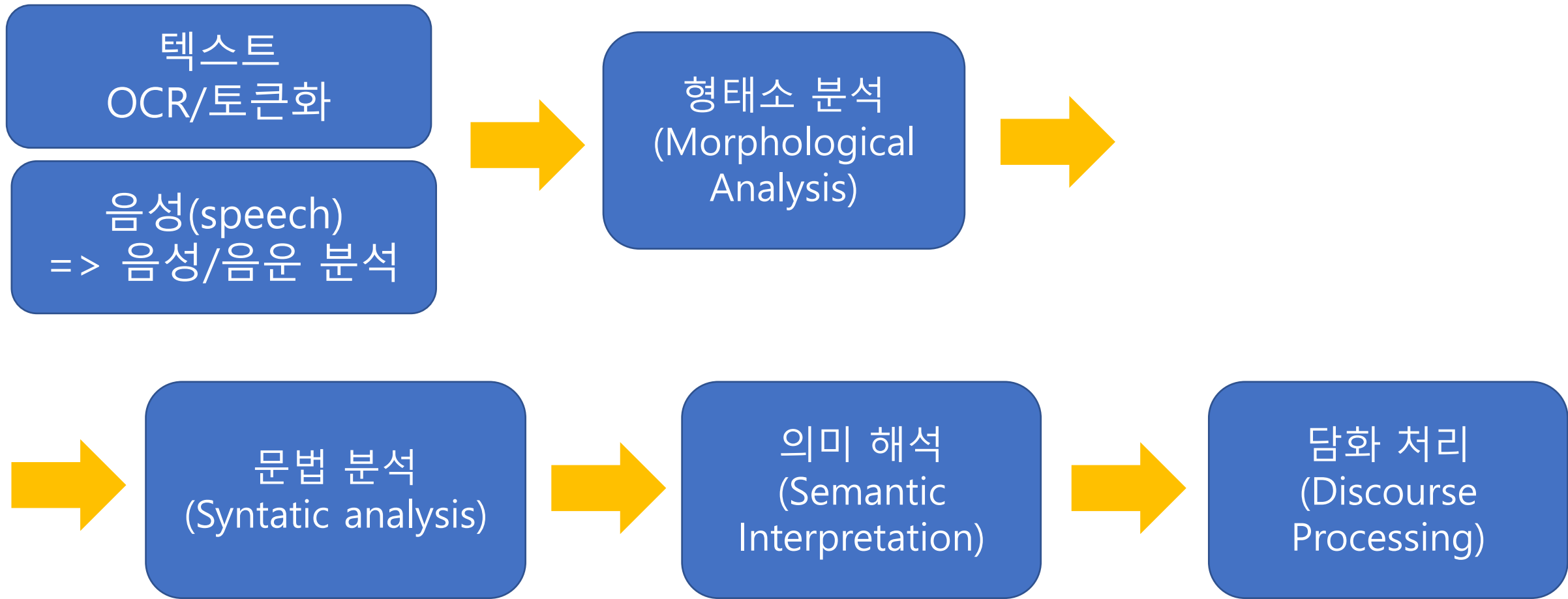


# 자연어 처리

(with Python)

# 1-1 자연어 처리 단계



# 1-2 자연어 처리의 응용분야

## (1) 간단한 응용

- A. 철자 검사
- B. 키워드 검색
- C. 유사어 검색

## (2) 중급 응용

- A. 웹 사이트로부터 정보 추출
- B. 감정 분석

## (3) 고급 응용

- A. 기계 번역
- B. 음성 인식
- C. 챗봇/음성 대화 시스템
- D. 복잡한 질의 응답

# 1-3 형태소에 대해 알아보기

**(1) 형태소는 의미를 전달하는 기본 단위이다.**

**(2) 종류**

A. 어간

B. 접사

C. 영문

# 1-4 기본 용어 정리

## (5) 품사(Part-of-Speech, POS, Word class)

- A. 단어를 기능, 형태, 의미에 따라 나눈 갈래
- B. 9가지 또는 22가지로 분류

## (6) 품사 태깅(Part of Speech Tagging)

- A. 문장을 형태소나 품사별로 나누어 줌.

## (7) 문장(통사, statement)

- A. 생각이나 감정을 말과 글로 표현할 때 완결된 내용을 나타내는 최소 단위

## (8) 담화(discourse)

- A. 둘 이상의 문장이 연속되어 이루어지는 말의 단위

## (9) 말뭉치(코퍼스, Corpus)

- A. 대량의 구조화된 텍스트(a large and structured set of texts)

# 1-4 기본 용어 정리

## (1) 음소(Phoneme)

- A. 더 이상 작게 나눌 수 없는 음운론상의 최소 단위 (예) ㅎ ㅏ ㄱ
- B. 하나 이상의 음소가 모여서 음절을 이룬다.

## (2) 음절(Syllable)

- A. 몇 개의 음소로 이루어지며, 모음은 단독으로 한 음절이 되기도 함.
- B. 하나의 종합된 음의 느낌을 주는 것 (예) 합

## (3) 형태소(Morpheme)

- A. 뜻을 가진 가장 작은 말의 단위
- B. '이야기 책'의 '이야기', '책'

## (4) 단어(Word)

- A. 형태소들로 구성되는 분리하여 자립적으로 쓸 수 있는 말.

# 1-5 자연어 처리 라이브러리

## (1) NLTK

- A. 자연어 처리를 위한 오픈소스

## (2) KoNLPy

- A. 한국어 자연어 처리를 위한 파이썬 패키지

## (3) mecab

- A. 자연어 처리 C++ 라이브러리
- B. 띄워쓰기 처리가 없음.
- C. 은전 한 앞.
- D. KoNLP에서 은전 한 앞을 파이썬으로 옮김.

## (4) FudanNLP

- A. 중국어 처리 라이브러리
- B. Github : <https://github.com/xpqiu/fnlp>

# 1-5 자연어 처리 라이브러리

## (5) Stanford Natural Language Processing Group NLP

A. <http://nlp.stanford.edu/software/index.html>



# 1-6 자연어 처리 라이브러리(NLTK)

## (1) 구성 요소

- A. 50개 이상의 말뭉치(corpus)와 어휘(lexicon)가 있음.
- B. 분류, 토큰화, 스템밍(Stemming), 태깅(tagging), 구문 분석(parsing), 의미 추리(semantic reasoning) 라이브러리 API
- C. NLTK 공식 추천 온라인 Book

<http://www.nltk.org/book> (파이썬 3.x용)

[http://www.nltk.org/book\\_1ed](http://www.nltk.org/book_1ed) (파이썬 2.x용)

# 1-6 자연어 처리 라이브러리(NLTK)

## (1) 패키지 구성 요소

### 데이터

- A. `nltk.data` : 말뭉치 문법, 기타 저장된 객체와 같은 NLTK 지원 파일의 로드
- B. `nltk.downloader` : 말뭉치, 모델, 기타 패키지 다운로드
- C. `nltk.corpus` : 말뭉치(Corpus)와 어휘(lexicon)에 대한 표준화된 API

### 텍스트 분석

- A. `nltk.probability` : 확률 정보의 표현과 처리를 위한 클래스들
- B. `nltk.tokenize` : 토큰화
- C. `nltk.stem` : 스테밍
- D. `nltk.collocations` : 연어(collocation)검출: t-test, chi-squared, point-wise mutual information

# 1-6 자연어 처리 라이브러리(NLTK)

## (1) 패키지 구성 요소

### 데이터

- A. `nltk.data` : 말뭉치 문법, 기타 저장된 객체와 같은 NLTK 지원 파일의 로드
- B. `nltk.downloader` : 말뭉치, 모델, 기타 패키지 다운로드
- C. `nltk.corpus` : 말뭉치(Corpus)와 어휘(lexicon)에 대한 표준화된 API

### 텍스트 분석

- A. `nltk.probability` : 확률 정보의 표현과 처리를 위한 클래스들
- B. `nltk.tokenize` : 토큰화
- C. `nltk.stem` : 스테밍
- D. `nltk.collocations` : 연어(collocation)검출: t-test, chi-squared, point-wise mutual information  
연어? 어떤 언어 내에서 특정한 뜻을 나타낼 때 흔히 함께 쓰이는 단어들의 결합.

# 1-6 자연어 처리 라이브러리(NLTK)

## (1) 패키지 구성 요소

### 텍스트 분석

- A. `nltk.tag` : 품사(Part-of-speech:POS) 태깅(tagging) : n-gram, backoff, Brill, HMM, Tnt
- B. `nltk.chunk` : 정규 표현, n-gram, 개체명(고유명사)

### 구문 분석

`nltk.grammar`, `nltk.parse`, `nltk.app`, `nltk.ccg`

### 의미 분석

`Nltk.sem`, `nltk.inference`

### 기계학습

`nltk.classify` : 결정 트리, 나이브 베이즈, maximum entropy

# 1-6 자연어 처리 라이브러리(NLTK)

## (1) 패키지 구성 요소

### 기계학습

- A. nltk.classify 패키지 : 결정 트리, 나이브 베이즈, maximum entropy
- B. nltk.cluster 패키지 : [군집] k-means, EM
- C. nltk.tbi 패키지 : [변환](transformation) 기반 학습
- D. nltk.metrics 패키지 : precision, recall, agreement, coefficients

### 자연어 처리 응용

- A. nltk.sentiment 패키지 : 감성 분석
- B. nltk.translate 패키지 : 기계번역
- C. nltk.chat 패키지 : 간단한 챗봇(chatbot) API

## 1-7 토큰(Token)

- (1) 토큰(token) : 의미를 갖는 문자열(단어, 절, 문장 등)
- (2) 토크나이징(tokenizing) : 토큰을 나누는 작업
- (3) 영문-공백
- (4) 한글 - 합성어, 조사합성
- (5) 작업기준

## 1-8 정규식 참조

(1) <http://www.nextree.co.kr/p4327>

## 2-1 알아보기

**(1) Stemming**

**(2) Tagging**

**(3) WordCloud**

**(4) StopWord**

**(5) Tf-idf**



## 2-1 알아보기

- (1) Stemming : 정규화
- (2) Tagging : 토큰에 대한 품사 태깅.
- (3) WordCloud
- (4) StopWord
- (5) Tf-idf

## 2-3 형태소 분석기의 성능 비교

<https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/05/10/postag/>

## 2-4 Konlpy 사용

1. KKMA
2. Hannanum
3. Twitter : OpenKoreanText 오픈 소스 한국어 처리기
4. Eunjeon : 은전한닢 프로젝트
5. KOMORAN : Junsoo Shin 님의 코모란
6. 트위터 : 빠른 분석
7. 꼬꼬마 : 정확한 품사 정보 필요
8. 정확성, 시간 모두 중요 : 코모란

## 2-5 한국어 형태소 분석기

### C/C++

- KTS (1995) GPL v2
  - 이상호, 서정연, 오영환 (KAIST & 서강대)
  - code
- MACH (2002) custom
  - 심광섭 (성신여대)
- MeCab-ko (2013) GPL LGPL BSD
  - 이용운, 유명호

<https://konlpy-ko.readthedocs.io/ko/v0.4.3/references/#corpora> 참조

## 2-5 한국어 형태소 분석기

### 자바

- 아리랑 (2009) Apache v2
  - 이수명
  - code
- 한나눔 (1999) GPL v3
  - KAIST 최기선 교수 연구팀
  - code, docs
- 꼬꼬마 (2010) GPL v2
  - 서울대 이상구 교수 연구팀
  - 동적 프로그래밍을 이용해 형태소 후보를 찾음
  - 형태소의 주변을 확인하고, 몇몇 휴리스틱을 사용하고, HMM을 사용하는 방식으로 품사를 태깅함
  - 개발자 블로그: 이동주
- KOMORAN (2013) Apache v2
  - By *shineware*

<https://konlpy-ko.readthedocs.io/ko/v0.4.3/references/#corpora> 참조

## 2-5 한국어 형태소 분석기

### 파이썬

- KoNLPy (2014) GPL v3
  - 박은정 (서울대)
- UMorpheme (2014) MIT
  - 김경훈 (UNIST)

### R

- KoNLP (2011) GPL v3
  - 전희원

<https://konlpy-ko.readthedocs.io/ko/v0.4.3/references/#corpora> 참조

## 2-5 한국어 형태소 분석기

### 그 외

- K-LIWC (아주대)
- KRISTAL-IRMS (KISTI)
  - 개발 후기
- Korean XTAG (UPenn)
- HAM (국민대)
- POSTAG/K (포스텍)
- Speller (부산대)
- UTagger (울산대)
- (No name) (고려대)

<https://konlpy-ko.readthedocs.io/ko/v0.4.3/references/#corpora> 참조

## 2-5 한국어 형태소 분석기

### 다른 NLP 도구

- Hangulize - By Heungsub Lee Python
  - Hangul transcription tool to 38+ languages
- Hanja - By Sumin Byeon Python
  - Hanja to hangul transcriptor
- Jamo - By Joshua Dong Python
  - Hangul syllable decomposition and synthesis
- KoreanParser - By DongHyun Choi, Jungyeul Park, Key-Sun Choi (KAIST) Java
  - 언어 파서
- Korean - By Heungsub Lee Python
  - Package for attaching particles (josa) in sentences

<https://konlpy-ko.readthedocs.io/ko/v0.4.3/references/#corpora> 참조



## 2-5 한국어 형태소 분석기

### 말뭉치

- 연세 말뭉치, 연세대, 1987.
  - 1960년 이후 한국어에 대한 4200만 어절
- 고려대학교 한국어 말뭉치, 1995
  - 1970-90년대 한국어에 대한 1000만 어절
- HANTEC 2.0, KISTI & 충남대, 1998-2003.
  - 12만 개의 테스트 문서 (237MB)
  - QA를 위한 50개의 TREC 형태 질의
- HKIB-40075, KISTI & 한국일보, 2002.
  - 텍스트 분류를 위한 40,075 테스트 문서 (88MB)
- KAIST Corpus, KAIST, 1997-2005.
- Sejong Corpus, National Institute of the Korean Language, 1998-2007.

<https://konlpy-ko.readthedocs.io/ko/v0.4.3/references/#corpora> 참조

## 2-6 NLP 관련 사이트

- [Google NLP publications](#)
- [Lingpipe](#)
- [Microsoft NLP group \(Redmond\)](#)
- [부산대 NLP 관련사이트 목록](#)
- [Sejong semantic search system](#)
- [한글 및 한국어 정보처리 학술대회](#)

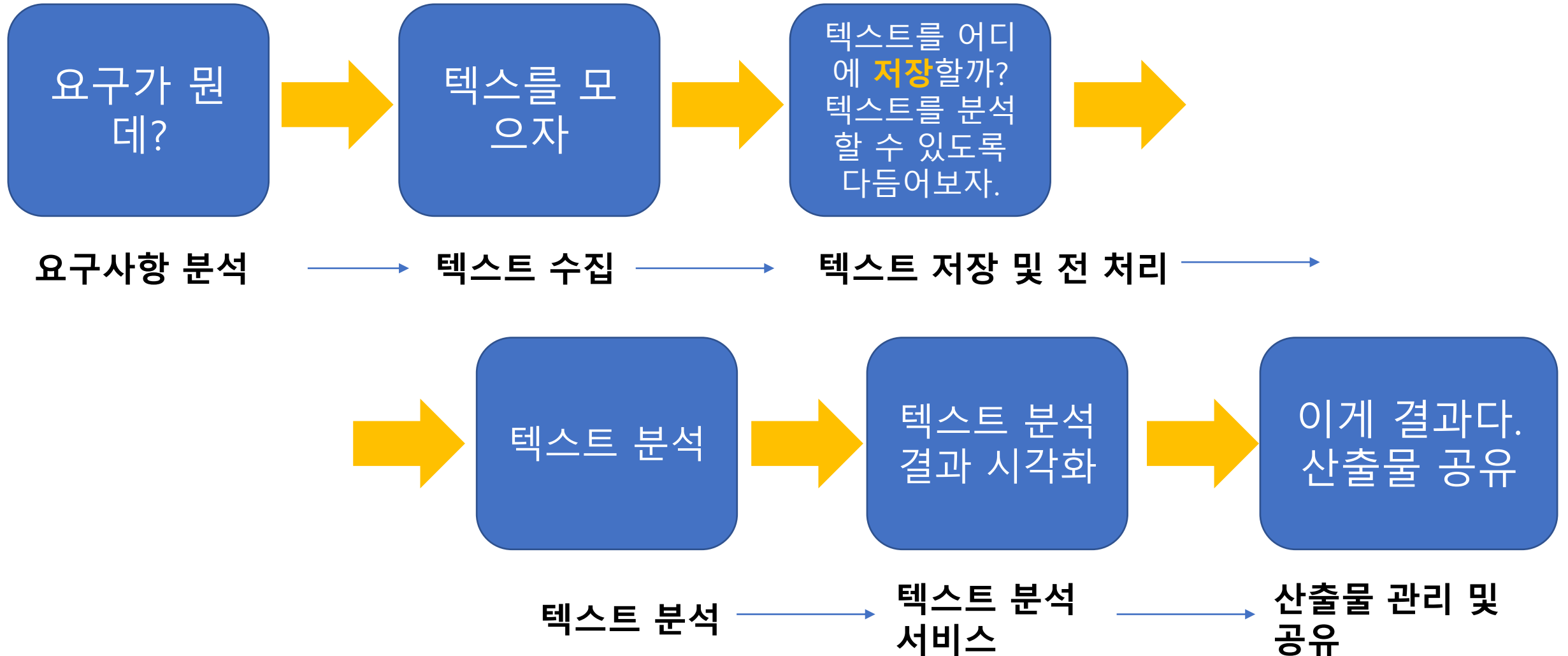
<https://konlpy-ko.readthedocs.io/ko/v0.4.3/references/#corpora> 참조

## 2-6 NLP 관련 사이트

- [Google NLP publications](#)
- [Lingpipe](#)
- [Microsoft NLP group \(Redmond\)](#)
- [부산대 NLP 관련사이트 목록](#)
- [Sejong semantic search system](#)
- [한글 및 한국어 정보처리 학술대회](#)

<https://konlpy-ko.readthedocs.io/ko/v0.4.3/references/#corpora> 참조

# 1-3 텍스트 분석 절차

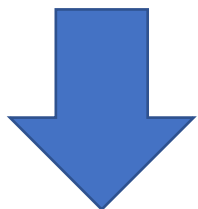


# 1-2 데이터 마이닝 vs 텍스트 마이닝

## ▶ 데이터 마이닝

이름	도시	나이
김철수	서울	45
김흥수	경기	20
임수하	원주	15

정형 데이터



유용하고 가치 있는 정보 추출

## ▶ 텍스트 마이닝

비정형 데이터

From today's featured article

"X-Cops" is the twelfth episode of the seventh season of the American science fiction television series *The X-Files*. Directed by Michael Watkins and written by Vince Gilligan, the installment originally aired on the Fox network in February 2000. In this episode, Fox Mulder (David Duchovny) and Dana Scully (Gillian Anderson), special agents for the Federal Bureau of Investigation, are interviewed for the Fox network reality television program *Cops* during an X-Files investigation. Mulder, hunting what he believes to be a werewolf, discovers that the monster terrorizing people craves the fear it provokes. While Mulder embraces the publicity of *Cops*, Scully is uncomfortable about appearing on national television. "X-Cops" is one of only two *X-Files* episodes that was shot in real time. The episode has been thematically analyzed for its use of postmodernism and its presentation as reality television. It has been named among the best episodes of *The X-Files* by several reviewers, for its humor and format. (Full article...)

Recently featured: Battle of Winterthur · 38th (Welsh) Infantry Division · *New Worlds* (magazine)

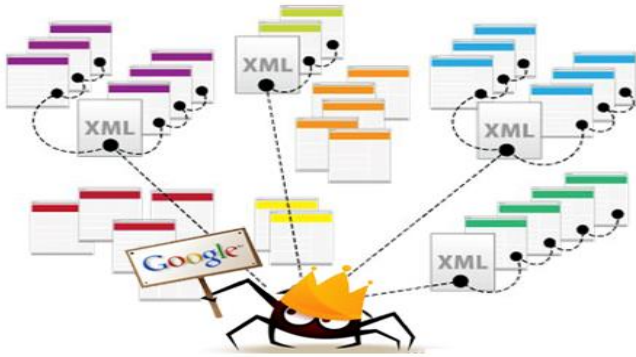
Are you subscribed by email · More featured articles



개체명(인명, 지역명 등), 패턴 혹은 단어-문장 관계 정보 추출

텍스트 분석 어떻게 해야 할까요?

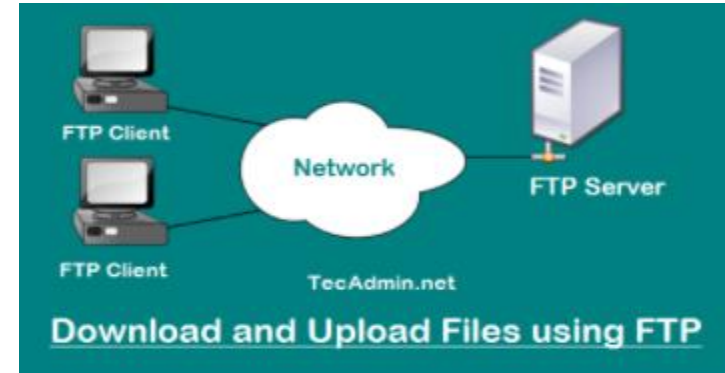
# 1-4 텍스트 수집



## Crawling

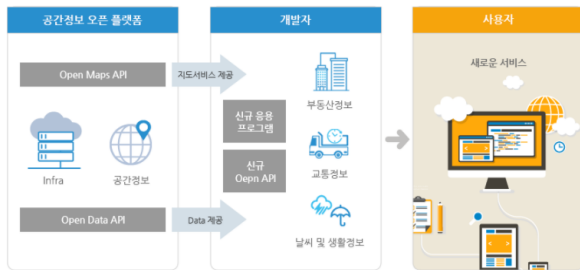


## Scraping



## FTP

공간정보 오픈플랫폼 개발자센터는 국가 공간정보의 개방, 공유, 참여를 통해 공간정보의 자율적이고 창조적인 다양한 애플리케이션을 개발할 수 있도록 2D/3D, 원격 오픈API 서비스와 기술을 제공합니다.

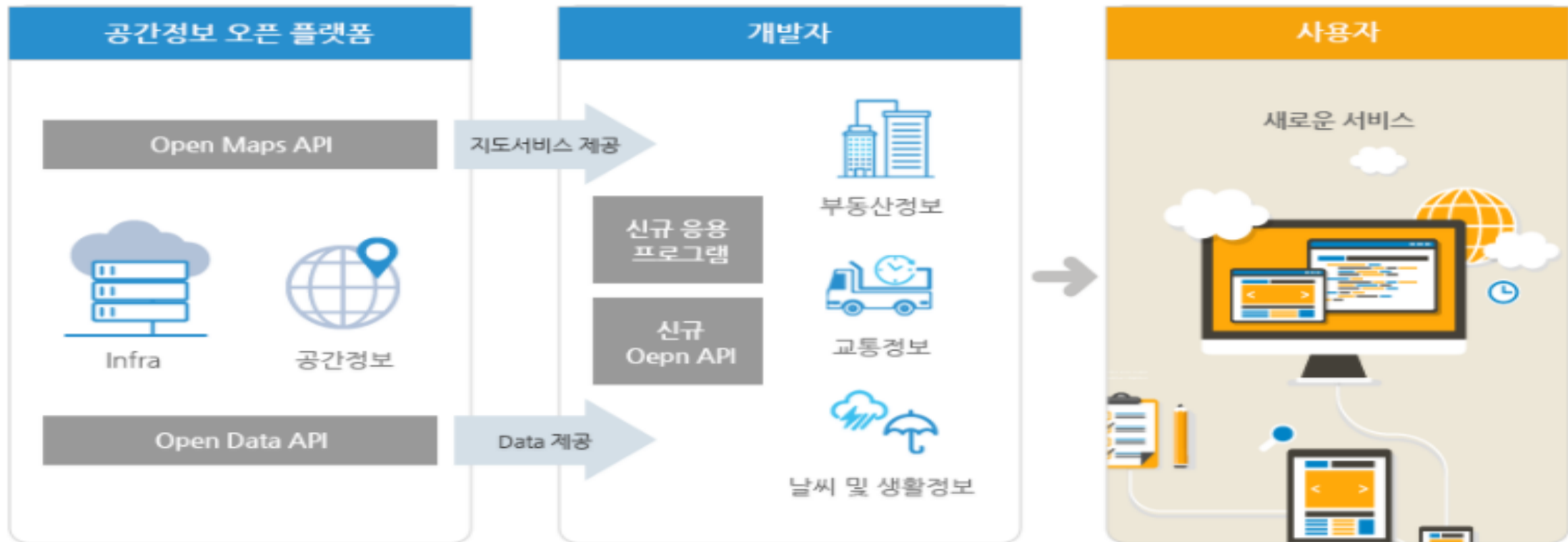


## 오픈 API

Ref : <https://www.promptcloud.com/blog/all-you-need-to-know-about-web-crawling>  
<https://nocodewebscraping.com/web-scraping-for-dummies-tutorial-with-import-io-without-coding/>  
[http://dev.vworld.kr/dev/v4dv\\_apiuse\\_s001.do](http://dev.vworld.kr/dev/v4dv_apiuse_s001.do)

# 1-5 텍스트 수집 – 오픈 API 이용하기

공간정보 오픈플랫폼 개발자센터는 국가 공간정보의 개방, 공유, 참여를 통해 공간정보의 자율적이고 창조적인 다양한 애플리케이션을 개발할 수 있도록 2D/3D, 검색 오픈API 서비스와 기술을 제공합니다.



## 오픈 API

Ref : [http://dev.vworld.kr/dev/v4dv\\_apiuse\\_s001.do](http://dev.vworld.kr/dev/v4dv_apiuse_s001.do)



텍스트를 분석하기 위해 텍스트를 어떻게  
분리할 수 있을까요?

## 2-1 형태소 분석

- 주어진 텍스트를 단어와 문법적 특성에 맞추어 명사, 동사, 꾸밈어, 조사 등의 형태소로 분리할 수 있다.

## 2-1 형태소 분석

### ▶ 형태소란 무엇인가?(Morphology)

- **형태소란**, 의미가 있는 최소 단위로서 더 이상 분리가 불가능한 가장 작은 의미 요소. 즉 일반적으로 문법적, 관계적인 뜻을 나타내는 단어 또는 단어의 부분이다.  
(매일경제 기획팀, 서울대 빅데이터 센터 참조)

## 2-1 형태소 분석

### ▶ 형태소 분석이란?

- 형태소 분석이란,

주어진 단어 또는 어절을 구성하는 각 형태소를 분리한 후, 분리된 형태소의 기본형 및 품사 정보를 추출.

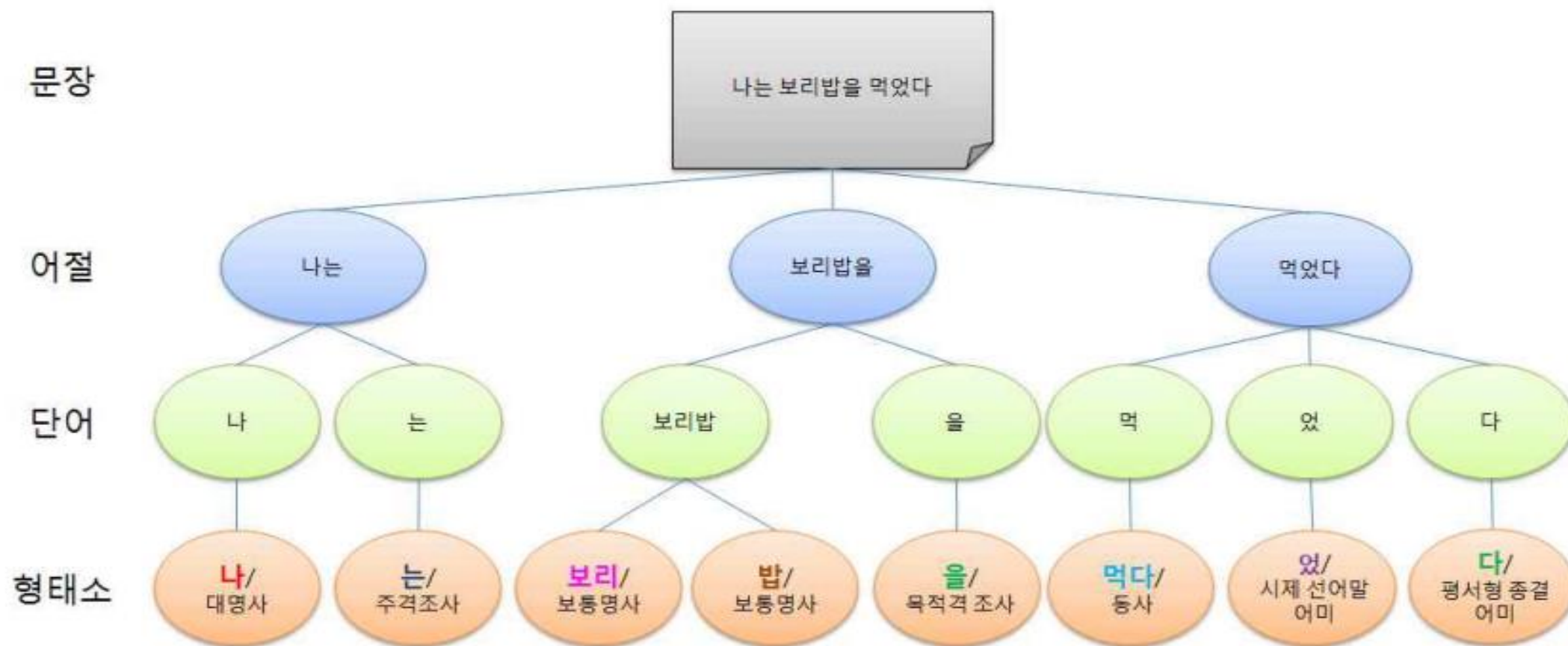


나는 보리밥을 먹었다

[그림 2-1] 형태소 예시

## 2-1 형태소 분석

### ▶ 문장, 어절, 단어, 형태소의 계층 구조



[그림 2-2] 문장, 어절, 단어, 형태소의 계층 구조

## 2-1 형태소 분석

- ▶ 형태소 개념 이해
- ▶ 텍스트 전처리(pre-processing) 이해
- ▶ 품사 태깅 이해

## 2-2 텍스트 전처리(text pre-processing)

- 텍스트 전처리 과정은 텍스트 분석을 위해 문장 분리, 불필요한 문장 성분을 제거하는 과정이다.
- 영미권에서는 예를 들면 대문자를 소문자로 변환하는 작업

## 2-2 텍스트 전처리(text pre-processing)



[그림 2-3] 전처리(pre-processing) 결과



## 2-3 품사 태깅

- 품사 태깅이란, (POS tagging, Part-Of-Speech tagging)  
하나의 단어가 여러 품사를 갖는다. 따라서 품사의 모호성 (혹은 중의성)을 제거하는 과정이 필요. 이를 수행하는 과정을 품사 태깅이라 한다.

## 2-3 품사 태깅



[그림 2-4] 품사 태깅 예시

## 2-4 키워드 추출

### ▶ 가용어의 이해

불용어가 아닌 단어들

### ▶ 불용어의 이해

단어 성분 중에서 문서의 정보(의미)를 표현하지 못하는 단어.  
즉, 문서와 관련이 없다.

### ▶ 키워드의 개념

가용어 중의 중심이 되는 단어

키워드 선정을 해 보자.  
어떻게 해야 할까

## 2-5 키워드 선정

일반적으로 분석하고자 하는 목적 및 데이터 세트에 영향을 받지만,

문서 내에서 발생 빈도가 높은 단어들을 키워드로 선정한다.

## 2-6 불용어, 가용어, 키워드

### ▶ 불용어

한국어 : 조사 ( '는', '을' 등)

영어 : '관사', '전치사' ('a', 'the', 'on', 'with')

### ▶ 키워드

불용어는 빈도가 높다고 키워드가 아니다.

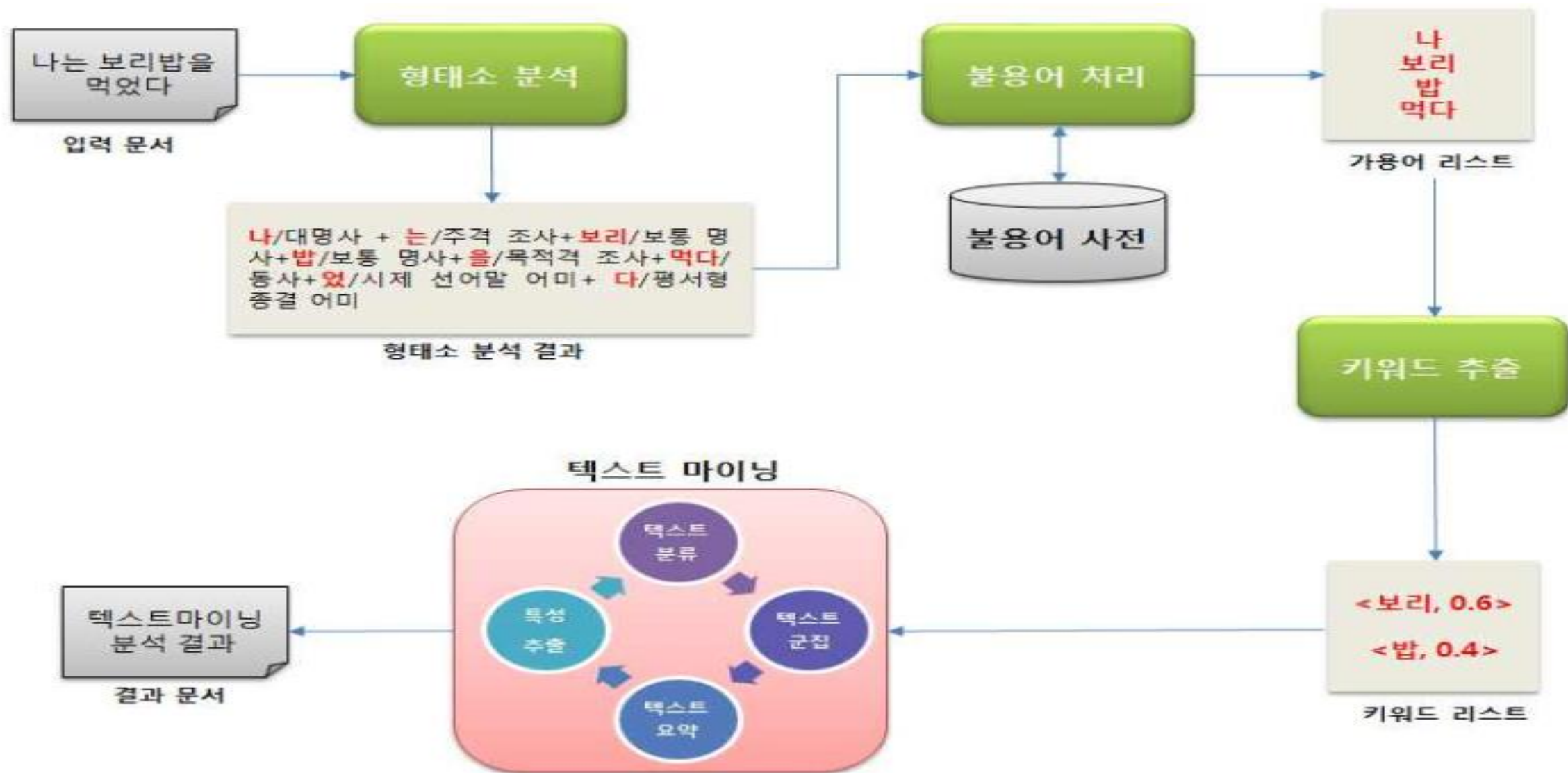
## 2-6 불용어 처리

- 불용어가 저장된 데이터 베이스를 참조하여 키워드 제거.

=> 형태소 분석 결과를 **불용어 사전에서 검색**하여 일치하는 내용이 나타나면 그 **내용을 삭제**한다.

## 2-6 키워드 추출(Text Wording) 절차

텍스트모델구축하기





- > 각 분야에서 필요로 하는 연구 재료로서 자료의 집합을 말한다.
- > 목적에 따라 작게는 소설 한편, 수십억 어절 이상의 말 또는 글로 표현된 자료 모음
- > 전산학적 말뭉치는 글로 표현된 자료에 대해 텍스트 정제, 통합, 변환의 절차를 거쳐서 구조화된 형태의 자료를 의미

(예제) 2개 이상의 언어로 만든 병렬 말뭉치. 영어 소설의 원본과 한국의 번역본의 데이터로 말뭉치를 만들면 **한영 소설어의 병렬 말뭉치**가 된다.

### ▶ 말뭉치 개념

#### ■ 말뭉치란?

- (1) 각 분야의 필요로 하는 연구 재료이다.
- (2) 언어의 본질적인 모습을 보여주는 자료의 집합

### ▶ 말뭉치 개념(전산학적)

#### ■ 말뭉치란?(Corpus)

- (1) 대규모 언어 데이터 베이스
- (2) 인간의 음성 언어(문어, 구어)를 대용량 컴퓨터에 저장하고 이를 필요에 따라 가공하여 언어 연구에 사용
- (3) 컴퓨터가 판독할 수 있는 형태(machine-readable form)

### ▶ 말뭉치 종류

#### ■ 가공 방법에 따른 종류

(1) 원시 말뭉치(raw corpus)

(2) 가공된 말뭉치(tagged corpus)

#### ■ 작성 방법에 따른 종류

(1) 샘플 말뭉치 : 텍스트를 일정량만 수집한 것으로 텍스트의 내용이 고정된 말뭉치

(2) 모니터 말뭉치 : 변화하는 언어의 실태 추적을 위한 것. 낡은 자료를 제외한 항상 새로운 정보를 수집 후, 최신 언어 정보를 데이터베이스화한 말뭉치

### ▶ 말뭉치 종류

- 기타

범용 말뭉치

특수목적 말뭉치

공시 말뭉치

통시 말뭉치

병렬 말뭉치....

## 2-8 단어와 문서 관계 표현

- > 말뭉치로부터 단어와 문서의 관계를 표현하기 위해 문서-단어 행렬 혹은 단어-문서 행렬을 작성할 수 있다.
- > 문서-단어 행렬 혹은 단어-문서 행렬로부터 단어의 빈도와 연관성등의 기본 집계 수행이 가능하다.

## 2-8 단어와 문서 관계 표현

### ▶ 단어-문서 행렬(문서-단어 행렬)

- (1) 각 단어는 문서의 의미를 나타내는 **가장 기본적인 단위**
- (2) 이러한 관계를 텍스트 마이닝에서 단어-문서 행렬로 표현. 이를 통해 **단어들 사이의 포함관계 쉽게 조망 가능**

## 2-8 단어와 문서 관계 표현

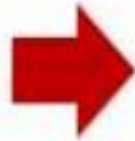
### ▶ 단어-문서 행렬예시

단어가 행을 구성,  
문서가 열을 구성한다.  
단어 포함(1), 불포함(0)

입력 텍스트 1: "파스타 먹방, 강남 파스타 데이트"

입력 텍스트 2: "강남 버스 파스타 맛집"

입력 텍스트 3: "강남 버스, 강남 파스타, 강남 맛집"



	입력텍스트 1	입력텍스트 2	입력텍스트 3	대상 텍스트 ←
파스타	1	1	1	
먹방	1	0	0	
강남	1	1	1	
데이트	1	0	0	
버스	0	1	1	
맛집	0	1	1	
...	...	...	...	

↑ 말뭉치에서 추출된 단어들

[그림 2-12] 단어-문서 행렬 예시



하나의 문서에서 단어의 중요성을 확인할 수 없을까?

## 2-9 하나의 문서에서 단어의 중요성 확인

텍스트모델구축하기

- ▶ TF(단어빈도) 특정한 단어가 문서 내에 얼마나 자주 등장하는가?
  - 이 값이 높을수록 문서에서 중요하다.

- ▶ DF(문서빈도, document frequency)

- 특정단어를 포함하고 있는 문서의 수
- df가 높다는 것은 많은 문서에서 나타난다.  
검색에서 별로 중요한 단어가 아니다.  
검색단어는 내가 필요한 문서에서 많이 나타나고 다른 문서에서 적게 나와야 정확도 높은 검색이 가능함.

파스타는 3개 문서 있으므로 3  
먹방은 1개 문서 있으므로 1  
강남은 3개 문서 있으므로 3

	입력텍스트 1	입력텍스트 2	입력텍스트 3	대상 텍스트 ←
파스타	1	1	1	
먹방	1	0	0	
강남	1	1	1	
데이트	1	0	0	
버스	0	1	1	
맛집	0	1	1	
...	...	...	...	

↑ 말뭉치에서 추출된 단어들

## 2-9 하나의 문서에서 단어의 중요성 확인

- ▶ IDF(역문서빈도) df(문서빈도)에서 역수를 취한 값.

$$\text{IDF} = \log\left(\frac{\text{전체 문서 수}}{\text{단어를 포함한 문서 수}}\right)$$



Log의 분모 부분이 0이 되면 안되기에 + 1

$$\text{IDF} = \log\left(\frac{\text{전체 문서 수}}{\text{단어를 포함한 문서 수} + 1}\right)$$

## 2-9 TF-IDF

### Term Frequency Inverse Document Frequency weighting scheme

- ▶ TF(단어빈도) \* IDF(문서빈도의 역수)

$$\text{공식 : } \text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

- ▶ 문서 내에 단어들에 각각 부여한 중요도를 나타내는 숫자 값.

- ▶ 어디에 사용되는가?

TFIDF 값을 부여한 후, 코사인 유사도(Cosine Similarity)등을 이용해서 문서들의 유사도를 구하는데 흔히 사용됨.

TFIDF 값을 계산을 통해 문서 내에서 상대적으로 더 중요한 단어가 무엇인지를 알 수 있다.

## 2-9 TF-IDF (버스)

	입력텍스트 1	입력텍스트 2	입력텍스트 3
파스타	0	0	0
데이트	0.24	0	0
버스	0	0.044	0.029
맛집	0	0.044	0.029
***	***	***	***

입력 텍스트 1 : "파스타 먹방, 강남 파스타 데이트 "

입력 텍스트 2 : "강남 버스 파스타 맛집 "

입력 텍스트 3 : "강남 버스, 강남 파스타, 강남 맛집 "

### TF(Term frequency)

입력 텍스트 1

$$\frac{0}{5} = 0$$

입력 텍스트 2

$$\frac{1}{4} = 0.25$$

입력 텍스트 3

$$\frac{1}{6} = 0.1667$$

$$TF = \frac{\text{문서내 단어수}}{\text{문서 내 모든 단어 수}}$$

$$IDF = \log\left(\frac{\text{전체 문서 수}}{\text{단어를 포함한 문서 수}}\right)$$

### IDF(inverse document frequency)

$$IDF = \log_{10} \left(\frac{3}{2}\right) = 0.1761$$

$$TF-IDF(D2) = 0.25 * 0.1761 = 0.044$$

## 2-9 TF-IDF (데이트)

	입력텍스트 1	입력텍스트 2	입력텍스트 3
파스타	0	0	0
먹방	0.0954	0	0
강남	0	0	0
데이트	0.0954	0	0
버스	0	0.044	0.029
맛집	0	0.044	0.029
...	...	...	...

$$TF = \frac{\text{문서내 단어수}}{\text{문서 내 모든 단어 수}}$$

$$IDF = \log\left(\frac{\text{전체 문서 수}}{\text{단어를 포함한 문서 수}}\right)$$

입력 텍스트 1 : "파스타 먹방, 강남 파스타 **데이트**"

입력 텍스트 2 : "강남 버스 파스타 맛집 "

입력 텍스트 3 : "강남 버스, 강남 파스타, 강남 맛집 "

### TF(Term frequency)

입력 텍스트 1

$$\frac{1}{5} = 0.2$$

입력 텍스트 2

$$\frac{0}{4} = 0$$

입력 텍스트 3

$$\frac{0}{6} = 0$$

### IDF(inverse document frequency)

$$IDF = \log_{10}\left(\frac{3}{1}\right) = 0.477$$

$$TF-IDF(D1) = 0.2 * 0.477 = 0.0954$$

## 2-9 TF-IDF (먹방)

	입력텍스트 1	입력텍스트 2	입력텍스트 3
파스타	0	0	0
먹방	0.0954	0	0
강남	0	0	0
데이트	0.0954	0	0
버스	0	0.044	0.029
맛집	0	0.044	0.029
...	...	...	...

$$TF = \frac{\text{문서내 단어수}}{\text{문서 내 모든 단어 수}}$$

$$IDF = \log\left(\frac{\text{전체 문서 수}}{\text{단어를 포함한 문서 수}}\right)$$

입력 텍스트 1 : "파스타 **먹방**, 강남 파스타 데이트 "

입력 텍스트 2 : "강남 버스 파스타 맛집 "

입력 텍스트 3 : "강남 버스, 강남 파스타, 강남 맛집 "

### TF(Term frequency)

입력 텍스트 1

$$\frac{1}{5} = 0.2$$

입력 텍스트 2

$$\frac{0}{4} = 0$$

입력 텍스트 3

$$\frac{0}{6} = 0$$

### IDF(inverse document frequency)

$$IDF = \log_{10}\left(\frac{3}{1}\right) = 0.477$$

$$TF-IDF(D1) = 0.2 * 0.477 = 0.0954$$

## 2-9 TF-IDF (먹방) – log2을 사용시(R)

텍스트모델구축하기

	Docs		
Terms	D1	D2	D3
강남	0.0000000	0.0000000	0.000000000
데이트	0.3169925	0.0000000	0.000000000
먹방	0.3169925	0.0000000	0.000000000
파스타	0.0000000	0.0000000	0.000000000
맛집	0.0000000	0.1462406	0.09749375
버스	0.0000000	0.1462406	0.09749375

입력 텍스트 1 : "파스타 **먹방**, 강남 파스타 데이트 "

입력 텍스트 2 : "강남 버스 파스타 맛집 "

입력 텍스트 3 : "강남 버스, 강남 파스타, 강남 맛집 "

### TF(Term frequency)

입력 텍스트 1

$$\frac{1}{5} = 0.2$$

입력 텍스트 2

$$\frac{0}{4} = 0$$

입력 텍스트 3

$$\frac{0}{6} = 0$$

$$TF = \frac{\text{문서내 단어수}}{\text{문서 내 모든 단어 수}}$$

$$IDF = \log\left(\frac{\text{전체 문서 수}}{\text{단어를 포함한 문서 수}}\right)$$

### IDF(inverse document frequency)

$$IDF = \log_2\left(\frac{3}{1}\right) = 1.5849$$

$$TF-IDF(D1) = 0.2 * 1.5849 = 0.3169$$



## 2-9 TF-IDF – 영문서(blue)- relative tf-idf (R에서)

(실습 1) BLUE에 대한 TF-IDF를 구해보자.

**불용어 처리 후, 입력 문서**

입력 텍스트 1 : "The sky blue"

입력 텍스트 2 : "The sun bright"

입력 텍스트 3 : "The sun sky bright"

$$TF = \frac{\text{문서내 단어수}}{\text{문서 내 모든 단어 수}}$$

$$IDF = \log\left(\frac{\text{전체 문서 수}}{\text{단어를 포함한 문서 수}}\right)$$

**TF(Term frequency)**

입력 텍스트 1    입력 텍스트 2    입력 텍스트 3

$$\frac{1}{3} = 0.333$$

$$\frac{0}{4} = 0$$

$$\frac{0}{4} = 0$$

**IDF(inverse document frequency)**

$$IDF = \log_2\left(\frac{3}{1}\right) = 1.5849$$

$$TF-IDF \Rightarrow 0.33333 * 1.5849 = 0.5283$$

**R에서 IDF를 구할 때,  $\log_{10}$ 이 아닌  $\log_2$  를 사용한다.  
이를 relative tf-idf라고 한다.**

## 2-9 TF-IDF – 영문서(bright)

(실습 2) bright에 대한 TF-IDF를 구해보자.

불용어 처리 후, 입력 문서

입력 텍스트 1 : "The sky blue"

입력 텍스트 2 : "The sun **bright**"

입력 텍스트 3 : "The sun sky **bright**"

$$TF = \frac{\text{문서내 단어수}}{\text{문서 내 모든 단어 수}}$$

$$IDF = \log\left(\frac{\text{전체 문서 수}}{\text{단어를 포함한 문서 수}}\right)$$

TF(Term frequency)

입력 텍스트 1

$$\frac{0}{3} = 0$$

입력 텍스트 2

$$\frac{1}{3} = 0.3333$$

입력 텍스트 3



$$\frac{1}{4} = 0.25$$

IDF(inverse document frequency)

$$IDF = \log_2\left(\frac{3}{2}\right) = 0.5849$$

$$TF-IDF \Rightarrow 0.33333 * 0.5849 = 0.1949$$

$$TF-IDF \Rightarrow 0.25 * 0.5849 = 0.1462$$

Terms					
Docs	blue	sky	the	bright	sun
D1	0.5283208		0	0.0000000	
D2	0.0000000		0	0.1949875	
D3	0.0000000		0	0.1462406	

## 2-9 TF-IDF – 영문서(bright)

(실습 1) 불용어 처리된 입력 문서는 다음과 같다. sky에 대한 단어의 TF-IDF를 구해보자.(밑수가 2인  $\log_2$ 를 사용해서 구해보자.)

### 불용어 처리 후, 입력 문서

입력 텍스트 1 : "The **sky** blue"

입력 텍스트 2 : "The sun bright"

입력 텍스트 3 : "The sun **sky** bright"

## 2-10 TF, DF, IDF

- ▶ TF(단어빈도) 한 문서내에서 term이 몇 번 나왔는가?
- ▶ DF(문서빈도, document frequency)  
특정 단어가 포함된 문서는 몇 개인가?
- ▶ Inverse Document Frequency = Inverse(역수) => 시간에 따라 변경 발전됨.

●  $idf = 1/df$  => 처음의 개념에서 점점 발전

●  $idf(t, D) = \log\left(\frac{\text{전체 문서의 수}(D)}{\text{특정단어}(t)\text{포함문서 수}}\right)$

●  $idf(t, D) = \log\left(\frac{\text{전체 문서의 수}}{\text{특정단어}(t)\text{포함문서 수} + 1}\right)$

D는 전체 문서의 수  
t는 특정 단어

## 2-10 코사인 유사도(Cosine Similarity)

- ▶ TF(단어빈도) \* IDF(문서빈도의 역수)

$$\text{tf-idf}(\text{문서}, \text{단어}) = \text{tf}(\text{문서}, \text{단어}(t)) * \text{idf}(\text{단어}(t))$$

- ▶ 문서 내에 단어들에 각각 부여한 중요도를 나타내는 숫자값.
- ▶ 어디에 사용되는가?

TFIDF 값을 부여한 후, 코사인 유사도(Cosine Similarity)등을 이용해서 문서들의 유사도를 구하는데 흔히 사용됨.

## 2-10 TF-IDF의 정리

- ▶ TF(단어빈도) : 특정한 단어가 문서 내에 얼마나 등장할까?
- ▶ DF(문서빈도) : 단어 자체가 문서들에서 얼마나 자주 사용될까?
- ▶ 어디에 사용되는가?

TFIDF 값을 부여한 후, 코사인 유사도(Cosine Similarity)등을 이용해서 문서들의 유사도를 구하는데 흔히 사용됨.

## 3-1 카테고리 키워드 개념

문서 분류의 품질 좌우 중요 요소는 무엇일까?

> 각 카테고리의 의미를 잘 표현하는 **키워드 집합을 선정하고 정의**

이를 위해 어떻게 실제로 하고 있나?

> 도메인 전문가들이 카테고리 키워드를 정의하는 방법

> 문헌 등을 참고하여 정의하는 방법

## 3-2 텍스트 분석 기반 카테고리 구축 절차

단어 사전 구축하기



[그림 3-1] 텍스트 분석 기반 카테고리 구축 예시





## 3-2 텍스트 분석 사전 구축

### 교재 학습 내용

- ▶ 주제어와의 유사 표현 및 동의어 처리를 위한 유의어 사전 규칙 작성할 수 있다.
- ▶ 긍정 및 부정 판단을 위한 감성 분석 사전을 구축할 수 있다.
- ▶ 추출된 텍스트 정보를 데이터베이스화하거나 저장 가능한 파일 형태로 가공할 수 있다.

# 3-3 유의어 사전

단어 사전 구축하기

## ▶ 유의어 사전

### (1) 낱말의 유의어 사전

<표 3-1> 어휘 “가격(價格)”에 대한 낱말 유의어 사전 예시

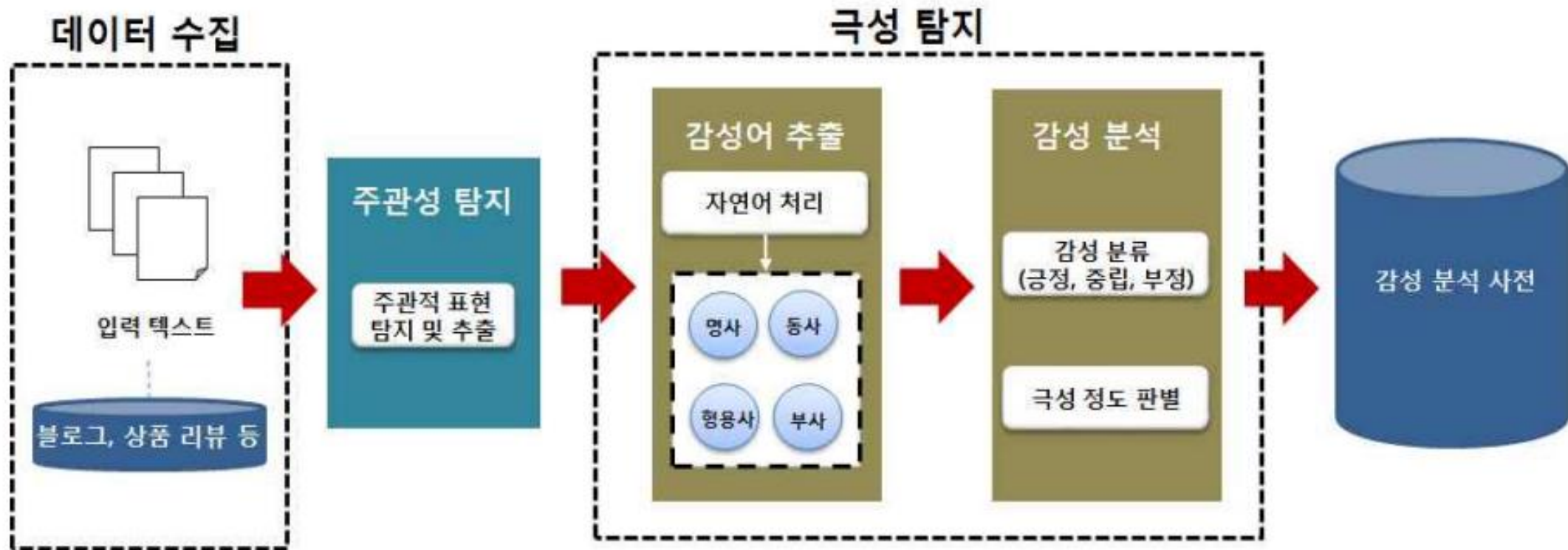
어휘	1차 유의어	2차 유의어
가격(價格)	값어치	값, 가치(價値), 진가(眞價)
	요금(料金)	사용료(使用料), 값, 대금(代金)
	금액(金額)	값, 액수(額數)
	...	...

# 3-4 감성 분석 사전 구축

단어 사전 구축하기

## ▶ 감성 분석 사전

(1) SNS 상의 분석을 통해 마케팅에 사용되고 있다.



[그림 3-2] 감성 분석 사전 구축 절차