AI Newsletter (No. 3)

연구관리처 AI연구센터 / (2023년 3월 21일)

K-water연구원 AI연구센터에서 정기적으로 발간하는 뉴스레터입니다.
AI 뉴스, Hands-on 프로젝트, 팁 등을 다양한 내용과 난이도로 담았습니다.

※ 코드 및 뉴스 등 외부링크가 다수 포함되어 있으므로 **인터넷 환경 PC 권장**

>> Hello, world!

#AI연구센터 #Hello, K-water #직접 돌려보기 #AI 2학년 2반

□ 안녕하세요. K-water연구원 연구관리처 AI연구센터입니다.

□ 최근 AI 분야에서 가장 화제라고 하면 단연 OpenAI의 ChatGPT일 것입니다. ChatGPT는 OpenAI의 생성형 언어모델인 GPT-3.5를 기반으로 개발되었는데, 최근 GPT-4 모델이 공개되었으며, 이 역시 놀라운 성능을 보여주고 있습니다.

□ 이번 호에서는 최근 트렌드에 맞춰 Hands-on AI 파트에서 자연어 전처리 및 간단한 응용분야에 많은 분량을 할애하였습니다. 데이터는 계속 활용해오던 진짜/가짜 뉴스 텍스트 데이터셋을 활용합니다.

□ AI연구센터에서는 AI기술 최신동향 및 실무적용 방법 확산을 위해 2023년 동안 약 15회에 걸쳐 AI 기술세미나를 개최할 예정입니다.

□ 또한, 지난 뉴스레터에서 말씀드 린 'K-water AI 기술 개발 보고서'를 정식으로 발간하여 문헌정보관이나 K-water 전자책을 통해 만나보실 수 있습니다.

모든 실습 예제와 링크는 "AI연구센터" 누리집(Homepage, GitHub)에 게시되어 활용가능합니다.

CONTENTS


```
> print(f"{YOUR_NAME}, Please Have FUN\n :D");
```

1. What's NEW in AI

#PIANO #T5 #GPT-4 #15분도시

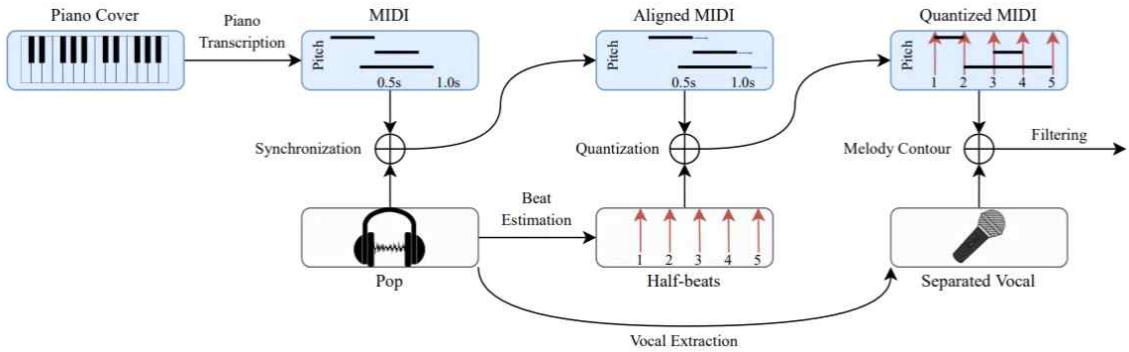
□ (논문) POP2PIANO

○ 일반 음악을 피아노로만 연주하기 위해서는 음악적 기술과 창조적인 능력이 필요합니다. 이를 AI로 대신할 수 있는 논문*이 발표('22.10.)되었습니다.

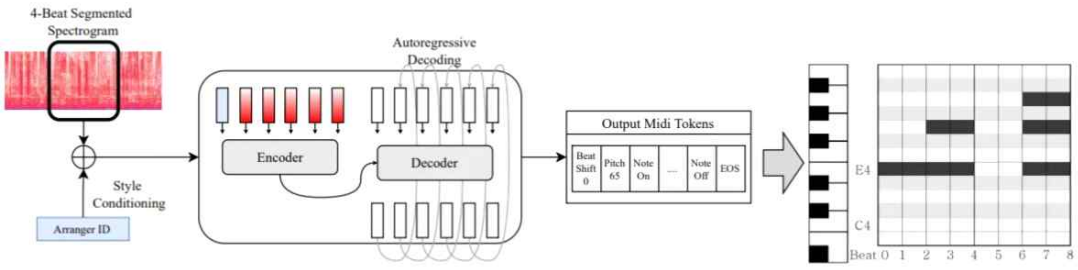
*Choi and Lee (2022), 'POP2PIANO : POP AUDIO-BASED PIANO COVER GENERATION'

○ 논문의 저자는 21명의 피아노커버 유튜브 채널에서 총 5,989개의 악보를 수집하였고, 실제 훈련에는 4,989개(307시간)의 자료를 사용했습니다.

○ ①피아노 악보와 실제 음악을 동기화시키고, ②비트를 추출하여 MIDI 파일을 정렬(quantize)한 다음, ③보컬을 추출하여 멜로디 라인을 구성했습니다.



○ 해당 논문에서는 훈련을 위해 transformer를 기반으로 하는 T5-small 모델을 사용했습니다. 훈련가능한 매개변수는 5,900만개이며 첫 번째 레이어에 더해지는 위치 임베딩(positinal embedding)에 상대적(relative)인 위치가 아닌 절대적(absolute)인 위치로 임베딩 하였습니다.



○ 해당 논문의 결과물은 Github에 쉽게 확인할 수 있도록 게시되어 있으며, Colab을 통해 본인이 원하는 음악을 MIDI파일로 생성하여 다운로드 할 수 있도록 소스코드와 함께 제공하고 있습니다.

※Links: https://arxiv.org/pdf/2211.00895.pdf
https://sweetcocoa.github.io/pop2piano_samples/
https://colab.research.google.com/drive/1rBAs2TkryDnnQOhcM-mtlrgtL2h3ekml?usp=sharing

□ (기술) GPT-4

○ 지난 뉴스레터에서 소개해드린 ChatGPT는 OpenAI의 생성형 AI 언어모델로 GPT-3.5를 기반으로 개발되었습니다. ChatGPT가 큰 화제가 되고있는만큼 GPT-4에 대한 관심 역시 증가하고 있습니다.

○ 3월 15일 OpenAI는 GPT-4를 발표하였으며, 현재 ChatGPT Plus($20/달)를 구독하면 사용해 볼 수 있습니다. MS의 Bing AI모델에 탑재된 '프로메테우스'라고 불리던 AI 모델은 GPT-3.5 기반이라고 알려져 있었으나, 사실 GPT-4의 검색 특화 버전이 적용되고 있었다고 발표했습니다.

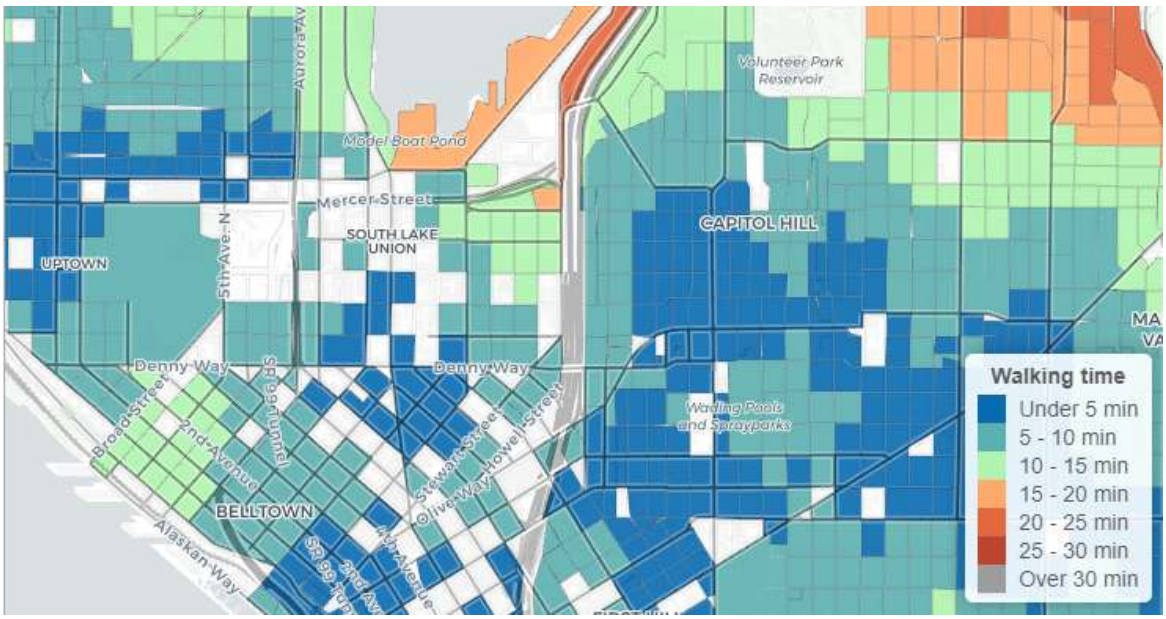
○ GPT-4는 이전 모델에 비해 많은 부분에서 개선되었으며, 이미지 처리도 가능한 멀티모달(multimodal) 모델로 개발되었습니다. 미국 변호사 시험에서 상위 10%, 생물학 올림피아드에서 상위 1%를 달성하는 등 많은 테스트와 AI 벤치마크 데이터셋에서 높은 성능을 달성했습니다.

※Links: https://openai.com/product/gpt-4
https://cdn.openai.com/papers/gpt-4.pdf

□ (시각화) 15분 도시

○ '15분 도시'라는 개념에 대해 들어보셨나요? 이 개념의 핵심은 자동차나 대중교통을 이용하지 않고도 도시에서 일상적인 필요와 서비스를 빠르게 이용할 수 있어야 한다는 것입니다.

○ 시애틀의 지리학자이자 연구원인 Nat Henry는 각 동네에서 학교, 레스토랑, 슈퍼마켓, 공원 등 시애틀의 다양한 편의시설까지 걸리는 시간을 매핑했습니다. 이를 위해 오픈스트리트맵 데이터 기반한 OpenRouteService를 사용했습니다.



※Links: https://nathenry.com/writing/2023-02-07-seattle-walkability.html

2. "10줄 코딩" Hands-on AI Project

#자연어 #NLP #전처리 #단어구름 #검색 #유사단어 #10줄?

□ 자연어처리의 전처리 및 응용 (초중급, 자연어)

※Link: https://colab.research.google.com/drive/1PpnFYMxFfIXIlgHvP7NMa9oSwiW-6HIn?usp=sharing
※가독성을 위해 일부 세부적인 라인은 생략되어 있습니다. 전체 코드는 위 링크를 참고해주세요.

○ 지난 뉴스레터에서 소개해드린 ChatGPT가 최근 큰 화제가 되고 있습니다.

○ 이번 호에서는 사람의 언어(자연어)를 컴퓨터가 처리하기 위해 필요한 전처리 및 간단한 자연어처리 응용에 대해 다뤄보겠습니다.

○ 영어는 대소문자 구분이 있어 일반적으로 소문자로 통일합니다.

```
def lower_case(text):
  return text.lower()
lower_case("HELLO world")
출력: 'hello world'
```

○ 공백, 엔터, 탭등을 제거하고 .join 메소드를 을 활용하여 다시 합쳐줍니다.

```
def remove_spaces_tabs(text):
  return " ".join(text.split())
remove_spaces_tabs("hello \n world \t")
출력: 'hello world'
```

○ 구두점(쉼표, 마침표 등) 제거

- maketrans와 string 라이브러리의 punctuation을 활용합니다.

```
import string
def remove_punct(text):
  translator = str.maketrans("", "", string.punctuation)
  return text.translate(translator)
remove_punct("Hello, World!")
출력: 'Hello world'
```

○ One character 제거

- 영어에서는 하나의 알파벳으로 이루어진 경우 대부분 큰 의미가 없습니다.
- 정규식 라이브러리(regex)를 활용합니다. 정규식 문법은 어렵기 때문에 신경쓰지 마세요.

```
import re # 정규식 라이브러리
def remove_single_char(text):
  return re.sub(r'\b[a-zA-Z]\b', '', text)
remove_single_char("this is a test")
출력: 'this is  test'
```

○ HTML 태그 제거

- 웹 스크래핑을 하다보면 HTML 태그(</body>,
 등)를 제거할 필요가 있습니다.
- 이것도 정규식 라이브러리(regex)를 활용하여 제거할 수 있습니다.

```
def remove_html(text):
  html = re.compile(r'<.*?>')
  return html.sub(r"", text)
remove_html("Hello <b>World</b>")
출력: 'Hello world'
```

○ 인터넷 주소 제거

- SNS, 뉴스 등을 스크래핑하면 자주 들어옵니다. 마찬가지로 정규식을 활용합니다.

```
def remove_url(text):
  url = re.compile(r"https?://\S+|www\.\S+")
  return url.sub(r"", text)
```

```
remove_url("go to https://www.google.com for answers")
```

```
출력: 'go to  for answers'
```

○ 이모지(emoji) 제거

- SNS나 채팅자료 등에 존재합니다. UNICODE 구간을 활용하여 제거할 수 있습니다.

```
def remove_emoji(text):
  emoji_pattern = re.compile(
    "["
    "\U0001F600-\U0001F64F"  # 이모티콘
    "\U0001F300-\U0001F5FF"  # 기호, 픽토그램
    "\U0001F680-\U0001F6FF"  # 교통, 지도 관련
    "\U0001F1E0-\U0001F1FF"  # iOS 플래그
    "]+",
    flags = re.UNICODE
  )
  return emoji_pattern.sub(r"", text)
```

```
remove_emoji("😊 Hello World 👍")
```

```
출력: ' Hello World '
```

○ 이모지를 단어로

- 제거할 수도 있지만, 의미를 갖고 있기 때문에 단어로 변환하는 것도 고려해봅니다.

```
import emoji

def emoji2word(text):
  words = text.split()
  words = [emoji.demojize(word) if word in emoji.EMOJI_DATA else word for word in words]
  return " ".join(words)
```

```
remove_emoji("😊 Hello World 👍")
```

```
출력: ':smiling_face_with_smiling_eyes: Hello World :thumbs_up:'
```

○ 불용어(stopword) 제거

- 불용어란 텍스트를 분석할 때 크게 도움이 되지 않는 단어들을 의미합니다.
- 영어의 'a', 'the', 한글의 '은/는', '그래서', '라고' 등과 같은 단어들입니다.
- nltk(natural language toolkit) 라이브러리를 활용합니다.

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

def remove_stopwords(text):
  STOPWORDS = set(stopwords.words("english"))   # 기존 설정된 영어 불용어 로드
  STOPWORDS.update(["time"])                    # 불용어 사전에 time 추가
  STOPWORDS -= {"no", "not"}                    # 불용어 사전에서 no와 not을 제거

  words = word_tokenize(text)
  words_filtered = [word for word in words if word not in STOPWORDS]

  return " ".join(words_filtered)
```

```
remove_stopwords("John has not done his homework in time")
```

```
출력: 'John not done homework'
```

○ 숫자를 단어로

- 문장 안에 '2'와 'two'가 동시에 나온다면 분석이 용이하지 않습니다.
- num2words 라이브러리를 활용합니다.

```
def convert_digits_to_words(text):
    words = text.split()
    words = [num2words(word) if word.isdigit() else word for word in words]
    return " ".join(words)
convert_digits_to_words("I have 2 dogs")
출력: 'I have two dogs'
```

○ Contractions 확장

- 영어에서는 'I am'을 'I'm'으로 'we have'를 'we've'으로 줄입니다.
- 이를 다시 되돌리기 위해 contractions 라는 라이브러리를 활용합니다.

```
def expand_contractions(text):
    return contractions.fix(text)
expand_contractions("I'm going to the store")
출력: 'I am going to the store'
```

○ 표제어 추출 (Lemmatization)

- 표제어는 기본 사전형 단어를 의미합니다
- 영어로 예를 들면 'am', 'are', 'is' 는 각각 다른 단어이지만 그 뿌리는'be'에 있습니다.
- nltk라이브러리의WordNet모듈이 제공하는 WordNetLemmatizer()를 활용할 수 있습니다.

```
def lemmatize_text_custom(text, lemmatizer):
  w_pos_tags = nltk.pos_tag(text.split())
  lemmatized = [lemmatize(w, wordnet_map.get(pos[0], wordnet.NOUN)) for w, pos in w_pos_tags]
  lemmatized = " ".join()
  return lemmatized_output
lemmatizer = WordNetLemmatizer()
lemmatize_text_custom("Lemmatizing removes the affixes of a sentence", lemmatizer)
출력: 'Lemmatizing remove the affix of a sentence'
```

○ 어간 추출 (Stemming)

- 단어의 접사 등을 제거하고 어간을 추출하는 과정입니다.
- 어간 추출을 한 후의 문장은 표준어처럼 보이지 않을 수 있습니다.

```
def stem_text_custom(text: str, stemmer) -> str:
    word_tokens = word_tokenize(text)
    stemmed_output = " ".join([stemmer.stem(w) for w in word_tokens])
    return stemmed_output
stem_text_custom("Stemming removes the affixes of a sentence", stemmer)
출력: 'stem remov the affix of a sentenc'
```

○ 오타 교정

- spellchecker 패키지를 활용하여 오타를 교정합니다.

```
def correct_spelling(text):
  for word in text.split():
    if word in misspelled_words:
      corrected_text.append(spell.correction(word))
    else:
      corrected_text.append(word)
    return " ".join(corrected_text)
correct_spelling("spellling is a bigg problem")
출력: 'spelling is a big problem'
```

○ 전체 수행

– 앞서 소개된 전처리를 수행하는 파이프라인을 구성하고 뉴스 데이터에 적용합니다.

```
pipeline = [lower_case, expand_contractions, remove_spaces_tabs, remove_url, remove_punct,
            remove_single_char, remove_html, remove_stopwords,] #추가 및 제거 가능

def pre_processing(text, pipeline, lemmatizer=None, stemmer=None):
  tokens = text
  for transform in pipeline:
    tokens = transform(tokens)
  return tokens
```

```
data_true['text_cleaned'] = data_true['text'].progress_apply(pre_processing, pipeline=pipeline)
```

원본 텍스트	전처리 수행 후
(Reuters) - Alabama officials on Thursday certified Democrat Doug Jones the winner of the state's U.S. Senate race, after a state judge denied a challenge by Republican Roy Moore, whose campaign was derailed by accusations of sexual misconduct with teenage girls ...	reuters alabama officials thursday certified democrat doug jones winner state ' yous senate race state judge denied challenge republican roy moore whose campaign derailed accusations sexual misconduct teenage girls ...

– 대부분 의도한대로 작동하고 있으나 "U.S." 가 "yous"로 변환되고 있습니다. 처리해줍니다

```
data_true['text_cleaned'].progress_apply(lambda x: x.replace('yous', 'united states'))
```

○ 텍스트를 분석해봅시다. 이번 Hands-on에서는 아래의 4개를 수행합니다.

①단어의 빈도 추출, ②워드클라우드, ③단어 기반 뉴스 검색, ④유사단어 찾기

① N-gram Frequency

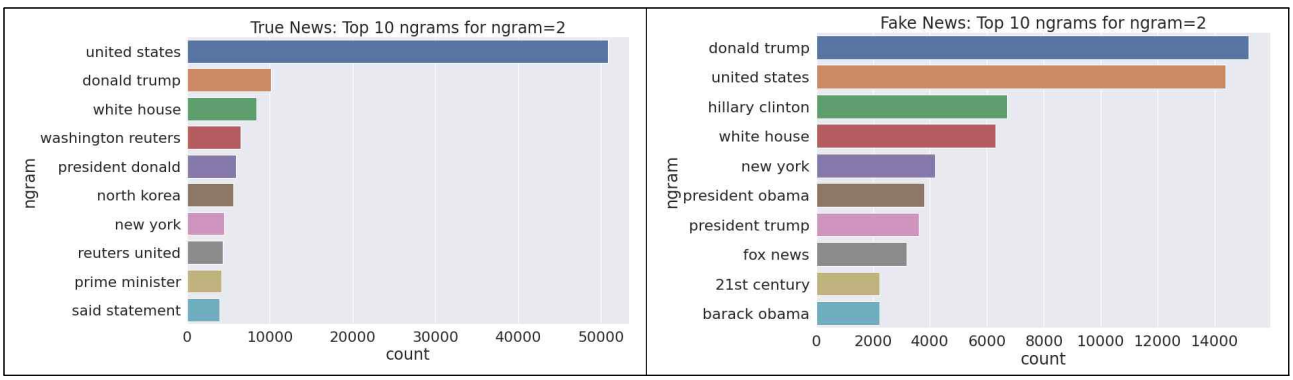
– n-gram은 n개의 연속적인 단어 나열을 의미합니다. (bi-gram(n=2), tri-gram(n=3), ...)

– scikit-learn의 CountVectorizer를 활용하면 쉽게 수행할 수 있습니다.

```
def get_top_ngrams(text, ngram=1, top_n=10):

  # ngram의 범위와, 불용어 등을 지정
  vec = CountVectorizer(ngram_range=(ngram, ngram), stop_words="english").fit(text)
  bag_of_words = vec.transform(text)

  # 각 단어의 빈도를 합산
  sum_words = bag_of_words.sum(axis=0)

  # 단어와 빈도를 추출하고 정렬
  words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
  words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)

  return words_freq[:top_n] # top_n으로 지정한 개수까지 추출 (기본 10개)
```

```
get_top_ngrams(data_true['text_cleaned'], ngram=2) # bigram으로 설정
```

```
출력: [('united states', 50844), ('donald trump', 10163), ('white house', 8394),...]
```

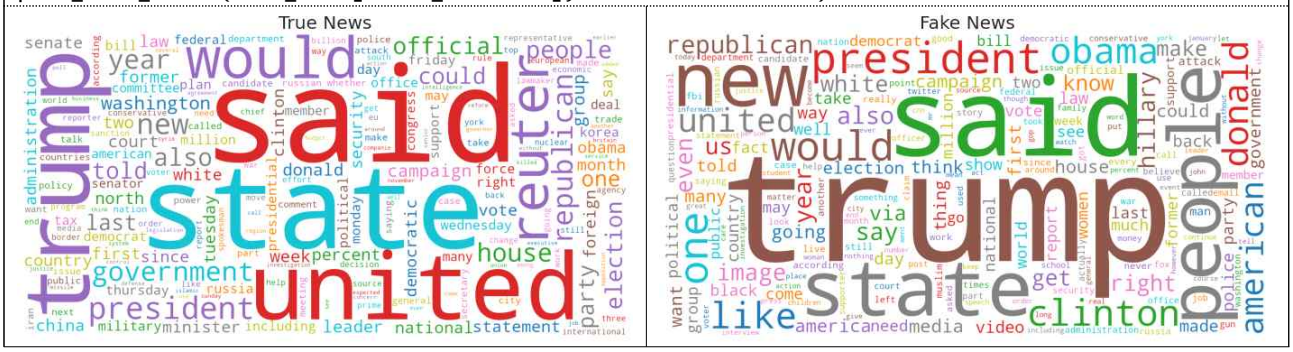


② WordCloud 그리기

- 진부해보일 수 있지만 대량의 텍스트를 하나의 그림으로 표현하기에는 이만한게 없습니다.

```
from wordcloud import WordCloud

STOPWORDS = set(stopwords.words("english"))

def plot_word_cloud(text, title=None):
  wordcloud = WordCloud().generate(" ".join(text))
  plt.imshow(wordcloud)
  plt.show()
```

```
plot_word_cloud(data_true['text_cleaned'], title='True News')
```



③ 텍스트 검색

- textacy의 extract 함수를 활용하여 특정 키워드에 대한 검색결과를 표출합니다.

```
def search_keyword(text, keyword):
    return list(extract.keyword_in_context(text, keyword, window_width=50))

def search_keyword_in_df(text, keyword, n=3):
  searched = text.apply(search_keyword, keyword=keyword)
  searched = searched[searched.apply(len) > 0] # 키워드가 없는 뉴스 제외
  sample_list = searched.sample(n) if len(searched) > n else searched

  for sample in sample_list:
    print(f"{sample[0][0]}{Fore.RED}{sample[0][1]}{Style.RESET_ALL}{sample[0][2]}")
```

```
search_keyword_in_df(data_true['text_cleaned'], "south korea")
```

```
stop beijing also sees united states south korea sharing responsibility rising tensions
seoul reuters south korea president moon jaein said wednesday north korea m
achievements trade matters tour took japan south korea china vietnam last leg philippines
```

④ 유사단어 검색

- 단어를 벡터로 표현하는 Word2Vec 모델을 학습시켜 결과를 살펴봅니다.
- 진짜뉴스 데이터를 기반으로 'oil'과 유사한 단어를 찾아본 결과를 확인해보세요.

```
from gensim.models import Word2Vec
all_words_true = [nltk.word_tokenize(text) for text in tqdm(data_true['text_cleaned'])]
model_true = Word2Vec(all_words_true, min_count=3, size=200, workers=4, window=4, iter=30)
```

```
model_true.wv.most_similar("oil")
```

```
[('shale', 0.527), ('petroleum', 0.519), ('gas', 0.481), ('crude', 0.479), ...]
```

3. TIPs

☐ pip

○ Python을 공부하다보면 '**pip install package_name**'과 같이 라이브러리를 설치하라는 문구를 자주 볼 수 있습니다. 여기서 pip은 뭘까요?

○ pip는 python으로 작성된 라이브러리를 관리하는 시스템입니다. python의 패키지는 https://pypi.org (Python Package Index)에서 볼 수 있습니다.

```
pip install package_name              # 설치
pip install package_name==2.1.3       # 특정버전 설치
pip install package_name --upgrade    # 최신 버전으로 업데이트
pip uninstall package_name            # 패키지 삭제
pip list                              # 설치된 패키지 및 버전 확인
```

☐ QR Code 만들기

○ Python에서도 간단하게 QR Code를 만들 수 있습니다.

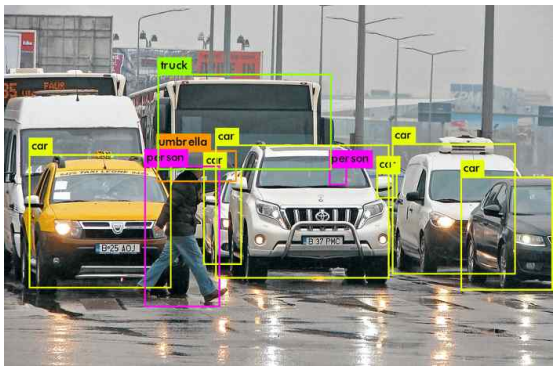
```
# 라이브러리 설치 및 불러오기
!pip install segno # 라이브러리 설치 (한번 설치 후에는 실행할 필요 없음)
import segno
# 아무 인터넷 주소나 넣어줍니다.
url = 'https://www.kwater.or.kr'
# QR Code 생성 및 그림으로 저장
qr = segno.make(url, micro=False)
qr.save("qr_code.png", scale=10)
# 수많은 URL을 QR Code로 만들어야 하는 경우에는 반복문을 이용할 수 있습니다.
for idx, url in enumerate(urls):
  qr = segno.make(url, micro=False)
  qr.save(f"qr_code_{idx}.png", scale=10)
```

☐ 컴퓨터 비전(Computer Vision; CV)의 활용분야

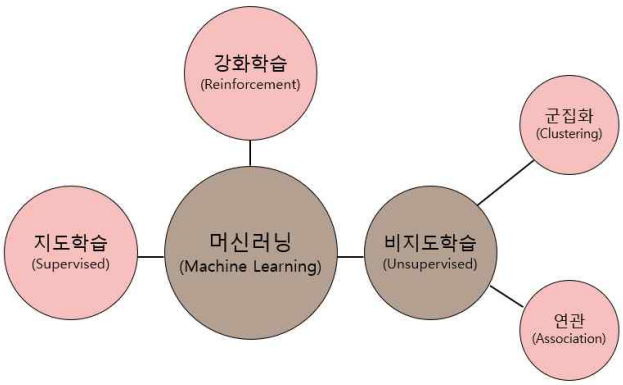
○ 컴퓨터 비전(CV)은 컴퓨터가 이미지를 보고, 이해하고, 분석할 수 있도록 하는 기술로 최근 AI 분야에서 빠르게 확장되고 있는 주제 중 하나입니다.

○ K-water에서도 CCTV, 위성을 활용한 수체 탐지, 관로 및 댐시설물의 결함 검출 등에 사용되고 있습니다. 이외에도 CV는 다양한 분야에 적용됩니다.

- (이미지) ①이미지 분류(의료 이미지에서 종양 분류), ②객체 식별(출입구에서 인물 인식), ③객체 감지 및 위치 지정(CCTV로 위험요소 식별), ④객체 및 인스턴스 분할(픽셀단위 객체 구분), ⑤자세 추정(사람 또는 물건의 자세 인식)
- (비디오) ①객체 추적(Amazon Go Store, CCTV, 자율주행), ②동작 인식(수화 번역), ③동작 추정(교통관리시스템, 3D 정보 오버레이)
- 이미지에 대해 사전학습(pre-trained)된 정보를 활용하여 품질 개선 및 인페인팅 수정
- 물체를 찍은 사진(또는 라이다 스캔)을 통해 3D 디지털 모델로 재구성(NeRF 등)



□ 비지도학습 (Unsupervised Learning)

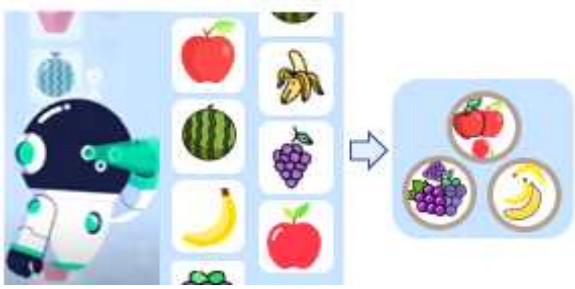


○ 비지도학습은 머신러닝의 한 분야로, **레이블(정답)이 없는** 데이터에서 패턴이나 구조를 찾아냅니다.

○ 비지도학습의 종류로는 비슷한 것끼리 묶어주는 군집화(Clustering), 음악추천과 같이 서로 관련된 특징을 찾아내는 연관(Association)분석 등이 있습니다.

○ 지도학습 vs 비지도학습

지난 호(2호)에서 살펴본 지도학습과의 차이점은 레이블(정답)의 유무입니다. 아래 그림처럼 지도학습은 정답이 있지만, 비지도학습의 경우 정답이 없는 많은 양의 데이터를 보여주면 그 안에서 관계나 패턴을 스스로 학습합니다.

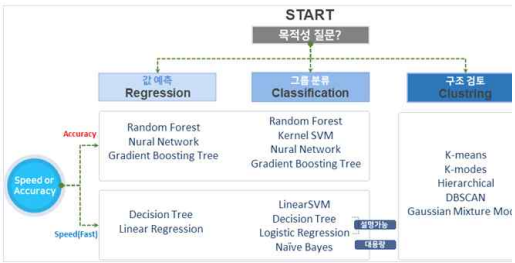
	
지도학습 레이블(정답) 있음	**비지도학습** 레이블(정답) 없음. 패턴 or 구조 찾기

4. Meanwhile, in K-water AI Lab.

☐ K-water AI 개발 기술 보고서 발간

○ AI연구센터에서 디지털 전환 가속화와 AI 기반 미래기술 R&D 선도를 위해 'K-water AI 개발 기술 보고서'를 발간했습니다.

○ 해당 보고서에는 AI 일반사항부터 실무 적용에 초점을 맞춘 상세한 가이드라인을 담았으며 문헌정보관에서 대여하거나, OASIS 문헌정보관 원문보기, 또는 회사 전자책 앱을 통해 e-Book으로도 확인하실 수 있습니다.



구분	이미지 분류	객체 탐지	이미지 분할
영상	 고양이	 강아지, 고양이	 강아지1, 고양이, 강아지2
다중인식	싱글	멀티	멀티
객체인식	X	O	O
객체구분	X	X	O

☐ K-water AI 기술세미나 연간계획 수립

○ AI연구센터에서는 최근 급속도로 발전하고 있는 AI 기술의 최신동향 파악 및 실무적용을 위해 올해 약 15회의 기술세미나 계획을 수립하였습니다.

○ 최근 화제가 되고 있는 자연어처리(NLP), NeRF등의 최신 기술과 사내 주요 사업과 직접적으로 관련된 AI 기술 및 실제 적용사례, AI를 활용하기 위한 Python 고급기술 등을 다룰 예정이니 많은 관심을 부탁드립니다.

☐ AAiCON 2023 참석

○ AI연구센터에서는 지난 2월에 개최된 제2차 실용 인공지능 학술대회(에이아이프렌즈학회 주관)에 참석하였습니다.

○ 세부 프로그램으로는 역량강화 부트캠프(NVIDIA Modulus, Deep Learning), 논문발표, 생성 AI(Generative AI) 워크숍 등이 있었습니다.