**Panduan Text Data Preprocessing untuk Metamorphosis**

(Oleh Sakinah Salsabilillah Hilaly, Tabita Yuni Susanto, dan Wilda Melia Udiatami)

Dokumen ini akan menjelaskan secara rinci tahapan-tahapan preprocessing untuk data (mental illness symptoms) yang didapatkan dari hasil crowdsourcing oleh website <https://metamorphosis.id>. Syntax secara keseluruhan telah dituliskan dalam Python Notebook:

<https://colab.research.google.com/drive/1hgmFupmS52qRPa_fNkFbprFMoUQBIXCt?usp=sharing#scrollTo=XCszzPFPCpL7>.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dataset yang diperlukan:

1. Data Symptoms (answers\_ok.csv)

Berisi keseluruhan jawaban kuesioner yang telah di-submit oleh responden Metamorphosis. Jawaban dari pertanyaan *“List all of your symptoms before being diagnosed and/or starting treatment, in order of severity.”* dimuat dalam kolom “q\_11”.

1. Synonym Directory

list\_temp\_lemmed (1).csv : directory separated by space ( ) untuk menghapus kata yang tidak terdapat pada list\_temp\_lemmed(1)

list\_temp\_lemmed.txt : directory separated by comma (,) untuk membuat synonym directory

list\_temp\_lemmed.csv : directory separated by underscore (\_) untuk menghapus kata/frase yang tidak terdapat pada list\_temp\_lemmed

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tahap 1 : Import Data

Package : pandas

import pandas as pd

datafull=pd.read\_csv(r'answers\_ok.csv',header=1)

data=pd.DataFrame(datafull["q\_11"])["q\_11"]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tahap 2 : Translate

Package : Translator (from googletrans==3.1.0a0)

Kata yang terdeteksi sebagai Bahasa Indonesia diterjemahkan menjadi Bahasa Inggris.

!pip install googletrans==3.1.0a0

from googletrans import Translator

translator=Translator()

trans=data.apply(translator.translate,

src='id',dest='en').apply(getattr,args=('text',))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tahap 3 : Case Folding

Package : -

Seluruh karakter dalam line (baris) diseragamkan menjadi lowercase.

casefolded=[]

for line in trans:

a=line.lower()

casefolded.append(a)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tahap 4 : Omit Numbers

Package : digits (from strings)

Seluruh karakter yang berupa angka dihapus dari line.

from string import digits

number=[]

for line in casefolded:

remove\_digits=str.maketrans(' ', ' ', digits)

result=line.translate(remove\_digits)

number.append(result)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tahap 5 : Omit Punctuations

Package : regex

Seluruh karakter yang merupakan tanda baca (kecuali koma) dihapus dari line.

import regex as re

punct=[]

for char in number:

result=re.sub(r"\p{P}(?<!,)","", char)

punct.append(result)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tahap 6 : Tokenizing (Split by comma)

Package : -

Line dipisahkan berdasarkan koma. Hasil tokenisasi bisa berupa kata, frase, atau kalimat.

splitted=[]

for line in punct:

a=line.split(',')

splitted.append(a)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tahap 7 : Lemmatizing

Package : nltk

import nltk

from nltk.corpus import wordnet

from nltk.stem import WordNetLemmatizer

nltk.download('averaged\_perceptron\_tagger')

lemmatizer = WordNetLemmatizer()

1. Memberi tanda (tag) apakah suatu kata merupakan

conjunction (CC) adverb (RB)

numeral (CD) “to” as preposition (TO)

determiner (DT) verb, base form (VB)

preposition (IN) verb, past tense (VBD)

adjective (JJ) verb, ing (VBG)

modal auxiliary (MD) WH-determiner (WDT)

noun (NN) etc.

pre-determiner (PDT)

pronoun (PRP)

Sub-kategori yang sejenis ditulis dengan huruf awal yang sama (lihat VB, VBD, dan VBG).

pos\_tagged=[]

for i in range(len(splitted)):

a=[]

for j in range(len(splitted[i])):

b=nltk.pos\_tag(nltk.word\_tokenize(splitted[i][j]))

a.append(b)

pos\_tagged.append(a)

1. Mendefinisikan fungsi pos\_tagger yang digunakan untuk memberi tanda (tag) apakah suatu kata harus dikembalikan (di-lemmatize) ke bentuk dasar adjective, noun, verb, atau adverb. Pada fungsi pos\_tagger pertama ini, seluruh kata dikembalikan ke bentuk dasar tag aslinya.

def pos\_tagger(nltk\_tag):

if nltk\_tag.startswith('J'):

return wordnet.ADJ

elif nltk\_tag.startswith('V'):

return wordnet.VERB

elif nltk\_tag.startswith('N'):

return wordnet.NOUN

elif nltk\_tag.startswith('R'):

return wordnet.ADV

else:

return None

1. Memberikan tag pada seluruh kata (berdasarkan pos\_tagger pertama)

wordnet\_tagged=[]

for i in range(len(pos\_tagged)):

a=[]

for j in range(len(pos\_tagged[i])):

b=list(map(lambda x: (x[0], pos\_tagger(x[1])),pos\_tagged[i][j]))

a.append(b)

wordnet\_tagged.append(a)

1. Mengembalikan seluruh kata ke bentuk dasar (lemmatize) berdasarkan tag yang telah diberikan

lem = []

for i in range(len(wordnet\_tagged)):

a=[]

for j in range(len(wordnet\_tagged[i])):

b=[]

for word, tag in wordnet\_tagged[i][j]:

if tag is None:

b.append(word)

else:

b.append(lemmatizer.lemmatize(word, tag))

b=" ".join(b)

a.append(b)

lem.append(a)

1. Mendefinisikan pos\_tagger kedua dimana noun dikembalikan ke bentuk dasar berupa verb.

Misal: anger → angry, hallucination → hallucinate

def pos\_tagger(nltk\_tag):

if nltk\_tag.startswith('J'):

return wordnet.ADJ

elif nltk\_tag.startswith('V'):

return wordnet.VERB

elif nltk\_tag.startswith('N'):

return wordnet.VERB

elif nltk\_tag.startswith('R'):

return wordnet.ADV

else:

return None

1. Memberi tanda (tag) awal lagi

pos\_tagged=[]

for i in range(len(lem)):

a=[]

for j in range(len(lem[i])):

b=nltk.pos\_tag(nltk.word\_tokenize(lem[i][j]))

a.append(b)

pos\_tagged.append(a)

1. Memberikan tag pada seluruh kata (berdasarkan pos\_tagger kedua)

wordnet\_tagged=[]

for i in range(len(pos\_tagged)):

a=[]

for j in range(len(pos\_tagged[i])):

b=list(map(lambda x: (x[0], pos\_tagger(x[1])),pos\_tagged[i][j]))

a.append(b)

wordnet\_tagged.append(a)

1. Mengembalikan seluruh kata ke bentuk dasar (lemmatize) berdasarkan tag yang telah diberikan

lem2 = []

for i in range(len(wordnet\_tagged)):

a=[]

for j in range(len(wordnet\_tagged[i])):

b=[]

for word, tag in wordnet\_tagged[i][j]:

if tag is None:

b.append(word)

else:

b.append(lemmatizer.lemmatize(word, tag))

b=" ".join(b)

a.append(b)

lem2.append(a)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tahap 8 : Removing Stopwords (1 & 2)

Package : nltk

nltk.download('stopwords')

1. Menghapus kata di dataset yang merupakan kata pada list stopwords (seperti: i, me, our, he, she, for, the, which, this, that, to, dan lain-lain)

# Remove Stopwords 1

stopwords = nltk.corpus.stopwords.words('english')

removestop = []

for i in range(len(lem2)):

a=[]

for j in range(len(lem2[i])):

word\_token=nltk.word\_tokenize(lem2[i][j])

word\_token=[word for word in word\_token if not word in stopwords]

a.append(" ".join(word\_token))

removestop.append(a)

1. Mendefinisikan file directory “list\_temp\_lemmed (1).csv” sebagai **sin\_direct2**

syn1 = pd.read\_csv("list\_temp\_lemmed (1).csv",header=None,sep=" ", error\_bad\_lines=False)

sin\_direct2= syn1[syn1.columns[0:]].apply(

lambda x: ','.join(x.dropna().astype(str)),

axis=1)

1. Membuat data list **sw1** yang berisi dataset dari **sin\_direct2**, tetapi antar kata (keyword dan synonyms semua baris) dipisahkan dengan koma, menjadi satu data list. Contoh: [‘breath’, ‘pant’, gasp’, ‘panic’, ‘alarm’, ‘confusion’, …..]

# Remove Stopwords 2

for n,line in enumerate(sin\_direct2):

if line.startswith("line"):

sin\_direct2[n] = " "+line.rstrip()

else:

sin\_direct2[n]=line.rstrip()

sw1=','.join(sin\_direct2)

sw1=sw1.split(',')

1. Menghapus kata/frase di dataset yang bukan merupakan kata pada list **sw1.** Sehingga pada dataset akan menyisakan kata/frase sesuai dengan keyword dan synonyms yang diharapkan (sesuai directory **sin\_direct2**).

removestop2 = []

for i in range(len(removestop)):

a=[]

for j in range(len(removestop[i])):

word\_token=nltk.word\_tokenize(removestop[i][j])

word\_token=[word for word in word\_token if word in sw1]

a.append(" ".join(word\_token))

removestop2.append(a)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tahap 9 : Replacing Words with Synonym

Package : Jieba

1. Synonym replacement untuk frase/kata yang dipisahkan oleh koma pada kalimat pada dataset **removestop2.**

Misalnya : **drink alcohol pain, inhale cry, kill myself, angry, change appetite eat** akan diganti sinonimnya dengan kata tertentu sesuai directory

drink alcohol pain → ??

Breathe cry → ??

Kill myself → ??

Angry → anger

Sehingga kalimat tersebut menjadi **drink alcohol pain, inhale cry, suicide, anger, change appetite eat**

import jieba

1. Mendefiniskan fungsi tihuan\_tongyici untuk mengganti kata/frase tertentu sesuai pada kamus sinonim yang dibuat. Misalnya pant, gasp, breath, inhale, exhale, dan respiration akan diganti dengan kata breathe. kill myself, take my own life, kill self, take own life akan diganti (replace) dengan kata suicide.

Combine\_dict adalah

#list\_temp\_lemmed.txt adalah tabel sinonim, setiap baris adalah rangkaian sinonim, dipisahkan oleh koma

def tihuan\_tongyici(string1):

combine\_dict = {}

for line in open("list\_temp\_lemmed.txt", "r"):

seperate\_word = line.strip().split(",")

num = len(seperate\_word)

for i in range(1, num):

combine\_dict[seperate\_word[i]] = seperate\_word[0]

f = string1

final\_sentence = ""

for word in f.split(", "):

if word in combine\_dict:

word = combine\_dict[word]

final\_sentence += word

else:

final\_sentence += word

return final\_sentence #print kalimat hasil akhir

1. Mengganti kata/frase di dataset (removestop2) dengan synonym pada directory **list\_temp\_lemmed.** Sehingga pada dataset akan menyisakan kata/frase sesuai dengan synonyms yang diharapkan.

syno = []

for i in range(len(removestop2)):

a=[]

for j in range(len(removestop2[i])):

word\_token=tihuan\_tongyici(removestop2[i][j])

a.append("".join(word\_token))

syno.append(a)

1. Synonym replacement untuk 1 kata atau kata yang dipisahkan oleh spasi (“ ”) pada kalimat di dataset **syno**. Bedanya dengan sebelumnya adalah pada f.split atau memisahkan kata/frase di kalimat pada dataset. Sebelumnya f.split menggunakan pemisah koma yang berarti dapat berupa kata atau frase, sedangkan tahap ini, f.split menggunakan pemisah spasi yang berarti yang akan diganti sinonimnya hanya berupa 1 kata. Hal ini dilakukan karena terdapat beberapa kata pada frase yang dipisahkan oleh koma (di dataset sebelumnya) yang ada dalam synonym directory, sehingga untuk menghindari penghapusan kata/frase yang seharusnya digunakan maka dilakukan synonym replacement 2 tahap.

Misalnya dari hasil sebelumnya :

**drink alcohol pain, inhale cry, suicide, anger, change appetite eat**

Dilakukan synonym replacement menjadi kalimat berikut

**alcohol alcohol pain, breathe cry, suicide, anger, fluctuation appetite eat**

def tihuan\_tongyici1(string1):

combine\_dict = {}

for line in open("list\_temp\_lemmed.txt", "r"):

seperate\_word = line.strip().split(",")

num = len(seperate\_word)

for i in range(1, num):

combine\_dict[seperate\_word[i]] = seperate\_word[0]

# Split the statement

seg\_list = jieba.cut(string1, cut\_all = False)

f = "/".join(seg\_list) # Without utf-8 encoding, it can't correspond to the words in the txt file.

final\_sentence = ""

for word in f.split("/"):

if word in combine\_dict:

word = combine\_dict[word]

final\_sentence += word

else:

final\_sentence += word

# print final\_sentence

return final\_sentence

1. Mengganti kata di dataset (**syno**) dengan synonym pada directori **list\_temp\_lemmed.** Sehingga pada dataset akan menyisakan kata sesuai dengan synonyms yang diharapkan.

syno1 = []

for i in range(len(syno)):

a=[]

for j in range(len(syno[i])):

word\_token=tihuan\_tongyici1(syno[i][j])

a.append("".join(word\_token))

syno1.append(a)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tahap 10 : Removing Stopwords (3)

1. Mendefinisikan file directory “list\_temp\_lemmed.csv” sebagai **sin\_direct**

syn=pd.read\_csv("list\_temp\_lemmed.csv",header=None,sep="\_",error\_bad\_lines=False)

sin\_direct= syn[syn.columns[0:]].apply(

lambda x: ','.join(x.dropna().astype(str)),

axis=1)

1. Membuat data list **sw2** yang berisi dataset dari **sin\_direct**, tetapi antar kata/phrase (keyword dan synonyms semua baris) dipisahkan dengan koma, menjadi satu data list. Contoh: [‘breath\_’, ‘pant\_’, gasp\_’, ‘suicide\_’, ‘kill myself\_’, ‘kill self\_’, …..]

#remove stopwords 3

for n,line in enumerate(sin\_direct):

if line.startswith("line"):

sin\_direct[n] = " "+line.rstrip()

else:

sin\_direct[n]=line.rstrip()

sw2=','.join(sin\_direct)

sw2=sw2.split(',')

1. Menghapus kata/frase di dataset yang bukan merupakan kata pada list **sw2.** Sehingga pada dataset akan menyisakan kata/frase sesuai dengan keyword dan synonyms yang diharapkan (sesuai directory **sin\_direct**)

removestop3 = []

for i in range(len(syno1)):

a=[]

for j in range(len(syno1[i])):

word\_token=nltk.word\_tokenize(syno1[i][j])

word\_token=[word for word in word\_token if word in sw2]

a.append(" ".join(word\_token))

removestop3.append(a)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tahap 11 : Join Token

Line digabungkan berdasarkan spasi dari hasil remove stopwords 3.

joined=[]

for line in removestop3:

a=" ".join(line)

joined.append(a)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tahap 12 : Vectorizing

Package : CountVectorizer from sklearn.feature\_extraction.text

from sklearn.feature\_extraction.text import CountVectorizer

Mengubah data teks menjadi variabel-variabel biner. Variabel “A” bernilai 1 apabila “A” disebutkan minimal satu kali dalam teks, bernilai 0 jika tidak disebutkan dalam teks.

tf\_vectorizer = CountVectorizer(max\_df=len(joined), min\_df=0,

decode\_error='ignore',binary=True)

tf = tf\_vectorizer.fit\_transform(joined)

self.symptoms = pd.DataFrame(tf.toarray(),

columns=tf\_vectorizer.get\_feature\_names())