

딥러닝을 활용한 자연어 분석 word cloud

김용범

무영인터내셔널



Yong Beom, Kim

YongBeomKim

from Quant to Machine Learning

Edit bio

MuYong International
 Suwon, Republic of Korea

Overview

Repositories 25

Stars 8

Followers 6

Following 19

Popular repositories

Customize your pinned repositories

FinanceBasic2018

파이썬금융 기본 tutorial

Jupyter Notebook ★ 3 5

googlefinance.get

Get the google finance data convert to DataFrame in python 3

Python ★ 3 1

Tacademy

T-Academy 스터디 자료정리

Jupyter Notebook ★ 2 2

Finance

국내 코스피 코스닥 시장분석

Jupyter Notebook 1

Django

python3.6 django 11.1 & 2.0(tested)

Jupyter Notebook 2

TensorFlow

Tensorflow Machine Learning Tutorials

Jupyter Notebook 2

932 contributions in the last year

Contribution settings ▾



[Learn how we count contributions.](#)

Less More

수업준비하기

Github 소스 다운로드 ([git](#))

https://github.com/YongBeomKim/nltk_tutorial

Introduction

Word Cloud

기본기능 활용하기

Python의 특징

1. 다양한 기능의 Open Source 를 제공
2. 모듈에서 정하는 포맷을 맞추는 전처리 필요 (python 기본기!!)
3. 기타 연산 및 출력은 자동

Python : <https://www.python.org>

워드 클라우드 만들기 / Word Cloud

```
In [1]: # ! pip3 install wordcloud
```

```
In [2]: f          = open('./data/테스트문서.txt', 'r')
        texts_org = f.read()
        f.close()
```

```
In [3]: # ! pip3 install matplotlib
```

```
In [4]: from matplotlib import rc
        rc('font', family='NanumGothic')
```

```
In [5]: %matplotlib inline
        from wordcloud import WordCloud
        wcloud = WordCloud('./data/D2Coding.ttf',relative_scaling = 0.2).generate(texts_org)

        import matplotlib.pyplot as plt
        plt.figure(figsize=(12,12))
        plt.imshow(wcloud, interpolation='bilinear')
        plt.axis("off")
```

```
Out[5]: (-0.5, 399.5, 199.5, -0.5)
```

워드 클라우드 출력



하지만..

1. 기타 번역, 챗봇 등도 **모듈**로써 해결가능
2. **오류 발생** , 또는 **결과물**이 맘에 안들 때
3. 자연어 분석단위를 이해하고, **절차별 수정/작업**

수업 시작

참고사이트

[NLTK BOOK](#)

[Gensim Tutorial](#)

[Konlpy Document](#)

<https://ratsgo.github.io>

<https://www.lucypark.kr/courses/2015-dm/text-mining.html#2-tokenize>

수강대상

1. 자연어 기본이론 부터 간단한 응용까지
2. 파이썬 썸 만져봤다

```
result = [ text.lower() for text in texts
```

```
if len(text) >= 4 ]
```

```
token = { text[0] : text[1] for text in texts }
```

오늘의 수업내용

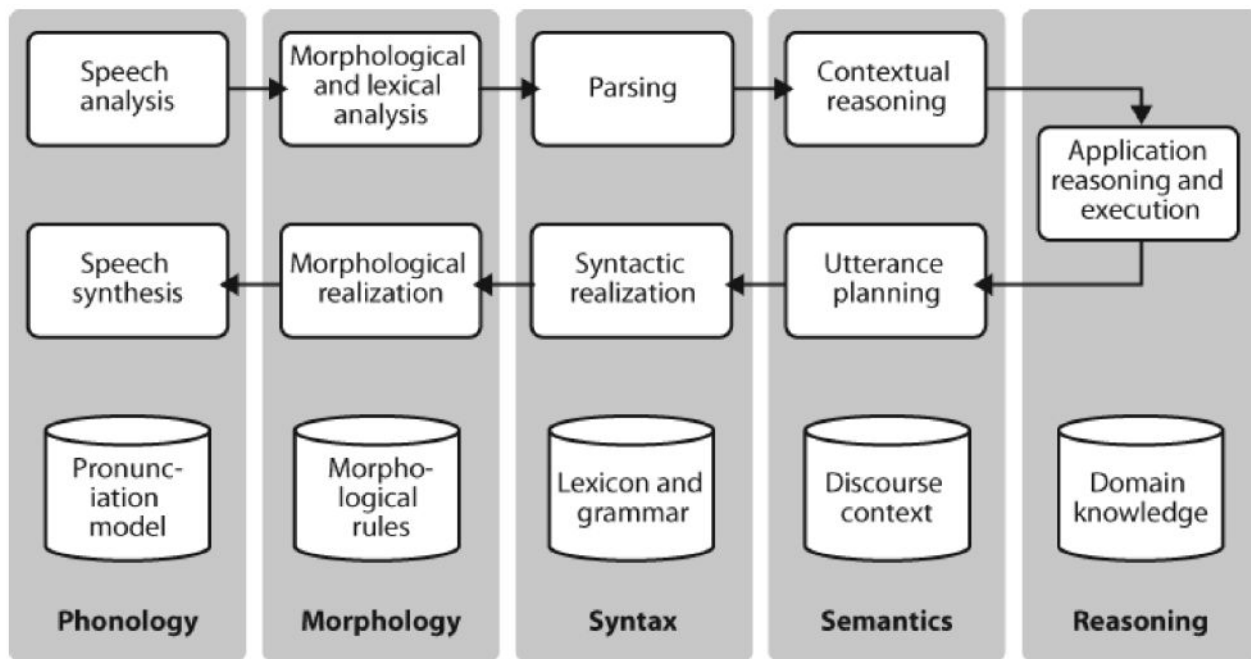
1. **Token** 개념
2. **형태소 / 문법** 태그
3. **문장 구조** 분석
4. 단어 / 문장의 **의미**분석
5. 나이브베이즈, HMM **머신러닝 이론** 이해
6. CNN, RNN, LSTM, Seq2Seq, Attention **딥러닝 이론** 이해

言語學

전체 작업과정

working Process

Working Process



Step1:
데이터수집 및 탐색

Step1:
전처리

Step2:
데이터구조화

Step2:
변수추출
(Feature Extraction)

Step4:
알고리즘 학습 및 평가

자연어 분석과정

1. 음운론(Phonology) : 말소리를 연구 ex) 음성인식
2. 형태론(Morphology) : 단어를 정규화/ 형태소
3. 통사론(syntax) : 문법적 구조(Parsing)
4. 의미론(Semantics) : 의미 차이 ex) 뉘앙스, 톤, 의도(긍/부정)

Natural Language Tool Kit

NLTK

pip3 install -U nltk

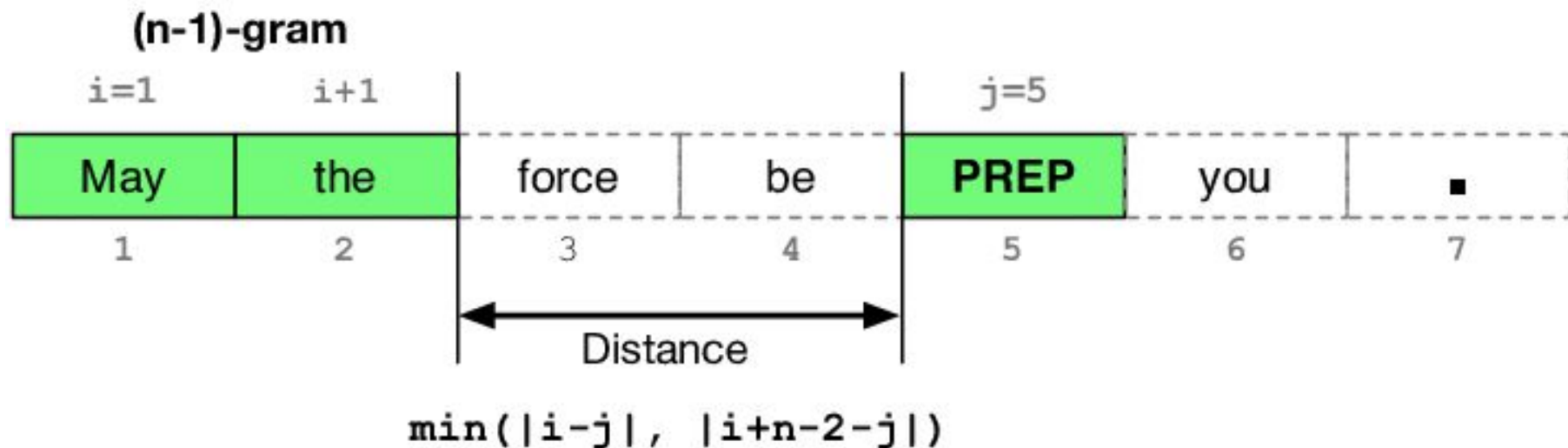
1. **Natural Language Tool Kit**
2. 공식 **Document** - <https://www.nltk.org/>
3. 자연어 처리 **기본**모듈
4. **영문**을 기본으로 제작

Token

토큰 (Token)

1. **토큰(token)** : 의미를 가지는 문자열 (단어, 절, 문장 등)
2. **토크나이징(tokenizing)** : 토큰을 나누는 작업
3. 영문은 **공백** 만으로도 충분히 토큰을 나눌 수 있다
4. 한글은 **합성어, 조사합성** 등의 별도 처리를 요한다

Tokenization (영문)



Tokenization (한글)

충실히 돕는 조사도 붙여 쓴다.

조사	격조사	주격 조사	가, 께서, 에서 etc.
		서술격 조사	이다
		관형격 조사	의
		목적격 조사	을
		보격 조사	이
		부사격 조사	에, 에서, 에게 etc.
		호격 조사	야, 여, 아, 이여, 시여 etc.
	접속조사		와, 과, (이)랑, (이)며 etc.
	보조사		은, 는, 도, 만, 까지, 조차, 부터, 마저 etc.

너**에게** 하고 싶은 말이 있어.

Tokenization

```
In [18]: text = """힘든 월요일, 아침.  
힘든 직장인 분들에게. 월요일 식혜를 제공합니다"""
```

```
In [19]: from nltk import sent_tokenize  
sent_tokenize(text)
```

```
Out[19]: ['힘든 월요일, 아침.', '힘든 직장인 분들에게.', '월요일 식혜를 제공합니  
다']
```

```
In [20]: from nltk import word_tokenize  
text = word_tokenize(text)  
text
```

```
Out[20]: ['힘든', '월요일', ',', '아침', '.', '힘든', '직장인', '분들에게',  
, '.', '월요일', '식혜를', '제공합니다']
```


Token 빈도분석

```
In [6]: from nltk import FreqDist  
        dict(FreqDist(text))
```

```
Out[6]: {'힘든': 2,  
        '월요일': 2,  
        ',': 1,  
        '아침': 1,  
        '.': 2,  
        '직장인': 1,  
        '분들에게': 1,  
        '식혜를': 1,  
        '제공합니다': 1}
```

연설문 Token 빈도분석

```
In [10]: texts_token = word_tokenize(texts_org)
         texts_token_dict = dict(FreqDist(texts_token))
```

```
In [11]: import pandas as pd
         texts_token_series = pd.Series(texts_token_dict)
         texts_token_series.sort_values(ascending=False)
```

```
Out[11]: .          168
         ,           67
         것입니다    28
         한반도     20
         함께        18
         있습니다    16
         합니다      16
         수          15
         위한        13
         북한의      12
         우리        12
         나는        11
         북한이      11
         한반도의    11
         이          11
         여러분      10
```

워드 클라우드 출력



Regex

정규식 활용

import re

1. 파이썬 기본제공 모듈, 다양한 언어에서도 활용
2. 특정한 규칙을 활용하여 문자열의 집합을 표현하는 언어
3. 코드가 간단한 대신, 가독성이 떨어져서 난이도가 있다

정규식 문법

정규표현식	표현	설명
[xy]	One of: "x" "y"	x,y중 하나를 찾습니다.
[^xy]	None of: "x" "y"	x,y를 제외하고 문자 하나를 찾습니다. (문자 클래스 내의 ^는 not을 의미합니다.)
[x-z]	One of: "x" - "z"	x~z 사이의 문자중 하나를 찾습니다.
\w^	"^"	^(특수문자)를 식에 문자 자체로 포함합니다. (escape)
\wb	word_boundary	문자와 공백사이의 문자를 찾습니다.
\WB	non_word_boundary	문자와 공백사이가 아닌 값을 찾습니다.
\wd	digit	숫자를 찾습니다.
\WD	non_digit	숫자가 아닌 값을 찾습니다.
\ws	white_space	공백문자를 찾습니다.
\WS	non_white_space	공백이 아닌 문자를 찾습니다.
\wt	tab	Tab 문자를 찾습니다.
\wv	vertical_tab	Vertical Tab 문자를 찾습니다.
\ww	word	알파벳 + 숫자 + _ 를 찾습니다.
\WW	non_word	알파벳 + 숫자 + _ 을 제외한 모든 문자를 찾습니다.

NLTK 의 정규식 문법 - 숫자 추출

```
In [10]: text = """Park 010-1234-1234 Kim 010-8888-9999
Lee 010-2123-1299 한남총 010-222-9999 메갈녀 010-555-2345"""

from nltk.tokenize import RegexpTokenizer
re_capt = RegexpTokenizer(r'\d+')
re_capt.tokenize(text)
```

```
Out[10]: ['010',
'1234',
'1234',
'010',
'8888',
'9999',
'010',
'2123',
'1299',
'010',
'222',
'9999',
'010',
'555',
'2345']
```

NLTK 의 정규식 문법 - 영문/ 한글 추출

```
In [11]: from nltk.tokenize import RegexpTokenizer  
re_capt = RegexpTokenizer(r'[A-z]\w+')  
re_capt.tokenize(text)
```

```
Out[11]: ['Park', 'Kim', 'Lee']
```

```
In [12]: from nltk.tokenize import RegexpTokenizer  
re_capt = RegexpTokenizer(r'[가-힣]\w+')  
re_capt.tokenize(text)
```

```
Out[12]: ['한남총', '메갈녀']
```


Stemming

정규화

Stemming (어간추출)

1. 접사 등을 제거하여, 단어의 **어간을 분리**한다
2. 단어들이 **동일한 어간**으로 맵핑되게 하는 것이 목적
3. **Penn Treebank Corpus, WordPunctTokenizer** 등의 다양한 구분기법이 **NLTK** 모듈에서 기본 제공

Treebank Corpus를 활용한 Stemming

```
In [1]: %%time
text = " Don't hesitate to ask questions"

# Penn Treebank Corpus 에 따른 기준을 사용하여, 문법별로 나눈다
from nltk.tokenize import TreebankWordTokenizer
tokenizer = TreebankWordTokenizer()
token = tokenizer.tokenize(text)
print(token)
```

```
['Do', "n't", 'hesitate', 'to', 'ask', 'questions']
CPU times: user 575 ms, sys: 368 ms, total: 943 ms
Wall time: 480 ms
```

```
In [2]: from nltk import pos_tag
pos_tag(token)
```

```
Out[2]: [('Do', 'VBP'),
          ("n't", 'RB'),
          ('hesitate', 'VB'),
          ('to', 'TO'),
          ('ask', 'VB'),
          ('questions', 'NNS')]
```

WordPunct를 활용한 Stemming

```
In [3]: %%time
from nltk.tokenize import WordPunctTokenizer
tokenizer = WordPunctTokenizer()
token     = tokenizer.tokenize(text)
print(token)

['Don', "'", 't', 'hesitate', 'to', 'ask', 'questions']
CPU times: user 1.3 ms, sys: 0 ns, total: 1.3 ms
Wall time: 1.28 ms
```

```
In [4]: from nltk import pos_tag
pos_tag(token)
```

```
Out[4]: [('Don', 'NNP'),
          ("'", 'POS'),
          ('t', 'NN'),
          ('hesitate', 'NN'),
          ('to', 'TO'),
          ('ask', 'VB'),
          ('questions', 'NNS')]
```

Stemming Tag 내용 살펴보기

```
In [24]: import nltk.help as nltk_help  
nltk_help.upenn_tagset('PRP') # 대명사
```

PRP: pronoun, personal

hers herself him himself hisself it itself me myself one oneself ours
ourselves ownself self she thee theirs them themselves they thou thy us

```
In [25]: nltk_help.upenn_tagset('JJ') # 형용사
```

JJ: adjective or numeral, ordinal

third ill-mannered pre-war regrettable oiled calamitous first separable
ectoplasmic battery-powered participatory fourth still-to-be-named
multilingual multi-disciplinary ...



Stemming 의 구조적 한계

1. 독립된 개별 단어의 태그 값을 추출
2. 평문, 일반문의 경우 규격화된 결과를 출력
3. 강조문/ 도치문/ 압축문 등 문장 특성별 차이 구분은 어렵다
4. Token 분류기준, Tag 생성기준이 별도로 존재

한글

Konlpy


한글

pip3 install konlpy

1. 설치공식문서 : <http://konlpy.org/ko/v0.5.1/install/>
2. Java 7 이상을 먼저 설치해야 한다
3. `sudo apt-get install g++ openjdk-7-jdk python-dev python3-dev`
4. `pip3 install JPytype1-py3 # Python 3.x`
5. `pip3 install konlpy`

Konlpy - <https://pypi.org/project/konlpy/#history>

konlpy 0.5.0

`pip install konlpy`

✓ Latest version

Last released: About 2 days ago

Python package for Korean natural language processing.

Navigation

- Project description
- Release history**
- Download files

Project links


- Homepage

Statistics


View statistics for this project via

Release history


THIS VERSION



0.5.0
About 2 days ago



0.4.5
About 2 days ago



0.4.4
Oct 24, 2015

Release notifications

konlpy 의 Stemming

1. Sentence 에서 **Token**을 추출하고
2. Token별 확률이 높은 **Stemming** 을 생성 후 적합한 **Tag**를 출력
3. 이 두가지 작업을 1개의 함수로 일괄 처리한다

```
In [1]: from konlpy.tag import Twitter
twitter = Twitter()
text = '단독입찰보다 복수입찰의 경우'
twitter.pos(text)
```

```
Out[1]: [('단독', 'Noun'),
          ('입찰', 'Noun'),
          ('보다', 'Josa'),
          ('복수', 'Noun'),
          ('입찰', 'Noun'),
          ('의', 'Josa'),
          ('경우', 'Noun')]
```

stem = True - Corpus 활용여부 (Twitter 모듈)
cf) false는 Tokenize 작업 결과만 갖고서 Tag 출력

```
In [30]: %%time  
text = '민병삼 대령의 항명행위로 조치했다'  
print(twitter.pos(text))
```

```
[('민병삼', 'Noun'), ('대령', 'Noun'), ('의', 'Josa'), ('항', 'Noun'), ('명', 'Suffix'), ('행  
위', 'Noun'), ('로', 'Josa'), ('조치', 'Noun'), ('했', 'Verb'), ('다', 'Eomi')]
```

```
CPU times: user 30.4 ms, sys: 609 µs, total: 31 ms
```

```
Wall time: 14.1 ms
```

```
In [31]: %%time  
text = '민병삼 대령의 항명행위로 조치했다'  
print(twitter.pos(text, stem=True))
```

```
[('민병삼', 'Noun'), ('대령', 'Noun'), ('의', 'Josa'), ('항', 'Noun'), ('명', 'Suffix'), ('행  
위', 'Noun'), ('로', 'Josa'), ('조치', 'Noun'), ('하다', 'Verb')]
```

```
CPU times: user 528 ms, sys: 21.2 ms, total: 549 ms
```

```
Wall time: 215 ms
```

Konlpy 0.5.1

1. **Twitter** 모듈은 속도에 있어서 상당부분 개선
2. **Stemming** 여부에 따른 속도차이가 줄어듬 (더 빠른데?)

```
In [39]: %%time
text = '민병삼 대령의 항명행위로 초치했다'
print(twitter.pos(text))

[('민병삼', 'Noun'), ('대령', 'Noun'), ('의', 'Josa'), ('항', 'Noun'), ('명', 'Suffix'), ('행', 'Noun'), ('로', 'Josa'), ('초치', 'Noun'), ('했다', 'Verb')]
CPU times: user 5.02 ms, sys: 0 ns, total: 5.02 ms
Wall time: 3.87 ms
```

```
In [40]: %%time
text = '민병삼 대령의 항명행위로 초치했다'
print(twitter.pos(text, stem=True))

[('민병삼', 'Noun'), ('대령', 'Noun'), ('의', 'Josa'), ('항', 'Noun'), ('명', 'Suffix'), ('행', 'Noun'), ('로', 'Josa'), ('초치', 'Noun'), ('하다', 'Verb')]
CPU times: user 4.37 ms, sys: 12 µs, total: 4.38 ms
Wall time: 3.51 ms
```

Konlpy

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

꼬꼬마		코모란		트위터	
태그	설명	태그	설명	태그	설명
EC	연결 어미	EC	연결 어미	Eomi	어미
ECD	의존적 연결 어미				
ECE	대등 연결 어미				
ECS	보조적 연결 어미				
EF	종결 어미	EF	종결 어미		
EFA	청유형 종결 어미				
EFI	감탄형 종결 어미				
EFN	평서형 종결 어미				
EFO	명령형 종결 어미				
EFQ	의문형 종결 어미				
EFR	존칭형 종결 어미				
ET	전성 어미				
ETD	관형형 전성 어미	ETM	관형형 전성 어미		
ETN	명사형 전성 어미	ETN	명사형 전성 어미		
EP	선어말 어미	EP	선어말어미	PreEomi	선어말어미
EPH	존칭 선어말 어미				
EPP	공손 선어말 어미				

Konlpy

1. **Hannanum** : 카이스트 최기선 교수 연구실
2. **KKMA** : 서울대 이상구 교수 연구실
3. **Twitter** : OpenKoreanText 오픈 소스 한국어 처리기
4. **Eunjeon** : 은전한닢 프로젝트 (**별도 설치과정 필요**)
5. **KOMORAN** : Junsoo Shin님의 코모란 v3.3.3
6. 빠른 분석이 중요할 때 : **트위터**
7. 정확한 품사 정보가 필요할 때 : **꼬꼬마**
8. 정확성, 시간 모두 중요할 때 : **코모란**

꼬꼬마, 한나눔 Stemming

CPU times: user 113 ms, sys: 0 ns, total: 113 ms

Wall time: 47.2 ms

In [4]:

```
%%time
from konlpy.tag import Kkma
kkma = Kkma()
print(kkma.pos(text))
```

```
[('민병', 'NNG'), ('삼', 'NNG'), ('대령', 'NNG'), ('의', 'JKG'), ('항명', 'NNG'), ('행위', 'NNG'), ('로', 'JKM'), ('초치', 'NNG'), ('하', 'XSV'), ('었', 'EPT'), ('다', 'EFN')]
```

CPU times: user 17.4 s, sys: 282 ms, total: 17.7 s

Wall time: 6.42 s

In [5]:

```
%%time
from konlpy.tag import Hannanum
han = Hannanum()
print(han.pos(text))
```

```
[('민병삼', 'N'), ('대령', 'N'), ('의', 'J'), ('항명행위', 'N'), ('로', 'J'), ('초치했다', 'N')]
```

CPU times: user 8.79 s, sys: 79.1 ms, total: 8.87 s

Wall time: 2.82 s

Word Cloud

Stemming

twitter 로 명사만 추출한 뒤, Text 업데이트

```
In [35]: # 독일 퀘르버 재단 연설문 : 베를린 선언
f        = open('./data/베를린선언.txt', 'r')
texts_org = f.read()
f.close()
```

```
In [36]: texts_nouns = twitter.nouns(texts_org)
texts_nouns[:10]
```

```
Out[36]: ['독일', '국민', '여러분', '고국', '국민', '여러분', '하울', '젠', '퀘르버', '재단']
```

```
In [37]: result_nouns = ''
for txt in texts_nouns:
    result_nouns += " " + txt

result_nouns[:500]
```

```
Out[37]: ' 독일 국민 여러분 고국 국민 여러분 하울 젠 퀘르버 재단 이사 모드 전 동독 총리 내외 귀빈 여러분 먼저 냉전 분단
통일 그 힘 유럽 통합 국제 평화 선도 독일 독일 국민 무한 경의 표 오늘 이 자리 주신 독일 정부 퀘르버 재단 감사
얼마 전 별세 헬 무트 쿨 총리 가족 독일 국민 은 애도 위로 마음 대한민국 냉전 시기 환경 속 적극 능동 외교 독일
통일 유럽 통합 주도 헬 무트 쿨 총리 업적 것 친 애하 내외 귀빈 여러분 곳 베를린 지금 년 전 한국 김대중 대통령
```

추출한 명사를 활용하여 WordCloud 출력

```
In [38]: %matplotlib inline
from matplotlib import rc
import matplotlib.pyplot as plt
rc('font', family='NanumGothic')

from wordcloud import WordCloud
wcloud = WordCloud('./data/D2Coding.ttf', relative_scaling = 0.2).generate(result_nouns)
plt.figure(figsize=(12,12))
plt.imshow(wcloud, interpolation='bilinear')
plt.axis("off")
```

Out[38]: (-0.5, 399.5, 199.5, -0.5)

