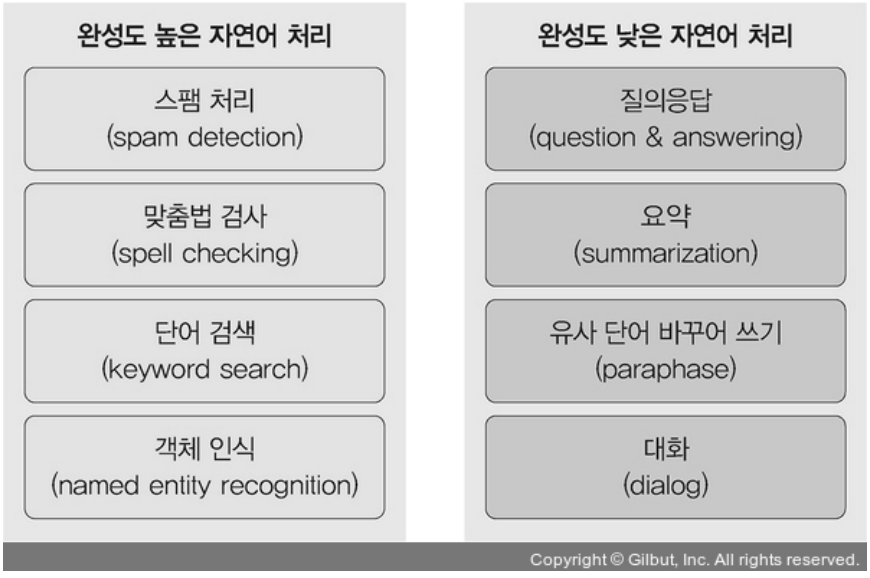
9장. 자연어 처리

9.1 자연어 처리란

자연어 처리란 우리가 일상생활에서 사용하는 언어 의미를 분석하여 컴퓨터가 처리할 수 있도록 하는 과정입니다. 자연어 처리는 딥러닝에 대한 이해도 필요하지만, 그에 앞서 인간 언어에 대한 이해도 필요하기 때문에 접근하기 어려운 분야입니다. 또한, 언어 종류가 다르고 그 형태가 다양하기 때문에 처리가 매우 어렵습니다.

다음 그림은 자연어 처리가 가능한 영역과 발전이 필요한 분야입니다. 예를 들어 스팸 처리 및 맞춤법 검사는 완성도가 높은 반면, 질의응답 및 대화는 아직 발전이 더 필요한 분야입니다.



9.1.1 자연어 처리 용어 및 과정

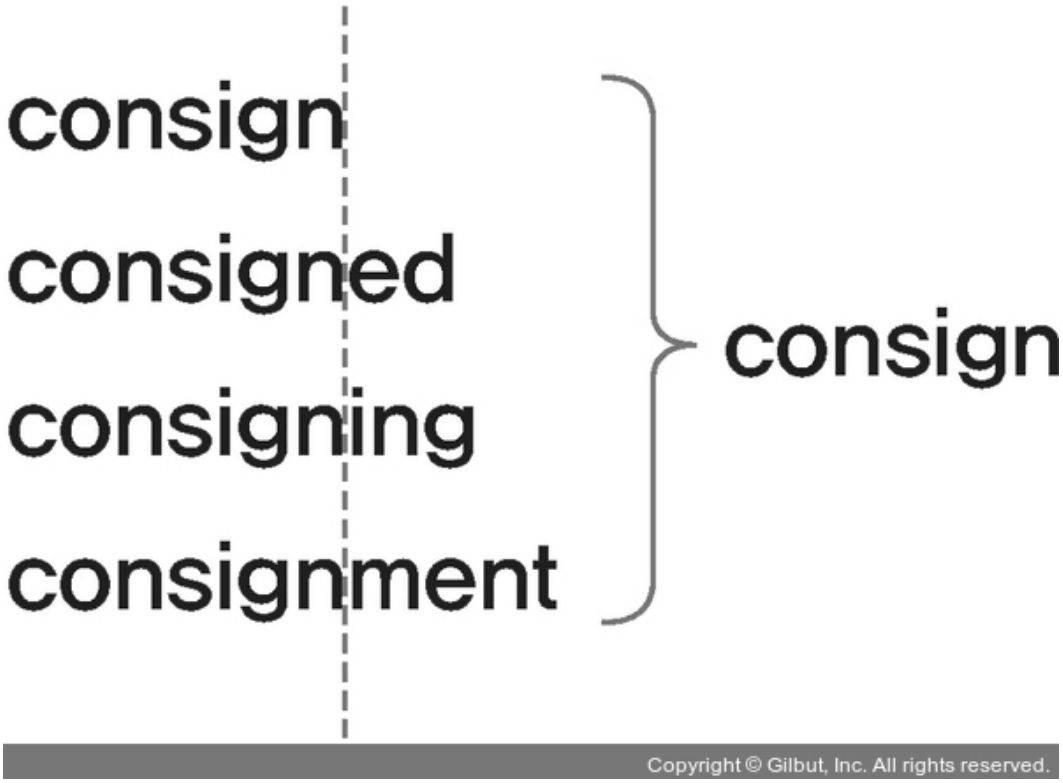
자연어 처리 관련 용어와 처리 과정을 먼저 알아보겠습니다.

자연어 처리 관련 용어

- **말뭉치(corpus(코퍼스))**: 자연어 처리에서 모델을 학습시키기 위한 데이터이며, 자연어 연구를 위해 특정한 목적에서 표본을 추출한 집합입니다.

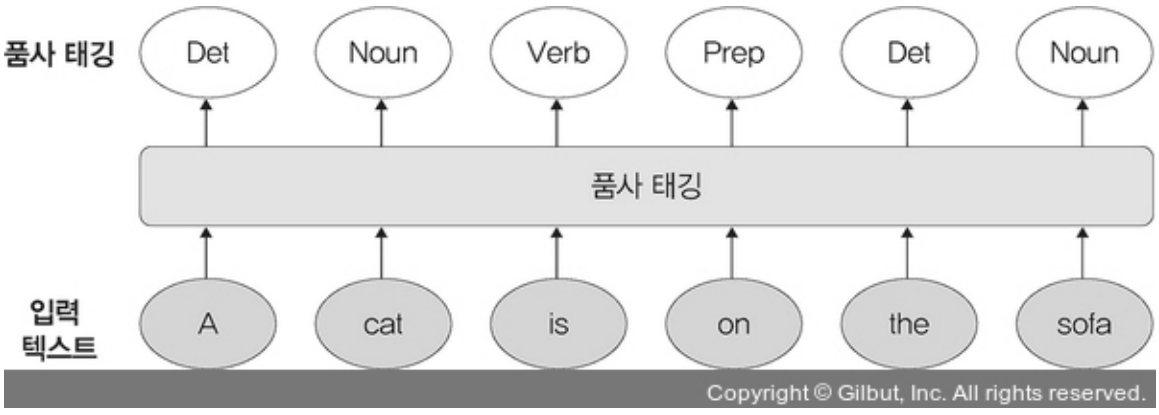
▲ **그림 9-2** 말뭉치(corpus)

- **토큰**(token): 자연어 처리를 위한 문서는 작은 단위로 나누어야 하는데, 이때 문서를 나누는 단위가 토큰입니다. 문자열을 토큰으로 나누는 작업을 토큰 생성(tokenizing)이라고 하며, 문자열을 토큰으로 분리하는 함수를 토큰 생성 함수라고 합니다.
- **토큰화**(tokenization): 텍스트를 문장이나 단어로 분리하는 것을 의미합니다. 토큰화 단계를 마치면 텍스트가 단어 단위로 분리됩니다.
- **불용어**(stop words): 문장 내에서 많이 등장하는 단어입니다. 분석과 관계없으며, 자주 등장하는 빈도 때문에 성능에 영향을 미치므로 사전에 제거해 주어야 합니다. 불용어 예로 “a”, “the”, “she”, “he” 등이 있습니다.
- **어간 추출**(stemming): 단어를 기본 형태로 만드는 작업입니다. 예를 들어 ‘consign’, ‘consigned’, ‘consigning’, ‘consignment’가 있을 때 기본 단어인 ‘consign’으로 통일하는 것이 어간 추출입니다.



▲ **그림 9-3** 어간 추출

- **품사 태깅**(part-of-speech tagging): 주어진 문장에서 품사를 식별하기 위해 붙여 주는 태그(식별 정보)를 의미합니다.



▲ **그림 9-4** 품사 태깅

품사 태깅을 위한 정보는 다음과 같습니다.

- **Det**: 한정사

- **Noun**: 명사
- **Verb**: 동사
- **Prep**: 전치사

자연어 처리 과정

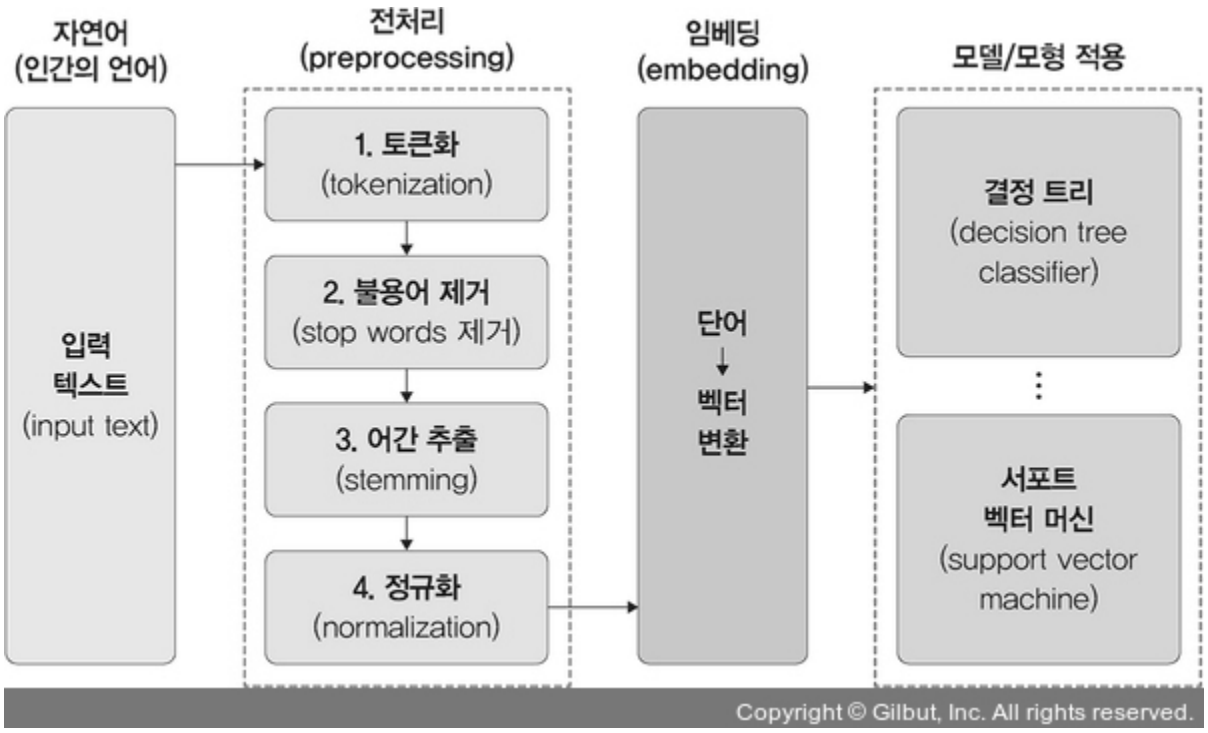
자연어는 인간 언어입니다. 인간 언어는 컴퓨터가 이해할 수 없기 때문에 컴퓨터가 이해할 수 있는 언어로 바꾸고 원하는 결과를 얻기까지 크게 네 단계를 거칩니다.

첫 번째로 인간 언어인 자연어가 입력 텍스트로 들어오게 됩니다. 이때 인간 언어가 다양하듯 처리 방식이 조금씩 다르며, 현재는 영어에 대한 처리 방법들이 잘 알려져 있습니다.

두 번째로는 입력된 텍스트에 대한 전처리 과정이 필요합니다.

세 번째로 전처리가 끝난 단어들을 임베딩합니다. 즉, 단어를 벡터로 변환하는 방법으로 '10장 자연어 처리를 위한 임베딩'에서 자세히 다룹니다.

마지막으로 컴퓨터가 이해할 수 있는 데이터가 완성되었기 때문에 모델/모형(**예** 결정 트리)을 이용하여 데이터에 대한 분류 및 예측을 수행합니다. 이때 데이터 유형에 따라 분류와 예측에 대한 결과가 달라집니다.



▲ **그림 9-6** 자연어 처리 과정

9.1.2 자연어 처리를 위한 라이브러리

NLTK

NLTK(Natural Language ToolKit)는 교육용으로 개발된 자연어 처리 및 문서 분석용 파이썬 라이브러리입니다. 다양한 기능 및 예제를 가지고 있으며 실무 및 연구에서도 많이 사용되고 있습니다.

다음은 NLTK 라이브러리가 제공하는 주요 기능입니다.

- 말뭉치
- 토큰 생성
- 형태소 분석
- 품사 태깅

KoNLPy

KoNLPy(코엔엘파이라고 읽음)는 한국어 처리를 위한 파이썬 라이브러리입니다. KoNLPy는 파이썬에서 사용할 수 있는 오픈 소스 형태소 분석기로, 기존에 공개된 꼬꼬마(Kkma), 코모란(Komoran), 한나눔(Hannanum), 트위터(Twitter), 메카브(Mecab) 분석기를 한 번에 설치하고 동일한 방법으로 사용할 수 있도록 해 줍니다.

참고로 KoNLPy에서 제공하는 주요 기능은 다음과 같습니다.

- 형태소 분석
- 품사 태깅

Gensim

Gensim은 파이썬에서 제공하는 워드투벡터(Word2Vec) 라이브러리입니다. 딥러닝 라이브러리는 아니지만 효율적이고 확장 가능하기 때문에 폭넓게 사용하고 있습니다.

다음은 Gensim에서 제공하는 주요 기능입니다.

- **임베딩**: 워드투벡터
- 토픽 모델링
- LDA(Latent Dirichlet Allocation)

사이킷런

사이킷런(scikit-learn)은 파이썬을 이용하여 문서를 전처리할 수 있는 라이브러리를 제공합니다. 특히 자연어 처리에서 특성 추출 용도로 많이 사용됩니다.

다음은 사이킷런에서 제공하는 주요 기능입니다.

- CountVectorizer: 텍스트에서 단어의 등장 횟수를 기준으로 특성을 추출합니다.
- Tfidfvectorizer: TF-IDF 값을 사용해서 텍스트에서 특성을 추출합니다.
- HashingVectorizer: CountVectorizer와 방법이 동일하지만 텍스트를 처리할 때 해시 함수를 사용하기 때문에 실행 시간이 감소합니다.

9.2 전처리

머신 러닝이나 딥러닝에서 텍스트 자체를 특성으로 사용할 수는 없습니다. 텍스트 데이터에 대한 전처리 작업이 필요한데, 이때 전처리를 위해 토큰화, 불용어 제거 및 어간 추출 등 작업이 필요합니다.

앞서도 살펴보았지만, 전처리 과정은 다음 그림과 같습니다.

▲ **그림 9-15** 전처리 과정

9.2.1 결측치 확인

결측치는 다음 표의 성춘향에 대한 '몸무게'처럼 주어진 데이터셋에서 데이터가 없는(NaN) 것입니다. 결측치 확인 및 처리는 다음 방법을 이용합니다.

9.2.2 토큰화

토큰화(tokenization)는 주어진 텍스트를 단어/문자 단위로 자르는 것을 의미합니다. 따라서 토큰화는 문장 토큰화와 단어 토큰화로 구분됩니다. 예를 들어 'A cat is on the sofa'라는 문장이 있을 때 단어 토큰화를 진행하면 각각의 단어인 'A', 'cat', 'is', 'on', 'the', 'sofa'로 분리됩니다.

9.2.3 불용어 제거

불용어(stop word)란 문장 내에서 빈번하게 발생하여 의미를 부여하기 어려운 단어들을 의미합니다. 예를 들어 'a', 'the' 같은 단어들은 모든 구문(phrase)에 매우 많이 등장하기 때문에 아무런 의미가 없습니다. 특히 불용어는 자연어 처리에 있어 효율성을 감소시키고 처리 시간이 길어지는 단점이 있기 때문에 반드시 제거가 필요합니다.

9.2.4 어간 추출

어간 추출(stemming)과 표제어 추출(lemmatization)은 단어 원형을 찾아 주는 것입니다. 예를 들어 '쓰다'의 다양한 형태인 writing, writes, wrote에서 write를 찾는 것입니다.

어간 추출은 단어 그 자체만 고려하기 때문에 품사가 달라도 사용 가능합니다. 예를 들어 어간 추출은 다음과 같이 사용됩니다.

- Automates, automatic, automation → automat

반면 **표제어 추출**은 단어가 문장 속에서 어떤 품사로 쓰였는지 고려하기 때문에 품사가 같아야 사용 가능합니다. 예를 들어 다음 표제어 추출이 가능합니다.

- am, are, is → be
- car, cars, car's, cars' → car

즉, 어간 추출과 표제어 추출은 둘 다 어근 추출이 목적이지만, 어간 추출은 사전에 없는 단어도 추출할 수 있고 표제어 추출은 사전에 있는 단어만 추출할 수 있다는 점에서 차이가 있습니다.

NLTK의 어간 추출로는 대표적으로 포터(porter)와 랭커스터(lancaster) 알고리즘이 있습니다. 이 둘에 대한 차이를 코드로 확인해 보겠습니다.

포터 알고리즘과 다르게 랭커스터 알고리즘은 단어 원형을 알아볼 수 없을 정도로 축소시키기 때문에 정확도가 낮습니다. 따라서 일반적인 상황보다는 데이터셋을 축소시켜야 하는 특정 상황에서나 유용합니다.

표제어 추출

일반적으로 어간 추출보다 표제어 추출의 성능이 더 좋습니다. 품사와 같은 문법뿐만 아니라 문장 내에서 단어 의미도 고려하기 때문에 성능이 좋습니다. 하지만 어간 추출보다 시간이 더 오래 걸리는 단점이 있습니다.

표제어 추출은 WordNetLemmatizer를 주로 사용합니다.

9.2.5 정규화

데이터셋이 가진 특성(혹은 칼럼)의 모든 데이터가 동일한 정도의 범위(스케일 혹은 중요도)를 갖도록 하는 것이 정규화(normalization)입니다.

머신 러닝/딥러닝은 데이터 특성들을 비교하여 패턴을 분석합니다. 이때 각각의 데이터가 갖는 스케일 차이가 크면 어떤 결과가 나타날까요? 예를 들어 다음과 같은 데이터셋이 있다고 가정해 봅시다. MonthlyIncome은 0~10000의 범위를 갖지만, RelationshipSatisfaction은 0~5의 범위를 갖습니다. 즉, MonthlyIncome과 RelationshipSatisfaction은 상당히 다른 값의 범위를 갖

는데, 이 상태에서 데이터를 분석하면 MonthlyIncome 값이 더 크기 때문에 상대적으로 더 많은 영향을 미치게 됩니다. 하지만 중요한 것은 값이 크다고 해서 분석에 더 중요한 요소라고 간주할 수 없기 때문에 정규화가 필요한 것입니다.

① **MinMaxScaler()**: 모든 칼럼이 0과 1 사이에 위치하도록 값의 범위를 조정합니다. 이때 특정 범위에서 많이 벗어난 데이터(이상치)의 경우 좁은 범위로 압축될 수 있습니다. 즉, 이상치에 매우 민감할 수 있기 때문에 주의해야 합니다.

MinMaxScaler()를 구하는 공식은 다음과 같습니다.

$$MinMaxScaler() = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

(x: 입력 데이터)

② **StandardScaler()**: 각 특성의 평균을 0, 분산을 1로 변경하여 칼럼 값의 범위를 조정합니다.

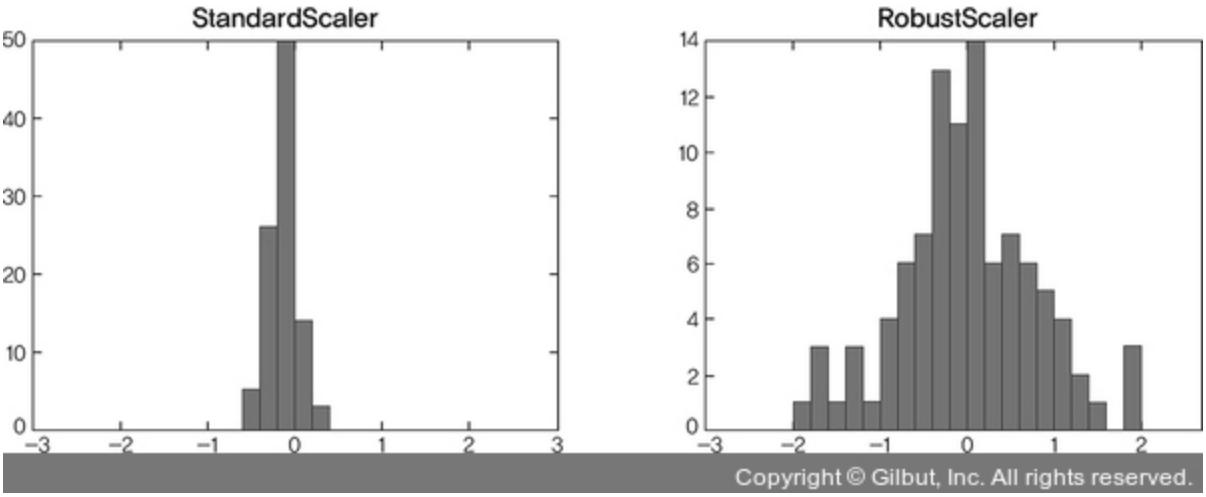
StandardScaler()를 구하는 공식은 다음과 같습니다.

$$StandardScaler() = \frac{x - \mu}{\sigma}$$

(x: 입력 데이터, μ: 평균, σ: 표준편차)

정규화 방법은 예제에서 구현한 MinMaxScaler(), StandardScaler() 외에도 두 가지가 더 있습니다.

- **RobustScaler():** 평균과 분산 대신 중간 값(median)과 사분위수 범위(InterQuartile Range, IQR)를 사용합니다. StandardScaler()와 비교하면 그림 9-18과 같이 정규화 이후 동일한 값이 더 넓게 분포되어 있는 것을 확인할 수 있습니다.



▲ **그림 9-18** StandardScaler와 RobustScaler 비교

MaxAbsScaler(): 절댓값이 0~1 사이가 되도록 조정합니다. 즉, 모든 데이터가 -1~1의 사이가 되도록 조정하기 때문에 양의 수로만 구성된 데이터는 MinMaxScaler()와 유사하게 동작합니다. 또한, 큰 이상치에 민감하다는 단점이 있습니다.

Note ≡ | 사분위수 범위(IQR)

사분위수란 전체 관측 값을 오름차순으로 정렬한 후 전체를 사등분하는 값을 나타냅니다. 따라서 다음과 같이 표현할 수 있습니다.

- 제1사분위수 = Q1 = 제25백분위수
- 제2사분위수 = Q2 = 제50백분위수
- 제3사분위수 = Q3 = 제75백분위수

이때 제3사분위수와 제1사분위수 사이 거리를 데이터가 흩어진 정도의 척도로 사용할 수 있는데, 이 수치를 사분위수 범위(IQR)라고 합니다. 따라서 사분위수 범위는 다음과 같이 표현할 수 있습니다.

사분위수 범위: IQR = 제3사분위수 - 제1사분위수 = Q3 - Q1