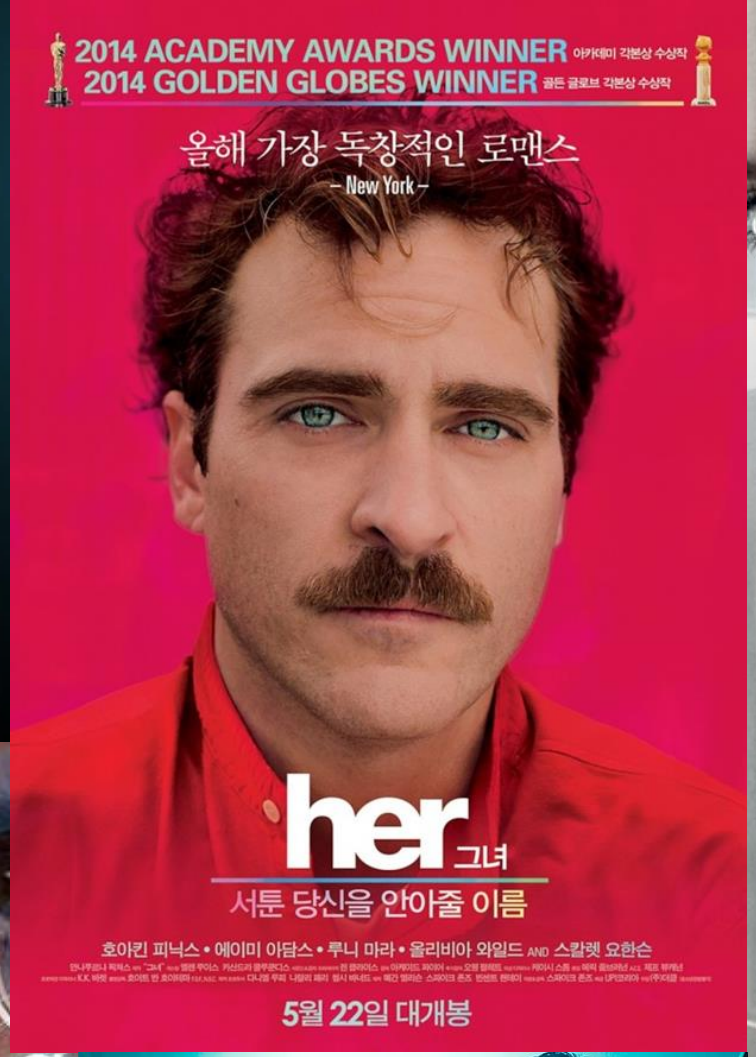
자연어처리(NLP)

박경규


J.A.R.V.I.S.


May
21
J.A.R.V.I.S
19:40
Tuesday
21 Tuesday
May 2013


2014 ACADEMY AWARDS WINNER 아카데미 각본상 수상작
2014 GOLDEN GLOBES WINNER 골든 글로브 각본상 수상작
올해 가장 독창적인 로맨스
– New York –
her 그녀
서툰 당신을 안아줄 이름
호아킨 피닉스 • 에이미 아담스 • 루니 마라 • 올리비아 와일드 AND 스칼렛 요한슨
5월 22일 대개봉


BLADE RUNNER


누구의 말도 믿지 마라
인간보다 매혹적인
엑스 마키나
FILM4 1월 21일, 전세계 최초개봉

자연어처리(Natural Language Processing)

https://www.aitimes.kr/news/articleView.html?idxno=15036

자동번역 (Machine Translation)

- 자동 번역의 가장 대표적인 사례로는 구글 번역(Google Translate)을 들 수 있습니다.
- 구글 번역에서는 문장 전체를 한 번에 번역하기 위해서 딥러닝 방식을 활용하고 있습니다.
- 이는 예제 기반의 자동 번역 방식을 사용하는데, 더 좋은 결과를 만들어 내기 위해서 수백개의 예제를 학습합니다.
- 하지만, 인간의 언어가 모호한 부분이 많기 때문에 시스템이 단어와 문장 그리고 의도를 이해하는 걸 어렵게 만들기도 합니다.
- 이때, 자연어 처리를 통해 문제점을 해결하는 데 도움을 받을 수 있습니다.

https://bit.ly/3t60zd1

대화형 사용자 인터페이스(CONVERSATIONAL USER INTERFACE)

- 대화형 사용자 인터페이스(CUI)는 컴퓨터가 실제 사람과의 대화를 모방하는 컴퓨터를 위한 인터페이스입니다.
- 대표적인 예로는 챗봇이 있습니다.

https://bit.ly/3t60zd1

텍스트 예측(TEXT PREDICTION)

- 텍스트 예측은 어떤 구문이나 문장에서 다음에 올 단어를 예측하는 프로세스입니다.
- 텍스트 예측에서 대표적인 사례는 바로 구글 검색으로, NLP 알고리즘인 버트(BERT) 모델을 사용하고 있습니다.

https://bit.ly/3t60zd1

감성 분석(SENTIMENT ANALYSIS)

- 감성 분석은 텍스트 데이터 안에서 감정을 해석하고 분류하는 프로세스입니다.
- 일반적으로 특정한 비즈니스와 관련된 애플리케이션을 사용하는 기업들이 감성 분석을 활용하면, 온라인 피드백에서 고객이 서비스, 브랜드, 제품에 대해서 보이는 정서를 (긍정적, 부정적, 중립적인지에 대해서) 파악할 수 있습니다.
- 제품 분석, 시장조사, 평판관리, 정밀한 타겟팅(Targeting), 시장 분석에 활용되고 있습니다.

https://bit.ly/3t60zd1

텍스트 분류(TEXT CLASSIFICATION)

- 이메일, 소셜 미디어, 웹사이트, 채팅 내용 등 일부 분야에서는 자연어 처리에 의한 텍스트 분류가 필수적입니다.
- 구글메일은 텍스트분류 기술로 기본(Primary), 소셜(Social), 프로모션(Promotion)으로 메일을 자동 분류하고 있습니다.
- 텍스트 분류가 뛰어난 능력을 발휘하는 또 다른 분야는 토론 포럼입니다. 이런 분야에서는, 어떤 댓글이 부적절하다고 표시가 될 필요가 있는지를 텍스트 분류 알고리즘을 통해서 판단합니다.

https://bit.ly/3t60zd1

맞춤법 검사(SPELL CHECK)

- 맞춤법 검사는 텍스트 안에서 잘못 표기된 철자나 오타가 있는지를 확인하고 수정해 기술입니다.
- 맞춤법 검사 프로그램의 대표적인 사례로 그래멀리(Grammarly)가 있습니다.

  사용자들이 내용을 계속해서 작성하는 동안 맞춤법 검사를 자동적으로 수행해서 수정안을 표시해 줍니다.

https://bit.ly/3t60zd1

음성인식(SPEECH RECOGNITION)

- 자연어 처리 기술이 발전하면서 목소리가 시스템에 입력하는 하나의 방식으로 받아 들여지고 있습니다.
  자연어 처리는 현재 음성 사용자 인터페이스(VUI)의 이면에 있는 핵심 기술입니다.
- 대표 사례는 애플 Siri, 구글 Assistance, 아마존 Alexa, KT 기가지니 등이 있습니다.
- 음성인식의 일반적인 또 하나의 사례로는 스마트폰에 있는 음성-텍스트 변환 기능이 (STT: Speech To Text) 있습니다.
- STT 기능을 이용하면 사용자들이 오디오를 통해서 입력을 할 수 있고, 그다음에는 텍스트로 변환이 됩니다.

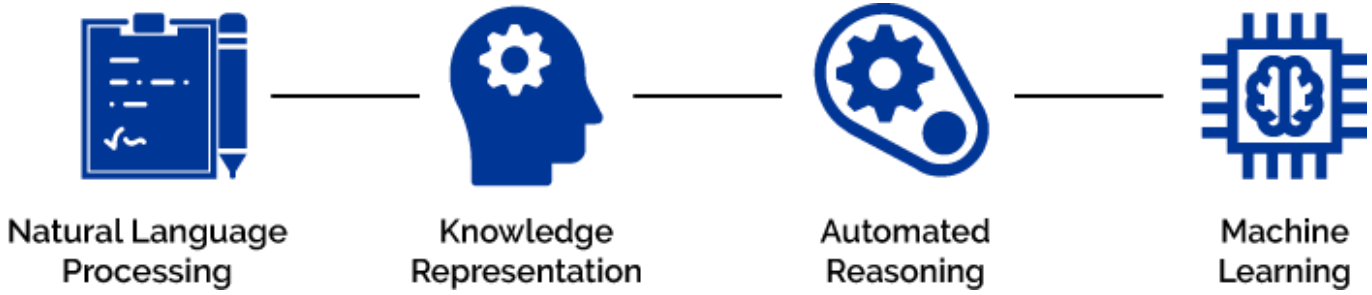
https://bit.ly/3t60zd1

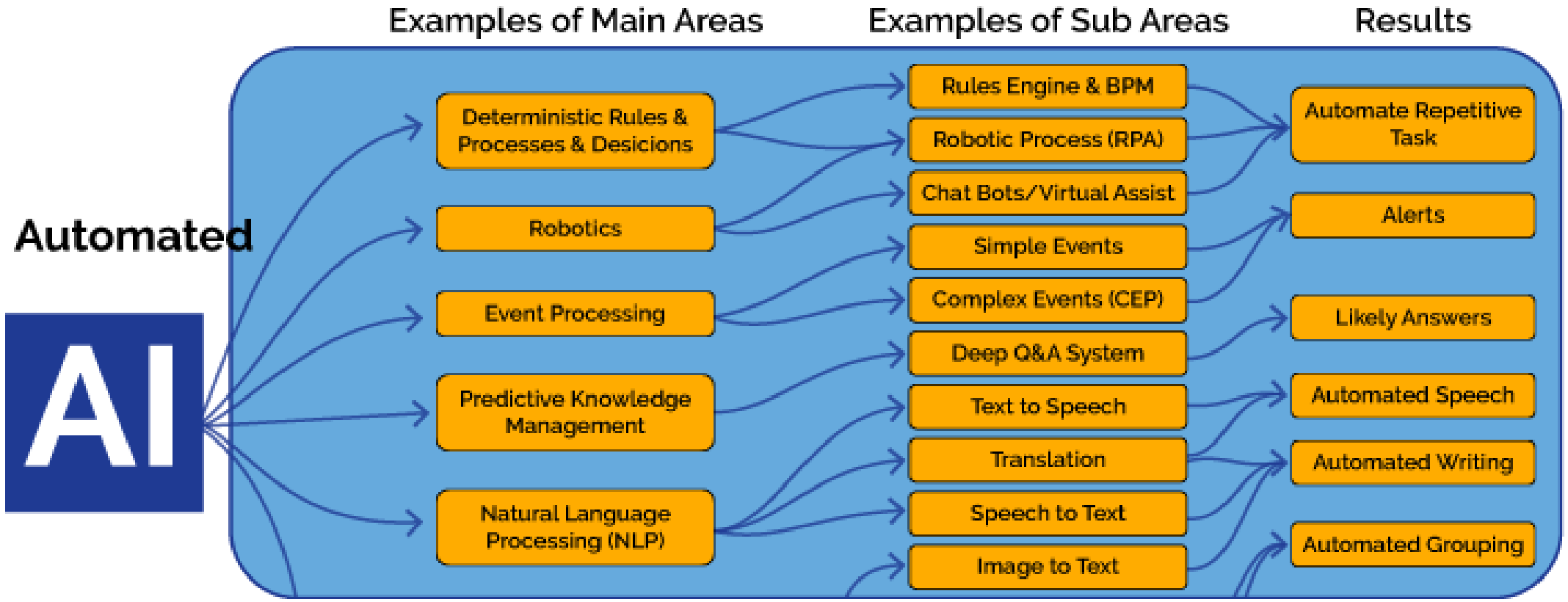
문자 인식(CHARACTER RECOGNITION)

- OCR은 손글씨나 타이핑 글씨, 인쇄된 텍스트의 이미지를 기계가 이해할 수 있는 코딩 언어로 변환하는 기술입니다.
- OCR(광학문자인식)과 NLP(자연어 처리)는 신분증이나 여권을 자동으로 읽어서 인식하고, 데이터를 다양한 양식이나 CRM(고객관계관리) 정보로 입력하고, 다양한 출처에서 얻은 고객 정보를 검증하고, 은행 카드/ 각종 지표/ 수표/ 티켓 등을 즉시 스캔하는 등의 문서 관련 업무에서 다양한 혜택을 제공합니다.

https://bit.ly/3t60zd1

NLP makes AI SYSTEM Enabled





출처 : https://www.upwork.com/hiring/for-clients/artificial-intelligence-and-natural-language-processing-in-big-data/

NLP offers great business benefits

It is true that a business can probably work okay without using NLP at the current time.
However, as data becomes even more critical to companies and the volume grows,
it will soon be impossible for humans to keep track,
and AI, NLP and other technologies
will have to take over the work.

https://medium.com/@ChrisKnightcms/natural-language-processing-offers-great-business-benefits-f450593e1089

자연어 처리

자연어 처리는 컴퓨터가 인간의 언어를 이해하고 모사할 수 있도록 연구하고 구현하는 AI 주요분야입니다.
어떻게 자연어를 컴퓨터에게 인식시킬 수 있을까요?

■ 단어 표현

- 문자 이진화 값 : 아스키코드, 유니코드

  '언' : 1100010110111000
  '어' : 1100010110110100
  } 언어적인 특성이 전혀 없습니다.

- **언어적인 특성을 반영하여 단어를 수치화 하여야 합니다. : 벡터로 표시 (Word Embedding)**

■ 자연어 처리 핵심 문제

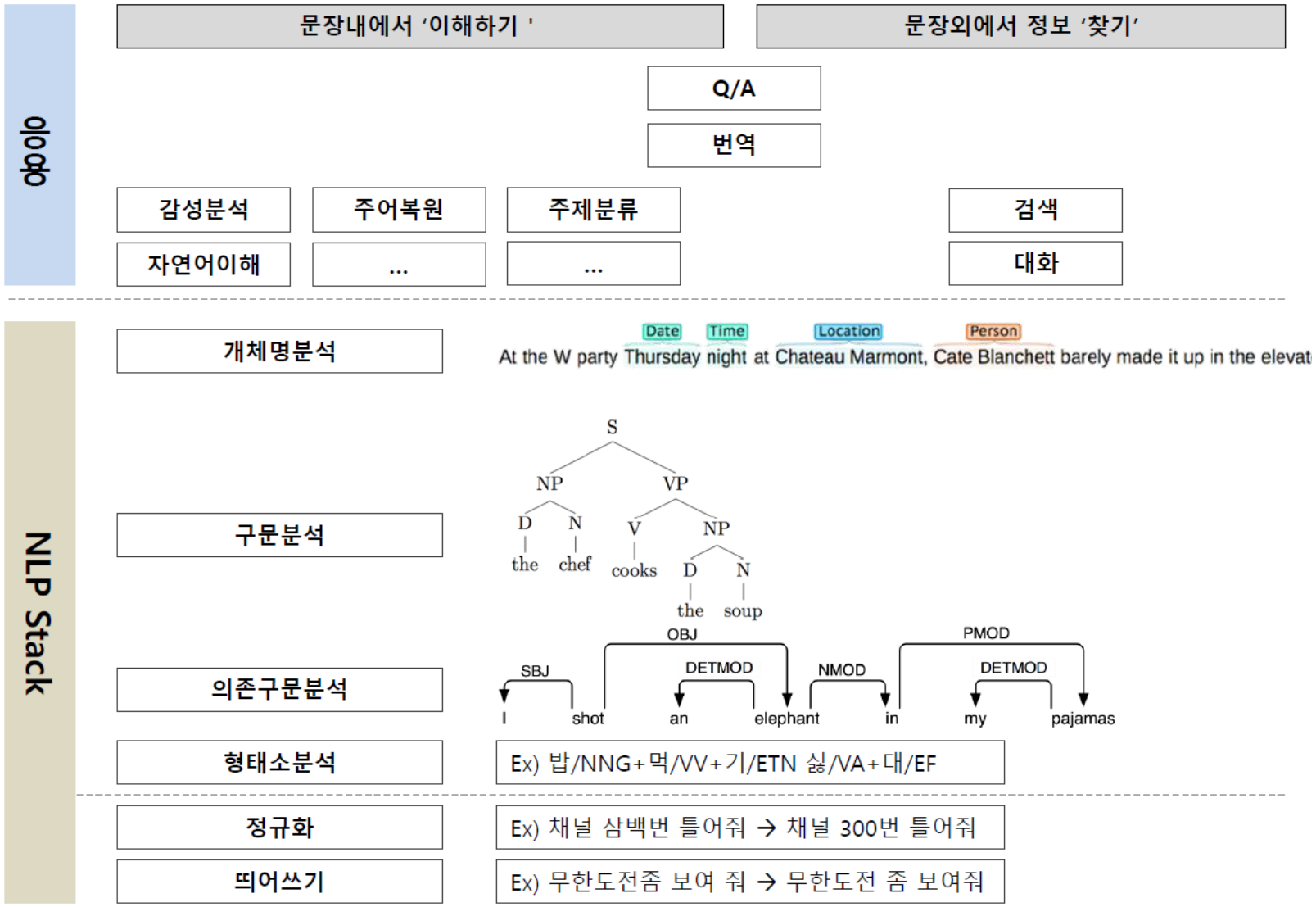
① 텍스트 분류 : 스팸메일 분류, 감정 분류, 뉴스 기사 분류

② 텍스트 유사도 : 이 노래 누가 만들었을까? 지금 나오는 노래의 작곡가는 누구야?

③ 텍스트 생성 : 챗봇 서비스, 신문기사 작성

④ 텍스트 이해 : 정보를 학습하고 사용자 질의에 응답

자연어 처리 문제



출처 : Deep Learning을 이용한 자연어 처리, 정상근, 2017.11

상당부분 해결

Spam detection

Let's go to Agra! ✓

Buy V1AGRA ... ✗

Part-of-speech (POS) tagging

ADJ ADJ NOUN VERB ADV

Colorless green ideas sleep furiously.

Named entity recognition (NER)

PERSON ORG LOC

Einstein met with UN officials in Princeton

성과를 내고 있음

Sentiment analysis

Best roast chicken in San Francisco!

The waiter ignored us for 20 minutes.

Coreference resolution

Carter told Mubarak he shouldn't run again.

Word sense disambiguation (WSD)

I need new batteries for my ***mouse***.

Parsing

I can see Alcatraz from the window!

Machine translation (MT)

第13届上海国际电影节开幕...

The 13th Shanghai International Film Festival...

Information extraction (IE)

You're invited to our dinner party, Friday May 27 at 8:30

Party
May 27
add

여전히 어려운 문제

Question answering (QA)

Q. How effective is ibuprofen in reducing fever in patients with acute febrile illness?

Paraphrase

XYZ acquired ABC yesterday

ABC has been taken over by XYZ

Summarization

The Dow Jones is up

The S&P500 jumped

Housing prices rose

Economy is good

Dialog

Where is Citizen Kane playing in SF?

Castro Theatre at 7:30.
Do you want a ticket?

https://bit.ly/3trdrM4

자연어 처리 용어

■ 자연어 처리 (Natural Language Processing)

- 우리가 일상 생활에서 사용하는 자연어의 의미를 분석하여 컴퓨터가 처리할 수 있도록 하는 것
- 번역 , 내용 요약, 감성 분석, 스팸 메일 분류, 뉴스 기사 카테고리 분류, 질의 응답 시스템, 챗봇에 사용

■ 말뭉치 (Corpus, 코퍼스)

- 자연어 연구를 위해 특정한 목적을 가지고 언어의 표본을 추출한 집합
- 컴퓨터의 발달로 말뭉치 분석이 용이해졌으며 분석의 정확성을 위해 해당 자연어를 형태소 분석하는 경우가 많음
- 언어의 빈도와 분포를 확인할 수 있는 자료이며, 현대 언어학 연구에 필수적인 자료

■ 토큰 (Token)

- 자연어 처리를 위해 문서를 작은 단위 토큰으로 로 분리
- 토큰의 단위가 상황에 따라 다르지만, 보통 의미있는 단위로 토큰을 정의

■ 토큰화 (Tokenization)

- 말뭉치를 토큰이라 불리는 단위로 나누는 작업

자연어 처리 용어

■ 정제(Cleaning), 정규화(Normalization)

- 코퍼스로부터 노이즈 데이터를 제거
- 규칙에 기반한 표기가 다른 단어들의 통합, 대소문자 통일, 불필요한 단어의 제거
- 정규 표현식은 규칙에 기반하여 한 번에 제거하는 방식으로서 매우 유용

■ 불용어(Stopword)

- 문장에 자주 등장하지만 분석에 도움이 되지 않는 단어는 사전에 제거(조사, 접미사, 관사 등)

■ 어간 추출(Stemming)

- 단어를 기본 형태로 만드는 작업(working, works, worked -> work)

■ 표제어 추출(Lemmatization)

- 단어들이 다른 형태를 가지더라도, 그 뿌리 단어를 찾아가서 단어의 개수를 줄이는 작업(am, are, is -> be)

■ 품사 태깅(Part Of Speech, POS)

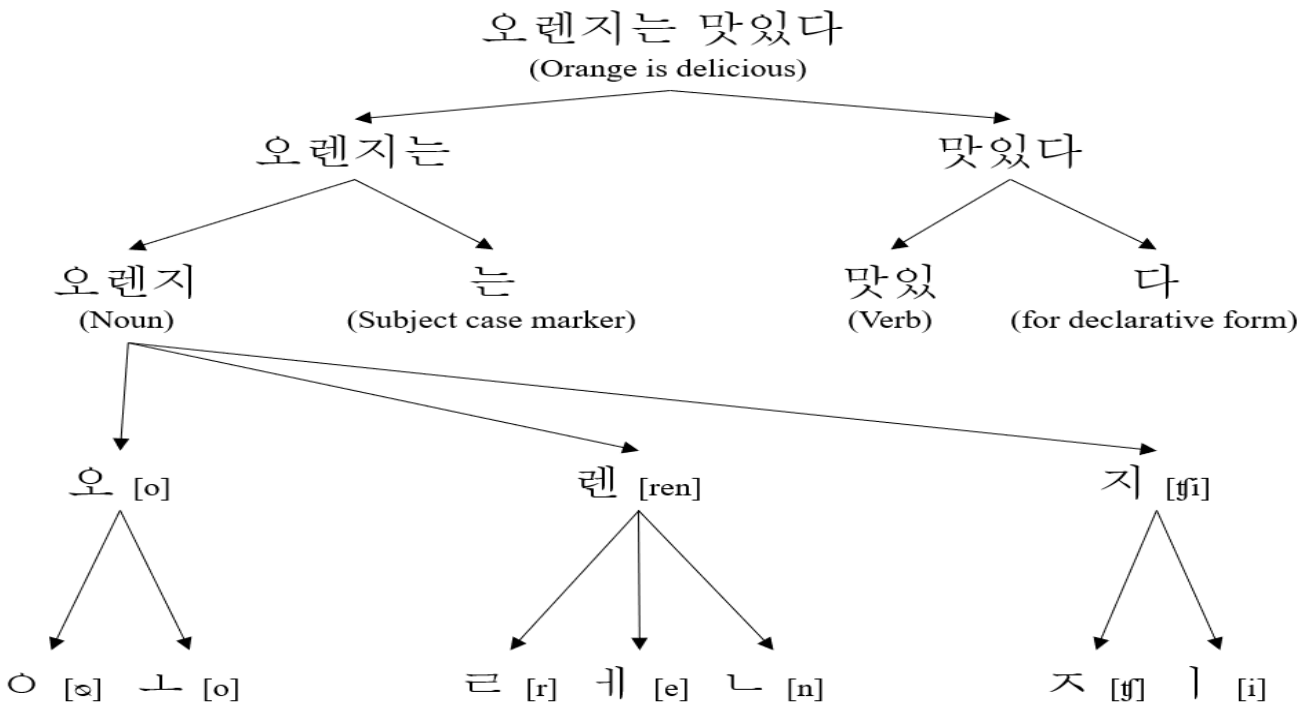
- 문장 내 단어들의 품사를 식별하여 태그를 붙여주는 것

자연어 처리 단계

- 전처리
  - 개행문자 제거
  - 특수문자 제거
  - 공백 제거
  - 중복 표현 제어
  - 이메일 제거
  - 링크 제거
  - 제목 제거
  - 불용어 제거
  - 조사 제거
  - 띄어쓰기
  - 문장분리
  - 사전 구축
- Tokenizing
  - 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

자연어 처리 단계

- 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



출처 : https://bit.ly/2QeHdF7

문자열 처리 메소드

string calculation

- len()
- min()
- max()
- count()

string search

- startswith()
- endswith()
- find()
- rfind()
- index()
- rindex()

encoding/decoding

- encode()
- decode()

number/ character

- isalnum()
- isalpha()
- isdigit()
- isnumeric()
- isdecimal()

lower/ upper

- islower()
- isupper()
- lower()
- upper()
- swapcase()
- istitle()
- title()
- capitalize()

space/ strip

- lstrip()
- rstrip()
- strip()
- isspace()
- center()

split/ join/ fill

- split()
- splitlines()
- replace()
- join()
- zfill()
- ljust()
- rjust()

https://rfriend.tistory.com/327

정규 표현식

LOG

Jul 1 03:26:12 syslog: [m_java][1/Jul/2013 03:27:12.818][j:[SessionThread<]^latcom/avc/abc/magr/service/find.something(abc/1235/locator/abc;Ljava/lang/String;)Labc/abc/abcd/abcd;(bytecode:7)

Phone Number
444-122-1234
123-122-78999
111-123-23
67-7890-2019

⌨️ https://bit.ly/2PStfsr

특징 추출 - BoW (Bag of Words)

어휘의 빈도에 대해 통계적 언어 모델을 적용해서 나타낸 것으로, 단순히 단어들의 개수를 사용하는 방법입니다.

- 머신러닝 알고리즘에서 텍스트를 사용하려고 하면, 이것을 수치적인 표현(numerical represesntation)으로 변환해야 하며, 이 수치형 표현을 특징 벡터(feature vector)라고 부릅니다.
- bag-of-words 는 문서에서 각 단어들의 빈도수를 사용하는 방법으로, 문서를 숫자 벡터로 변환하는
- 가장 기본적인 방법입니다.
- 전체 문서 {d1,d2,…,dn} 를 구성하는 고정된 단어장(vocabulary) {t1,t2,…,tm} 를 만들고 di라는 개별 문서에 단어장에 해당하는 단어들이 포함되어 있는지를 표시하는 방법입니다.

I love dogs
I hate dogs and knitting
Knitting is my hobby and my passion

	I	love	dog	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

특징 추출 - BoW (Bag of Words)

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

특징추출 – DTM/TDM (Document Term Matrix)

단어들이 각 문서에 몇번 나왔는지 행렬로 구성한 것으로 많이 나온 단어가 중요한 단어라는 것을 기반으로 합니다.

■ 문서 단어 행렬(Document Term Matrix)

- 문서 단어 행렬(Document Term Matrix) 횟수를 정리한 표
- 컬럼(Feature): 전체 문서에 나오는 모든 단어
- 행 : 문서
- 값(value) : 각 단어가 문서에 나온 횟수

	Term 1	Term 2	Term 3	...	Term M
Tweet 1	0	1	1	0	0
Tweet 2	0	1	0	0	0
Tweet 3	0	0	0	3	0
...	0	0	0	1	1
Tweet N	0	0	0	1	0

Document Term Matrix (DTM)

■ 단어 문서 행렬(Term Document Matrix)

- DTM을 전치 시킨 것
- 컬럼(Feature): 문서
- 행: 전체 문서에 나오는 모든 단어
- 값(value) : 각 단어가 문서에 나온 횟수

	Tweet 1	Tweet 2	Tweet 3	...	Tweet N
Term 1	0	0	0	0	0
Term 2	1	1	0	0	0
Term 3	1	0	0	0	0
...	0	0	3	1	1
Term M	0	0	0	1	0

Term Document Matrix (TDM)

특징 추출 - TF-IDF (Term Frequency - Inverse Document Frequency)

개별 문서에서 자주 등장하는 단어에 높은 가중치를 주고, 모든 문서에서 자주 등장하는 단어에는 페널티를 주는 방식으로 값을 부여합니다. 문서의 핵심어를 추출, 검색엔진 검색결과 순위 결정, 문서 유사도 계산에 사용합니다.

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

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

- TF(Term Frequency) : 문서에서 Term의 빈도
- DF(Document Frequency) : Term이 포함된 문서의 빈도
- IDF(Inverse Document Frequency) : DF의 역수를 취한 것, DF 정반대로 가중치 책정

특징 추출 - TF-IDF (Term Frequency - Inverse Document Frequency)

- D-1. I love dogs.
- D-2. I hate dogs and knitting.
- D-3. Knitting is my hobby and my passion.

dogs는 D-1에서 1회 나타나므로 TF는 1
dogs 는 전체 3개 문서 중 2개의 문서에서 나타나므로 DF는 2/3
IDF는 역수이므로 3/2 , 결과는 1 x (3/2) = 1.5

■ 문서 단어 행렬 (Document Term Matrix)

구분	I	love	dogs	hate	and	knitting	is	my	hobby	passion
D-1	1	1	1							
D-2	1		1	1	1	1				
D-3					1	1	1	2	1	1

dogs는 D-1에서 1회 나타나므로 TF는 1
dogs 는 전체 3개 문서 중 2개의 문서에서 나타나므로 DF는 2/3
IDF는 역수이므로 3/2 , 결과는 1 x (3/2) = 1.5

■ TF * (N/DF)

	I	love	dogs	hate	and	knitting	is	my	hobby	passion
D-1	1.5	**3**	1.5							
D-2	1.5		1.5	**3**	1.5	1.5	1.5			
D-3					1.5	1.5	**3**	**6**	**3**	**3**

Scikit-Learn 패키지

DictVectorizer

- 각 단어의 수를 세어놓은 사전에서 BOW 인코딩 벡터를 만든다.

CountVectorizer

- 문서 집합에서 단어 토큰을 생성하고 각 단어의 수를 세어 BOW 인코딩 벡터를 만든다.

TfidfVectorizer

- CountVectorizer와 비슷하지만 TF-IDF 방식으로 단어의 가중치를 조정한 BOW 인코딩 벡터를 만든다.

HashingVectorizer

- 해시 함수(hash function)을 사용하여 적은 메모리와 빠른 속도로 BOW 인코딩 벡터를 만든다.

Scikit-Learn - DictVectorizer

각 단어의 수를 세어놓은 사전에서 BOW 인코딩 벡터를 만듦

```
>>> from sklearn.feature_extraction import DictVectorizer
>>> v = DictVectorizer(sparse=False)
>>> D = [{'foo': 1, 'bar': 2}, {'foo': 3, 'baz': 1}]
>>> X = v.fit_transform(D)
>>> X
array([[2., 0., 1.],
       [0., 1., 3.]])
>>> v.inverse_transform(X) == [{'bar': 2.0, 'foo': 1.0},
...                            {'baz': 1.0, 'foo': 3.0}]
True
>>> v.transform({'foo': 4, 'unseen_feature': 3})
array([[0., 0., 4.]])
```

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.DictVectorizer.html

Scikit-Learn - CountVectorizer

문서 집합에서 단어 토큰을 생성하고 각 단어의 수를 세어 BOW 인코딩 벡터를 만듦

```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = CountVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
>>> print(X.toarray())
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Scikit-Learn - TfidfVectorizer

TF-IDF 방식으로 단어의 가중치를 조정한 BOW 인코딩 벡터를 만듬

```
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = TfidfVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
>>> print(X.shape)
(4, 9)
```

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Scikit-Learn - HashingVectorizer

해시 함수(hash function)을 사용하여 적은 메모리와 빠른 속도로 BOW 인코딩 벡터를 만듬

```
>>> from sklearn.feature_extraction.text import HashingVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = HashingVectorizer(n_features=2**4)
>>> X = vectorizer.fit_transform(corpus)
>>> print(X.shape)
(4, 16)
```

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.HashingVectorizer.html

전처리

머신러닝/딥러닝에서 텍스트 자체를 특성으로 사용할 수 없으며, 토큰화, 불용어 제거, 어간 추출등의 작업이 필요합니다.

Basic

- 가장 기초적인 전처리
- html tag 제거
- 숫자, 특수문자 제거
- Lowercasing
- "@%*=()/+ 와 같은 punctuation 제거

Spell check

- 사전 기반의 오탈자 교정
- 줄임말 원형 복원

  (e.g. I'm not happy -〉 I am not happy)

Part-of-Speech

- 형태소 분석
- Noun, Adjective, Verb, Adverb만 학습에 사용

Stemming

- 형태소 분석 이후 동사 원형 복원

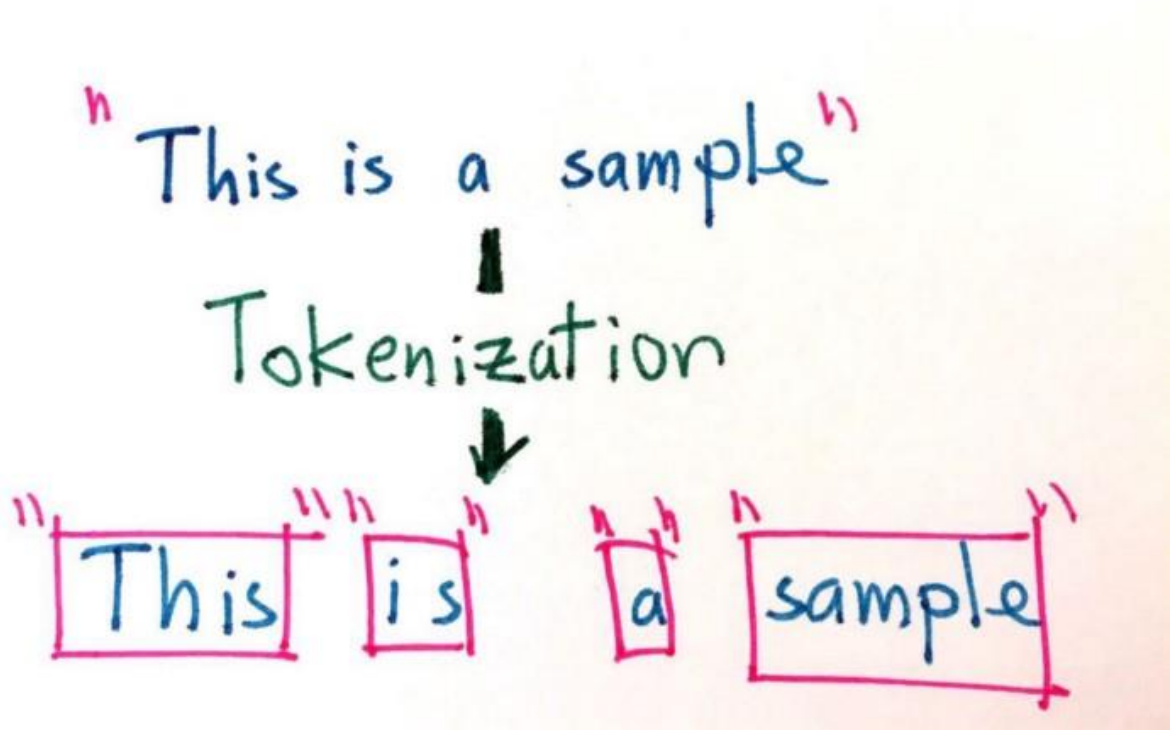
Stopwords

- 불용어 제거

토큰화(Tokenizing)

데이터를 의미 있는 작은 단위(Token)단위로 나누는 작업으로, 어절 , 단어 , 형태소 , 음절 , 자소 등으로 토큰화를 합니다.

- 한글 문장 분리기: kss(Korean Sentence Splitter)
- 영문 자연어 처리: NLTK(Natural Language Toolkit)
- 한글 자연어 처리: KoNLPy
  - 한글 형태소 분석기: Mecab, Komoran, Hannanum, Kkma, Okt(Open Korean Text, 예전이름 Twitter)



품사

품사를 분류하는 기준은 언어마다 천차만별이지만, 실체를 가리키는 단어 부류(명사)와 동작을 가리키는 단어 부류(동사)가 있다는 것은 모든 언어에 공통입니다.

언어의 품사				
체언	명사	대명사	수사	
용언	동사	조동사	형용사	계사
수식언	관형사	부사	관사	접속사
관계언 & 독립언	조사	전치사	감탄사	

https://namu.wiki/w/품사

NLTK 패키지

교육용으로 개발된 자연어 처리 및 문서 분석용 파이썬 패키지로 다양한 기능 및 예제를 가지고 있습니다.
말뭉치, 토큰 생성, 형태소 분석, 품사 태깅 기능을 제공합니다.

nltk_usage.ipynb

```
from nltk.tag import pos_tag
from nltk.tokenize import word_tokenize

nltk.download()
nltk.download('punkt')
pos_tag(word_tokenize("John's big idea isn't all that bad."))

[Out]
[('John', 'NNP'), ("'s", 'POS'), ('big', 'JJ'), ('idea', 'NN'),
 ('is', 'VBZ'), ("n't", 'RB'), ('all', 'PDT'), ('that', 'DT'),
 ('bad', 'JJ'), ('.', '.')]
```

한글 문장 분리기

https://github.com/hyunwoongko/kss

- 형식적으로 올바른 글이라면 '. (마침표)' 등으로 문장을 나누면 되겠지만, 특히 웹 상에 작성된 많은 글들은 마침표를 찍지 않은 문장들도 많고 문장으로 끊기에 애매한 문장도 많이 존재합니다.
- 설치 : pip install kss

[일상] 강남역 맛집 / 강남 맛집 : 강남 토끼정 | 낙서장

2019. 1. 9. 12:04

https://blog.naver.com/jully1211/221437777873 복사

번역하기 지도로 보기

강남역 맛집으로 소문난 강남 토끼정에 다녀왔습니다.
회사 동료 분들과 다녀왔는데 분위기도 좋고 음식도 맛있었어요
다만, 강남 토끼정이 강남 쉑쉑버거 골목길로 쭉 올라가야 하는데
다들 쉑쉑버거의 유혹에 넘어갈 뻔 했답니다

https://blog.naver.com/jully1211/221437777873

한글 문장 분리기

https://github.com/hyunwoongko/kss

```
>>> from kss import split_sentences
>>> text = "미리 예약을 할 수 있는 시스템으로 합리적인 가격에 여러 종류의 생선, 그리고 다양한 부위를 즐길 수 있기
>>> split_sentences(text)
```

```
['미리 예약을 할 수 있는 시스템으로 합리적인 가격에 여러 종류의 생선, 그리고 다양한 부위를 즐길 수 있기 때문이다',
'계절에 따라 모둠회의 종류는 조금씩 달라지지만 자주 올려주는 참돔 마스까와는 특히 맛이 매우 좋다',
'제철 생선 5~6가지 구성에 평소 접하지 못했던 부위까지 색다르게 즐길 수 있다']
```

한국어 전처리 – 맞춤법

https://github.com/ssut/py-hanspell

- py-hanspell은 네이버 맞춤법 검사기를 이용한 파이썬용 한글 맞춤법 검사 라이브러리입니다.
- 설치 : pip install git+https://github.com/haven-jeon/PyKoSpacing.git

```
from hanspell import spell_checker

sent = "대체 왜 않돼는지 설명을 해바"
spelled_sent = spell_checker.check(sent)
checked_sent = spelled_sent.checked

print(checked_sent)
```

대체 왜 안되는지 설명을 해봐

한국어 전처리 – 띄어쓰기

https://github.com/haven-jeon/PyKoSpacing

- 띄어쓰기가 되어있지 않은 문장을 띄어쓰기를 한 문장으로 변환해주는 패키지입니다.
- 대용량 코퍼스를 학습하여 만들어진 띄어쓰기 딥러닝 모델로 준수한 성능을 가지고 있습니다.
- 설치 : pip install git+https://github.com/haven-jeon/PyKoSpacing.git

```
from pykospacing import spacing

print(spacing("지난해12월27일부터올해1월10일까지의데이터를분석했다."))
```

지난해 12월 27일부터 올해 1월 10일까지의 데이터를 분석했다.

KoNLPy 패키지

KoNLPy(코엔엘파이)는 형태소 분석, 품사분석 등 한국어 정보처리를 위한 파이썬 패키지입니다.
형태소(morpheme)는 의미를 가지는 언어 단위 중 가장 작은 단위로 의미 혹은 문법적 기능의 최소단위입니다.

https://konlpy.org/ko/latest/


```
>>> 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),
 (와, JKM),
 (함께, MAG),
 (설명, NNG),
 (을, JKO),
 (최대한, NNG),
 (상세히, MAG),
 (!, SF),
 (^^, EMO)]
```

KoNLPy 패키지

KoNLPy는 다음과 같은 다양한 형태소 분석, 태깅 라이브러리를 파이썬에서 쉽게 사용할 수 있도록 모아 놓았습니다.

- Hannanum(한나눔) : KAIST Semantic Web Research Center 개발
http://semanticweb.kaist.ac.kr/hannanum/

- Kkma(꼬꼬마) : 서울대학교 IDS(Intelligent Data Systems) 연구실 개발
http://kkma.snu.ac.kr/

- Komoran(코모란) : Shineware에서 개발
https://github.com/shin285/KOMORAN

- Mecab(메캅) : 일본어용 형태소 분석기를 한국어를 사용할 수 있도록 수정
https://bitbucket.org/eunjeon/mecab-ko

- Open Korean Text(오픈 소스 한국어 분석기) : 과거 트위터 형태소 분석기
https://github.com/open-korean-text/open-korean-text

KoNLPy 실습

konlpy_usage.ipynb

기타 패키지

https://pypi.org/project/customized-KoNLPy/

- 확실히 알고 있는 단어들에 대해서는 라이브러리를 거치지 않고 주어진 어절을 아는 단어들로 토크나이징 / 품사판별을 하는 기능을 제공합니다.
- 이를 위해 template 기반 토크나이징을 수행합니다.
- 설치 : pip install customized_konlpy

https://github.com/lovit/soynlp

- 품사 태깅, 단어 토큰화 등을 지원하는 단어 토크나이저입니다.
- 비지도 학습으로 토큰화를 한다는 특징을 갖고 있으며, 데이터에 자주 등장하는 단어들을 단어로 분석합니다.
- 설치 : pip install soynlpsoynlp

한국어 전처리 실습

KoreanPreprocessing.ipynb

참고 : 한국어 자연어처리 튜토리얼(김성현)

https://github.com/MrBananaHuman/KorNlpTutorial

Thank you

kgpark88@gmail.com