LAMPIRAN

1. Kode Program Pengambilan Data Twitter Berbasis Phython

```
to use this script you can pass the following attributes:
username: Username of a specific twitter account (without @)
since: The lower bound date (yyyy-mm-aa)
until: The upper bound date (yyyy-mm-aa)
 querysearch: A query text to be matched
 near: A reference location area from where tweets were generated
 within: A distance radius from "near" location (e.g. 15mi)
 maxtweets: The maximum number of tweets to retrieve
 toptweets: Only the tweets provided as top tweets by Twitter (no parameters required)
 output: A filename to export the results (default is "output_got.csv")

Examples:
# Example 1 - Get tweets by username [barackobama]
python Exporter.py --username "barackobama" --maxtweets 1

# Example 2 - Get tweets by query search [europe refugees]
python Exporter.py --querysearch "europe refugees" --maxtweets 1

# Example 3 - Get tweets by username and bound dates [barackobama, '2015-09-10', '2015-
09-12']
python Exporter.py --username "barackobama" --since 2015-09-10 --until 2015-09-12 --
maxtweets 1

# Example 4 - Get the last 10 top tweets by username
python Exporter.py --username "barackobama" --maxtweets 10 --toptweets

# Example 5 - Get tweets by query search and bound dates
python Exporter.py --querysearch "atma jogja" --since 2018-01-01 --until 2018-06-30

cara install :
- download and install python 3.7
- download and install notepad++
- buat folder di memory tertentu
- buka cmd, lalu cd ke foldernya dan ketik : git clone https://github.com/Jefferson-
Henrique/GetOldTweets-python.git
- python -m pip install --upgrade pip
- pip install --upgrade setuptools
- pip install --upgrade lxml
- pip install --upgrade pyquery
```

2. Kode Program N-gram Bagi kata Dengan Bahasa Java

```
package n.gram;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
```

```
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.*;
import java.util.function.Function;
import java.util.stream.Collectors;
import static java.util.stream.Collectors.counting;
import java.util.stream.Stream;
/**
 *
 * @author andjar
 */
public class BagiKata {
 public static List<String> ngrams( String str) {
    int n=1;
      List<String> ngrams = new ArrayList<String>();
      String[] words = str.split(" ");

      for (int i = 0; i < words.length - n + 1; i++)
      {

if(words[i].equalsIgnoreCase("belum")||words[i].equalsIgnoreCase("bukan")||words[i].eq
ualsIgnoreCase("tak")||words[i].equalsIgnoreCase("tanpa")||words[i].equalsIgnoreCase("t
anpa")||words[i].equalsIgnoreCase("tidak")||words[i].equalsIgnoreCase("pantang")||words
[i].equalsIgnoreCase("jangan")||words[i].equalsIgnoreCase("bukanlah")||words[i].equalsI
gnoreCase("gak")||words[i].equalsIgnoreCase("enggak"))
          {
             n=2;
             ngrams.add(concat(words, i, i+n));
             i++;
          }
          else
          {
             n=1;
             ngrams.add(concat(words, i, i+n));
          }
      }
      return ngrams;
   }

   public static String concat(String[] words, int start, int end) {
      StringBuilder sb = new StringBuilder();
      for (int i = start; i < end; i++)
         sb.append((i > start ? " " : "") + words[i]);
      return sb.toString();
   }

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) throws IOException {
      // TODO code application logic here
      //for (int n = 1; n <= 2; n++) {
      String temp="";
      String filePath = "E:\\error.txt"; // letak file yang akan di pecah per kata
```

```
        for (String ngram : ngrams(readLineByLineJava8( filePath )))
           {
              System.out.println(ngram); //menampilkan hasil output kata kata, setelah itu copy ke notepad lalu lanjut ke HitungKata.java

           }

      }
      private static String readLineByLineJava8(String filePath)
      {
         StringBuilder contentBuilder = new StringBuilder();
         try (Stream<String> stream = Files.lines( Paths.get(filePath), StandardCharsets.UTF_8))
         {
            stream.forEach(s -> contentBuilder.append(s).append("\n"));
         }
         catch (IOException e)
         {
            e.printStackTrace();
         }
         return contentBuilder.toString();
      }
         static void countWords(String word) throws IOException {
         Arrays.stream(word.split("[\\r\\n]+"))
         .collect(Collectors.groupingBy(Function.<String>identity(), TreeMap::new, counting())).entrySet()
         .forEach(System.out::println);
      }
}
```

3. Kode Program Hitung Kata

```
package n.gram;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Arrays;
import java.util.TreeMap;
import java.util.function.Function;
import java.util.stream.Collectors;
import static java.util.stream.Collectors.counting;

/**
 *
 * @author andjar
// */
public class HitungKata {
    public static void main(String[] args) throws FileNotFoundException, IOException {
```

```
    Path path = Paths.get("E:\\/bagi_kata_2014-2019.txt");    //file txt hasil dari output BagiKata.java
     countWords(path);

       //System.out.println(words);
   }

   static void countWords(final Path file) throws IOException {
   Arrays.stream(new                                    String(Files.readAllBytes(file), StandardCharsets.UTF_8).split("[\\r\\n]+")) //    \\W+
       .collect(Collectors.groupingBy(Function.<String>identity(),                TreeMap::new, counting())).entrySet()
       .forEach(System.out::println);
}
}
```

4. Kode Program Stop Word

```
package com.uttesh.exude.stemming;

import com.uttesh.exude.ExudeData;
import com.uttesh.exude.exception.InvalidDataException;
import static com.uttesh.exude.stemming.Stemmer.c;
import static com.uttesh.exude.stemming.Stemmer.path;
import static com.uttesh.exude.stemming.Stemmer.url;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

/**
 *
 * @author Andjar
 */
public class Stopwords {

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) throws InvalidDataException {
      // TODO code application logic here
    String sql = "SELECT * FROM Sheet1";
      System.out.println("Sedang diproses....");
      try
      {
         c=DriverManager.getConnection(url+path);
         System.out.println("Berhasil konekk database");
         Statement state = c.createStatement();
         ResultSet rs = state.executeQuery(sql);
         if(rs!=null)
         {
            while(rs.next())
```

```
            {
               int id = rs.getInt("ID");
               String username= rs.getString("username");
              String inputData = rs.getString("text");
              String tanggal = rs.getString("date");
              //int retweet = rs.getInt("retweet");
              //int favorites=rs.getInt("favorites");
              //String mention = rs.getString("mentions");
              //String hastags =rs.getString("hastags");
              //String geo_location=rs.getString("geo_location");

               String output =
ExudeData.getInstance().filterStoppingsKeepDuplicates(inputData);
              //String sql2="UPDATE Stopwords set Status = ? where ID =?"; //versi edit
               String sql2 = "INSERT into stop_2016 values(?,?,?,?)"; //versi insert
               System.out.println("output ID "+id+" : "+output);
                     PreparedStatement pStmt = c.prepareStatement(sql2);
                  // pStmt.setInt(1, c.GetRowDataTwitter());
                   pStmt.setInt(1, id);
                   pStmt.setString(2, username);
                   pStmt.setString(4, output);
                   pStmt.setString(3, tanggal);
                   //pStmt.setInt(5, retweet);
                   //pStmt.setInt(6, favorites);
                   //pStmt.setString(7, mention);
                   //pStmt.setString(8, hastags);
                   //pStmt.setString(9, geo_location);

                   pStmt.executeUpdate();
               }
            }
            rs.close();
            state.close();
             c.close();
             System.out.println("database ditutup");
          }
          catch(Exception EX)
          {
             System.out.println("Error Reading From database. . .");
             System.out.println(EX);
          }


          //mydao.closeConnection();


          //System.out.println("input : "+inputData);
          //versi file txt
         // String output2=ExudeData.getInstance().getSwearWords(inputData);
//          try {
//             String inputData = "E:\\testaja.txt";
//
//             File newTextFile = new File("E:\\testaja_ed.txt");
//
//             FileWriter fw = new FileWriter(newTextFile);
```

```
//            String output = ExudeData.getInstance().filterStoppingsKeepDuplicates(inputData);
//            fw.write(output);
//            fw.close();
//
//        } catch (IOException iox) {
//            //do stuff with exception
//            iox.printStackTrace();
//        }

 //System.out.println("output : "+output2);
    }

}
```

5. Stoplist Bahasa Indonesia

ada	dia	makanya	semua
adanya	dialah	makin	semuanya
adalah	dini	malah	sendiri
adapun	diri	malahan	sendirinya
agak	dirinya	mampu	seolah
agaknya	terdiri	mampukah	seperti
agar	dong	mana	sepertinya
akan	dulu	manakala	sering
akankah	enggak	manalagi	seringnya
akhirnya	enggaknya	masih	serta
aku	entah	masihkah	siapa
akulah	entahlah	semasih	siapakah
amat	terhadap	masing	siapapun
amatlah	terhadapnya	mau	disini
anda	hal	maupun	disinilah
andalah	hampir	semaunya	sini
antar	hanya	memang	sinilah
diantaranya	hanyalah	mereka	sesuatu
antara	harus	merekalah	sesuatunya
antaranya	haruslah	meski	suatu
diantara	harusnya	meskipun	sesudah
apa	seharusnya	semula	sesudahnya
apaan	hendak	mungkin	sudah
mengapa	hendaklah	mungkinkah	sudahkah
apabila	hendaknya	nah	sudahlah
apakah	hingga	namun	supaya
apalagi	sehingga	nanti	tadi
apatah	ia	nantinya	tadinya
atau	ialah	nyaris	tak
ataukah	ibarat	oleh	tanpa
ataupun	ingin	olehnya	setelah
bagai	inginkah	seorang	telah
bagaikan	inginkan	seseorang	tentang
sebagai	ini	pada	tentu
sebagainya	inikah	padanya	tentulah
bagaimana	inilah	padahal	tentunya
bagaimanapun	itu	paling	tertentu
sebagaimana	itukah	sepanjang	seterusnya

bagaimanakah
bagi
bahkan
bahwa
bahwasanya
sebaliknya
banyak
sebanyak
beberapa
seberapa
begini
beginian
beginikah
beginilah
sebegini
begitu
begitukah
begitulah
begitupun
sebegitu
belum
belumlah
sebelum
sebelumnya
sebenarnya
berapa
berapakah
berapalah
berapapun
betulkah
sebetulnya
biasa
biasanya
bila
bilakah
bisa
bisakah
sebisanya
boleh
bolehkah
bolehlah
buat
bukan
bukankah
bukanlah
bukannya
cuma
percuma
dahulu
dalam
dan
dapat
dari
daripada
dekat
itulah
jangan
jangankan
janganlah
jika
jikalau
juga
justru
kala
kalau
kalaulah
kalaupun
kalian
kami
kamilah
kamu
kamulah
kan
kapan
kapankah
kapanpun
dikarenakan
karena
karenanya
ke
kecil
kemudian
kenapa
kepada
kepadanya
ketika
seketika
khususnya
kini
kinilah
kiranya
sekiranya
kita
kitalah
kok
lagi
lagian
selagi
lah
lain
lainnya
melainkan
selaku
lalu
melalui
terlalu
lama
lamanya
selama
selama
pantas
sepantasnya
sepantasnyalah
para
pasti
pastilah
per
pernah
pula
pun
merupakan
rupanya
serupa
saat
saatnya
sesaat
saja
sajalah
saling
bersama
sama
sesama
sambil
sampai
sana
sangat
sangatlah
saya
sayalah
se
sebab
sebabnya
sebuah
tersebut
tersebutlah
sedang
sedangkan
sedikit
sedikitnya
segala
segalanya
segera
sesegera
sejak
sejenak
sekali
sekalian
sekalipun
sesekali
sekaligus
sekarang
sekarang
sekitar
sekitarnya
sela
tapi
tetapi
setiap
tiap
setidaknya
tidak
tidakkah
tidaklah
toh
waduh
wah
wahai
sewaktu
walau
walaupun
wong
yaitu
yakni
yang
depan
di
dengan
semacam
macam
maka
sementara
sempat
semakin

demi	selamanya	selain
demikian	lebih	selalu
demikianlah	terlebih	seluruh
sedemikian	bermacam	seluruhnya

6. Kode Program Sentistrength

```
# coding: utf-8

import re
from collections import OrderedDict
import numpy as np

class sentistrength:
    def __init__(self, config=dict()):
        self.negasi = [line.replace('\n','') for line in open("negatingword.txt").read().splitlines()]
        self.tanya = [line.replace('\n','') for line in open("questionword.txt").read().splitlines()]
        #create sentimen words dictionary
        self.sentiwords_txt = [line.replace('\n','').split(":") for line in
open("sentiwords_id.txt").read().splitlines()]
        self.sentiwords_dict = OrderedDict()
        for term in self.sentiwords_txt:
            self.sentiwords_dict[term[0]] = int(term[1])
        #create emoticon dictionary
        self.emoticon_txt = [line.replace('\n','').split(" | ") for line in
open("emoticon_id.txt").read().splitlines()]
        self.emoticon_dict = OrderedDict()
        for term in self.emoticon_txt:
            self.emoticon_dict[term[0]] = int(term[1])
        #create idioms dictionary
        self.idioms_txt = [line.replace('\n','').split(":") for line in
open("idioms_id.txt").read().splitlines()]
        self.idioms_dict = OrderedDict()
        for term in self.idioms_txt:
            self.idioms_dict[term[0]] = int(term[1])
        #create boosterwords dictionary
        self.boosterwords_txt = [line.replace('\n','').split(":") for line in
open("boosterwords_id.txt").read().splitlines()]
        self.boosterwords_dict = OrderedDict()
        for term in self.boosterwords_txt:
            self.boosterwords_dict[term[0]] = int(term[1])
        self.negation_conf = config["negation"]
        self.booster_conf = config["booster"]
        self.ungkapan_conf = config["ungkapan"]
        self.consecutive_conf = config["consecutive"]
        self.repeated_conf = config["repeated"]
        self.emoticon_conf = config["emoticon"]
        self.question_conf = config["question"]
        self.exclamation_conf = config["exclamation"]
        self.punctuation_conf = config["punctuation"]
        self.mean_conf = False

    def senti(self,term):
        try:
            return self.sentiwords_dict[term]
```

```
    except:
        return 0

def emosikon(self,term):
    try:
        return self.emoticon_dict[term]
    except:
        return 0

def ungkapan(self,term):
    try:
        return self.idioms_dict[term]
    except:
        return 0

def booster(self, term):
    try:
        return self.boosterwords_dict[term]
    except:
        return 0

def cek_negationword(self, prev_term, prev_term2):
    #jika kata sebelumnya (index-1) adalah kata negasi, negasikan nilai -+nya
    if prev_term in self.negasi or prev_term2+" "+prev_term in self.negasi:
        # print prev_term
        self.score = -abs(self.score) if self.score>0 else abs(self.score)

def cek_boosterword(self,term):
    booster_score = self.booster(term)
    if booster_score !=0 and self.score>0: self.score += booster_score
    if booster_score !=0 and self.score<0: self.score -= booster_score

def cek_consecutive_term(self, prev_term):
    if self.prev_score>0 and self.score >=3: self.score+=1
    if self.prev_score<0 and self.score <=-3: self.score-=1

def cek_ungkapan(self, bigram,trigram, i):
    bigram = ' '.join(bigram)
    trigram = ' '.join(trigram)
    ungkapan_score = self.ungkapan(bigram)
    if ungkapan_score==0:
        ungkapan_score = self.ungkapan(trigram)
    if ungkapan_score!=0:
        self.score = ungkapan_score
        self.prev_score = 0
        self.pre_max_pos[i-1] = 1
        self.pre_max_neg[i-1] = -1
        self.max_pos = self.pre_max_pos[i-2] #if len(self.pre_max_pos)>1 else 1
        self.max_neg = self.pre_max_neg[i-2] #if len(self.pre_max_neg)>1 else -1
        self.sentence_score[i-1] = re.sub(r'\[\d\]','',self.sentence_score[i-1])

def cek_repeated_punctuation(self, next_term):
    if re.search(r'!{2,}',next_term) and self.score >=3: self.score+=1
    if re.search(r'!{2,}',next_term) and self.score <=-3: self.score-=1
```

```
def remove_extra_repeated_char(self, term):
    return re.sub(r'([A-Za-z])\1{2,}',r'\1',term)
def plural_to_singular(self, term):
    return re.sub(r'([A-Za-z]+)\-\1', r'\1',term)
def classify(self):
    result = "neutral"
    try:
        if self.mean_conf:
            mean_p = np.mean(self.mean_pos)
            mean_n = np.mean(self.mean_neg)
            print mean_p, mean_n
            if mean_p > mean_n:
                result = "positive"
            elif mean_p < mean_n and not self.is_tanya:
                result = "negative"
            elif mean_p < mean_n and self.is_tanya:
                result = "neutral"
        else:
            if abs(self.sentences_max_pos) > abs(self.sentences_max_neg):
                result = "positive"
            elif abs(self.sentences_max_pos) < abs(self.sentences_max_neg):
                result = "negative"
            elif abs(self.sentences_max_pos) == abs(self.sentences_max_neg):
                result = "neutral"
    except:
        print "error ",self.sentences_max_pos, self.sentences_max_neg
    return result
def cek_neutral_term(self,terms,i):
    if terms[i-1] in self.neutral_term or terms[i+1] in self.neutral_term: self.score=1

def main(self,sentence):
    self.neutral_term = ['jika','kalau']
    sentences = sentence.split('.')
    self.sentences_max_neg = -1
    self.sentences_max_pos = 1
    self.sentences_score = []
    self.sentences_text = []
    for sentence in sentences:
        self.max_neg = -1
        self.max_pos = 1
        self.mean_neg = [1]
        self.mean_pos = [1]
        self.sentence_score=[]
        terms = sentence.split()
        # terms = re.split(r'[\s,.]',sentence)
        terms_length = len(terms)
        self.is_tanya = False
        self.sentence_text = ''
        # print self.max_pos, self.max_neg
        #SEMUA KALIMAT YANG MEMILIKI TANDA SERU MEMILIKI +ve minimal
2
        if self.exclamation_conf and re.search('!',sentence): self.max_pos = 2
        self.prev_score = 0
        self.pre_max_pos = []
        self.pre_max_neg = []
```

```
        for i,term in enumerate(terms):
            # repeated_term = ''
            is_extra_char = False
            plural = ''
            self.score = 0
            # if re.search(r'[A-Za-z\-.]+',term):
            # print term
            if re.search(r'([A-Za-z])\1{3,}',term):
                is_extra_char = True
                # repeated_term =term
            term = self.remove_extra_repeated_char(term)
            if re.search(r'([A-Za-z]+)\-\1',term):
                plural = term
                term = self.plural_to_singular(term)
            #GET SENTI SCORE#
            self.score = self.senti(term)
            # print "senti score",term, self.score

            #NEGATION HANDLER#
            if self.negation_conf and self.score !=0 and i>0:self.cek_negationword(terms[i-
1],terms[i-2])
            # print  "negation score",term, self.score

            #BOOSTERWORD HANDLER#
            if self.booster_conf and self.score !=0 and i>0 and i<=(terms_length-
1):self.cek_boosterword(terms[i-1])
            if self.booster_conf and self.score !=0 and i>=0 and i<(terms_length-
1):self.cek_boosterword(terms[i+1])
            # print  "booster score",term, self.score

            #IDIOM/UNGKAPAN HANDLER#
            if self.ungkapan_conf and i>0 and i<=(terms_length-
1):self.cek_ungkapan([terms[i-1],term],[terms[i-2],terms[i-1],term],i)
            # if self.ungkapan_conf and i>=0 and i<(terms_length-
1):self.cek_ungkapan([term,terms[i+1]])
            # print  "idiom score",term, self.score

            #CONSECUTIVE SENTIMEN WORD#
            if self.consecutive_conf and i>0 and i<=(terms_length-1) and self.score
!=0:self.cek_consecutive_term(terms[i-1])
            # print  "consecutive score",term, self.score

            #+1 SENTI SCORE IF REPEATED CHAR ON POSITIVE/NEGATIVE +2 IF
NEUTRAL TERM
            if self.repeated_conf and is_extra_char==True and self.score>0: self.score+=1
            if self.repeated_conf and is_extra_char==True and self.score<0: self.score-=1
            if self.repeated_conf and is_extra_char==True and self.score==0: self.score=2
            # print  "repeat char score", term, self.score
            if self.punctuation_conf and i>=0 and i<(terms_length-1):
self.cek_repeated_punctuation(terms[i+1])
            # CEK APAKAH TERDAPAT KATA TANYA
            if self.question_conf and (term in self.tanya or re.search(r'\?',term)):self.is_tanya =
True
            # CEK neutral term
            if self.score!=0 and i>1 and i<(terms_length-2): self.cek_neutral_term(terms,i)
```

```
            # if self.score!=0 and i>0 and i<(terms_length-4): self.cek_neutral_term(terms,i)
            if self.emoticon_conf and self.score==0: self.score = self.emosikon(term)

            self.prev_score = self.score
            if self.mean_conf and self.score>0: self.mean_pos.append(self.score)
            if self.mean_conf and self.score<0: self.mean_neg.append(abs(self.score))
            #GET MAX SCORE +ve/-ve
            self.max_pos= self.score if self.score > self.max_pos else self.max_pos
            self.max_neg= self.score if self.score < self.max_neg else self.max_neg
            #insert score info current term
            self.pre_max_pos.append(self.max_pos)
            self.pre_max_neg.append(self.max_neg)
            # print self.pre_max_pos, self.pre_max_neg
            if plural !="": term = plural
            self.sentence_text += ' {}'.format(term)
            if self.score != 0:term = "{} [{}]".format(term, self.score)
            self.sentence_score.append(term)

        self.sentences_text.append(self.sentence_text)
        self.sentences_score.append(" ".join(self.sentence_score))
        if self.is_tanya:
            self.max_neg = -1
        self.sentences_max_pos = self.max_pos if self.max_pos > self.sentences_max_pos
else self.sentences_max_pos
        self.sentences_max_neg = self.max_neg if self.max_neg < self.sentences_max_neg
else self.sentences_max_neg
        # print self.sentences_max_pos, self.sentences_max_neg
      sentence_result = self.classify()
      # print self.sentences_text
      #return {"classified_text":". ".join(self.sentences_score),"RESULT :
":self.sentences_max_pos+self.sentences_max_neg ,"tweet_text":".
".join(self.sentences_text),"sentence_score":self.sentences_score,"max_positive":self.sentenc
es_max_pos,"max_negative":self.sentences_max_neg,"kelas":sentence_result}

     # return {"RESULT : ": self.sentences_max_pos + self.sentences_max_neg,
      #       "max_positive": self.sentences_max_pos,
       #       "max_negative": self.sentences_max_neg,
        #      "kelas": sentence_result}
      return {"sentimen": sentence_result}
     # return{sentence_result}
config = dict()
config["negation"] = True
config["booster"]  = True
config["ungkapan"]  = True
config["consecutive"]  = True
config["repeated"]  = True
config["emoticon"]  = True
config["question"]  = True
config["exclamation"]  = True
config["punctuation"]  = True
senti = sentistrength(config)

list1= ['agnezmo22222 malas dan jelek sekali tetapi lintah darat :)','Maafkan aku Ham, aku
udah bener-bener gak bisa sama kamu, aku tuh udah terlanjur mencintai dia, bahkan lebih
dari cinta aku ke kamu, maaf','Aku benar-benar mencintaimu tapi tidak suka adik dingin
```

```
Anda'];
list2=[line.strip() for line in open("E:/robert.txt", 'r')]; #E:/00. SKRIPSIIIIIIIIIIIIIIIII
FIIXXXXXXXXXXX/program/DATA/novdesss1.txt
#print senti.main("agnezmo malas dan jelek sekali tetapi lintah darat :)")
#print senti.main("Maafkan aku Ham, aku udah bener-bener gak bisa sama kamu, aku tuh
udah terlanjur mencintai dia, bahkan lebih dari cinta aku ke kamu, maaf ")
#print senti.main("Aku benar-benar mencintaimu tapi tidak suka adik dingin Anda.")
for p in list2: print senti.main(p)

#print senti.main(list1)
```

7. Query Pemberian kode Untuk Korpus

```
# coding: utf-8

import re
from collections import OrderedDict
import numpy as np

class sentistrength:
    def __init__(self, config=dict()):
        self.negasi = [line.replace('\n','') for line in open("negatingword.txt").read().splitlines()]
        self.tanya = [line.replace('\n','') for line in open("questionword.txt").read().splitlines()]
        #create sentimen words dictionary
        self.sentiwords_txt = [line.replace('\n','').split(":") for line in
open("sentiwords_id.txt").read().splitlines()]
        self.sentiwords_dict = OrderedDict()
        for term in self.sentiwords_txt:
            self.sentiwords_dict[term[0]] = int(term[1])
        #create emoticon dictionary
        self.emoticon_txt = [line.replace('\n','').split(" | ") for line in
open("emoticon_id.txt").read().splitlines()]
        self.emoticon_dict = OrderedDict()
        for term in self.emoticon_txt:
            self.emoticon_dict[term[0]] = int(term[1])
        #create idioms dictionary
        self.idioms_txt = [line.replace('\n','').split(":") for line in
open("idioms_id.txt").read().splitlines()]
        self.idioms_dict = OrderedDict()
        for term in self.idioms_txt:
            self.idioms_dict[term[0]] = int(term[1])
        #create boosterwords dictionary
        self.boosterwords_txt = [line.replace('\n','').split(":") for line in
open("boosterwords_id.txt").read().splitlines()]
        self.boosterwords_dict = OrderedDict()
        for term in self.boosterwords_txt:
            self.boosterwords_dict[term[0]] = int(term[1])
        self.negation_conf = config["negation"]
        self.booster_conf = config["booster"]
        self.ungkapan_conf = config["ungkapan"]
        self.consecutive_conf = config["consecutive"]
        self.repeated_conf = config["repeated"]
        self.emoticon_conf = config["emoticon"]
```

```
    self.question_conf = config["question"]
    self.exclamation_conf = config["exclamation"]
    self.punctuation_conf = config["punctuation"]
    self.mean_conf = False

def senti(self,term):
    try:
        return self.sentiwords_dict[term]
    except:
        return 0

def emosikon(self,term):
    try:
        return self.emoticon_dict[term]
    except:
        return 0

def ungkapan(self,term):
    try:
        return self.idioms_dict[term]
    except:
        return 0

def booster(self, term):
    try:
        return self.boosterwords_dict[term]
    except:
        return 0

def cek_negationword(self, prev_term, prev_term2):
    #jika kata sebelumnya (index-1) adalah kata negasi, negasikan nilai -+nya
    if prev_term in self.negasi or prev_term2+" "+prev_term in self.negasi:
        # print prev_term
        self.score = -abs(self.score) if self.score>0 else abs(self.score)

def cek_boosterword(self,term):
    booster_score = self.booster(term)
    if booster_score !=0 and self.score>0: self.score += booster_score
    if booster_score !=0 and self.score<0: self.score -= booster_score

def cek_consecutive_term(self, prev_term):
    if self.prev_score>0 and self.score >=3: self.score+=1
    if self.prev_score<0 and self.score <=-3: self.score-=1

def cek_ungkapan(self, bigram,trigram, i):
    bigram = ' '.join(bigram)
    trigram = ' '.join(trigram)
    ungkapan_score = self.ungkapan(bigram)
    if ungkapan_score==0:
        ungkapan_score = self.ungkapan(trigram)
    if ungkapan_score!=0:
        self.score = ungkapan_score
        self.prev_score = 0
        self.pre_max_pos[i-1] = 1
        self.pre_max_neg[i-1] = -1
```

```
        self.max_pos = self.pre_max_pos[i-2] #if len(self.pre_max_pos)>1 else 1
        self.max_neg = self.pre_max_neg[i-2] #if len(self.pre_max_neg)>1 else -1
        self.sentence_score[i-1] = re.sub(r'\[\d\]','',self.sentence_score[i-1])

def cek_repeated_punctuation(self, next_term):
    if re.search(r'!{2,}',next_term) and self.score >=3: self.score+=1
    if re.search(r'!{2,}',next_term) and self.score <=-3: self.score-=1

def remove_extra_repeated_char(self, term):
    return re.sub(r'([A-Za-z])\1{2,}',r'\1',term)
def plural_to_singular(self, term):
    return re.sub(r'([A-Za-z]+)\-\1', r'\1',term)
def classify(self):
    result = "neutral"
    try:
        if self.mean_conf:
            mean_p = np.mean(self.mean_pos)
            mean_n = np.mean(self.mean_neg)
            print mean_p, mean_n
            if mean_p > mean_n:
                result = "positive"
            elif mean_p < mean_n and not self.is_tanya:
                result = "negative"
            elif mean_p < mean_n and self.is_tanya:
                result = "neutral"
        else:
            if abs(self.sentences_max_pos) > abs(self.sentences_max_neg):
                result = "positive"
            elif abs(self.sentences_max_pos) < abs(self.sentences_max_neg):
                result = "negative"
            elif abs(self.sentences_max_pos) == abs(self.sentences_max_neg):
                result = "neutral"
    except:
        print "error ",self.sentences_max_pos, self.sentences_max_neg
    return result
def cek_neutral_term(self,terms,i):
    if terms[i-1] in self.neutral_term or terms[i+1] in self.neutral_term: self.score=1

def main(self,sentence):
    self.neutral_term = ['jika','kalau']
    sentences = sentence.split('.')
    self.sentences_max_neg = -1
    self.sentences_max_pos = 1
    self.sentences_score = []
    self.sentences_text = []
    for sentence in sentences:
        self.max_neg = -1
        self.max_pos = 1
        self.mean_neg = [1]
        self.mean_pos = [1]
        self.sentence_score=[]
        terms = sentence.split()
        # terms = re.split(r'[\s,.]',sentence)
        terms_length = len(terms)
        self.is_tanya = False
```

```
        self.sentence_text = ''
        # print self.max_pos, self.max_neg
        #SEMUA KALIMAT YANG MEMILIKI TANDA SERU MEMILIKI +ve minimal
        if self.exclamation_conf and re.search('!',sentence): self.max_pos = 2
        self.prev_score = 0
        self.pre_max_pos = []
        self.pre_max_neg = []
        for i,term in enumerate(terms):
            # repeated_term = ''
            is_extra_char = False
            plural = ''
            self.score = 0
            # if re.search(r'[A-Za-z\-.]+',term):
            # print term
            if re.search(r'([A-Za-z])\1{3,}',term):
                is_extra_char = True
                # repeated_term =term
            term = self.remove_extra_repeated_char(term)
            if re.search(r'([A-Za-z]+)\-\1',term):
                plural = term
                term = self.plural_to_singular(term)
            #GET SENTI SCORE#
            self.score = self.senti(term)
            # print "senti score",term, self.score

            #NEGATION HANDLER#
            if self.negation_conf and self.score !=0 and i>0:self.cek_negationword(terms[i-
1],terms[i-2])
            # print  "negation score",term, self.score

            #BOOSTERWORD HANDLER#
            if self.booster_conf and self.score !=0 and i>0 and i<=(terms_length-
1):self.cek_boosterword(terms[i-1])
            if self.booster_conf and self.score !=0 and i>=0 and i<(terms_length-
1):self.cek_boosterword(terms[i+1])
            # print  "booster score",term, self.score

            #IDIOM/UNGKAPAN HANDLER#
            if self.ungkapan_conf and i>0 and i<=(terms_length-
1):self.cek_ungkapan([terms[i-1],term],[terms[i-2],terms[i-1],term],i)
            # if self.ungkapan_conf and i>=0 and i<(terms_length-
1):self.cek_ungkapan([term,terms[i+1]])
            # print  "idiom score",term, self.score

            #CONSECUTIVE SENTIMEN WORD#
            if self.consecutive_conf and i>0 and i<=(terms_length-1) and self.score
!=0:self.cek_consecutive_term(terms[i-1])
            # print  "consecutive score",term, self.score

            #+1 SENTI SCORE IF REPEATED CHAR ON POSITIVE/NEGATIVE +2 IF
NEUTRAL TERM
            if self.repeated_conf and is_extra_char==True and self.score>0: self.score+=1
            if self.repeated_conf and is_extra_char==True and self.score<0: self.score-=1
            if self.repeated_conf and is_extra_char==True and self.score==0: self.score=2
            # print  "repeat char score", term, self.score
```

```
            if self.punctuation_conf and i>=0 and i<(terms_length-1):
self.cek_repeated_punctuation(terms[i+1])
            # CEK APAKAH TERDAPAT KATA TANYA
            if self.question_conf and (term in self.tanya or re.search(r'\?',term)):self.is_tanya =
True
            # CEK neutral term
            if self.score!=0 and i>1 and i<(terms_length-2): self.cek_neutral_term(terms,i)
            # if self.score!=0 and i>0 and i<(terms_length-4): self.cek_neutral_term(terms,i)
            if self.emoticon_conf and self.score==0: self.score = self.emosikon(term)

            self.prev_score = self.score
            if self.mean_conf and self.score>0: self.mean_pos.append(self.score)
            if self.mean_conf and self.score<0: self.mean_neg.append(abs(self.score))
            #GET MAX SCORE +ve/-ve
            self.max_pos= self.score if self.score > self.max_pos else self.max_pos
            self.max_neg= self.score if self.score < self.max_neg else self.max_neg
            #insert score info current term
            self.pre_max_pos.append(self.max_pos)
            self.pre_max_neg.append(self.max_neg)
            # print self.pre_max_pos, self.pre_max_neg
            if plural !='': term = plural
            self.sentence_text += ' {}'.format(term)
            if self.score != 0:term = "{} [{}]".format(term, self.score)
            self.sentence_score.append(term)

        self.sentences_text.append(self.sentence_text)
        self.sentences_score.append(" ".join(self.sentence_score))
        if self.is_tanya:
            self.max_neg = -1
        self.sentences_max_pos = self.max_pos if self.max_pos > self.sentences_max_pos
else self.sentences_max_pos
        self.sentences_max_neg = self.max_neg if self.max_neg < self.sentences_max_neg
else self.sentences_max_neg
        # print self.sentences_max_pos, self.sentences_max_neg
      sentence_result = self.classify()
      # print self.sentences_text
      #return {"classified_text":". ".join(self.sentences_score),"RESULT :
":self.sentences_max_pos+self.sentences_max_neg ,"tweet_text":".
".join(self.sentences_text),"sentence_score":self.sentences_score,"max_positive":self.sentenc
es_max_pos,"max_negative":self.sentences_max_neg,"kelas":sentence_result}

     # return {"RESULT : ": self.sentences_max_pos + self.sentences_max_neg,
      #       "max_positive": self.sentences_max_pos,
       #      "max_negative": self.sentences_max_neg,
        #     "kelas": sentence_result}
      return {"sentimen": sentence_result}
     # return{sentence_result}
config = dict()
config["negation"] = True
config["booster"]  = True
config["ungkapan"]  = True
config["consecutive"]  = True
config["repeated"]  = True
config["emoticon"]  = True
config["question"]  = True
```

```
config["exclamation"]  = True
config["punctuation"]  = True
senti = sentistrength(config)

list1= ['agnezmo22222 malas dan jelek sekali tetapi lintah darat :)','Maafkan aku Ham, aku udah bener-bener gak bisa sama kamu, aku tuh udah terlanjur mencintai dia, bahkan lebih dari cinta aku ke kamu, maaf','Aku benar-benar mencintaimu tapi tidak suka adik dingin Anda'];
#print senti.main(list1)
```

8. Contoh Hasil Penyusunan Korpus

	Tanggible			**reliability**			**reponsive**	
teknologi:1	fasilitas:2	fisik:3		kecepatan:1	kesalahan:2		cepat:1	cepat:2
alat:1	sarana:2	karyawan:3		ketepatan:1	sikap:2		responsif:1	penyampain:2
colokan:1	peralatan:2	tampilan:3		waktu:1	simpatik:2		penyampain:1	jelas:2
kabel:1	penampilan:2	seragam:3		kesalahan:1	akurat:2		jelas:1	cekatan:2
ruter:1	fisik:2	kostum:3		kemampuan:1	kehebatan:2		ngerti:1	info:2
roter:1	bangunan:2	baju:3		pelayanan:1	kejempolan:2		cekatan:1	kuliahpengganti:2
lab:1	kelas:2	celana:3		kesetiaan:1	kemahiran:2		mohon:1	tanggap:2
lampu:1	kls:2	sepatu:3		ketaatan:1	kepandain:2		dosen:1	kinerja:2
keran:1	kampus:2	satpam:3		keunggulan:1	kepercayaan:2		dsn:1	tindakan:2
KTM:1	universitas:2	cleaning service:3		sesi:1	keterampilan:2		menyesuaikan:1	dimengerti:2
lift:1	ruang:2	CS:3		dosen:1	keunggulan:2		mahasiswa:1	dipahami:2
proyektor:1	auditorium:2	rapi:3		dsn:1	kosong:2		karyawan:1	paham:2
komputer:1	prodi:2	bersih:3		kuliahpengganti:1	ujian:2		perkeuliahan:1	ngerti:2
keyboard:1	uajy:2	wangi:3		informasi:1	workshop:2		feedback:1	lambat:2
mouse:1	kampus:2	dosen:3		kuliahkosong:1	lama:2		ngerti:1	slow:2
speker:1	parkir:2	rektor:3		infokuliahpengganti:1	baru:2		nanya:1	lemot:2
monitor:1	aquinas:2			ujian:1	menjelaskan:2		bertanya:1	lama:2
dispenser:1	fakultas:2			workshop:1	harapan:2		mengerti:1	terlama:2
cip:1	gedung:2			kemudahan:1	berharap:2		mudeng:1	tercepat:2
tumbler:1	tampilan:2			mudah:1	harap:2		singkat:1	terlama:2
ATM:1	atap:2			akses:1	sosialisasi:2		padat:1	selow:2
layar:1	lobi:2			mengakses:1	penyuluhan:2		terbuka:1	respon:2
laptop:1	teras:2			situs:1	memberitakan:2		respon:1	kritis:2
computer:1	parkiran:2			siatma:1	berita:2		spesifik:1	peka:2
mic:1	kantin:2			jasa:1	kabar:2		tegas:1	reaktif:2
mirophone:1	lantai:2			kinerja:1	mengabarakan:2		terjelas:1	aktif:2

9. Contoh Data Status Twitter

	username	date	retwe	favo	text	geo	mentions	hashtags
2	emmeliapa	30/06/2016 06:12	0	1	@ uajy min kok siatma ku error ya pas buat ngeklik evaluasi dosen ? Padahal kalo klik yang lain		@	
3	christinedk	28/01/2016 12:46	0	1	@ uajy min ni aku liat nilai lwt ortu ipnya naik, tp kok di siatma yg bagian transkrip nilai, blm be		@	
4	brisnipdan	07/01/2016 10:29	0	0	@achy_moci pass ku itu salah kalo buka remedi uajy. Tapi kalo buka siatma nggak salah itu aku		@achy_moci	
5	brisnipdan	07/01/2016 09:51	0	1	@ uajy min mau nanyak ? Knapa buka remdi uajy nggak bisa yaa ? Sedangkan buka siatma bisa		@	
6	bintangyap	04/01/2016 18:42	0	0	Min @ uajy kok banyak nilai yang belum dimuat di situs siatma? saya sudah update data dan ku		@	
7	ReginaAyu	01/01/2016 09:51	0	1	@ uajy min mau nanya. situs siatma buat upload KK kok error trs ya? solusinya gmn? padahal s		@	
8	rafifnugrah	20/12/2016 23:30	0	1	@ uajy @ ksiuajy min ini situs siatma down mulu, gimana nih?		@ @	
9	antoniusbr	07/11/2016 00:40	0	2	@ uajy ini kenapa siatma di presensi masih pertemuan 8/9 aja yak padahal terhitung kelas sepe		@	
10	fani_christ	16/08/2016 14:11	0	1	SIATMA error (at @ uajy) â€" https://path.com/p/fZ5CC		@	
11	Aminsabia	08/08/2016 09:19	0	0	Coba cek --> siatma(dot)uajy(dot)ac(dot) https://twitter.com/theocahya/status/762472574764789760 â€¦			
12	jardineep /	18/12/2016 13:49	0	0	haii untuk konferensinya 1 tim brp orang yahh? Thankyou :)			
13	hmpsafeua	12/12/2016 11:04	0	0	Keluarga Besar HMPSA FE UAJY mengucapkan Selamat Hari Maulid Nabi Muhammad SAW 12 Rabiul Awal #			
14	redaksimj	11/12/2016 23:39	0	0	regram @septridoo [SEMINAR NASIONAL 2017] SENAT MAHASISWA FE UAJY PROUDLY PRESE		@septridoo	
15	fe_felz	07/12/2016 12:29	0	0	With prince charles, Adrian, and 5 others at Basement Kampus 2 UAJY â€" https://path.com/p/39g2tm			
16	hmpsafeua	04/12/2016 15:27	0	0	Kunjungan HMPSA FE UAJY ke HIMA Fakultas Ekonomi dan Bisnis UMY pada Sabtu, 03 Desember 2016. # I #			
17	hmpsafeua	04/12/2016 15:27	0	0	Kunjungan HMPSA FE UAJY ke HIMA Fakultas Ekonomi dan Bisnis UMY pada Sabtu, 03 Desember 2016. # I # #			
18	FandySoa1	03/12/2016 18:33	0	0	Kamajaya Futsal FE UAJY. (with Leon, Deka, and 4 others at Forza Futsal) [pic] â€" https://path.com/p/2g7XoR			
19	posteryk	03/12/2016 15:34	0	0	Regrann from @ vnctianih - [SEMINAR NASIONAL 2017] SENAT MAHASISWA FE UAJY PROUDL		@	
20	HIMAFEBU	03/12/2016 10:43	1	3	[live report] Kunjungan dari HMPSA FE UAJY ke HIMA FEB UMY dalam rangka Studi Banding. Semangat!! pic.twitter.c			
21	Jimmy_Sar	03/12/2016 07:48	0	0	Nerjang hujan demi kuliah pengganti (@Ruang 105 FE UAJY in Yogyakarta) https://www.swarm		@Ruang	
22	emmiliaku	02/12/2016 12:09	0	0	Bergetar hati ini saat melihat dirimu, padahal bapak mainan komputer tapiâ€¦ (at Ruang Kaprodi Akuntansi FE UAJY)			
23	Jimmy_Sar	02/12/2016 08:13	0	0	Tekor perjalanan rumah kampus 1,5 jam (@Ruang 403 FE UAJY in Yogyakarta, Indonesia) https		@Ruang	
24	Jimmy_Sar	29/11/2016 08:19	0	0	Ternyata ac nya dingin banget (@Rk 411 FE UAJY) https://www.swarmapp.com/c/lMUgi2tZx0		@Rk	
25	hmpsafeua	26/11/2016 19:40	0	0	Kebersamaan SLB Helen Keller Indonesia-Yogyakarta dengan HMPSA FE UAJY tadi siang. # HMPSAsatu # B # #			
26	hmpsafeua	26/11/2016 19:40	0	0	Kebersamaan SLB Helen Keller Indonesia-Yogyakarta dengan HMPSA FE UAJY tadi siang. # HMPSAsatu # B # # #			
27	fe_felz	24/11/2016 16:18	0	0	Presentasi Rab insyayesus (with Monica, eni, and handy at Ruang 2402 FT. Architecture UAJY) â€" https://path.com/			
28	Jimmy_Sar	24/11/2016 13:46	0	0	Ppt nya full animasi (with Yuli at Rk 414 FE UAJY) â€" https://path.com/p/3TyHeO			