“KOSAC(한국어감성분석코퍼스) 감성사전

http://word.snu.ac.kr/kosac/lexicon.php

# 한국어 감성 분석기

<https://github.com/mrlee23/KoreanSentimentAnalyzer>

군산대

<http://dilab.kunsan.ac.kr/knusl.html>

https://github.com/park1200656/KnuSentiLex

논문:

https://www.koreascience.or.kr/article/CFKO201832073078582.pdf

관련 코드

#!/usr/bin/env python3

import socket, sys, json

from konlpy.tag import Kkma

from konlpy.utils import pprint

kkma = Kkma()

FINISH\_FLAG = bytes('\_\_finished\_\_', 'utf-8')

def checkFinished (data):

global FINISH\_FLAG

if data[-len(FINISH\_FLAG):] == FINISH\_FLAG:

return True

else:

return False

def eliminateFinished (data):

global FINISH\_FLAG

return data[:-len(FINISH\_FLAG)]

def sendData (connection, data):

global FINISH\_FLAG

try:

data = json.JSONEncoder().encode(data)

data = bytes(data, 'utf-8')+FINISH\_FLAG

connection.sendall(data)

return True

except:

print('[Send error]')

def recvData (connection):

global FINISH\_FLAG

try:

dataCollection = bytes('', 'utf-8')

while True:

data = connection.recv(30000)

print('Received data(length: {})'.format(len(data)))

if len(data) < 1:

break

dataCollection += data

if checkFinished(dataCollection):

dataCollection = eliminateFinished(dataCollection)

break

print('Collected data(length: {})'.format(len(dataCollection)))

if (len(dataCollection) > 0):

dataCollection = json.JSONDecoder().decode(dataCollection.decode())

return dataCollection

except:

print('[Recv error')

def parseData (data):

return parseKkma(data)

def parseKkma (data):

if (type(data) == str):

return kkma.pos(data)

elif (type(data) == list):

ret = []

for doc in data:

ret.append(parseKkma(doc))

return ret

def listen(host, port):

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_address = (host, port)

try:

print("[{}:{}] Starting up".format(\*server\_address))

sock.bind(server\_address)

sock.listen(10)

except:

print("[{}:{}] Network is busy. exit.".format(\*server\_address))

exit()

while True:

print("[{}:{}] Waiting for a connection.".format(\*server\_address))

connection, client\_address = sock.accept()

try:

print("[{}:{}] connection from".format(\*server\_address), client\_address)

while True:

data = recvData(connection)

if data:

parsedData = parseData(data)

del data

sendData(connection, parsedData)

del parsedData

else:

print("[{}:{}] no data from".format(\*server\_address), client\_address)

break

finally:

connection.close()

if \_\_name\_\_ == '\_\_main\_\_':

# listen(sys.argv[1], sys.argv[2])

if len(sys.argv) == 3:

host, port = sys.argv[1:]

listen(host, int(port))

else:

listen('localhost', 7000)