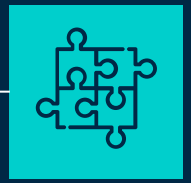
Analisis Sentimen Jokowi di Media Sosial Twitter

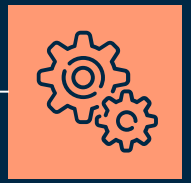
1119003	Levin Martinus
1119005	Aristo Demos
1119006	William Juniar
1119007	Timothy Ray
1119009	Julian Ely
1119011	Andreas Virdian

TABLE OF CONTENTS



01

DATA
COLLECTION

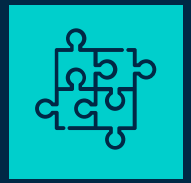


02

TEXT
PREPROCESSING

03

SVM
MODELLING



01. DATA COLLECTION

Dataset

Dataset yang digunakan didapatkan melalui Twitter API dengan kata kunci “jokowi”. Hasilnya didapat sebanyak 277 tweet berbahasa Indonesia yang diambil dari tanggal 20 Februari 2022 sampai tanggal 23 Februari 2022. Tweet tersebut disimpan dalam ekstensi file .csv.

	Unnamed: 0	id	text	created_at
0	0	1495415872164229120	RT @LurahIstana: Jokowi sudah mengesahkan pera...	2022-02-20T15:11:42.000Z
1	1	1495415869673177090	RT @NabilaAr__: Kami para santri sangat tepat ...	2022-02-20T15:11:41.000Z
2	2	1495415863377539074	RT @Sangkuriang5551: FOKUS KE JOKOWI!\n\nMasal...	2022-02-20T15:11:39.000Z
3	3	1495415855546798083	RT @jokowi: Srikandi-srikandi bulutangkis putr...	2022-02-20T15:11:38.000Z
4	4	1495415849015930880	RT @CasanovaX8X: LOMBOK #MandalikaCircuit\n#Jo...	2022-02-20T15:11:36.000Z
5	5	1495415838991859713	RT @geloraco: BPJS Kesehatan Jadi Syarat Jual ...	2022-02-20T15:11:34.000Z
6	6	1495415827960516623	RT @KangUtang04: #JokowiTheRealDictator \n#Jok...	2022-02-20T15:11:31.000Z
7	7	1495415826912264194	RT @kompascom: Hasil 71 persen tersebut merupa...	2022-02-20T15:11:31.000Z
8	8	1495415825309724675	RT @LurahIstana: Jokowi sudah mengesahkan pera...	2022-02-20T15:11:30.000Z
9	9	1495415817768300546	RT @OposisiCerdas: Tidak Hanya Jual-beli Tanah...	2022-02-20T15:11:29.000Z
10	10	1495415816275505155	RT @CNNIndonesia: Survei Indikator: 70 Persen ...	2022-02-20T15:11:28.000Z

Tweet Extraction - Initialization

Dataset tersebut diekstraksi dengan menggunakan API Twitter.

Pertama, BEARER_TOKEN harus diisi terlebih dahulu. BEARER_TOKEN akan diberikan secara rahasia saat kita membuat akun Twitter Developer Platform di developer.twitter.com.

Fungsi bearer_oauth bertujuan untuk melakukan authorization dari token yang sudah diisi.

Kemudian, fungsi create_url akan menciptakan url tujuan beserta query, max. Results, dan field-field yang dibutuhkan untuk menampung data ekstraksi

Fungsi connect_to_endpoint bertujuan menghubungkan dengan endpoint yang dituju beserta status dari response tersebut

```
BEARER_TOKEN = ''
```

```
def bearer_oauth(r):
    r.headers['Authorization'] = f"Bearer {BEARER_TOKEN}"
    r.headers['User-Agent'] = "v2RecentSearchPython"
    return r
```

```
def create_url(keyword, max_results=10):
    search_url = 'https://api.twitter.com/2/tweets/search/recent'

    query_params = {
        'query': keyword,
        'max_results': max_results,
        'tweet.fields': 'id,text,created_at,lang',
        'next_token': {}
    }
    return (search_url, query_params)
```

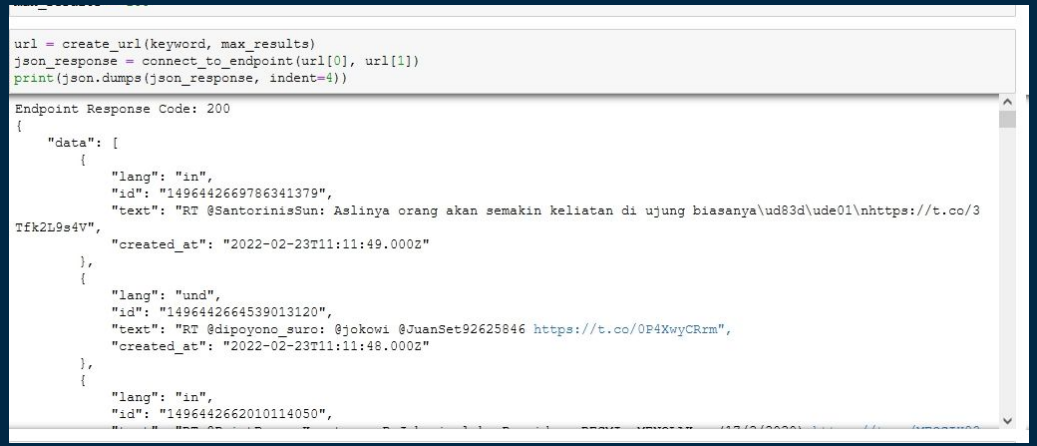
```
def connect_to_endpoint(url, params, next_token=None):
    params['next_token'] = next_token
    response = requests.get(url, auth=bearer_oauth, params=params)
    print("Endpoint Response Code: " + str(response.status_code))
    if response.status_code != 200:
        raise Exception(response.status_code, response.text)
    return response.json()
```

Tweet Extraction - Query

Setelah fungsi-fungsi dibuat, maka dicantumkan apa yang ingin kita lakukan dengan API. Dalam pengambilan data tweet, cantumkan keyword beserta max_results (berapa banyak).

Lalu, cantumkan kedua data tersebut sebagai parameter dalam membuat url.

Setelah url dibuat, lakukan koneksi dengan endpoint tersebut dan hasilnya akan disimpan ke dalam format JSON seperti gambar di samping.



```
url = create_url(keyword, max_results)
json_response = connect_to_endpoint(url[0], url[1])
print(json.dumps(json_response, indent=4))
```

```
Endpoint Response Code: 200
{
    "data": [
        {
            "lang": "in",
            "id": "1496442669786341379",
            "text": "RT @SantorinisSun: Aslinya orang akan semakin keliatan di ujung biasanya\ud83d\ude01\nhttps://t.co/3
Tfk2L9s4V",
            "created_at": "2022-02-23T11:11:49.000Z"
        },
        {
            "lang": "und",
            "id": "1496442664539013120",
            "text": "RT @dipoyono_suro: @jokowi @JuanSet92625846 https://t.co/0P4XwyCRrm",
            "created_at": "2022-02-23T11:11:48.000Z"
        },
        {
            "lang": "in",
            "id": "1496442662010114050",
```

```
keyword = "jokowi"
max_results = 100
```

Tweet Extraction – Storing Data

Setelah data berhasil diambil dengan melakukan koneksi terhadap endpoint, data dimasukkan ke dalam variabel df sebagai data frame.

Data akan di-filter untuk memilih tweet yang berbahasa Indonesia saja.

Kemudian hasil extraction akan disimpan ke dalam sebuah file csv bernama "jokowi_twitter_sentiment.csv" agar data lebih mudah diolah.

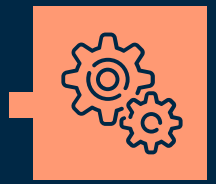
```
df = pd.DataFrame(json_response['data'])
```

```
df_filtered = df[df['lang'] == 'in'].drop('lang', axis=1)
df_filtered
```

...	...	...	...
95	1496442136132452360	RT @CNNIndonesia: "Kalau publik enggak ribut, ...	2022-02-23T11:09:42.000Z
96	1496442134698020865	RT @djaritakirana: Terus kawal kasus duo terdu...	2022-02-23T11:09:42.000Z
97	1496442132844134404	RT @OposisiCerdas: Soroti Kinerja Jokowi, Riza...	2022-02-23T11:09:41.000Z
98	1496442127999733766	RT @dennysirregar7: Mau muntah gua... \n\nhttp...	2022-02-23T11:09:40.000Z
99	1496442120005369859	RT @nafikulla: @jokowi betul pak, msyrkat yg k...	2022-02-23T11:09:38.000Z

92 rows × 3 columns

```
df_filtered.to_csv('jokowi_twitter_sentiment.csv', mode='a')
```



02. TEXT PREPROCESSING

Data Cleaning

Pertama-tama, file jokowi_twitter_sentiment.csv dibaca menggunakan pandas dalam bentuk tabel data.

Kemudian, file jokowi_twitter_sentiment_labeled.txt dimasukkan ke tabel tersebut dalam kolom label.

Lalu, data yang kolom textnya terduplikat akan dibuang dari tabel data.

```
df = pd.read_csv('jokowi_twitter_sentiment.csv')
```

```
with open('jokowi_twitter_sentiment_labeled.txt', 'r') as file:
    label = file.read().splitlines()
df['label'] = label
df
```

Drop duplicate

```
clean_df = df[df['text'].duplicated()==False]
clean_df.reset_index(inplace=True)
clean_df
```

Manual Labelling

id	text	created_at	label
1495415872164229120	RT @LurahIstana: Jokowi sudah mengesahkan pera...	2022-02-20T15:11:42.000Z	Positive
1495415869673177090	RT @NabilaAr__: Kami para santri sangat tepat ...	2022-02-20T15:11:41.000Z	Positive
1495415863377539074	RT @Sangkuriang5551: FOKUS KE JOKOWI!\n\nMasal...	2022-02-20T15:11:39.000Z	Negative
1495415855546798083	RT @jokowi: Srikandi-srikandi bulutangkis putr...	2022-02-20T15:11:38.000Z	Positive
1495415849015930880	RT @CasanovaX8X: LOMBOK #MandalikaCircuit\n#Jo...	2022-02-20T15:11:36.000Z	Neutral

Pemberian label Positive, Negative atau Neutral dilakukan secara manual dengan mengecek tweet di dataset. Label tersebut tersimpan di file jokowi_twitter_sentiment_labeled.txt.

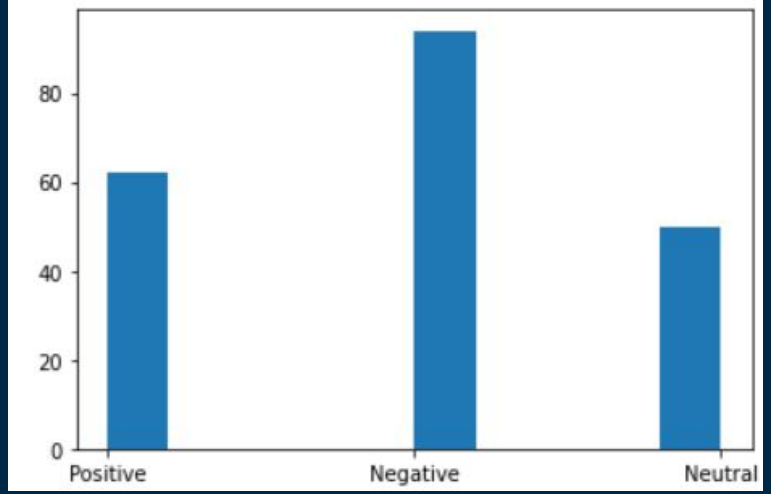
Manual Labelling

Dari hasil manual labelling, sebanyak 94 data termasuk dalam sentimen negatif, 62 sentimen positif, dan 50 netral.

```
1  data['manual_label'].value_counts()
```

```
Negative    94
Positive    62
Neutral     50
Name: manual_label, dtype: int64
```

```
1  plt.figure()
2  plt.hist(data['manual_label'])
3  plt.show()
```



Lexicon Labelling

Negation words

```
# negation words, negate the sentiment value
with open('negation.txt', 'r') as file:
    negation = file.read().splitlines()
```

Booster words

```
# booster words, add more value to sentiment word
with open('booster.txt', 'r') as file:
    booster = file.read().splitlines()
booster = [sent.split() for sent in booster]
booster_words = [row[0] for row in booster]
```

Sentiment words

```
# sentiment words with value range -5 to 5
with open('sentiment_value.txt', 'r') as file:
    sentiment_valued = file.read().splitlines()
sentiment_valued = [sent.split() for sent in sentiment_valued]
sentiment_valued_words = [row[0] for row in sentiment_valued]
```

Pemberian label secara lexicon/rule-based menggunakan 3 jenis kata penting:

Negation words: daftar kata negasi seperti tidak, sulit, jangan dan lainnya. Kata ini mengubah nilai sentimen dari positif (+) menjadi negatif (-) dan sebaliknya.

Booster words: daftar kata tambah seperti sangat, agak, lebih dan lainnya. Kata ini memberikan bobot tambahan ke nilai sentimen antara -2 sampai 2.

Sentiment words: daftar kata yang mengandung sentimen positif dan sentimen negatif dengan pemberian bobot -5 sampai 5.

Lexicon Labelling

Pemrosesan data teks memerlukan filter:

Stopwords: kata umum yang sering muncul dan dianggap tidak memiliki makna. Kata ini disaring kembali agar kata penting tidak ikut terhapus.

Noise words: kata-kata yang tidak jelas, seperti salah ketik/typo.

Slang words: kata gaul/singkatan/musiman yang kemudian diubah menjadi bentuk kata baku.

Noise words

```
# noise words, unnecessary words after folding
with open('noise.txt', 'r') as file:
    noise = file.read().splitlines()
```

Slang words

```
# slang words
slang = pd.read_csv('slang.csv')
slang_list = slang.to_numpy(dtype='str')
```

```
def unslang_word(word):
    if word not in slang_list[:,0]:
        return [word]

    index = slang_list[:,0].tolist().index(word)
    new_word = slang_list[index,1]

    return [w for w in new_word.split()]
```

Stopwords

```
stopwords = set(nltk.corpus.stopwords.words('indonesian'))
```

```
# filter stopwords if exist in segmentations, negation, booster
stopwords_copy = stopwords.copy()
for word in stopwords_copy:
    if word in negation or word in booster_words or word in sentiment_valued_words:
        stopwords.discard(word)
```

Lexicon Labelling

```
word_dict = {}
processed_tweets = []

for tweet in tweets:

    # sentence segmentation
    sentences = nltk.tokenize.sent_tokenize(tweet)
    new_words = []

    for sentence in sentences:

        # punctuations and numbers to be removed, '@' to identify mentioned user, '-' to identify sentiment word
        remove = string.punctuation.replace('@','').replace('-','')+'"'+'0123456789'

        # replace word, remove non ASCII, remove punctuation, remove number, remove whitespaces, to lower
        folded = sentence.replace('&amp','&').replace('&', 'dan').replace('\n',' ').replace('RT','').replace('minyak goreng','mi
nyak-goreng').encode('ascii','ignore').decode('ascii').translate(str.maketrans('','',remove)).strip().lower()

        # word tokenizing
        words = nltk.word_tokenize(folded)

        isUser = False
        for word in words:

            # remove mentioned user
            # the next word after '@' will be user
            if isUser:
                isUser = not isUser
                continue
            if word == '@':
                isUser = not isUser
                continue

            # remove url
            if word.startswith('htt'):
                continue
```

Text Normalization

- Menghapus punctuation
- Mengubah kata menjadi lowercase
- Menghapus angka
- Menghapus spasi
- Mengganti beberapa kata dan simbol
- Menghapus mention
- Menghapus “RT”
- Menghapus noise
- Mengubah kata gaul/singkatan/musiman

Lexicon Labelling

Hasil dari kata yang sudah dinormalisasi

```
            # remove noise
            if word in noise:
                continue

            # normalize slang word
            for new_word in unslang_word(word):

                # stopword removal
                if new_word in stopwords:
                    continue

                # add processed word
                new_words.append(new_word)
                if new_word in word_dict.keys():
                    word_dict[new_word] += 1
                else:
                    word_dict[new_word] = 1

    processed_tweets.append(new_words)

processed_tweets
```

```
[['jokowi',
  'mengesahkan',
  'peraturan',
  'wajib',
  'badan',
  'penyelenggara',
  'jaminan',
  'sosial',
  'diterapkan',
  'aspek',
  'baik',
  'pengurusan',
  'surat',
  'surat'],
 ['santri',
  'sangat',
  'memilih',
  'erick',
  'thohir',
  'penerus',
  'presiden',
  'jokowi',
  'indonesia',
```

Bag of words

```
1  word_dict
```

```
{'jokowi': 107,
 'mengesahkan': 1,
 'peraturan': 1,
 'wajib': 3,
 'badan': 15,
 'penyelenggara': 8,
 'jaminan': 8,
 'sosial': 7,
 'diterapkan': 1,
 'aspek': 1,
 'baik': 3,
 'pengurusan': 1,
 'surat': 6,
 'santri': 1,
 'sangat': 4,
 'memilih': 1,
 'erick': 3,
 'thohir': 3,
 'penerus': 1,
 'presiden': 29,
 'indonesia': 25,
```

Lexicon Labelling

```python
def sentiment_labelling(words):
    val = 0

    for i in range(len(words)):
        sentiment_val = 0

        # check if the word is sentiment word
        if words[i] in sentiment_valued_words:
            sentiment_val = int(sentiment_valued[sentiment_valued_words.index(words[i])][1])

            # check if the word before is a booster
            boosterBefore = False
            if i > 0:
                if words[i-1] in booster_words:
                    boosterBefore = True

                    # if sentiment positive then add, if sentiment negative then substract
                    if sentiment_val > 0:
                        sentiment_val += int(booster[booster_words.index(words[i-1])][1])
                    elif val < 0:
                        sentiment_val -= int(booster[booster_words.index(words[i-1])][1])

            # check if the word after is a booster
            if i < len(words)-1:
                if words[i+1] in booster_words:

                    # if sentiment positive then add, if sentiment negative then substract
                    if sentiment_val > 0:
                        sentiment_val += int(booster[booster_words.index(words[i+1])][1])
                    elif val < 0:
                        sentiment_val -= int(booster[booster_words.index(words[i+1])][1])

            # check if the word before the booster is a negation
            if boosterBefore and i > 1:
                if words[i-2] in negation:
                    sentiment_val *= -1

            # if word before is not booster, check if the word before is a negation
            elif words[i-1] in negation:
                sentiment_val *= -1

        val += sentiment_val

    return val
```

```python
sentiment_values = [sentiment_labelling(words) for words in processed_tweets]
print(sentiment_values)
```

```
[5, -1, -6, 0, 0, -2, -8, 3, 1, 2, -8, -2, -8, 1, 1, 2, 2, 2, 0, -1, -1, 4, -2, -4,
4, -2, 5, 0, 0, 1, 1, 0, -1, -5, -8, 3, 1, -2, 0, 0, 6, 2, -3, -3, -2, -5, -6, 0, 0
0, -8, 0, 0, 0, -2, 0, 2, -8, -10, 11, 0, 0, 0, 2, 0, 0, 0, 0, 0, -3, -2, 0, 0, 2,
0, 0, -4, 0, -2, 2, 2, -2, -2, -3, 2, -3, 7, 1, 0, 0, 0, -1, 1, 0, -2, 2, 0, -4, -3
2, 4, -7, 5, 2, 0, 0, 0, 0, -5, -2, -2, -1, 4, 1, 0, -2, -3, 0, -7, 3, -2, 4, 2, -2
2, 0, 7, 5, 1, -4, 0, -2, 0, -4, -5, -1, 0, 2, -3, 0, -4, 0, -2, -2]
```

```python
def classify_value(value):
    if value > 0:
        return 'Positive'
    elif value == 0:
        return 'Neutral'
    else:
        return 'Negative'
```

Data teks yang ternormalisasi akan diberikan nilai sentimen dengan memperhatikan kata sentimen, kata tambah sebelum dan sesudah kata sentimen, dan kata negasi sebelum kata sentimen.
Nilai sentimen diubah menjadi label “Positive” untuk nilai > 0, label “Neutral” untuk nilai = 0, dan label “Negative” untuk nilai < 0.

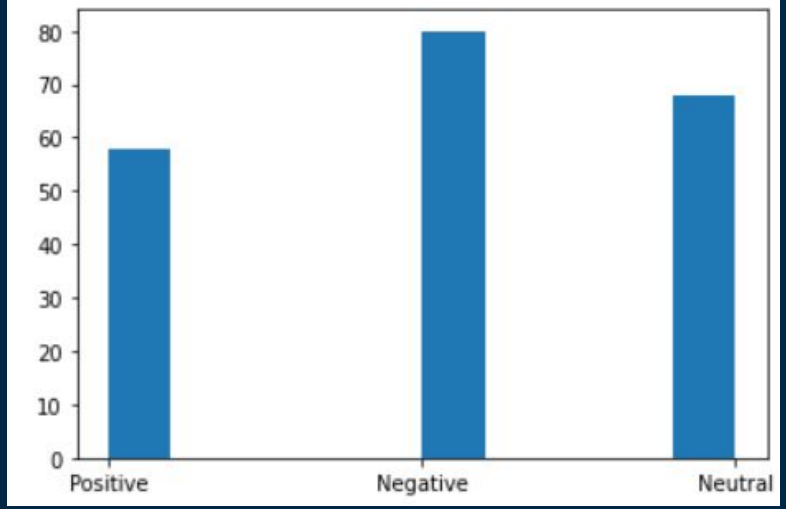
Lexicon Labelling

Dari hasil lexicon labelling, sebanyak 80 data termasuk dalam sentimen negatif, 68 sentimen positif, dan 58 netral.

```
1  data['lexicon_label'].value_counts()
```

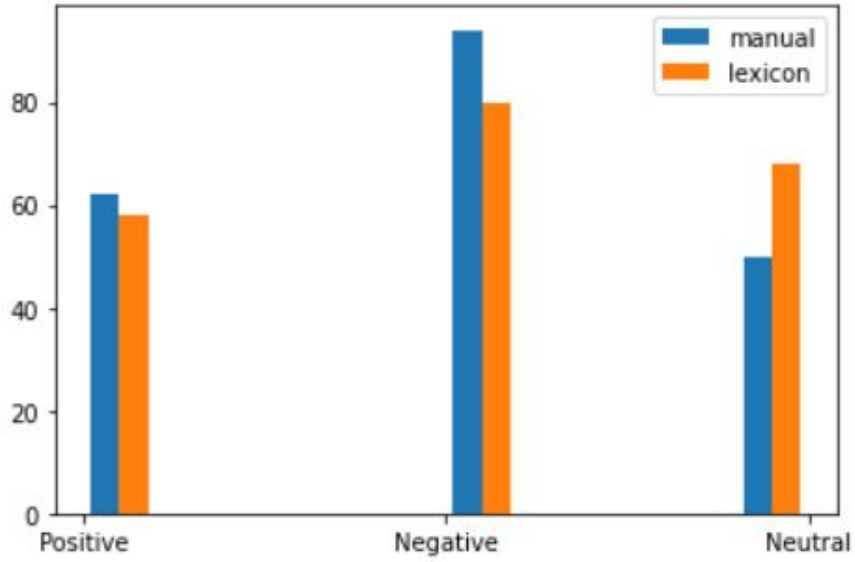
```
Negative    80
Neutral     68
Positive    58
Name: lexicon_label, dtype: int64
```

```
1  plt.figure()
2  plt.hist(data['lexicon_label'])
3  plt.show()
```



Perbandingan Jumlah Positive, Negative dan Neutral manual_label dengan lexicon_label

```
plt.hist([data['manual_label'], data['lexicon_label']], label=['manual', 'lexicon'])
plt.legend()
plt.show()
```



Cara **manual** menghasilkan sentimen **positif** dan **negatif** yang lebih **banyak** dibandingkan lexicon, tetapi **lexicon** lebih **banyak** menghasilkan sentimen **netral**

03. SVM MODELLING.

SVM Modelling pada Lexicon Label

Membuat model diawali dengan membagi antara data training dan data testing

Komposisi yang kami gunakan adalah 80% training dan 20% testing, kemudian akan dimasukkan ke dalam data frame baru, lexicon_df_train80 dan lexicon_df_train20

```
lexicon_train_X, lexicon_test_X, lexicon_train_Y, lexicon_test_Y = model_selection.train_test_split(data['tweet'], data['lexicon_label'], test_size = 0.2, random_state = 0)
```

```
lexicon_df_train80 = pd.DataFrame()
lexicon_df_train80['tweet'] = lexicon_train_X
lexicon_df_train80['lexicon_label'] = lexicon_train_Y

lexicon_df_test20 = pd.DataFrame()
lexicon_df_test20['tweet'] = lexicon_test_X
lexicon_df_test20['lexicon_label'] = lexicon_test_Y
```

TF-IDF lexicon labelling

Hasil TF-IDF dari lexicon labelling

```
# TF-IDF

from sklearn.feature_extraction.text import TfidfVectorizer

lexicon_tfidf_vect_8020 = TfidfVectorizer(max_features = 5000)
lexicon_tfidf_vect_8020.fit(data['tweet'])
lexicon_train_X_tfidf_8020 = lexicon_tfidf_vect_8020.transform(lexicon_df_train80['tweet'])
lexicon_test_X_tfidf_8020 = lexicon_tfidf_vect_8020.transform(lexicon_df_test20['tweet'])
```

```
lexicon_tfidf_vect_8020
```

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=1.0, max_features=5000,
                min_df=1, ngram_range=(1, 1), norm='l2', preprocessor=None,
                smooth_idf=True, stop_words=None, strip_accents=None,
                sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, use_idf=True, vocabulary=None)
```

Hasil TF-IDF lexicon manual dan testing

```
print(lexicon_train_X_tfidf_8020)
```

```
  (0, 1047)     0.25796689169680226
  (0, 743)      0.22626075585226202
  (0, 711)      0.38910924001544345
  (0, 654)      0.25796689169680226
  (0, 471)      0.45252151170452404
  (0, 414)      0.2394199911849753
  (0, 377)      0.07811620323624119
  (0, 357)      0.2394199911849753
  (0, 338)      0.2965633657027805
  (0, 332)      0.25796689169680226
  (0, 132)      0.25796689169680226
  (0, 113)      0.22626075585226202
  (0, 21)       0.25796689169680226
  (1, 980)      0.3673969731176012
```

```
print(lexicon_test_X_tfidf_8020)
```

```
  (0, 968)      0.18379737951971664
  (0, 804)      0.21983825902084567
  (0, 607)      0.6023531103123709
  (0, 548)      0.25064440424955425
  (0, 520)      0.21983825902084567
  (0, 477)      0.21983825902084567
  (0, 470)      0.21983825902084567
  (0, 446)      0.16109431039727098
  (0, 440)      0.19496676604082638
  (0, 411)      0.25064440424955425
  (0, 377)      0.07589884536577327
  (0, 339)      0.21983825902084567
  (0, 321)      0.25064440424955425
  (0, 117)      0.25064440424955425
```

Pembuatan model Support Vector Machine untuk lexicon labelling

SVM

In [39]:
```
from sklearn.svm import SVC

lexicon_model = SVC(kernel='linear')
lexicon_model.fit(lexicon_train_X_tfidf_8020,lexicon_train_Y)
```

Out[39]:
```
SVC(kernel='linear')
```

In [40]:
```
from sklearn.metrics import accuracy_score

lexicon_predictions_SVM_8020 = lexicon_model.predict(lexicon_test_X_tfidf_8020)
lexicon_test_prediction_8020 = pd.DataFrame()
lexicon_test_prediction_8020['Sentiment'] = lexicon_test_X
lexicon_test_prediction_8020['Label'] = lexicon_predictions_SVM_8020
lexicon_SVM_accuracy_8020 = accuracy_score(lexicon_predictions_SVM_8020, lexicon_test_Y)*100
lexicon_SVM_accuracy_8020 = round(lexicon_SVM_accuracy_8020,1)
```

Hasil dari model Machine Learning untuk lexicon labelling

```
In [42]: lexicon_SVM_accuracy_8020
Out[42]: 61.9
```

Akurasi menggunakan model SVM sebesar 61.9%

```
In [43]: from sklearn.metrics import classification_report

         print ("\nHere is the classification report:")
         print (classification_report(lexicon_test_Y, lexicon_predictions_SVM_8020))
```

```
Here is the classification report:
              precision    recall  f1-score   support

    Negative       0.67      0.80      0.73        20
     Neutral       0.44      0.31      0.36        13
    Positive       0.67      0.67      0.67         9

    accuracy                           0.62        42
   macro avg       0.59      0.59      0.59        42
weighted avg       0.60      0.62      0.60        42
```

SVM Modelling pada Manual Label

Membuat model diawali dengan membagi antara data training dan data testing

Komposisi yang kami gunakan adalah 80% training dan 20% testing, kemudian akan dimasukkan ke dalam data frame baru, manual_df_train80 dan manual_df_train20

```
manual_train_X, manual_test_X, manual_train_Y, manual_test_Y = model_selection.train_test_split(data['tweet'], data['manual_label'], test_size = 0.2, random_state = 0)
```

```
manual_df_train80 = pd.DataFrame()
manual_df_train80['tweet'] = manual_train_X
manual_df_train80['manual_label'] = manual_train_Y

manual_df_test20 = pd.DataFrame()
manual_df_test20['tweet'] = manual_test_X
manual_df_test20['manual_label'] = manual_test_Y
```

TF-IDF manual labelling

Hasil TF-IDF dari manual labelling

```
# TF-IDF

from sklearn.feature_extraction.text import TfidfVectorizer

manual_tfidf_vect_8020 = TfidfVectorizer(max_features = 5000)
manual_tfidf_vect_8020.fit(data['tweet'])
manual_train_X_tfidf_8020 = manual_tfidf_vect_8020.transform(manual_df_train80['tweet'])
manual_test_X_tfidf_8020 = manual_tfidf_vect_8020.transform(manual_df_test20['tweet'])

```

```
manual_tfidf_vect_8020
```

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=1.0, max_features=5000,
                min_df=1, ngram_range=(1, 1), norm='l2', preprocessor=None,
                smooth_idf=True, stop_words=None, strip_accents=None,
                sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, use_idf=True, vocabulary=None)
```

Hasil dari TF-IDF manual training dan testing

```
print(manual_train_X_tfidf_8020)
```

```
  (0, 1047)     0.25796689169680226
  (0, 743)      0.22626075585226202
  (0, 711)      0.38910924001544345
  (0, 654)      0.25796689169680226
  (0, 471)      0.45252151170452404
  (0, 414)      0.2394199911849753
  (0, 377)      0.07811620323624119
  (0, 357)      0.2394199911849753
  (0, 338)      0.2965633657027805
  (0, 332)      0.25796689169680226
  (0, 132)      0.25796689169680226
  (0, 113)      0.22626075585226202
  (0, 21)       0.25796689169680226
  (1, 980)      0.3673969731176012
  (1, 862)      0.4272708447227949
```

```
print(manual_test_X_tfidf_8020)
```

```
  (0, 968)      0.18379737951971664
  (0, 804)      0.21983825902084567
  (0, 607)      0.6023531103123709
  (0, 548)      0.25064440424955425
  (0, 520)      0.21983825902084567
  (0, 477)      0.21983825902084567
  (0, 470)      0.21983825902084567
  (0, 446)      0.16109431039727098
  (0, 440)      0.19496676604082638
  (0, 411)      0.25064440424955425
  (0, 377)      0.07589884536577327
  (0, 339)      0.21983825902084567
  (0, 321)      0.25064440424955425
  (0, 117)      0.25064440424955425
  (0, 77)       0.20181781927028114
  (1, 918)      0.25767687200776235
  (1, 900)      0.23915082287838885
  (1, 698)      0.22600638183696656
```

Pembuatan model Support Vector Machine untuk manual labelling

```
from sklearn.svm import SVC

manual_model = SVC(kernel='linear')
manual_model.fit(manual_train_X_tfidf_8020,manual_train_Y)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

```
from sklearn.metrics import accuracy_score

manual_predictions_SVM_8020 = manual_model.predict(manual_test_X_tfidf_8020)
manual_test_prediction_8020 = pd.DataFrame()
manual_test_prediction_8020['Sentiment'] = manual_test_X
manual_test_prediction_8020['Label'] = manual_predictions_SVM_8020
manual_SVM_accuracy_8020 = accuracy_score(manual_predictions_SVM_8020, manual_test_Y)*100
manual_SVM_accuracy_8020 = round(manual_SVM_accuracy_8020,1)
```

Hasil dari model Machine Learning untuk manual labelling

```
1  manual_SVM_accuracy_8020
```

```
76.2
```

Hasil akurasi berbeda dari lexicon label, akurasinya naik menjadi 76.2%

```
1  from sklearn.metrics import classification_report
2
3  print ("\nHere is the classification report:")
4  print (classification_report(manual_test_Y, manual_predictions_SVM_8020))
```

```
Here is the classification report:
              precision    recall  f1-score   support

    Negative       0.72      0.96      0.82        24
     Neutral       1.00      0.33      0.50         6
    Positive       0.88      0.58      0.70        12

    accuracy                           0.76        42
   macro avg       0.86      0.62      0.67        42
weighted avg       0.80      0.76      0.74        42
```

Perbandingan Hasil Manual Label Dengan Lexicon Label

Manual

Here is the classification report:

	precision	recall	f1-score	support
Negative	0.72	0.96	0.82	24
Neutral	1.00	0.33	0.50	6
Positive	0.88	0.58	0.70	12
accuracy			0.76	42
macro avg	0.86	0.62	0.67	42
weighted avg	0.80	0.76	0.74	42

Lexicon

Here is the classification report:

	precision	recall	f1-score	support
Negative	0.67	0.80	0.73	20
Neutral	0.44	0.31	0.36	13
Positive	0.67	0.67	0.67	9
accuracy			0.62	42
macro avg	0.59	0.59	0.59	42
weighted avg	0.60	0.62	0.60	42

Machine Learning memiliki akurasi untuk memprediksi sentimen yang dibuat manual lebih baik dari pada memprediksi sentimen yang dibuat berdasarkan lexicon dengan selisih 14%. Cara manual menggunakan pemahaman manusia yang memiliki pemahaman kata yang banyak sehingga pelabelan lebih baik dibandingkan lexicon yang memiliki kamus kata yang terbatas

THANK YOU