BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan pada data *tweet* tentang wisata di Yogyakarta pada bulan November 2016 hingga Desember 2016 dapat diambil kesimpulan bahwa *Terms* pada kamus *Sentistrength* dapat diasosiasikan ke dalam bentuk *Level of Place Attachment* dan dikategorikan ke dalam *Urban Qualities* berdasarkan teori yang sudah ada. Hal ini dapat digunakan sebagai acuan dalam meningkatkan kualitas tempat-tempat wisata di Yogyakarta dengan menggunakan analisis sentimen. Proses klasifikasi *tweet* ke dalam kelas sentimen dengan menggunakan metode *Sentistrength* memberikan hasil polaritas sentimen yang lebih bagus dibandingkan metode berbasis leksikon biasa karena metode ini memberi bobot *range* 1 sampai 5 masing-masing untuk kelas positif dan negatif. Analisis untuk data keseluruhan memperlihatkan hasil bahwa sentimen yang dihasilkan lebih cenderung menghasilkan sentimen yang positif dibandingkan negatif.

5.2 Saran

Adapun beberapa saran yang disampaikan untuk peningkatan kualitas tempat-tempat wisata di Yogyakarta adalah sebagai berikut :

1. Dari hasil penelitian ini ditemukan bahwa *terms* pada kamus *default Sentistrength* masih kurang lengkap dengan kebutuhan kosakata bahasa Indonesia. Disarankan untuk peneliti selanjutnya untuk menambah isi *term* pada kamus *Sentistrength* tanpa mengubah aturan yang diberlakukan oleh metode *Sentistrength.*
2. Dari hasil penelitian ini penulis menyarankan untuk menambah *keyword* pada saat mengumpulkan data dari Twitter.

DAFTAR PUSTAKA

Altman, I. (1992). Place Attachment, 314. https://doi.org/10.1007/978-1-4684-8753-4

Arifidin, S. (2016). Pembangunan Aplikasi Rekomendasi Berita berbasis preferensi pengguna twitter.

Buntoro, G. (2014). Sentiment Analysis Twitter dengan Kombinasi Lexicon Based dan Double Propagation. *research gate*, 43.

Cavnar, W. B., Trenkle, J. M., & Mi, A. A. (1994). N-Gram-Based Text Categorization. *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, 161–175. https://doi.org/10.1.1.53.9367

Feldman, R., & Sanger, J. (2006). *The Text Mining Handbook*. https://doi.org/10.1017/CBO9780511546914

Habibi, R. (2016). Pemetaan Gaya Belajar Mahasiswa dan Kecendrungan emosi pada Twitter. *Sentiment analysis*, 71.

Hadianto, M. (2016). Pengembangan Sosial Intelejensi Bisnis Akademik memanfaatkan media social twitter.

Hantoro, P. A. (2014). Jaringan Komunikasi Rumor Bencana Erupsi Gunung Kelud 13 Februari 2014 di Situs Microblogging Twitter.com.

Jain, A. P. (2015). Sentiments Analysis Of Twitter Data Using Data Mining, 807–810.

Karamibekr, M., & Ghorbani, A. A. (2013). A Structure for Opinion in Social Domains, 264–271. https://doi.org/10.1109/SocialCom.2013.44

Kominfo. (2013, 11 07). Pengguna Internet di Indonesia 63 Juta Orang.

Manalu, B. U. (2014). ANALISIS SENTIMEN PADA TWITTER

MENGGUNAKAN TEXT MINING SKRIPSI Boy Utomo Manalu. *Teknologi Informasi Fak. ILKOM UNSUT*.

Mujilahwati, S. (2016). Pre-Processing Text Mining Pada Data Twitter. *Seminar Nasional Teknologi Informasi Dan Komunikasi*, *2016*(Sentika), 2089–9815.

Muthutantrige, S. R., & Weerasinghe, A. R. (2016). Sentiment Analysis in Twitter Messages Using Constrained and Unconstrained Data Categories, 304–310. https://doi.org/10.1109/ICTER.2016.7829935

Oh, C. O., Lyu, S. O., & Hammitt, W. E. (2012). Predictive Linkages between Recreation Specialization and Place Attachment. *Journal of Leisure Research*, *44*(1), 70–87. https://doi.org/10.1080/00222216.2012.11950255

Rubinstein, R. I., & Parmelee, P. A. (1992). Attachment to Place and the Representation of the Life Course by the Elderly. *Place Attachment*, 139–163. https://doi.org/10.1007/978-1-4684-8753-4_7

Setyabudi, H. (2017). *Analisis collective memory Dari media sosial untuk meningkatkan Level of place attachment Pada tempat wisata di kota yogyakarta*.

Sosial, J. I., & Volume, I. P. (2016). Studi Eksploratif Mengenai Yogyakarta sebagai Pengirim Wisatawan Keluarga, *20*, 84–96.

Spencer, J., & Uchyigit, G. (2012). Sentimentor: Sentiment analysis of twitter data. *CEUR Workshop Proceedings*, *917*, 56–66. https://doi.org/10.1007/978-3-642-35176-1_32

Sulistya, A. B. (2016). Tren Perkembangan Pariwisata Daerah Istimewa Yogyakarta Periode 2006-2014 Skripsi, 115.

Thelwall, M., Buckley, K., & Paltoglou, G. (2012). Sentiment Strength Detection for the Social Web 1, *63*, 163–173.

Utomo, D. A. (2013). Motif Pengguna Jejaring Sosial Google + Di Indonesia. *E-Komunikasi*, *1 No.3*, 147–156.

Wahid, D. H., & SN, A. (2016). Peringkasan Sentimen Esktraktif di Twitter Menggunakan Hybrid TF-IDF dan Cosine Similarity. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *10*(2), 207. https://doi.org/10.22146/ijccs.16625

Wibowo, S. K. A., & Mirawati, I. (2013). Realitas Politik Indonesia dalam Kacamata Pengguna Twitter. *Jurnal Kajian Komunikasi*, *1/1*, 11–17.

LAMPIRAN

1. Kode Program untuk Data Collection dengan bahasa Java

```
package me.jhenrique.main;

import control.Control;
import java.io.File;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.List;
import me.jhenrique.manager.TweetManager;
import me.jhenrique.manager.TwitterCriteria;
import me.jhenrique.model.Tweet;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.regex.Matcher;
import javax.management.Query;
public class Main {
    private static Connection connection;
    public static String url ="jdbc:ucanaccess://";
    public          static          final          String
path="E:"+File.separator+"Malioboro.mdb";

    private static final String USERNAME = "Username: ";
    private static final String RETWEETS = "Retweets: ";
    private static final String TEXT = "Text: ";
    private static final String DATE ="Date: ";
    private static final String MENTIONS = "Mentions: ";
    private static final String HASHTAGS = "Hashtags: ";
       private static final String LOCATION ="Location: ";

    public   static   void   main(String[]   args)   throws
SQLException, ClassNotFoundException {
         /**
          * Reusable objects
          */
     Control c= new Control();
                 try {
           connection                                        =
DriverManager.getConnection(url+path);
           System.out.println("sukses");
       } catch (SQLException ex) {

Logger.getLogger(Main.class.getName()).log(Level.SEVERE,
null, ex);
       }
```

```
          TwitterCriteria criteria = null;
          Tweet t = null;


     do{
 criteria = TwitterCriteria.create()

.setQuerySearch("wisata jogja")
.setSince("2016-11-01")
.setUntil("2016-12-10");

List<Tweet> tweets = TweetManager.getTweets(criteria);
int i =1;
for ( Tweet tw : tweets)
                              {
System.out.println(i);
System.out.println("### Example 2 - Get tweets by query
search");
System.out.println(USERNAME + tw.getUsername());
System.out.println(RETWEETS + tw.getRetweets());
System.out.println(TEXT + tw.getText());
System.out.println(DATE + tw.getDate());
System.out.println(MENTIONS + tw.getMentions());
System.out.println(HASHTAGS + tw.getHashtags());
System.out.println(LOCATION + tw.getGeo());
System.out.println();
i=i+1;

String sql="INSERT INTO BI(username, Status, tanggal,
retweet, favorites, mentions, hastags, geo_location)
VALUES(?,?,?,?,?,?,?,?)";

                                  PreparedStatement pStmt =
connection.prepareStatement(sql);
                           //                  pStmt.setInt(1,
c.GetRowDataTwitter());
pStmt.setString(1, tw.getUsername().toString());
pStmt.setString(2, tw.getText());
pStmt.setString(3, tw.getDate().toString());
pStmt.setInt(4, tw.getRetweets());
pStmt.setInt(5, tw.getFavorites());
pStmt.setString(6, tw.getMentions());
pStmt.setString(7,tw.getHashtags());
pStmt.setString(8,tw.getGeo().toString());
pStmt.executeUpdate();
    }
  break;

}while(TweetManager.getTweets(criteria).size()!=0);
    }
```

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 {
        // TODO code application logic here

        String sql = "SELECT * FROM Twitter ";
        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("Status");
String tanggal = rs.getString("tanggal");
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(input
Data);
//String sql2="UPDATE Stopwords set Status = ? where ID =?";
//versi edit
String     sql2     =     "INSERT     into     stopwords
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(3, output);
                        pStmt.setString(4, 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);
            }
        }
}
```

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 cari tempat wisata yang ada

```
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("tugu")||words[i].equalsIgnore
Case("museum")||words[i].equalsIgnoreCase("benteng")||words
[i].equalsIgnoreCase("taman”))
            {
                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:/testing.txt"; // letak file
yang akan di pecah per kata

        for (String ngram : ngrams(readLineByLineJava8(
filePath )))
            {
                System.out.println(ngram);
            }

    }
    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_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,
           #       "sentimen": 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=[line.strip() for line in open("E:/testing.txt",
'r')];          #E:/00.          SKRIPSIIIIIIIIIIIIIIIII
FIIXXXXXXXXXX/DATA/datafix_novdes1.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 list1: print senti.main(p)
#print senti.main(list1)
```

6. **Kode Program untuk *Level of Place Attachment* dan *Urban Qualities***

```
# coding: utf-8

import re
from collections import OrderedDict
import numpy as np

class sentistrength:
    def __init__(self, config=dict()):

        #create Level Of Attachment 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])

        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 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 = "Level 1"
        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

                #+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_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       {"classification       text       ":"
".join(self.sentence_score)       }#,       "Level       ":
self.sentences_max_pos}

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)

list2=[line.strip()  for  line  in  open("E:/testing.txt",
'r')];          #E:/00.          SKRIPSIIIIIIIIIIIIIIIII
FIIXXXXXXXXXX/DATA/datafix_novdes2.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)

# 1 access, 2 control, 3 fit, 4 sense, 5 vitality


#print senti.main(list1)
```

7. Query untuk menghitung jumlah polaritas pada urban qualities dan level tertentu

HITUNG SENTIMEN
positive,negative,neutral

ACCESS

LEVEL 1

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like  '%[11]%'  AND  `levelurban`  not  like  '%2]%'  AND
`levelurban` not like '%3]%' AND `levelurban` not like
'%4]%' and sentiment=''
```

LEVEL 2

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like  '%[12]%'  AND  `levelurban`  not  like  '%3]%'  AND
`levelurban` not like '%4]%' and sentiment=''
```

LEVEL 3

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[13]%' AND `levelurban` not like '%4]%' and
sentiment=''
```

LEVEL 4

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[14]%' and sentiment=''
```

CONTROL

LEVEL 1

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[21]%' AND `levelurban` not like '%2]%' AND
`levelurban` not like '%3]%' AND `levelurban` not like
'%4]%' and sentiment=''
```

LEVEL 2

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[22]%' AND `levelurban` not like '%3]%' AND
`levelurban` not like '%4]%' and sentiment=''
```

LEVEL 3

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[23]%' AND `levelurban` not like '%4]%' and
sentiment=''
```

LEVEL 4

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[24]%' and sentiment=''
```

FIT

LEVEL 1

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[31]%' AND `levelurban` not like '%2]%' AND
`levelurban` not like '%3]%' AND `levelurban` not like
'%4]%' and sentiment=''
```

LEVEL 2

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[32]%' AND `levelurban` not like '%3]%' AND
`levelurban` not like '%4]%' and sentiment=''
```

LEVEL 3

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[33]%' AND `levelurban` not like '%4]%' and
sentiment=''
```

LEVEL 4

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[34]%' and sentiment=''
```

SENSE

LEVEL 1

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[41]%' AND `levelurban` not like '%2]%' AND
`levelurban` not like '%3]%' AND `levelurban` not like
'%4]%' and sentiment=''
```

LEVEL 2

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[42]%' AND `levelurban` not like '%3]%' AND
`levelurban` not like '%4]%' and sentiment=''
```

LEVEL 3

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[43]%' AND `levelurban` not like '%4]%' and
sentiment=''
```

LEVEL 4

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[44]%' and sentiment=''
```

VITALITY

LEVEL 1

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[51]%' AND `levelurban` not like '%2]%' AND
`levelurban` not like '%3]%' AND `levelurban` not like
'%4]%' and sentiment=''
```

LEVEL 2

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[52]%' AND `levelurban` not like '%3]%' AND
`levelurban` not like '%4]%' and sentiment=''
```

LEVEL 3

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[53]%' AND `levelurban` not like '%4]%' and
sentiment=''
```

LEVEL 4

```
SELECT count(*) 'total' FROM `mytable` WHERE `levelurban`
like '%[54]%' and sentiment=''
```

8. Hasil Kamus asosiasi corpus sentimen dengan level of place attachment dan urban qualities

KATA	NILAI SENTIMENT	LEVEL of PLACE ATTACHMENT	URBAN QUALITIES
bangkrut	-5	1	control
tandus	-5	1	control
tengik	-5	1	control
mabuk	-5	1	fit
astaga	-5	1	sense
barbari	-5	1	sense
celakalah	-5	1	sense
naudzubillah	-5	1	sense
nyungsep	-5	1	vitality

berjudi	-5	2	control
kere	-5	2	control
norak	-5	2	fit
udik	-5	2	fit
amoral	-5	2	sense
mencibir	-5	2	sense
keserakahan	-5	2	vitality
dendam	-5	3	control
penipu	-5	3	control
bencong	-5	3	fit
berbohong	-5	3	fit
haram	-5	3	fit
angkuh	-5	3	sense
gagal	-5	3	sense
kesombongan	-5	3	sense
muak	-5	3	sense
sakit-sakitan	-5	3	sense
aib	-5	3	vitality
anarki	-5	3	vitality
anarkis	-5	3	vitality
dirampas	-5	3	vitality
menipu	-5	3	vitality
parah	-5	3	vitality
terkutuk	-5	3	vitality
diskriminasi	-5	4	control
kemunafikan	-5	4	control
munafik	-5	4	control
arogan	-5	4	fit
bajingan	-5	4	fit
brengsek	-5	4	fit
busuk	-5	4	fit
cabul	-5	4	fit
sialan	-5	4	fit
tai	-5	4	fit
tolol	-5	4	fit
zina	-5	4	fit
zina	-5	4	fit
agresi	-5	4	sense
antipati	-5	4	sense
babu	-5	4	sense
bacot	-5	4	sense

9. Contoh data status twitter

ID	username	Status	tanggal	retweet	favorites	mentions	hastags
1	____frtyra	azwarunddin takpela dpn pon nk jogja dgn vietnam relax bonding	Nov 01 00	0	1	@AzwaRunddin	
2	____akbar	jogja transportasi umun mencapai rumah terpencil kemana mama	Nov 01 11:	0	0		
3	____ewh	megantari jogja terima kasih menunggu cek nomor kartuas jagoan	Dec 13 15	0	0		
4	____ewh	megantari jogja terima kasih menunggu cek nomor kartuas jagoan	Dec 13 15	0	0		
5	____ewh	megantari jogja 1 hai maaf keluhan akses internet lambat nomor	Dec 23 0	0	0		
6	____ewh	megantari jogja 1 hai maaf keluhan akses internet lambat nomor	Dec 23 0	0	0		
7	____ikatika	enak icecream aiceicecream yogyakarta https www instagram com	Dec 08 16	0	0	@	#icecream #aice
8	____ikatika	selamat pagi jogja selamat jumat pic https path com p 4dzlmc	Dec 09 0	0	0		
9	____ikatika	selamat pagi jogja selamat jumat pic https path com p 4dzlmc	Dec 09 0	0	0		
10	____ikatika	enak yogyakarta https www instagram com p bnycz2la21a	Dec 09 1	0	0	@	
11	____ikatika	semalam monjali monumen jogja https www instagram com p bn2 w	Dec 11 09	0	0	@	
12	____ikatika	semalam monjali monumen jogja https www instagram com p bn2 w	Dec 11 09	0	0	@	
13	____ikatika	jajanan raden alun alun utara yogyakarta https path com p 4hxuhh	Dec 11 12	0	0		
14	____ikatika	jogja makanan banyak yg murah badan raden taman sari water	Dec 11 13	0	0		
15	____ikatika	jogja makanan banyak yg murah badan raden taman sari water	Dec 11 13	0	0		
16	____ikatika	last trip bye jogja taman sari water castel https www instagram com	Dec 11 13	0	0	@	
17	____ikatika	last trip bye jogja taman sari water castel https www instagram com	Dec 11 13	0	0	@	
18	____ikatika	alun alun selatan kraton jogjakarta bringin kembar https www	Dec 11 13	0	0	@	
19	__amalina	dasta windii giwangan yogyakarta https path com p 374mzw	Dec 23 16	0	0		
20	__amalina	dasta windii giwangan yogyakarta https path com p 374mzw	Dec 23 16	0	0		
21	__azza	dienmatina purnama warung yogya sederhana	Dec 08 19	0	0	@DienMatina	
22	__azza	liburan yogya bisa menghubungi sahabat mbuddul lokasi lokasi	Nov 25 12	6	5	@MbudDul	
23	__azza	sama sama fun yogya tolong temen mbuddul dipromo temen dan	Nov 25 12	0	0	@MbudDul	
24	__BisnisRumah	liburan keluarga yogya malang 2016 day http youtu be ooijtarngym	Nov 26 15	0	0	@YouTube	
25	__blueshine	iya ngumah wingi wingi cah lanangan mitup ng jogja	Dec 18 15	0	0		
26	__blueshine	iya ngumah wingi wingi cah lanangan mitup ng jogja	Dec 18 15	0	0		
27	__bosman	selalutauyangseru google indonesia jogja pengen kesanaaaa jogja	Nov 14 19	0	0		#SelaluTauYang
28	__bttfly	hahahahhaa udah berantem bodo oohh kirain swasta nggak	Dec 18 22	0	0		
29	__bttfly	hahahahhaa udah berantem bodo oohh kirain swasta nggak	Dec 18 22	0	0		
30	__cholis	angkringan versi yogya eks karesidenan surakarta menyebutnya hik	Dec 27 16	0	0		
31	__dindafkj	mampir bar kondangan jogja city mall https path com p 1ueasp	Nov 05 20	0	0		
32	__doni___	via jogja yg enak sama2 enak	Nov 07 21	0	0	@via_jogja	