2025-2 딥러닝응용프로그래밍 기말고사 (20241519 조예성)

이미지 상 물체 인식: 객체인식
- 역할 2개: Bounding Box 제공, 객체 클래스 식별

위치 찾기 다른 말: 지역화; Localization

객체인식 1단계 종류: YOLO, SSD

객체인식 2단계 종류: R-CNN, Fast-R-CNN, Faster-R-CNN

모델 한 번에 객체인식 하는 것: 1단계

분류와 검출을 따로 진행하는 것: 2단계

이미지를 n*n 그리드로 나누고, 중심점 보유 객체 유무를 파악하는 알고리즘: YOLO

한번 예측하여 다양한 크기 및 비율을 예측하는 알고리즘: SSD
- 고해상도 feature Map 강한 부분: 작은 물체
- 저해상도 Feature Map 강한 부분: 큰 물체

초기 영역을 통합하여 객체가 있을 법한 부분만 인식하는 알고리즘: R-CNN
- 단계: 초기 영역 생성, 작은 영역 통합, 후보 영역 생성, 모양 맞추기
- 단점: 바운딩 박스가 많음에 따라 메모리 사용량이 많다.

기존 CNN의 입력 크기가 동일해야 하여 발생하는 문제 해결: 공간 피라미드 풀링; SPP
- 순서: 합성곱 -> MaxPooling -> 연결하기 -> 완전연결층 -> BoundingBox, SVM
- 적용한 망: SppNet
- 겹치는 BoundingBox: NMS; Non Maximum Suppression

이미지 합성곱 한번 통과 후 나온 특징 맵을 기반으로 처리하는 신경망: Fast R-CNN
- 하는 처리: ROI 풀링
    - 기능 서술: 합성곱 통과 특징맵을 n*n으로 MaxPooling
- 순서: 합성곱 -> ROI 풀링 -> 완전 연결층 -> BoundingBox, SVM

후보 생성까지 CNN내부에서 하는 것: Faster R-CNN
- 기존 Fast R-CNN서 추가된 것: RPN; Region Proposal Network
- RPN 위치: 합성곱 마지막
- 비율 고정 문제 해결: anchor box
- 미리 정한 (레퍼런스 박스)를 활용한다

신경망 훈련 통해 이미지를 픽셀 단위 분할: 이미지 분할
- 핵심: FCN, Convolutional, Deconvolutional Network, U-Net, PspNet, DeepLab

FCN 설명: 기존 합성곱 신경망의 완전연결층을 1x1 합성곱으로 대체하여 Feature Map 출력

역 합성곱 방법: 각 픽셀 주변에 ZeroPadding

의학 사용 많고 Trade-off 하지 않는 신경망: U-Net

U-Net 활용 인식 방식: 패치인식방식

이미지 안 객체 위치 정보 출력: 지역화

U-Net 핵심 구성: 수축 경로, 확장 경로, Skip Connection

기존 모델의 작은 패턴만 보고 분석하는 단점 극복하기 위한 신경망: PSPNet
- 구조: BackBone CNN -> BackBone CNN, Final Feature Map -> PPM -> Segmemtation Head -> Upsampling -> Final Segmentation Map

시간의 흐름에 따라 수집된 데이터: 시계열 자료

시간에 따라 변하는 데이터를 사용하여 추이를 분석: 시계열 분석

예측 불가능한 돌발적 요인으로 인한 변동: 불규칙 변동

장기적인 변화 추세: 추세 변동

2~3년 단위 반복으로 인한 파동적 변화: 순환변동

1년 주기 변동: 계절 변동

명확한 패턴 존재: 규칙적 시계열 패턴 거의 없고 변동성 큰 데이터: 불규칙적 시계열

시계열 분석 모델: AR, MA, ARMA, ARIMA
- 자기회귀: AR, 이동평균: MA, 두게 합에 차분: ARIMA

현재 값이 과거 값들의 선형 결합으로 설명되는 모델: AR모델
- 이 모델에서 p의 의미: 사용하는 과거의 수

예측이 불가능한 오차: 백색잡음

관성이 있는 현실의 많은 데이터 다루기 적합한 모델: AR모델

현재 값이 과거 오차의 누적 효과로 설명되는 모델: MA 모델

충격이 서서히 (사라짐) 값:b

데이터의 흐름과 충격의 잔향이 동시에 작용한다: ARMA

추세 제거 후 과거 값과 충격의 패턴 적용: ARIMA
- 수식: ARIMA(p, d, q)
  - d: 차분 횟수

**차분을 하는 이유**: 추세가 있는 데이터에서 추세를 제거하기 위함
- 결과: 변화량 만 남음

적합한 경우: 장기적 추세가 강한 데이터
부적합한 경우: 계절성 강함, 변동 폭 매우 작음

p: 과거 값 참고 개수, i: 차수 횟수, q: 과거 예측 오차 참고 개수

시퀀스 데이터 처리 위한 신경망: RNN

특징: 같은 가중치를 모든 시점에서 공유

$h_1, h_2, h_3$ 뜻: 각 시점의 **은닉상태**

RNN 종류: 1:n, n:1, n:n

RNN 문제: 기울기 소실, 장기 의존성

입력 텐서를 시간 축 방향으로 잘라서 변환해주는 함수: unstack()

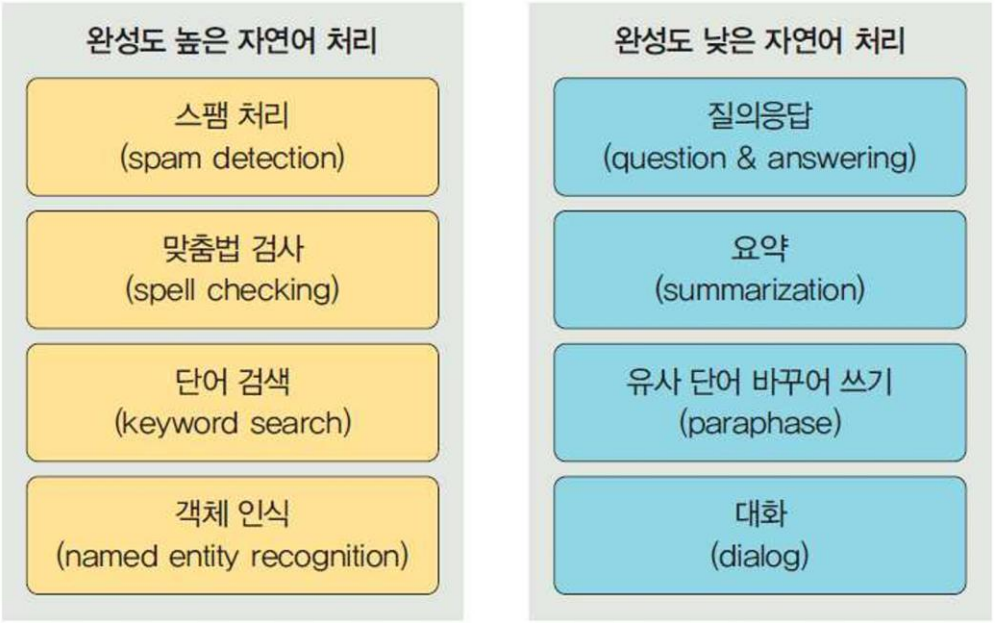
RNN 장기 기울기 소실과 장기 의존성 문제 해결: LSTM
- 도입된 개념: cell state

LSTM 4가지 게이트: Forget Gate, Candidate Gate, Input Gate, **Output Gate**

Output Gate 출력 내용: hidden state

2개의 게이트만 갖는 구조: GRU
- 게이트: Update Gate, Reset Gate
- 사용하는 스테이트: Hidden state

과거 뿐만이 아닌 미래 시점 데이터 참고까지: 양방향 RNN




말뭉치 다른 용어로: 코퍼스

문서 나누는 단위: 토큰
- 만들기: 토큰화

단어 분석에 관계는 없지만, 등장 자주: 불용어

단어를 기본 형태로 만드는 작업: 어간 추출

주어진 문장에서 품사를 식별하기 위해 붙여주는 태그: 품사태깅
- 품사태깅 정보: Det, Noun, Verb, Prep

품사 태깅 라이브러리: NLTK

자연어 처리 과정: 입력, 전처리, 임베딩, 모델 활용 예측

전처리 종류: 토큰화, 불용어 제거, 어간 추출, 정규화

의미를 갖는 최소 단위 명칭: 형태소

어간 추출과 표제어 추출 차이점: 단어에 대한 사전 지식 필요 여부

표현 방법이 다른 단어를 통합시켜 같은 단위로 만들기: 정규화
- 값이 크다고 분석에 더 중요한 요소라고 간주할 수는 없기에

```
std_scale = preprocessing.StandardScaler().fit(train_norm)
```

평균0, 분산 1로 정규화: StandardScaler()

사분위수활용: RobustScaler()

최대 최소: MinMaxScaler()
- 유사: MaxAbsScaler

자연어를 컴퓨터가 이해가능한 언어 형태인 벡터로 변환한 결과 혹은 과정: 임베딩

대부분의 값이 0으로 채워진 임베딩: 희소 표현 기반 임베딩
- 예시: 원 핫 인코딩

출연 빈도수 이용 인코딩 하여 벡터화: 카운터벡터
- 라이브러리, 클래스: Scikit-Learn, Count Vectorizer()
- 기능 3개: 문서를 토큰 리스트로 변환, 토큰 출현 빈도 카운트, 문서 인코딩 후 벡터화

정보 검색론서 가중치 구할 때 사용하는 알고리즘: TD-IDF
- 문서 내 단어 등장 빈도: TF
- 흔하지만 중요하지 않은 단어 빈도: IDF
- 결과: 문서 간 코사인 유사도 행렬

```
[[1.       0.224325 0.      ]
 [0.224325 1.       0.      ]
 [0.       0.       1.      ]]
```

신경망 구조를 활용한 임베딩: 예측기반 임베딩
- 대표적: Word2Vec

문서의 문장단위 분리 메소드: sent_tokenize()

문장을 단어 단위로 분리: word_tokenize()

주변 단어 보고 중심 단어 예측: CBOW

CBOW 반대로 특정 단어로 문맥단어 예측: skip-gram

기존 워드 투 벡터의 단점 보완: 페스트텍스트

글로벌 동시 발생확률: 글로브
- 포함: 통계정보 + skip gram

어텐션 구조: Encode, Decoder

2018년 말 구글 공개한 AI 언어 모델: BERT

인코더 블록 구조: 셀프 어텐션, 전방향 신경망

새로운 이미지를 만들어 내는 모델: 생성 모델
- 종류: GMM, 은닉 마르코프 모델, 오토인코더, 변분 오토인코다, GAN, Diffusion Model

오토인코더 진행과정: 입력 -> latent vector -> 복원
- 벡터가 있는 공간: 잠재공간, latent space

오토인코더 중요한 이유: 데이터 압축, 차원의 저주 예방, 특성 추출

오토인코더 + 확률적 분포: 변분 오토인코더
- 장점: 생성되는 이미지가 안정적
- GAN 보다 이미지가 (흐림)

VAE 동작: 확률 분포 제작 -> 가우시안 분포 활용 출력데이터

코드 13-11 인코더와 디코더 네트워크 생성

```
encoder = [
    tf.keras.layers.InputLayer(input_shape=(28,28,1)),
    tf.keras.layers.Conv2D(
        filters=32, kernel_size=3, strides=(2,2), activation="relu"
    ),
    tf.keras.layers.Conv2D(
        filters=64, kernel_size=3, strides=(2,2), activation="relu"
    ),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units=2*2),
] ------①


decoder = [
    tf.keras.layers.Dense(units=7 * 7 * 64, activation="relu"),
    tf.keras.layers.Reshape(target_shape=(7,7,64)),
    tf.keras.layers.Conv2DTranspose(
        filters=64, kernel_size=3, strides=(2,2), padding="SAME", activation="relu"
    ),
    tf.keras.layers.Conv2DTranspose(
        filters=32, kernel_size=3, strides=(2,2), padding="SAME", activation="relu"
    ),
    tf.keras.layers.Conv2DTranspose(
        filters=1, kernel_size=3, strides=(1,1), padding="SAME", activation="sigmoid"
    ),
] ------②
```

생성자 판별자 경쟁: GAN
생성자는 인코더, 디코더 중: 디코더
판별자는 인코더, 디코더 중: 인코더

이미지 생성 원리: 노이즈 데이터 주입

생성자가 만든 이미지를 디코더가 원본이라고 판별하면 손실:
생성자 손실: ▼, 판별자 손실: ▲
생성자가 만든 이미지를 인코더가 가짜라고 판별하면 손실:
생성자 손실: ▲, 판별자 손실: ▼

GAN 진행 과정:
노이즈 생성 -> 가짜 이미지 생성 -> 예측 -> 업데이트