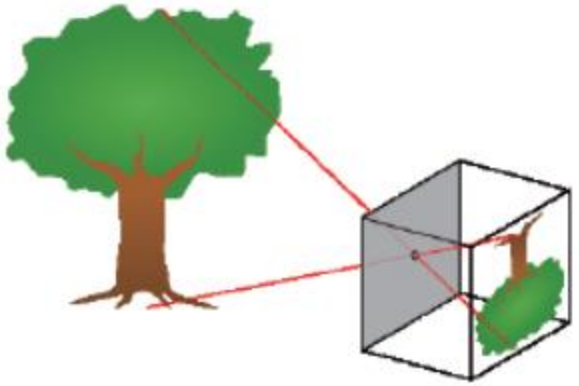
이미지 분석

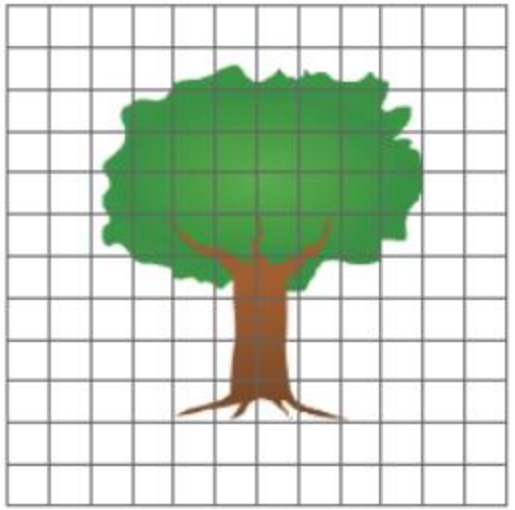
디지털 영상의 획득과 표현

샘플링과 양자화

- 2차원 영상 공간을 $M*N$으로 샘플링 ($M*N$을 해상도라 부름)
- 명암을 L 단계로 양자화 (L을 명암 단계라 부름, 즉 명암은 $[0,L-1]$ 사이 분포)
- 아래 예) M=12, N=12, L=10인 경우



(a) 핀홀 카메라 모델



(b) 샘플링과 양자화

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	3	4	2	3	4	3	0	0	0
0	0	3	7	8	8	8	7	6	3	0	0
0	0	4	8	9	9	9	8	7	5	1	0
0	0	4	7	8	9	9	8	7	5	0	0
0	0	3	6	7	8	8	7	7	3	0	0
0	0	0	2	4	7	8	4	3	0	0	0
0	0	0	0	0	4	7	0	0	0	0	0
0	0	0	0	0	5	6	0	0	0	0	0
0	0	0	0	2	3	4	2	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

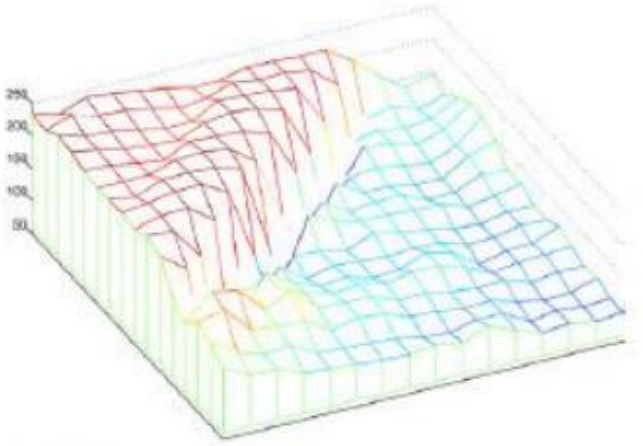
(c) 디지털 영상

디지털 영상의 획득과 표현

(a) 영상

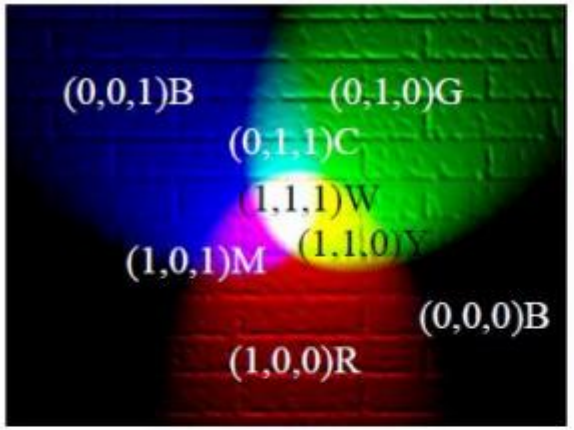
233	224	239	235	230	224	215	215	226	225	223	223	196	163	135	147
229	244	233	231	223	214	219	230	226	229	222	191	127	137	122	136
243	232	229	223	214	215	237	235	232	226	167	122	131	124	129	151
237	231	223	219	216	234	240	235	223	146	81	136	132	120	134	164
231	229	222	217	235	234	231	218	148	91	121	126	120	112	128	164
225	225	226	237	240	235	206	111	70	142	119	118	111	111	134	147
229	222	239	240	238	225	97	99	145	119	124	125	108	110	129	123
228	234	241	242	220	112	99	153	136	126	126	121	122	108	115	124
225	234	236	208	78	73	125	121	112	130	120	115	107	102	111	111
236	232	185	86	95	139	111	121	116	114	116	116	103	104	112	110
225	197	85	110	160	137	119	124	113	115	132	122	93	105	106	122
183	125	157	169	155	140	130	133	124	133	133	119	102	107	110	112
164	203	195	156	174	138	137	136	119	122	114	108	112	98	104	102
188	196	156	150	150	125	134	129	116	113	108	111	99	91	93	106
176	152	138	142	120	118	117	113	104	102	112	111	90	96	93	94
158	137	138	122	117	114	111	110	113	108	122	107	93	98	90	94

(b) 숫자 배열



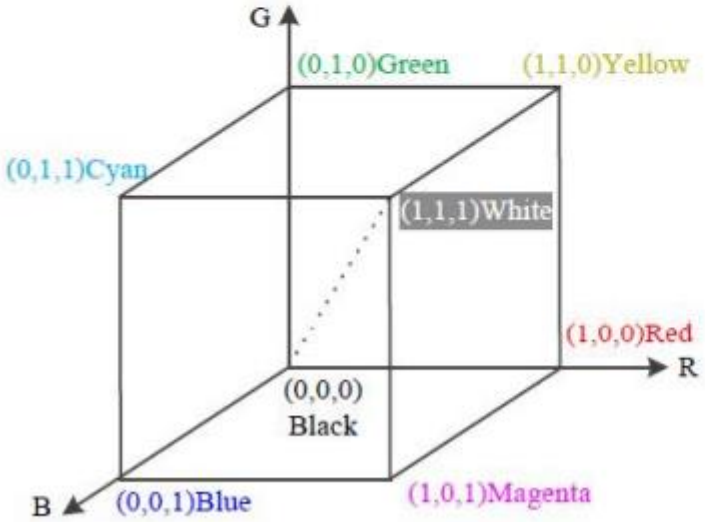
(c) 지형

컬러 영상 (RGB, 3차원)

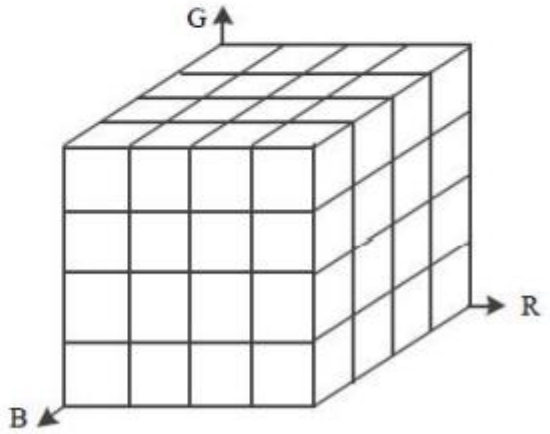


(1,0,0)=Red
(0,1,0)=Green
(0,0,1)=Blue
(0,1,1)=Cyan
(1,0,1)=Magenta
(1,1,0)=Yellow
(0,0,0)=Black
(1,1,1)=White

(a) 빛의 혼합



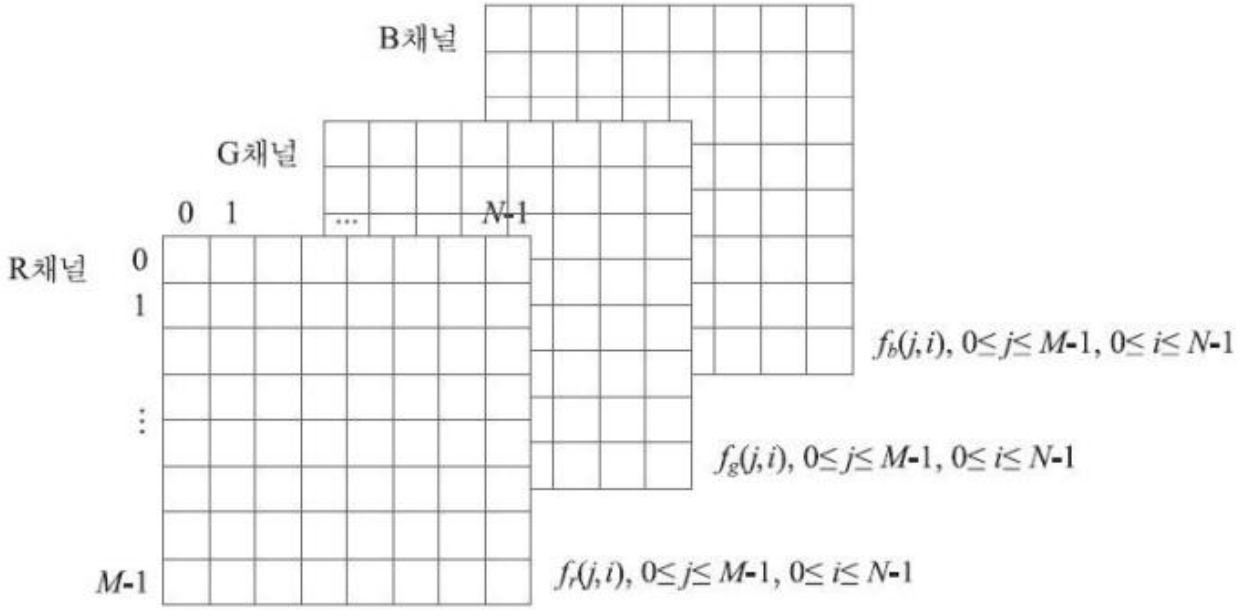
(b) RGB 큐브



(c) 양자화된 4×4×4 RGB 큐브

컬러 영상 (RGB, 3차원)

- f_r f_g f_b의 세 채널로 표현



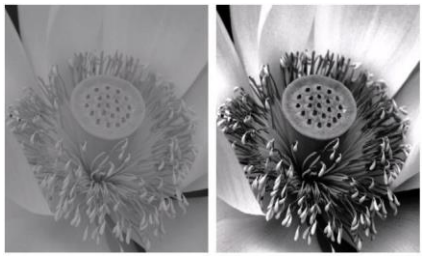
> RGB 영상(원래 영상) > f_r > f_g > f_b

영상처리(Image Processing)

- 디지털 영상을 수정 및 조작하는 것으로 영상에 여러 연산이나 기술을 적용하여 사용자가 원하는 결과를 새롭게 얻어내는 과정

  - 영상을 더욱 높은 질의 영상으로 만들거나 일그러뜨리는 것
  - 영상의 두드러진 특징을 더욱 두드러지게 하는 것
  - 다른 영상의 일부분으로 새로운 영상을 만드는 것
  - 영상 획득 시, 변질된 영상을 복원시키는 것

영상 처리 기술의 사례



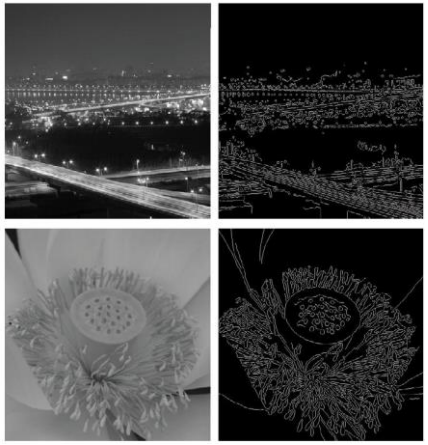
영상 개선 (평활화)

영상 복원

영상 변환 (이산 코사인 변환)

영상 인식 (지문 인식)

영상 압축 (JPEG 압축, 1:12/6/2)



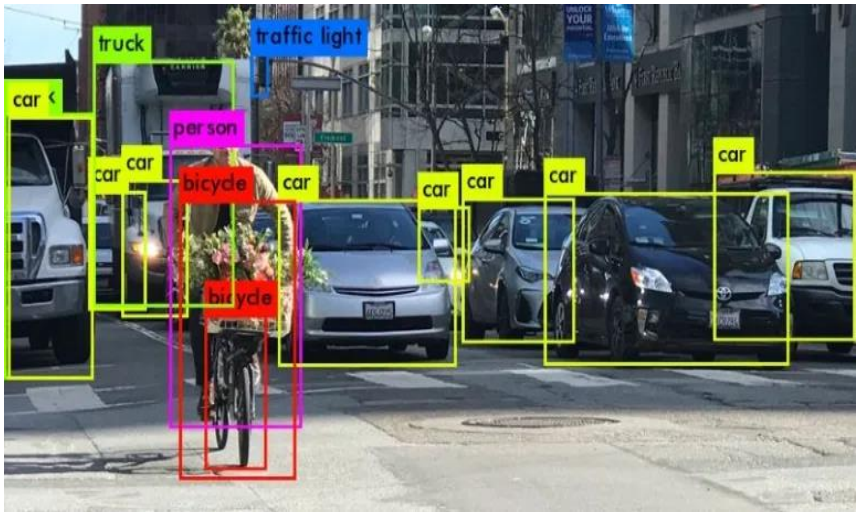
영상 분석 (윤곽선 검출)

컴퓨터 비전(Computer Vision)

- 컴퓨터 비전(Computer Vision)은 영상처리를 포함하는 포괄적인 개념으로, 영상에서 의미 있는 정보를 추출하는 기술
- 영상처리 기술을 통해 원본 영상을 원하는 새로운 영상으로 수정 및 조작한 뒤, 컴퓨터 비전 기술로 원하는 정보를 얻어내는 과정

객체 인식 (Object Recognition)	영상 속 물체가 무엇인지 인식하는 기술 및 여러 컴퓨터 비전 기술의 포괄적인 개념
객체 검출 (Object Detection)	영상 속에서 찾고자 하는 물체가 어디에 있는지 검출하는 기술
객체 추적 (Object Tracking)	영상 속 물체가 어디로 움직이는지 추적하고 해당 물체에 대한 검출과 인식을 유지하는 기술

컴퓨터 비젼 기술의 사례



객체 검출(Object Detection)

객체 추적(Object Tracking)

OpenCV 소개

- Open Source Computer Vision Library
- 영상처리(Image Processing)와 컴퓨터 비전(Computer Vision) 분야의 가장 대표적인 라이브러리(Library)

  - 사진 혹은 영상을 처리할 수 있는 기능
    - 영상 파일의 읽기 및 쓰기
    - 비디오 캡처 및 저장
    - 영상처리(Image Processing) 알고리즘 및 기술
    - 영상이나 비디오에서 얼굴, 눈, 자동차 등과 같은 특정 물체 분류(Classification) 등

  - 기본적인 기계학습(Machine Learning), 딥러닝(Deep Learning)을 응용할 수 있게 “Tensorflow”, “Torch/PyTorch” 등과 같은 프레임워크(framework)도 지원

- openCV
  - image, 동영상을 다루는 라이브러리
  - image 형식 변환, 크기 및 색상 변환, 필터링, 얼굴 인식, 물체 인식 등
  - image 읽을 때
    - imread() 함수 사용
      - 이미지를 읽어서 숫자 배열 ndarray 형태로 변경

- 이미지 인식
  - 최근 딥 러닝 기술의 발전으로 이미지 인식 기술이 크게 발전
  - 여기서는 딥 러닝이 아닌 openCV가 제공하는 패턴인식 기술 이용

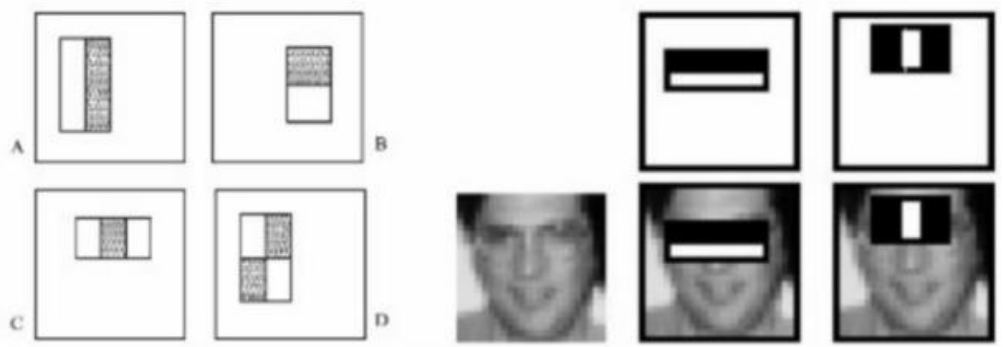
OpenCV 주요 함수

- https://docs.opencv.org/4.5.0/index.html

함수	기능	함수	기능
cv2.imread()	이미지 읽기	cv2.imwrite()	선, 사각형, 원, 타원 도형 그리기
cv2.cvtColor()	이미지 컬러공간 변환	cv2.line()	
cv2.resize()	이미지 사이즈 변경	cv2.rectangle()	
cv2.imshow()	이미지 출력	cv2.circle()	
cv2.putText()	텍스트 삽입	cv2.ellipse()	
cv2.split()	이미지 채널 분리	cv2.normalize()	값의 범위 변경
cv2.merge()	이미지 채널 병합		

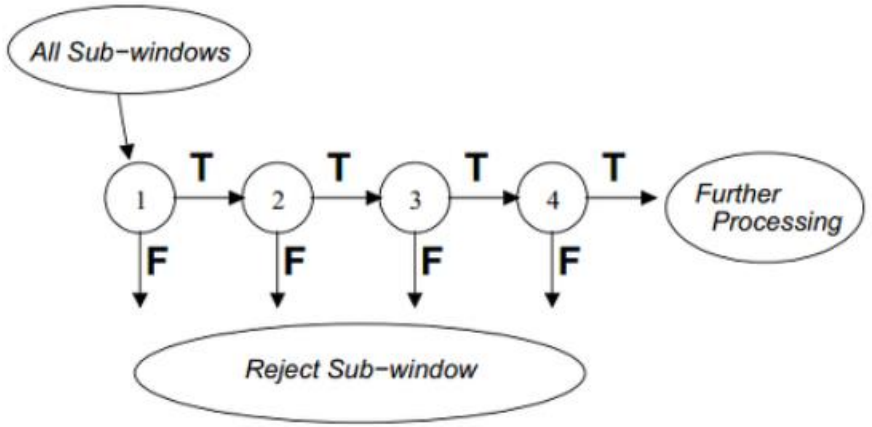
얼굴 인식 (openCV)

- **Haar-like 특징 학습을 이용한 얼굴 인식**
  - 사람 얼굴에는 공통된 특징이 있다
    - 예) 모든 사람의 눈, 코, 입 부분의 명암이 매우 유사한 패턴
  - 3종류의 특징 사용
    - A, B(Two-rectangle feature)
    - C(Three-rectangle feature)
    - D(four-rectangle feature)
  - 입력영상에서 A,B,C,D와 같은 Sub-Window를 Sliding시켜 탐색
  - 영상에서 검정 부분과 흰색 부분의 밝기 값을 빼서 임계값 이상인 것을 찾는다.
  - 이때 Sub-Window내의 feature의 크기와 모양은 다양할 수 있다.
    - 단, sliding window(검정부분과 흰색부분의 크기와 모양)은 동일



얼굴 인식 (openCV)

- **Cascade 방식** (참고: http://opencv-master.tistory.com/90)
  - 사람 얼굴 고유의 특징을 database화하여 이미지에서 추출하는 방법 사용
  - 연산이 복잡한 강 분류기를 모든 영역에 적용시키는 것은 비효율적
    - Object는 입력 영상의 일부분일 뿐 모든 부분에 퍼져 있지 않다는 가정
    - 예, 단체 사진에서 사람의 얼굴이 사진에서 차지하는 비율은 극히 일부분
  - 전체 특징을 여러 분류기 단계로 그룹화, 단계별로 하나씩 적용
    - 처음 몇 단계에는 매우 적은 수의 특징을 포함
    - 해당 단계에서 실패하면 나머지 특징에 대한 검출은 포기하고 바로 다음 창으로 이동
    - 통과하면 기능의 두 번째 단계를 적용하고 해당 특징들을 검출



얼굴 인식 (openCV)

- Haar-like 특징 학습을 이용한 얼굴 인식
  - 단점
    - 영상의 밝기 값을 이용하기 때문에 조명, 대조비에 많이 영향
    - sample의 양이나 질에 따라 성능에 크게 영향

텍스트 분석

텍스트 분석의 개요

- 텍스트 분석의 목적
  - 텍스트의 의미를 알아내는 것
    - 글의 목적
    - 글쓴이의 성향(찬성/반대)
    - 기분(기쁨/슬픔/우울함 등)
    - 제품 피드백 등
- 텍스트 자체는 대표적인 비정형 데이터
- 의미를 추출하려면 비정형 데이터에서 정형화된 정보를 먼저 얻어야 함
  - 텍스트 구문을 분석하여 의미를 파악하고 이것을 정량적으로 측정함

텍스트 분석

- SNS(트위터, 블로그, 페이스 북 등) 글을 분석
  - 소비자들의 반응, 감성, 트렌드를 파악
  - 개인별 마케팅, 상품 피드백을 분석하는데 사용
- 이메일, 웹사이트 댓글, 신문기사, 콜센터 상담기록, 도서 등을 분석
  - 글의 주요 내용을 파악
  - 문서의 특징을 추출
  - 유사한 글이나 저자를 찾는 작업 등을 수행
- 참고문헌이나 본문 인용의 관계를 통해서 문서간의 연계성, 전문가들의 인적 네트워크 등을 파악하는데도 사용
- 인공지능 스피커, 챗봇 등에서도 기본적으로 텍스트 분석이 필요

텍스트 분석 응용

- 챗봇 (Chatbot)
  - 사람과 대화하듯이 음성, 키보드 입력으로 대화를 나누는 인공지능 서비스
  - 챗봇의 유형
    - 미리 답을 준비하여 관련 질문이 나오면 해당 답을 하는 간단한 방식 (저수준)
    - 신경망을 사용하여 최적의 답을 찾아주는 방식 (고수준)

- QA 시스템
  - 질문을 하면 검색을 통해 적절한 답을 찾아주는(대답) 서비스
    - 대한민국의 수도는?
    - 오늘 날씨는?

- 자연어 처리
  - 언어 모델을 사용
    - 가장 자연스러운 다음 문장을 완성
    - 문장을 번역
    - 문서 요약, 주제 분석, 감성 분석 등을 수행

텍스트 표현 방법

- 사람이 단어나 문장의 의미를 인식하듯이 컴퓨터가 단어 자체 의미를 직접 파악할 수는 없다

- 텍스트 데이터 처리
  - 대표적인 비정형 데이터
  - 먼저 비정형 데이터인 글자로부터 정형화된 데이터인 수치 데이터로 변환

- 토큰화(tokenize) : 텍스트 분석의 첫 단계
  - 주어진 텍스트를 토큰으로 나누는 작업
  - 컴퓨터가 다루는 텍스트의 단위 : 토큰
    - 단어 (word) or 글자(character)

코퍼스 (말 뭉치)

- **말뭉치(corpus)**
  - **데이터 분석에 주어진 전체 문서 집합**
- **문서(document)**
  - **코퍼스 내의 한 단위의 텍스트**
  - **예) 하나의 블로그는 문서이고, 분석할 대상 블로그가 1천개이면 이 1천개 블로그 집합이 말뭉치**
- **파싱(parsing)**
  - **코퍼스에서 의미 있는 단어를 추출하는 작업**
- Stop words(**불용어**)
  - **언어**(language)**에서 너무 흔하게 자주 나타나는 단어들**(“the”, “is”, etc)

토큰화

- 토큰화 단위 (크게 3가지)
  - 단어(word)
    - 사람이 말을 이해할 때, 단어 단위로 인식하기 때문에 많은 연구에서 선호
  - 글자(character)
  - n-gram
- 단어 단위로 정보를 표현하는 과정에서 많은 정보를 잃게 된다
  - “정말”, “정말로”, “정말은” 등 단어
    - 같은 단어로 취급, 아니면 각각 다른 단어로 처리할지에 따라 분석 결과 달라짐
    - 같은 단어로 취급하기 위해 단어를 어근(stem)으로 변환하면 어미 변화를 무시하거나 조사를 무시하게 되어 텍스트에 들어 있던 정보를 잃게 된다
- 일반적으로 단어의 종류는 보통 10만 단어 이상 (언어마다 상이)
  - 신조어, 특수한 단어 포함하면 수십만개로 확대

토큰화

- 글자 단위로 토큰화를 하면 어근으로 변환할 때 정보를 잃는 문제를 피할 수 있다.
  - “정”, “말 “, “로”, “은 “ 등
- 음절 단위 토큰의 수
  - 연속적인 말소리를 '하나의 종합된 음'이라는 느낌이 나는 분절단위로 나눌 때, 그 단위
  - 영어
    - 음절단위의 토큰의 수가 매우 적다 : 알파벳이 26글자
  - 한글
    - 음절의 수가 수천 가지 이상

토큰화 – n-gram

- n-gram
  - n개의 연속된 단어를 하나로 취급하는 방법

- 예를 들어 “러시아 월드컵” 이라는 표현을 “러시아” 와 “월드컵” 두 개의 독립된 단어로만 취급하지 않고 두 단어로 구성된 하나의 토큰으로 취급
  - n=2 경우, bi-gram
  - 단어의 수가 매우 크게 증가
    - 실제로는 빈도 수가 최소한 몇 개 이상인 것만 다룬다

토큰화 – n-gram (예)

텍스트: “어제 러시아에 갔다가 러시아 월드컵을 관람했다”

단어토큰: { “어제”, “러시아”, “갔다”, “러시아”, “월드컵”, “관람” }

2-gram 토큰: { “어제 러시아”, “러시아 갔다”, “갔다 러시아”, “러시아 월드컵”, “월드컵 관람” }

토큰화

- n-gram을 허용하면 토큰화 대상의 수가 매우 크게 증가
  - 이론적으로는 10만개의 단어를 두 개 붙여서 나올 수 있는 경우의 수는 10만의 자승이 된다.
- 실제로는 빈도수가 최소한 몇 개 이상인 것만을 다룬다.
- 토큰화한 결과를 수치로 만드는 방법
  - 원핫(one-hot) 인코딩
  - BOW(단어모음)
  - 단어벡터(Word Vector) 방법

원 핫 (One-hot) 인코딩

- **원 핫 인코딩**
  - **토큰**에 고유 번호를 배정
  - 모든 고유번호 위치의 한 컬럼만 1, 나머지 컬럼은 0인 **벡터**로 표시

텍스트: "어제 러시아에 갔다가 러시아 월드컵을 관람했다"

토큰 사전: { "어제" :0, "러시아" :1, "갔다" :2, "월드컵" :3, "관람" :4}

원핫 인코딩:

어제 = [1, 0, 0, 0, 0]
러시아 = [0, 1, 0, 0, 0]
갔다 = [0, 0, 1, 0, 0]
월드컵 = [0, 0, 0, 1, 0]
관람 = [0, 0, 0, 0, 1]

원 핫 (One-hot) 인코딩

- 원핫 인코딩 방식으로 단어(토큰)을 표현하면
  - 단어의 수가 적을 때에는 문제가 안되지만
  - 단어가 모두 10만개이면
    - 모든 단어가 항목이 10만개인 (0과 1로 구성된) 벡터로 표시
  - 주어진 텍스트가 20개의 단어로 구성되어 있다면
    - 20 x 100,000개 크기의 벡터가 필요

BOW (Bag of Word, 단어 모음)

- 텍스트 분석은 “문장” 을 단위로 하는 경우가 많다
- 단어 모음(BOW) 방식 : 한 문장을 하나의 벡터로 만드는 방법
  - 한 문장을 단어 사전 크기의 벡터로 표현하고 그 문장에 들어 있는 단어의 컬럼만 1로, 단어가 없는 컬럼은 모두 0으로 표현
- 먼저 단어 사전을 만들고 각 문장에 어떤 단어가 들어 있는지 조사하여 해당 컬럼만 1로, 나머지는 0으로 코딩

BOW

- 단어 사전: { “어제” :0, “오늘” :1, “미국” :2, “러시아” :3,
  “갔다” :4, “축구” :5, “월드컵” :6, “올림픽” :7,
  “관람” :8, “나는” :9, …, “중국” :4999 }
- Text_1: “어제 러시아에 갔다가 러시아 월드컵을 관람했다” 를
  BOW로 표현하면

문장번호	0	1	2	3	4	5	6	7	8	9	10	…	4998	4999
Text_1	1	0	0	1	1	0	1	0	1	0	0	0	0	0
Text_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Text_3	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Text_4	0	0	0	0	0	0	0	1	0	0	0	0	1	0
…														
Text_50	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BOW

- **문서-단어(document-term) 행렬**
  - 문장 단위로 어떤 단어들이 있는지를 나타내는 BOW의 확장
  - 문서(document) 단위로 어떤 단어들이 있는지를 표현
  - 같은 단어가 여러번 등장하면 1 이상의 값을 갖는다
  - CountVectorizer() 함수

문서번호	0	1	2	3	4	5	6	7	8	9	10	...	4998	4999
Doc_1	1	2	3	1	4	0	2	0	1	3	0	0	0	0
Doc_2	0	0	0	0	2	0	0	0	0	0	0	0	2	0
Doc_3	0	0	0	1	0	0	0	0	3	0	0	0	0	1
Doc_4	4	0	0	0	0	0	0	1	0	0	4	0	1	0
...														
Doc_100	0	2	0	0	0	0	1	4	0	1	0	0	0	0

td-idf

- term frequency-inverse document frequency
- tf : 어떤 단어가 각 문서에서 발생한 빈도
- df(document frequency) : 그 단어가 등장한 '문서' 의 빈도
- 적은 문서에서 발견될수록 가치 있는 정보
- 많은 문서에 등장하는 단어일수록
  - 일반적인 단어
  - 이러한 공통적인 단어는 tf가 크다고 하여도 비중을 낮추어야 분석이 제대로 이루어질 수 있다.
- 따라서 단어가 특정 문서에만 나타나는 희소성을 반영하기 위해서 idf(df의 역수)를 tf에 곱한 값을 tf 대신 사용

td-idf

$$\mathrm{TF-IDF}(w,d) = \frac{\mathrm{TF}(w,d)}{\mathrm{DF}(w)}$$

$$\mathrm{IDF}(w,d) = \log\left(\frac{n}{1+\mathrm{DF}(w)}\right)$$

- IDF는 DF의 역수
- 위 식에서 log와 1을 분모에서 더한 이유
  - n은 문서의 수를 의미
  - n이 커질수록 IDF의 값이 기하급수적으로 커지기 때문에 그것을 방지하기 위해 log를 사용
- TfIdfVectorizer() 함수

단어 임베딩

단어 임베딩의 정의

- 앞에서 소개한 세 가지 텍스트 코딩 방식인 원핫 인코딩, BOW(단어모음), 문서-단어 행렬방식은 단어마다 고유번호를 배정하여 사용
- 그러나 이 고유 번호 숫자에는 아무런 의미가 들어 있지 못하며 단지 인덱스의 성격만 갖는다.
- 단어를 인덱싱이 아니라, 의미 있는 숫자 들의 집합, 즉, 벡터로 표현하는 방법이 단어 임베딩 (Word Embeding)이다.

단어 벡터

- 단어 벡터
  - 각 단어를 50~300개 정도의 차원으로 구성된 벡터로 표현

    학교 = [0.23, 0.58, 0.97, … , 0.87, 0.95]

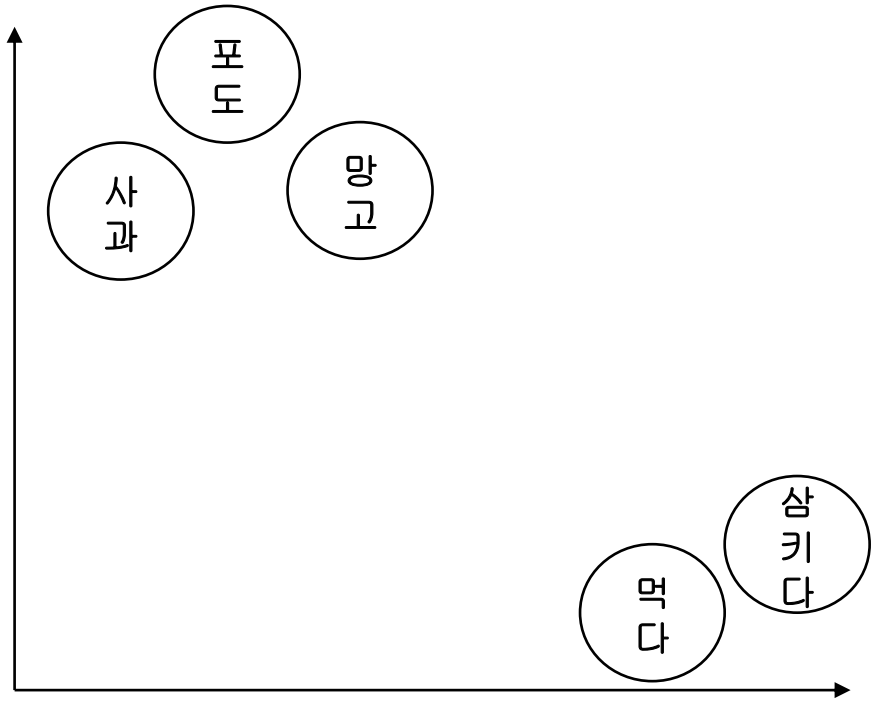
    바다 = [0.45, 0.37, 0.81, … , 0.22, 0.64]

- 단어 벡터를 사용하면
  - 각 단어들 사이의 "거리" 를 계산이 가능
  - 거리를 기반으로 유의어/반대어 등을 찾아낼 수 있다
  - 동물의 성별, 단수/복수, 동사/명사를 구분할 수도 있다
  - 그러나 각 벡터 값의 의미는 알 수 없다

단어 벡터

- 단어 벡터는 대형 말뭉치로부터 학습
  - 말뭉치의 문장들을 계속 입력하여 학습을 시키면 단어 벡터를 얻을 수 있다
  - 예를 들어 음식과 관련된 다음과 같은 문장들로 학습을 시키면 다음과 같은 단어 벡터를 얻을 수 있을 것이다.
  - 학습에 사용된 문장 예:

"나는 어제 바나나를 맛있게 먹었다"
"이 망고는 먹기가 힘들다"
"이 사과는 씹는 맛이 아주 좋다"
"바나나가 사과보다 맛있다"
"잘 씹어야 맛있게 먹을 수 있다"
…



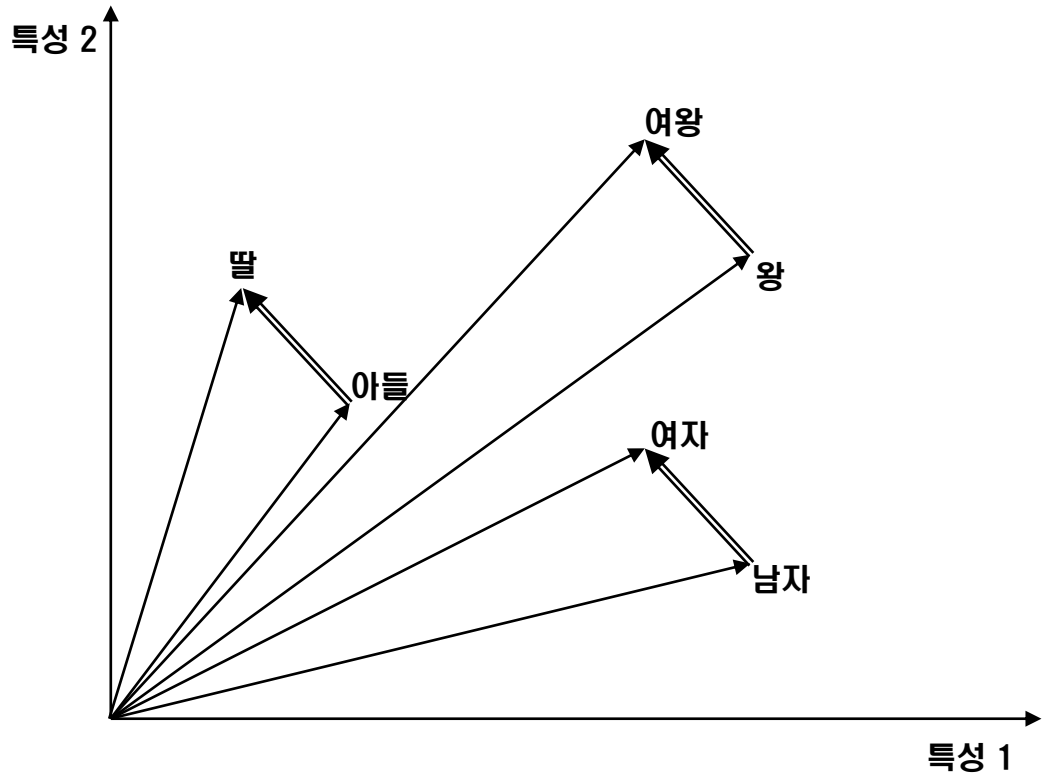
단어 벡터

- **이미 만들어져 있는 단어 벡터를 가져다 사용할 수도 있다.**
- **glove**
  - **2014년 스탠포드에서 만든** Global Vectors for Word Representations
  - **위키피디아 데이터로부터 학습**
  - **40만개 단어를 100차원으로 임베딩**
  - npl.stanford.edu/projects/glove **에서 다운로드**

단어 벡터

- 단어 벡터를 사용한 A:B = C: ?의 관계를 만족하는 ?를 찾을 수 있다.
  - 왕 : 여왕 = 아들 : ? → ? 부분 : 딸
  - 이러한 연산은 (B-A) 벡터, 즉 (왕 – 여왕) 성분을 구한 후 이를 벡터 C(아들)에 더하면 딸을 구할 수 있다.
  - 이들의 관계는 아래와 같다.

http://word2vec.kr



네이버 영화 평점 분석

- **네이버 영화 평점 데이터를 이용한 감성분석 방법을 소개**
  - **문서-단어 행렬을 이용**
  - **데이터**
    - Naver sentiment movie corpus v1.0 (github.com/e9t/nsmc/) **사용**
    - **영화 리뷰 20만 건이 저장**
    - **각 평가 데이터는 0과 1로 레이블링 (0 : 부정, 1 : 긍정 리뷰)**
  - **한글 자연처 처리**
    - konlpy **패키지에서 제공하는** Twitter **문서 분석 라이브러리를 사용**

단어 벡터 생성

- **단어 벡터 만드는 과정을 소개**
  - **가장 널리 사용되는 라이브러리 : Gensim**
    - pip install gensim

```
from gensim.models.word2vec import Word2Vec
model = Word2Vec(sentense_list, min_count=1)
model.most_similar(positive="조선")

##
[('일본', 0.9953970909118652),
 ('관련', 0.9941188097000122),
 ('인물', 0.9938454031944275),
 ('러시아', 0.9931197166442871),
 ('주요', 0.9918481111526489),
 ('대원군', 0.9915156960487366),
...
```

문장 유사도 측정

- 단어의 유사도
  - 두 개의 문자열이 얼마나 다른지를 나타내는 편집 거리를 이용
- 편집 거리
  - 한 단어에서 다른 단어로 바꿀 때 필요한 최소한의 편집 행동의 횟수
  - 편집 행동
    - 글자를 추가, 제거, 변경
- 두 문장의 편집 거리 계산
  - NLTK 라이브러리를 활용

형태소 분석

- 단어 구분
  - 영어
    - 단어들이 대부분 스페이스로 구분, 단어 구분이 어렵지 않다
    - 예) I am a boy
  - 한글
    - 스페이스로 나눠진 단어가 조사를 포함하거나 복합명사인 경우 등이 있어 품사를 구분하는 작업이 영어처럼 간단하지 않다
    - 예) 나는 소년이다
      - 단어 구분 : '나는', '소년이다'
      - 추가적인 형태소 분석 : '나 ', '는', '소년 ', '이다 '
- 형태소 분석(morphological analysis)
  - 한글 문장을 처리하려면 단어를 다시 더 작은 단위인 형태소로 나누는 절차가 필요

형태소 분석기

- 형태소 분석기(예)
  - Hannanum (한나눔) : KAIST
  - Kkma (꼬꼬마) : 서울대
  - Komoran (코모란) : Shineware
  - Mecab (메카브)
    - 일본어용 형태소 분석기를 한국어를 사용할 수 있도록 수정
  - Okt (Open-korean-text) : twitter 개발
    - Twitter

토픽 모델링

- 토픽 모델링
  - 문서의 주제(카테고리)를 구분하는 것
  - 미리 카테고리가 정해져 있지 않으므로 비지도 학습에 해당
- 관련된 단어나 문서의 집합을 찾는 방법이 필요
  - 잠재 디리클레 할당, LDA(Latent Dirichlet Allocation) 방법을 주로 사용
  - 관련성이 높은 단어들이 발생 → 같은 토픽으로 분류
- 한 문서에는 여러 토픽이 복합적으로 존재할 수 있다
  - 각 토픽의 비중은 다를 수 있다

LDA (Latent Dirichlet Allocation)

- DLA
  - 문서의 각 토픽들이 디리클레 분포를 따른다고 가정
  - 각 문서를 각 토픽에 “할당” 하는 방식으로 동작
    - 문서마다 토픽이 어떻게 분포되어 있는지, 그리고 토픽마다 단어의 분포가 어떤지 파악
    - 토픽에 따라 단어의 분포를 결정하고 그중 가장 높은 확률의 단어를 선택

LDA

- **LDA**
  - **말뭉치로부터 대표적인 토픽을 먼저 선정**
  - **해당 토픽으로부터 단어들을 뽑아서 문서를 생성**

- LDA**의 학습 과정**
  - **주어진 문서에 등장한 단어들이 어떤 토픽에서 뽑혔고**
  - **그 토픽의 확률이 어떻게 분포하였는지를 추론해 내는 것**

- **말뭉치의 단어가 어떤 토픽에 해당하는지 명시적으로 표시되어 있지 않기 때문에 학습을 통해 ‘잠재적’ 인 정보를 추출해야한다**

뉴스 분석 예

- 뉴스기사를 사용하여 토픽 모델링을 수행하는 예
  - 먼저 여러 문서에서 자주 나타나는 공통 단어를 제거
  - 상위 10,000개의 단어를 선택하여 BOW 모델을 생성
- LDA 분석으로 얻은 결과는 주제를 구별하는데 도움을 주지만 비지도 학습이기 때문에 완벽한 정답은 아니다
  - 주제에 할당된 문서를 확인하여 평가, 검증하는 과정은 사람이 해주어야 한다

수고하셨습니다.

Q & A