

3. 텍스트 분석 & NLP - I

텍스트분석 & NLP 개요

텍스트분석 개요

NLP 개요

솔루션 및 알고리즘 소개 - STT, TTS 등

<https://www.bigkinds.or.kr/>

1. 빅데이터 분석 개요

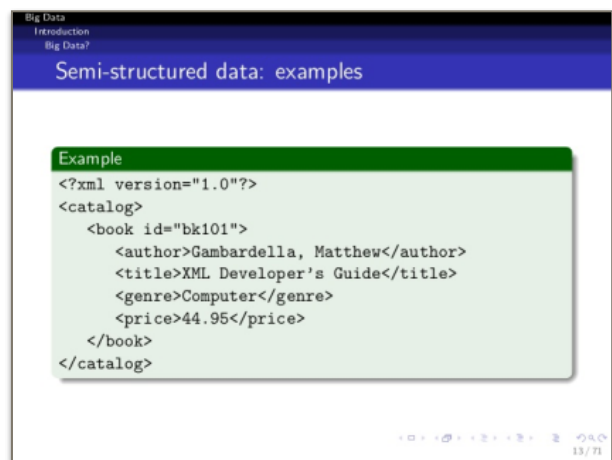
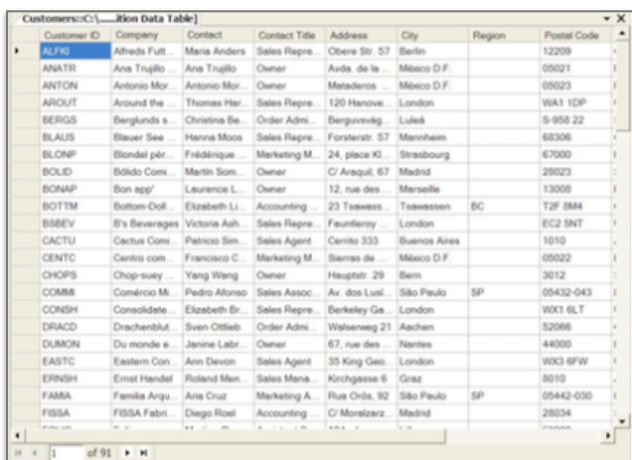
(1) 빅데이터 정의와 종류

- 정의 :

- 기존 데이터베이스 관리 도구로 데이터를 수집, 저장, 관리, 분석할 수 있는 역량을 넘어서는 대량의 정형 또는 비정형 데이터 집합
- 큰 규모를 활용해 더 작은 규모에서는 불가능했던 새로운 통찰이나 새로운 형태의 가치를 추출해 내는 일
- 데이터의 사이즈 : Tera -> Peta -> Exa -> Zetta -> Yotta -> Zona -> Weka

- 데이터의 종류

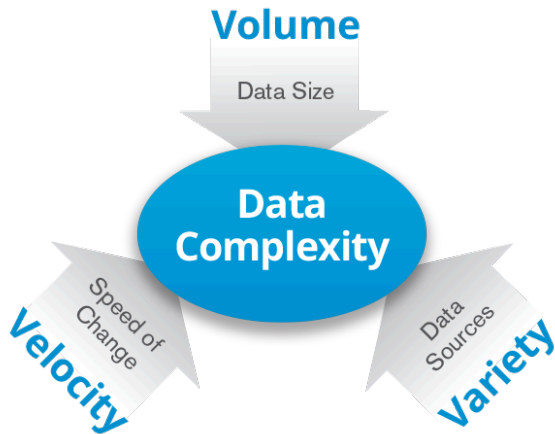
종류	설명
정형 (structured)	고정된 필드에 저장된 데이터. 관계형 데이터베이스 및 스프레드시트 등을 예로 들 수 있다.
반정형 (semi-structured)	고정된 필드에 저장되어 있지는 않지만, 메타데이터나 스키마 등을 포함하는 데이터. XML이나 HTML 텍스트 등을 예로 들 수 있다.
비정형 (unstructured)	고정된 필드에 저장되어 있지 않은 데이터. 텍스트 분석이 가능한 텍스트 문서 및 이미지/동영상/음성 데이터 등을 예로 들 수 있다.



```
<?xml version="1.0"?>
<catalog>
  <book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
  </book>
</catalog>
```

(2) 빅데이터의 특징 : 3V

- 양(Volume) : 데이터 규모가 페타바이트 수준으로 방대
- 속도(Velocity) : 생성 주기가 짧고 급속도로 생산
- 다양성(Variety) : 수치, 문자, 영상 등 다양한 형태

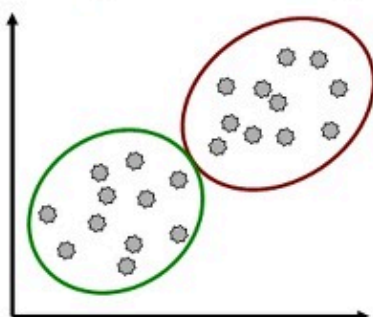


(3) 빅데이터 분석기법

- Text classification (텍스트 분류) : 분류체계로 나누기
- Text clustering (텍스트 군집) : 그룹으로 묶기
- Text summarization (요약)
- Sentiment analysis (정서 분석. 긍정/부정) : <https://cloud.google.com/natural-language/>
- Emotion analysis (감정 분석, Joy, Anger, Disgust 등) : <https://natural-language-understanding-demo.mybluemix.net/>
- Entity extraction and recognition (엔티티 추출 및 인식) : 사람, 회사, 장소, 통화, 숫자 등
- Similarity analysis and relation modeling (유사도 분석 및 관계 모델링)

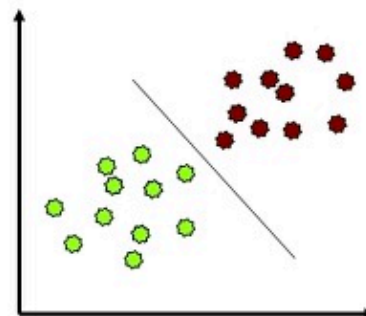
CLUSTERING

- Data is not labeled
- Group points that are "close" to each other
- Identify structure or patterns in data
- Unsupervised learning



CLASSIFICATION

- Labeled data points
- Want a "rule" that assigns labels to new points
- Supervised learning



언어 처리 (Natural Language Processing, NLP)

(1) 자연어

- 정의
 - 컴퓨터 언어처럼 인공적으로 만들어진게 아니라 사람들이 사용하면서 개발하고 진화해온 것
- 종류
 - 영어, 중국어, 한국어 등
 - 자연어가 소통되는 형태 : 말하기, 글쓰기, 기호(sign)
 - 텍스트 데이터는 비구조화 데이터로 특정 언어에 속하며, 특정 구문과 의미를 따름
- 특징
 - 형태론적 다양성 : 첨가어(다수의 형태소 결합), 굴절어(어간 변화), 스와히리어(수가 문두에 붙음), 아랍어(모음이 시제)
 - 통사적 다양성 : Postfix언어(동사가 문장 뒤), Infix 언어(동사가 문장 중간), Prefix 언어(동사가 문장 처음)

(2) 자연어 처리

- 정의 :
 - 사람이 사용하는 자연어를 기계가 이해할 수 있는 형태로 변환하는 기술. 혹은 다시 인간이 이해할 수 있는 언어로 변환
 - 컴퓨터과학, 인공지능, 언어학이 합쳐진 분야
- 종류 :

항목	내용
Machine Translation (기계 번역)	<ul style="list-style-type: none">- 문법/구문 구조보다는 통계/말뭉치(코퍼스) 활용- ex. Google 번역
Speech Recognition Systems (음성 인식 시스템)	<ul style="list-style-type: none">- NLP에서 가장 어려운 영역- 음성합성/분석/구문분석/추론 등의 기술 발전으로 향상됨
Question Answering Systems (질의 응답 시스템)	<ul style="list-style-type: none">- 질문응답시스템- 다양한 도메인의 거대한 지식기반 필요
Contextual Recognition and Resolution (문맥 인식 및 해결)	<ul style="list-style-type: none">- 주어진 문장에서 단어의 문맥적 의미 파악- Coreference 솔루션 이용
Text Summarization (텍스트 요약)	<ul style="list-style-type: none">- 텍스트 요약 시스템- 일반요약과 쿼리기반 요약(질문에 대한 답만 요약)으로 구분
Text Categorization (텍스트 카테고리화)	<ul style="list-style-type: none">- 문서 내용을 기반으로 카테고리나 분류체계 식별- NLP 및 머신러닝에서 가장 많이 사용

(출처: Dipanjan Sarkar. 'Text Analytics with Python.' iBooks)

- 자연어 처리 단계
 - 1. 형태소 분석 (Morphological Analysis) : 입력된문자열을분석하여 형태소(morpheme)라는 최소 의미 단위로 분리
 - 2. 구문 분석 (Syntax Analysis) : 문법을 이용하여 문장의 구조를 찾아내기
 - 3. 의미 분석(Semantic Analysis) : 통사 분석 결과에 해석을 가하여 문장이 가진 의미를 분석
 - 4. 화용 분석(Pragmatic Analysis) : 문장이 실세계와 갖는 연관관계 분석 (문법은 맞지만 현재 활용이 불가능할수 있음)

3. 언어데이터의 활용

(1) 언어데이터 소스와 분석 가능성

웹데이터

소셜네트워크 데이터

말뭉치(Corpus) : 세종말뭉치 등

딥러닝(Deep learning) 알고리즘 사용

발성된 단어 이해 -> 음성 단어 텍스트화 -> 텍스트의 내용, 의미, 감성 분석

(2) 텍스트 코퍼스

- 쓰거나 말해진 텍스트 데이터를 방대한 양으로 구조화하여 디지털 상태로 모은 것.
- 코퍼스의 용도 : 언어학, 통계적 분석, NLP툴
- 코퍼스는 1950년대에 처음 언어의 특성과 구조를 분석하기 위해 만들어져서, 통계적 양적 방법으로 분석이 수행됨. 통계분석할 만한 대량의 데이터를 모으지 못함. 인지 학습과 행동과학에 초점을 두고 노암 촘스키가 고도화된 rule-based 언어모델을 개발할 목적으로 분석함
- 말뭉치 주석(Corpora Annotation)
 - 텍스트 코퍼스에 풍성하게 메타데이터로 주석을 달게 되면 NLP 처리에 매우 유용함

주석의 종류

- * POS tagging
- * Word stems
- * Word lemmas
- * Dependency grammars
- * Constituency grammars
- * Semantic types and roles

<class>경제 / 근로자복지 / 근로자복지정책및운영 / 근로자교육지원 / 기능대회및기능인지원</class>

<title>제51회 전국기능경기대회 관련 열린경기장 사용 계획서 제출 요청</title>

<contents>

1. 서부공원복지사업소 공원운영과-11110(2016.7.15.)호와 관련입니다. 2. 서부공원복지사업소「일시적 장소사용 운영지침」에 의거 대회 개최에 따른 요구사항을 포함한 최종 계획서를 2016. 7. 19.(화)까지 제출하여 주시기 바랍니다. 가. 요구사항 끝.

</contents>

<origin>일자리정책과</origin>

<retention>5년</retention>

<tag>기능대회및기능인지원</tag>

<relation>민간위탁 사업 계약심사 결과 알림(전국기능경기대회 참가 지원),전광판 등 영상매체 표출 요청(제51회 전국기능경기대회),민간위탁 계약심사(전국기능경기대회 참가 지원) 보완자료 제출,민간위탁 계약심사(전국기능경기대회 참가 지원) 보완요청,제51회 전국기능경기대회 홍보부스 참가 신청 안내,급수차 지원 요청(제51회 전국기능경기대회),홍보물 매체배정 요청(전국기능경기대회),민간위탁 계약 심사요청(전국기능경기대회 참가 지원),기능경기대회 관련 조치사항 송부,사용일자 변경 요청(장충체육관),2016년 제51회 서울특별시 전국기능경기대회 어플리케이션 공모전 홍보 요청,전광판 등 영상매체 표출 요청(제51회 전국기능경기대회)</relation>

<class>여성가족 / 청소년정책 / 청소년관련정책수립 / 청소년보호지원 / 청소년보호및지원사업운영</class>

<title>청소년유해매체물(간행물) 목록표(여성가족부 고시 제2017-27호) 통보</title>

<contents>

1. 청소년보호환경과-1845(2017.05.29.)호와 관련입니다. 2. 간행물윤리위원회가 청소년보호법 제7조제1항에 따라 청소년유해매체물로 심의·결정하여 고시 요청한 간행물을 동법 제21조제2항에 따라 여성가족부가 관보에 고시하고 불임으로 통보하오니 업무에 참고하시기 바라며, 산하기관(도서관 포함) 및 회원단체 등에도 통보하여 주시기 바랍니다. 행정처분 및 형사처분에 관한 사항은 청소년보호법을 참조하시기 바랍니다. 끝.

</contents>

<origin>청소년담당관</origin>

<retention>10년</retention>

<tag>청소년보호및지원사업운영,유해환경감시단운영,간행물,여성가족부,청소년유해매체물,청소년</tag>

<relation>「청소년 보호법 시행령」,일부개정 알림,청소년유해매체물(간행물) 목록표(여성가족부 고시 제2017-32호) 통보, (국회의원 요구자료) 「생활불편신고,오피 신고 및 처리현황 자료 요청,청소년유해매체물(방송물) 목록표(여성가족부 고시 제2017-31호) 통보,청소년유해매체물(간행물) 목록표(여성가족부 고시 제2017-30호) 통보,2017년 청소년의달 청소년유해환경 활동점검·단속 추진실적 보고서 제출,2017년 여름휴가철 피서지 청소년유해환경 개선활동 계획 수립·시행,2017년 여름방학 대비 청소년근로보호 합동점검 계획안 제출,청소년의 달 청소년유해환경 개선활동 추진계획 송부 및 협조요청,청소년유해매체물(음반 및 음악파일) 목록표(여성가족부 고시 제2017-26호) 통보,청소년 보호법 관련 업무모음 전파(6월),청소년유해환경감시단 전문인력 경력증명서 발급</relation>

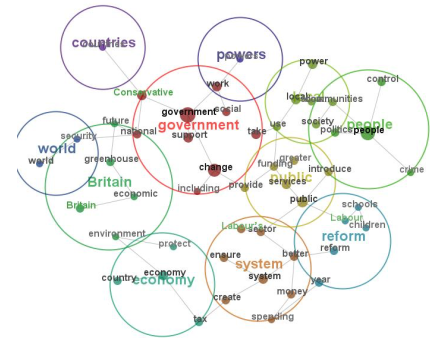
(2) 분석 방법별 결과

정형데이터 분석 : 통계학, 데이터마이닝, 기계학습 알고리즘 사용

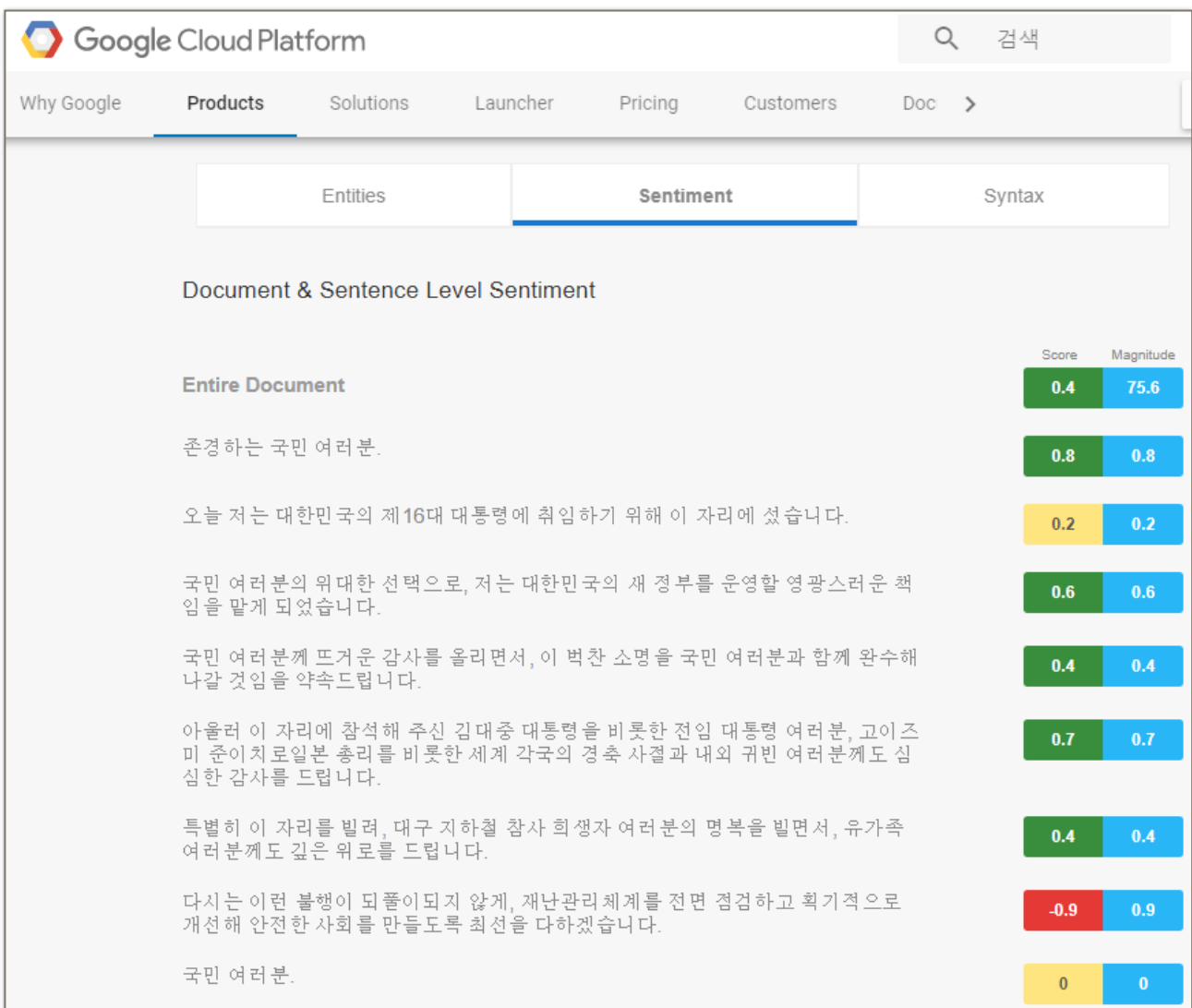
비정형데이터 분석 : 시맨틱 네트워크 분석, 감성분석, 군집분석 사용

시맨틱 네트워크 분석 : 텍스트에 사용된 핵심 단어(concept)를 추출하고, 단어 사이의 의미론적 연관을 살펴보고 이를 시각화

시맨틱 네트워크 분석 절차 : 텍스트 데이터 수집 -> 메시지 내 주요 단어 빈도 분석 -> 핵심 키워드 도출 -> 네트워크 다이어그램 설계 -> 노드들 간의 링크 표시(링크의 연결정도와 밀도) -> 중심성(centrality) 분석



정서분석(Sentiment analysis) : 텍스트를 긍정, 부정, 중립으로 판별하여 선호도 측정하는 기법



감정 분석(Emotion analysis) : 화(Anger), 혐오(Disgust), 공포(Fear), 기쁨(Joy), 슬픔(Sadness)

Entities

Extracts people, companies, organizations, cities, geographic features, and other entities from your content, and optionally detects the sentiment of each entity.

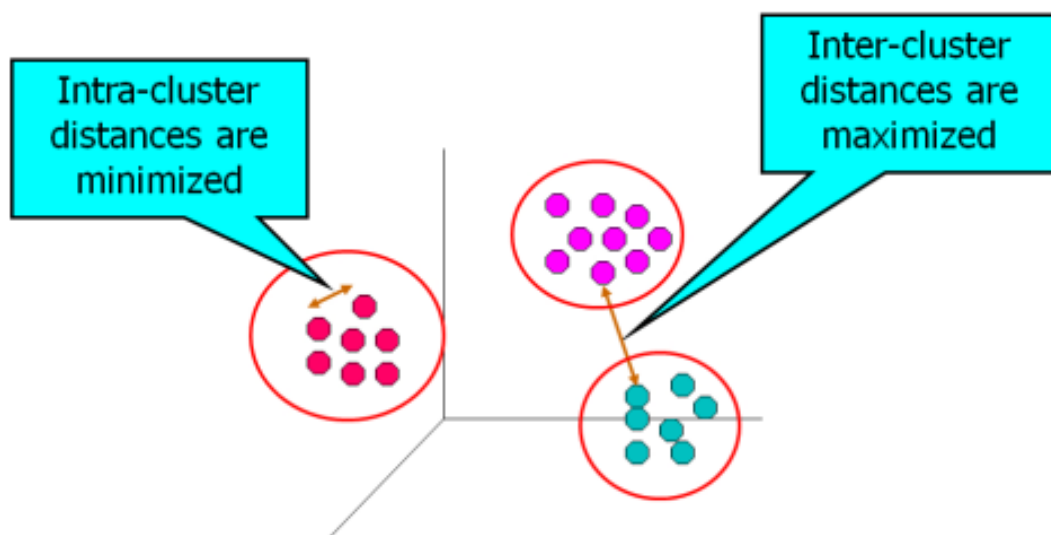
[View JSON](#)

Entity	Relevance	Sentiment	Type
Elliot Turner	0.89066	negative	Person

Emotion	Score
Anger	0.085606
Disgust	0.080078
Fear	0.084827
Joy	0.174876
Sadness	0.221841

Entity	Relevance	Sentiment	Type
PR Newswire	0.252507	neutral	PrintMedia

군집분석(Cluster Analysis) : 비슷한 특성을 가진 개체를 합쳐 최종적으로 유사 특성의 집단을 발굴하는데 사용



(3) 머신러닝 기반 자연어 처리 기법

- 자연언어처리에 이용되는 지식을 자동으로 학습
- 통계적/경험적 인공지능 기법

(4) 딥러닝 기반 자연어 처리 기법

데이터로부터 특징을 자동적으로 학습

종래보다 폭넓은 문맥정보를 다룸

모델에 최적인 출력을 다루기가 간단

특성이 다른 사진, 음성 등과 같은 모델들 간의 친화성이 높게 되어 멀티모달 모델 구축이 쉬워짐

(출처: <https://www.slideshare.net/ssuser06e0c5/ss-64417928>, 7쪽)

Ngram

: 입력한 문자열을 N 개의 단위로 절단

Ngram

- ▶ 입력한 문자열을 N개의 단위로 절단

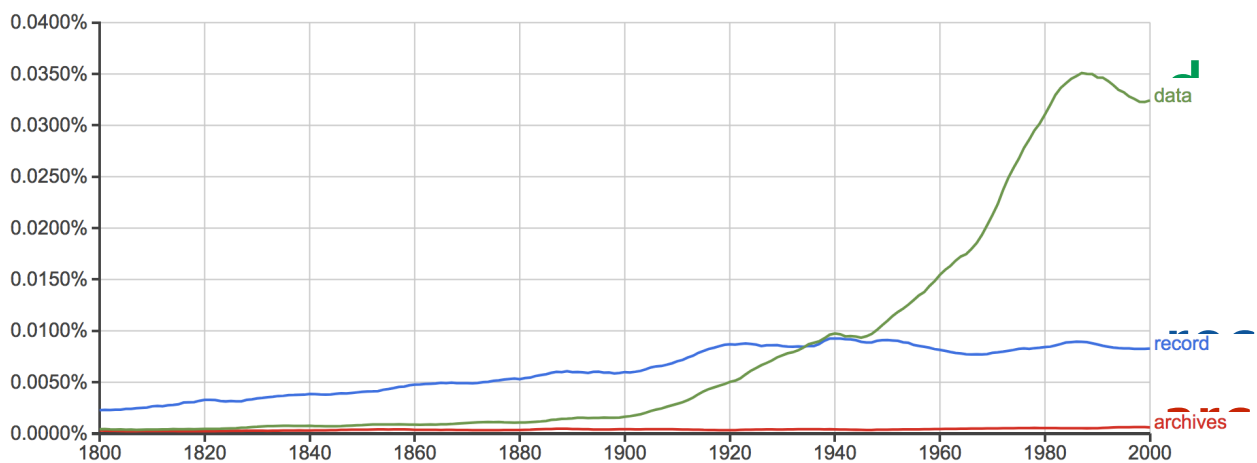
Here is a dog

2-gram (단어)

“Here is” “is a” “a dog”

3-gram (문자)

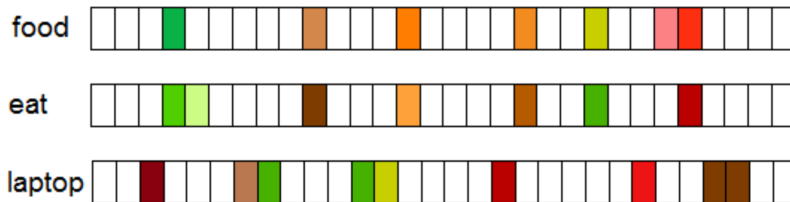
“Her” “ere” “re_” “e_i” “is_” “_a_” “a_d” “_do” “dog”



구글 ngram 사용해 보기 : <https://books.google.com/ngrams>

BOW (Back Of Words) 모델

- 통계 기반의 자연어처리 기법(Statistical NLP)은 복잡한 자연어를 모델링하는 데 기본 옵션으로 부상
- 문서에서 사용된 모든 단어가 포함된 어휘 목록을 만들고, 문서별 단어 빈도 정보가 포함된 단어 행렬 생성
- 분산표상으로 벡터 표현(Distributional vectors) 또는 단어 임베딩(Word Embedding)
- 이 가정은 비슷한 의미를 지닌 단어는 비슷한 문맥에 등장하는 경향이 있을 것이라는 내용이 핵심
- 따라서 이 벡터들은 이웃한 단어의 특징을 잡아내고자 함
- 분산표상 벡터의 주된 장점은 이 벡터들이 단어 간 유사성을 내포하고 있다는 점
- 코사인 유사도 같은 지표를 사용함으로써 벡터간 유사성을 측정할 수 있다.



Bag of Words Model

“Represent each document which the bag of words it contains”

d1 : Mary loves Movies, Cinema and Art

Class 1 : Arts

d2 : John went to the Football game

Class 2 : Sports

d3 : Robert went for the Movie Delicatessen

Class : Arts

	Mary	Loves	Movies	Cinema	Art	John	Went	to	the	Delicatessen	Robert	Football	Game	and	for
d1	1	1	1	1	1									1	
d2						1	1	1	1			1	1		
d3			1				1		1		1				1

Bag of Words Model + Weighting eiFeatures

Weighting features example TF-IDF

- TF-IDF \approx Term Frequency / Document frequency
- Motivation : Words appearing in large number of documents are not significant

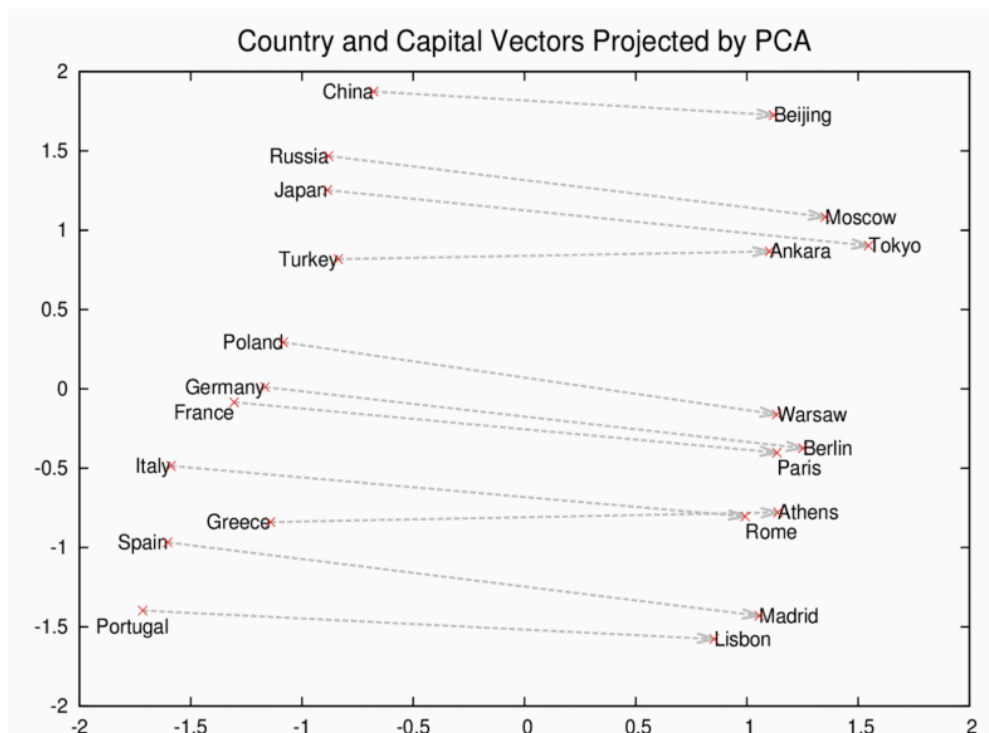
	Mary	Loves	Movies	Cinema	Art	John	Went	to	the	Delicatessen	Robert	Football	Game	and	for
d1	0.3779	0.3779	0.3779	0.3779	0.3779									0.0001	
d2							0.4402	0.001	0.02			0.4558	0.458		
d3			0.001				0.01		0.01		0.458				0.0001

Word2Vec (Word to Vector) : 워드투벡터

- 정의
 - 텍스트를 처리하는 인공 신경망
 - 말뭉치(corpus)말뭉치의 단어를 벡터로 표현
 - 딥러닝을 적용한 NLP
- word embeddings
 - 단어의 의미와 맥락을 고려하여 단어를 벡터로 표현
 - 목적 : 유사한 단어일 수록 가까운 거리에 위치하도록 각 단어에 해당하는 벡터 값을 찾는 것 (말뭉치 데이터만을 사용)
- 특징
 - 데이터의 양이 충분하면 Word2vec은 단어의 의미를 꽤 정확하게 파악
 - 이를 이용하면 단어의 뜻 뿐만 아니라 여러 단어의 관계를 파악 가능
- 원리
 - 어떤 단어의 출현확률은 이전 (n-1)개의 단어에 의존한다. n=4의 경우
...man stood **still** as they **slowly** **walked** through the... 조건 <—이것을 n-1차 마코프 과정이라고 함.

Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408

<스웨덴과 가장 거리가 가까운(유사한) 단어>



CNN (Convolutional Neural Network) : 콘볼루션 신경망

- 워드 임베딩이 인기를 끌고 그 성능 또한 검증된 이후, 단어 결합이나 n-gram으로부터 높은 수준의 피처를 추출해내는 효율적인 함수의 필요성이 증대됐다.
- 이러한 추상화된 피처들은 감성분석, 요약, 기계번역, 질의응답(QA) 같은 다양한 NLP 문제에 사용될 수 있다.
- 문장 모델링에서 CNN은 multi-task learning을 사용하며 품사태깅, 청킹, 개체명인식, 의미역결정, 의미적으로 유사한 단어 찾기, 랭귀지모델 같은 NLP 과제를 수행한다.
- 참조테이블(look up table)은 각 단어를 사용자가 정의한 차원의 벡터로 변형해 사용된다.

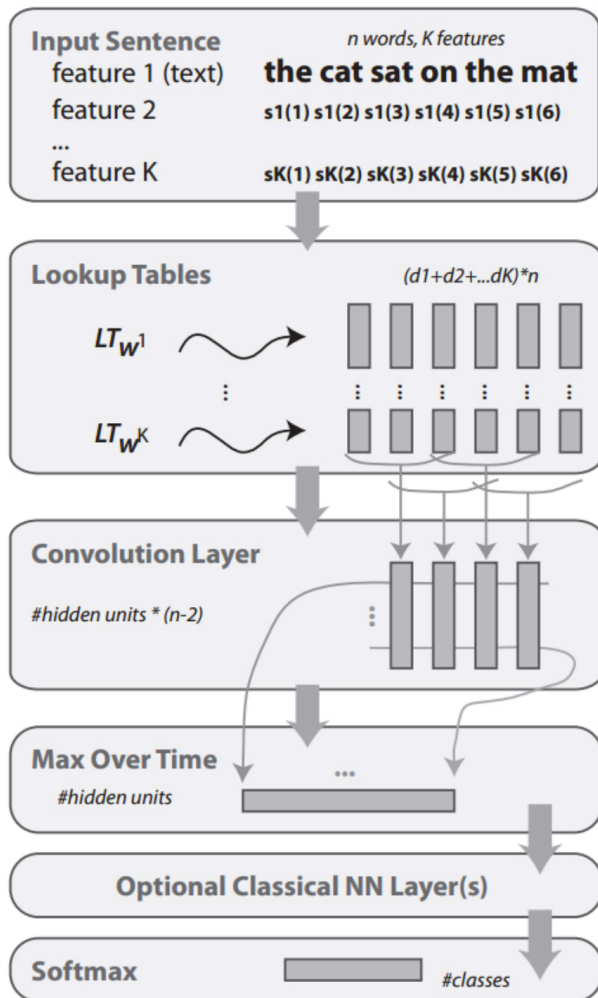
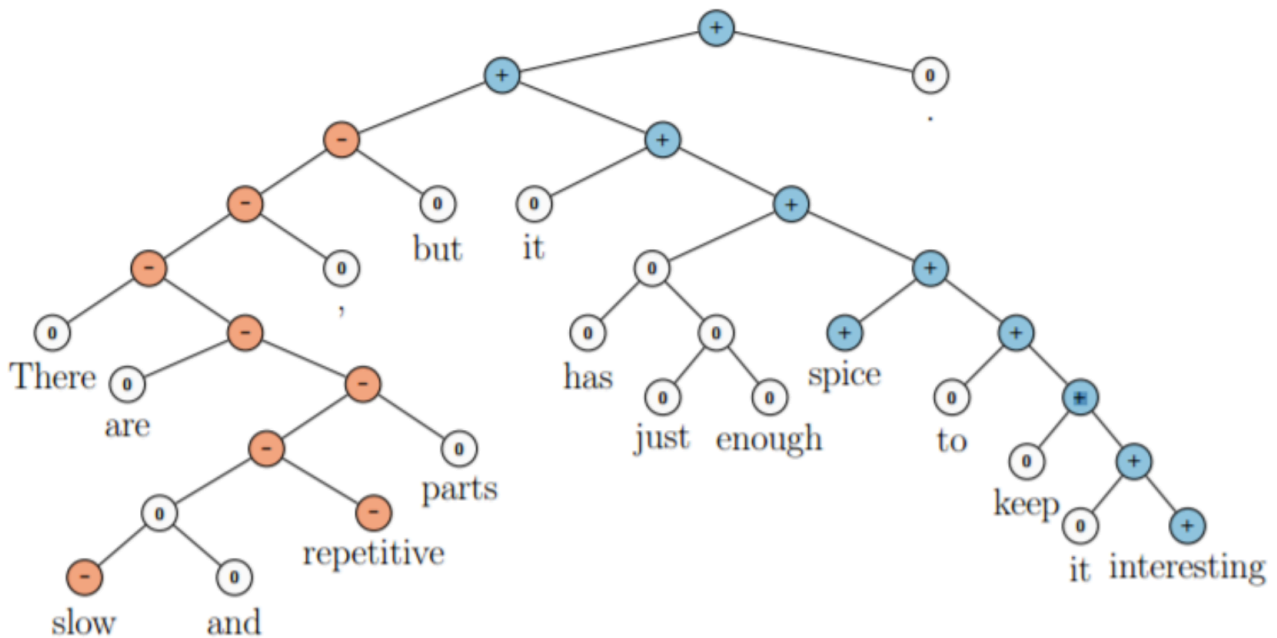


그림5: *Colobert and Weston(2008)*이 제안한 CNN 프레임워크. 그들은 단어 범주 예측에 이 모델을 사용했다

참고 : <https://deeplearning4j.org/kr/convolutionnets>

RNN (Recurrent Neural Networks) : 순환 신경망

- Recurrent Neural Network는 시퀀스(순차적인 데이터) 모델링에 강점을 지닌 기법이다.
- 그러나 자연언어는 단어와 단어가 계층적인 방식으로 구(phrase)로 결합되는 재귀적인(recursive) 구조를 나타낸다.
- 이러한 구조는 문장 구성성분 분석 트리(constituency parsing tree)로 표현될 수 있다.
- 이 때문에 문장의 문법적 구조 해석을 보다 용이하게 하기 위해 트리 구조 모델이 사용되었다(Tai et al., 2015).
- 특히 Recursive Neural Network에서 각 노드는 자식 노드의 표현(representation)에 의해 결정된다
- Socher et al. (2013)은 구문(phrase), 문장 수준 감성 예측에 이 모델을 적용했다.
- 저자는 이 연구에서 파싱 트리의 모든 노드(단어)에 대해 감성 스코어를 시각화했다.
- 여기에선 전체 문장의 감성에 큰 영향을 미치는 'not' 또는 'but' 같은 단어에 대한 모델의 민감도가 나타난다.



(출처: <https://ratsgo.github.io/natural%20language%20processing/2017/08/16/deepNLP/>)

(5) 언어학의 세부분야

언어학의 기원은 기원전 4세기 인도 학자이자 언어학자인 Panini가 산스크리트어를 설명한 것이라 함. 언어학이 학문으로 정 의된 것은 1847년

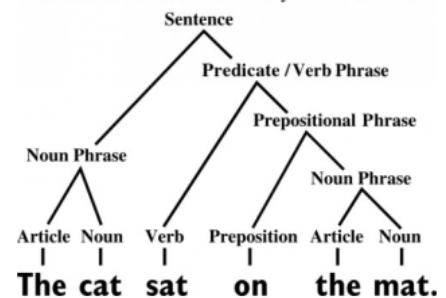
언어학의 10가지 분야

- 1) 음성학 Phonetics 사람 발성의 음향학적 특성
- 2) 음운론 Phonology 소리의 패턴, 액센트, 톤, 음절구조
- 3) 구문 Syntax 문장, 구절, 단어, 그들의 구조, 순서
- 4) 의미론 Semantics. 어휘적 의미론 Lexical semantics/구성적 의미론 Compositional semantics
- 5) 형태학 Morphology 단어, 접두어, 접미어 등 구별되는 형태소 morpheme 들이 갖는 의미와 결합되는 규칙
- 6) 사전 Lexicon 단어와 구절들의 특성, 언어의 어휘를 만들어내는 방식
- 7) 화용론 Pragmatics 문맥이나 시나리오같은 언어적 비언어적 요소가 의미에 미치는 영향
- 8) 담론분석 Discourse analysis 대화분석
- 9) 스타일리즘 Stylistics 음색, 액센트, 대화, 문법, 음성유형 등 작문스타일 연구
- 10) 기호학 Semiotics 기호와 부호과정, 의미전달 방법. 유추, 비유, 상징주의

(6) 언어의 구조와 문법

- 문장 => 절 => 구 => 단어 => 형태소
- 문법(Grammar)
 - 문법은 구문이나 구조를 만드는데 필요함.
 - 구문과 구조를 재현하는 방식에 따라 dependency grammar, constituency grammar 로 나뉨

Basic constituent structure analysis of a sentence:



(7) 언어의 의미

- 어휘 의미 관계(Lexical Semantic Relations)는 어휘단위들 간 의미 관계를 밝힘.
- 표제어와 단어의 형태(Lemmas and Wordforms) : lexeme {eating, ate, eats}가 wordform들을 포괄하고 있고, 이 중에서 lemma는 eat
- 단어의 의미(wordsense) : 단어는 위치, 연관단어들, 문맥에 따라 의미가 달라짐
 - 예) I go to the annual fair. I hope the judgement is fair.
 - 구분하기 어려운 단어들이 많으며 이런 단어들이 NLP 처리과정에서 문제를 일으킴. 실제 문맥과 의미를 알아내는 것이 쉽지 않음
- 의미 네트워크와 모델(Semantic Networks and Models) : 지식과 개념을 표현. semantic network의 기본단위는 entity, concept. concept은 추상적인 것과 구체적인 것 모두 포괄. concept 간의 관계 : is-a, has-a, part-of, related-to

4. STT, TTS, Parser 솔루션 사용해보기

(1) IBM Watson Speech To Text

<https://www.ibm.com/watson/services/speech-to-text/>

(2) Google SpeechRecognition

<https://cloud.google.com/speech/>

<https://chrome.google.com/webstore/search/TTS>

(3) TTS Demo

<http://www.readspeaker.com/voice-demo/>

<http://text-to-speech.imtranslator.net/speech.asp?dir=ko>

<https://www.ispeech.org/text.to.speech>

(4) 구문분석기

<http://nlp.stanford.edu:8080/parser/index.jsp>

5. 자연어처리(NLP) 실습

(1) 파이썬이 설치된 곳에 가서 다음을 실행하세요

```
$ pip install nltk
```

(2) 코퍼스 데이터와 프로그램 패키지 nltk를 설치하세요

```
>>> import nltk
>>> nltk.download('all')
[nltk_data] Downloading collection 'all'
[nltk_data] |
[nltk_data] | Downloading package abc to
[nltk_data] | C:\Users\samsung\AppData\Roaming\nltk_data...
[nltk_data] | Unzipping corpora\abc.zip.
[nltk_data] | Downloading package alpino to
[nltk_data] | C:\Users\samsung\AppData\Roaming\nltk_data...
[nltk_data] | Unzipping corpora\alpino.zip.
[nltk_data] | Downloading package biocreative_ppi to
[nltk_data] | C:\Users\samsung\AppData\Roaming\nltk_data...
....
```

(시간이 걸려도 끝까지 기다릴 것)

텍스트 데이터 토큰화

토큰화 : 효율적인 텍스트 분석을 위해 텍스트를 단어나 문장과 같은 작은 조각으로 나누는 작업

토큰 : 나뉜 조각들

#패키지 호출

```
from nltk.tokenize import sent_tokenize, word_tokenize, WordPunctTokenizer
```

#입력테스트 정의

```
input_text = "Do you know how tokenization works? It's actually quite interesting! Let's analyze a couple of sentences and figure it out."
```

#입력 텍스트를 문장 토큰으로 나눔

```
print("\nSentence Tokenizer:")
print(sent_tokenize(input_text))
```

#결과

Sentence Tokenizer:

```
['Do you know how tokenization works?', 'It's actually quite interesting!', 'Let's analyze a couple of sentences and figure it out.']
```

#입력 텍스트를 단어 토큰으로 나눔

```
print("\nWord tokenizer : ")
print(word_tokenize(input_text))
```

#결과

Word tokenizer :

```
['Do', 'you', 'know', 'how', 'tokenization', 'works', '?', 'It', "'", 's', 'actually', 'quite', 'interesting', '!', 'Let', "'", 's', 'analyze', 'a', 'couple', 'of', 'sentences', 'and', 'figure', 'it', 'out', '.']
```

#Word punct tokenizer사용해 입력 텍스트를 단어 토큰으로 나눔

```
print("\nWord Punct Tokenizer : ")
print(WordPunctTokenizer().tokenize(input_text))
```

#결과

Word Punct Tokenizer :

```
['Do', 'you', 'know', 'how', 'tokenization', 'works', '?', 'It', "'", 's', 'actually', 'quite', 'interesting', '!', 'Let', "'", 's', 'analyze', 'a', 'couple', 'of', 'sentences', 'and', 'figure', 'it', 'out', '.']
```

※ Sentence Tokenizer 와 Word tokenizer의 차이

구두점이 다르게 작동

예) It's를 Sentence Tokenizer는 It, 's 로 구분함 Word tokenizer는 It, ', s로 구분

어간 추출을 통해 단어 기본형으로 변형

스테머(Stemmer)는 어간을 추출하기 위해 사용되는 방법

스테머의 목적은 서로 다른 형태로 표현된 단어에서 공통되는 어간을 추출해 기본형으로 사용하는 것

#패키지 불러옴

```
from nltk.stem.porter import PorterStemmer
from nltk.stem.lancaster import LancasterStemmer
from nltk.stem.snowball import SnowballStemmer
```

#사용할 단어 입력

```
input_words = ['writing', 'calves', 'be', 'branded', 'horse', 'randomize', 'possibly', 'provision', 'hospital', 'kept', 'scratchy',
'code']
```

#스테머 객체 생성

```
porter = PorterStemmer()
lancaster = LancasterStemmer()
snowball = SnowballStemmer('english')
```

#출력을 위한 문자열 포맷 정의

```
stemmer_names = ['PORTER', 'LANCASTER', 'SNOWBALL']
formatted_text = '{:>16}' * (len(stemmer_names)+1)
print('\n', formatted_text.format('INPUT WORD', *stemmer_names), '\n', '='*68)
```

#입력 단어별로 어간 추출해 출력

```
for word in input_words:
...   output=[word, porter.stem(word), lancaster.stem(word), snowball.stem(word)]
...   print(formatted_text.format(*output))
```

결과

INPUT WORD	PORTER	LANCASTER	SNOWBALL
writing	write	writ	write
calves	calv	calv	calv
be	be	be	be
branded	brand	brand	brand
horse	hors	hors	hors
randomize	random	random	random
possibly	possibl	poss	possibl
provision	provis	provid	provis
hospital	hospit	hospit	hospit
kept	kept	kept	kept
scratchy	scratchi	scratchy	scratchi
code	code	cod	code

표제화를 통해 단어를 기본형으로 변형하기

표제화란 : 단어를 기본형으로 줄이는 또 다른 방법 문제를 해결하기 위해 좀더 구조화된 접근 방식

#패키지 호출

```
from nltk.stem import WordNetLemmatizer
```

#사용할 단어 입력

```
input_words = ['writing', 'calves', 'be', 'branded', 'horse', 'randomize', 'possibly', 'provision', 'hospital', 'kept', 'scratchy', 'code']
```

#표제화(Lemmatizer) 객체 생성

```
lemmatizer = WordNetLemmatizer()
```

#출력을 위한 문자열 포맷 정의

```
lemmatizer_names = ['NOUN LEMMATIZER', 'VERB LEMMATIZER']
```

```
formatted_text = '{:>24}' * (len(lemmatizer_names) + 1)
```

```
print('\n', formatted_text.format('INPUT WORD', *lemmatizer_names), '\n', '='*75)
```

#단어별 명사, 동사 표제화 적용

```
for word in input_words:
```

```
    output = [word, lemmatizer.lemmatize(word, pos='n'),
```

```
              lemmatizer.lemmatize(word, pos='v')]
```

```
    print(formatted_text.format(*output))
```

결과

INPUT WORD	NOUN LEMMATIZER	VERB LEMMATIZER
------------	-----------------	-----------------

=====

writing	writing	write
calves	calf	calve
be	be	be
branded	branded	brand
horse	horse	horse
randomize	randomize	randomize
possibly	possibly	possibly
provision	provision	provision
hospital	hospital	hospital
kept	kept keep	
scratchy	scratchy	scratchy
code	code	code

텍스트 데이터를 단어 묶음으로 나누기

패키지 설치

```
pip install numpy
```

#패키지 호출

```
import numpy as np
```

```
from nltk.corpus import brown
```

#입력 텍스트를 단어 묶음으로 나눔

#단어 묶음별로 N개의 단어 포함

```
def chunker(input_data, N):
```

```
    input_words = input_data.split(' ')
```

```
    output = []
```

#매개변수에 따라 텍스트를 N 단어 묶음으로 변환하는 함수 정의 리스트 반환

```
    cur_chunk = []
```

```
    count = 0
```

```
    for word in input_words:
```

```
        cur_chunk.append(word)
```

```
        count += 1
```

```
    if count == N:
```

```
        output.append(' '.join(cur_chunk))
```

```
        count, cur_chunk = 0, []
```

```
    output.append(' '.join(cur_chunk))
```

```
    return output
```

#메인 함수 정의, 브라운 말뭉치를 입력데이터로 사용

#12,000개 단어를 읽어서 입력 데이터로 사용

```
if __name__ == '__main__':
```

```
    # 브라운 코퍼스에서 12,000개 단어를 읽어옴
```

```
    input_data = ' '.join(brown.words()[:12000])
```

#묶음에 포함될 단어 수정의

```
    chunk_size = 700
```

#입력 텍스트를 단어 묶음으로 나누고 결과 표시

```
    chunks = chunker(input_data, chunk_size)
```

```
    print("\nNumber of text chunks =", len(chunks), '\n')
```

```
    for i, chunk in enumerate(chunks):
```

```
        print('Chunk', i+1, '==>', chunk[:50])
```

백오브워드 모델 사용해 단어 빈도 추출하기

```
#패키지 호출
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import brown
from text_chunker import chunker

#브라운 말뭉치에서 데이터읽기
input_data = ' '.join(brown.words()[:5400])

# 단어 묶음에 포함될 단어수
chunk_size = 800

#텍스트를 단어 묶음으로 나눔
text_chunks = chunker(input_data, chunk_size)

# 사전 구조체로 변경

chunks = []
for count, chunk in enumerate(text_chunks):
    d = {'index': count, 'text': chunk}
    chunks.append(d)

#CountVectorizer함수를 사용해 단어별 빈도를 체크, 문서 단어 행렬 생성
#CountVectorizer함수의 첫 번째 매개변수는 최소 문서 빈도며 두 번째 매개변수는 최대 문서 빈도다. 최소 문서 빈도보다 빈
도가 적거나 최대 빈도수보다 높은 단어는 무시
# 문서 단어 행렬 만들기
count_vectorizer = CountVectorizer(min_df=7, max_df=20)
document_term_matrix = count_vectorizer.fit_transform([chunk['text'] for chunk in chunks])

# 어휘 목록 추출 후 화면에 출력
vocabulary = np.array(count_vectorizer.get_feature_names())
print("\nVocabulary:\n", vocabulary)

# 단어 묶음별 이름 붙이기
chunk_names = []
for i in range(len(text_chunks)):
    chunk_names.append('Chunk-' + str(i+1))

# 문서 단어 행렬 출력
print("\nDocument term matrix:")
```

```
formatted_text = '{:>12}' * (len(chunk_names) + 1)
print('\n', formatted_text.format('Word', *chunk_names), '\n')
for word, item in zip(vocabulary, document_term_matrix.T):
    # 'item' 은 희소 행렬 'csr_matrix'는 데이터 구조체
    output = [word] + [str(freq) for freq in item.data]
    print(formatted_text.format(*output))
```