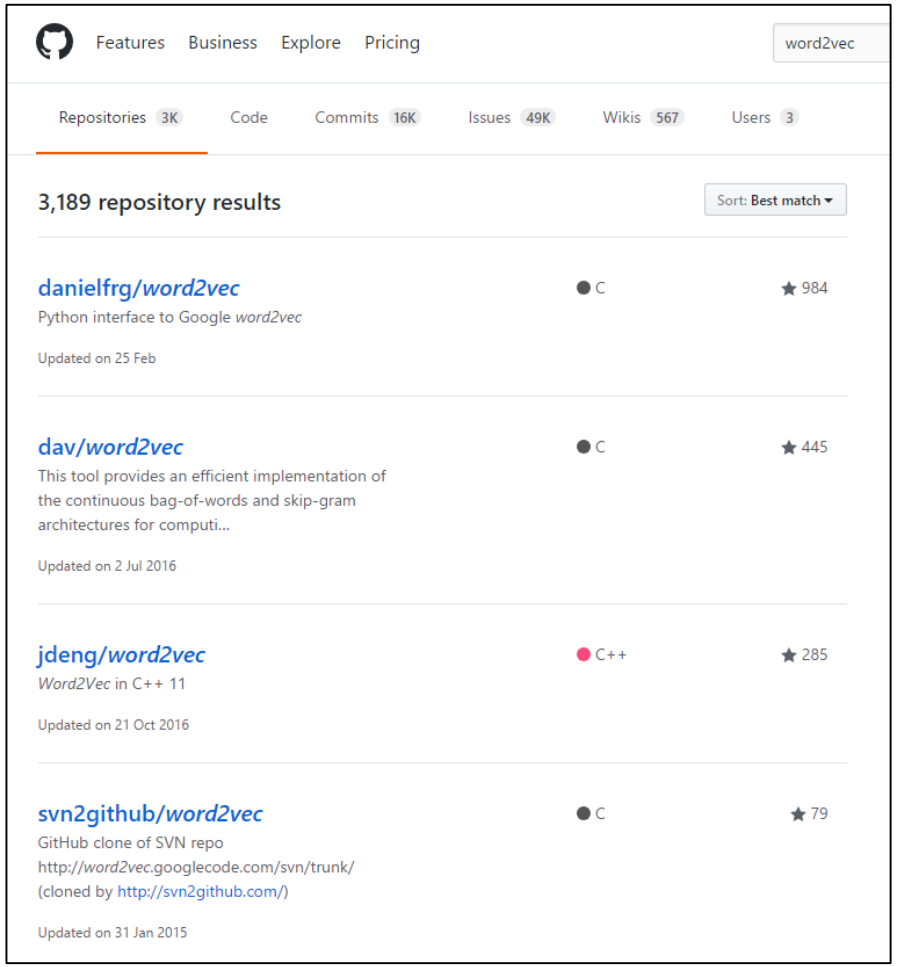
Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10.00
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62

<u>Levied</u> is closest in meaning to:
imposed, believed, requested, correlated

Word2vec Tools

- 다양한 Word2vec 툴이 존재
  - Word2vec (C, Original)
    - https://code.google.com/archive/p/word2vec/
  - Word2vec in genism (Python)
    - https://radimrehurek.com/gensim/models/word2vec.html
  - Word2vec in tensorflow (Python)
    - https://www.tensorflow.org/tutorials/word2vec
  - Word2vec in deeplearning4j and Scala
    - https://deeplearning4j.org/kr/word2vec
  - fastText (C++, Facebook)
    - https://github.com/facebookresearch/fastText
  - …

- 기구축된 Word2vec 모델
  - https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md
  - https://insikk.github.io/awesome-korean-nlp/
  - https://github.com/Kyubyong/wordvectors



'word2vec' in github

(실습) Word2vec 구축 및 평가

```
from __future__ import print_function

import codecs
# Open-text   import
from konlpy.tag import Twitter
pos_tagger = Twitter()

def read_data(filename):
    with codecs.open(filename, 'r', 'utf-8') as f:
        data = [line.split('\t')[1] for line in f.read().splitlines()]
    return data

data = read_data('wiki.txt')
print(len(data))

def tokenize(doc):
    return ['/'.join(t) for t in pos_tagger.pos(doc, norm=True, stem=True)]

import time
start_time = time.time()
tagged_docs = [tokenize(row) for row in data]
print(len(tagged_docs))
print("--- %s seconds ---" % (time.time() - start_time))

import gensim

# sg defines the training algorithm.
# (sg=0), CBOW is used.
# (sg=1), skip-gram is employed.
# size is the dimensionality of the feature vectors.
# window is the maximum distance between the current and predicted word within a sentenc
# model = gensim.models.Word2Vec(sentences, min_count=10, size=100, window=2, sg=0)
start_time = time.time()
model = gensim.models.Word2Vec(tagged_docs, min_count=1, size=100, window=5, sg=0)
print("--- %s seconds ---" % (time.time() - start_time))
```

```
model.save(os.path.join('model', 'wiki.w2v'))
```

```
for t in model.wv.most_similar('고양이'):
    print(t)
```

```
model.wv.most_similar(positive=['파리', '일본'], negative=['도쿄'])
```

```
model.wv.most_similar(positive=['여성', '왕'], negative=['남성'])
```

```
questions = Sentences(os.path.join('eval','word2vec'))
```

```
result = model.wv.accuracy(os.path.join('eval','word2vec','questions-words-kor.txt'))
for r in result:
    print('%s₩t%s₩t%s' % (r['section'], len(r['correct']), len(r['incorrect'])))
```

Word analogy task

- : capital-common-countries
  - 도쿄 : 일본 = 파리 : ? ➔ 프랑스
- : capital-world
  - 방콕 : 타이 = 베를린 : ? ➔ 독일
- : currency
  - 유럽 : 유로 = 베트남 : ? ➔ 동
- : city-in-state
  - 어바인 : 캘리포니아 = 시카고 : ? ➔ 일리노이
- : family
  - 아버지 : 어머니 = 할아버지 : ? ➔ 할머니

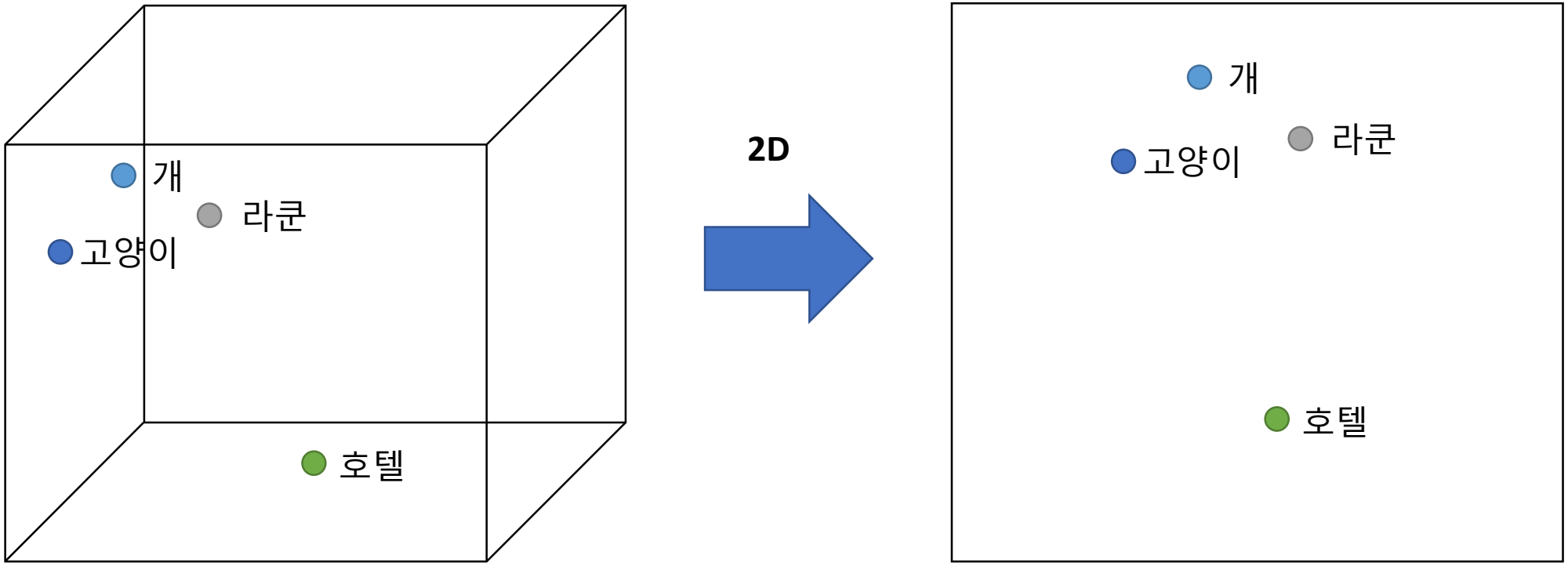
Word2vec Visualization

- 100~300차원 단어벡터
  - Cat = (0.01, -1.34, …)
  - Dog = (0.03, -1.24, …)
  - Hotel = (-1.20, 1.24, …)
  - Motel = (-1.11, 0.95, …)

- **Sim(“Cat”, “Dog”) > Sim(“Cat”, “Hotel”)**

t-distributed stochastic neighbor embedding (tSNE)

t-distributed stochastic neighbor embedding (tSNE)



Project that preserves the neighborhood structure of the data

http://lvdmaaten.github.io/tsne/

(실습) tSNE

```
import matplotlib.pyplot as plt
from matplotlib import font_manager, rc
font_fname = '/Library/Fonts/NanumBarunGothicLight.otf'     # A font of your choice
font_name = font_manager.FontProperties(fname=font_fname).get_name()
rc('font', family=font_name)

from sklearn.manifold import TSNE
# X = model[model.wv.vocab]
X = model[model.wv.vocab][0:100]

tsne = TSNE(n_components=2) # n_components = Dimension of the embedded space.
X_tsne = tsne.fit_transform(X)

plt.figure(figsize=(10, 10))
plt.scatter(X_tsne[:, 0], X_tsne[:, 1])
for label, x, y in zip(model.wv.vocab, X_tsne[:, 0], X_tsne[:, 1]):
    plt.annotate(label, xy=(x, y), xytext=(0, 0), textcoords='offset points')
# plt.show()
plt.savefig('w2v.png', dpi=600)
```

Further reading

- 다양한 단어 임베딩이 계속 제안되고 있음
  - Glove, Swivel, fastText
- 영어 word embedding 평가 사이트
  - http://wordvectors.org
- 한국어 평가 데이터
  - https://github.com/SungjoonPark/KoreanWordVectors

감성분석 (sentiment analysis)

상영중인 영화 “안시성” 보러 갈래? 응! 평이 좋더라!

우리 제품 지금 분위기 어떻지? 리뷰를 분석해보니 망했어요 ㅠㅠ

감성분석

11번가 so***** | 2017.06.12 5/5

최고예요

와이프가 찬바람을 바로 쐬지 못해 구매한 무풍에어컨... 생각보다 슬림한 크기였지만 성능은 아주 좋네요... 배송 및 설치해주신 기사분들이 너무 꼼꼼히, 친절히 일처리를 해주셔서 너무 기분이 좋았습니다.... 앞으로도 사업 번창하세용~~~

영화를 본 직후라 과격한 표현이 될 수 있겠으나 결과를 말하자면 이런 기대는 6년에 걸친 망상이 되어버렸다. 시리즈의 마지막다운 커다란 스케일... 1~3편에서 보아왔던 유머감 등 초-중반 까지는 그래도 꽤 볼 만 하였다. 그러나 떡밥 회수, 개연성, 인물들간의 구도 등 모든것을 똥통에 쳐박아버린 후반은 시리즈 팬으로서 도저히 눈을 뜨고 보기 힘들었다.

2

SentiWordNet

Korean Sentiment Analysis Corpus

한글 오픈데이터 플랫폼

집단지성... 그리고 한글 감성을 나누다

감성분석

- 극성 분석(Polarity Detection)
  - 주어진 데이터가 **긍정**인지, 혹은 **부정**인지를 판단

11번가 so***** | 2017.06.12

최고예요

와이프가 찬바람을 바로 쐬지 못해 구매한 무풍에어컨... 생각보다 슬림한 크기였지만 성능은 아주 좋네요... 배송 및 설치해주신 기사분들이 너무 꼼꼼히, 친절히 일처리를 해주셔서 너무 기분이 좋았습니다.... 앞으로도 사업 번창하세용~~~

영화를 본 직후라 과격한 표현이 될 수 있겠으나 결과를 말하자면 이런 기대는 6년에 걸친 망상이 되어버렸다. 시리즈의 마지막다운 커다란 스케일... 1~3편에서 보아왔던 유머감 등 초-중반 까지는 그래도 꽤 볼 만 하였다. 그러나 떡밥 회수, 개연성, 인물들간의 구도 등 모든것을 똥통에 쳐박아버린 후반은 시리즈 팬으로서 도저히 눈을 뜨고 보기 힘들었다.

미리 정의됨

긍정

부정

데이터 → 감성분류 모델 → 긍정

영화리뷰 감성분석

- 영화리뷰 데이터
  - 한국어 140자평 영화 데이터
  - http://github.com/e9t/nsmc/

★★★★★ 10 베스트 이번에 분노의 질주를 처음봤는데 앞 시리즈 전부 찾아보게 생겼네요...ㅎ 진짜 최고였어요

gusw**** | 2017.04.12 12:39 | 신고

공감 188 비공감 47

★★★★☆ 8 베스트 항상 믿고보는 분노의질주 시리즈 여전히 실망시키지 않고 박진감 넘치게 회가 거듭할수록 화려하게 질주한다

스타일(dkdl****) | 2017.04.12 12:20 | 신고

공감 194 비공감 58

★☆☆☆☆ 1 아...제시간을 돌려주세요..ㅜㅜ

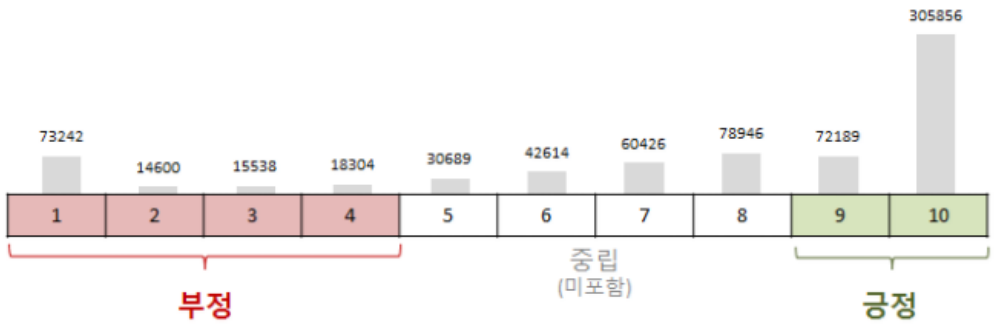
hsj0**** | 2017.04.13 22:47 | 신고

공감 0 비공감 1

★☆☆☆☆ 1 보고나와서 개빡쳐있는데 왜 난 이걸 못봤을까.베스트평점들 공감이랑 비공감 비율만봐도 답나오는영화.알아서 거를거라 믿는다;;

aqwe**** | 2017.04.13 22:31 | 신고

공감 3 비공감 1



https://www.lucypark.kr/slides/2015-pyconkr/#39

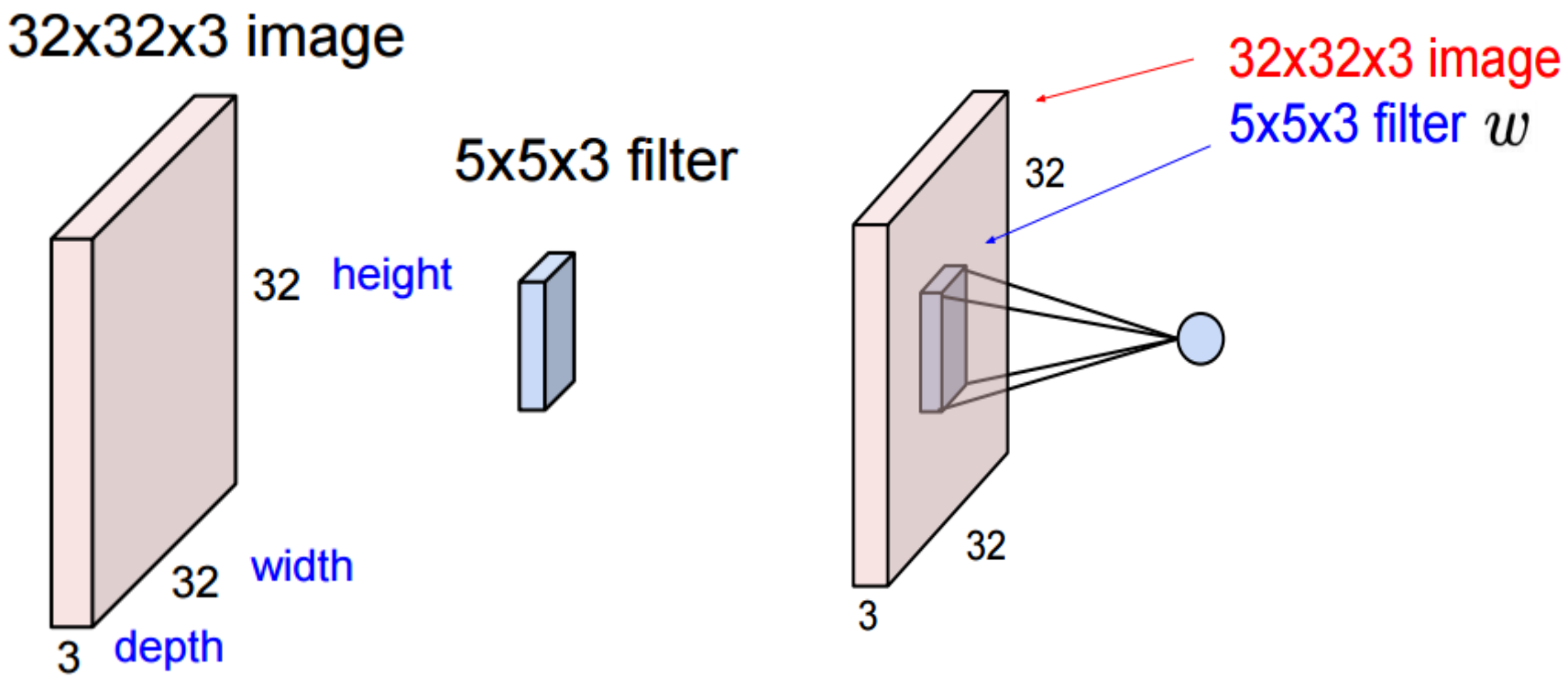
NSMC (Naver Sentiment Movie Corpus)

- NSMC Dataset
  - 네이버 영화 리뷰 크롤링 데이터 총 200K
  - 평가 9~10: 긍정(1), 1~4:부정(0), 이외의 리뷰문은 학습데이터에서 제외
  - 리뷰문 예

ID	Document	Label
4655635	폴리스스토리 시리즈는 1부터 뉴까지 버릴께 하나도 없음.. 최고	1(긍정)
9251303	와.. 연기가 진짜 개쩔구나.. 지루할거라고 생각했는데 몰입해서 봤다.. 그래 이런게 진짜 영화지	1(긍정)
3653446	훈훈한 내용이지만 스토리가 다소 억지스럽고 부족하다	0(부정)
6484287	재미가 없어요 시간이 아깝고	0(부정)
9279041	완전 감동입니다 다시봐도 감동	1(긍정)

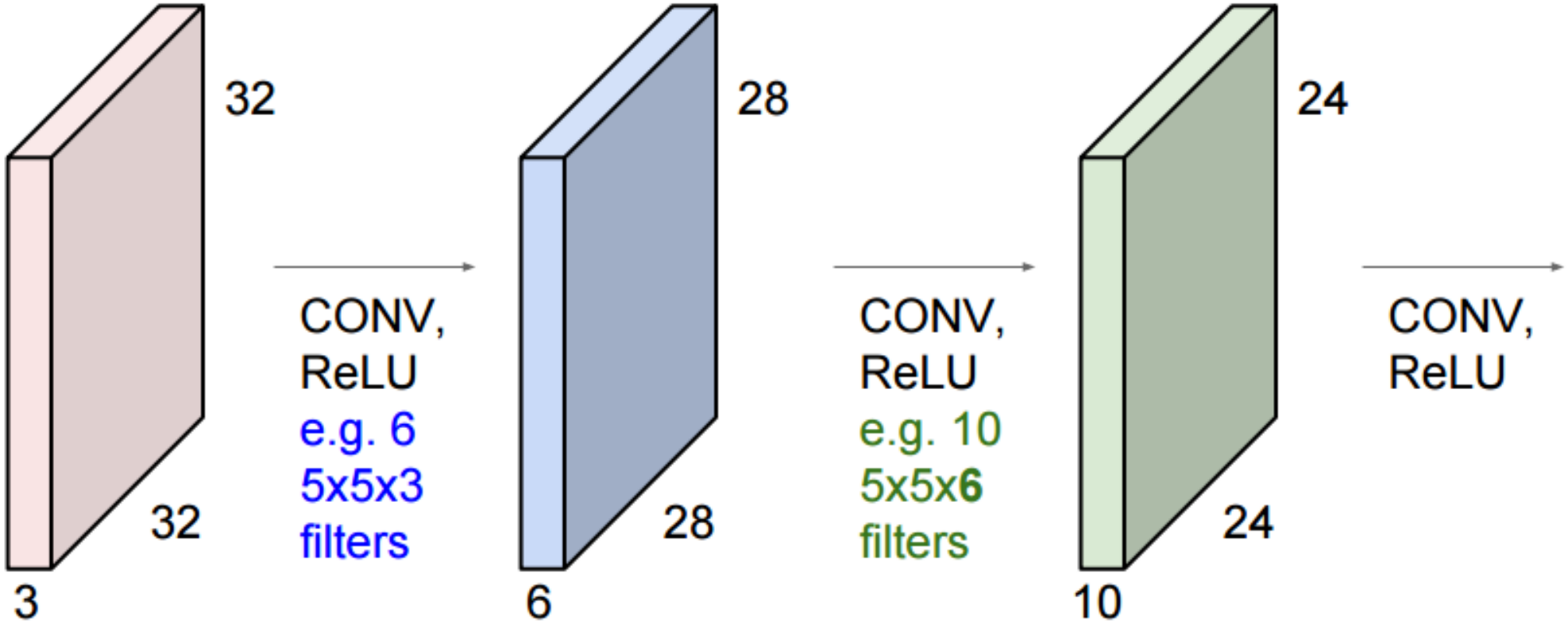
Convolution Neural Network (CNN)

- 컨볼루션 (Convolution) Layer와 풀링 (Pooling) Layer들이 순차적으로 쌓여져 있는 Network



From CS231n

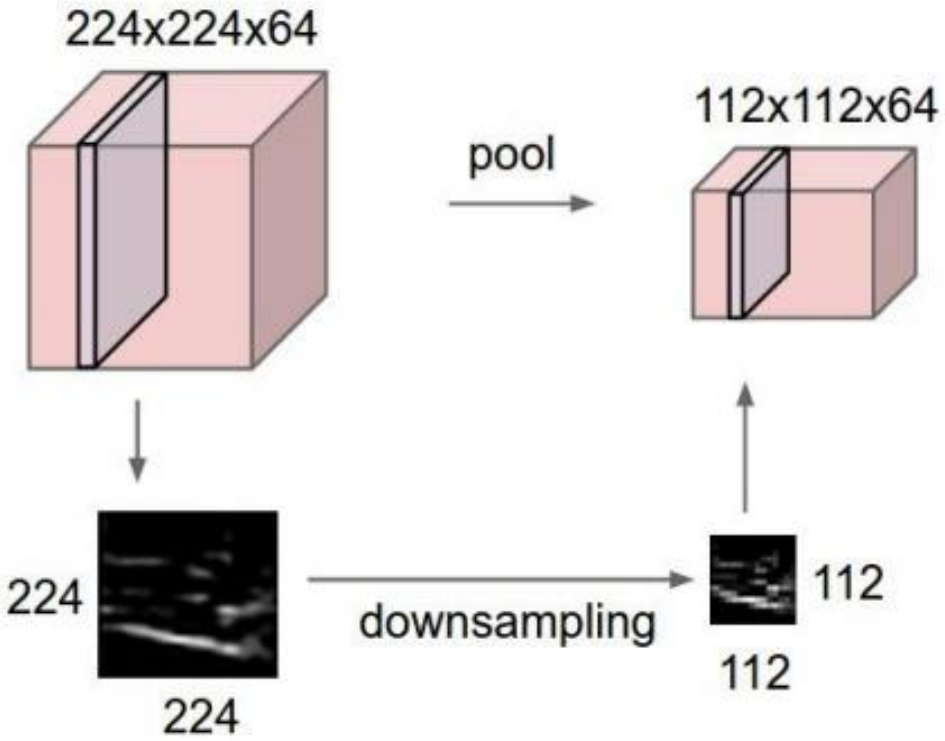
Convolution Neural Network (CNN)

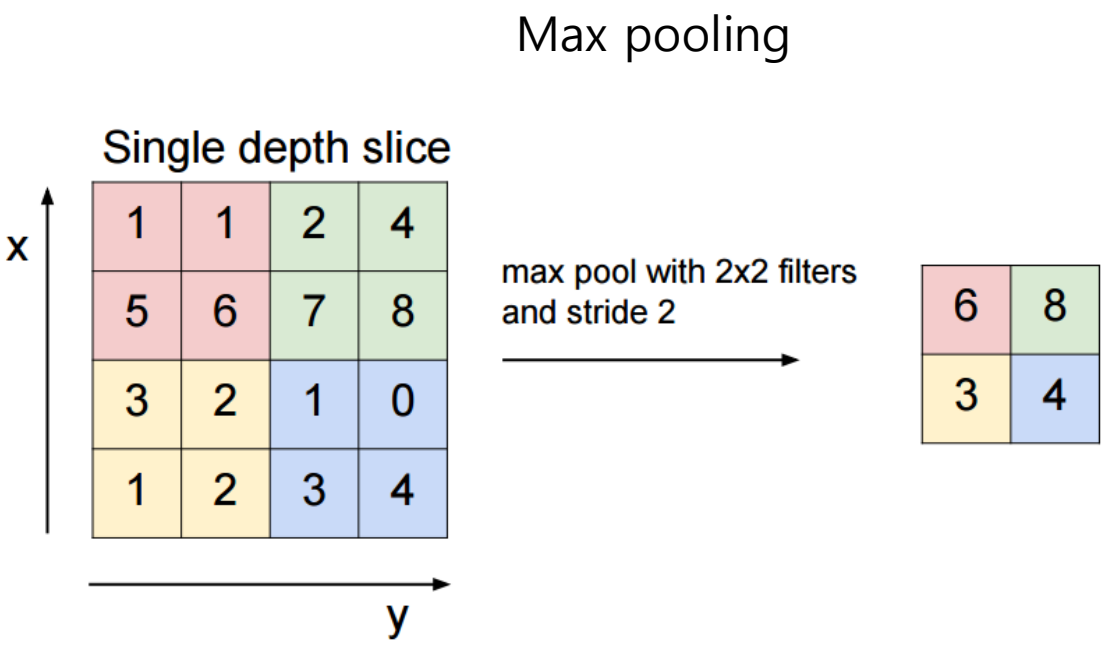


From CS231n

Convolution Neural Network (CNN)

- Pooling Layer

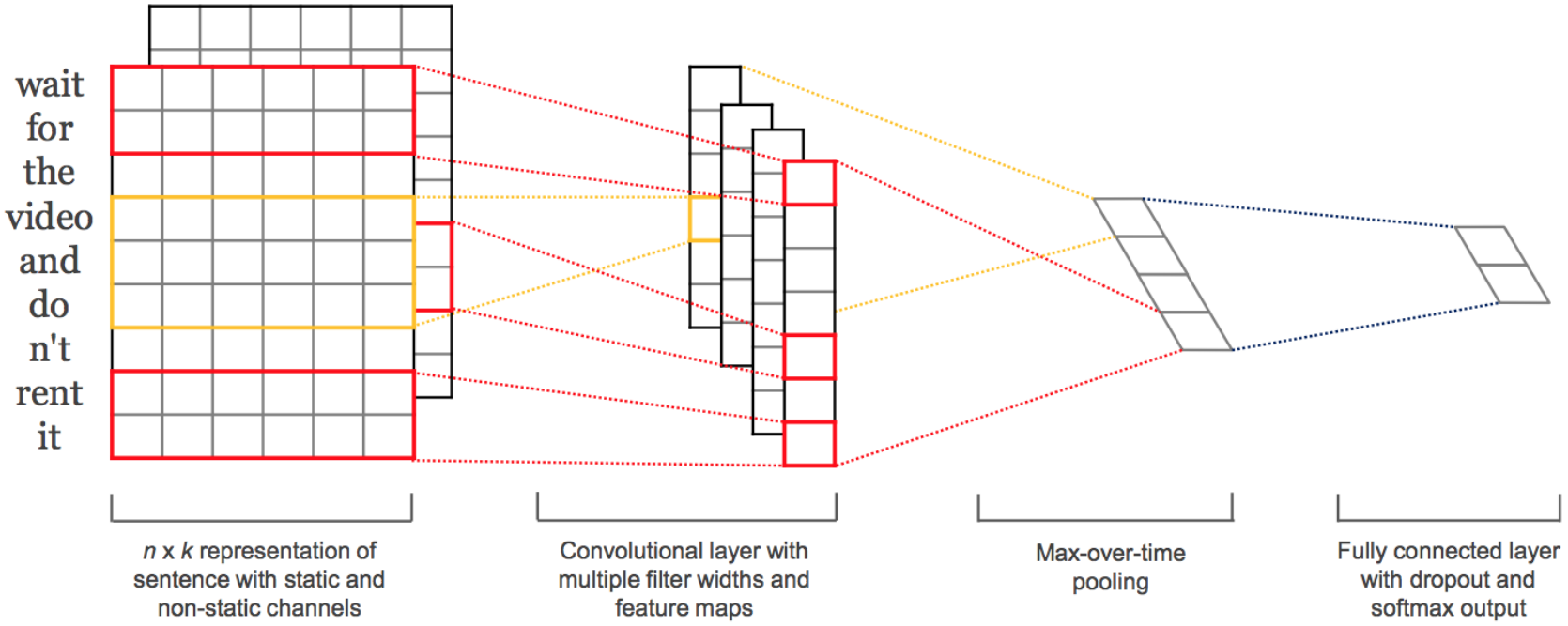




From CS231n

CNN을 활용한 텍스트 분류

- Shows state-of-the-art performance in sentence classification (sentiment analysis, question classification, and so on)



Text to embedded text

class **torch.nn.Embedding**(*num_embeddings, embedding_dim, padding_idx=None, max_norm=None, norm_type=2, scale_grad_by_freq=False, sparse=False, _weight=None*) [source]

A simple lookup table that stores embeddings of a fixed dictionary and size.

This module is often used to store word embeddings and retrieve them using indices. The input to the module is a list of indices, and the output is the corresponding word embeddings.

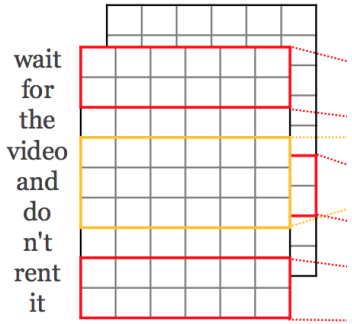
Parameters:

- **num_embeddings** (*int*) – size of the dictionary of embeddings
- **embedding_dim** (*int*) – the size of each embedding vector
- **padding_idx** (*int, optional*) – If given, pads the output with the embedding vector at `padding_idx` (initialized to zeros) whenever it encounters the index.
- **max_norm** (*float, optional*) – If given, will renormalize the embedding vectors to have a norm lesser than this before extracting.
- **norm_type** (*float, optional*) – The p of the p-norm to compute for the max_norm option. Default `2`.
- **scale_grad_by_freq** (*boolean, optional*) – if given, this will scale gradients by the inverse of frequency of the words in the mini-batch. Default `False`.
- **sparse** (*bool, optional*) – if `True`, gradient w.r.t. `weight` matrix will be a sparse tensor. See Notes for more details regarding sparse gradients.

Variables: **weight** (*Tensor*) – the learnable weights of the module of shape (num_embeddings, embedding_dim)

Shape:

- Input: LongTensor of arbitrary shape containing the indices to extract
- Output: *(*, embedding_dim)*, where * is the input shape



B x 100

- Input: Tensor (B x N)
  - B: Batch size
  - N: Max sentence
- Embedding
  - Input (B x N) ➔ Embedded Input (B x N x D)

```
input = torch.randn(10, 100)

m = nn.Embedding(1000,50)
output = m(input)
```

Image example and Convolution layer

class `torch.nn.Conv2d`(*in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True*) [source]

Parameters:

- **in_channels** (*int*) – Number of channels in the input image
- **out_channels** (*int*) – Number of channels produced by the convolution
- **kernel_size** (*int or tuple*) – Size of the convolving kernel
- **stride** (*int or tuple, optional*) – Stride of the convolution. Default: 1
- **padding** (*int or tuple, optional*) – Zero-padding added to both sides of the input. Default: 0
- **dilation** (*int or tuple, optional*) – Spacing between kernel elements. Default: 1
- **groups** (*int, optional*) – Number of blocked connections from input channels to output channels. Default: 1
- **bias** (*bool, optional*) – If `True`, adds a learnable bias to the output. Default: `True`

Shape:

- Input: $(N, C_{in}, H_{in}, W_{in})$
- Output: $(N, C_{out}, H_{out}, W_{out})$ where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 * \text{padding}[0] - \text{dilation}[0] * (\text{kernel_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 * \text{padding}[1] - \text{dilation}[1] * (\text{kernel_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

MNIST image

1 x 28 x 28

ImageNet image

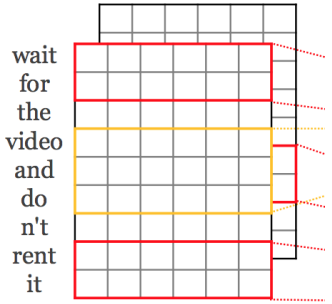
3 x 256 x 256

```
input = torch.randn(20, 1, 28, 28)
# Why 4D

m = nn.Conv2d(1, 16, 5, stride=2)
output = m(input)
```

Image example and Text example

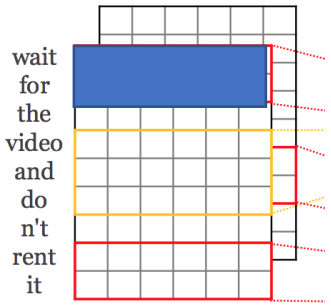
- 1) 텍스트는 채널 정보가 없다
  - Conv2d를 사용하려면, 채널 정보를 추가할 필요가 있다. ➔ Use **unsqueeze**

B x **3** x 256 x 256



B x 100 x 100

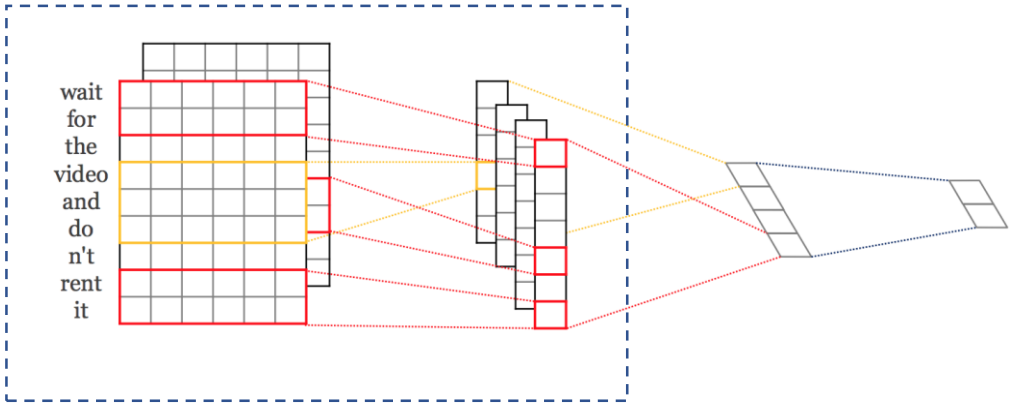
- 2) Convolution (filter)
  - Image: 3 x 3 filter
  - Text: 3 x embedding dim

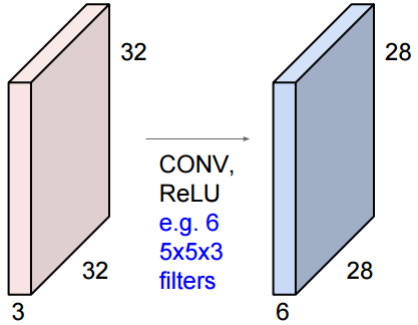



```
input = torch.randn(20, 100, 100)
input = input.unsqueeze(1) # 20 x 1 x 100 x 100

m = nn.Conv2d(1, 100, (3, 100)) # 100 = D

output = m(input) # ? x ? x ? x ?
```

Conv layer with activation function (Text)

```
input = torch.randn(20, 100, 100)
input = input.unsqueeze(1) # 20 x 1 x 100 x 100

m1 = nn.Conv2d(1, 100, (3, 100))
m2 = nn.Conv2d(1, 100, (4, 100))
m3 = nn.Conv2d(1, 100, (5, 100))

output1 = m1(input) # 20 x 100 x 98 x 1
output2 = m2(input) # 20 x 100 x 97 x 1
output3 = m3(input) # 20 x 100 x 96 x 1

output1 = F.relu(output1)
...
```





Pooling layer (Text)

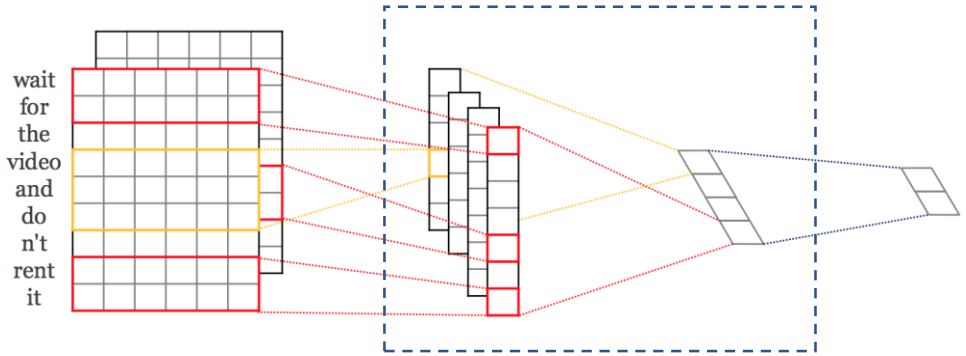
output1 = F.relu(output1) # 20 x 100 x 98 x 1
output2 = F.relu(output2)
output3 = F.relu(output3)

How to pooling? What dims?

p1 = nn.MaxPool1d(output1.size(2))
 or F.max_pool1d(output1.size(2))

output1 = p1(output1) ➔ Error!! Why?
➔ Use squeeze (remove the dimensions of input of size 1)

output1 = p1(output1.squeeze(3)) # ? x ? x ?
…



class `torch.nn.MaxPool1d`(*kernel_size, stride=None, padding=0, dilation=1, return_indices=False, ceil_mode=False*) [source]

Applies a 1D max pooling over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C, L) and output (N, C, L_{out}) can be precisely described as:

$$\text{out}(N_i, C_j, k) = \max_{m=0,\ldots,\text{kernel_size}-1} \text{input}(N_i, C_j, \text{stride} * k + m)$$

If `padding` is non-zero, then the input is implicitly zero-padded on both sides for `padding` number of points. `dilation` controls the spacing between the kernel points. It is harder to describe, but this link has a nice visualization of what `dilation` does.

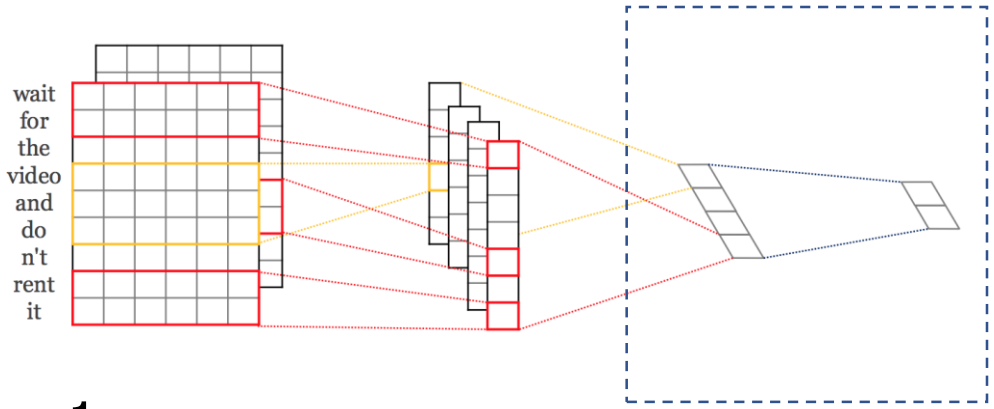
Parameters:

- **kernel_size** – the size of the window to take a max over
- **stride** – the stride of the window. Default value is `kernel_size`
- **padding** – implicit zero padding to be added on both sides
- **dilation** – a parameter that controls the stride of elements in the window
- **return_indices** – if `True`, will return the max indices along with the outputs. Useful when Unpooling later
- **ceil_mode** – when True, will use *ceil* instead of *floor* to compute the output shape

Shape:

- Input: (N, C, L_{in})
- Output: (N, C, L_{out}) where

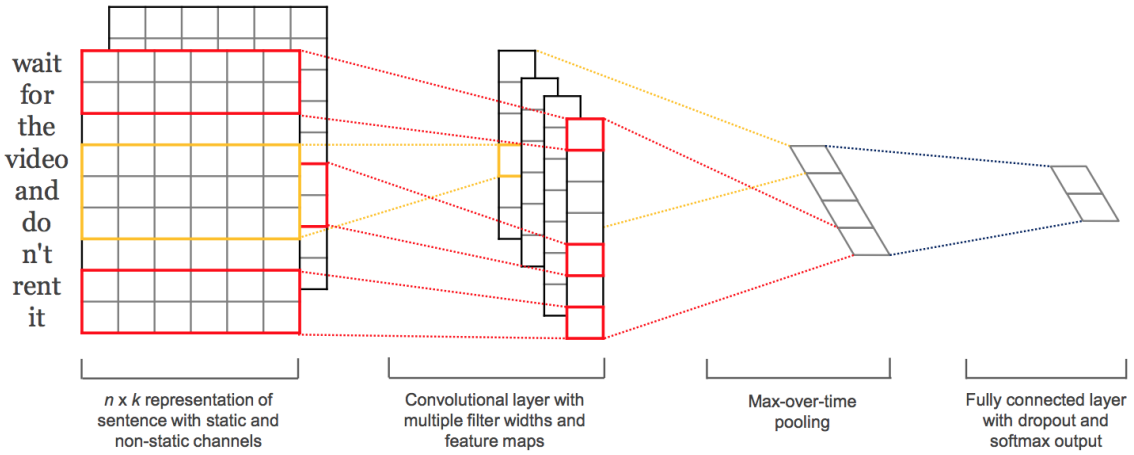
Fc layer (Text)




```
output1 = p1(output1.squeeze(3)) # 20 x 100 x 1
output2 = p2(output2.squeeze(3)) # 20 x 100 x 1
output3 = p3(output3.squeeze(3)) # 20 x 100 x 1

# concatenate (output1, output2, output3)
output1 = output.squeeze(2) # 20 x 100
…
output = torch.cat((output1,output2,output3), 1)

fc = nn.Linear(?, C) # C is class num, ?
out = fc(x)
```

https://github.com/Shawn1993/cnn-text-classification-pytorch

(실습) CNN 기반 감성분석, 의도분석



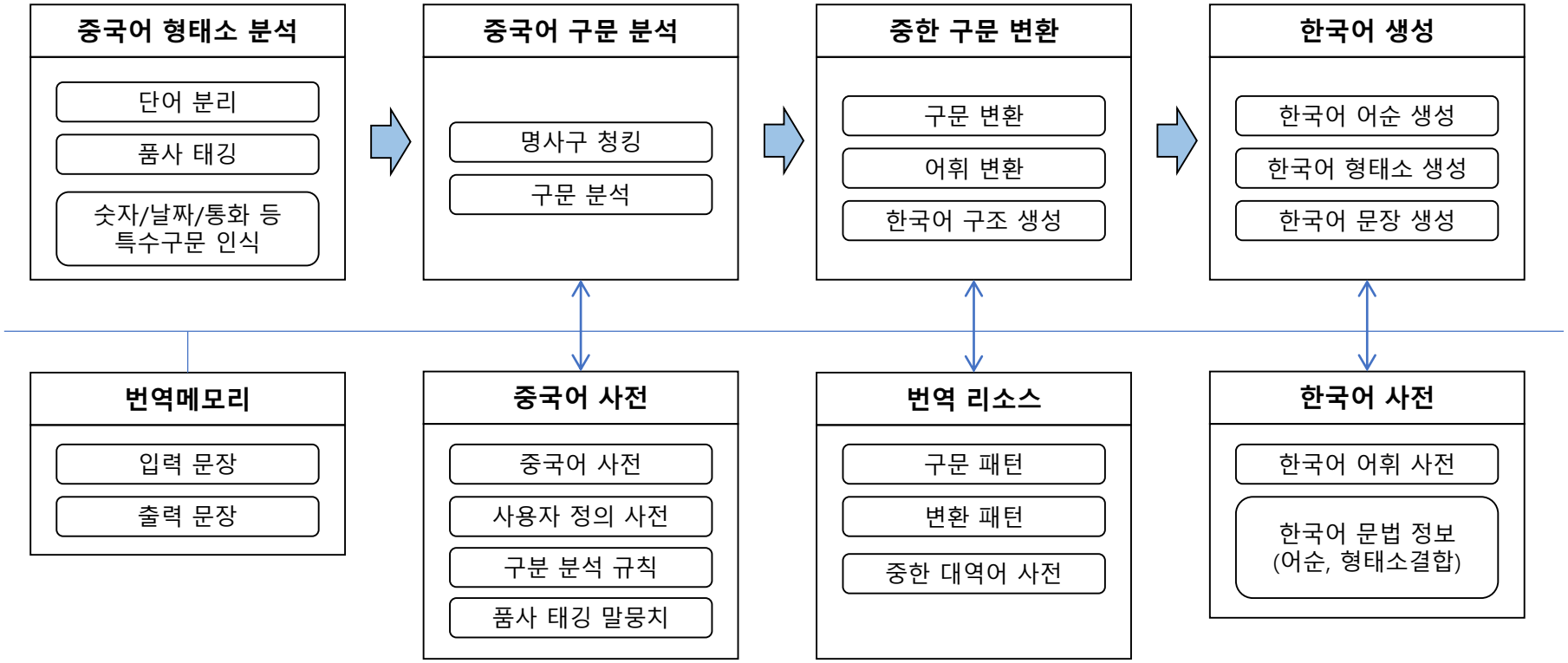

```python
class CNN_Text(nn.Module):

    def __init__(self, embed_num, class_num):
        super(CNN_Text, self).__init__()
        V = embed_num
        C = class_num
        Co = 50 #args.kernel_num
        Ks = [2,3,4]

        self.embed = nn.Embedding(V, 100)
        self.convs1 = nn.ModuleList([nn.Conv2d(1, Co, (K, 100)) for K in Ks])
        self.dropout = nn.Dropout(0.2)
        self.fc1 = nn.Linear(len(Ks)*Co, C)

    def forward(self, x):
        x = self.embed(x)  # (N, W, D)
        x = x.unsqueeze(1)  # (N, Ci, W, D)
        x = [F.relu(conv(x)).squeeze(3) for conv in self.convs1]  # [(N, Co, W), ...]*len(Ks)
        x = [F.max_pool1d(i, i.size(2)).squeeze(2) for i in x]  # [(N, Co), ...]*len(Ks)
        x = torch.cat(x, 1)
        x = self.dropout(x)  # (N, len(Ks)*Co)
        logit = self.fc1(x)  # (N, C)
        return logit
```

기계번역 (Machine Translation)

I love cats. ➔ 나는 고양이를 사랑합니다.

기계번역 시스템 구성도 예

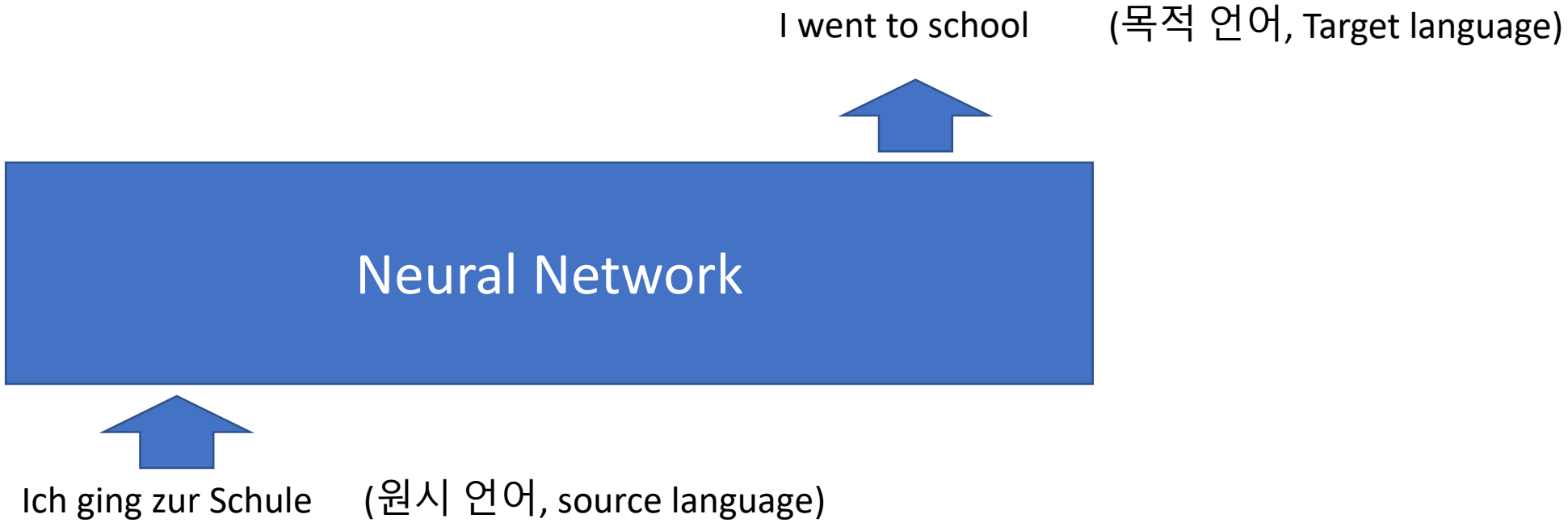
중한 기계번역 시스템 (Rule-Based MT)



Too complicated!!

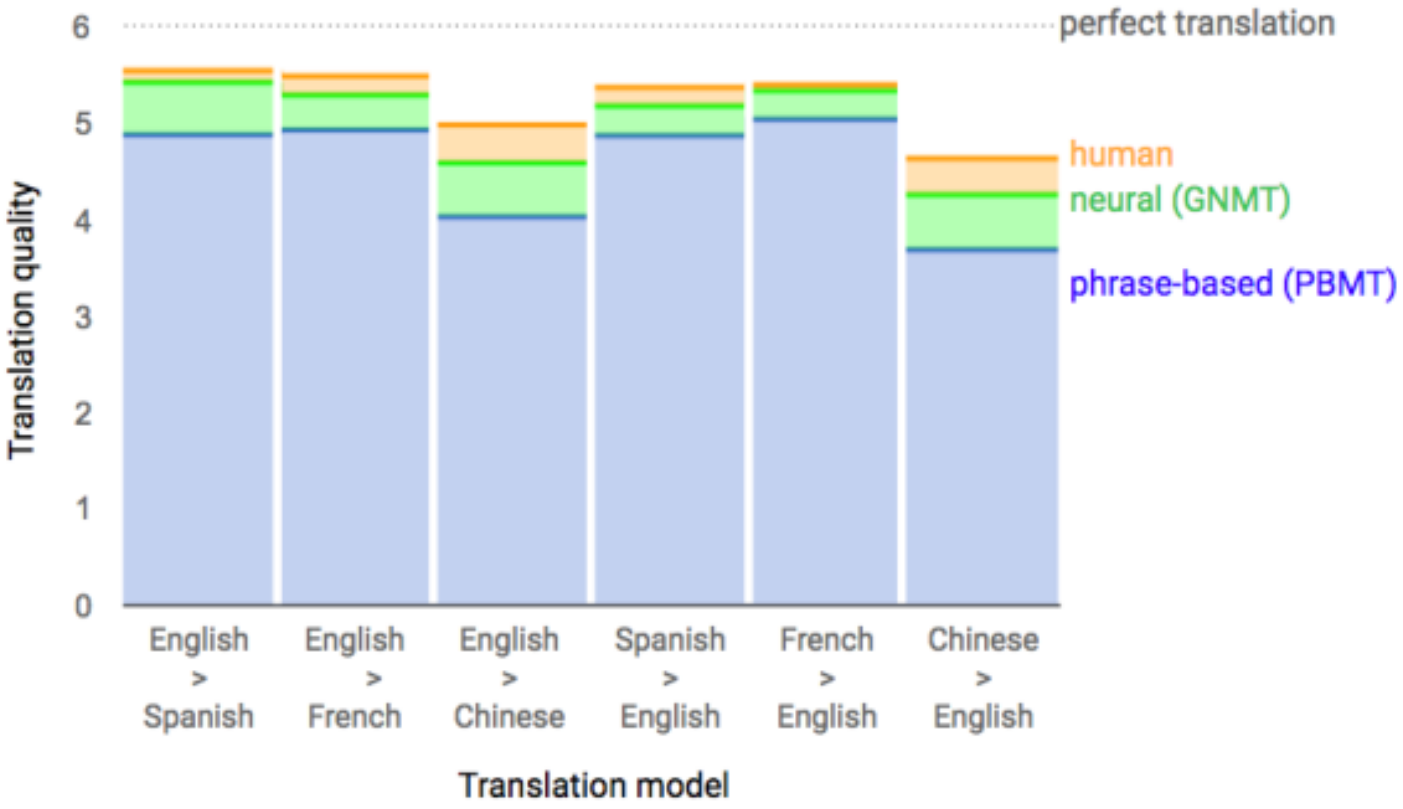
신경망 기계번역 (Neural Machine Translation, NMT)

- The approach of modeling the **entire MT process** via **one big artificial neural network** (End-to-End fashion)

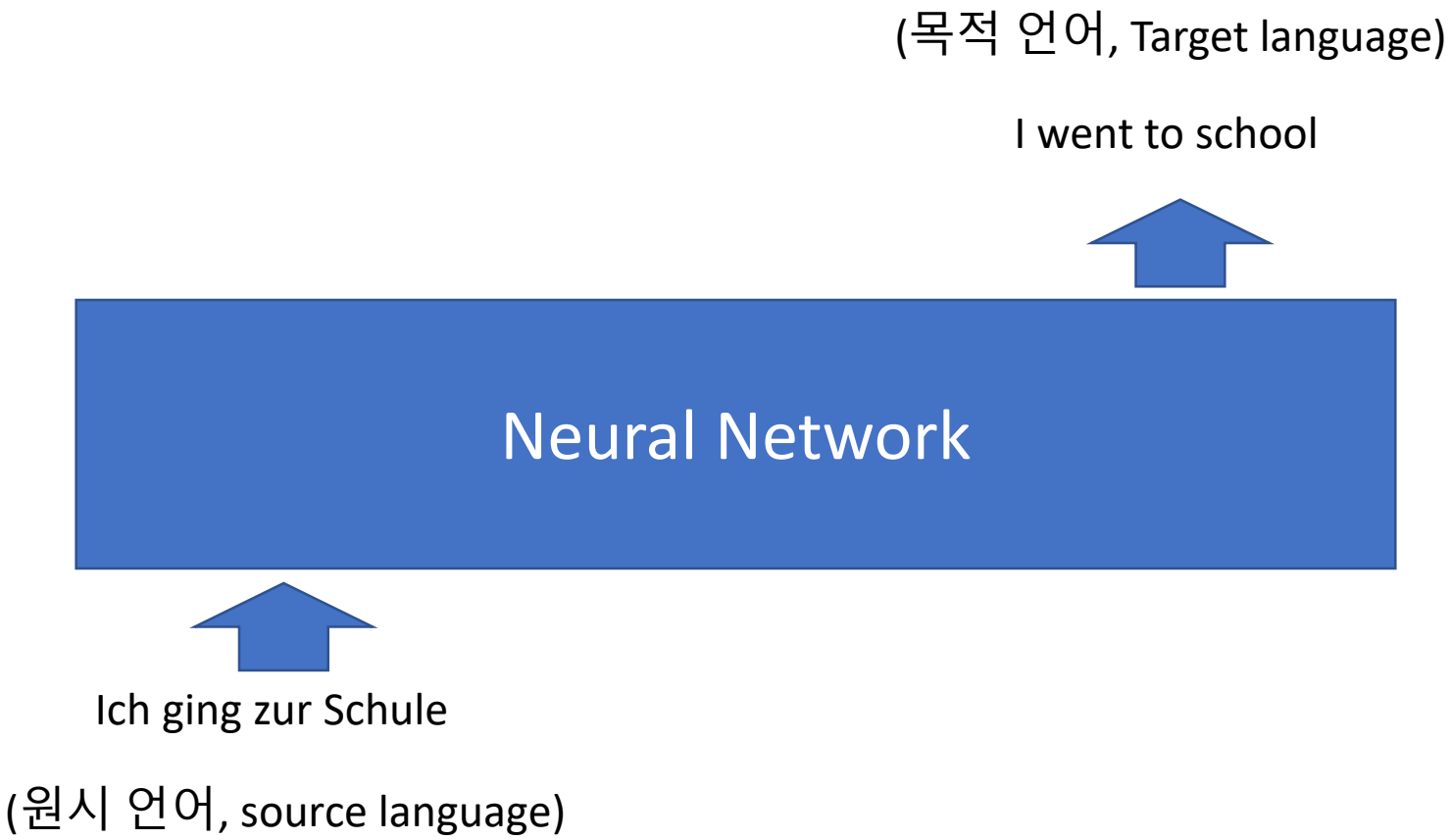


신경망 기계번역

- 딥러닝을 이용한 번역 ➔ 인간수준으로 근접

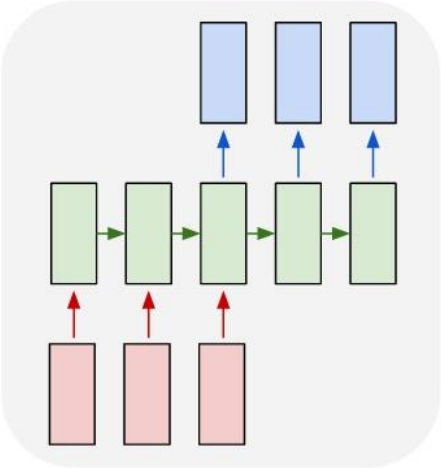


신경망 기계번역


(목적 언어, Target language)
I went to school
Neural Network
Ich ging zur Schule
(원시 언어, source language)

신경망 기계번역

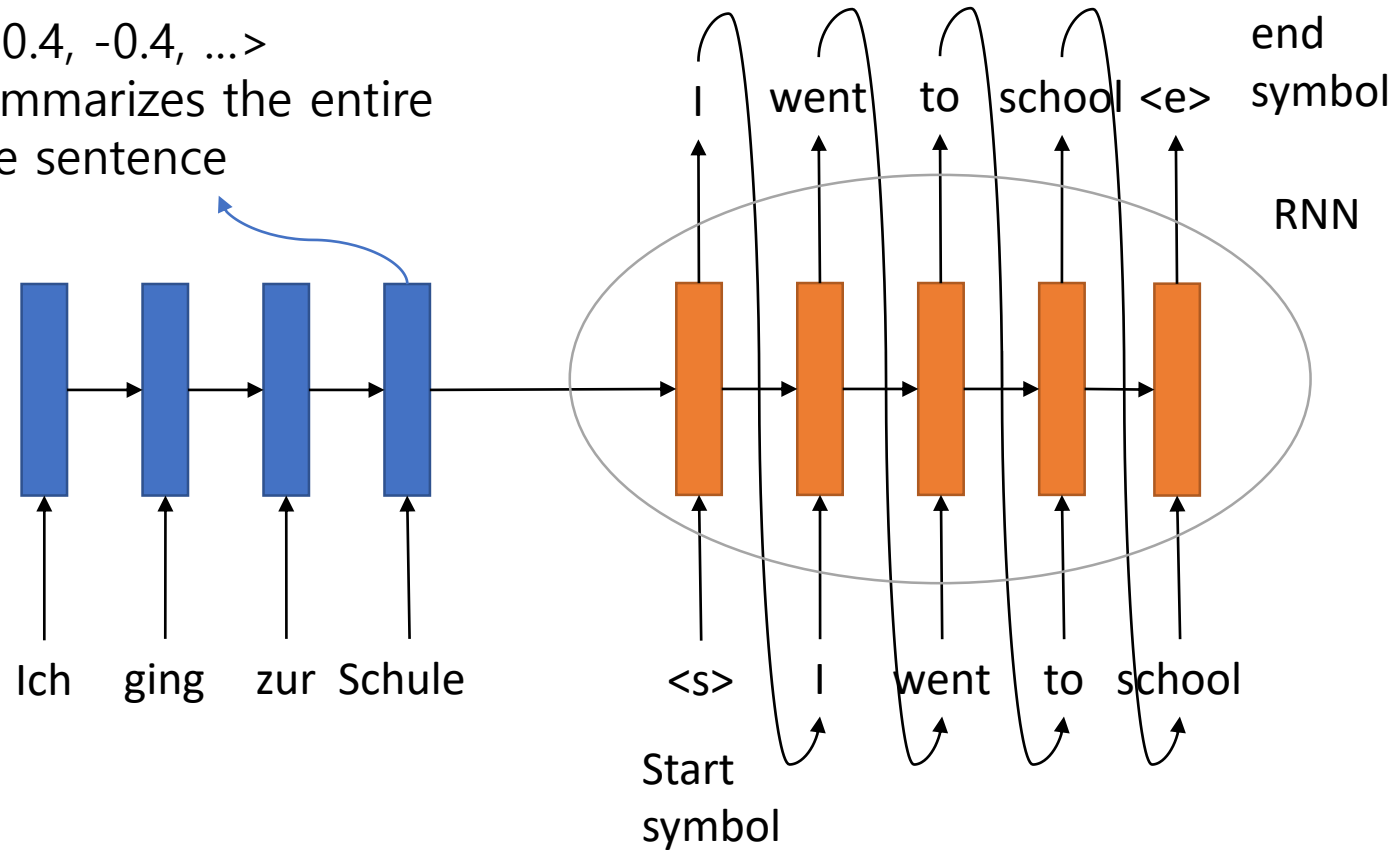
many to many



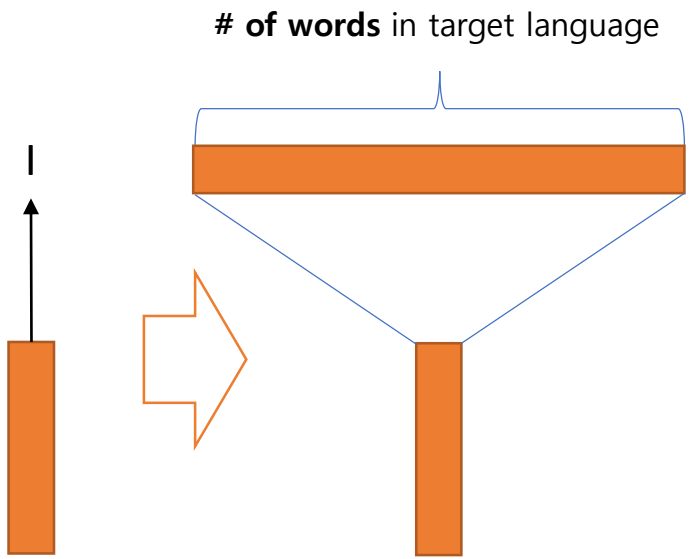
Encoder

<0.1, 0.4, -0.4, ...>
➔ Summarizes the entire source sentence

Decoder



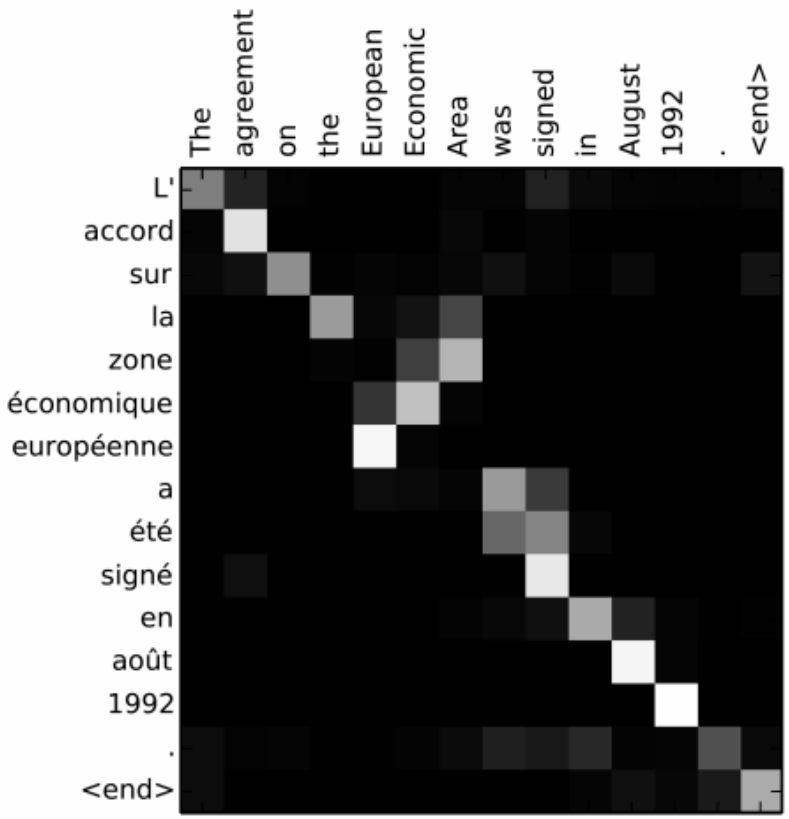
Encoder-Decoder (Seq2Seq)



Vocabulary Generation

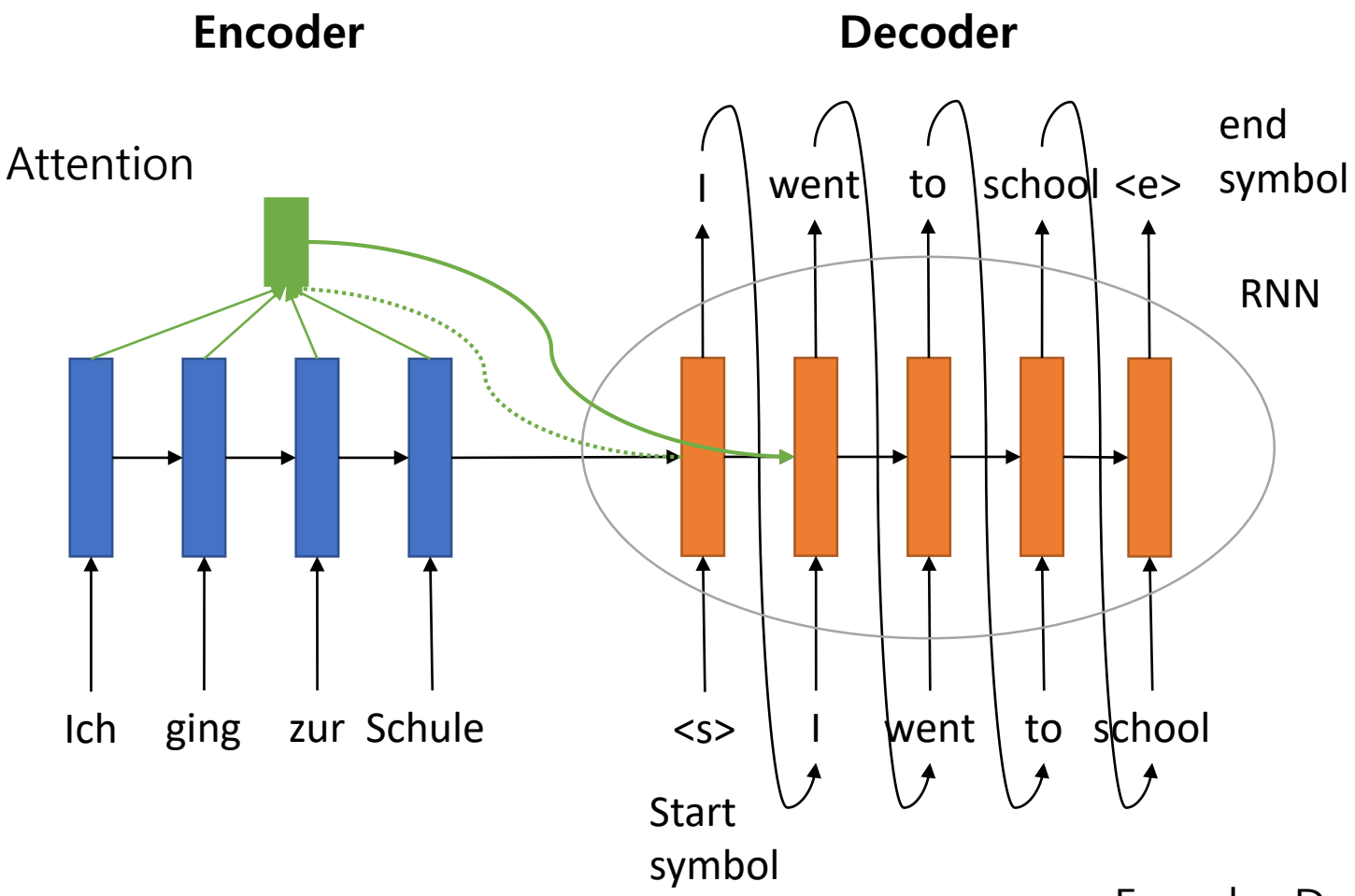
신경망 기계번역

- 원시 언어 문장이 길이에 상관없이 고정된 차원으로 표현 (500 dims)
  - Summarizes the entire source sentence
- 긴 문장을 번역할 경우, 성능 하락
- 목적 언어로 번역시, 원시 언어의 특정 부분에 더 포커스가 되도록
- **Attention mechanism**



Neural Machine Translation by Jointly Learning to Translate and Align, ICLR 2015

Attention mechanism



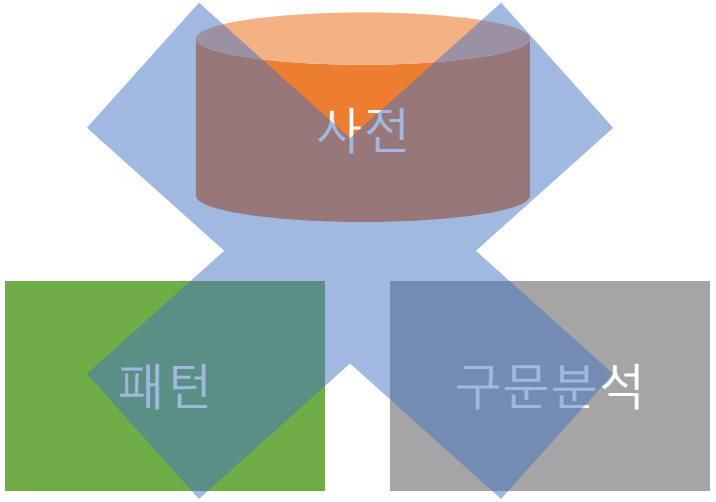
Encoder-Decoder (Seq2Seq)

(실습) 심플한 영한 번역기 만들어봅시다!

1. 데이터 준비
   - 영어-한국어 병렬 말뭉치

2. 데이터 전처리

3. NMT 모델 학습

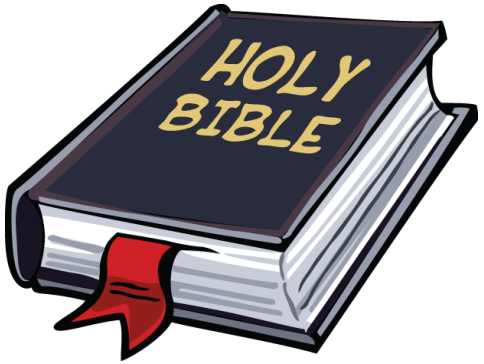


병렬 말뭉치

국립국어원 언어정보나눔터

대본 – 자막

European Parliament



(영한) 병렬 말뭉치

TED

WIT3

Web Inventory of Transcribed and Translated Talks

- https://wit3.fbk.eu/mono.php?release=XML_releases&tinfo=cleanedhtml_ted

de, German	download	km, Central Khmer	download
el, Greek	download	kn, Kannada	download
en, English	download	ko, Korean	download
eo, Esperanto	download	ku, Kurdish	download
es, Spanish, Castilian	download	ky, Kirghiz, Kyrgyz	download

(영한) 병렬 말뭉치

A public, Dewey long ago observed, is constituted through discussion and debate.

If we are to call the tyranny of assumptions into question, and avoid doxa, the realm of the unquestioned, then we must be willing to subject our own assumptions to debate and discussion.

듀이 는 오래전에 사회는 논의와 토론을 통해 구성이 된다고 관측하였습니다.

우리가 여러가지 가설의 횡포에 대해 의문을 제기하려면, 그리고 토론없이 사실로 받아드리는 "진리"를 피하려면 우리 자신의 가설들도 토론과 논의의 대상으로 할 의지가 있어야 합니다.

	English	Korean
전체 문장 개수	**195,818**	
전체 단어 개수	3,671,025	2,101,301 (4,030,020)
유니크 개수	61,484	58,137

데이터 전처리

- 병렬 코퍼스를 토큰화

A public, Dewey long ago observed, is constituted through discussion and debate.
➔ A public , Dewey long ago observed , is constituted through discussion and debate .

듀이는 오래전에 사회는 논의와 토론을 통해 구성이 된다고 관측하였습니다.
➔ 듀 이 는 오래전 에 사회 는 논의 와 토론 을 통해 구성 이 된 다고 관측 하였 습니다 .

- 문장 쌍 구축

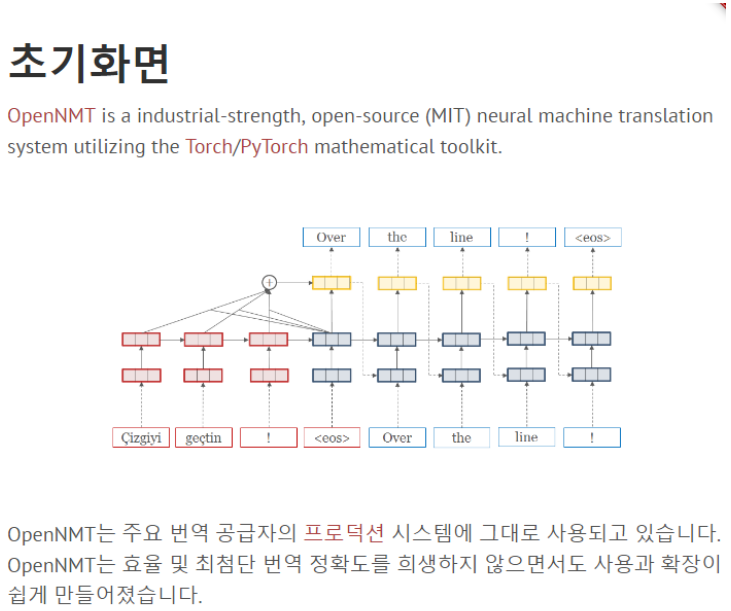
A public , Dewey long ago observed , is constituted through discussion and debate . — 듀 이 는 오래전 에 사회 는 논의 와 토론 을 통해 구성 이 된 다고 관측 하였 습니다 .

OpenNMT (기계번역 오픈툴)

- http://opennmt.net/
  - (한국어 버전: http://ko.opennmt.net/)
  - **systran** + Harvard NLP groups
  - **최신 기계번역 기술들 구현**
  - **Easy to use**
  - Language: Lua

- PyTorch version (python)
  - https://github.com/OpenNMT/OpenNMT-py

- Tensorflow version (python)
  - https://www.tensorflow.org/tutorials/seq2seq
  - https://github.com/OpenNMT/OpenNMT-tf



학습

- 일반적으로 GPU가 필요

- 학습시간
  - 옵션과 모델 크기에 따라 달라짐
  - 기본 파라미터를 사용한 경우 **> 5시간 (980ti)**
  - GPU를 사용해도 생각보다 시간이 오래 걸림

결과

```
src input :  I love cats .
result :  ['전 ', '고 양 이 ', '를 ', '사 랑 합 ', '니 다 ', '.']
전  고 양 이  를  사 랑 합  니 다  .
input some sentence:
Self-driving vehicles will play a crucial role in improving transportation safety .
src input :  Self-driving vehicles will play a crucial role in improving transportation safety .
result :  ['운 전 ', '차 량 ', '은 ', '교 통 ', '안 전 ', '을 ', '향 상 ', '시 키 는 ', '중 요 한 ', '역 할 ', '을 ', '할 ', '것 ', '입 니 ', '다 ', '.']
운 전  차 량  은  교 통  안 전  을  향 상  시 키 는  중 요 한  역 할  을  할  것  입 니  다  .
input some sentence:
You ' ll meet him again if it ' s meant to be .
src input :  You ' ll meet him again if it ' s meant to be .
result :  ['여 러 분 ', '은 ', '이 ', '것 ', '이 ', '있 다 ', '면 ', '다 시 ', '그 ', '를 ', '만 날 ', '것 ', '입 니 ', '다 ', '.']
여 러 분  은  이  것  이  있 다  면  다 시  그  를  만 날  것  입 니  다  .
input some sentence:
I would like to book a flight to New York .
src input :  I would like to book a flight to New York .
result :  ['저 ', '는 ', '뉴 욕 ', '의 ', '비 행 기 ', '에 ', '대 한 ', '책 ', '을 ', '좋 아 합 ', '니 다 ', '.']
저  는  뉴 욕  의  비 행 기  에  대 한  책  을  좋 아 합  니 다  .
```

- 성능은 ...
  - < 200,000 병렬 문장으로 만든 번역

- 신경망 기계번역에서 중요한 것
  - **좋은 퀄리티의 병렬 말뭉치 (>2,000,000)**

이슈 - Robustness

한국어 감지

이 그룹은 모두 합쳐 25명이었다.

영어

All together, there were 25 people in the group.

한국어 감지

이 그룹은 모두 합쳐 26명이었다.

영어

The group had a total of 26.

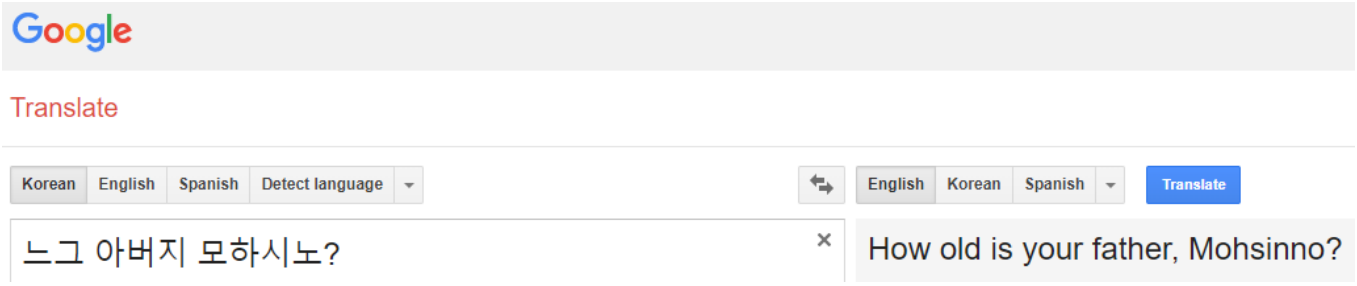
한국어 감지

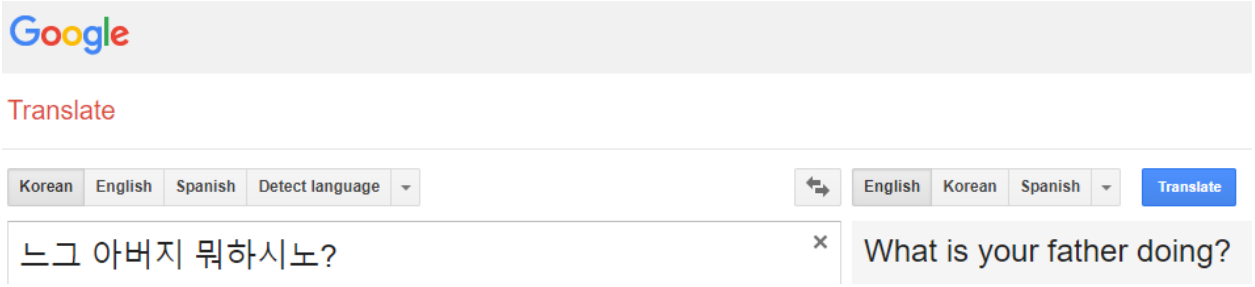
이 그룹은 모두 합쳐 100명이었다.

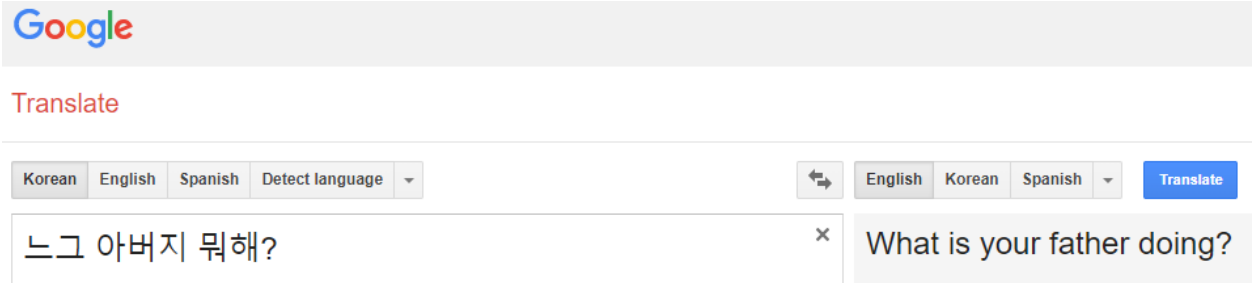
영어

The group numbered 100 in total.

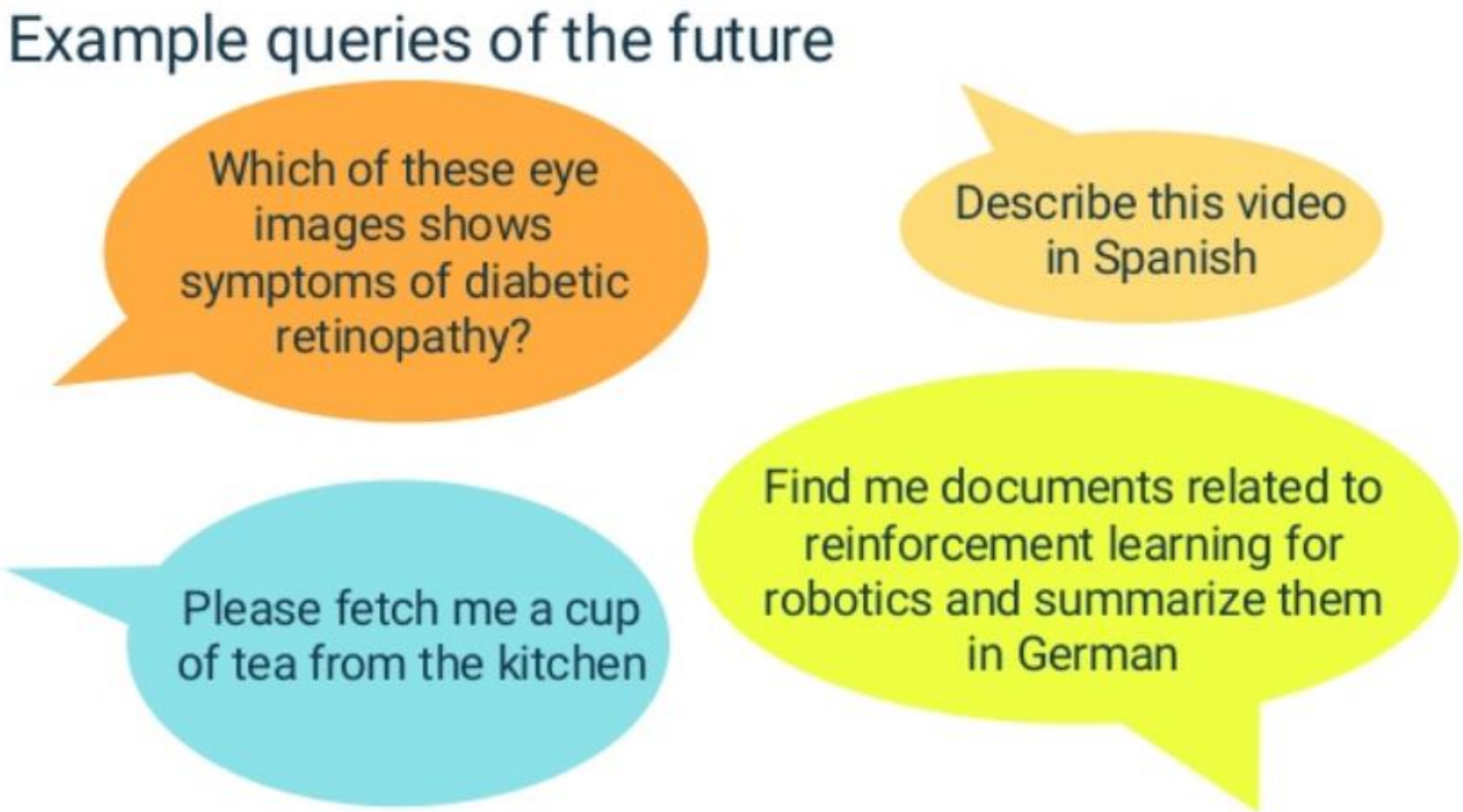
이슈 - Robustness







NLP와 ML이 실현해줄 가까운 미래



Jeff Dean at Jan. 2017 AI Frontiers Conference
(https://www.slideshare.net/AIFrontiers/jeff-dean-trends-and-developments-in-deep-learning-research)

End of Document