BAB IV
HASIL & PEMBAHASAN

4.1 Hasil Pengumpulan Data

Data ulasan yang diambil secara acak di Tripadvisor.com, berhasil didapatkan dalam susuan paragraf. Data kemudian melalui tokenisasi menjadi kalimat-kalimat. Total kalimat atau total data yang digunakan berjumlah sebanyak 42970 data, yang kemudian dibagi menjadi data uji dan data latih.

4.2 *Preprocessing*

Tahap yang dilakukan selanjutnya yaitu *preprocessing*, adapun *preprocessing* yang digunakan beserta kode dan penjelasan dalam kasus ini yaitu :

a. *Casefolding*

Langkah pertama dalam melakukan pembersihan data yaitu dengan melakukan *casefolding* atau membuat semua huruf menjadi huruf kecil. Gambar 4.1 merupakan kode program untuk melakukan *casefolding*.

```
def casefolding(s):
    new_str = s.lower()
    return new_str
```

Gambar 4.1 Kode program *casefolding*

b. Menghapus karakter berulang

Setelah semua kata diubah menjadi kecil semua, langkah ini yaitu menghapus karakter yang berulang lebih dari 2 kali. Gambar 4.2 merupakan kode program untuk menghapus karakter berulang.

```
def remove_repeated_character(str):
    str = re.sub(r'(.)\1{2,}', r'\1', str)

    return str
```

Gambar 4.2 Kode program menghapus karakter berulang

c. Menghapus tanda baca

Langkah selanjutnya yaitu menghapus tanda baca yang berada pada kalimat ulasan. Kode program Gambar 4.3 akan menghapus seluruh karakter yang bukan huruf dan angka.

```
def remove_punctuation(text):
    """ Remove all punctuation in text. """
    return re.sub(r'[^\s\w]', ' ', text)
```

Gambar 4.3 Kode program menghapus tanda baca

d. Menghapus digit

Gambar 4.4 untuk menghapus digit yang terkandung dalam kalimat ulasan.

```
def remove_digit_number(text):
    """ Remove all digit number in text. """
    return re.sub(r'[^a-z ]*([.0-9])*\d', '', text)
```

Gambar 4.4 Kode program menghapus digit

e. Menghapus karakter tidak dipakai

Langkah selanjutnya yaitu menghapus karakter tidak dipakai. Gambar 4.5 menunjukkan kode program untuk menghapus karakter yang berjumlah kurang dari 3 karakter.

```
def remove_unused_character(text):
    """ Remove characters that are less than three character. """
    text_list = text.split(' ')
    text_list_temp = []

    for index in range(len(text_list)):
        if len(text_list[index]) > 3:
            text_list_temp.append(text_list[index])

    return ' '.join(text_list_temp)
```

Gambar 4.5 Kode program menghapus karakter tidak dipakai

f. Menghapus spasi berlebih

Setelah melalui langkah-langkah sebelumnya, terdapat kemungkinan suatu kalimat memiliki spasi lebih dari satu, oleh karena itu perlu dilakukan pembersihan spasi agar setiap kata memiliki jeda satu spasi saja. Gambar 4.6 di bawah ini menunjukkan kode program untuk menghapus spasi berlebih.

```
def remove_extra_space(text):
        """ Make extra space into one space. """
        text_list = text.split(' ')
        text_list_temp = []

        for word in text_list:
            if word.strip():
                text_list_temp.append(word.strip())

        return ' '.join(text_list_temp)

import re
```

Gambar 4.6 Kode program menghapus spasi berlebih

g. Formalisasi kata

Bagian ini digunakan untuk melakukan formalisasi kata dengan mengubah kata informal atau abreviasi yang ada pada kalimat ulasan dan mengubahnya menjadi kata yang formal. Pengubahan kata berdasarkan dari kamus kata yang sudah dimiliki dan disimpan pada file .csv. Gambar 4.7 menunjukkan kode program untuk melakukan formalisasi kata.

```
def formalize_slang_word(str):
    text_list = str.split(' ')
    slang_words_raw = pandas.read_csv('slang_word_list.csv', sep=',', header=None)
    slang_word_dict = {}

    for item in slang_words_raw.values:
        slang_word_dict[item[0]] = item[1]

        for index in range(len(text_list)):
            if text_list[index] in slang_word_dict.keys():
                text_list[index] = slang_word_dict[text_list[index]]

    return ' '.join(text_list)
```

Gambar 4.7 Kode program formalisasi kata

h. *Negation handling*

Langkah ini digunakan untuk menggabungkan kalimat yang mengandung kata negasi "tidak", "bukan", "jangan", dan "kurang" di dalamnya. Jika ditemui kata negasi ini, maka kata negasi yang diikuti kata setelahnya akan digabungkan menjadi satu. Gambar 4.8 dibawah ini merupakan kode program untuk *negation handling*.

```
def join_negation(text):
    """  Join negation word with delimiter. """
    text_list = text.split(' ')

    for index in range(len(text_list)):
        if (text_list[index] == 'tidak' or text_list[index] == 'kurang' or
text_list[index] == 'jangan'
            or text_list[index] == 'bukan'):
            if index < len(text_list) - 1:
                text_list[index] = text_list[index] + "_" + text_list[index
+ 1]
                text_list[index + 1] = ''
            else:
                text_list[index] = ''
    return ' '.join(' '.join(text_list).split())
```

Gambar 4.8 Kode program *negation handling*

i. Memisahkan Kata Konjungsi Berlawanan

Bagian ini digunakan untuk memisahkan memisahkan kalimat yang terdapat kata "tetapi", "tapi", "meskipun", "walaupun", "padahal", "namun" karena kalimat dapat mengandung lebih

dari 1 aspek atau 1 sentimen. Gambar 4.9 merupakan kode program untuk memisahkan kalimat dengan kata konjungsi berlawanan.

```
def Split_Con(list):
    rsl = []
    for val in list:
        for kal in val.split():
            if (kal == "tapi" or kal == "tetapi" or kal == "walaupun" or kal ==
"meskipun" or kal == "padahal"
                or kal == "namun"):
                 if kal == "tapi":
                     tmp = val.split("tapi")
                 elif kal == "tetapi":
                     tmp = val.split("tetapi")
                 elif kal == "walaupun":
                     tmp = val.split("walaupun")
                 elif kal == "meskipun":
                     tmp = val.split("meskipun")
                 elif kal == "padahal":
                     tmp = val.split("padahal")
                 elif kal == "namun":
                     tmp = val.split("namun")
                 break
            else :
                 tmp = [val]
        for vall in tmp:
            vall = ' '. join(vall.split())
            rsl.append(vall)
    return revbar
```

Gambar 4.9 Kode program untuk memisahkan kalimat yang mengandung kata konjungsi berlawanan

j. Menghapus *Stopwords*

Gambar 4.10 merupakan kode *stopwords* untuk menghilangkan kata yang tidak signifikan dalam membantu mendapatkan sentimen.

```
stopwords =
["pernah", "dan", "untuk", "yang", "dengan", "sangat", "banyak", "dari", "kita",
"saya", "juga", "bisa", "karena", "kalau", "tapi", "akan", "sudah", "kami",
"adalah", "anda", "lagi", "buat", "salah", "sampai", "dapat", "dalam", "lebih",
"pada", "sekali", "atau", "masih", "jika", "apa", "beberapa", "menjadi", "tetap",
"saja", "terdapat", "boleh", "begitu", "hanya", "paling", "sehingga", "jadi",
"sambil", "harus", "memang", "setiap", "selalu", "berada", "kamu", "sebagai",
"bagi", "sana", "lain", "setelah", "semua", "seperti", "dulu", "dahulu", "ketika",
"terlalu", "mungkin", "namun", "hingga", "ada", "bila", "agak", "tersebut",
"sebuah", "selain", "sungguh", "bahkan", "tetapi", "apalagi", "belum", "telah",
"terus", "meskipun", "lalu", "sama", "agar", "pula", "secara", "selama", "tiap",
"bagian", "meski", "yaitu", "serta", "seorang", "orang", "walaupun", "tertentu",
"maka", "seolah", "cuma", "sang", "alhamdulillah", "seakan", "bakal", "sekaligus",
"kebanyakan", "sebelum", "senantiasa", "adanya", "saat","itulah", "tadi",
"terkait", "begitulah", "kira", "ke", "di", "nya", "antara", "ialah", "ya",
"ini", "itu"]
def stopword(list):
    rsl = []
    for val in list:
        querywords = val.split()
        resultwords  = [word for word in querywords if word.lower() not in stopwords]
        result = ' '.join(resultwords)
        rsl.append(result)
    return rsl
```

Gambar 4.10 Kode program *stopwords*

k. *Stemming*

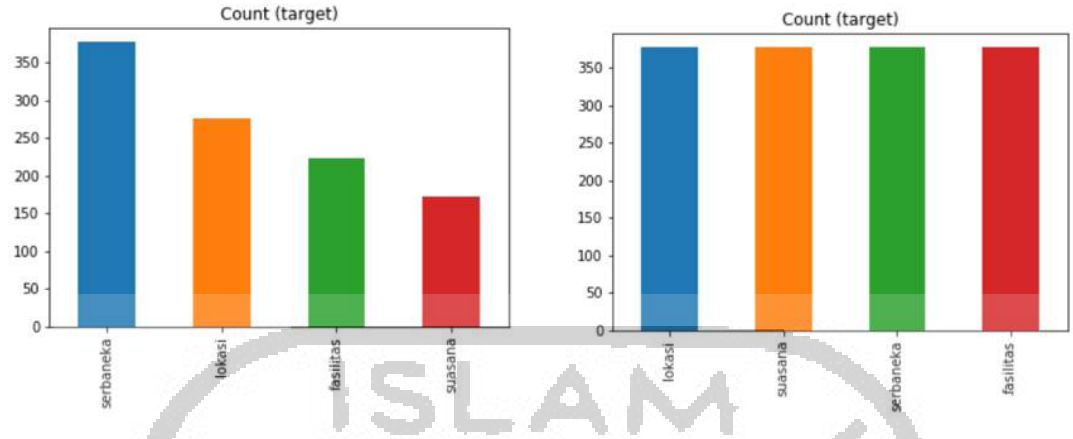
Gambar 4.11 merupakan kode program untuk melakukan *stemming*. Tahap ini digunakan untuk membantu proses mendapatkan aspek kategori saja.

```
def Stemming(list):
    resl = []
    for val in list:
        katadasar = stemmer.stem(val)
        resl.append(katadasar)
    return resl
```

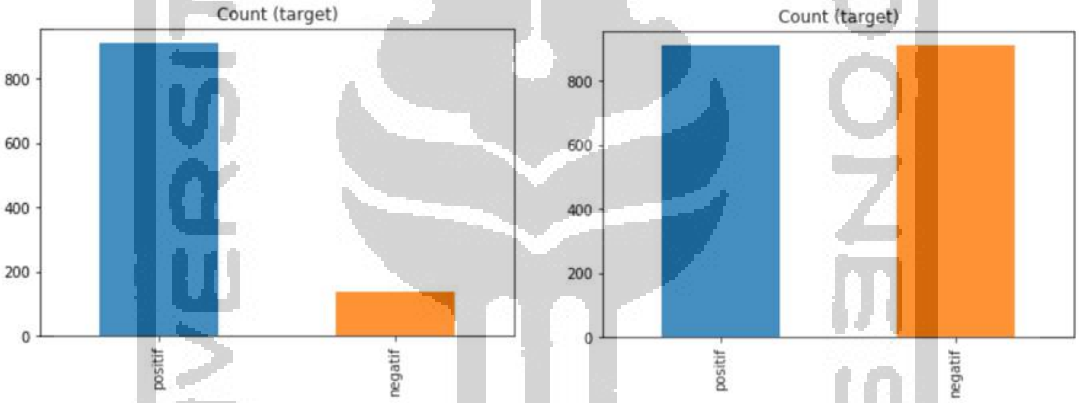
Gambar 4.11 Kode program *stemming*

4.3 Pelabelan Data

Karena data ulasan yang dimiliki tidak seimbang, oleh karena itu dilakukan teknik *oversampling* pada 1000 data terlabeli saat melakukan *pseudo labeling* untuk mendapatkan sentimen dan aspek kategori ulasan yang lebih akurat. Gambar 4.12 dan Gambar 4.13 merupakan contoh hasil dari *oversampling* yang dilakukan pada data aspek kategori dan sentimen.



Gambar 4.12 Hasil *oversampling* pada label aspek kategori



Gambar 4.13 Hasil *oversampling* pada label sentimen

Tabel 4.1 menunjukkan masing-masing akurasi model yang didapatkan dari melatih 1000 kalimat ulasan setelah dilakukan teknik *oversampling*. Total data aspek setelah melalui *oversampling* berjumlah 1508 dengan masing-masing kelas lokasi, fasilitas, serbaneka, dan suasana sebanyak 377. Untuk membagi data, digunakan 33% secara acak sebagai data uji dan sisanya sebagai data latih. Data uji berjumlah 1010, dan data latih bejumlah 498. Kemudian ketika pelatihan data, pada data latih dibagi kembali sebanyak 33% menjadi data uji dan data latih sebagai validasi model, sehingga nilai data latih menjadi 676, dan data uji menjadi 334.

Sedangkan untuk data sentimen setelah melalui *oversampling* berjumlah 1820 dengan masing-masing kelas positif dan negatif sebanyak 910. Untuk membagi data, digunakan 33% secara acak sebagai data uji dan sisanya sebagai data latih. Data uji berjumlah 1219, dan data latih bejumlah 601. Kemudian ketika pelatihan data, pada data latih dibagi kembali sebanyak

33% menjadi data uji dan data latih sebagai validasi model, sehingga nilai data latih menjadi 816, dan data uji menjadi 403.

Tabel 4.1 akurasi model pada label sentimen dan akurasi

Label	**Akurasi(%)**
Sentimen	93
Aspek	80

Setelah dilakukan *pseudo labeling* pada 42970 data yang belum terlabeli. Tabel 4.2 merupakan jumlah label yang didapat :

Tabel 4.2 Tabel jumlah data *pseudo labeling*

Sentimen	**Aspek**	**Jumlah Data**
Positif	Suasana	5756
	Lokasi	9371
	Fasilitas	3860
	Serbaneka	17940
Negatif	Suasana	722
	Lokasi	1294
	Fasilitas	643
	Serbaneka	3383

Dalam pelabelan data aspek kategori digunakan 4 label yaitu fasilitas, lokasi, suasana, dan serbaneka. Dalam tahap selanjutnya untuk pelatihan data dalam mendapatkan aspek kategori, hanya digunakan label fasilitas, lokasi, dan suasana saja.

4.4 Ekstraksi Fitur

POS Tag dilakukan untuk mendapatkan kelas kata seperti kata sifat, kata benda, dll. Dilakukan juga pengambilan kata negasi yang sebelumnya melalui *negation handling*. Gambar 4.14 merupakan kode program untuk mendapatkan POS Tag dari suatu ulasan.

```
def negasi(str):
    tmp = ""
    for v in str:
        if v == "_":
            tmp = str
            return tmp
        else :
            tmp = ""
    return tmp

kal_sent = d_sent["Review_Stopwords_Joined"]
spl_sent = [val.split() for val in kal_sent]
hasil_sent = [ct.tag_sents([val]) for val in spl_sent]
hasil_sent = [item for sublist in hasil_sent for item in sublist]
sentimen = []
for val in hasil_sent:
    tmp = ""
    for vall in val:
        if(vall[1] == "NEG" or vall[1] == "JJ" or vall[1] == "VB"):
            tmp = tmp + vall[0] + " "
        tmp = tmp + negasi(vall[0]) + " "
    tmp = tmp.split()
    tmp = ' '.join(tmp)
    sentimen.append(tmp)
```

Gambar 4.14 Kode program POS Tag

4.5 Sentimen Analisis Berbasis Fitur

Dalam melakukan sentimen analisis berbasis fitur, terdapat beberapa tahapan yang perlu dilakukan. Berikut merupakan tahapan yang diperlukan untuk mendapatkan sentimen dan aspek kategori :

a. *Tokenizer*

Tokenizer berperan dalam mengubah teks korpus menjadi vektor dengan mengubah setiap teks menjadi susunan urutan integer, dimana setiap integer akan menjadi indeks dari kamus token. `fit_on_texts` digunakan untuk memperbarui kosa kata internal sesuai dengan daftar kata pada teks. Metode ini menciptakan indeks kosa kata bergantung pada frekuensi kemunculan kata. `texts_to_sequences` digunakan untuk mengubah setiap teks menjadi urutan integer. Dengan metode ini setiap kata pada ulasan akan diubah menjadi angka integer yang sesuai dari kamus indeks kata sebelumnya. Sedangkan `pad_sequences` digunakan untuk memastikan bahwa seluruh urutan dalam sebuah array memiliki panjang yang sama. Standarnya padding akan menambahkan angka 0 di setiap awal sebuah *sequence* hingga

panjangnya sama dengan *sequence* terpanjang. Gambar 4.15 merupakan kode program untuk tahap ini.

```
tk = Tokenizer()
maxlen = max([len(i.split()) for i in text])

tk.fit_on_texts(text)
x = tk.texts_to_sequences(x)
x = sequence.pad_sequences(x, maxlen)
vocab = max([len(tk.word_index)]) + 1
```

Gambar 4.15 Kode program tokenizer

b. *Splitting Data*

Dalam menguji sebuah model diperlukan pembagian dataset yaitu data latih dan data uji. Untuk melakukan pembagian dataset digunakan the SciKit library dengan nama kelas 'train_test_split.'. Dengan menggunakan 'train_test_split' dapat membagi data latih dan data uji secara acak menjadi dalam berbagai proporsi. Pada kasus ini digunakan data uji sebanyak 0.33 bagian seperti yang tertera pada kode program dengan Gambar 4.16.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y_aspect,
test_size=0.33, random_state=42)
```

Gambar 4.16 Kode program splitting data

c. Membangun Model

Tipe model yang digunakan pada kasus ini yaitu *Sequential*. Dengan *Sequential*, membangun model dapat dibuat dari layer ke layer secara berurutan. Setiap layernya memiliki bobot yang bersesuaian dengan layer berikutnya. Penambahan layer pada model dapat ditambahkan dengan mudah menggunakan fungsi 'add()'.

'*Activation'* adalah fungsi aktivasi untuk lapisan model. Tujuan dari aktivasi yaitu untuk mengatasi kasus rumit dan non linier yang kompleks. Fungsi aktivasi yang digunakan yaitu ReLU atau Rectified Linear Activation, Sigmoid dan Softmax. Pada model ini digunakan ReLU karena merupakan satu dari beberapa fungsi aktivasi yang terbaik (Wallace, 2014).

Layer terakhir yang digunakan yaitu layer Dense atau *Fully Connected layer* sebagai lapisan keluaran. Dalam Dense layer, semua *node* di lapisan sebelumnya terhubung ke *node* di lapisan saat ini. Pada layer ini, untuk model sentimen memanfaatkan fungsi aktivasi sigmoid karena fungsi ini yang paling sesuai untuk kategori 2 kelas, sedangkan untuk model aspek menggunakan fungsi aktivasi *softmax* karena memiliki kategori yang cukup banyak.

Untuk *compile* model dibutuhkan 2 parameter, yaitu *opimizer* dan *loss*. *Optimizer* bertugas untuk mengontrol *learning rate*. Pada kasus ini, digunakan *optimizer* '*adam'* yang dianggap *optimizer* bagus dan sering digunakan. Untuk fungsi *loss*, digunakan *binary_crossentropy* pada

model sentimen karena paling sesuai untuk kategori dengan 2 kelas, sedangkan *categorical_crossentropy* digunakan pada model aspek.

Gambar 4.17 dan Gambar 4.18, merupakan kode program untuk membangun model CNN untuk mendapatkan sentimen dan aspek.

```
def model_cnn_sent(dropout):
    model_sent = Sequential()
    model_sent.add(Embedding(vocab_sent,
                        embedding_dims,
                        input_length=maxlen_sent))
    model_sent.add(Dropout(dropout))
    model_sent.add(Conv1D(filters,
                    kernel_size,
                    padding='valid',
                    activation='relu',
                    strides=1))
    model_sent.add(GlobalMaxPooling1D())
    model_sent.add(Dense(hidden_dims))
    model_sent.add(Dropout(dropout))
    model_sent.add(Activation('relu'))
    model_sent.add(Dense(2))
    model_sent.add(Activation('sigmoid'))
    model_sent.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    return model_sent
```

Gambar 4.17 Kode program membangun model sentimen CNN

```
def model_cnn_asp(dropout):
    model_aspect = Sequential()
    model_aspect.add(Embedding(vocab_aspect,
                        embedding_dims,
                        input_length=maxlen_aspect))
    model_aspect.add(Dropout(dropout))
    model_aspect.add(Conv1D(filters,
                     kernel_size,
                     padding='valid',
                     activation='relu',
                     strides=1))

    model_aspect.add(GlobalMaxPooling1D())
    model_aspect.add(Dense(hidden_dims))
    model_aspect.add(Dropout(dropout))
    model_aspect.add(Activation('relu'))

    model_aspect.add(Dense(3))
    model_aspect.add(Activation('softmax'))

    model_aspect.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    return model_aspect
```

Gambar 4.18 Kode program membangun model aspek CNN

Gambar 4.19 dan Gambar 4.20 merupakan model CNN + LSTM untuk mendapatkan aspek dan sentimen. Pada model ini ditambahkan layer LSTM dengan unit lstm 64 sebelum *output* di layer FC.

```
def model_cnn_lstm_sent(dropout, lstm_unit):
    model = Sequential()
    model.add(Embedding(vocab,
                        embedding_dims,
                        input_length=maxlen))
    model.add(Dropout(dropout))
    model.add(Conv1D(filters,
                     kernel_size,
                     padding='valid',
                     activation='relu',
                     strides=1))
    model.add(MaxPooling1D())
    model.add(Dense(hidden_dims)) # ini
    model.add(Dropout(dropout))
    model.add(Activation('relu'))
    model.add(LSTM(lstm_unit))
    model.add(Dense(2))
    model.add(Activation(sigmoid))

    model.compile(loss=binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    return model
```

Gambar 4.19 Kode program membangun model sentimen CNN + LSTM

```
def model_cnn_lstm_aspek(dropout, lstm_unit):
    model = Sequential()
    model.add(Embedding(vocab,
                        embedding_dims,
                        input_length=maxlen))
    model.add(Dropout(dropout))
    model.add(Conv1D(filters,
                     kernel_size,
                     padding='valid',
                     activation='relu',
                     strides=1))
    model.add(MaxPooling1D())
    model.add(Dense(hidden_dims))
    model.add(Dropout(dropout))
    model.add(Activation('relu'))
    model.add(LSTM(lstm_unit))
    model.add(Dense(3))
    model.add(Activation('softmax'))

    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    return model
```

Gambar 4.20 Kode program membangun model aspek CNN + LSTM

Gambar 4.21 dan Gambar 4.22 merupakan model CNN + GRU untuk mendapatkan sentimen dan aspek. Model ini ditambahkan layer GRU sebelum *output* pada *FC layer*.

```
def model_cnn_gru_sent(dropout, gru_unit):
    model = Sequential()
    model.add(Embedding(vocab,
                        embedding_dims,
                        input_length=maxlen))
    model.add(Dropout(dropout))
    model.add(Conv1D(filters,
                     kernel_size,
                     padding='valid',
                     activation='relu',
                     strides=1))
    model.add(MaxPooling1D())
    model.add(Dense(hidden_dims))
    model.add(Dropout(dropout))
    model.add(Activation('relu'))

    model.add(GRU(gru_unit))

    model.add(Dense(2))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    return model
```

Gambar 4.21 Kode program membangun model sentimen CNN + GRU

```
def model_cnn_gru_aspek(dropout, gru_unit):
    model = Sequential()
    model.add(Embedding(vocab,
                        embedding_dims,
                        input_length=maxlen))
    model.add(Dropout(dropout))
    model.add(Conv1D(filters,
                     kernel_size,
                     padding='valid',
                     activation='relu',
                     strides=1))
    model.add(MaxPooling1D())
    model.add(Dense(hidden_dims))
    model.add(Dropout(dropout))
    model.add(Activation('relu'))

    model.add(GRU(gru_unit))

    model.add(Dense(3))
    model.add(Activation('softmax'))

    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    return model
```

Gambar 4.22 Kode program membangun model sentimen CNN + GRU

d. *Training Model*

Pelatihan model deep learning menggunakan fungsi 'fit()' yang diikuti 5 parameter: *training data* (train_X), *target data* (train_y), *validation split*, dan *callbacks*. Penggunaan *validation split* akan membagi data secara acak menjadi data latih dan data uji. Pada kasus ini nilai *validation split* yaiitu 0.33, sehingga 33% data dari data training akan digunakan untuk menguji performa model. Ketika pelatihan data, dengan adanya pembagian data akan muncul *validation loss*. *Validation loss* yaitu *error* yang muncul setelah menjalankan *data validation* melalui model.

Epoch merupakan berapa kali sebuah model akan berputar pada data. Hingga batas angka tertentu, bergantung pada kasus, *epoch* dengan nilai tinggi akan memberikan performa yang semakin bagus. Jika melebihi batas tersebut, performa model tidak akan meningkat. Pada kasus ini digunakan '*early stopping'* yang akan menghentikan model dari pelatihan sebelum angka *epoch* yang ditetapkan tercapai, jika model tidak mengalami peningkatan performa. Fungsi '*early stopping'* dibuat dengan memonitor sebanyak 10 *epochs*. Jika selama 10 *epoch*s berturut-turut model tidak mengalami peningkatan, pelatihan akan dihentikan. Gambar 4.23 merupakan kode program untuk melatih model.

```
history = model.fit(x_train, y_train,
          batch_size=64,
          epochs=50,
          validation_split = 0.33,
          callbacks=[EarlyStopping(monitor='val_loss', patience=10,
min_delta=0.0001)])
```

Gambar 4.23 Kode progam melatih model

e. Prediksi hasil

Untuk membuat prediksi pada data baru, digunakan fungsi 'predict_classes()' dengan memasukkan data baru pada fungsi tersebut. Gambar 4.24 dan Gambar 4.25 secara berututan merupakan prediksi untuk mendapatkan sentimen dan aspek.

```
def ABSA_sent(list):
    rsl = []
    spl_sent = [val.split() for val in list]
    hasil_sent = [ct.tag_sents([val]) for val in spl_sent]
    hasil_sent = [item for sublist in hasil_sent for item in sublist]
    for val, ko in zip(hasil_sent, list):
        tmp = ""
        for vall in val:
            if(vall[1] == "NEG" or vall[1] == "JJ" or vall[1] == "VB"):
                tmp = tmp + vall[0] + " "
            if('_' in vall[0]):
                tmp = tmp + vall[0] + " "
        tmp = tmp.split()
        tmp = ' '.join(tmp)
        k = tk_sent.texts_to_sequences([tmp]) #represent a completely new
set of word sequence
        k = sequence.pad_sequences(k, maxlen=maxlen_sent)
        hsl                                                              =
labelencoder_Y_sent.inverse_transform([model_sent.predict_classes(k)])
        rsl.append(hsl)
    return rsl
```

Gambar 4.24 Prediksi sentimen

```
def ABSA_aspect(list):
    rsl = []
    spl_aspect = [val.split() for val in stem(list)]
    hasil_aspect = [ct.tag_sents([val]) for val in spl_aspect]
    hasil_aspect = [item for sublist in hasil_aspect for item in sublist]
    for val in hasil_aspect:
        tmp = ""
        for vall in val:
            if(vall[1] == "NNP" or vall[1] == "NN" or vall[1] == "VB"):
                tmp = tmp + vall[0] + " "
        tmp = tmp.split()
        tmp = ' '.join(tmp)
        k = tk_aspect.texts_to_sequences([tmp])
        k = sequence.pad_sequences(k, maxlen=maxlen_aspect)
        hsl                                                              =
labelencoder_Y_aspect.inverse_transform([model_aspect.predict_classes(k)])
        rsl.append(hsl)
    return rsl
```

Gambar 4.25 Prediksi aspek kategori

4.6 Evaluasi

Pada tahap evaluasi, hasil evaluasi didapatkan dari data uji dan data latih yang sebelumnya melalui tahap *pseudo-labeling*. Data yang digunakan berjumlah 42970 baik aspek maupun sentimen. Namun karena data yang tidak seimbang, perlu dilakukan *oversampling* sehingga tidak ada informasi dari data latih asli yang terbuang, karena dengan teknik ini akan menyimpan seluruh data minoritas maupun mayoritas (A. Y. Liu, 2004). Setelah dilakukan *oversampling* secara acak, total data sentimen berjumlah 73854 dengan masing-masing kelas negatif dan positif sebanyak 36927. Karena melalui ekstraksi fitur dan untuk menghilangkan data kosong/terbuang, data kemudian berjumlah sebanyak 67815. Untuk membagi data, digunakan 33% secara acak sebagai data uji dan sisanya sebagai data latih. Data uji berjumlah 22379, dan data latih bejumlah 45436. Kemudian ketika pelatihan data, pada data latih dibagi kembali sebanyak 33% menjadi data uji dan data latih sebagai validasi model, sehingga nilai data latih menjadi 30442, dan data uji menjadi 14994.

Total data aspek setelah melalui *oversampling* berjumlah 31995 dengan masing-masing kelas lokasi, fasilitas dan suasana sebanyak 10665. Karena melalui ekstraksi fitur dan untuk menghilangkan data kosong/terbuang, data kemudian berjumlah sebanyak 31556. Untuk membagi data, digunakan 33% secara acak sebagai data uji dan sisanya sebagai data latih. Data uji berjumlah 10414, dan data latih bejumlah 21442. Kemudian ketika pelatihan data, pada data latih dibagi kembali sebanyak 33% menjadi data uji dan data latih sebagai validasi model, sehingga nilai data latih menjadi 14165, dan data uji menjadi 6977.

Pada pembahasan evaluasi ini di bawah ini, digunakan data evaluasi aspek sebanyak 10414, dan data evaluasi sentimen sebanyak 22379. Berikut pembahasan evaluasi dari setiap skenario :

a. Analisis Skenario 1

Tabel 4.3 merupakan tabel skenario 1 yang telah dibuat dengan tiga model yaitu, M1, M2, dan M3.

Tabel 4.3 Skenario 1

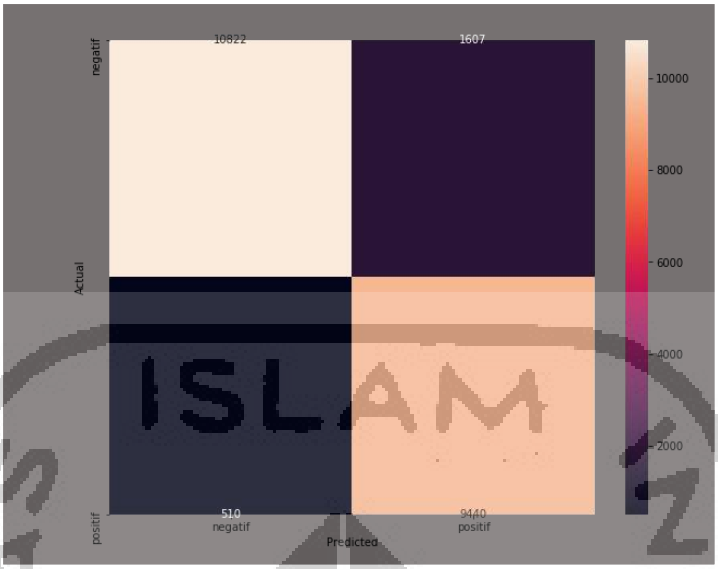
Skenario	Model	Metode	Kernel	Filter	Hidden_Dims	Dropout
1	M1	CNN	5	100	32	0.25
	M2		9	200	128	0.25
	M3		9	200	128	0.5

1. Sentimen

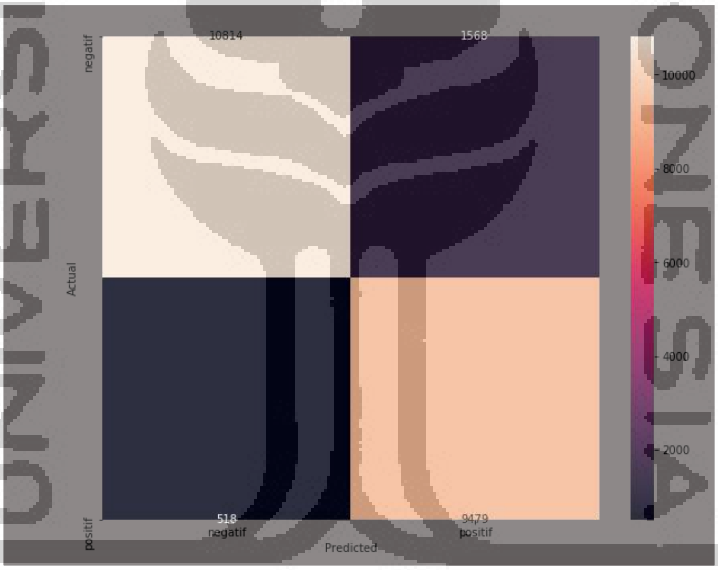
Pada sentimen Skenario 1 dengan metode CNN, dari ketiga model M1, M2, dan M3. Model M2 menggunakan *kernel, filter* dan *hidden_dims* berturut turut 9, 200 dan 128 dengan *dropout* sebesar 0.25. Dibandingkan dengan M3 yang menggunakan *kernel, filter* dan *hidden_dims* yang sama, dan *dropout* sebesar 0.5, M2 memiliki nilai akurasi yang paling tinggi mencapai 90.67%, presisi 90.61%, dan *recall* 91.07% dengan *epochs* sebanyak 17. Tabel 4.4 merupakan perbandingan *accuracy, precision,* dan *recall* sentimen pada skenario 1. Gambar 4.26, Gambar 4.27, dan Gambar 4.28 merupakan *confusion matrix* sentimen dari M1, M2 dan M3.

Tabel 4.4 Analisis Sentimen Skenario 1

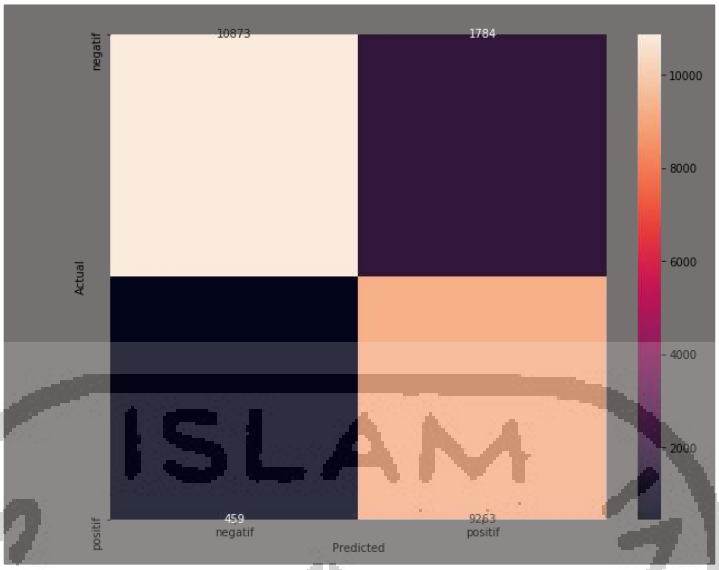
Skenario	Model	*Epochs*	Accuracy	Precision	Recall
1	M1	15	0.9054	0.9047	0.9097
	M2	**17**	**0.9067**	**0.9061**	**0.9107**
	M3	23	0.8997	0.8990	0.9059



Gambar 4.26 *Confusion Matrix* Sentimen M1



Gambar 4.27 *Confusion Matrix* Sentimen M2



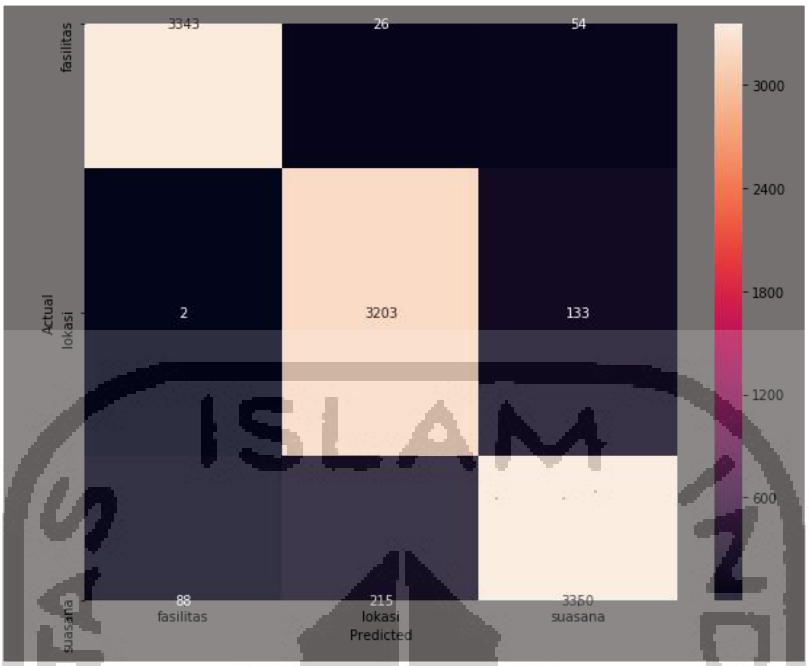
Gambar 4.28 *Confusion Matrix* Sentimen M3

2. Aspek

Pada aspek Skenario 1 dengan metode CNN, dari ketiga model M1, M2, dan M3. Model M2 menggunakan *kernel, filter* dan hidden_dims berturut turut 9, 200 dan 128 dengan dropout sebesar 0.25. Dibandingkan dengan M3 yang menggunakan *kernel, filter* dan *hidden_dims* yang sama, dan *dropout* sebesar 0.5, M2 memiliki nilai akurasi yang paling tinggi mencapai 95.21%, presisi 95.24%, dan *recall* 95.21% dengan *epochs* sebanyak 12. Tabel 4.5 merupakan perbandingan *accuracy*, *precision*, dan *recall* aspek pada skenario 1. Gambar 4.29, Gambar 4.30, dan Gambar 4.31 merupakan *confusion matrix* aspek dari M1, M2, dan M3.

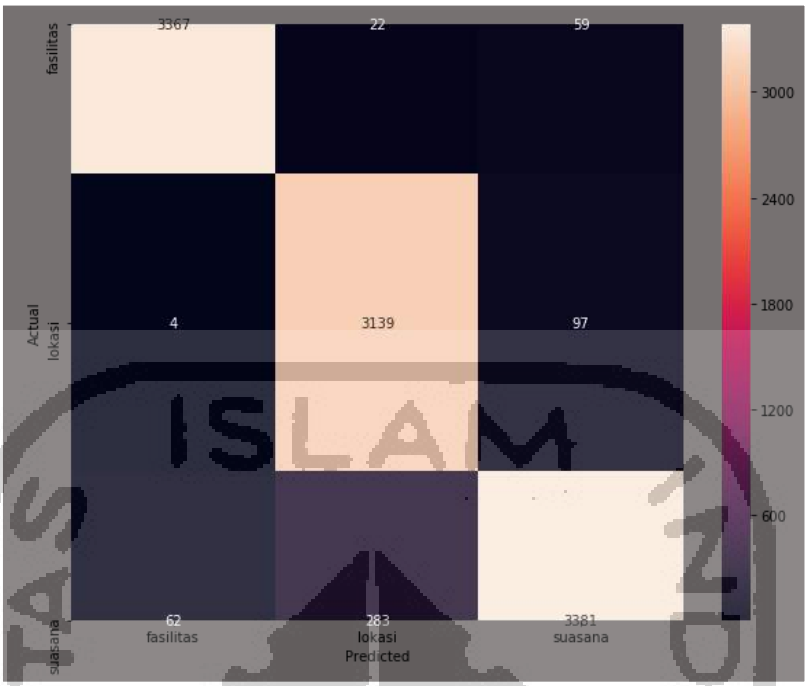
Tabel 4.5 Analisis Aspek Skenario 1

Skenario	**Model**	***Epoch*s**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**
1	M1	13	0.9502	0.9503	0.9510
	M2	**12**	**0.9521**	**0.9524**	**0.9521**
	M3	14	0.9493	0.9493	0.9509



Gambar 4.29 *Confusion Matrix* Aspek M1

Gambar 4.30 *Confusion Matrix* Aspek M2



Gambar 4.31 *Confusion Matrix* Aspek M3

b. Analisis Skenario 2

Tabel 4.6 merupakan tabel skenario 2 yang telah dibuat dengan dua model yaitu, M4, dan M5.

Tabel 4.6 Skenario 2

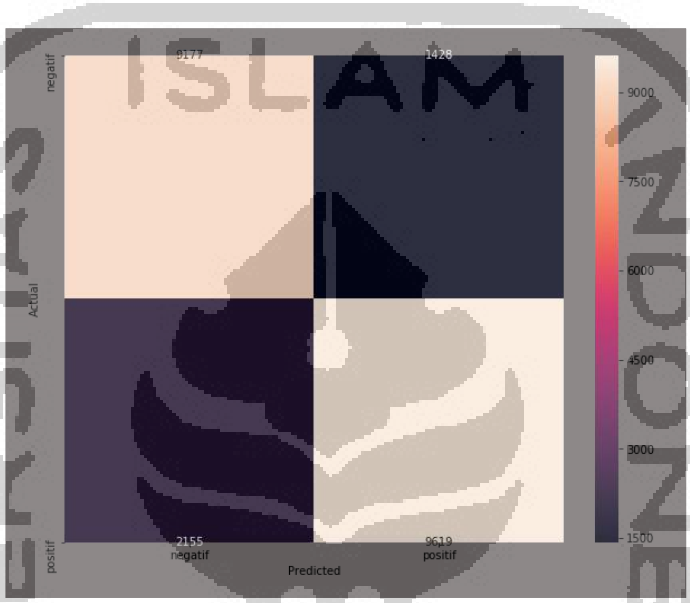
Skenario	Model	Metode	Kernel	Filter	Hidden_Dims	Dropout	Unit
2	M4	CNN + LSTM	9	200	128	0.25	64
	M5		9	200	128	0.5	64

1. Sentimen

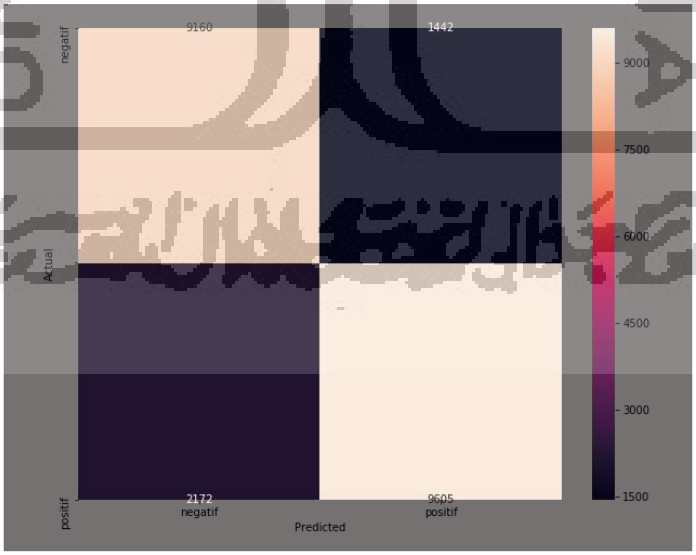
Pada aspek Skenario 2 dengan metode CNN + LSTM, dari kedua model M4, dan M5. Model M4 menggunakan kernel, filter dan hidden_dims berturut turut 9, 200, 128 dan unit LSTM sebesar 64 unit dengan dropout sebesar 0.25. Dibandingkan dengan M5 yang menggunakan kernel, filter dan hidden_dims yang sama, dan dropout sebesar 0.5, M4 memiliki nilai akurasi yang tertinggi yaitu 83.98%, presisi 84.02%, dan *recall* 84.11% dengan *epoch*s sebanyak 13. Tabel 4.7 merupakan perbandingan accuracy, precision, dan recall sentimen pada skenario 2. Gambar 4.32 dan Gambar 4.33 merupakan *confusion matrix* sentimen dari M4 dan M5.

Tabel 4.7 Analisis Sentimen Skenario 2

Skenario	Model	*Epochs*	Accuracy (%)	Precision (%)	Recall (%)
2	**M4**	**13**	**0.8398**	**0.8402**	**0.8411**
	M5	20	0.8385	0.8388	0.8397



Gambar 4.32 *Confusion Matrix* Sentimen M4



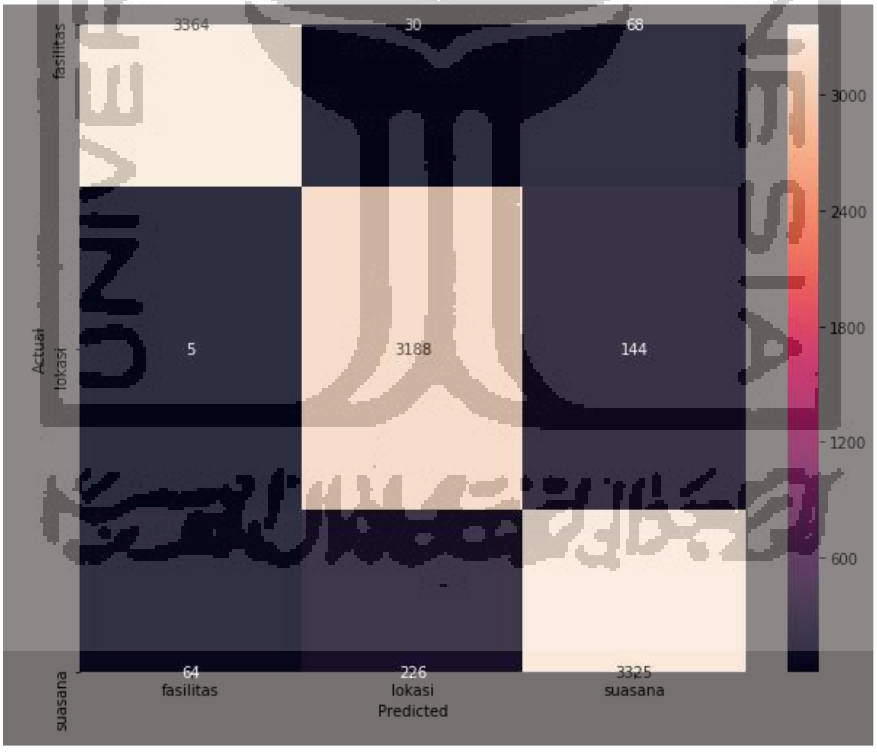
Gambar 4.33 *Confusion Matrix* Sentimen M5

2. Aspek

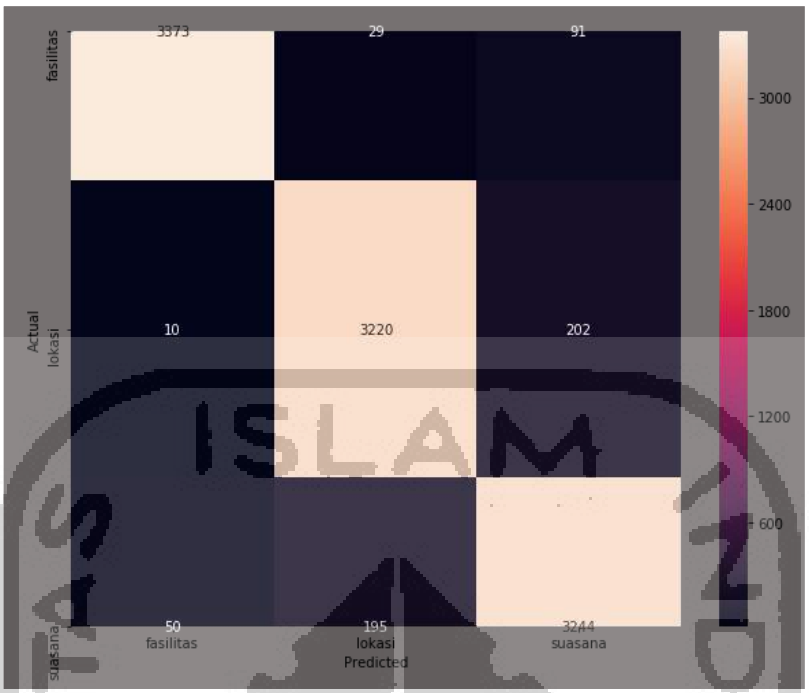
Pada aspek Skenario 2 dengan metode CNN + LSTM, dari kedua model M4, dan M5. Model M4 menggunakan *kernel*, *filter* dan *hidden_dims* berturut turut 9, 200, 128 dan unit LSTM sebesar 64 unit dengan *dropout* sebesar 0.25. Dibandingkan dengan M5 yang menggunakan *kernel*, *filter* dan *hidden_dims* yang sama, dan *dropout* sebesar 0.5, M4 memiliki nilai akurasi yang tertinggi yaitu 94.84%, presisi 94.85%, dan *recall* 94.89% dengan *epochs* sebanyak 11. Tabel 4.8 merupakan perbandingan *accuracy*, *precision*, dan recall aspek pada skenario 2. Gambar 4.34 dan Gambar 4.35 merupakan *confusion matrix* aspek dari M4 dan M5.

Tabel 4.8 Analisis Aspek Skenario 2

Skenario	Model	*Epochs*	Accuracy (%)	Precision (%)	Recall (%)
2	**M4**	**11**	**0.9484**	**0.9485**	**0.9489**
	M5	13	0.9445	0.9448	0.9445



Gambar 4.34 *Confusion Matrix* Aspek M4



Gambar 4.35 *Confusion Matrix* Aspek M5

c. Analisis Skenario 3

Tabel 4.9 merupakan tabel skenario 3 yang telah dibuat dengan dua model yaitu, M6, dan M7.

Tabel 4.9 Skenario 3

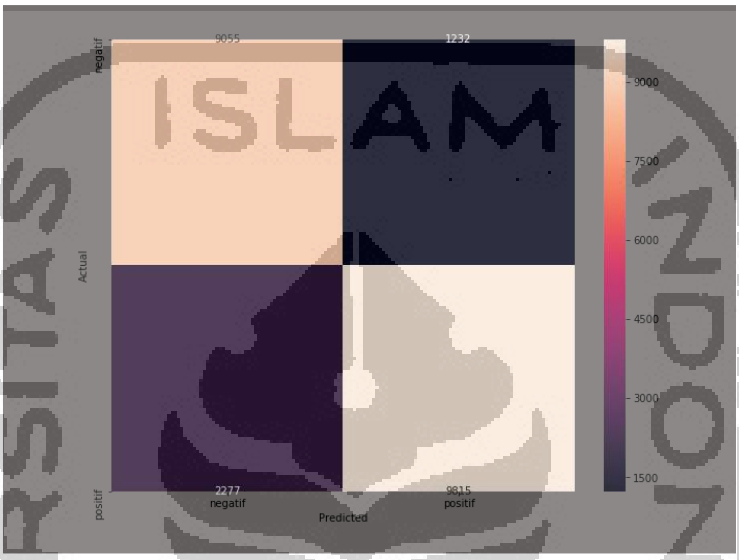
Skenario	**Model**	**Metode**	**Kernel**	**Filter**	***Hidden_dims***	**Dropout**	**Unit**
3	M6	CNN + GRU	9	200	128	0.25	64
	M7		9	200	128	0.5	64

1. Sentimen

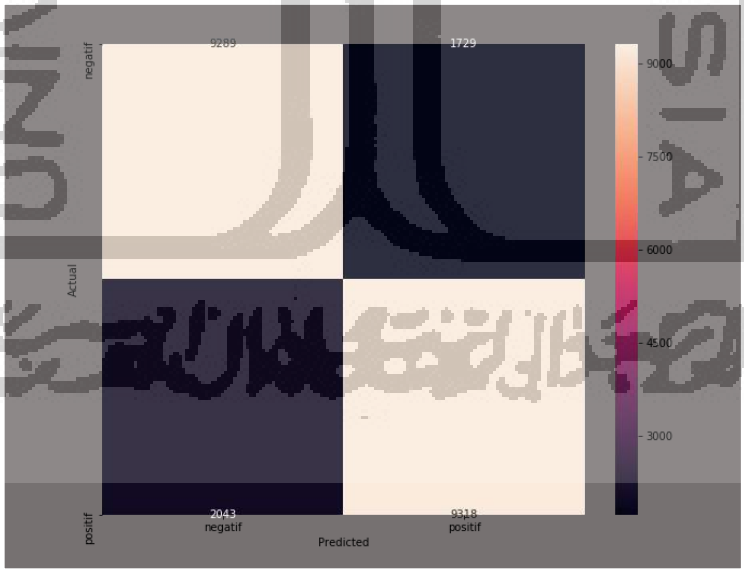
Pada aspek Skenario 3 dengan metode CNN + GRU, dari kedua model M6, dan M7. Model M6 menggunakan kernel, filter dan hidden_dims berturut turut 9, 200, 128 dan unit GRU sebesar 64 unit dengan dropout sebesar 0.25. Dibandingkan dengan M7 yang menggunakan kernel, filter dan hidden_dims yang sama, dan dropout sebesar 0.5, M6 memiliki nilai akurasi yang tertinggi yaitu 84.32%, presisi 84.37%, dan *recall* 84.59% dengan *epoch*s sebanyak 16. Tabel 4.10 merupakan perbandingan accuracy, precision, dan recall sentimen pada skenario 3. Gambar 4.36 dan Gambar 4.37 merupakan *confusion matrix* sentimen dari M6 dan M7.

Tabel 4.10 Analisis Sentimen Skenario 3

Skenario	Model	*Epochs*	Accuracy (%)	Precision (%)	Recall (%)
3	**M6**	**16**	**0.8432**	**0.8437**	**0.8459**
	M7	19	0.8314	0.8316	0.8316



Gambar 4.36 *Confusion Matrix* Sentimen M6



Gambar 4.37 *Confusion Matrix* Sentimen M7

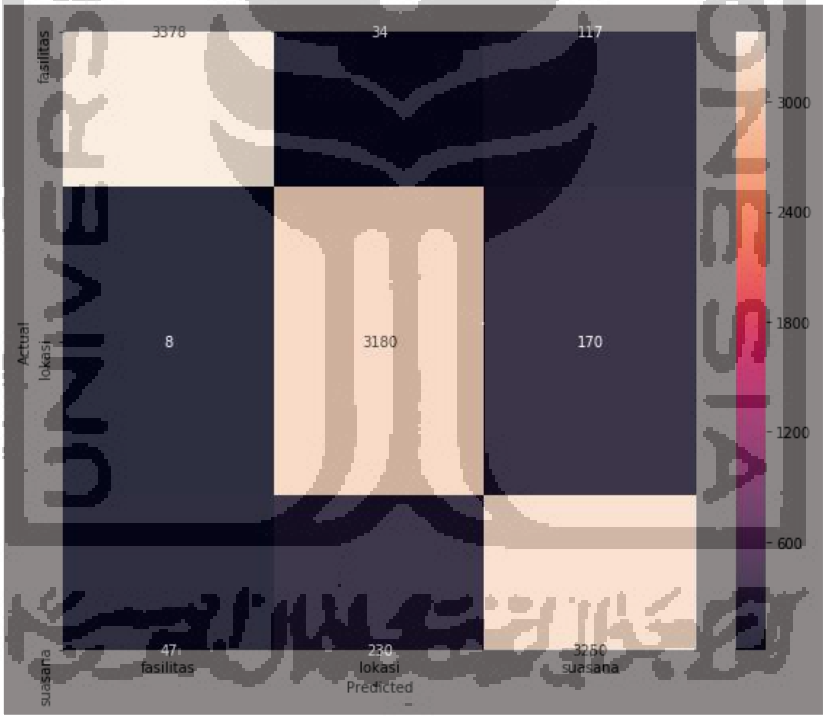
2. Aspek

Pada aspek Skenario 3 dengan metode CNN + GRU, dari kedua model M6, dan M7. Model M6 menggunakan *kernel*, *filter* dan *hidden_dims* berturut turut 9, 200, 128 dan unit

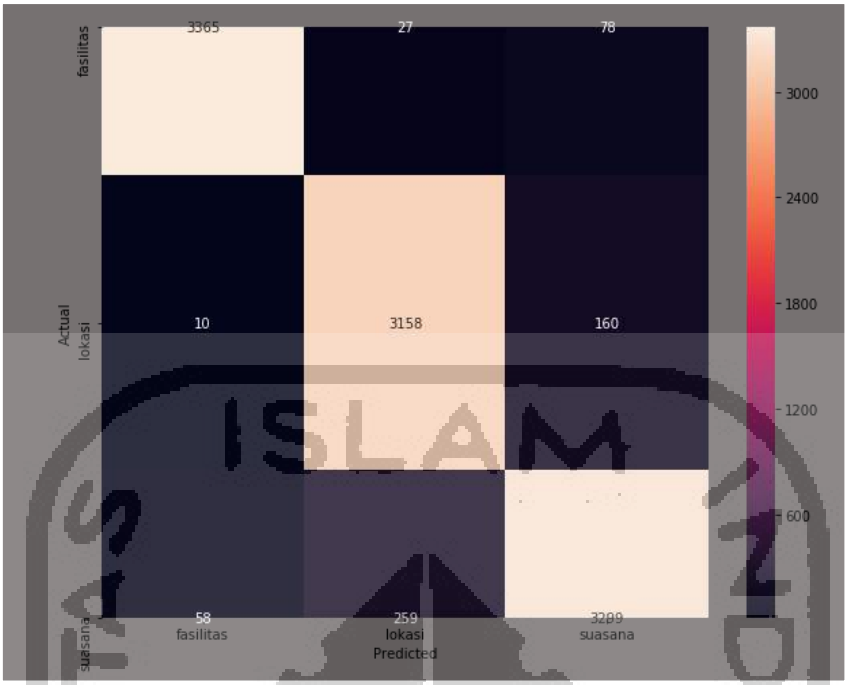
GRU sebesar 64 unit dengan *dropout* sebesar 0.25. Dibandingkan dengan M7 yang menggunakan *kernel, filter* dan *hidden_dims* yang sama, dan *dropout* sebesar 0.5, M6 memiliki nilai akurasi yang tertinggi yaitu 94.31%, presisi 94.32%, dan *recall* 94.36% dengan *epochs* sebanyak 16. Tabel 4.11 merupakan perbandingan *accuracy*, *precision*, dan *recall* aspek pada skenario 3. Gambar 4.38 dan Gambar 4.39 merupakan *confusion matrix* aspek dari M6 dan M7.

Tabel 4.11 Analisis Aspek Skenario 3

Skenario	**Model**	***Epochs***	**Accuracy (%)**	**Precision (%)**	**Recall (%)**
3	**M6**	**16**	**0.9431**	**0.9432**	**0.9436**
	M7	18	0.9418	0.9420	0.9418



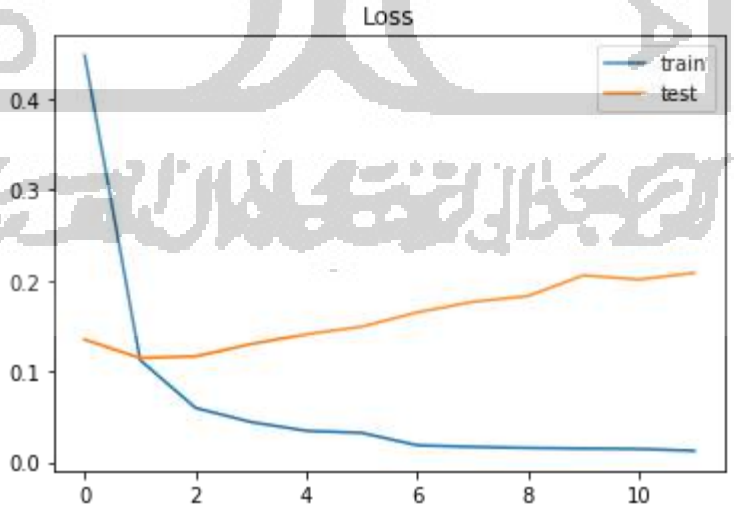
Gambar 4.38 *Confusion Matrix* Aspek M6



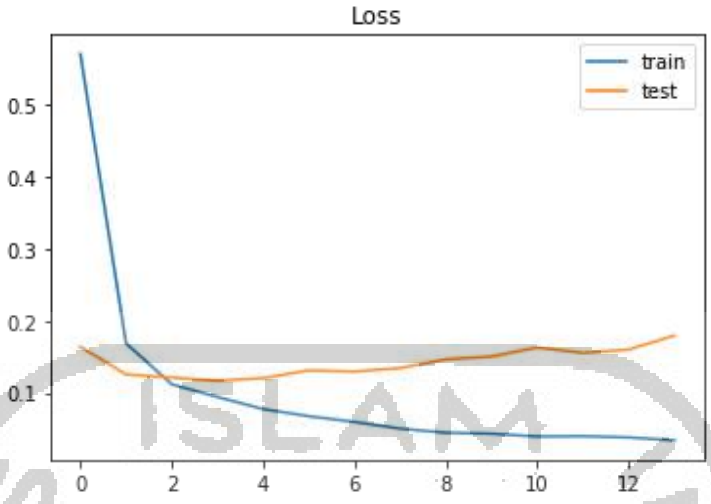
Gambar 4.39 *Confusion Matrix* Aspek M7

d. Analisis Penggunaan *Dropout Layer*

Pada seluruh skenario diterapkan *dropout* layer agar dapat mengurangi *overfitting* ketika pelatihan data. Dari beberapa skenario aspek kategori dan sentimen, model M2 merupakan model yang memiliki akurasi tertinggi dibandingkan M3. M2 dengan *dropout* 0.25 memiliki *loss* 0.215, sedangkan M3 dengan *dropout* 0.5 memiliki *loss* 0.186. Gambar 4.40 dan Gambar 4.41 merupakan *loss* pada model Aspek M2 dan Aspek M3.



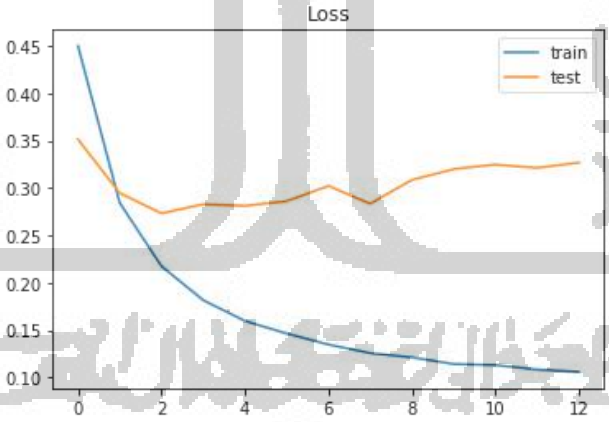
Gambar 4.40 Loss M2 aspek dengan dropout 0.25



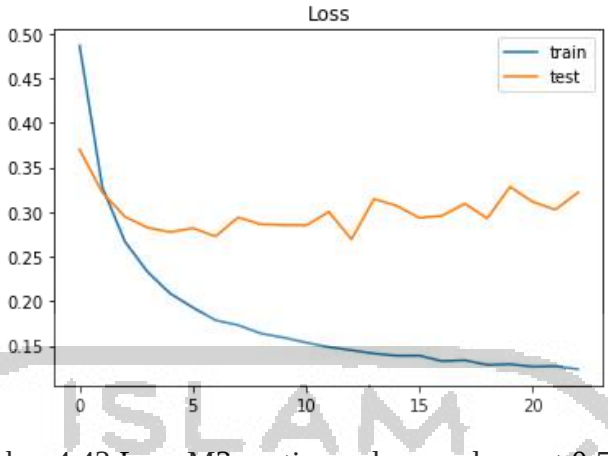
Gambar 4.41 Loss M3 aspek dengan dropout 0.5

Untuk sentimen, M2 dengan *dropout* 0.25 memiliki *loss* 0.327, sedangkan M3 dengan *dropout* 0.5 memiliki *loss* 0. 324. Gambar 4.42 dan Gambar 4.43 merupakan *loss* pada model Sentimen M2 dan Sentimen M3.

Dari kedua *dropout* yang digunakan, menunjukkan bahwa model sentimen maupun aspek kategori dengan *dropout* 0.5 memiliki nilai *loss* yang cukup rendah dan cukup mengatasi *overfitting*.



Gambar 4.42 Loss M2 sentimen dengan dropout 0.25



Gambar 4.43 Loss M3 sentimen dengan dropout 0.5

e. Analisis Penggunaan *Negation* Handling

Pada M2 dilakukan percobaan penggunaan *negation handling*. Tabel 4.12 menunjukkan hasil penggunaan *negation* handling. Dapat disimpulkan bahwa *negation handling* dapat membantu meningkatkan akurasi sebanyak 0.422 ketika melakukan klasifikasi sentimen.

Tabel 4.12 Perbandingan penggunaan *negation handling*

Model	***Negation* Handling**	**Akurasi**	**Precision**	**Recall**
M2	Ya	0.9067	0.9061	0.9107
	Tidak	0.8645	0.8639	0.8684

4.7 Hasil ABSA

Gambar 4.44 merupakan hasil dari penggunaan model CNN untuk mendapatkan resensi ulasan maupun sentimen sesuai pada ulasan yang dimasukkan.

```
Sayangnya tempat ini ramai dan sesak ketika musim libur.
['Sentimen : negatif']
['Aspek Kategori: suasana']

Fasilitasnya tidak memadai.
['Sentimen : negatif']
['Aspek Kategori: fasilitas']

Lokasi tidak panas dan tidak ramai
['Sentimen : positif']
['Aspek Kategori: lokasi']

Hawanya sedikit dingin dengan udara yang segar.
['Sentimen : positif']
['Aspek Kategori: suasana']

Lokasi sama sekali tidak menarik.
['Sentimen : negatif']
['Aspek Kategori: lokasi']
```

Gambar 4.44 Hasil ABSA untuk mendapatkan resensi ulasan