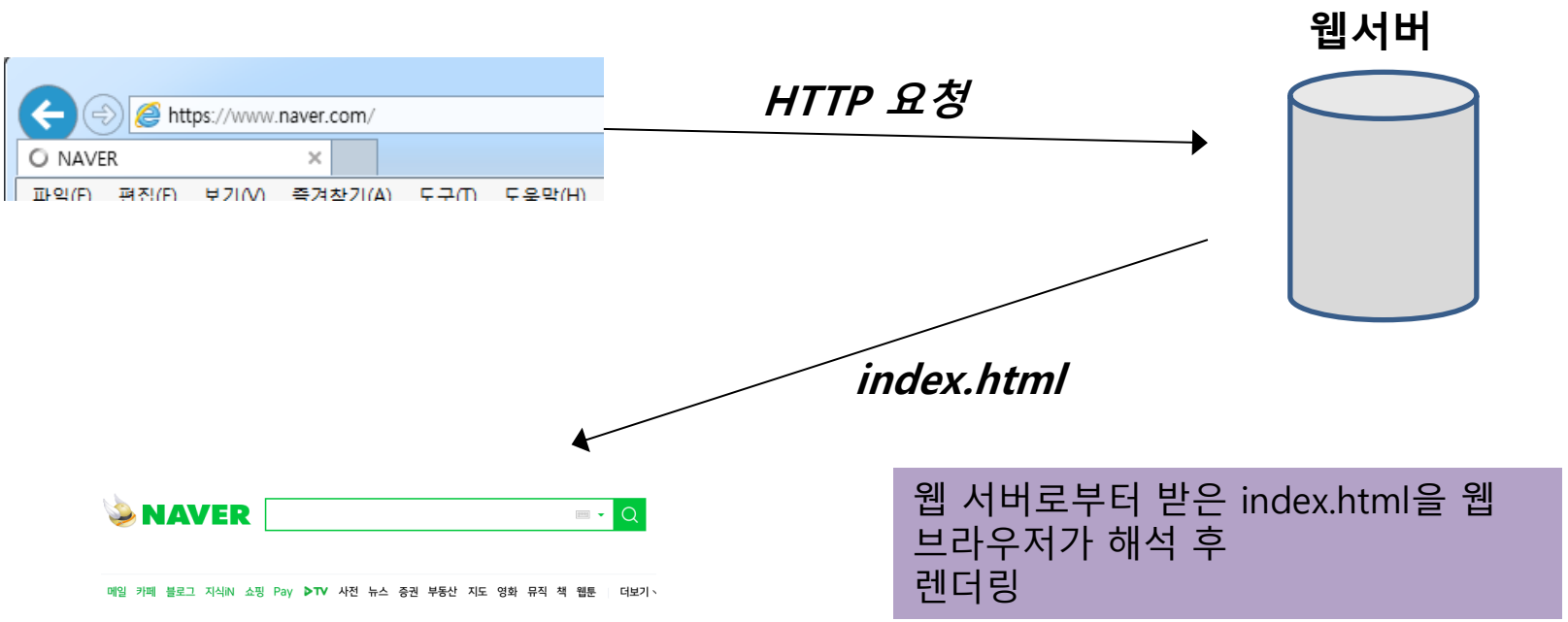
NLP와 Text Mining

2025

1. 웹 수집

- **웹 서버 (Web server)**
  - 인터넷 브라우저 등 클라이언트로부터 HTTP 요청을 받아들이고, HTML 등의 웹 페이지를 반환하는 프로그램



HTML (Hyper Text Markup Language)

- 마크업 언어
  - 문서나 데이터의 구조를 기술
  - 태그(tag)를 이용, 값을 표현

<태그> 데이터 </태그>

HTML 주요 태그

- **HTML 태그: 시작과 끝**
  - <html> 내용 </html>, <head> 내용 </head>

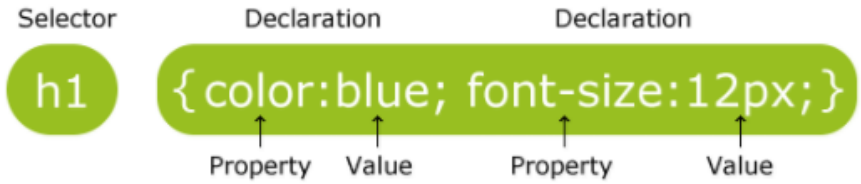
주요 태그	태그 설명
<html>	HTML 문서임을 나타내는 태그
<head>	문서 정보, 메타 데이터, 외부 파일 정보 등을 기술
<body>	본문을 정의하는 태그로 이미지, 표, 문자를 표현
<table>	표를 정의하는 태그
	순서가 있는 목록을 표현하는 HTML 태그
	순서가 없는 목록을 표현하는 HTML 태그

1. 웹 수집

- **CSS (Cascading Style Sheet)**
  - Markup Language가 실제 표시되는 방법 기술

CSS의 구성

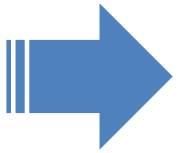
- Selector: 스타일을 어디에 적용할지 결정
- 종류
  - Element selector
  - ID selector
  - Class selector



Element Selector

- CSS를 통해 스타일 적용할 태그를 직접 지정하는 방식

```
<head>
    <style>
        p { color:blue; }
    </style>
</head>
```

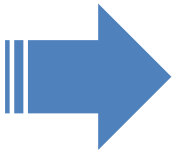


```
<p> para1 </p>
<p> para2 </p>
...
```

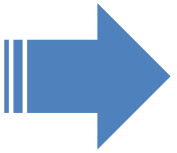
ID Selector

- id에 Style 지정: id는 한 html 파일에서 유일

```
<head>
    <style>
    #title {
        color:red;
    }
    </style>
</head>
```



```
<p id = "title"> Hello </p>
<p> Python </p>
<p> Web Scraping </p>
```



Hello

Python

Web Scraping

Class Selector

- 동일한 Class를 갖는 모든 tag에 적용

```
<head>
    <style>
    .paragraph {
        color:blue;
    }
    </style>
</head>
```

```
<p id = "title"> Hello </p>
<p class="paragraph"> Python </p>
<p class="paragraph"> Web Scraping </p>
```

Hello

Python

Web Scraping

class가 포함된 태그 선택

웹 수집 = Web Crawling = Web Scraping

- 1) 크롤링(Crawling), 스크래핑(Scarping)
- 2) HTTP(Hyper Text Transfer Protocol): 인터넷에서 데이터를 주고 받을 수 있는 프로토콜
- 3) HTML: 웹에 사용되는 표준 Markup언어
- 4) 파싱(Parsing) : 어떤 페이지(문서, html)에서 원하는 데이터를 특정 패턴이나 순서로 추출하여 정보 가공

웹 수집 = Web Crawling = Web Scraping

- URL로 호출한 웹페이지로 부터 원하는 정보를 수집!
  - Requests 패키지 사용

- 원하는 정보: Html, CSS 등으로 선택
  - BeuatifulSoup 패키지 사용

CSS Combinators: CSS 연결자

- **Selectors들 간의 관계 설명**
  - Descendant selector (space)
    - 모든 하위 selector들
  - Child selector (>)
    - 하위 selector만 대상

Descendant Selector

- UL 안의 <li> 안의 데이터만을 선택

```
<ul class="data1">
        <li>a</li>
        <li>b</li>
        <ul>
            <li>c</li>
            <li>d</li>
        </ul>
</ul>

<ul class="data2">
        <li>e</li>
        <li>f</li>
</ul>
```

```
<style>
        .data1     li { color:red; }
</style>
```

```
<style>
        ul      li { color:red; }
</style>
```

Child Selector

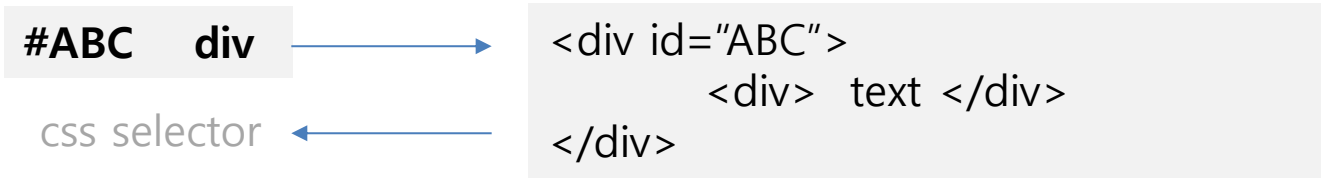
- > 를 사용하면 후손 중 자식만 지정

```
<ul class="data1">
        <li>a</li>
        <li>b</li>
        <ul>
            <li>c</li>
            <li>d</li>
        </ul>
</ul>

<ol class="data2">
        <li>e</li>
        <li>f</li>
</ul>
```

```
<style>
        .data1 > li  { color:red; }
</style>
```

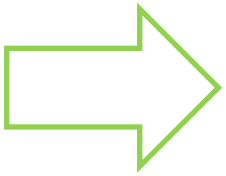
CSS Combinators



Example

- "hi" 문자열에 대한 CSS Selector

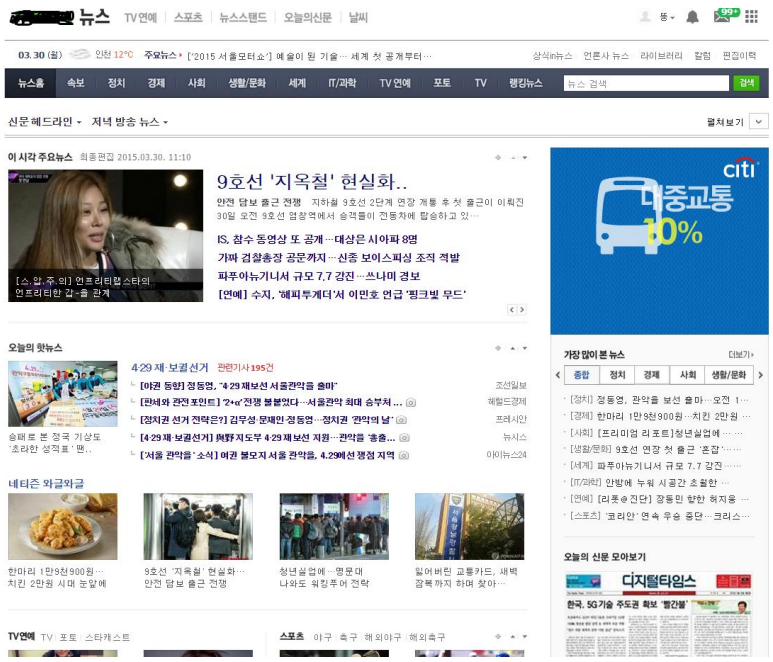
```
<tag id="ABC">
        <div>
                < p class="name"> hi  </p>
        </div>
</tag1>
```



```
#ABC    div             p
#ABC        p
.name
```

2. 비정형데이터와 텍스트 전처리

- **왜 Unstructured data? Text ?**
  - 대략 전세계 데이터의 80%는 unstructured formats
  - 텍스트 데이터! :Web, PDF, etc…

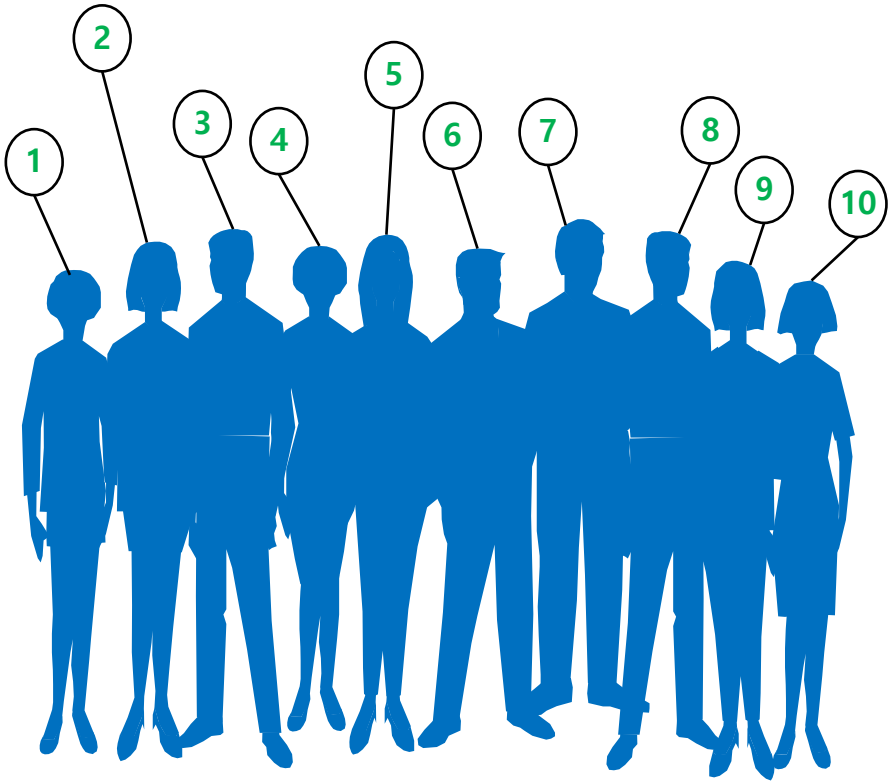


Web contents

PDF contents

2. 비정형데이터와 텍스트 전처리

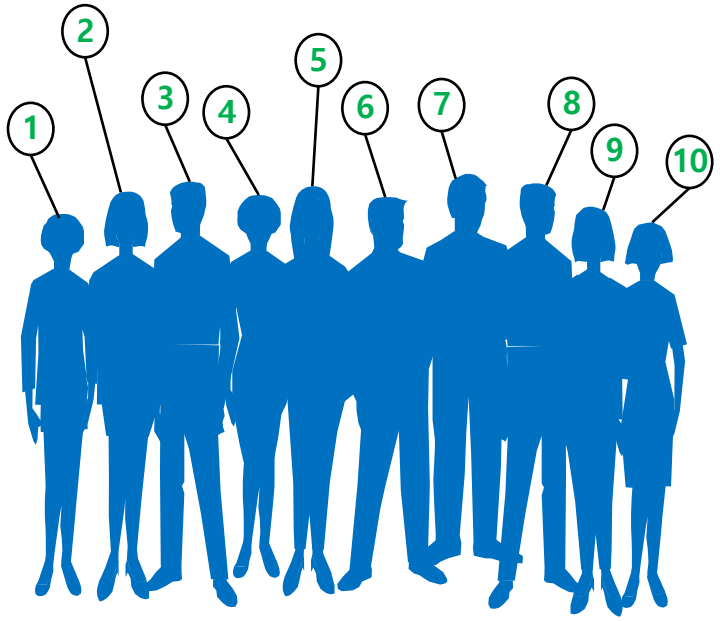
- **데이터**
  - 예를 들어, 10명의 사람들의 데이터



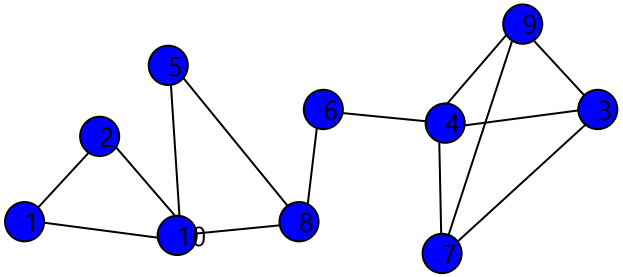
10명의 사람들을 이해한다면?

2. 비정형데이터와 텍스트 전처리

- **데이터의 유형**
  - 이름/성별/나이/거주지/직업
  - 월별 요금/데이터사용량
  - 휴대폰 교체주기/휴대폰 선호도
  - 서로 전화하는 관계 여부
  - 이용자의 VOC /서비스 선호도
  - 주로 방문하는 인터넷 사이트

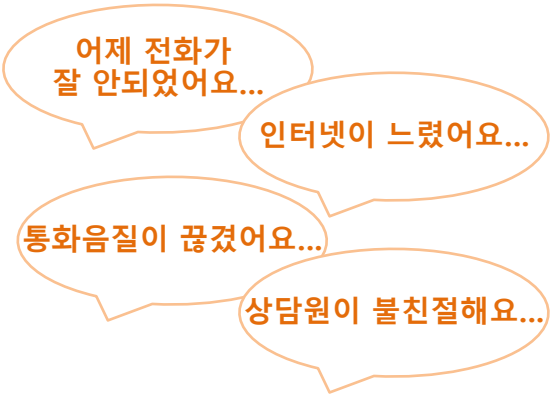


2. 비정형데이터와 텍스트 전처리



반정형데이터

비정형데이터



정형데이터

이름	성별	나이	거주지	직업	요금	데이터 사용량	휴대폰 선호도	서비스선 호도
AAA	F	20	서울	회사원	55000	3GB	LG	5
BBB	F	19	인천	자영업	45000	9GB	삼성	4
CCC	M	25	김포	회사원	35000	1GB	샤오미	3
DDD	F	42	대전	회사원	75000	4GB	LG	5
EEE	F	27	서울	자영업	65000	2GB	소니	4
FFF	M	20	서울	회사원	55000	3GB	LG	5
GGG	M	43	서울	자영업	45000	9GB	삼성	4
HHH	M	25	대전	회사원	95000	11GB	샤오미	3
III	F	42	김포	회사원	45000	3GB	LG	5
JJJ	F	27	인천	자영업	40000	4GB	소니	4

2. 비정형데이터와 텍스트 전처리

- **Examples**
  - 네이버 부동산 뉴스

2. 비정형데이터와 텍스트 전처리

- **Examples**
  - 네이버 부동산 뉴스 수집

"450조 전세보증금, 월세화에 가계부채 '뇌관'되나"
전세의 월세화가 진행되면서 450조원에 달하는 전세보증금이 가계부채 문제의 뇌관이 될 수 있다는 지적이 나왔다. 세입자들은 월세에 부담을 느껴 주택 매입에 나서고, 집주인들은 보증금을...

다시 늘어난 아파트 미분양...부동산 열기 냉각?
뜨겁게 달아오르던 부동산 시장이 다시 냉각되는 조짐입니다. 아파트 미분양이 최근 다시 크게 늘어난 것인데요. 앞으로 부동산시장에 찬물을 끼얹을 것으로 우려됩니다. 김대도 기자가 취재...

노후주택 많은 지역 '신규분양' 주목
최근 몇 년간 신규 공급이 없었던 경기 안산·의정부·포천·오산 등지에서 하반기 신규 분양이 이뤄질 예정이서 눈길을 끌고 있다. 공급 가뭄 지역은 기존 주택의 노후화로 새 아파트로 갈아타...

"가계부채 관리-부동산 경기부양" 두마리 토끼 잡을 수 있을까?
정부가 작년 8월부터 시행중인 주택담보대출비율(LTV)과 총부채상환비율(DTI) 규제 완화 조치를 내년까지 1년 연장키로 확정했다. 이미 행정예고 등의 절차를 거쳐 기정사실화된 사안이지만 ...

2. 비정형데이터와 텍스트 전처리

- **Examples**
  - 네이버 부동산 뉴스
    - 각 기사의 의미는 단어에 의해 전달
    - 각 기사별 단어를 파악하는 것이 중요

"450조 전세보증금, 월세화에 가계부채 '뇌관'되나"
전세의 월세화가 진행되면서 450조원에 달하는 전세보증금이 가계부채 문제의 뇌관이 될 수 있다는 지적이 나왔다. 세입자들은 월세에 부담을 느껴 주택 매입에 나서고, 집주인들은 보증금을...

다시 늘어난 아파트 미분양...부동산 열기 냉각?
뜨겁게 달아오르던 부동산 시장이 다시 냉각되는 조짐입니다. 아파트 미분양이 최근 다시 크게 늘어난 것인데요. 앞으로 부동산시장에 찬물을 끼얹을 것으로 우려됩니다. 김대도 기자가 취재...

노후주택 많은 지역 '신규분양' 주목
최근 몇 년간 신규 공급이 없었던 경기 안산·의정부·포천·오산 등지에서 하반기 신규 분양이 이뤄질 예정이서 눈길을 끌고 있다. 공급 가뭄 지역은 기존 주택의 노후화로 새 아파트로 갈아타...

"가계부채 관리-부동산 경기부양" 두마리 토끼 잡을 수 있을까?
정부가 작년 8월부터 시행중인 주택담보대출비율(LTV)과 총부채상환비율(DTI) 규제 완화 조치를 내년까지 1년 연장키로 확정했다. 이미 행정예고 등의 절차를 거쳐 기정사실화된 사안이지만 ...

2. 비정형데이터와 텍스트 전처리

- **Examples**
  - 네이버 부동산 뉴스
    - 주요 단어를 추출: 예를 들어 명사만 선택

전세보증금 월세화 가계부채 뇌관 전세 월세화 진행 전세보증금 가계부채 문제 뇌관 지적 세입자 월세 부담 주택 매입 집주인 보증금

아파트 미분양 부동산 냉각 부동산 시장 냉각 조짐 아파트 미분양 부동산시장 찬물 우려

노후주택 지역 신규분양 주목 신규 공급 경기 안산 의정부 포천 오산 하반기 신규 분양 예정 눈길 공급 가뭄 지역 기존 주택 노후화 아파트

가계부채 관리 부동산 경기부양 토끼 정부 주택담보대출비율 총부채상환비율 규제 완화 조치 연장 확정 행정예고 절차 기정사실화 사안

2. 비정형데이터와 텍스트 전처리

- **Examples**
  - 네이버 부동산 뉴스
    - 주요 단어를 추출: 예를 들어 명사만 선택

기사1 전세보증금 월세화 가계부채 뇌관 전세 월세화 진행 전세보증금 가계부채 문제 뇌관 지적 세입자 월세 부담 주택 매입 집주인 보증금

기사2 아파트 미분양 부동산 냉각 부동산 시장 냉각 조짐 아파트 미분양 부동산시장 찬물 우려

기사3 노후주택 지역 신규분양 주목 신규 공급 경기 안산 의정부 포천 오산 하반기 신규 분양 예정 눈길 공급 가뭄 지역 기존 주택 노후화 아파트

기사4 가계부채 관리 부동산 경기부양 토끼 정부 주택담보대출비율 총부채상환비율 규제 완화 조치 연장 확정 행정예고 절차 기정사실화 사안

2. 비정형데이터와 텍스트 전처리

- **Examples**
  - 네이버 부동산 뉴스
    - 주요 단어를 추출: 예를 들어 명사만 선택
    - 단어를 문서별로 정리하기

	전세보증금	월세화	가계부채	미분양	경기부양	…
기사1	2	2	1	0	0	
기사2	0	0	0	1	0	
기사3	0	0	0	0	0	
기사4	0	0	1	0	1	

2. 비정형데이터와 텍스트 전처리

- **Examples**
  - 네이버 부동산 뉴스
    - 정리된 결과를 다음과 같이 표현…

변수처럼 사용

Observation처럼 사용

	전세보증금	월세화	가계부채	미분양	경기부양	…
기사1	2	2	1	0	0	
기사2	0	0	0	1	0	
기사3	0	0	0	0	0	
기사4	0	0	1	0	1	

3. 텍스트마이닝

- **Text Mining**
  - Text-based (digitized) documents을 대상으로 함
    - e-mails, corporate Web pages, customer surveys, résumés, medical records, DNA sequences, technical papers, incident reports, news stories
  - 모든 대상 문서로 부터, 지식이나 요약 정보를 얻고자 함

- **Text Mining Application**

  ① 정보 추출/요약 및 시각화

  ② 문서의 군집화 및 주제 발견

  ③ 문서 분류

  ④ 추천

  ⑤ 정형데이터와 같이 사용

3. 텍스트마이닝

- **Term / document matrix**
  - Most common form of representation in text mining
  - Can be large: 크기가 매우 큰 행렬 -> Sparse matrix
  - Can be binary, or use counts: 행렬의 값은 문서 내 단어의 빈도 또는 1/0으로 표시

Example: 10 documents: 6 terms

	Regression	Classification	Clustering	Exploration	Process	Opensource
D1	24	21	9	0	0	3
D2	32	10	5	0	3	0
D3	12	16	5	0	0	0
D4	6	7	2	0	0	0
D5	43	31	20	0	3	0
D6	2	0	0	18	7	6
D7	0	0	1	32	12	0
D8	3	0	0	22	4	4
D9	1	0	0	34	27	25
D10	6	0	0	17	4	23

$$D_1 = (d_{i1}, d_{i2}, \ldots, d_{it})$$

- 각 문서는 단어들로 이뤄진 Vector->Vector Space Model

3. 텍스트마이닝

- **KoNLPy**
  - 코엔엘피와이
  - 한글 처리를 위한 모듈


```
>>> from konlpy.tag import Kkma
>>> from konlpy.utils import pprint
>>> kkma = Kkma()
>>> pprint(kkma.sentences(u'네, 안녕하세요. 반갑습니다.'))
[네, 안녕하세요..,
 반갑습니다.]
>>> pprint(kkma.nouns(u'질문이나 건의사항은 깃헙 이슈 트래커에 남겨주세요.'))
[질문,
 건의,
 건의사항,
 사항,
 깃헙,
 이슈,
 트래커]
>>> pprint(kkma.pos(u'오류보고는 실행환경, 에러메세지와함께 설명을 최대한상세히!^^'))
[(오류, NNG),
 (보고, NNG),
 (는, JX),
 (실행, NNG),
 (환경, NNG),
 (,, SP),
 (에러, NNG),
 (메세지, NNG),
```

- **설치 과정**
  - Java 1.7 이상
  - JAVA_HOME 설정
  - JPype1 (0.5.7 이상)을 설치
  - Pip를 이용한 설치

3. 텍스트마이닝

- **KoNLPy 설치**
  - pip install --upgrade pip
  - pip install JPype1-0.5.7-cp27-none-win_amd64.whl
  - pip install konlpy

- **코퍼스와 사전**
  - 코퍼스:
    - 연세 말뭉치
    - 고려대 한국어 말뭉치
    - HANBTEC 2.0
    - HKIB-40075
    - KAIST
    - Sejong
  - 사전:
    - Hannanum
    - Kkma
    - Mecab

3. 텍스트마이닝

Open Korean Text (ntags=19)	Komoran (ntags=42)	Mecab-ko (ntags=43)	Kkma(ntags=56)	Hannanum (ntags=22)
명사 (Nouns, Pronouns, Company Names, Proper Noun, Person Names, Numerals, Standalone, Dependent)	일반 명사	일반 명사	보통명사	보통명사
동사	고유 명사	고유 명사	고유명사	고유명사
형용사	의존 명사	의존 명사	일반 의존 명사	의존명사
관형사 (ex: 새, 헌, 참, 첫, 이, 그, 저)	수사	단위를 나타내는 명사	단위 의존 명사	수사
부사 (ex: 잘, 매우, 빨리, 반드시, 과연)	대명사	수사	수사	대명사
접속사	동사	대명사	대명사	동사
감탄사 (ex: 헐, 어머나, 얼씨구)	형용사	동사	동사	형용사
조사 (ex: 의, 에, 에서)	보조 용언	형용사	형용사	보조 용언
선어말어미 (ex: 었)	긍정 지정사	보조 용언	보조 동사	관형사
어미 (ex: 다, 요, 여, 하댘ㅋㅋ)	부정 지정사	긍정 지정사	보조 형용사	부사
접미사	관형사	부정 지정사	긍정 지정사, 서술격 조사 '이다'	감탄사
구두점	일반 부사	관형사	부정 지정사, 형용사 '아니다'	격조사
외국어, 한자 및 기타기호	접속 부사	일반 부사	수 관형사	보조사
알파벳	감탄사	접속 부사	일반 관형사	서술격 조사
숫자	주격 조사	감탄사	일반 부사	선어말어미
미등록어	보격 조사	주격 조사	접속 부사	종결 어미
(ex: ㅋㅋ)	관형격 조사	보격 조사	감탄사	연결 어미
트위터 해쉬태그 (ex: #히히)	목적격 조사	관형격 조사	주격 조사	전성 어미
트위터 아이디 (ex: @echojuliett)	부사격 조사	목적격 조사	보격 조사	접두사
이메일 주소	호격 조사	부사격 조사	관형격 조사	접미사
웹주소	인용격 조사	호격 조사	목적격 조사	기호
	접속 조사	인용격 조사	부사격 조사	외국어
	보조사	접속 조사	호격 조사	
	선어말어미	보조사	인용격 조사	
	종결 어미	선어말어미	접속 조사	
	연결 어미	종결 어미	보조사	
	명사형 전성 어미	연결 어미	존칭 선어말 어미	
	관형형 전성 어미	명사형 전성 어미	시제 선어말 어미	
	체언 접두사	관형형 전성 어미	공손 선어말 어미	
	명사파생 접미사	체언 접두사	평서형 종결 어미	

3. 텍스트마이닝

Open Korean Text (ntags=19)	Komoran (ntags=42)	Mecab-ko (ntags=43)	Kkma(ntags=56)	Hannanum (ntags=22)
	형용사 파생 접미사	동사 파생 접미사	명령형 종결 어미	
	어근	형용사 파생 접미사	청유형 종결 어미	
	마침표, 물음표, 느낌표	어근	감탄형 종결 어미	
	줄임표	마침표, 물음표, 느낌표	존칭형 종결 어미	
	따옴표,괄호표,줄표	줄임표 …	대등 연결 어미	
	쉼표,가운뎃점,콜론,빗금	여는 괄호 (, [	보조적 연결 어미	
	붙임표(물결,숨김,빠짐)	닫는 괄호),]	의존적 연결 어미	
	기타기호 (논리수학기호, 화폐기호)	구분자 , · / :	명사형 전성 어미	
	한자	기타 기호	관형형 전성 어미	
	외국어	한자	체언 접두사	
	숫자	외국어	용언 접두사	
	명사추정범주	숫자	명사파생 접미사	
	용언추정범주		동사 파생 접미사	
	분석불능범주		형용사 파생 접미사	
			어근	
			마침표, 물음표, 느낌표	
			줄임표	
			따옴표,괄호표,줄표	
			쉼표,가운뎃점,콜론,빗금	
			붙임표(물결,숨김,빠짐)	
			기타기호 (논리수학기호, 화폐기호)	
			한자	
			외국어	
			숫자	
			명사추정범주	

3. 텍스트마이닝

- **Term / document matrix**
  - Most common form of representation in text mining
  - Can be large: 크기가 매우 큰 행렬 -> Sparse matrix
  - Can be binary, or use counts: 행렬의 값은 문서 내 단어의 빈도 또는 1/0으로 표시

Example: 10 documents: 6 terms

	Regression	Classification	Clustering	Exploration	Process	Opensource
D1	24	21	9	0	0	3
D2	32	10	5	0	3	0
D3	12	16	5	0	0	0
D4	6	7	2	0	0	0
D5	43	31	20	0	3	0
D6	2	0	0	18	7	6
D7	0	0	1	32	12	0
D8	3	0	0	22	4	4
D9	1	0	0	34	27	25
D10	6	0	0	17	4	23

$$D_1 = (d_{i1}, d_{i2}, \ldots, d_{it})$$

- 각 문서는 단어들로 이뤄진 Vector->Vector Space Model

3. 텍스트마이닝

- **Stop words and Stemming**
  - Text mining을 통해 발견된 단어의 리스트를 모두 사용?

  - **Stop Words**
    - 텍스트마이닝 및 정보검색에서는 큰 의미가 없는 가장 빈번하게 사용되는 단어
      - 예: the, of, and, to, ….
      - 대략 400-500개 단어임(영어)
    - Stop words 처리를 통해
      - 데이터 사이즈 축소: 일반적으로 전체 단어의 20-30%정도를 차지
      - 효율성 제고!: 정보검색이나 텍스트 마이닝에 큰 의미가 없는 단어들 제거

  - **Stemming**
    - 단어의 어근(root/stem)을 찾는 기법
      - 예
        - user / users /used /using -> use
        - Engineering/engineered -> engineer
    - 효용성
      - 정보검색 및 텍스트 마이닝 성능 향상
      - 뜻은 같지만 형태가 다른 단어들의 매치
      - 데이터 사이즈 축소
        - » 일반적으로 40-50%정도의 크기를 줄이는 것으로 알려짐(영어)

3. 텍스트마이닝

- **Weighting in Term Document Matrix**
  - 모든 단어가 똑같지 않음!
    - 예: "햄버거"는 "와퍼"보다 덜 중요
    - 많은 문서에서 많이 출현하는 단어는 그만큼 특정 문서에 대한 discriminatory power가 적음
    - 가중치 사용을 통해 보완: inverse-document frequency

$$\mathrm{IDF} = \log(N/n_j)$$

    - Term importance = Term Frequency (TF) x IDF
      - nj= # of docs containing the term
      - N = total # of docs
      - 해석: TF가 높으면서 IDF가 높은 단어가 중요
      - **TF x IDF:** 단어 중요도에 대한 일반적 measure

3. 텍스트마이닝

	Regression	Classification	Clustering	Exploratio n	Process	Opensource
D1	24	21	9	0	0	3
D2	32	10	5	0	3	0
D3	12	16	5	0	0	0
D4	6	7	2	0	0	0
D5	**43**	31	20	0	3	0
D6	2	0	0	18	7	6
D7	0	0	1	32	12	0
D8	3	0	0	22		
D9	1	0	0	34		
D10	6	0	0	17		

	Regression	Classification	Clustering	Exploration	Process	Opensource
D1	2.53	14.6	4.6	0	0	2.1
D2	3.3	6.7	2.6	0	1.0	0
D3	1.3	11.1	2.6	0	0	0
D4	0.7	4.9	1.0	0	0	0
D5	4.5	**21.5**	10.2	0	1.0	0
D6	0.2	0	0	12.5	2.5	11.1
D7	0	0	0.5	**22.2**	4.3	0
D8	0.3	0	0	15.2	1.4	1.4
D9	0.1	0	0	**23.56**	9.6	17.3
D10	0.6	0	0	11.8	1.4	16.0

3. 텍스트마이닝

- **텍스트 마이닝**
  - 비정형 및 반정형 텍스트 데이터에서 자연어 처리 기술을 사용하여 유용한 정보를 추출 및 가공하는 기법/방법
  - 텍스트마이닝을 통하여 Corpus에서 의미있는 정보를 추출하여 단순한 정보 검색 이상의 결과를 기대

- **텍스트 마이닝 응용 분야**
  - 분류(Categorization)/감성 분석(Sentiment Analysis)/요약(Summarization) 및 군집화

- **텍스트 마이닝 절차**
  - 텍스트-> 전처리 -> 구조화->모델링 -> 평가
  - 자연어 처리를 통한 구조화: 자연어로 구성된 비정형 데이터를 정형화된 수치형 데이터로 표현하는 것이 필수
  -

3. 텍스트마이닝

- **Encoding**
  - Word Encoding: 각 문헌에 각 용어가 몇 번(Term Frequency) 나왔는지 수치화
    - TF
    - TF-IDF
  - 참고: 범주형 변수의 One hot encoding

  - 단어의 수 만큼 n차원 벡터를 만들어 수치화: Sparse Vector
  - 단어 간 유사도 계산에 제한
  - Bag of Words
  - 전통적인 텍스트 전처리

- **Embedding**
  - 고정된 저차원의 벡터로 용어를 표현
  - Dense Vector로 표현
  - 모형을 통해 학습

3. 텍스트마이닝

- **텍스트 전처리**
  - Cleansing : 텍스트 분석에 불필요한 문자/기호 등을 제거. 예를 들어 HTML, XML 태크나 특정 기호 등을 사전에 제거
  - Tokenization : 문장 토큰화, 단어 토큰화, N-gram
  - 필터링/스톱 워드(불용어) 제거/철자 수정: 불필요한 단어나 분석에 큰 의미가 없는 단어(a, the, is, will등) 그리고 잘못된 철자 수정
  - Stemming / Lemmatization : 어근(단어 원형) 추출, Lemmatization이 Stemming보다 정교하고 의미론적 기반에서 단어원형을 발견

- **토큰(token)**
  - 문법적으로 더 이상 나눌 수 없는 언어요소
  - 텍스트 토큰화(Text Toeknization)란 Corpus로부터 토큰을 분리

3. 텍스트마이닝

- **N-gram이란**
  - 문장을 개별 단어 별 토큰화 할 경우, 문맥적인 의미는 손실
  - N-gram은 연속된 n개의 단어를 하나의 토큰화 단위로 분리하는 것
  - n개 단어 크기 윈도우를 만들어 문장의 처음부터 오른쪽으로 움직이면서 토큰화 수행

  - 예 'Here is a dog'
  - 1-gram (단어 단위)
    - Here, is, a, dog
  - 1-gram (with 3-sized window)
    - "Her", "ere", "re_", "e_i", "_*is*", "*is*_", "s_a", "_*a*_", "a_d", "_do", "dog"
  - 2-gram(단어 단위)
    - "Here is", "is a", "a dog"

3. 텍스트마이닝

- **한글에서의 토큰화 이슈**
  - 교착어 특징
    - 한글 NLP에서 조사의 분리!
    - 영어NLP: 독립적인 단어가 띄어쓰기 단위로 토큰화
    - 한국어는 어절이 독립적인 단어로 구성되는 것이 아니며, 조사 등이 붙어있는 상태여서 분리가 필요
  - 형태소?
    - 한국어 토큰화의 형태소(morpheme)
    - 형태소(morpheme)란 뜻을 가진 가장 작은 말의 단위
      - 자립 형태소 : 접사, 어미, 조사와 상관없이 자립하여 사용할 수 있는 형태소. 그 자체로 단어. 체언(명사, 대명사, 수사), 수식언(관형사, 부사), 감탄사 등.
      - 의존 형태소 : 다른 형태소와 결합하여 사용되는 형태소. 접사, 어미, 조사, 어간 등.

3. 텍스트마이닝

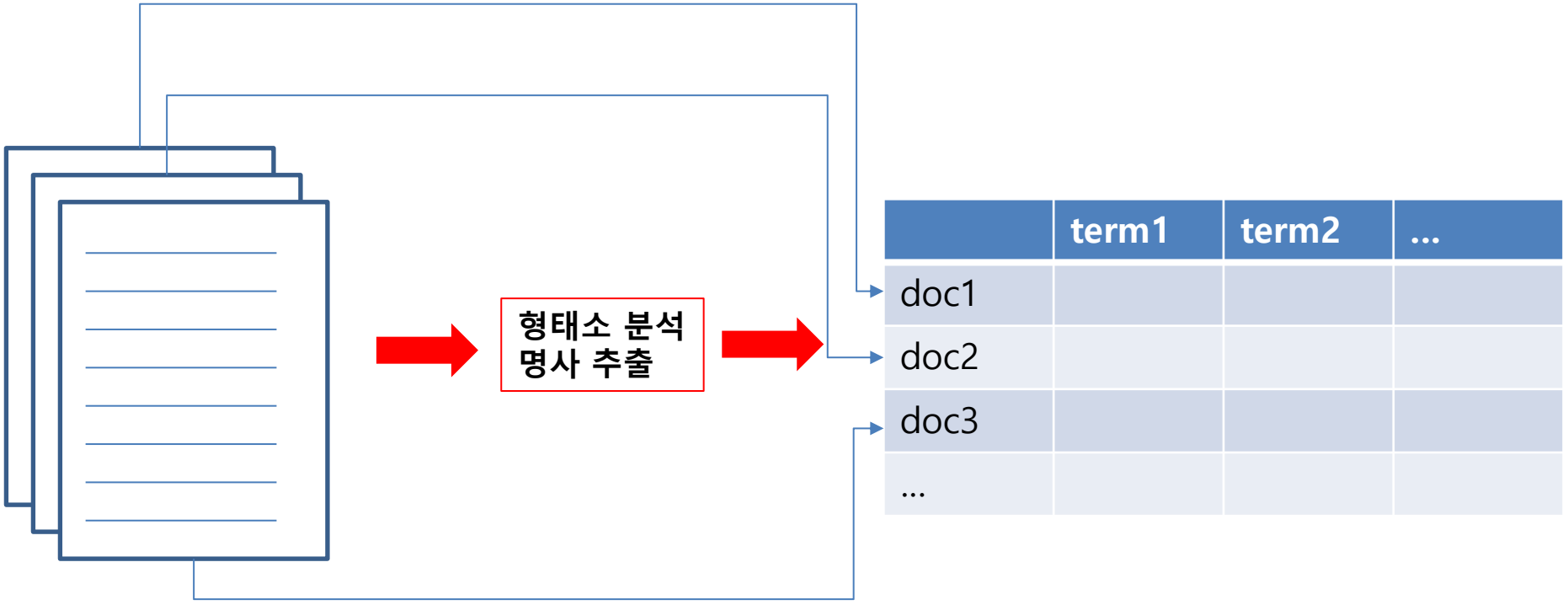
- **형태소 토큰화 예시**
  - 정통이가 통계책을 읽었다
  - 자립 형태소 : 정통이, 통계책
  - 의존 형태소 : -가, -을, 읽-, -었, -다
  - 한글 NLP에서는 어절 토큰화가 아니라 형태소 토큰화를 수행해야 함
- **띄어쓰기의 문제**
  - 한글 띄어쓰기의 어려움
  - 한글: 띄어쓰기가 지켜지지 않아도 이해가능한 언어
  - 띄어쓰기가 없던 한국어에 띄어쓰기가 보편화된 것도 근대(1933년, 한글맞춤법통일안)에 시작

4. 유사도

➢ **Bag of Words ?**

- 단어들의 순서는 고려하지 않고, 단어들의 출현 빈도 (frequency)로 텍스트를 수치화

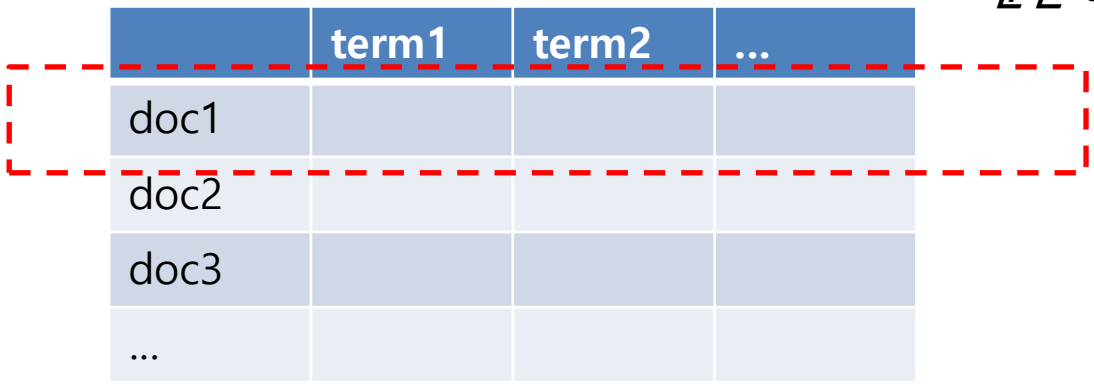
- BoW의 과정
  - 각 단어에 고유한 정수 인덱스 부여
  - 각 인덱스의 위치에 해당 단어 토큰의 등장 횟수를 기록한 벡터 생성



4. 유사도

- **Bag of Words**
  - 장점
    - 쉽고 빠른 구축
    - 예상보다 문서의 특징을 잘 표현
    - 전통적으로 여러 분야에서 활용
  - 단점
    - 문맥 의미(Semantic Context) 미반영 문제
    - 희소 행렬 문제

문헌 하나가 하나의 벡터



4. 유사도

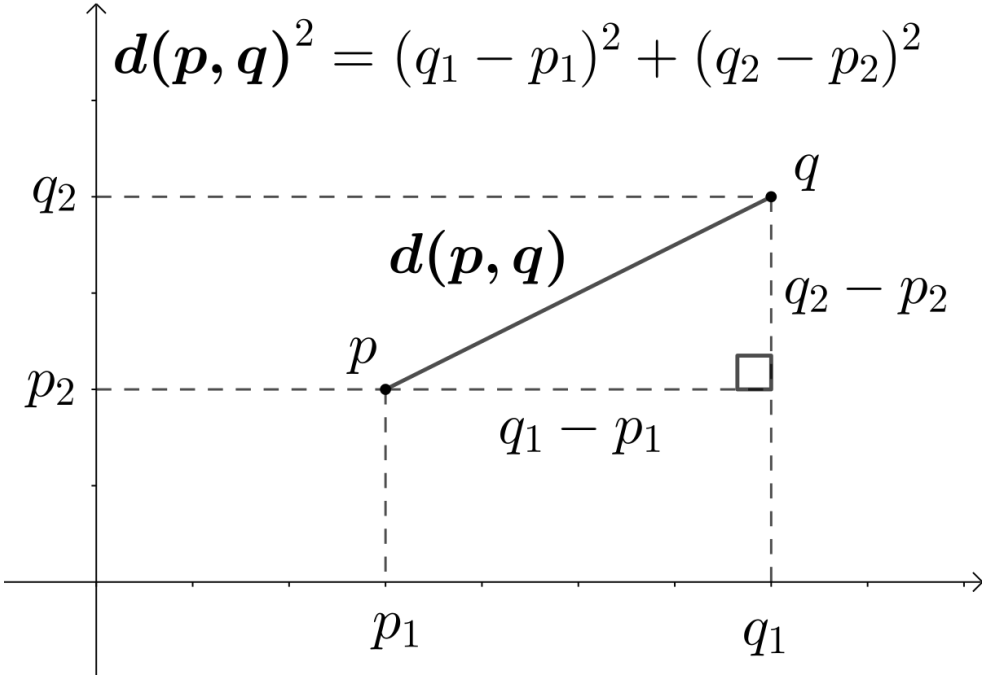
- **벡터의 유사도(Vector Similarity)**
  - 일반적인 문서(문헌)의 유사도: 문서들 간에 동일한 단어 또는 비슷한 단어가 얼마나 공통적으로 많이 출현하는지를 측정
  - 알고리즘 활용 문서(문헌)의 유사도: 각 문헌의 수치화된 단어들을 사용하여, 단어들의 차이를 어떻게 측정하는지에 따라 다름

- **예**
  - Corpus
    - 문서1 : 저는 사과 좋아요
    - 문서2 : 저는 바나나 좋아요
    - 문서3 : 저는 바나나 좋아요 저는 바나나 좋아요
  - DTM

-	바나나	사과	저는	좋아요
문서1	0	1	1	1
문서2	1	0	1	1
문서3	2	0	2	2

4. 유사도

- **1. 유클리드 거리(Euclidean distance)**
  - 다차원 공간에서 두개의 점 p 와 q 가 각각 p=(p1,p2,p3,⋯,pn) 과 q=(q1,q2,q3,...,qn) 의 좌표
  - 두 점 사이의 거리를 계산하는 유클리드 거리



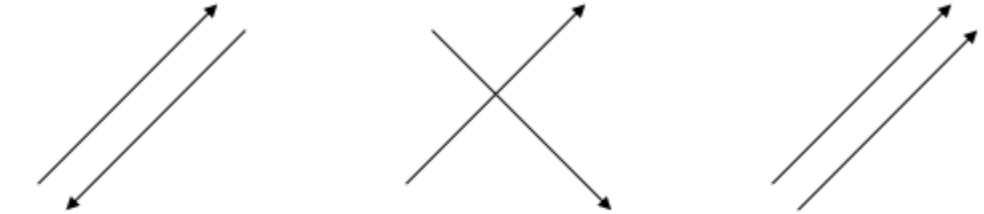
4. 유사도

➢ 2. 코사인 유사도(Cosine Similarity)

- 두 벡터 간의 코사인 각도를 이용하여 구할 수 있는 두 벡터의 유사도

$$similarity = cos(\Theta) = \frac{A \cdot B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

- 두 벡터의 방향이 완전히 동일한 경우에는 1, 90도 의 각을 이루면 0, 180도로 반대의 방향을 가지면 -1
- -1 이상 1이하의 값을 가지며 값이 1에 가까울수록 유사도가 높음



코사인 유사도 : -1　　코사인 유사도 : 0　　코사인 유사도 : 1

4. 유사도

➢ 코사인 유사도

-	바나나	사과	저는	좋아요
문서1	0	1	1	1
문서2	1	0	1	1
문서3	2	0	2	2

- 문서3은 문서2에서 단지 모든 단어의 빈도수가 1씩 증가
- 문서1과 문서2의 코사인 유사도와 문서1과 문서3의 코사인 유사도가 같음
- 문서2와 문서3의 코사인 유사도가 1
  - 한 문서 내의 모든 단어의 빈도수가 동일하게 증가하는 경우에는 기존의 문서와 코사인 유사도의 값이 1

- 코사인 유사도의 장점
  - 코사인 유사도가 아닌 경우, 원문의 길이가 긴 경우 높은 유사도
  - 코사인 유사도는 문서의 길이가 다른 상황에서 비교적 공정한 비교

4. 유사도

- **3. 자카드 유사도(Jaccard similarity)**
  - 두 집합에 대한 자카드 유사도는 0과 1사이의 값
  - 만약 두 집합이 동일하다면 1의 값을 가지고, 두 집합의 공통 원소가 없다면 0

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

  - 두 개의 비교할 문서가 각각 doc1, doc2
  - doc1과 doc2의 자카드 유사도

$$J(doc_1, doc_2) = \frac{doc_1 \cap doc_2}{doc_1 \cup doc_2}$$

4. 유사도

- **4. 레벤슈타인 거리(Levenshtein distance)**
  - 편집거리 알고리즘: 편집할 때 몇 번의 문자열 조작이 필요한지를 계산해서 거리 측정
  - 문자열이 얼마나 비슷한 지를 측정
  - 비슷한 어구 검색, DNA 배열의 유사성 판단 등 다양한 분야에서 활용

		E	L	E	P	H	A	N	T
	0	1	2	3	4	5	6	7	8
R	1	1	2	3	4	5	6	7	8
E	2	1	2	2	3	4	5	6	7
L	3	2	1	2	3	4	5	6	7
E	4	3	2	1	2	3	4	5	6
V	5	4	3	2	2	3	4	5	6
A	6	5	4	3	3	3	3	4	5
N	7	6	5	4	4	4	4	3	4
T	8	7	6	5	5	5	5	4	3

4. 유사도

- **4. 레벤슈타인 거리(Levenshtein distance)**
  - 편집거리 알고리즘: 편집할 때 몇 번의 문자열 조작이 필요한지를 계산해서 거리 측정
  - 문자열이 얼마나 비슷한 지를 측정
  - 비슷한 어구 검색, DNA 배열의 유사성 판단 등 다양한 분야에서 활용

삽 입 삭 제 변 경

단계별 이해

① 레벤슈타인 거리를 구하기 위해 2차원의 표(문장길이+1, 문장길이+1) 생성
② 첫 행, 첫 열은 문자열의 길이로 초기화
③ 표에서 빨간 부분을 채울 때,
- 노랑+1 : 문자 삽입
- 초록 +cost : 문자 변경, cost값은 비교하는 문자가 같으면 0 아니면 1이다.
- 파랑+1 : 문자 삭제
- 위의 3가지 값 중에서 최소값이 빨간 부분에 주어지며, 여기서는 0, cost값은 비교하는 문자가 둘 다 ' 가 ' 로 동일하게 때문에 0

④ 값을 계속 채워서 마지막열, 마지막 행 부분인 회색 부분이 최종적인 레벤슈타인 거리

1. 유 사 도 나 분 석 | 할 까 요
2. 얼 마 | 나 분 석 이 | 될 까 요

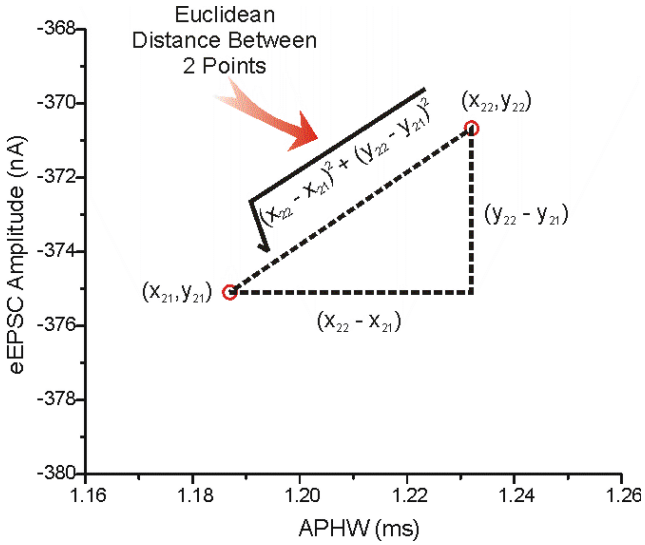
비용 계산 : 1 2 3 4 5

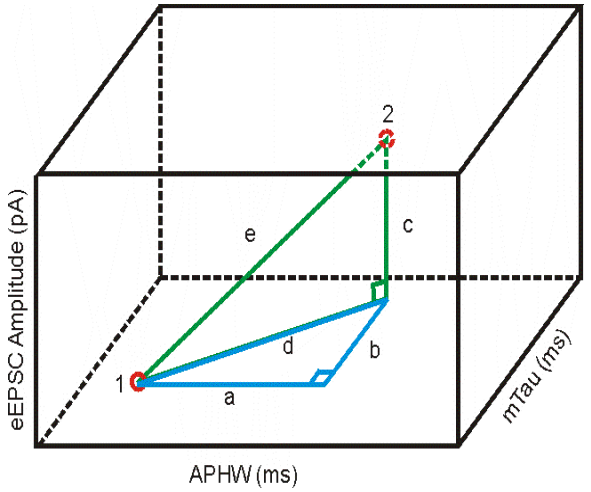
		가	나	다	라
	0	1	2	3	4
가	1				
마	2				
바	3				
라	4				

		가	나	다	라
	0	1	2	3	4
가	1	0	1	2	3
마	2	1	1	2	3
바	3	2	2	2	3
라	4	3	3	3	2

4. 유사도

- **Clustering?**
  - Cluster의 개수나 구조에 관한 특별한 사전 가정없이, 개체들 사이의 유사성/거리에 근거해 cluster를 찾고 다음 단계의 분석을 하게 하는 기법
  - 유사한 개체들을 cluster로 그룹화하여 각 집단의 성격을 파악





5. 토픽모델링

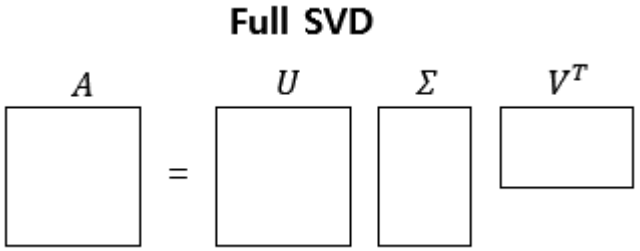
- **Topic modeling**
  - Topic Model
    - Generative probabilistic models for the term frequency of documents in a given corpus
    - 여러 방법들(LSI, pLSI, LDA, etc)이 있으며, LDA도 그 중의 하나

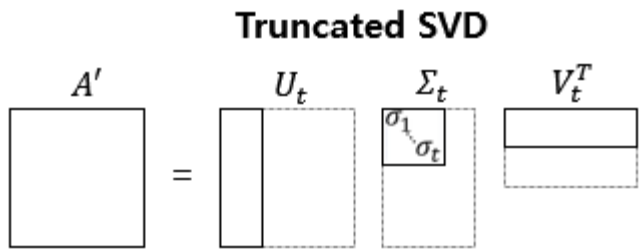
    - LSA(Latent SemanticAnalysis,), pLSA
    - LDA(Latent Dirichlet Allocation)
    - NMF(Non Negative Factorization)

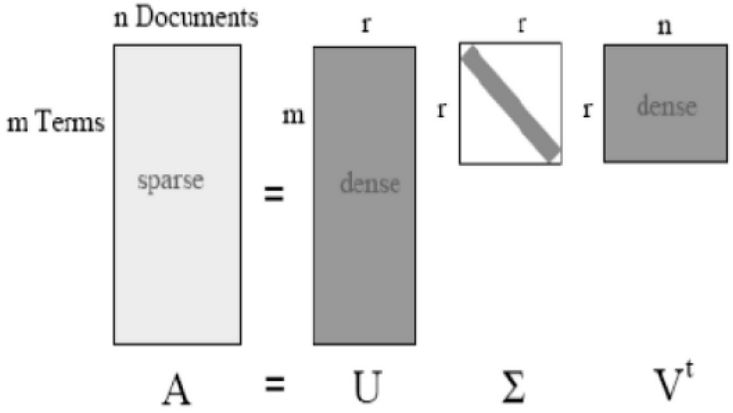
    - *LSA와 NMF는 행렬 분해 기반 토픽 모델링*
    - *pLSA와 LDA 확률 기반의 토픽 모델링*

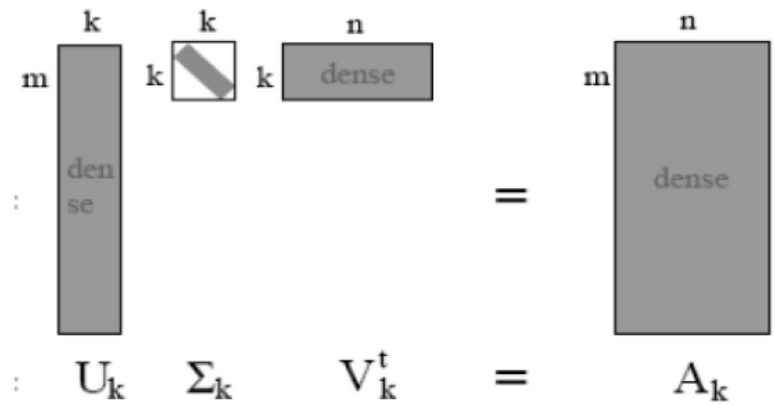
5. 토픽모델링

- **Topic modeling**
  - LSA(Latent Semantic Analysis,): Truncated SVD









5. 토픽모델링

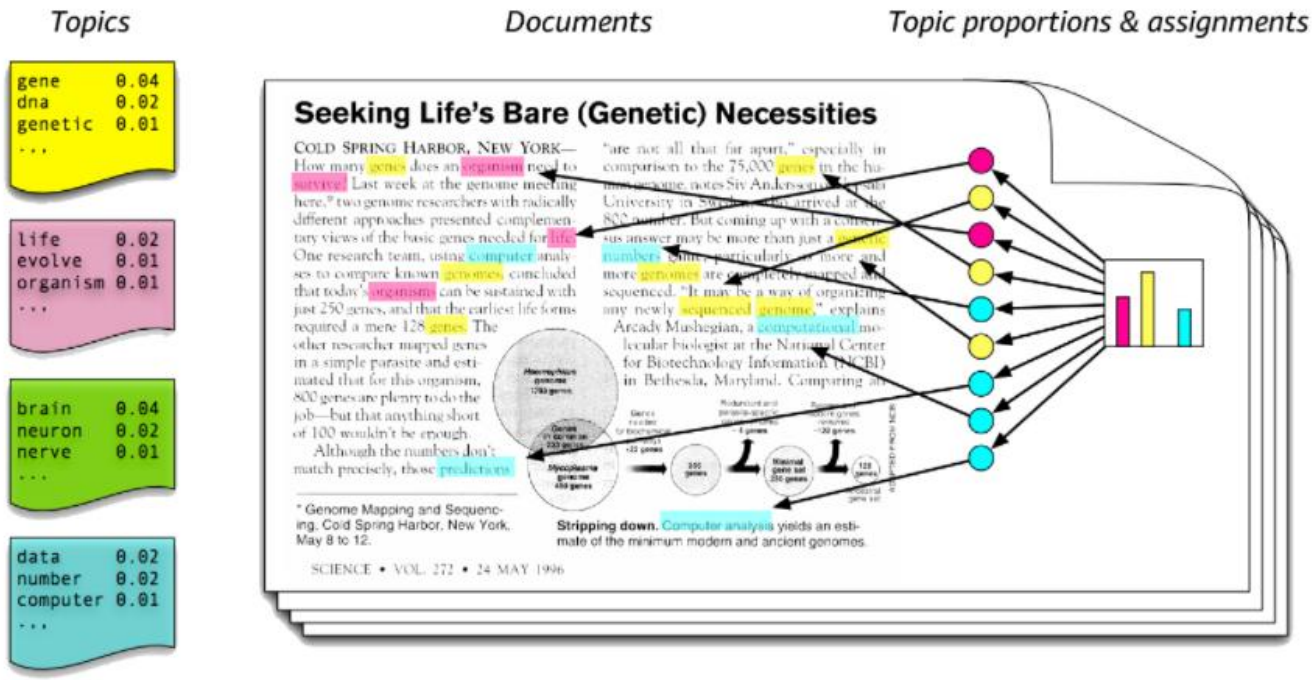
- **Topic modeling**
  - LDA
    - Latent Dirichlet Allocation

    - 배경
      - 예를 들어, 문서1과 문서2가 주제는 유사해도 각 문서에 등장하는 단어의 종류나 빈도는 다를 수 있는데, 단순한 키워드 기반의 모델로는 유사도를 계산하거나 주제 분류를 하는 데에는 한계가 발생
      - 많은 텍스트에 기초에 α와 β를 찾고, 개별 문서의 θ를 계산할 수 있으면, 이 θ를 가지고 유사도 계산이나 분류 작업을 할 수 있음

    - 특징
      - Bayesian mixture model for discrete data (Topics are assumed to be uncorrelated)
      - Mixture Membership Model (문서는 하나의 토픽에만 속하는 것이 아니라 여러 다른 토픽들의 혼합(Mixture)으로 정의할 수 있음)

5. 토픽모델링

- **Topic modeling**
  - LDA
    1. 개별 문서는 혼합된 여러 개의 주제로 구성
    2. 개별 주제는 여러 개의 단어로 구성
    3. BoW에 기반한 DTM이나 TF-IDF는 기본적으로 단어의 빈도 수를 이용한 수치화
    4. 단어의 의미나 순서를 고려하지 못함



5. 토픽모델링

- **Topic modeling**

Journal of Machine Learning Research 3 (2003) 993-1022 Submitted 2/02; Published 1/03

Latent Dirichlet Allocation

David M. Blei BLEI@CS.BERKELEY.EDU
Computer Science Division
University of California
Berkeley, CA 94720, USA

Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department
Stanford University
Stanford, CA 94305, USA

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720, USA

Editor: John Lafferty

Abstract

We describe *latent Dirichlet allocation* (LDA), a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document. We present efficient approximate inference techniques based on variational methods and an EM algorithm for

Dirichlet distribution
-연속 확률분포
-parameter: k차원의 실수(>0)를 갖는 continuous r.v.
-beta distribution의 multivariate generalization
-벡터 내 모든 element를 더한 값이 1인 경우, 다항분포의 모수에 사용
-다항분포에 대한 conjugate prior

$$f(x_1, \cdots, x_k; \alpha_1, \cdots, \alpha_k) = \frac{1}{\mathrm{B}(\alpha)} \prod_{i=1}^{k} x_i^{\alpha_i - 1}$$

$$\mathrm{B}(\alpha) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)} \quad (\Gamma \text{is gamma function})$$

5. 토픽모델링

- **Multinomial & Dirichlet Distribution**
  - Multinomial Distribution:여러 개의 값을 가질 수 있는 독립 확률변수들에 대한 확률분포로, 여러 번의 독립적 시행에서 각각의 값이 특정 횟수가 나타날 확률을 정의

$$p(x_1, x_2, \cdots, x_n; n, p_1, \cdots, p_k) = \frac{n!}{x_1! x_2! \cdots x_k!} p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k}$$

  - Dirichlet Distribution: k차원의 실수 벡터 중 벡터의 요소가 양수이며 모든 요소를 더한 값이 1인 경우에 대해 확률값이 정의되는 분포

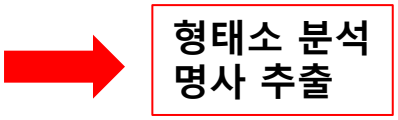
$$f(x_1, \cdots, x_k; \alpha_1, \cdots, \alpha_k) = \frac{1}{\mathrm{B}(\alpha)} \prod_{i=1}^{k} x_i^{\alpha_i - 1} \qquad \mathrm{B}(\alpha) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}$$

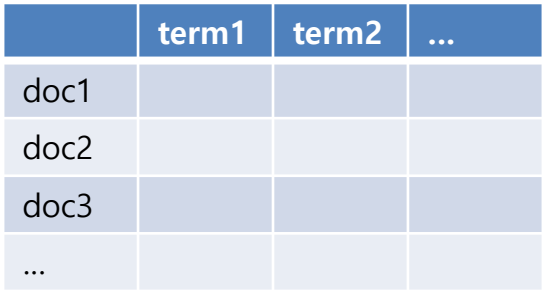
  - Conjugate Prior
    - posterior와 prior가 동일한 분포를 따르면, prior를 likelihood의 conjugate prior

Likelihood	Conjugate Prior	Posterior
Binomial(N, θ)	Beta(r,s)	Beta(r+n, s+N-n)
Multinomial($\theta_1, \ldots, \theta_K$)	Dirichlet(($\alpha_1, \ldots, \alpha_K$)	Dirichlet($\alpha_1 + n_1, \ldots, \alpha_K + n_K$)

5. 토픽모델링

- **Topic modeling**





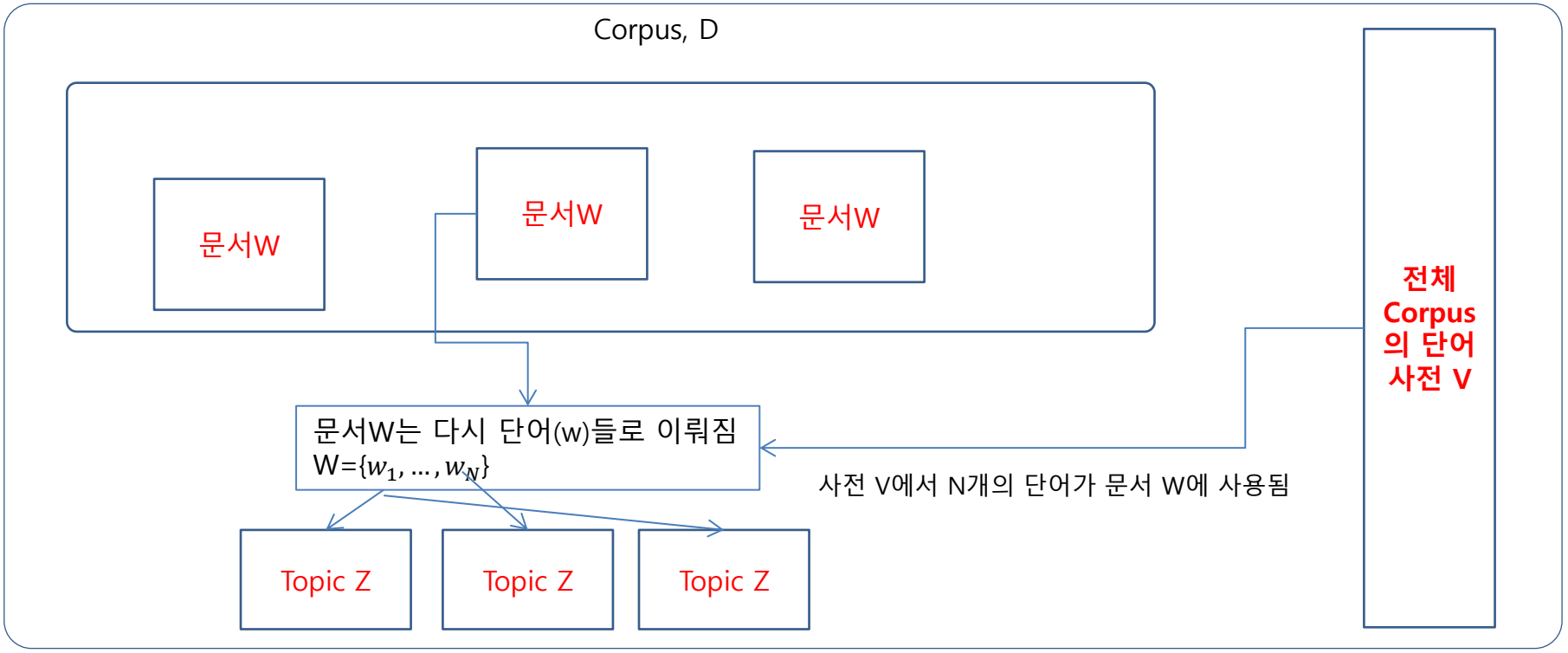
"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

Figure 8: An example article from the AP corpus. Each color codes a different factor from which the word is putatively generated.

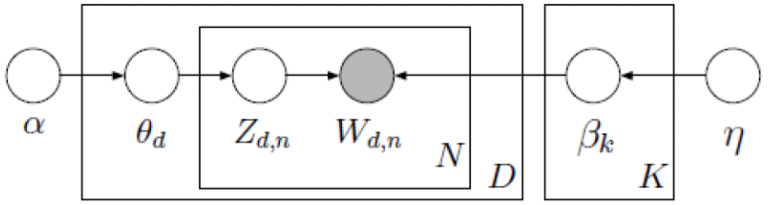
5. 토픽모델링

- **LDA(Latent Dirichlet Allocation)**
  - Topics의 수 K는 사전에 결정되어야 함
  - 문서 집합(Corpus)에 대한 Generative Probabilistic Model이며, 1) 문서의 주제 분포와 2) 주제별 단어의 분포를 알고 있다면, 특정 문서가 만들어질 확률을 알 수 있음



5. 토픽모델링

- **LDA Step**
  - Step 1
    - 각 주제(k=1,...,K)에 대해서 Draw Vector of Term Proportion from $\beta_k \sim Dirichlet_V(\eta)$
  - Step 2
    - 각 문서(d=1,...,D)에 대해서, Draw Topic proportions θ from $\theta_d \sim Dirichlet_K(\alpha)$
  - Step 3
    - For each of the N words(n=1,...,N) $w_{d,n}$
      - Choose a topic $z_{d,n} \sim Multinomial_K(1, \theta_d)$
      - Choose a word $w_{d,n}$ from a multinomial probability distribution conditioned on the topic z_i, $Multinomial_V(1, \beta_{z_{d,n}})$



5. 토픽모델링

- **More on LDA Steps…**
  - 위의 과정을 다시 표현하면 아래와 같음
    - $w_{d,n} \mid z_{d,n}, \theta_d, \beta \sim Multinomial_V(1, \beta_{z_{d,n}})$
    - $z_{d,n} \mid \theta_d \sim Multinomial_K(1, \theta_d)$
    - $\theta_d \sim Dirichlet_K(\alpha)$
    - $(\beta_1, \ldots, \beta_K) \sim \prod_{k=1}^{K} Dirichlet_V(\eta)$

  - α와 η가 주어질때, (θ, z, β, w)의 $Joint\ Distribution$은 $p(\theta, z, \beta, w; \alpha, \eta)$이며 아래와 같이 표시됨

$$p(\theta, z, \beta, w; \alpha, \eta) = \left[\prod_{k=1}^{K} p(\beta_K|\eta)\right] \prod_{d=1}^{D} \left[p(\theta_d|\alpha) \prod_{n=1}^{N} p\left(z_{d,n}|\theta_d\right) p(w_{d,n}|z_{d,n}, \theta_d, \beta)\right]$$

  - θ, z, β는 $parameter$이고 w는 관측된 데이터여서, (θ, z, β)의 $Posterior\ Distribution$은 아래와 같음

$$= \frac{(\theta,z,\beta,w)\text{의 } Joint\ Distribution}{w\text{의 } Marginal\ Distribution} = \frac{p(\theta,z,\beta,w|\alpha,\eta)}{\int\int \sum_z p(\theta,z,\beta,w|\alpha,\eta) d\theta d\beta}$$

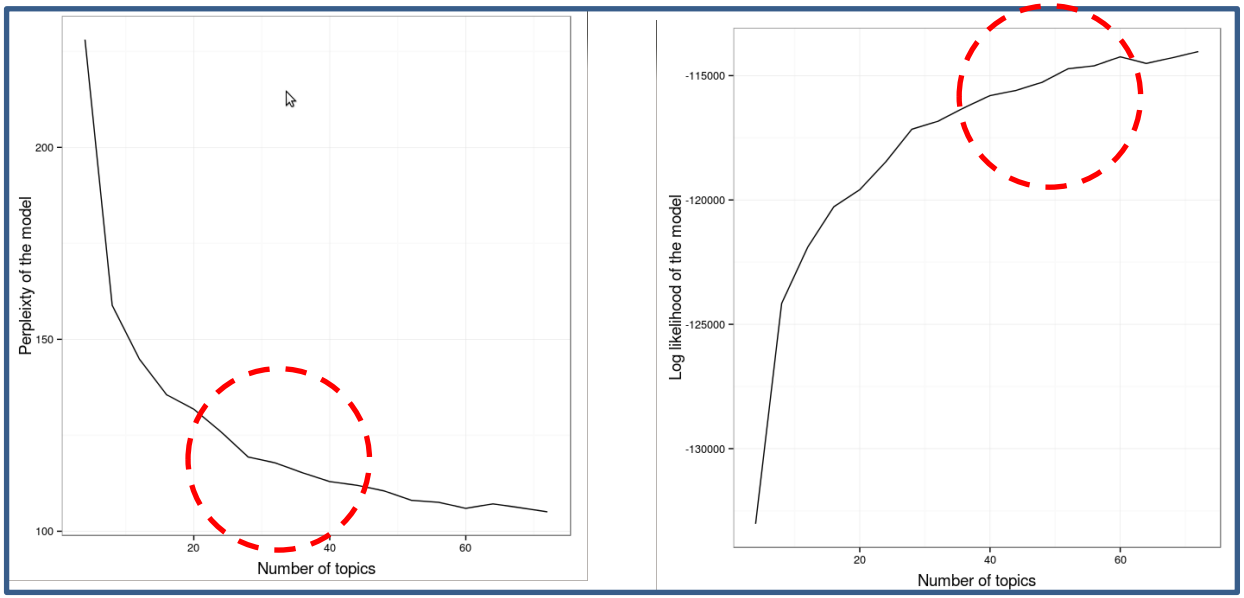
  - **LDA를 통해 아래의 값을 구하고자 함**
    - Topic probability of term $\hat{\beta}_{K,V} = E_\pi\left[\beta_{K,V}|w\right]$
    - Per-Document topic proportion $\hat{\theta}_{d,k} = E_\pi\left[\theta_{d,k}|w\right]$
    - Per-word topic proportion $\hat{z}_{d,n,k} = Pr_\pi\left(z_{d,n} = k|w\right)$

5. 토픽모델링

- **Parameter Estimation**
  - 데이터의 log-likelihood 를 최대화하는 Parameter 추정
  - 이번 분석에서는 Variational Expectation Maximization Algorithm을 사용

$$l(\alpha,\beta) = \log\big(p(w|\alpha,\beta)\big) = log \int \left\{ \sum_{z} \left[\prod_{i=1}^{N} p(w_i|z_i,\beta) p(z_i|\theta) \right] \right\} p(\theta|\alpha) d\theta$$

Model Selection: Choose K



5. 토픽모델링

- **Topic modeling**
  - LDA
    - ① 단순 Count 기반 Document-Term 행렬을 생성 : 주어진 단어들의 빈도수에 기반하므로 Tf-idf 방법이 아닌 Count에 기반
    - ② 토픽의 개수를 사전에 설정
    - ③ 각 단어들을 임의의 토픽으로 최초 할당한 후 문서별 토픽 분포와 토픽별 단어 분포가 결정
    - ④ 특정 단어를 하나 추출하고 추출한 해당 단어를 제외하고 문서의 토픽 분포와 토픽별 단어 분포를 다시 계산(Gibbs Sampling)
    - ⑤ 추출된 단어는 새롭게 토픽 할당 분포를 계산
    - ⑥ 다른 단어를 추출하고 4번 단계를 다시 수행, 계속해서 다른 단어를 추출하고 모든 단어들이 재계산되도록 반복
    - ⑦ 지정된 반복 횟수(하이퍼파라미터로 지정)만큼 4~6번 단계를 수행하면서 모든 단어들의 토픽 할당 분포가 변경되지 않고 수렴할 때까지 수행

6. 임베딩

1. 임베딩(Embedding)과 워드2벡터: Embedding은 자연어처리에서 단어를 수치화하여 표현하며 다양한 방식이 있고, 워드2벡터는 단어를 벡터로 바꾸는 대표적인 임베딩 모형
2. 워드2벡터의 원리: 단어를 벡터로 변환하는 알고리즘으로 인공신경망 언어모형을 따르면서도 학습 속도와 성능을 개선하며, CBOW와 Skip Gram등의 기법이 있음
3. **Word2Vec에서 Doc2Vec으로: 단어 임베딩 모형인 Word2Vec에 이어 발표된 Doc2Vec 모형은 Paragraph2Vec이라고도 지칭되며 문장/단락/문서에 대한 임베딩을 할 수 있음**
4. **Doc2Vec 원리: 다음 단어를 예측하며 로그 확률 평균을 최대화하고 문장/단락/문서의 임베딩 표현을 수행함. PV(Paragraph Vector)-DM(Distributed Memory)과 PV(Paragraph Vector)- DBOW(Distributed Bag of Words)의 두 가지 방식이 있음**

6. 임베딩

임베딩?

- 자연어처리에서 많이 활용
- 사람이 사용하는 자연어의 단어를 수치화하여 벡터로 표현한 결과가 그 과정
- 다양한 방법이 사용

6. 임베딩

워드2벡터(Word2Vec)

- 워드2벡터는 단어를 벡터로 바꾸는 대표적인 임베딩 모형
- 단어를 벡터로 변환하는 알고리즘으로 인공신경망 언어모형을 따르면서 학습 속도와 성능을 개선
- 단어 유사도 계산/ 단어와 특정 쿼리의 유사도 계산 / 문장 분류

6. 임베딩

워드2벡터(Word2Vec)의 대표적인 알고리즘: CBOW

Continuous Bag of Words

주변 단어(Context Word)로 중심의 단어(Center Word)를 예측

This cat jumps onto the chair

입력 출력

Context word Center word

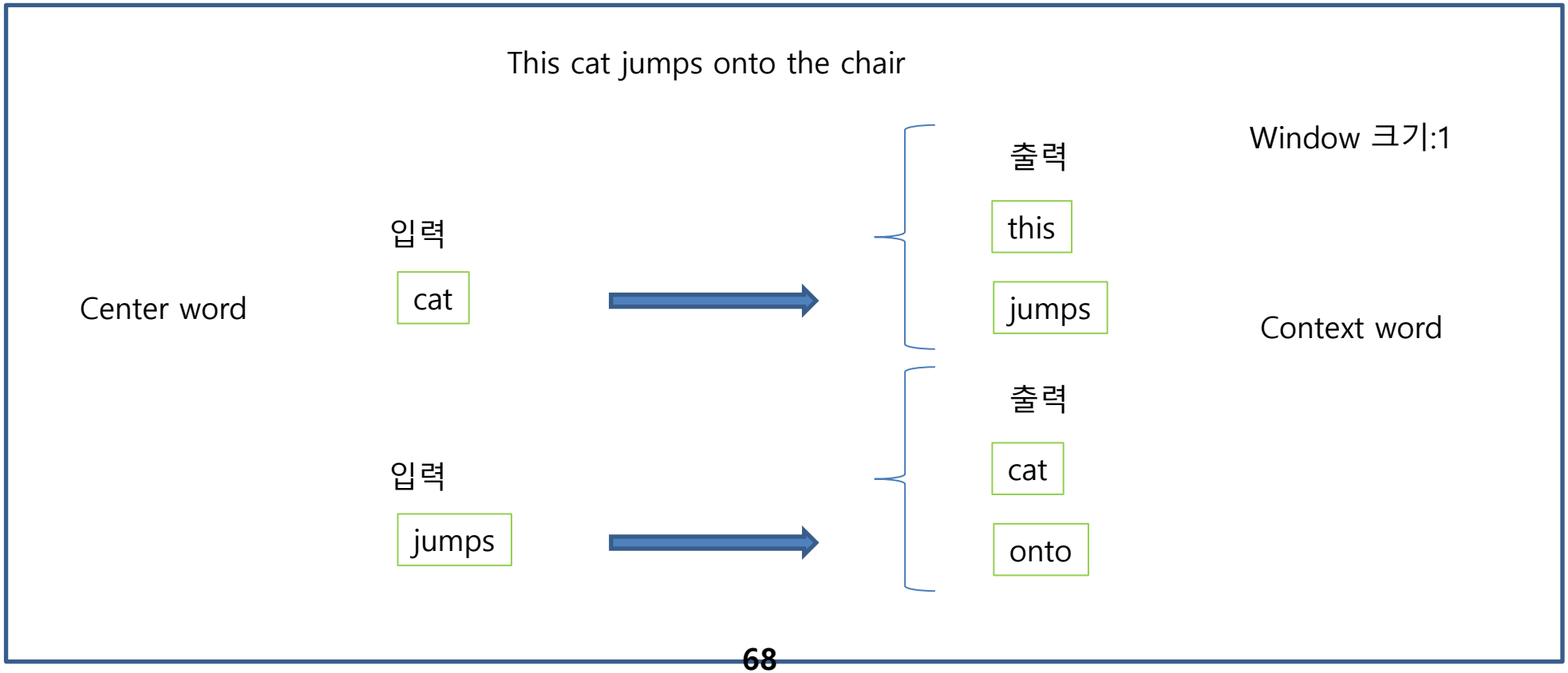
- Window: Center word 기준으로 앞 뒤 몇 개의 단어를 참고하는지
- Sliding Window: 중심 단어를 바꿔가며 Window 적용해서 학습데이터 생성

6. 임베딩

워드2벡터(Word2Vec)의 대표적인 알고리즘: Skip Gram

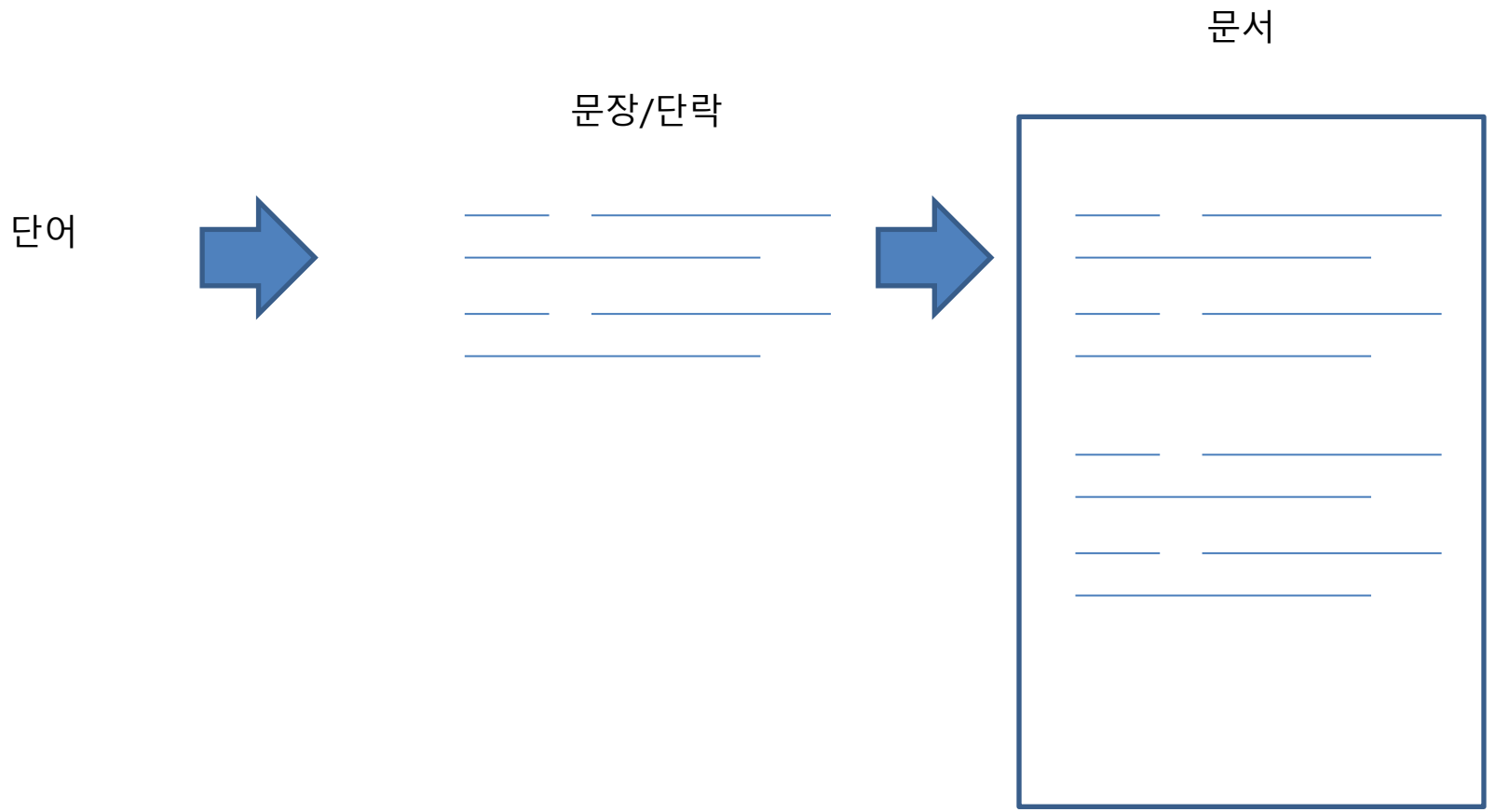
Skip Gram

중심 단어(Center Word)로 주변 단어(Context Word)를 예측



6. 임베딩

Word2Vec에서 Doc2Vec으로



6. 임베딩

Doc2Vec이란?

- *Word2Vec에 이어 2014년 구글에서 개발한 모형*
- *다음 단어를 예측하며 로그 확률 평균을 최대화하고 문장/단락/문서의 임베딩 표현을 수행함.*

6. 임베딩

PV(Paragraph Vector)-DM(Distributed Memory), 성능 우수!

Paragraph ID: para_1

Sentence: The dog sleep on the sofa

Y: Target 단어

X: Target 이전의 k개 단어+Paragraph ID

D2v는 단락에서 단어를 예측하며, 로그확률평균을 최대화하는 학습

예: k=2

X		Y
[para_1, the, dog]		sleep
[para_1, dog, sleep]	on	
[para_1, sleep, on]	the	
[para_1, on, the]		sofa

PV(Paragraph Vector)- DBOW(Distributed Bag of Words)

Paragraph ID: para_1

Sentence: The dog sleep on the sofa

Paragraph ID가 입력, 문장의 단어들이 Target

X	Y
[para_1]	the
[para_1]	dog
[para_1]	sleep
[para_1]	on
[para_1]	the
[para_1]	sofa

6. 임베딩

Doc2Vec의 활용

- Q&A에 활용
- 기계번역에 활용
- 문장 주제 찾기 및 분류에 활용
- 사람과의 대화에 활용
- 다양한 파생모형!

6. 임베딩

다양한 임베딩 모형

Item2Vec

Topic Embedding

LDA2Vec **Graph Embedding**

Sentiment Embedding, Word2Sent

QnA