LAMPIRAN

Lampiran 1. *Wordcloud*

Gambar 1 *Wordcloud* Data Negatif (0)

Gambar 2 *Wordcloud* Data Netral (1)

Gambar 3 *Wordcloud* Data Positif (2)

Lampiran 2. Hasil Pelatihan Model CNN dan Model GRU dengan *Word2vec*

Tabel 1 Hasil Pelatihan Model CNN dengan *Word2vec*

epoch	*fold-1*	*fold-2*	*fold-3*	*fold-4*	*fold-5*
1	loss: 0.8861 - accuracy: 0.6269 - val_loss: 0.8445 - val_accuracy : 0.6587	loss: 0.3549 - accuracy: 0.8763 - val_loss: 0.2019 - val_accuracy : 0.9440	loss: 0.1880 - accuracy: 0.9354 - val_loss: 0.0994 - val_accuracy : 0.9779	loss: 0.1523 - accuracy: 0.9503 - val_loss: 0.0959 - val_accuracy : 0.9762	loss: 0.1181 - accuracy: 0.9647 - val_loss: 0.0877 - val_accuracy : 0.9745
2	loss: 0.8389 - accuracy: 0.6579 - val_loss: 0.8464 - val_accuracy : 0.6621	loss: 0.2647 - accuracy: 0.9125 - val_loss: 0.2082 - val_accuracy : 0.9304	loss: 0.1516 - accuracy: 0.9537 - val_loss: 0.1782 - val_accuracy : 0.9303	loss: 0.1223 - accuracy: 0.9622 - val_loss: 0.1049 - val_accuracy : 0.9677	loss: 0.1025 - accuracy: 0.9660 - val_loss: 0.1198 - val_accuracy : 0.9541
3	loss: 0.8198 - accuracy: 0.6532 - val_loss: 0.8371 - val_accuracy : 0.6469	loss: 0.2112 - accuracy: 0.9278 - val_loss: 0.1764 - val_accuracy : 0.9491	loss: 0.1201 - accuracy: 0.9664 - val_loss: 0.1026 - val_accuracy : 0.9694	loss: 0.0975 - accuracy: 0.9707 - val_loss: 0.1170 - val_accuracy : 0.9711	loss: 0.0889 - accuracy: 0.9724 - val_loss: 0.0912 - val_accuracy : 0.9694
4	loss: 0.7717 - accuracy: 0.6774 - val_loss: 0.7887 - val_accuracy : 0.6825	loss: 0.1569 - accuracy: 0.9520 - val_loss: 0.1742 - val_accuracy : 0.9406	loss: 0.0930 - accuracy: 0.9707 - val_loss: 0.1279 - val_accuracy : 0.9456	loss: 0.0797 - accuracy: 0.9783 - val_loss: 0.1852 - val_accuracy : 0.9371	loss: 0.0774 - accuracy: 0.9792 - val_loss: 0.1256 - val_accuracy : 0.9541
5	loss: 0.6609 - accuracy: 0.7284 - val_loss:	loss: 0.1310 - accuracy: 0.9601 - val_loss:	loss: 0.0878 - accuracy: 0.9741 - val_loss:	loss: 0.0697 - accuracy: 0.9826 - val_loss:	loss: 0.0611 - accuracy: 0.9851 - val_loss:

epoch	*fold-1*	*fold-2*	*fold-3*	*fold-4*	*fold-5*
	0.6933 - val_accuracy : 0.7097	0.1869 - val_accuracy : 0.9304	0.1116 - val_accuracy : 0.9575	0.1183 - val_accuracy : 0.9524	0.1078 - val_accuracy : 0.9609
6	loss: 0.5249 - accuracy: 0.7871 - val_loss: 0.6590 - val_accuracy : 0.7317	loss: 0.1156 - accuracy: 0.9677 - val_loss: 0.1852 - val_accuracy : 0.9338	loss: 0.0703 - accuracy: 0.9796 - val_loss: 0.1342 - val_accuracy : 0.9490	loss: 0.0615 - accuracy: 0.9860 - val_loss: 0.1250 - val_accuracy : 0.9524	loss: 0.0526 - accuracy: 0.9894 - val_loss: 0.1357 - val_accuracy : 0.9524
7	loss: 0.4259 - accuracy: 0.8304 - val_loss: 0.7031 - val_accuracy : 0.7148	loss: 0.0950 - accuracy: 0.9724 - val_loss: 0.1930 - val_accuracy : 0.9355	early stopping	early stopping	early stopping
8	loss: 0.3529 - accuracy: 0.8687 - val_loss: 0.6908 - val_accuracy : 0.7419	loss: 0.0858 - accuracy: 0.9732 - val_loss: 0.2042 - val_accuracy : 0.9304			
9	loss: 0.3062 - accuracy: 0.8836 - val_loss: 0.7732 - val_accuracy : 0.7317	early stopping			
10	loss: 0.2668 - accuracy: 0.9086 - val_loss: 0.7481 -				

epoch	*fold-1*	*fold-2*	*fold-3*	*fold-4*	*fold-5*
	val_accuracy : 0.7419				

Tabel 2 Hasil Pelatihan Model GRU dengan *Word2vec*

epoch	*fold-1*	*fold-2*	*fold-3*	*fold-4*	*fold-5*
1	loss: 0.8709 - accuracy: 0.6443 - val_loss: 0.8304 - val_accuracy : 0.6638	loss: 0.3488 - accuracy: 0.8610 - val_loss: 0.2321 - val_accuracy : 0.9066	loss: 0.2169 - accuracy: 0.9189 - val_loss: 0.1125 - val_accuracy : 0.9694	loss: 0.1725 - accuracy: 0.9418 - val_loss: 0.1102 - val_accuracy : 0.9711	loss: 0.1459 - accuracy: 0.9537 - val_loss: 0.0770 - val_accuracy : 0.9745
2	loss: 0.8299 - accuracy: 0.6587 - val_loss: 0.7910 - val_accuracy : 0.6825	loss: 0.2447 - accuracy: 0.9125 - val_loss: 0.2047 - val_accuracy : 0.9202	loss: 0.1678 - accuracy: 0.9380 - val_loss: 0.1073 - val_accuracy : 0.9609	loss: 0.1529 - accuracy: 0.9494 - val_loss: 0.1913 - val_accuracy : 0.9269	loss: 0.1259 - accuracy: 0.9622 - val_loss: 0.0612 - val_accuracy : 0.9830
3	loss: 0.7392 - accuracy: 0.7059 - val_loss: 0.6937 - val_accuracy : 0.7250	loss: 0.1935 - accuracy: 0.9303 - val_loss: 0.2124 - val_accuracy : 0.9168	loss: 0.1413 - accuracy: 0.9533 - val_loss: 0.1092 - val_accuracy : 0.9660	loss: 0.1141 - accuracy: 0.9622 - val_loss: 0.1458 - val_accuracy : 0.9473	loss: 0.1150 - accuracy: 0.9609 - val_loss: 0.0885 - val_accuracy : 0.9626
4	loss: 0.5818 - accuracy: 0.7760 - val_loss: 0.6342 - val_accuracy : 0.7487	loss: 0.1513 - accuracy: 0.9473 - val_loss: 0.2141 - val_accuracy : 0.9202	loss: 0.1230 - accuracy: 0.9571 - val_loss: 0.1239 - val_accuracy : 0.9609	loss: 0.0944 - accuracy: 0.9694 - val_loss: 0.1390 - val_accuracy : 0.9507	loss: 0.0839 - accuracy: 0.9741 - val_loss: 0.0609 - val_accuracy : 0.9779

epoch	*fold-1*	*fold-2*	*fold-3*	*fold-4*	*fold-5*
5	loss: 0.4668 - accuracy: 0.8096 - val_loss: 0.6447 - val_accuracy : 0.7538	loss: 0.1335 - accuracy: 0.9524 - val_loss: 0.2228 - val_accuracy : 0.9185	loss: 0.1144 - accuracy: 0.9626 - val_loss: 0.1312 - val_accuracy : 0.9524	loss: 0.0860 - accuracy: 0.9703 - val_loss: 0.1578 - val_accuracy : 0.9473	loss: 0.0814 - accuracy: 0.9711 - val_loss: 0.0669 - val_accuracy : 0.9745
6	loss: 0.3909 - accuracy: 0.8317 - val_loss: 0.6438 - val_accuracy : 0.7606	loss: 0.1081 - accuracy: 0.9626 - val_loss: 0.2376 - val_accuracy : 0.9202	loss: 0.0864 - accuracy: 0.9754 - val_loss: 0.1366 - val_accuracy : 0.9575	loss: 0.0672 - accuracy: 0.9809 - val_loss: 0.1572 - val_accuracy : 0.9507	loss: 0.0783 - accuracy: 0.9741 - val_loss: 0.0940 - val_accuracy : 0.9609
7	loss: 0.3099 - accuracy: 0.8729 - val_loss: 0.6548 - val_accuracy : 0.7284	loss: 0.0983 - accuracy: 0.9656 - val_loss: 0.2676 - val_accuracy : 0.9168	early stopping	early stopping	loss: 0.0765 - accuracy: 0.9741 - val_loss: 0.1112 - val_accuracy : 0.9609
8	early stopping	early stopping			early stopping

Lampiran 3. *Source Code* Pembangunan Model *Deep Learning* (CNN dan GRU)

Source Code 1 untuk *Preprocessing*

```
import re
import nltk
import pandas as pd
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import json
import requests
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('stopwords')

#cleaning data
def clean_text(text):
    # Hapus username
    text = re.sub(r'@[A-Za-z0-9_]+', '', text)
    # hapus hashtag
    text = re.sub(r'#[A-Za-z0-9]+', '', text)
    # Hapus URL
    text = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-
_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', '', text)
    # Hapus tanda baca
    text = re.sub(r"[^a-zA-Z]", " ", text)
    # Hapus angka
    text = re.sub(r"\d+", "", text)
    # Hapus spasi ekstra
    text = re.sub('\s+', ' ', text).strip()
    # Lowercase
    text = text.lower()
    # # Normalisasi slang words
    # text = ' '.join(slang_words_dict.get(word, word) for
word in text.lower().split())
    # Hapus emoji
    emoj = re.compile("["
                      u"\U0001F600-\U0001F64F"  # emoticon
                      u"\U0001F300-\U0001F5FF"  # simbol &
piktograf
                      u"\U0001F680-\U0001F6FF"  # transport &
simbol peta
                      u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                      u"\U00002500-\U00002BEF"  # karakter
china
                      u"\U00002702-\U000027B0"
                      u"\U00002702-\U000027B0"
                      u"\U000024C2-\U0001F251"
```

```
                        u"\U0001f926-\U0001f937"
                        u"\U00010000-\U0010ffff"
                        u"\u2640-\u2642"
                        u"\u2600-\u2B55"
                        u"\u200d"
                        u"\u23cf"
                        u"\u23e9"
                        u"\u231a"
                        u"\ufe0f"  # dingbats
                        u"\u3030"
                        "]+", re.UNICODE)
    text = re.sub(emoj, '', text)
    text = re.sub(r'\bw+k+\b', '', text)
    return text

twitter['clean'] = [clean_text(i) for i in tweet]
tweet = twitter['clean']

#tokenisasi
tokens = tweet.apply(word_tokenize)
twitter['Token'] = twitter['clean'].apply(word_tokenize)

#Normalisasi
from io import StringIO
import json
with open('/content/drive/My Drive/Colab Notebooks/custom-
slang-words.json') as f:
  slang_words = json.load(f)
# print(slang_words)
normalized_word  = pd.read_csv('/content/drive/My
Drive/Colab Notebooks/data/slangwords.csv',
encoding='latin1')
# Membuat kamus normalisasi
normalized_word_dict =
dict(zip(normalized_word['kata'].str.lower(),
normalized_word['formal'].str.lower()))
def normalized_term(document):
    return [slang_words[term] if term in slang_words else
normalized_word_dict.get(term, term) for term in document]

twitter['Normalisasi'] =
twitter['Token'].apply(normalized_term)

#penghapusan slangwords
stop_words = set(stopwords.words("indonesian"))
additional_stop_words = set(['yuhuuuuuu', 'hooh', 'iyaaaaa',
'kuuuu', 'xixixi', 'nahhh', 'sipsipp', 'sksk', 'hehehehehe',
'jir', 'hadeuh', 'lah', 'buseeeee','hadeh', 'yaelahhhhh',
```

```
'cie', 'nya', 'nih', 'sih', 'si', 'tuh', 'ya', 'anjrit',
'huhu', 'coy', 'duh', 'okay', 'wkakakak', 'aduh', 'ssstt',
,'hehe', 'wiiiw', 'lagiiiii', 'nan', 'loh', 'ckckck',
'ehhemm', 'hei', 'oiya', 'wkwkk', 'atuh', 'siii','eh',
'deh', 'hah', 'anjrittt', 'woaaah', 'doang', 'assoii',
'yeayyy', 'woiiii', 'hhh', 'aswww', 'brou', 'hmmm',
'yak','woah', 'lololol', 'wgwgwg','zsazsa', 'woyyyyyy',
'oalah', 'aaaa','kack', 'jderr', 'ayolaaah','wiiih',
'omonaaaaa', 'omo', 'omoo', 'ooo', 'aaaaaaaa', 'wii',
'wahhh', 'waeee', 'nyaaa', 'deg', 'ji', 'bbies',
'gelaseeeehhh','bi', 'yakaan', 'aduhh', 'eehh', 'adh',
'kaaaaan', 'hufttt', 'wey', 'coyyy',
'yakan','mih','blablabla', 'ush', 'lahhhh', 'njuk','brow',
'asdfgshshsjk','idih', 'sooo', 'siieeee', 'eeeh', 'uaah',
'lyke', 'bambankkkk', 'kannn', 'duhh', 'busyettt', 'yaaaa',
'loch', 'hiks', 'broo', 'yeay',
'buseeeeet','heleh','sksksk','jirr', 'eww', 'hikz', 'siiii',
'siy', 'ciahh', 'cuz', 'wakwaw', 'laah', 'hshs', 'omooo',
'heheh','wih', 'nich','whoaaa','yakkkk','mwah','yeah',
'woy', 'chuaaakkksss', 'ohhh', 'laaah', 'kannnnn','lahh',
'ehhhh', 'yaaaaaa', 'ooh','yaampunnn', 'cieee', 'aaaaaak',
,'ahhhh', , 'iya', 'lho','hoh','nihh', 'ebuseeet', 'brou',
'iiiiihhhh', 'hhii', 'ndul', 'yeu', 'arghh', 'beb’,
'beuhhh', 'hoalah', 'njirr', 'hemmm','siss', 'yuuuhuuu',
'widihh', 'aaakk', 'aaaaah', 'waah', 'aih', 'akh', 'halahh',
'oo', 'tsay', 'hue', 'mwehehe', 'idihh','wehehe', 'atulah',
'se', 'cieee', 'oy' 'adh', 'sii’, 'ebuset', 'assoii', 'nge',
'waw', 'sjjshsjkw', 'weh', 'blablabla', 'fa', 'fi', 'fu',
'hue', 'beuh', 'blabla', 'yuhu', 'blablabla', 'fafifu',
'wasweswos', 'hah', 'heh', 'hoh', 'huhu', 'woy', 'kah',
'wele', 'an', 'oy', 'oh','woi', 'mu', 'a', 'ber', 'in',
'ups', 'sc', 'bro', 'hi', 'yaelah', 'ah', 'ih', 'oalah',
'yekan', 'e', 'bla', 'halah', 'ok', 'hayo', 'shit', 'dih',
'yok', 'hmm', 'hayoloh', 'ye', 'hadah', 'uh', 'hey', 'y',
'c', 'yekan', 'yas', 'halah', 'walah', 'hi', 'lalala',
'euy', 'hey', 'epep', 'ku', 'tit', 'huft', 'ih', 'halah',
'hi', 'yups', 'elah', 'woi', 'brou', 'yak', 'nya', 'haduh',
'akh', 'sihh', 'ha', 'waao', 'blablablabla', 'ahh', 'yes',
'wihh', 'tuu', 'ew’,'widih', 'yakali', 'itu','nahh'])
stop_words.update(additional_stop_words)

def stopword(text):
    # Ubah ke string jika text bukan string
    if not isinstance(text, str):
        text = ' '.join(text)

    # Memisahkan teks menjadi kata-kata
    words = text.split()
```

```
    # Menghapus stop words
    filtered_words = [word for word in words if word not in
stop_words]

    return ' '.join(filtered_words)
twitter['Stopword'] = twitter['Normalisasi'].apply(stopword)

#penghapusan kata-kata khusus
import nltk
from nltk.tokenize import word_tokenize

nama_artis_korea = ["lucas", "siwon", "nct", "skz", "dream",
"twice", "ikon", "itzy", "dreamies", "enhypen", "joon",
"gfriend", "sehunnie", "haechan", "chenle", "minhoo","yeo",
"jingoo", "sehun", "jehun", "enha", "trejo", "once",
"nctzen", "army", "red", "velvet", "bunga", "citra",
"lestari", "rafi", "vernon","txt", "exo", "exol", "han",
"so", "hee", "sohee", "elf", "svt", "sungjin","bcl",
"maudy", "ayunda", "syifa", "yoona", "fuji", "dion","lisa",
"blackpink", "bp", "shenina", "tukul", "jaeh", "igun",
"taec", "nicholas", "saputra", "kepler", "nicsap", "ljs",
"ayuna", "sooman","soohyun", "irene", "ye", "jin",
"seventeen", "choi", "yeo", "jingoo", "lee", "min", "ho",
"suju", "sj", "najwa", "sihab", "dian", "muh","han", "so",
"hee", "sohee", "kim", "soo", "hyun", "seonho", "red",
"velvet", "song", "song", "wendy", "hadju", "sastro",
"suhay", "salim","joong", "ki", "ji", "chang", "wook",
"wayv", "dahyun", "sana", "jaemin", "treasure", "taeil",
"dita", "jonghyun", "jonghyu", "raden", "sebong", "key",
"yeji", "bts", "bangtan", "prilly", "ohmygirl",
"redvelfess", "songjoongki", "teume", "teumezen", "teuarmy",
"carat","ilichil", "sule", "isyana", "jaehyun", "seungyopn",
"kyuhyun", "wiyoko", "soodam", "latuconsina", "songjongki",
"jkt", "kotaro", "shiba", "renjun","hendery", "donghyuk",
"ikon", "wanna one", "wanna", "one", "stray", "kids",
"kidz", "drippin", "dreamin", "xunqis",
"cepmek","raisa","cherrybelle", "rv", "suho", "angga",
"sasongko", "winner", "prince", "gabriel", "sijeuni",
"luna", "maya", "woo", "shik", "sejeong",
"hitritmen","mileapo", "kimsejeong", "iconic", "park","bum",
"ten", "jennie", "jisoo", "jihyo", "iis", "dahlia", "junho",
"yangun", "younghoon", "ana","kino", "mino", "jun", "ji",
"hyun", "hera", "babe", "cabita", "hyung", "sik", "sihoo",
"bruno", "mars", "taylor", "kyungsoo", "nicher","swift",
"super", "junior", "songkang", "winwin", "changwook",
"niscap", "jina", "hyunjin", "newjeans", "fadil","jaidi",
"onsu", "raffi", "ahmad","rans", "astro", "ateez", "oliver",
"nayeon", "dpr", "ian", "raena", "btob", "tvxq",
"tohoshinki", "tohosinki", "ganjar", "pranowo", "yeaji",
```

```
"chungmuro", "rafathar","raline", "bigbang", "faveny",
"ziva", "magnolya", "ryeowook", "dewo", "eko", "krisna",
"rangga", "agung", "kadir", "bela", "surya", "hyun
been","karina", "nikita", "kusuma", "titan", "tyra", "gigi",
"hadid", "kendal","jenner", "fendy", "chanyeol", "midzy",
"jaehyuk", "chelsea", "islan",
"ayu", "tingting", "ting", "dk", "jinan", "ikonic",
"donghwan", "agnes", "mo", "nassar", "iqbal", "ramadhan",
"refal", "iqbaal", "ikonics","dongi", "jungwoo", "selena",
"gomez", "hansol", "baim", "shinee","sjk", "bona", "inyoup",
"greyap", "jungwon", "seono", "joghyun", "sulli",
"joghyu","jeon", "so", "min", "oshbar", "michael",
"hansohi", "kimsono", "felito", "alan", "walker", "hybe",
"jackson", "namjoon", "jina", "liahyoo", "icang","dispatch",
"marvel", "pacmann", "siti", "badriah", "amelia", "tantono",
"one", "piece", "ginting", "hitritmen", "mark", "jo1",
"itzyxultra", "sn","injeuni", "kiming", "fel", "jumpil",
"engene", "ph", "tibo", "wang", "yibo", "aeri", "phi",
"bai", "rvf", "sista", "karang", "aespa", "seon", "minbejo",
"sk", "s", "nuest", "greya", "son", "hu", "hoo", "jongki",
"smtown", "corla", "jart", "sasi", "minho", "joongki",
"hyungsik", "minho", "tasya", "farasha", "deddy",
"nay","sophia", "latjuba", "patricia", "gouw", "jeketi",
"hwang", "inyeop", "trans", "pacminn", "sn", "ilchil",
"jin", "yejin", "xh", "jiniso", "mahabaratha",
"mahabarata","jo", "na", "chilzen", "jeno", "s", "jokowi",
"phs", "avo", "tara", "basro", "adinia", "aldi", "taher",
"kang", "cha", "eun", "latukontraina", "niscap" "jay",
"kevin", "yu", "nabi", "snsd", "seohyun", "jessica", "cho",
"leeteuk", "reveluv", "oneus", "thelegendaris", "inseret",
"lesti", "psj", "seo", "minhyuk", "jong","irun", "yoo",
"ah", "yea", "hyunbeen", "ryewook", "icon", "samuel",
"yujin", "v", "waiji", "gitasav", "jongki", "kendall",
"tinytan", "rj", "hs", "minvo", "seon","jim", "ami",
"manoppo", "inyong", "nunu", "mave", "ran", "afgan",
"erina", "hj", "chen", "natasha", "wilona", "lucinta",
"tatsun", "agnez", "kimsana", "hansohi","koreaidfess",
"straykids", "taehyung", "loona", "runa", "naura", "ohsbar"]

# data tweet
tweet = twitter['Stopword']

def remove_korean_artist_names(tweet):
    cleaned_tokens = []
    for word in word_tokenize(tweet):  # Tokenisasi tweet
menjadi kata-kata
        if word.lower() not in [nama.lower() for nama in
nama_artis_korea]:
            cleaned_tokens.append(word)
```

```
    return cleaned_tokens

# Terapkan fungsi ke setiap tweet dalam kolom 'tweet'
twitter['RemoveSpesific'] = twitter['Stopword'].apply(lambda
x: remove_korean_artist_names(x))

#penghapusan nama brand
nama_brand = ["somethinc", "neo", "coffee", "sedaap",
"brandk", "whitelab", "scarlett", "apieu", "samsung",
"wardah", "everwhite", "etude", "house", "scarlet",
"kopiko","sedap", "trueve", "scarlettt", "tokped",
"tokopedia", "shopee", "lazada", "skarlet", "victoria",
"secret", "msglow", "p0nds", "misedap", "neokopi",
"axis","ms. glow", "baifern", "maybelline", "gojek",
"sariayu", "lan*ige", "erigo", "azarine", "hanasui",
"implora", "emina", "the", "originote",
"skintific","avoskin", "skingame", "click", "azzarin",
"dior", "gucci", "prada", "chanel", "lv", "ysl",
"lemonilo","l3m0nil0", "elips", "ultra", "milk", "nu",
"green", "tea", "blibli", "bukalapak", "indomie", "viva",
"nike", "s0methinc", "labore", "bnb", "blp", "dearme",
"nature", "republic", "republik", "natrep", "barenbliss",
"realfood", "real", "food", "ms", "glow", "bloomka", "kara",
"sasa", "revlon", "este", "menantea", "mie", "sedap",
"some", "by", "mi", "elzatta", "kintakun","ruangguru",
"teh", "botol", "sosro", "nyamnyung", "makeover", "scar",
"skarlet", "chuba", "scrwhtn", "calvin", "fila", "pantene",
"sabana", "cerave", "minvoo", "martha",
"tilaar", "qtelq", "mogu", "qtela", "kusuka", "channel",
"laneige", "madame", "gie",  "bts", "meal", "mcd",
"siminvest", "sinarmas", "indomilk", "hermes", "djarum",
"sunsilk", "histoire naturelle", "mujigae", "cosrx", "anua",
"wl", "miranda", "garnier", "beauty", "of", "joseon",
"nutrishe", "y.o.u", "you", "snail", "truecica", "kfc",
"mcd", "f&b",
"amanda", "sweet", "kokona", "nyx", "mac", "estee",
"lauder", "lancome", "clio", "npure", "vavl", "ajaib", "mr",
"potato", "swalow", "mineral", "botanica",
"innisfree","nacific", "calvin", "puma", "wingsfood",
"histoire",  "naturelle", "lacoco", "pyunkangyul",
"barenblis",  "purbasari", "inez", "noolab", "romand",
"peripera","holika", "wclinic", "joseon", "ordinary",
"olay", "saem", "sbm", "cold", "brew", "paldofood",
"elizabeth", "axis-y", "somebymi", "benton", "penshoppe",
"rojo", "lele","Saé'Kin", "saekin", "ivy", "club", "saint",
"laurent", "scarlet", "benings", "plossa", "eucalyptus",
"iphone", "evershine", "bobby", "brown", "top", "lady",
"ck",
```

```
"burberry", "valentino", "bvlgari", "deelel", "bighit",
"wings", "paragon", "fmcg", "tbz", "deine", "gofood",
"sunlight", "soklin", "carasun", "ultramilk",
"manyo","shopeeid", "bukopin", "kb", "aspabukopin",
"changwookkintakun", "ponds", "scarllet", "bloomate",
"oren", "sopi", "shoppe", "nutriville", "pororo", "emon",
"bioderma", "klosap", "ellips", "purivera", "cew", "nr",
"siminvestasi", "jdid", "got", "nyugrinti", "sumsang",
"shopback", "gacoan", "inisfree", "rinso", "celine",
"charm","schapp", "ultraxitzy", "givenchy", "chajun",
"nuya", "senka", "ellips", "eco", "running", "man", "tsn",
"jjh", "balmain", "dearmebeauty", "versace", "nabati",
"google", "hada", "labo", "loreal", "saekin", "saman",
"helvetia", "jay", "marina", "ajay"]

tweet = twitter['RemoveSpesific']

def remove_brand_names(tweet):
    # Gabungkan kembali daftar kata-kata menjadi satu string
    tweet_string = ' '.join(tweet)

    cleaned_tokens = []
    for word in word_tokenize(tweet_string):  # Tokenisasi
tweet menjadi kata-kata
        if word.lower() not in [nama.lower() for nama in
nama_brand]:
            cleaned_tokens.append(word)
    return cleaned_tokens

# Terapkan fungsi ke setiap tweet dalam kolom 'tweet'
twitter['RemoveBrandName'] =
twitter['RemoveSpesific'].apply(lambda x:
remove_brand_names(x))

#stemmer
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import swifter
# Daftar kata khusus yang ingin dipertahankan
kata_khusus = ["semoga", "setuju", "semua", "bali",
"alasan", "nangis", "agensi", "senang", "relasi", "nilai",
"korelasi", "busana", "ketik", "kesan", "pesan",
               "muda", "laris", "capai"]

# create stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# stemmed
def stemmed_wrapper(term):
```

```
    # Periksa apakah term ada dalam daftar kata khusus
    if term in kata_khusus:
        return term  # Jika term ada dalam daftar kata
khusus, kembalikan term tanpa melakukan stemming
    else:
        return stemmer.stem(term)

term_dict = {}

for document in twitter['RemoveBrandName']:
    for term in document:
        if term not in term_dict:
            term_dict[term] = ' '

for term in term_dict:
    term_dict[term] = stemmed_wrapper(term)

# apply stemmed term to dataframe
def get_stemmed_term(document):
    return [term_dict.get(term, term) for term in document]

# Contoh penggunaan
twitter['Stemmer'] =
twitter['RemoveBrandName'].apply(get_stemmed_term)
```

Source Code 2 untuk Pembuatan *Dictionary*

```
# Menginisialisasi dictionary untuk merepresentasikan kelas
kata
kelas_kata = {}

# Mengisi dictionary dengan kata-kata positif, negatif, dan
netral
for kata in kata_positif:
    kelas_kata[kata] = 'positif'

for kata in kata_negatif:
    kelas_kata[kata] = 'negatif'

for kata in kata_netral.iloc[:, 0].tolist():
    kelas_kata[kata] = 'netral'

# data tweet
data_tweet = twitter['Preprocess']

# Membuat dictionary untuk merepresentasikan kelas-kelas
dari data tweet
kelas_tweet = {}
```

```
# Inisialisasi jumlah kata positif, negatif, dan netral
dalam sebuah tweet
# Iterasi melalui setiap tweet
for tweet in data_tweet:
    kata_tweet = tweet.lower().split()
    jumlah_positif = 0
    jumlah_negatif = 0
    jumlah_netral = 0

    for kata in kata_tweet:
        if kata in kata_positif:
            jumlah_positif += 1
        elif kata in kata_negatif:
            jumlah_negatif += 1
        elif kata in kata_netral:
            jumlah_netral += 1
        else:
            # Jika kata tidak ditemukan dalam vokabular,
beri label netral pada tweet tersebut
            kelas_tweet[tweet] = 'netral'

    # Menentukan sentimen berdasarkan jumlah kata-kata
positif, negatif, dan netral
    if jumlah_positif > jumlah_negatif:
        kelas_tweet[tweet] = 'positif'
    elif jumlah_negatif > jumlah_positif:
        kelas_tweet[tweet] = 'negatif'
    else:
        kelas_tweet[tweet] = 'netral'
```

Source Code 3 untuk Membangun dan Melatih Model *Word2vec*

```
from tqdm import tqdm
tqdm.pandas(desc="progress-bar")
import gensim
from gensim.models.word2vec import Word2Vec
import multiprocessing
from sklearn import utils

# parameter-parameter untuk model Word2Vec
cores = multiprocessing.cpu_count()
vector_size = 100
window = 5
sg = 1  # Skip-gram (sg=1), CBOW (sg=0)
negative = 5
threshold_freq = 2
```

```
workers = cores
alpha = 0.005
min_alpha = 0.005
# Bangun model Word2Vec
word2vec_model = Word2Vec(vector_size=vector_size,
window=window, sg=sg, negative=negative,
min_count=threshold_freq, workers=workers, alpha=alpha,
min_alpha=min_alpha, sample=1e-3)
# Bangun kosakata (vocabulary) untuk model Word2Vec
word2vec_model.build_vocab([x.split() for x in tqdm(X)])

#melatih model word2vec
#melatih model wor2vec yang telah dibangun sebelumnya
import time
start_time = time.time()
for epoch in range(10):
  word2vec_model.train(utils.shuffle([x.split() for x in
tqdm(X)]), total_examples=len(X), epochs=1)
  word2vec_model.alpha -= 0.002
  word2vec_model.min_alpha = word2vec_model.alpha

end_time = time.time()
execution_time = end_time - start_time
```

Source Code 4 untuk Proses Tokenisasi dengan *Tokenizer* dan Proses *Padding*

```
# menggunakan modul Tokenizer dari Keras untuk melakukan
tokenisasi teks dan mengubahnya menjadi urutan angka
(sequences)
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer()
tokenizer.fit_on_texts(X)
sequences = tokenizer.texts_to_sequences(X)

#paading pada data X
sequences_test = tokenizer.texts_to_sequences(X)
X_seq = pad_sequences(sequences_test, maxlen=max(length))
```

Source Code 5 untuk Matriks *Embedding*

```
import numpy as np
num_words = len(tokenizer.word_index) + 1
embedding_matrix = np.zeros((num_words, 100))
```

```
for word, i in tokenizer.word_index.items():
    if i >= num_words:
        continue
    embedding_vector = embedding_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

Source Code 6 untuk Pembentukan Model CNN

```
from keras.models import Sequential
from keras.layers import Embedding, Conv1D, MaxPooling1D,
Dense, Dropout, Flatten
from keras.optimizers import Adam
from keras import regularizers

model = Sequential()
model.add(Embedding(num_words, 100,
weights=[embedding_matrix], input_length=max(length),
trainable=True))
model.add(Conv1D(filters=32, kernel_size=2,
activation='relu'))
model.add(MaxPooling1D(pool_size=2, strides=None,
padding='valid'))
model.add(Dropout(0.1))
model.add(Flatten())
model.add(Dense(32, activation='relu',
kernel_regularizer=regularizers.l2(1e-4)))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer=Adam(learning_rate=0.001,
beta_1=0.9, beta_2=0.999),
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

Source Code 7 untuk Pembentukan Model GRU

```
from keras.models import Sequential
from keras.layers import Embedding, GRU, MaxPooling1D,
Dense, Dropout, GlobalMaxPooling1D, SpatialDropout1D
from keras.optimizers import Adam
from keras import regularizers

model = Sequential()
model.add(Embedding(num_words, 100,
weights=[embedding_matrix], input_length=max(length),
trainable=True))
model.add(SpatialDropout1D(0.2))
model.add(GRU(32, activation='tanh', return_sequences=True))
```

```
model.add(GlobalMaxPooling1D())
model.add(Dropout(0.2))
model.add(Dense(32, activation='relu',
kernel_regularizer=regularizers.l2(1e-4)))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer=Adam(learning_rate=0.001,
beta_1=0.9, beta_2=0.999),
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

Source Code 8 untuk Penggunaan *K-Fold Cross Validation*

```
from sklearn.model_selection import KFold
kFold = KFold(n_splits=5, shuffle=True, random_state=42)
for train_index, test_index in kFold.split(X_seq, y):
    X_train, X_test = X_seq[train_index], X_seq[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

Source Code 9 untuk Pelatihan Model CNN

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
filepath = "CNN_best_weights.{epoch:02d}-
{val_accuracy:.4f}.hdf2"
checkpoint = ModelCheckpoint(filepath,
monitor='val_accuracy', verbose=1, save_best_only=True,
mode='max')
early_stopping = EarlyStopping(monitor='val_accuracy',
patience=5, verbose=1, restore_best_weights=True)
# Callback list yang akan digunakan dalam fungsi fit
callbacks_list = [checkpoint, early_stopping]
history_list = []
fold_no = 1
for train_index, test_index in kFold.split(X_seq, y):
    print('------------------------------------------------
--------------------')
    print(f'Training for fold {fold_no} ...')
    X_train, X_test = X_seq[train_index], X_seq[test_index]
    y_train, y_test = y[train_index], y[test_index]

    history = model.fit(X_train, y_train,
validation_data=(X_test, y_test), epochs=10, batch_size=32,
callbacks=callbacks_list)
    # Simpan histori pelatihan
    history_list.append(history)
    fold_no = fold_no + 1
```

Source Code 10 untuk Pelatihan Model GRU

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
filepath = "GRU_best_weights.{epoch:02d}-
{val_accuracy:.4f}.hdf2"
checkpoint = ModelCheckpoint(filepath,
monitor='val_accuracy', verbose=1, save_best_only=True,
mode='max')
early_stopping = EarlyStopping(monitor='val_accuracy',
patience=5, verbose=1, restore_best_weights=True)
# Callback list yang akan digunakan dalam fungsi fit
callbacks_list = [checkpoint, early_stopping]
history_list = []
fold_no = 1
for train_index, test_index in kFold.split(X_seq, y):
    print('------------------------------------------------
--------------------')
    print(f'Training for fold {fold_no} ...')
    X_train, X_test = X_seq[train_index], X_seq[test_index]
    y_train, y_test = y[train_index], y[test_index]

    history = model.fit(X_train, y_train,
validation_data=(X_test, y_test), epochs=7, batch_size=16,
callbacks=callbacks_list)
    # Simpan histori pelatihan
    history_list.append(history)
    fold_no = fold_no + 1
```

Source Code 11 untuk Pengujian dengan Data Baru

```
import pandas as pd
from sklearn.metrics import confusion_matrix,
classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Path ke file CSV untuk data baru
file_path2 = '/content/drive/My Drive/Colab
Notebooks/data/tweet_baru.csv'
# Baca file CSV menggunakan pandas
data_tunggal = pd.read_csv(file_path2)
x_single = data_tunggal['Tweet']
y_single = data_tunggal['Kelas'].map({'positif': 2,
'negatif': 0, 'netral': 1}).values
cnn_model = load_model('CNN_best_weights.01-0.9776.hdf2')
gru_model = load_model('GRU_best_weights.02-0.9830.hdf2')
# Membuat prediksi dengan model CNN
# Menentukan sentimen dari hasil prediksi
```

```
predictions_single = cnn_model.predict(x_single_pad)
predicted_sentiment_single_CNN =
np.argmax(predictions_single, axis=1)
#dengan model gru
test_predictions = gru_model.predict(x_single_pad)
predicted_sentiment_single_GRU = np.argmax(test_predictions,
axis=1)

# Evaluasi model CNN
loss, accuracy = cnn_model.evaluate(x_single_pad, y_single)
print(f'\nTest Loss CNN: {loss:.4f}, Accuracy CNN:
{accuracy:.4f}')

# Hitung dan tampilkan confusion matrix
conf_mat = confusion_matrix(y_single,
predicted_sentiment_single_CNN)
print('\nConfusion Matrix Model CNN:')
print(conf_mat)

# Tampilkan confusion matrix menggunakan heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues',
xticklabels=['Negatif', 'Netral', 'Positif'],
yticklabels=['Negatif', 'Netral', 'Positif'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Evaluasi model GRU
loss, accuracy = gru_model.evaluate(x_single_pad, y_single)
print(f'\nTest Loss GRU: {loss:.4f}, Accuracy GRU:
{accuracy:.4f}')
# Hitung dan tampilkan confusion matrix
conf_mat = confusion_matrix(y_single,
predicted_sentiment_single_GRU)
print('\nConfusion Matrix Model GRU:')
print(conf_mat)

# Tampilkan confusion matrix menggunakan heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues',
xticklabels=['Negatif', 'Netral', 'Positif'],
yticklabels=['Negatif', 'Netral', 'Positif'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show(
```