LAMPIRAN

1. Query Program untuk Data Collecting dengan menggunakan python 3.7 dengan tools cmd

```
python Exporter.py --querysearch "tugu jogja" --since 2015-09-10 --until 2015-09-12
```

2. Kode Program *Stopwords*

```
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 {

   try {

            String inputData = "E:\\STOPWORDDATE.txt";


            File newTextFile = new File("E:\\STOPWORDDATE2.txt");


            FileWriter fw = new FileWriter(newTextFile);

            String output =
ExudeData.getInstance().filterStoppingsKeepDuplicates(inputData);

            fw.write(output+"\n");

            fw.close();

        } catch (IOException iox) {
```

```
    //do stuff with exception
    iox.printStackTrace();
}
```

3. 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
akulah
amat
amatlah
anda
andalah
antar
diantaranya
antara
antaranya
diantara
apa
apaan
mengapa
apabila
apakah
apalagi
apatah
atau
ataukah
ataupun
bagai
bagaikan
sebagai
sebagainya
bagaimana
bagaimanapun
sebagaimana
bagaimanakah
bagi
bahkan
bahwa
bahwasanya
sebaliknya
banyak
sebanyak
beberapa
seberapa
begini
beginian
beginikah
beginilah
sebegini
begitu
begitukah
begitulah
entah
entahlah
terhadap
terhadapnya
hal
hampir
hanya
hanyalah
harus
haruslah
harusnya
seharusnya
hendak
hendaklah
hendaknya
hingga
sehingga
ia
ialah
ibarat
ingin
inginkah
inginkan
ini
inikah
inilah
itu
itukah
itulah
jangan
jangankan
janganlah
jika
jikalau
juga
justru
kala
kalau
kalaulah
kalaupun
kalian
kami
kamilah
kamu
kamulah
kan
masihkah
semasih
masing
mau
maupun
semaunya
memang
mereka
merekalah
meski
meskipun
semula
mungkin
mungkinkah
nah
namun
nanti
nantinya
nyaris
oleh
olehnya
seorang
seseorang
pada
padanya
padahal
paling
sepanjang
pantas
sepantasnya
sepantasnyalah
para
pasti
pastilah
per
pernah
pula
pun
merupakan
rupanya
serupa
saat
saatnya
sesaat
saja
sajalah
siapa
siapakah
siapapun
disini
disinilah
sini
sinilah
sesuatu
sesuatunya
suatu
sesudah
sesudahnya
sudah
sudahkah
sudahlah
supaya
tadi
tadinya
tak
tanpa
setelah
telah
tentang
tentu
tentulah
tentunya
tertentu
seterusnya
tapi
tetapi
setiap
tiap
setidaknya
tidak
tidakkah
tidaklah
toh
waduh
wah
wahai
sewaktu
walau
walaupun
wong
yaitu

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
demi
demikian
demikianlah
sedemikian
dengan
depan
di

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
selamanya
lebih
terlebih
bermacam
macam
semacam
maka

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
selain
selalu
seluruh
seluruhnya
semakin
sementara
sempat

yakni
yang

4. Kode Program N-Gram untuk Mencari *Corpus* dan *Keyword* Bukalapak

```
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].equalsIgnor
eCase("tak")||words[i].equalsIgnoreCase("tanpa")||words[i].equalsIgnoreCase("tanpa")||words[i].eq
ualsIgnoreCase("tidak")||words[i].equalsIgnoreCase("pantang")||words[i].equalsIgnoreCase("janga
n")||words[i].equalsIgnoreCase("bukanlah")||words[i].equalsIgnoreCase("gak")||words[i].equalsIgn
oreCase("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 = "D:/BUKALAPAK.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);
    }
}
```

5. Kode Program *Sentistrength* dengan Bahasa Python

```
# 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 sentiment 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 SENTIMENT 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.sentences_m
ax_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 {"text ": " ".join(self.sentence_score) , "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("D:/2019.txt", 'r')]; #E:/00. SKRIPSIIIIIIIIIIIIIIII
FIIXXXXXXXXXX/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)
```

6. Kode Program untuk Dimensi *Corporate Reputation* dengan Bahasa Python

```
# 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 sentiment 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 SENTIMENT 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.sentences_m
ax_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 {"text ": " ".join(self.sentence_score) , }
     # return{sentence_result}
config = dict()
config["negation"] = False
config["booster"]  = False
config["ungkapan"]  = False
config["consecutive"]  = False
config["repeated"]  = False
config["emoticon"]  = False
config["question"]  = False
config["exclamation"]  = False
config["punctuation"]  = False
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("D:/datacontoh.txt", 'r')]; #E:/00. SKRIPSIIIIIIIIIIIIIIII
FIIXXXXXXXXXX/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* untuk Menghitung Jumlah Polaritas pada *Dimensi Corporate Reputation* dan Nilai *Sentiment*

```
SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'positive' AND `Corporate` like '%[1]%'

SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'neutral' AND `Corporate` like '%[1]%'

SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'negative' AND `Corporate` like '%[1]%'


SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'positive' AND `Corporate` like '%[2]%'

SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'neutral' AND `Corporate` like '%[2]%'

SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'negative' AND `Corporate` like '%[2]%'


SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'positive' AND `Corporate` like '%[3]%'

SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'neutral' AND `Corporate` like '%[3]%'

SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'negative' AND `Corporate` like '%[3]%'


SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'positive' AND `Corporate` like '%[4]%'

SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'neutral' AND `Corporate` like '%[4]%'

SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'negative' AND `Corporate` like '%[4]%'


SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'positive' AND `Corporate` like '%[5]%'

SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'neutral' AND `Corporate` like '%[5]%'

SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'negative' AND `Corporate` like '%[5]%'


SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'positive' AND `Corporate` like '%[6]%'
```

SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'neutral' AND `Corporate` like '%[6]%'

SELECT count(*) 'total' FROM `skripsi` WHERE `SENTIMEN` = 'negative' AND `Corporate` like '%[6]%'

8. Hasil Kamus Asosiasi *Corpus Sentiment* dengan *Dimensi Corporate Reputation*

	kata	kode corporate	nilai sentimen		KODE + KATA
310	amanah	5	amanah :4		amanah :5
311	sombong	5	sombong :-4		sombong :5
312	etika	5	etika :4		etika :5
313	maafkan	5	maafkan :4		maafkan :5
314	kreatif	5	kreatif:4		kreatif:5
315	integritas	6	integritas:4		integritas:6
316	karakter	6	karakter:4		karakter:6
317	kelakuan	6	kelakuan:-3		kelakuan:6
318	kepribadian	6	kepribadian:4		kepribadian:6
319	lagak	6	lagak:-3		lagak:6
320	perbuatan	6	perbuatan:-4		perbuatan:6
321	sepak terjang	6	sepak terjang:4		sepak terjang:6
322	sifat	6	sifat:4		sifat:6
323	sikap	6	sikap:4		sikap:6
324	telatah	6	telatah:1		telatah:6
325	temperamen	6	temperamen :-4		temperamen :6
326	baik	6	baik :4		baik :6
327	tindak-tanduk	6	tindak-tanduk:-3		tindak-tanduk:6
328	tingkah laku	6	tingkah laku:2		tingkah laku:6
329	ulah	6	ulah:-3		ulah:6
330	watak	6	watak:-2		watak:6

9. Contoh Data dari Status *Tweet*

username	date	retwe	favo	text	geo	mentions	hashtags	id	permalink
latieffebrian	##########	39	50	# UNINSTALLBUKALAPAK # UNINSTALLBUKALAPAK # UNINSTALL	@		#######	1.10E+18	https://twitter
benkkurniawan bul	##########	2	1	bikin haztag aja # uninstallbukalapak bukaemazbukapalak kwkwkw			#	1.10E+18	https://twitter
kebondanas achma	##########	2	0	# UninstallBukalapak :) blum pernah trending sih...			#	1.10E+18	https://twitter
gorila_bengong1	##########	0	0	# uninstallbukalapak			#	1.10E+18	https://twitter
mardikaaris1923	##########	0	0	# Uninstallbukalapak			#	1.10E+18	https://twitter
aditiamaruli kemal	##########	7	5	saya ga rispek lagi. # uninstallbukalapak			#	1.10E+18	https://twitter
khupcom	##########	4	2	# UNINSTALLBUKALAPAK Saatnya pake # TOPED # TOKOPEDIA			###	1.10E+18	https://twitter
Lizalisakusuma SJW	##########	0	0	# UNINSTALLBUKALAPAK			#	1.10E+18	https://twitter
khilafvck achmadza	##########	5	4	# uninstallbukalapak			#	1.10E+18	https://twitter
Lizalisakusuma yus	##########	24	62	udah langsung # UNINSTALLBUKALAPAK			#	1.10E+18	https://twitter
limajam123 achma	##########	7	5	Padahal udah promo ke temenÃ² Nunggu barnagku sampai # uninstallbukal			#	1.10E+18	https://twitter
VersiDeni kemalars	##########	2	1	Ikutan # uninstallbukalapak			#	1.10E+18	https://twitter
LeZtiary	##########	6	4	# uninstallbukalapak # boikotbukalapak			##	1.10E+18	https://twitter
WahonoSulistio ach	##########	0	0	# uninstallBukaLapak			#	1.10E+18	https://twitter
AloysiusAS	##########	3	3	Yang koar-koar # UninstallTraveloka sama # UninstallBukalapak gaada beda			##	1.10E+18	https://twitter
F4R1SYOUS	##########	5	10	# BoikotBukalapak # UninstallBukalapak https://www.facebook.com/10000			##	1.10E+18	https://twitter
pakde_lukman ach	##########	23	116	Ga ada terimakasihnya, kemarin sudah dibantu pak @ jokowi p	@		#	1.10E+18	https://twitter
FebriAndhika84 ach	##########	102	562	Sudah 3 bulan saya sering belanja di BL dari yang lain. Kalo CEO nya kaya gin			#	1.10E+18	https://twitter
rockybrown007	##########	0	0	# uninstallbukalapak			#	1.10E+18	https://twitter
rasjogja	##########	30	14	Sudahkah kalian UNINSTALL @ bukalapak hari ini? yg sudah uni	@		#	1.10E+18	https://twitter