

# 자연어처리 기본

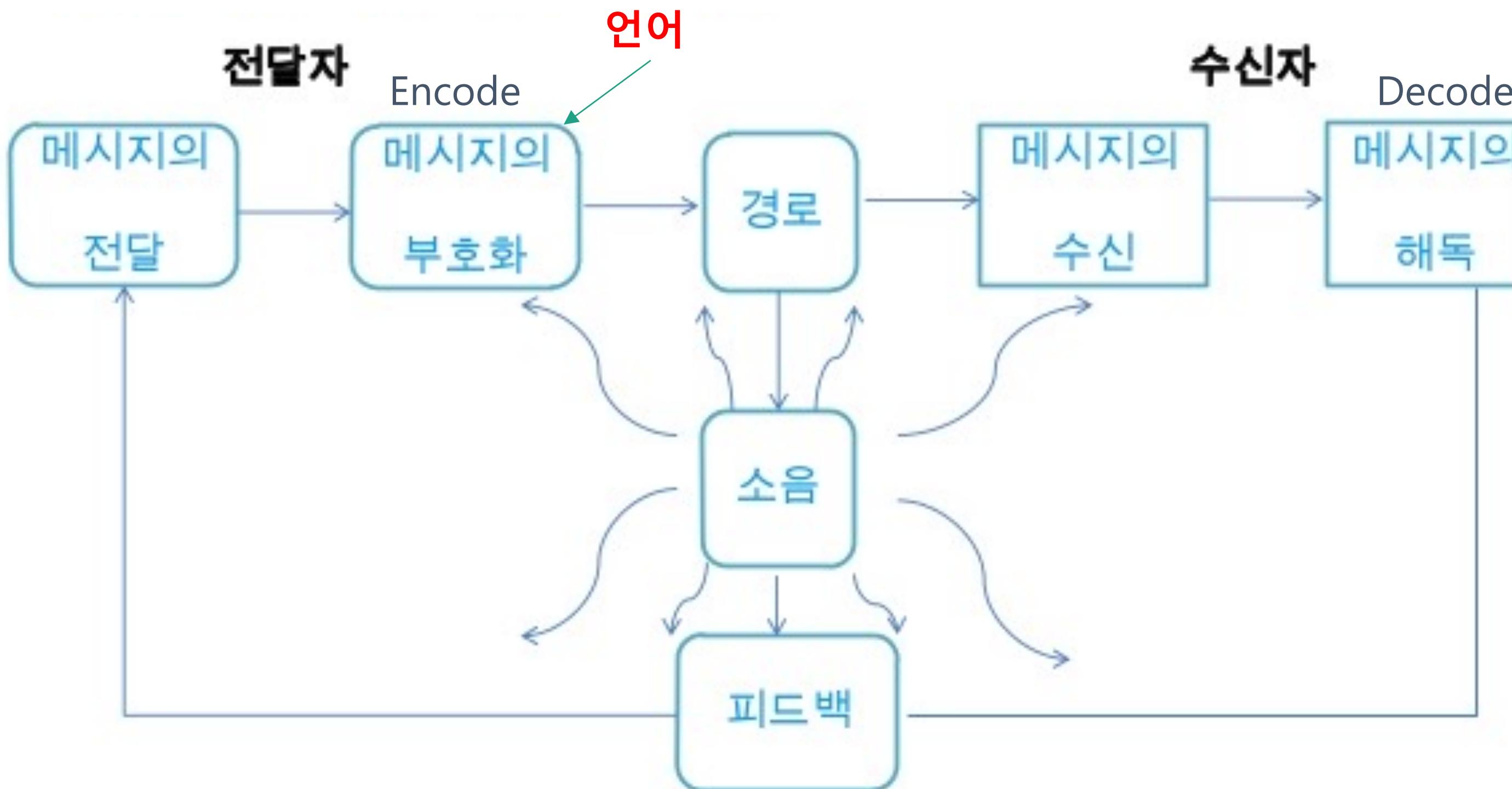
자연어처리란 무엇인가?

# 자연어란?

## 자연어 처리 (Natural Language Processing, NLP)

언어<sup>1</sup> 言語 ⏪ ★ +

명사 생각, 느낌 따위를 나타내거나 전달하는 데에 쓰는 음성, 문자 따위의 수단. 또는 그 음성이나 문자 따위의 사회 관습적인 체계.



# 자연어란?

자연어 처리 (Natural Language Processing, NLP)

자연 언어 自然言語 +

언어 일반 사회에서 자연히 발생하여 쓰이는 언어.



자연언어 : 한국어, 영어, 일본어



인공언어 : 프로그래밍 언어, 에스페란토어

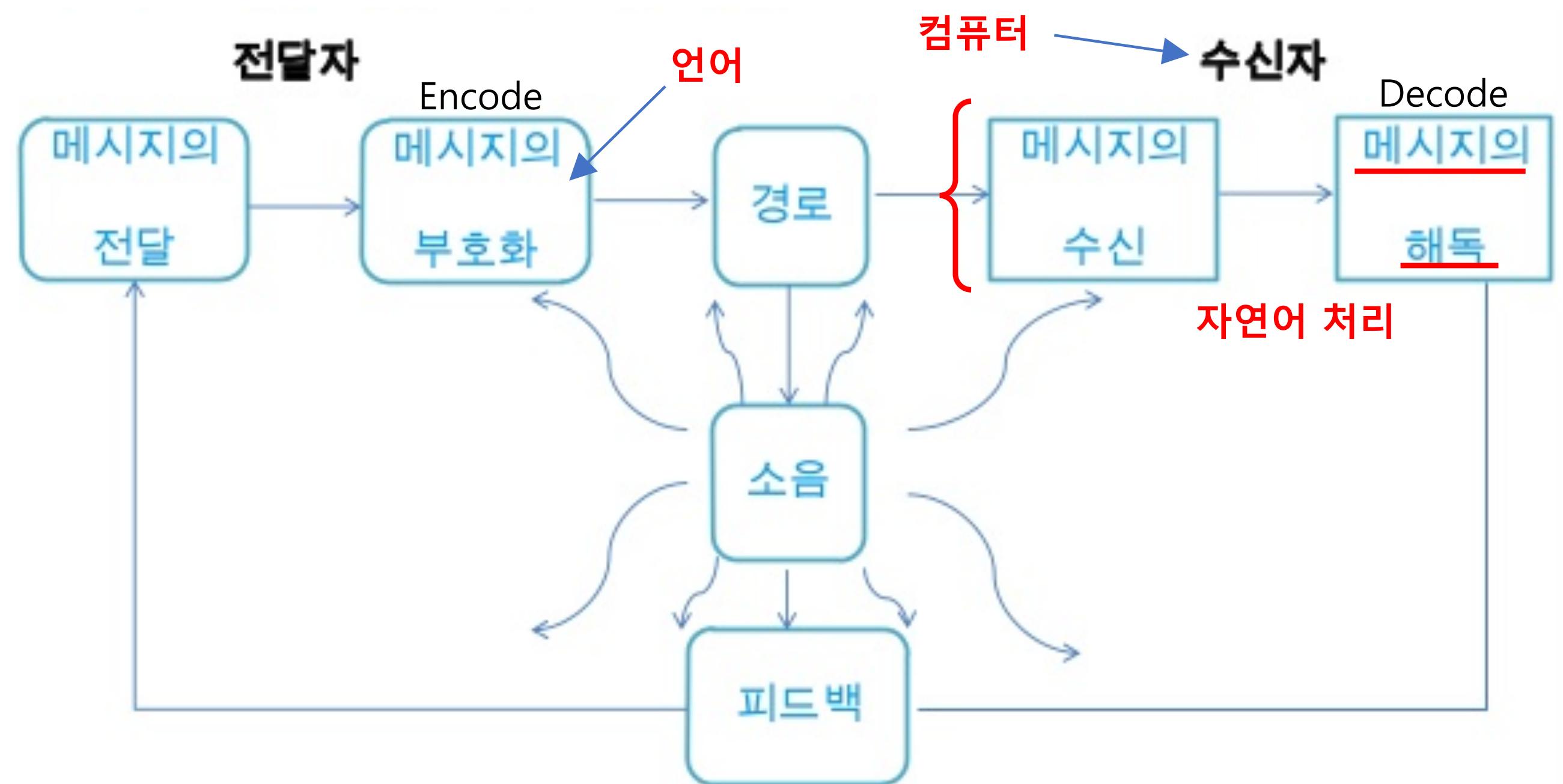
\* Chisung Song, 시나브로 배우는 자연어처리 바벨피쉬 송치성  
<https://www.slideshare.net/shuraba1/ss-56479835>

# 자연어처리란?

자연어 처리 (Natural Language Processing, NLP)

자연어 처리 自然語處理 +

정보·통신 컴퓨터를 이용하여 인간 언어의 이해, 생성 및 분석을 다루는 인공 지능 기술.



# 다양한 자연어 처리 기술

자연어 처리 (Natural Language Processing, NLP)

- 자연어 처리란, '자연어를 컴퓨터가 해독하고 그 의미를 이해하는 기술'

## Symbolic approach

- 규칙/지식 기반 접근법

```
@Override
boolean isAnswerableQuestion(String messagePattern) {
    boolean isAnswerable = false;

    if (messagePattern.matches("ChannelNm(NOW)?(PROGRAM)?WHAT")) {
        // 국회TV에서 지금 뭐해?
        isAnswerable = true;
    } else if (messagePattern.matches("ChannelNm(NOW)?WHATPROGRAM")) {
        // 국회TV에서 지금 무슨 방송해?
        isAnswerable = true;
    } else if (messagePattern.matches("ChannelNm(NOW)?PROGRAMHOW")) {
        // 국회TV에서 지금 무슨 방송해?
        isAnswerable = true;
    }
    return isAnswerable;
}
```

100 원	$100 \text{ (Number)} + \text{원}(G\_ExchangeRateKRW : KRW=원)$
100 달러	$100(\text{Number}) + \text{달러}(G\_ExchangeRateKRW : USD=\text{달러}),$ $S\_UNIT\_USD\_money$
100 m	$100(\text{Number}) + m (S\_UNIT\_m\_length)$
100 미터	$100(\text{Number}) + m (S\_UNIT\_m\_length)$

## Statistical approach

- 확률/통계 기반 접근법
- TF-IDF를 이용한 키워드 추출
- TF (Term frequency): 단어가 문서에 등장한 개수  
-> TF가 높을수록 중요한 단어
- DF (Document frequency): 해당 단어가 등장한 문서의 개수  
-> DF가 높을수록 중요하지 않은 단어

$$w_{x,y} = tf_{x,y} \times \log \left( \frac{N}{df_x} \right)$$

**TF-IDF**  
Term  $x$  within document  $y$

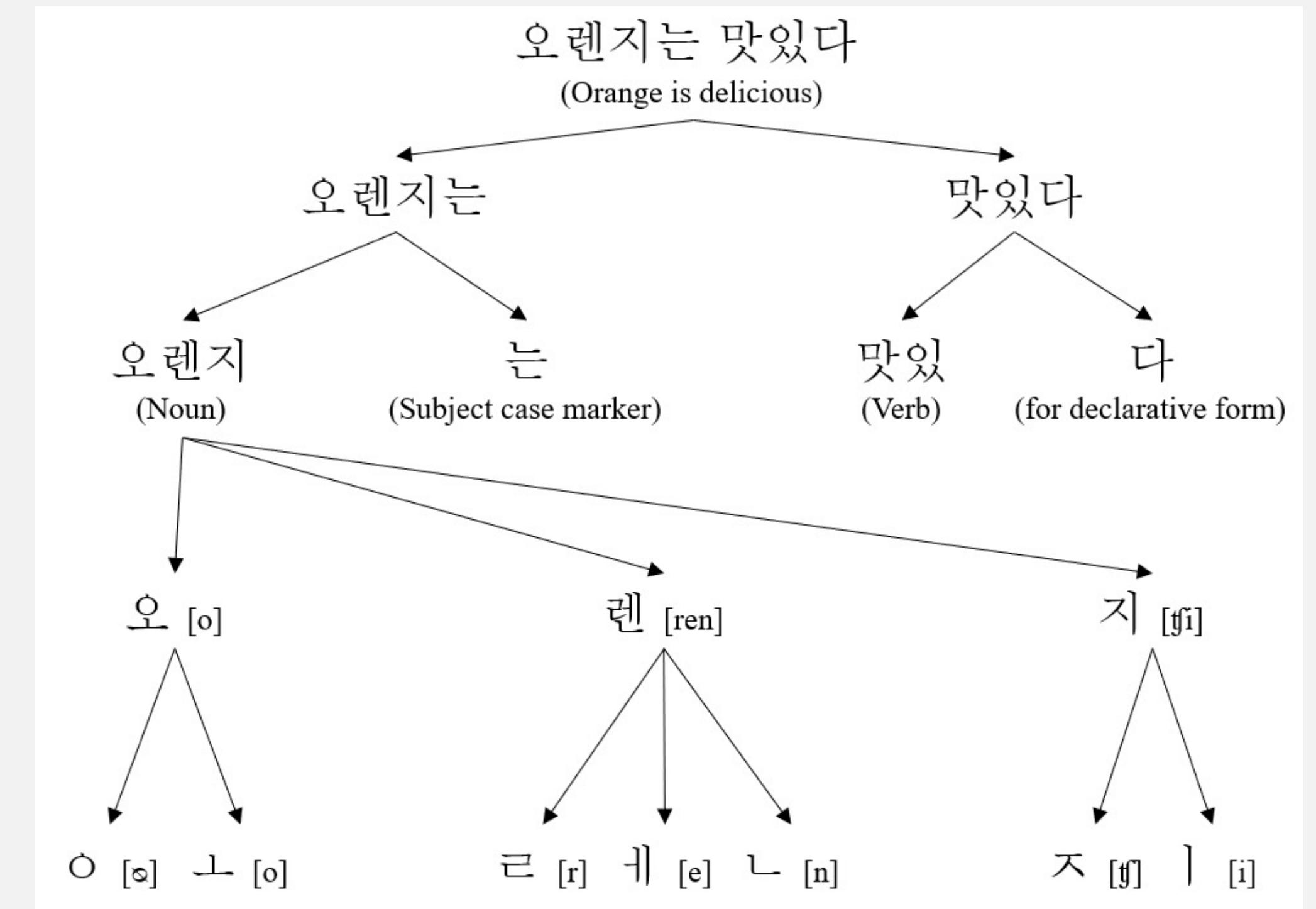
$tf_{x,y}$  = frequency of  $x$  in  $y$   
 $df_x$  = number of documents containing  $x$   
 $N$  = total number of documents

\* sujin oh, 실생활에서 접하는 빅데이터 알고리즘  
[https://www.slideshare.net/osujin121/ss-44186451?from\\_action=save](https://www.slideshare.net/osujin121/ss-44186451?from_action=save)

# 자연어 처리의 단계

자연어 처리 (Natural Language Processing, NLP)

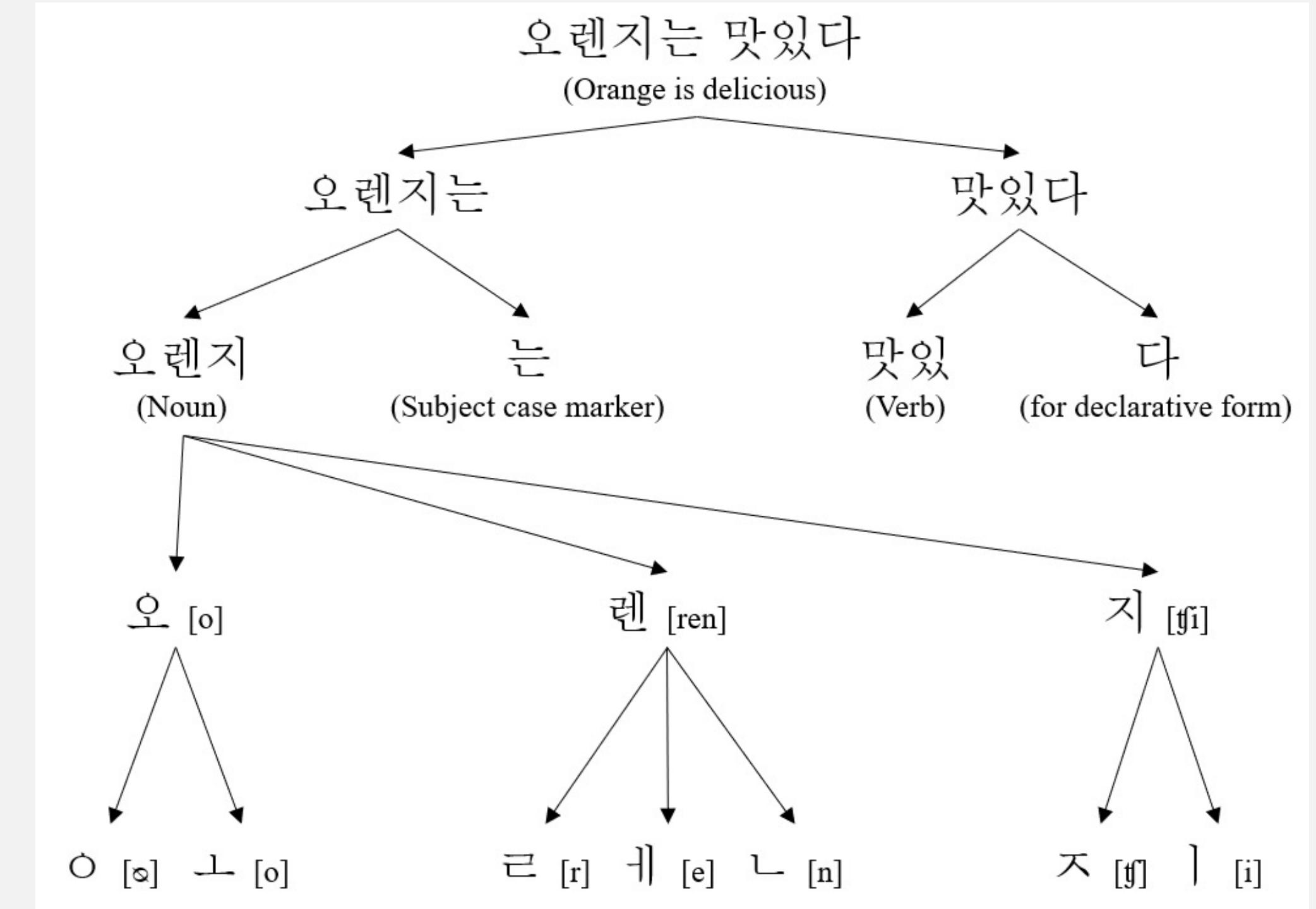
- 전처리
  - 개행문자 제거
  - 특수문자 제거
  - 공백 제거
  - 중복 표현 제거 (ㅋㅋㅋㅋㅋ, ㅠㅠㅠㅠ, ...)
  - 이메일, 링크 제거
  - 제목 제거
  - 불용어 (의미가 없는 용어) 제거
  - 조사 제거
  - 띄어쓰기, 문장분리 보정
  - 사전 구축
- Tokenizing
- Lexical analysis
- Syntactic analysis
- Semantic analysis



# 자연어 처리의 단계

## 자연어 처리 (Natural Language Processing, NLP)

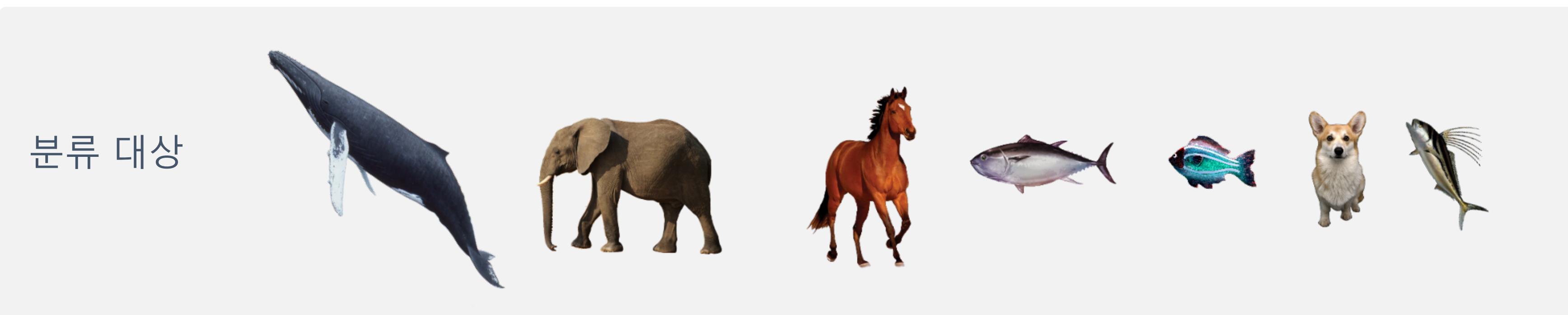
- 전처리
- Tokenizing
  - 자연어를 어떤 단위로 살펴볼 것인가
  - 어절 tokenizing
  - 형태소 tokenizing
  - $n$ -gram tokenizing
  - WordPiece tokenizing
- Lexical analysis
  - 어휘 분석
  - 형태소 분석
  - 개체명 인식
  - 상호 참조
- Syntactic analysis
  - 구문 분석
- Semantic analysis
  - 의미 분석



# 특징 추출과 분류

자연어 처리 (Natural Language Processing, NLP)

- '분류'를 위해선 데이터를 수학적으로 표현
- 먼저, 분류 대상의 특징 (Feature)을 파악 (Feature extraction)



분류 대상의 특징

크기가 다양  
다리의 개수가 다양



# 특징 추출과 분류

자연어 처리 (Natural Language Processing, NLP)

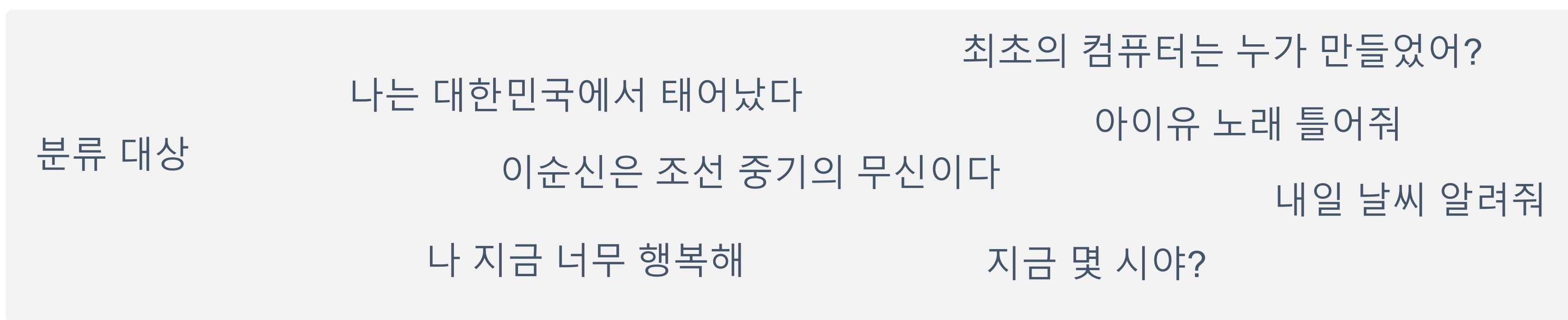
- 분류 대상의 특징(feature)을 기준으로, 분류 대상을 **그래프** 위에 표현 가능
- 분류 대상들의 경계를 수학적으로 나눌 수 있음 (Classification)
- 새로운 데이터도 특징을 기준으로 그래프에 표현하면, 어떤 그룹과 유사한지 파악 가능



# 자연어에서의 특징 추출과 분류

자연어 처리 (Natural Language Processing, NLP)

- 과거에는 사람이 직접 특징(feature)을 파악해서 분류
- 실제 복잡한 문제들에서 분류 대상의 특징을 사람이 파악하기 어려울 수 있다
- 이러한 특징을 컴퓨터가 스스로 찾고(Feature extraction), 스스로 분류(Classification)하는 것이 ‘기계학습’의 핵심 기술

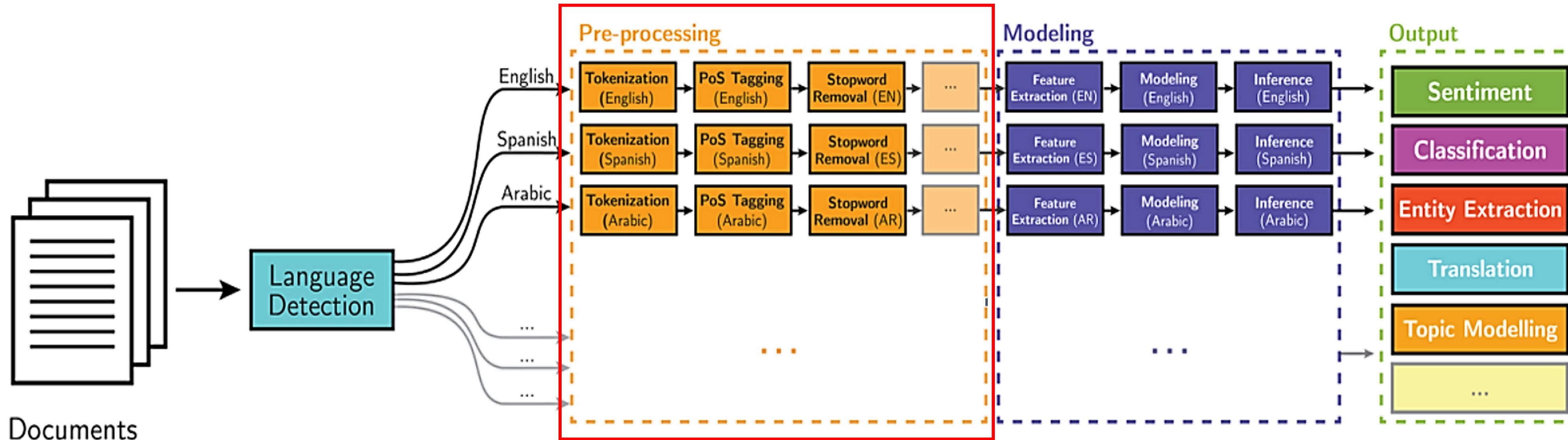


분류 대상의 특징



# 자연어에서의 특징 추출과 분류

자연어 처리 (Natural Language Processing, NLP)



**Feature engineering**

# 다양한 자연어 처리 Applications

자연어 처리 (Natural Language Processing, NLP)

- 문서 분류
- 문법, 오타 교정
- 정보 추출
- 음성 인식결과 보정
- 음성 합성 텍스트 보정
- 정보 검색
- 요약문 생성
- 기계 번역
- 질의 응답
- 기계 독해
- 챗봇
- 형태소 분석
- 개체명 분석
- 구문 분석
- 감성 분석
- 관계 추출
- 의도 파악

# 다양한 자연어 처리 Applications

자연어 처리 (Natural Language Processing, NLP)

чат봇

+

음성 합성

+

감성 분류

+

개체명 인식

+

추천 시스템

+

기계 독해

+

지식 그래프

+

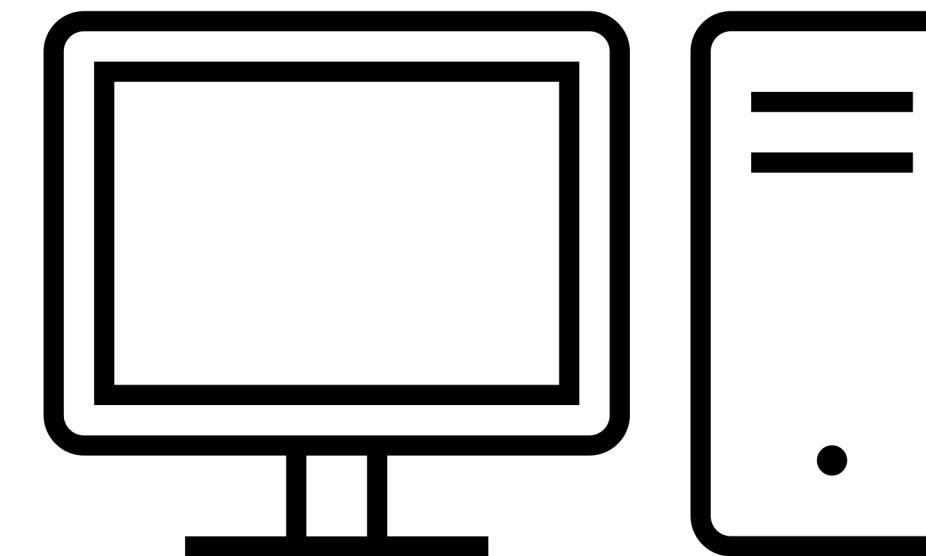
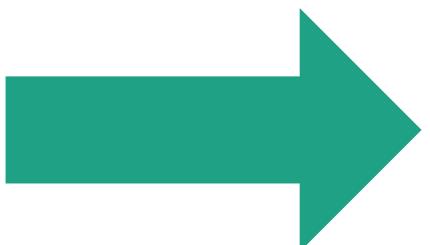
관계 추출

+

플러그인

:

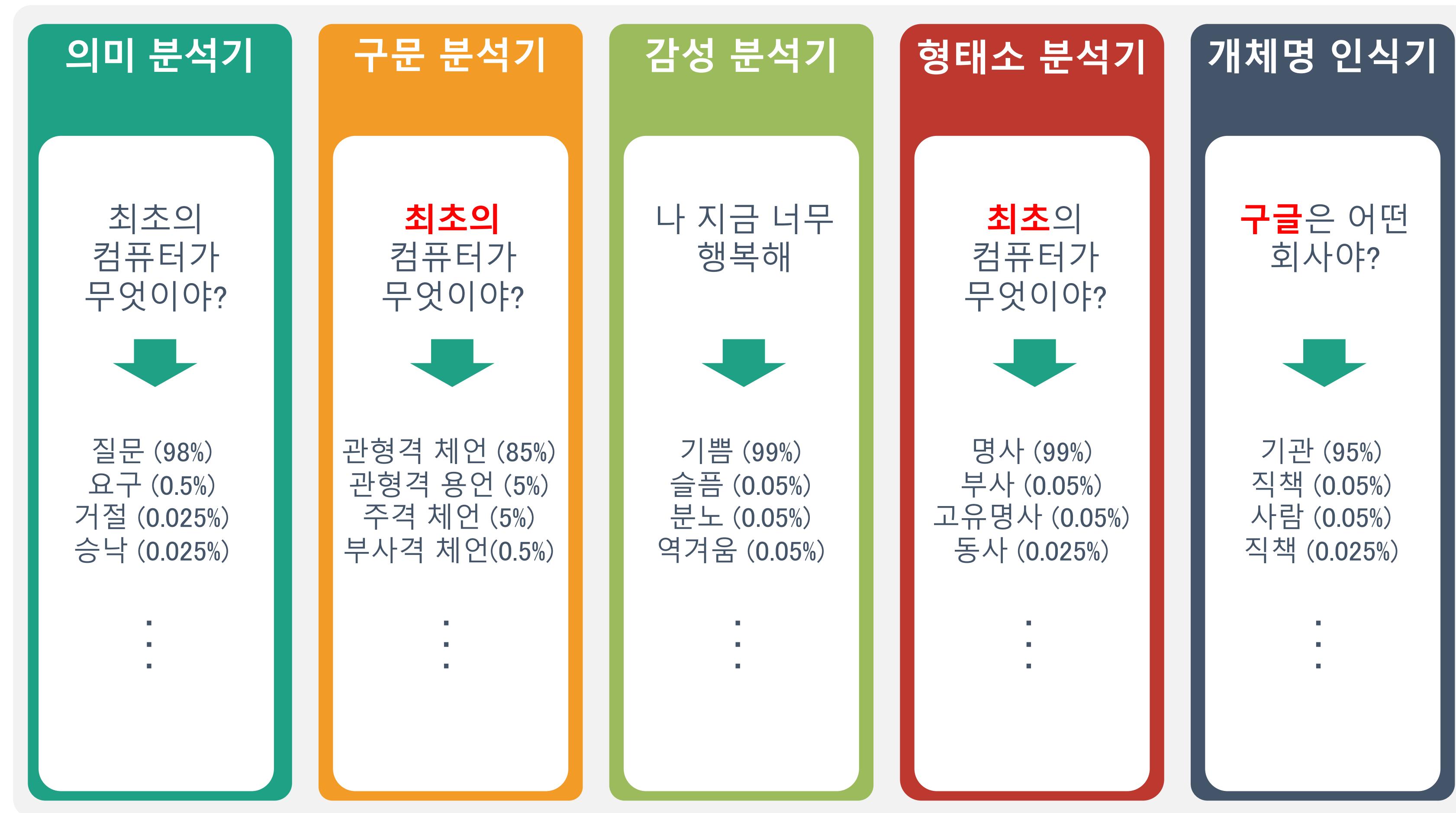
9개 이상의 NLP 기술이 합쳐진 컴비네이션



# 다양한 자연어 처리 Applications

## 자연어 처리 (Natural Language Processing, NLP)

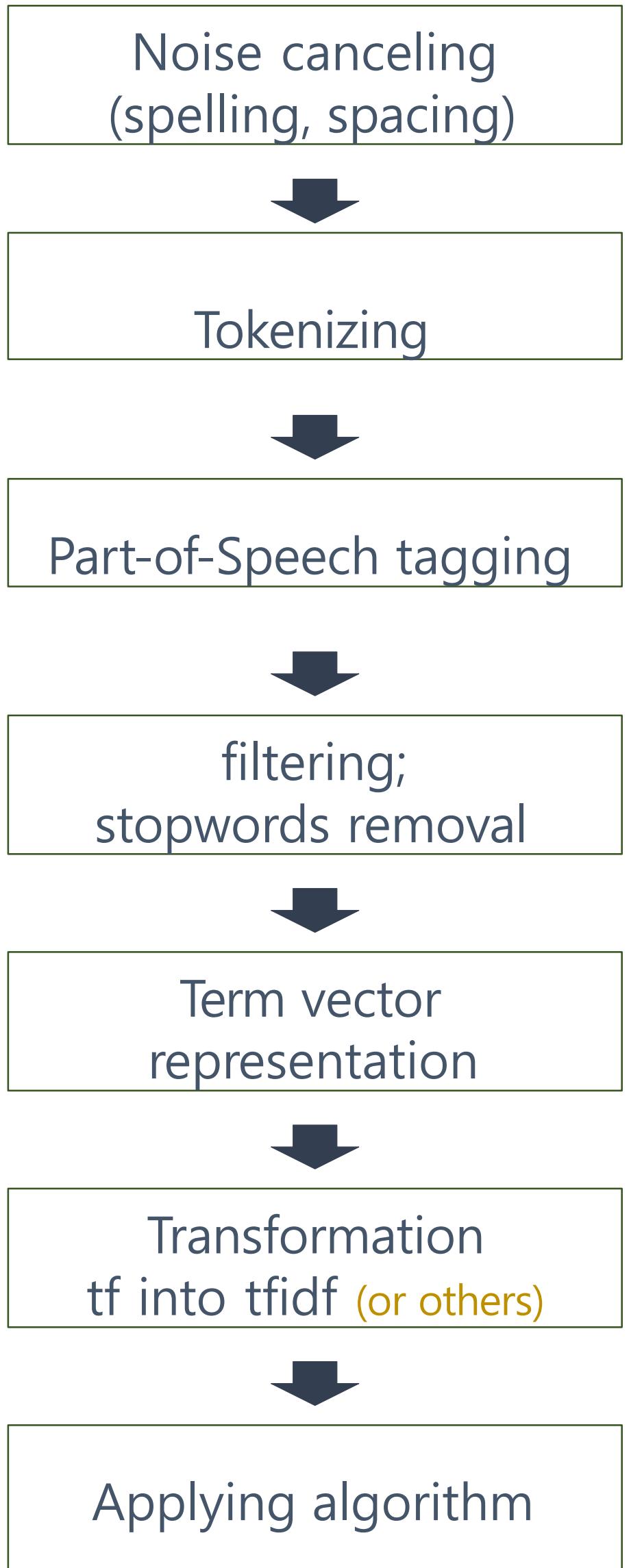
- 형태소 분석, 문서 분류, 개체명 인식 등, 대부분의 자연어 처리 문제는 '분류'의 문제



# Text Processing

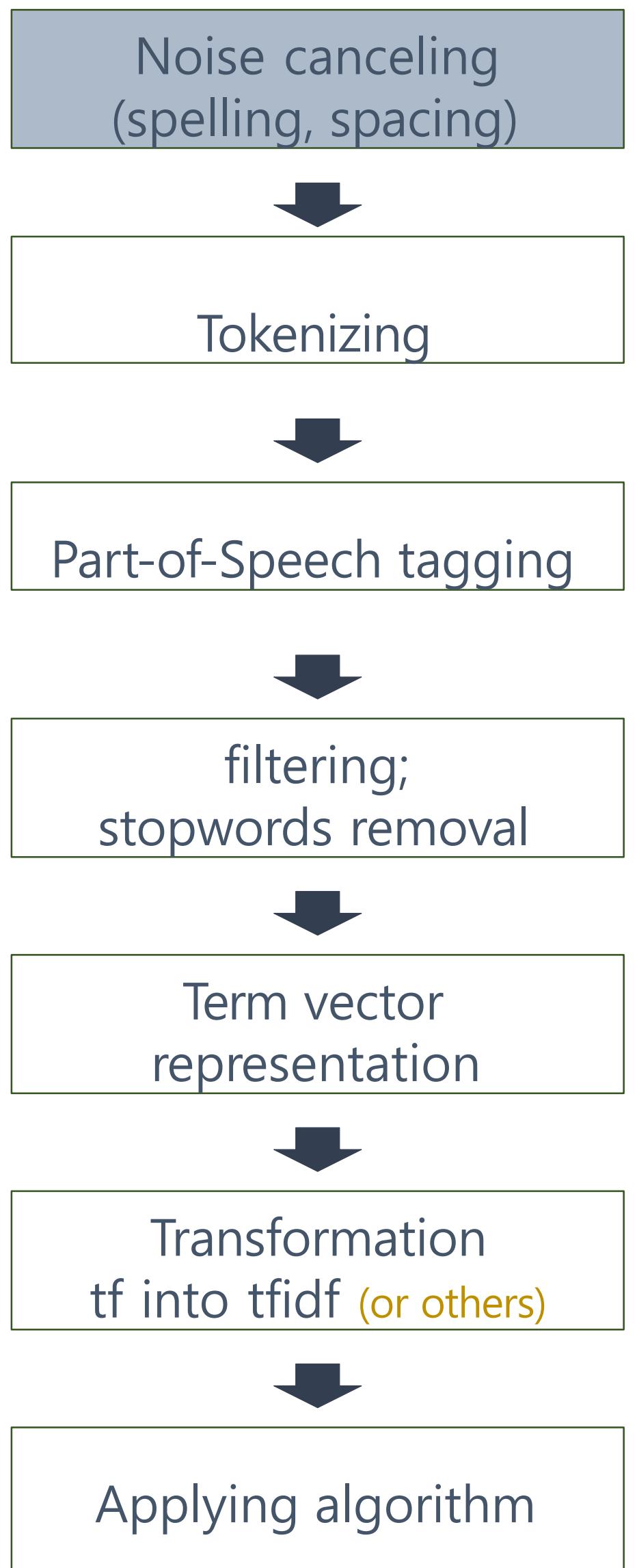
Feat. TF-IDF and some other techniques

# Framework



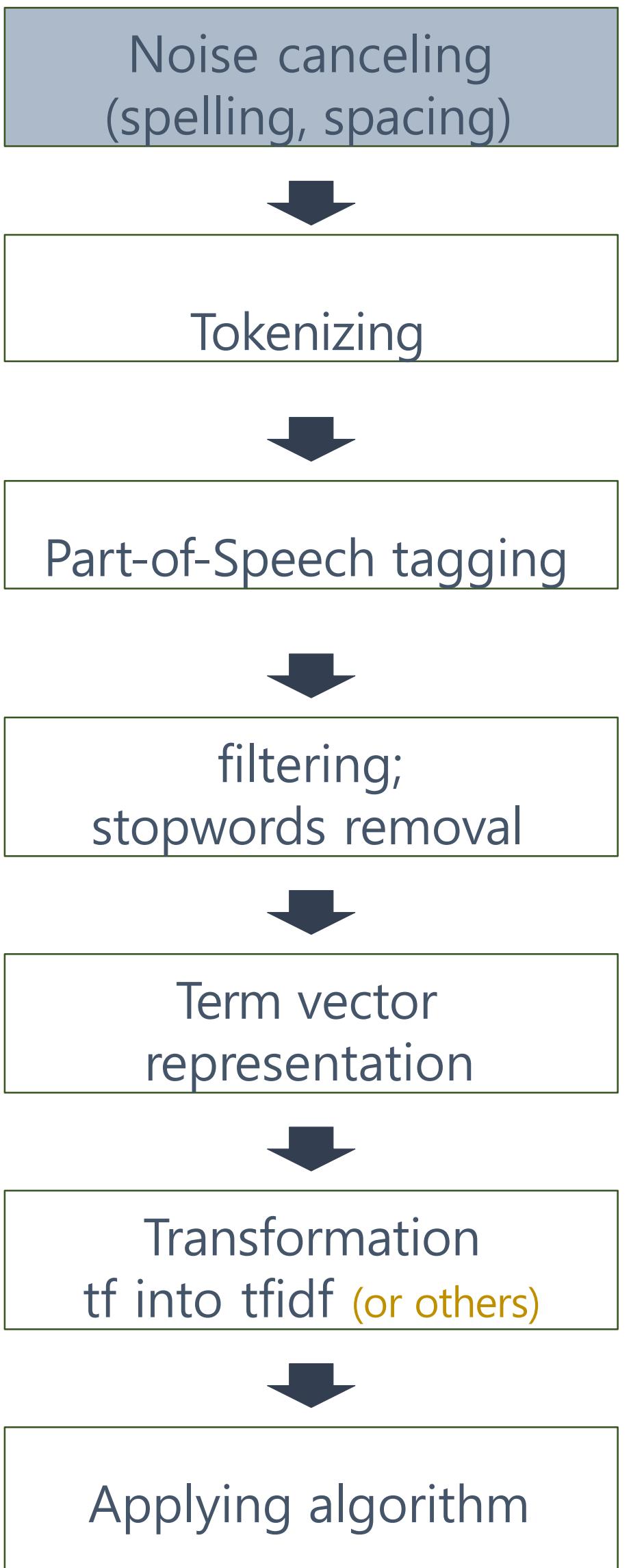
# Noise Canceling - Spelling Check

- 사전에 존재하는 올바른 단어로 수정
  - Edit distance(String distance) 비교를 통해 수정
- Python 라이브러리 활용 가능
- 오탈자이외 은어, 비속어 등 올바른 표현으로 고쳐야하는 경우 종종 발생



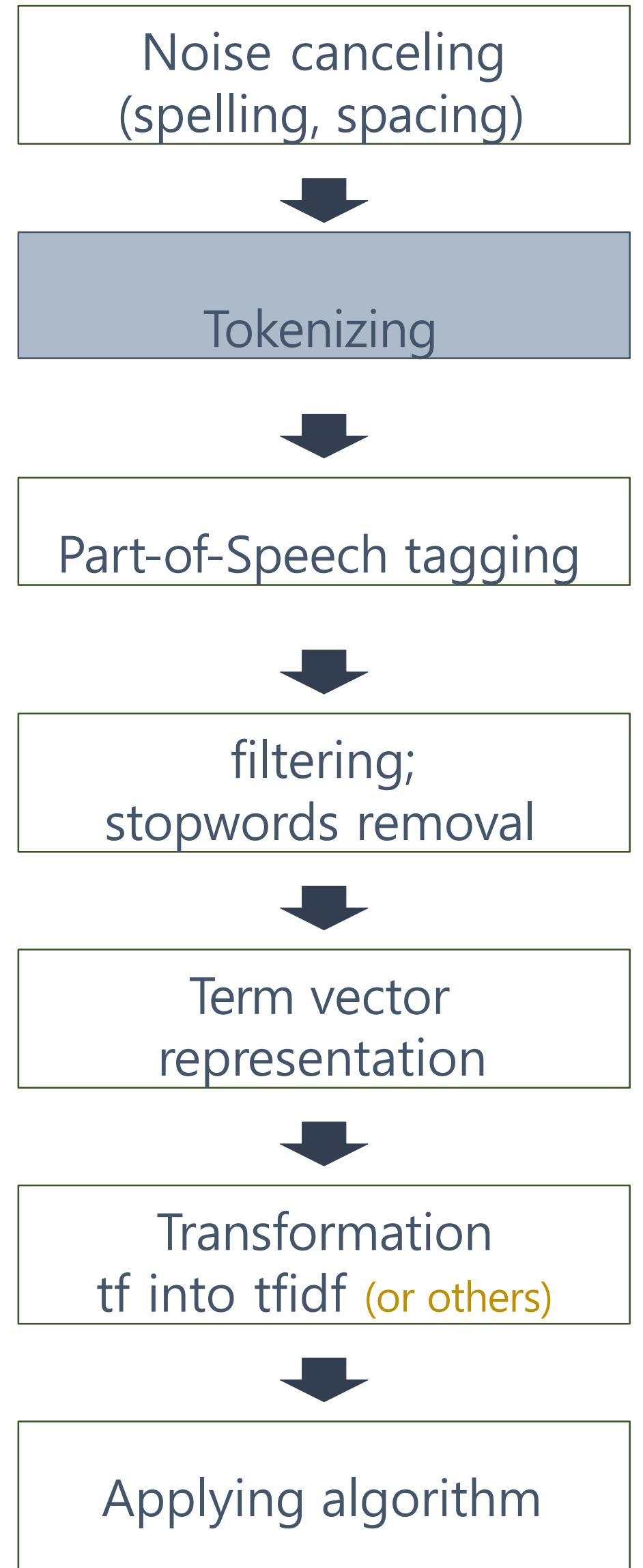
# Noise Canceling - Spacing Check

- 한국어의 어절은 띄어쓰기로 구분
  - 띄어쓰기 오류는 정확도와 학습시간에 영향
  - 띄어쓰기 오류에 대응할 수 있는 토크나이저 혹은 띄어쓰기 오류 교정이 필요



# Tokenizing- 형태소 분석

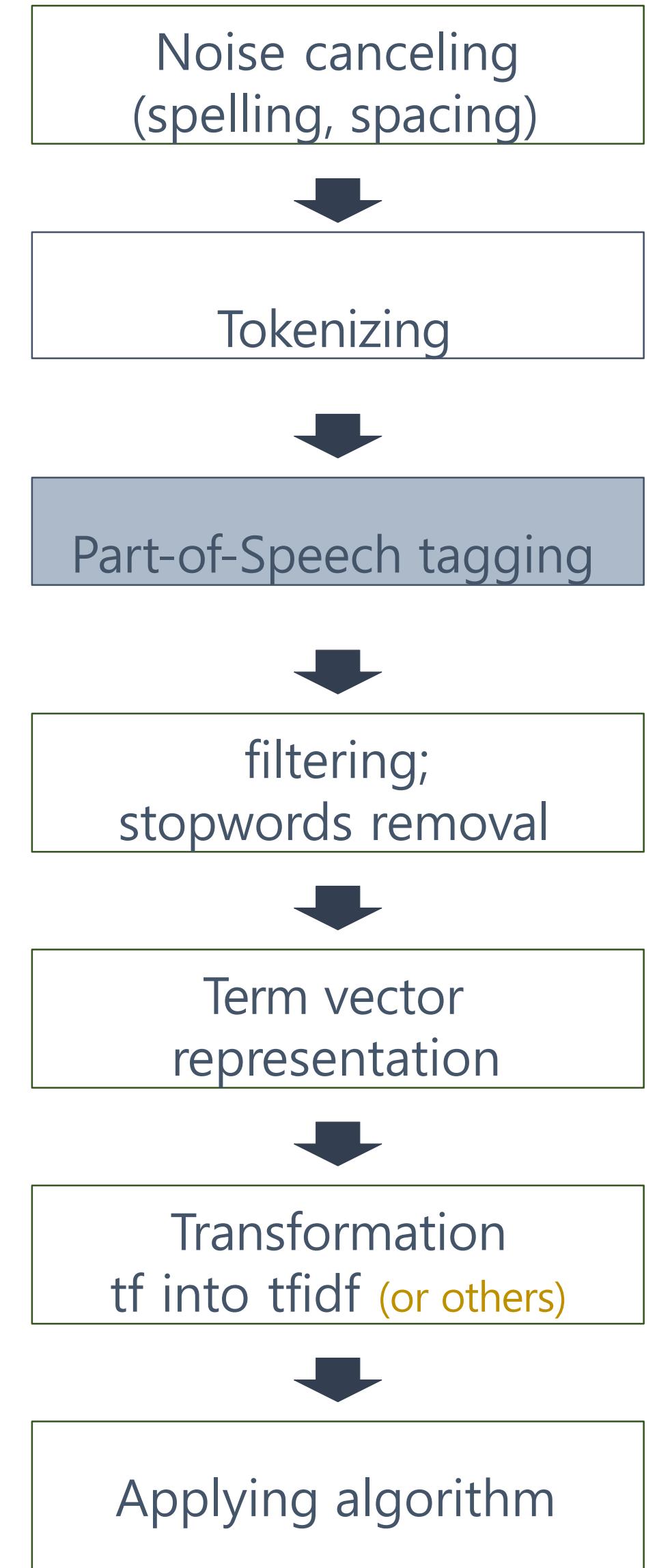
- 토크나이징은 단어를 어절로 나누는 것이다
  - [토크나이징, 은, 단어, 를, 어절, 로, 나누는, 것, 이다]
- 다른 말로 “문장”을 “**토큰**”으로 나누는 것입니다
  - 토큰은 n-gram, 어절, 단어, phrase 등으로 목적에 따라 다르게 정의
  - 토큰을 나누는 다양한 방법이 학습에 영향을 줌



# Part-of-Speech tagging

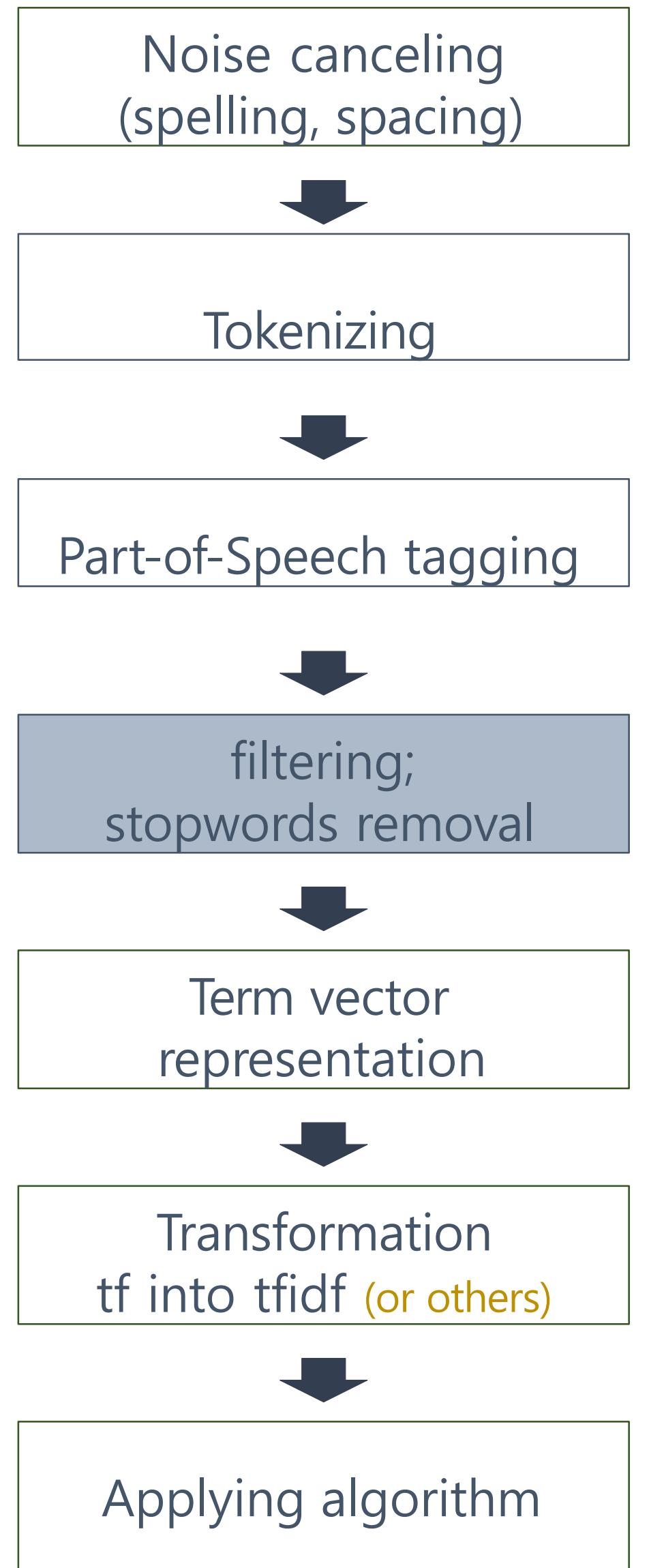
- 품사 판별은 주어진 단어의 품사를 구분합니다

- [토크나이징, 은, 단어, 를, 어절, 로, 나누는, 것, 입니다] →  
 [ (토크나이징, 명사),  
 (은, 조사),  
 (단어, 명사),  
 (를, 조사),  
 (어절, 명사),  
 (로, 조사),  
 (나누는, 동사),  
 (것, 명사),  
 (입니다, 형용사) ]



# Morphological analysis

- 형태소 분석은 단어의 형태소를 인식
  - 형태소는 단어를 구성하는 최소단위
  - 품사 판별: “입니다” → 형용사
  - 형태소 분석: “입니다”
   
→ 이/형용사어근 + ㅂ니다/어미
- 형태소 분석을 바탕으로 단어의 품사를 추정

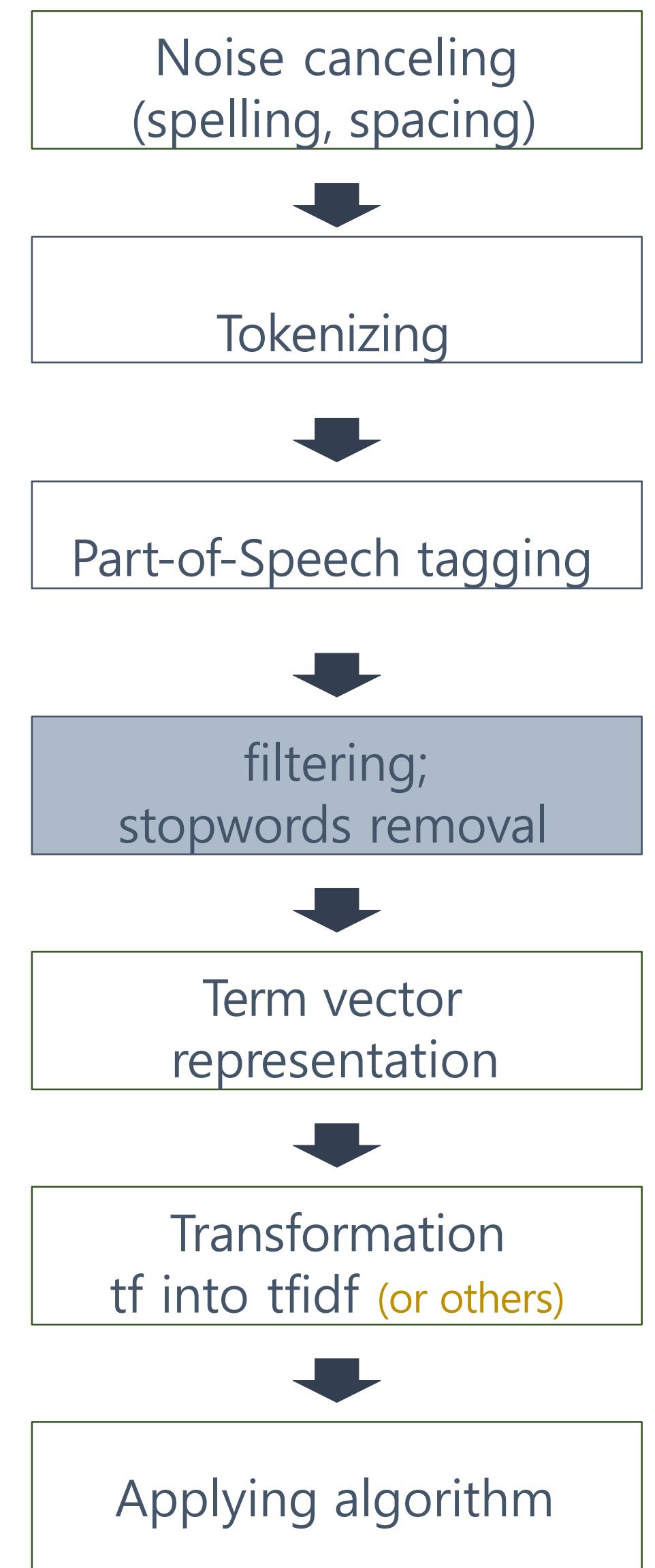


# Stopwords removal

- 문장의 불필요한 단어를 제거

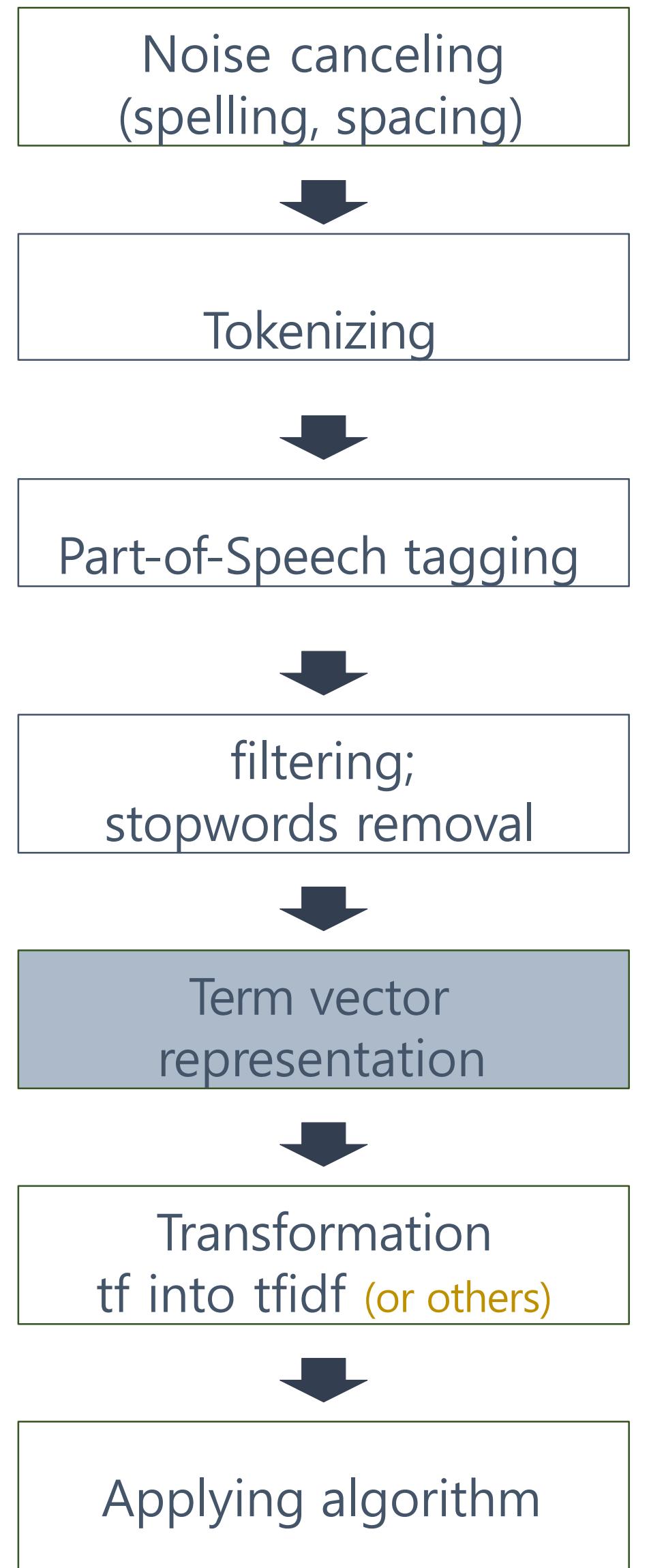
- 거의 등장하지 않는 단어
- -은, -는과 같은 조사(영어에서는 a, the, am, are, ...)
- 키워드 추출을 위한 명사 선택
- 특수문자

"`", ', ◊, `", "`, '`, '`, \", ., ., △, ●, ■, (, ), ,\", >>, `, /, -, ~, =, •<, >, ., ?, !, [, ], ..., ♦, %"



# Term vector representation

- Term weighting
  - $(i, j) = \text{weight}$
  - Term frequency vector 는 문서  $i$ 에서의 단어  $j$ 의 중요도를 단어의 빈도수로 표현
  - $(i, j)$ 의 중요도는 정의하기 나름이며, 반드시 TF 혹은 TF-IDF 를 이용해야 하는 것도 아닙니다



# Term vector representation

- TF-IDF는 Information Retrieval 을 위하여 제안된 term weighting 이다
  - 흔하게 등장하는 단어의 영향력을 최소화할 수 있다

$$\text{TF - IDF}(w, d) = \text{TF}(w) \times \log\left(\frac{N}{\text{DF}(w)}\right)$$

TF: 단어  $w$ 가 문서  $d$ 에서 등장한 빈도 수

DF: 단어  $w$ 가 등장한 문서의 개수

N: 문서집합에서 문서의 총 개수.

- $\text{DF}(w) = N$  이면, 그 단어는 정보력이 없기 때문에  $\text{TF-IDF}(w, d) = 0$

# Term vector representation

- Document frequency (df)가 큰 단어는 정보력이 적습니다
  - 어디에나 등장하는 토큰은 정보력이 없음을 의미
  - 무의미하거나 문법적인 역할(조사)
  - 흔하게 등장하기 때문에 문서 간 거리를 계산할 때 무시

Word	Document frequency
1위	50
야구팀은	500
엘지 트윈스	1000
-은, -는	10,000

# Term vector representation

흔한 단어의 영향력은 낮추고

	올해	LG트윈스	의	는	실력	1위	작년	잠실	두산
Doc 1	5	12	8	15	8	3	3	2	0
Doc 2	1	0	7	8	0	4	1	0	8
Doc 3	2	1	5	7	1	1	0	2	4

문서 집합 전체에서 흔하게 등장하지 않고, 특정 문서에서 자주 나오는 단어의 영향력을 높임

	이번	LG트윈스	의	는	실력	1위	작년	잠실	두산
Doc 1	0.3	2.5	0.2	0.1	3.2	3.6	0.3	0.8	0
Doc 2	0.06	0	0.175	0.57	0	4.8	1	0	3.3
Doc 3	0.12	0.08	0.125	0.48	0.4	1.2	0	0.8	1.65

## article



박태환이 금지 약물 양성반응 통보를 받은 이후에 '도핑 파문'이 일어난 T 병원 김모 원장과 나눈 대화 내용을 녹음해 검찰에 제출한 것으로 알려졌다. 일부 매체는 이에 대해 "박태환이 김 원장에게 '아무 문제가 없는 주사약이라고 해놓고 이게 무슨 일이냐'라고 강하게 따진 것으로 전해졌다 ..."



### 토크나이징/ 품사판별 후

[박태환/N] [이/J] [금지/N] [약물/N] [양성반응/N] [통보/N] [를/J] ...



### Stopword removal

Term	박태환/N	-이/J	금지/N	약물/N	양성반응/N	통보/N	-를/J	...
frequency	28	35	12	15	13	5	32	...

**의미를 지닌 단어 선택:** 명사, 동사, 형용사  
**문법 기능을 하는 단어 배제:** 조사, 어미



### Vector representation

Term	1		55	21	3	27		...
frequency	28		12	15	13	5		...

# Preprocessing & Tokenization

전처리와 형태소 분석

# 전처리와 토크나이징 실습

자연어 처리 (Natural Language Processing, NLP)

## 대소문자의 변환

함수	설명
upper()	모두 대문자로 변환
lower()	모두 소문자로 변환
capitalize()	문자열의 첫 문자를 대문자로 변환
title()	문자열에서 각 단어의 첫 문자를 대문자로 변환
swapcase()	대문자와 소문자를 서로 변환

## 편집, 치환

함수	설명
strip()	좌우 공백을 제거
rstrip()	오른쪽 공백을 제거
lstrip()	왼쪽 공백을 제거
replace(a, b)	a를 b로 치환

# 전처리와 토크나이징 실습

자연어 처리 (Natural Language Processing, NLP)

## 분리, 결합

함수	설명
split()	공백으로 분리
split('₩t')	탭을 기준으로 분리
' '.join(s)	리스트 s에 대하여 각 요소 사이에 공백을 두고 결합
lines.splitlines()	라인 단위로 분리

## 구성 문자열 판별

함수	설명
isdigit()	숫자 여부 판별
isalpha()	영어 알파벳 여부 판별
isalnum()	숫자 혹은 영어 알파벳 여부 판별
islower()	소문자 여부 판별
isupper()	대문자 여부 판별
isspace()	공백 문자 여부 판별
startswith('hi')	문자열이 hi로 시작하는지 여부 파악
endswith('hi')	문자열이 hi로 끝나는지 여부 파악

# 전처리와 토크나이징 실습

자연어 처리 (Natural Language Processing, NLP)

## 검색

함수	설명
count('hi')	문자열에서 hi가 출현한 빈도 리턴
find('hi')	문자열에서 hi가 처음으로 출현한 위치 리턴, 존재하지 않는 경우 -1
find('hi', 3)	문자열의 index에서 3번부터 hi가 출현한 위치 검색
rfind('hi')	문자열에서 오른쪽부터 검사하여 hi가 처음으로 출현한 위치 리턴, 존재하지 않는 경우 -1
index('hi')	find와 비슷한 기능을 하지만 존재하지 않는 경우 예외발생
rindex('hi')	rfind와 비슷한 기능을 하지만 존재하지 않는 경우 예외발생
count('hi')	문자열에서 hi가 출현한 빈도 리턴

# 전처리와 토크나이징 실습

자연어 처리 (Natural Language Processing, NLP)

토큰화  
(Tokenizing)

문장 토큰화  
(Sentence Tokenizing)

단어 토큰화  
(Word Tokenizing)

- 주어진 데이터를 토큰(Token)이라 불리는 단위로 나누는 작업
- 토큰이 되는 기준은 다를 수 있음  
(어절, 단어, 형태소, 음절, 자소 등)

- 문장 분리

- 구두점 분리, 단어 분리  
"Hello, World!" ->  
"Hello", ",", "World", "!"

# 전처리와 토크나이징 실습

자연어 처리 (Natural Language Processing, NLP)

## 문장 토큰화(Sentence Tokenization. 문장 분리)

```
from nltk.tokenize import sent_tokenize
```

```
text = "Hello, world. These are NLP tutorials."  
print(sent_tokenize(text))
```

# 전처리와 토크나이징 실습

자연어 처리 (Natural Language Processing, NLP)

## 단어 토큰화(Word Tokenization. 단어 분리, 구두점 분리)

```
import nltk  
from nltk import WordPunctTokenizer  
  
nltk.download('punkt')  
  
text = "Hello, world. These are NLP tutorials."  
print(WordPunctTokenizer().tokenize(text))
```

# 전처리와 토크나이징 실습

자연어 처리 (Natural Language Processing, NLP)

## 한국어 토큰화

- 영어는 New York과 같은 합성어 처리와 it's와 같은 줄임말 예외처리만 하면, 띄어쓰기를 기준으로도 잘 동작하는 편
- 한국어는 조사나 어미를 붙여서 말을 만드는 교착어로, 띄어쓰기만으로는 부족 예시) he/him -> 그, 그가, 그는, 그를, 그에게
- 한국어에서는 조사, 어미를 분리하는 형태소 분석을 통하여 토큰화를 수행 예시) 안녕하세요 -> 안녕/NNG, 하/XSA, 세/EP, 요/EC

# Vector Representation

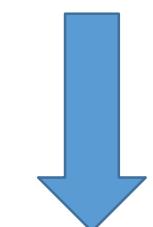
언어를 컴퓨터가 알아듣도록 숫자로 표현하기

# Vector space representation

- One hot representation (Bag of Words model)
  - (row, column) 은 (문서, 단어) 해당하는 값은 단어의 중요도 혹은 빈도수를 의미

	기계	학습	은	텍스트	마이닝	는
Doc 1	3	2	5	0	0	0
Doc 2	0	0	0	3	5	5
...	...	...	...	...	...	...

Doc 1 = [(0, 3), (1, 2), (2, 5)]  
 Doc 2 = [(3, 3), (4, 5), (5, 5)]



	0	1	2	3	4	5
Doc 1	3	2	5	0	0	0
Doc 2	0	0	0	3	5	5
...	...	...	...	...	...	...

# Vector space representation

- One hot representation (Bag of Words model)

	0	1	2	3	4	5
Doc 1	3	2	5	0	0	0
Doc 2	0	0	0	3	5	5
...	...	...	...	...	...	...

- Column 개수  $|V|$ 는 문서 전체에서 등장한 단어 종류로, 매우 큽니다.
- 한 문서에 등장하는 단어의 개수는 적기 때문에, 대부분의 값이 0입니다 (Sparse vector).
- 문서에 등장한 단어를 쉽게 확인할 수 있어 해석이 쉽지만, 모든 단어는 다른 단어로 취급합니다. 단어간 유사성을 표현하기 어렵습니다.

# Vector space representation

- Distributed representation
  - 단어/문서를 Word2Vec 등으로 d차원 공간의 벡터로 표현
  - 단어의 “**의미적 유사성**”을 벡터 공간에 표현
    - 벡터 공간의 거리가 가까운 단어 또는 문서는 의미가 비슷

```
'dog'= [0.31, -0.21, 2.01, 0.58, ... ]
```

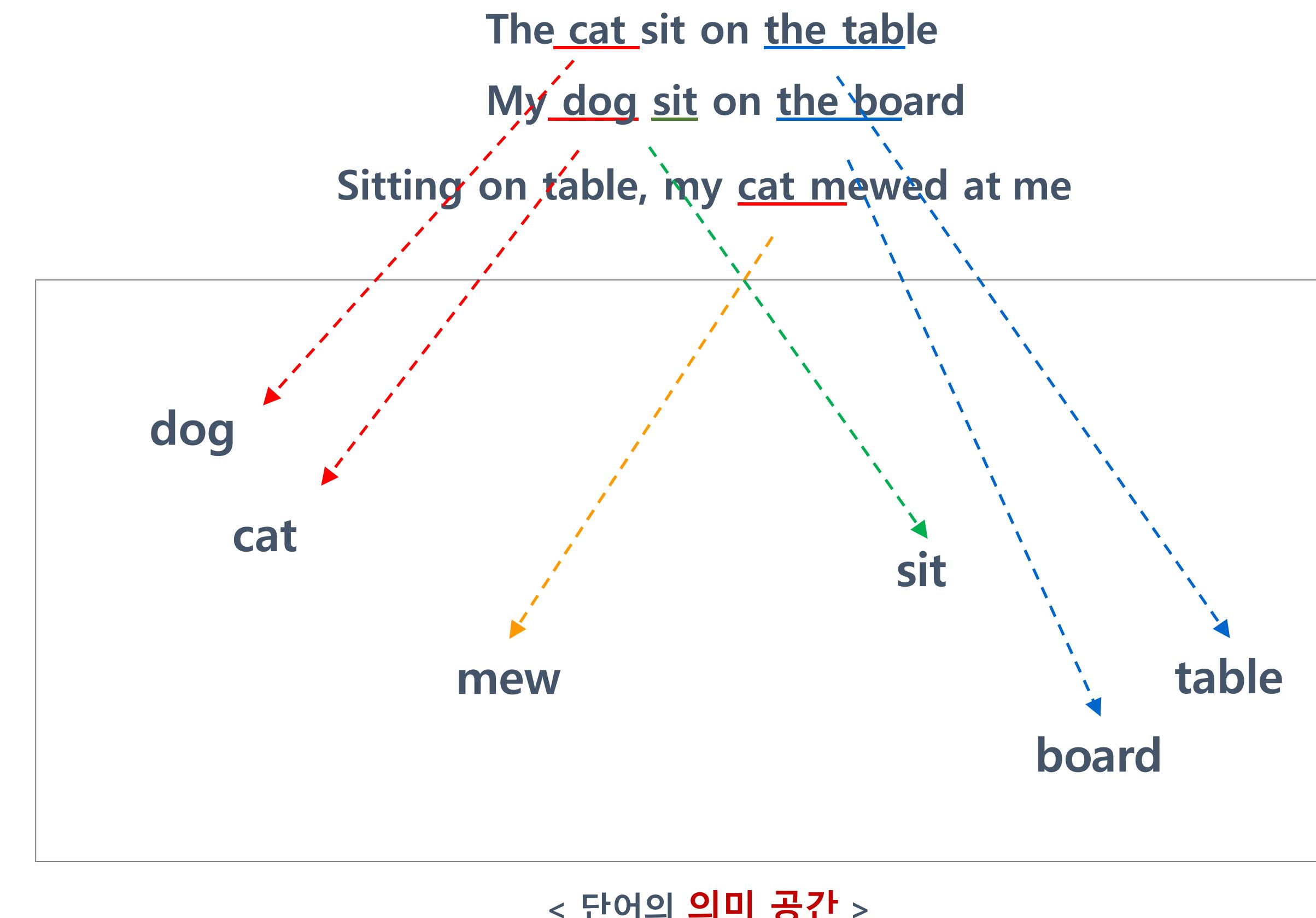
```
'cat'= [0.45, -0.17, 1.79, 0.61, ... ]
```

```
'topic modeling'= [-2.01, 0.03, 0.22, 0.54, ... ]
```

```
'dim. reduction'= [-1.88, 0.11, 0.19, 0.45, ... ]
```

# Vector space representation

- Distributed representation
  - 각 벡터는 “의미 공간”에서 좌표 역할



# String distance

- 단어 또는 문자열 벡터는 다양한 metric 을 이용하여 유사도 측정
  - Euclidean, Cosine, ...
- 단어의 형태간 거리와 단어의 의미간 거리로 비교 분석
  - 의미적으로 비슷한 단어  
Ex. (점심, 저녁) vs (점심, 자동차)
  - 형태적으로 비슷한 단어  
Ex. (서비스, 써비스) vs. (서비스, 자동차)

# String distance

- 오탈자와 정자의 관계는 형태적 거리가 아주 가까운 단어입니다.
  - 두 단어의 형태가 비슷하면서 한 단어의 빈도수가 매우 크다면 희귀한 단어를 오탈자로, 빈번한 단어를 정자로 생각할 수 있습니다.
  - 희귀한 단어를 빈번한 단어로 치환하여 오탈자를 수정합니다.
    - 써비스 → 서비스

# Levenshtein (Edit) distance

- 가장 대표적인 string distance metric입니다. 단어 A에서 B로 수정하기 위한 횟수를 거리로 정의합니다.
- 단어의 수정 방법은 세 가지로 정의됩니다.
  - **Deletion**: '서어비스' → '서비스' (거리 = 1)
  - **Insertion**: '데이터마닝' → '데이터마이닝' (거리 = 1)
  - **Substitution**: '데이터마이닝' → '데이터마이닝' (거리 = 1)

# Levenshtein (Edit) distance

- String metric 중 가장 대표적인 척도입니다. 단어 A에서 B로 수정하기 위한 횟수를 거리로 정의합니다.
- 데이터마닝 → 데이터마이닝  
1단계: 데이터마닝 → 데이터**타**마닝 (누적 거리 = 1)  
2단계: 데이터마닝 → 데이터마**이**닝 (누적 거리 = 2)

## Jaccard distance

- Jaccard distance 는 집합 간의 유사도를 정의하는 방법입니다.
- 각 단어를 글자 (units) 의 집합으로 생각합니다.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

## Jaccard distance

- 각 단어를 글자 (units) 의 집합으로 생각합니다.

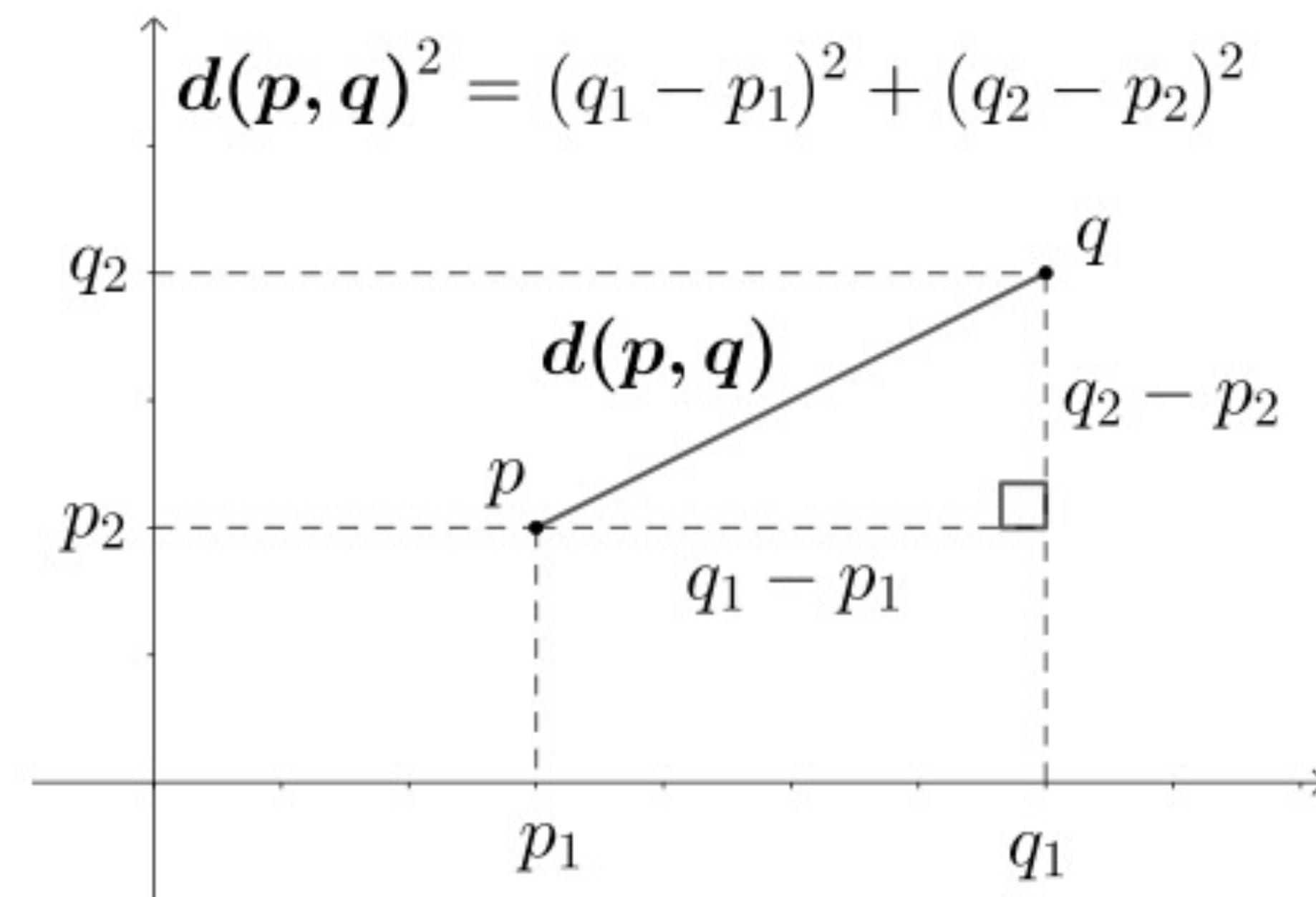
$$J(\text{'데이터마이닝'}, \text{'데이터마닝'}) = \frac{|\{\text{데}, \text{이}, \text{마}, \text{닝}\}|}{|\{\text{데}, \text{이}, \text{터}, \text{타}, \text{마}, \text{닝}\}|} = \frac{4}{6}$$

- Similarity 를 distance 로 변환합니다.  $J_D(A, B) = 1 - J(A, B)$
- 한 글자(unit)의 등장 횟수는 고려하지 않습니다.**

# Euclidean Distance

- 두 점 사이에 줄을 긋고, 그 줄의 길이를 계산하는 것!

$$\text{distance}(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2}$$



2D에서 euclidean distance

# Cosine Similarity

- Cosine similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

- $\cos((3, 0, 2), (1, 2, 0)) = \frac{3 \times 1 + 0 \times 2 + 2 \times 0}{\sqrt{3^2 + 2^2} \times \sqrt{1^2 + 2^2}} = \frac{3}{\sqrt{13} \times \sqrt{5}}$

- Vector A 와 Vector B

- 같은 방향(0) = 1
- 완전히 반대 방향(180) = -1
- 서로 독립적(90) = 0

# Document clustering

- 군집화 (Clustering)는 비슷한 데이터를 하나의 집합으로 그룹화합니다.
  - 리뷰가 비슷한 영화들의 군집화 결과

cluster # 1	cluster # 2	cluster # 3
해무	인터스텔라	응답하라 1988
베를린	미스터 고	인턴
내가 살인범이다	다크 나이트	님아, 그 강을 건너지 마오
신세계	영웅: 샐러맨더의 비밀	카트
곡성(哭聲)	인셉션	인사이드 아웃
검은 사제들	트랜스포머 3	형
악마를 보았다	배틀쉽	비긴 어게인
용의자	스카이라인	두근두근 내 인생
감기	2012	라라랜드
감시자들	그래비티	반창꼬

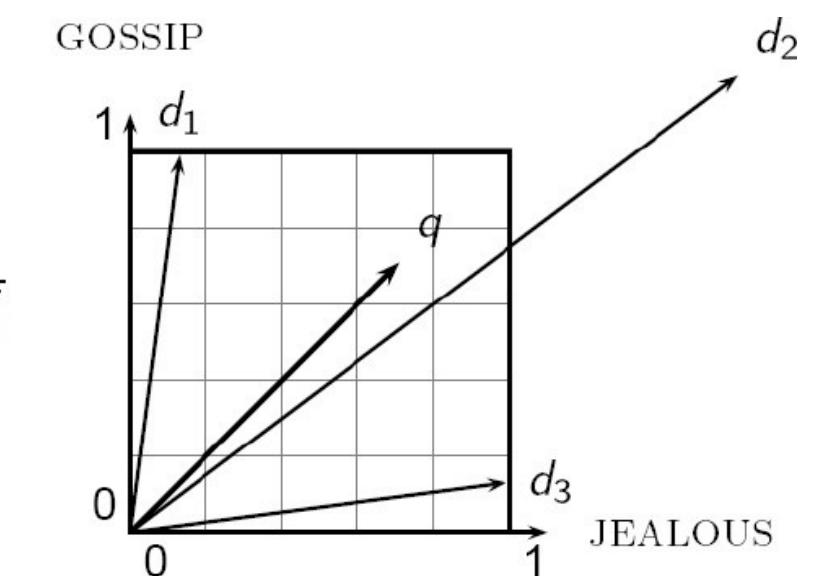
# Document clustering

- Bag of words model 에는 Euclidean 보다 Cosine 이 적절합니다.
  - 두 문서에 공통으로 등장한 단어에 대해서만 유사성을 판단합니다.
  - Cosine은 문서 길이에<sub>(벡터의 크기, norm)</sub> 영향을 받지 않습니다
  - Sparse representation 에서는 벡터의 방향이 가장 중요합니다.

	Term 1	Term 2	Term 3	Term 4	Term 5
Doc 1	1	1	1		
Doc 2			2	1	1
Doc 3	2	2	2		1

Euclidean( $d_1, d_2$ ) = Euclidean( $d_1, d_3$ )이지만,  $d_1$ 과  $d_3$ 이 공통된 단어가 많기 때문에 더 비슷

$$Sim(D_1, D_2) = \frac{\sum_i x_{1i}x_{2i}}{\sqrt{\sum_j x_j^2} \sqrt{\sum_k x_k^2}}$$



# Document clustering

- Document distance/similarity 를 계산할 때에는 Cosine 이 적합합니다.
  - 문서 표현에 distributed representation 을 이용한다 하더라도 벡터의 방향이 가장 중요합니다.
  - Logistic regression, Neural network 등의 머신 러닝 알고리즘도 벡터 방향이 큰 영향을 미칩니다.