BAB V
HASIL DAN PEMBAHASAN

Bab ini terdiri dari dua sub bab yaitu hasil dan pembahasan. Sub bab hasil membahas hasil pengembangan sistem yang telah dilakukan pada penelitian ini. Sub bab pembahasan berisi penjelasan dan analisis lebih lanjut mengenai model NLP dan pengujian sistem.

5.1 Hasil

Bagian ini membahas hasil penelitian yang telah dilakukan yaitu pemodelan dan implementasi antarmuka.

5.1.1 Pra-proses Data dan Pembuatan Model

(i) Case Folding

```
def case_folding(value: str) -> str:
    """ func. to lowercase string"""
    return value.lower().strip()
```

Gambar 5. 1 Kode Penggalan *Case Folding*

Gambar 5.1 merupakan penggalan kode untuk melakukan *case folding*. Nama fungsi yang didefinisikan adalah *case_folding* dengan 1 argumen bernama *value* dan tipe data harus *string*. *Output* atau *return* dari fungsi ini adalah *string* yang bebas dari huruf kapital dan spasi pada awal-akhir kalimat yang telah dihilangkan.

Tabel 5. 1 Hasil *Case Folding*

Sebelum	Sesudah
@siapkawalpolri Setuju nih sama usulan baru mengenai ETLE ini dari pak @ListyoSigitP agar tidak terjadi lagi hal² yang berbau pungli oleh oknum polisi yang hanya ingin memperkaya diri! https://t.co/cUEmAFuFgA	@siapkawalpolri setuju nih sama usulan baru mengenai etle ini dari pak @listyosigitp agar tidak terjadi lagi hal² yang berbau pungli oleh oknum polisi yang hanya ingin memperkaya diri! https://t.co/cuemafufga

Tabel 5.1 merupakan perbandingan kalimat sebelum dan sesudah proses *case folding*. Perbedaan yang terlihat yaitu huruf kapital berubah menjadi huruf kecil.

(ii) Cleaning

```
def cleaning_text(value):
  result = value.lower().strip()
  result = remove_three_same_char(result)
  result = ' '.join(result.split())
  result = re.sub(r'(@|https?)\S+|#[A-Za-z0-9_]+', '', result).replace("&amp;", "dan")
  result = re.sub(r'RT[\s]+','',result)
  result = begone_emoji.sub(repl='', string=result)
  result = re.sub(r'[0-9]+','',result)
  result = result.replace('\n', ' ')
  result = result.translate(str.maketrans('','',string.punctuation))
  return result
```

Gambar 5. 2 Penggalan Kode *Cleaning*

Gambar 5.2 merupakan penggalan kode untuk proses cleaning. Terdapat beberapa tahap yang menggunakan regex untuk mencari pola dan menggantinya dengan string yang diinginkan. *Method replace* digunakan untuk mengganti kata secara spesifik tanpa menggunakan pola dinamis seperti *regex*. Hasil cleaning data dapat dilihat pada Tabel 5.2.

Tabel 5. 2 Hasil *Cleaning*

Sebelum	Sesudah
@siapkawalpolri Setuju nih sama usulan baru mengenai ETLE ini dari pak @ListyoSigitP agar tidak terjadi lagi hal² yang berbau pungli oleh oknum polisi yang hanya ingin memperkaya diri! https://t.co/cUEmAFuFgA	setuju nih sama usulan baru mengenai etle ini dari pak agar tidak terjadi lagi hal yang berbau pungli oleh oknum polisi yang hanya ingin memperkaya diri

Tabel 5.2 merupakan perbandingan sebelum dan sesudah cleaning data. Adapun perbedaan yang terlihat yaitu hilangnya *mentions*, *hyperlinks* dengan pola *http/s*, tanda baca, dan angka.

(iii) Normalisasi Kata

```
def remove_slang(value):
    kamus = eval(open("combined_slang_words.txt").read())
    pattern = re.compile(r"\b(%s)\b" % "|".join(kamus))
    converted = pattern.sub(lambda word: kamus.get(word.group()), value)
    # clear = re.sub(r'[^\w\s]', ' ', converted)
    return converted
```

Gambar 5. 3 Penggalan Kode Normalisasi Kata

Gambar 5.3 merupakan penggalan kode untuk melakukan normalisasi kata. Variabel kamus diisi dengan *method eval* dengan input kamus Bahasa gaul beserta subtitusi kata bakunya dalam bentuk *txt*. Pola yang digunakan untuk mengidentifikasi. Hasil normalisasi dapat dilihat pada Tabel 5.3.

Tabel 5. 3 Hasil Normalisasi Kata

Sebelum	Sesudah
setuju nih sama usulan baru mengenai etle ini dari pak agar tidak terjadi lagi hal yang berbau pungli oleh oknum polisi yang hanya ingin memperkaya diri	setuju ini sama usulan baru mengenai etle ini dari pak agar tidak terjadi lagi hal yang berbau pungutan liar oleh oknum polisi yang hanya ingin memperkaya diri

Tabel 5.3 merupakan perbandingan setelah dilakukan normalisasi kata sesuai kamus yang dipakai. Perbedaan terlihat pada kata pungli yang berubah menjadi pungutan liar.

(iv) Tokenisasi

```
def tokenizing_text(value):
  result= word_tokenize(value)
  return result
```

Gambar 5. 4 Penggalan Kode Tokenisasi Kata

Gambar 5.4 merupakan penggalan kode untuk melakukan tokenisasi. *Library* yang digunakan untuk memisahkan kata yaitu *word_tokenize* dari *sklearn*. Tipe data kalimat berubah dari *string* menjadi *list*.

Tabel 5. 4 Hasil Tokenisasi Kata

Sebelum	Sesudah
setuju ini sama usulan baru mengenai etle ini dari pak agar tidak terjadi lagi hal yang berbau pungutan liar oleh oknum polisi yang hanya ingin memperkaya diri	['setuju', 'ini', 'sama', 'usulan', 'baru', 'mengenai', 'etle', 'ini', 'dari', 'pak', 'agar', 'tidak', 'terjadi', 'lagi', 'hal', 'yang', 'berbau', 'pungutan', 'liar', 'oleh', 'oknum', 'polisi', 'yang', 'hanya', 'ingin', 'memperkaya', 'diri']

Tabel 5.4 menunjukan perbandingan sebelum dansetelah tokenisasi. Kalimat yang terdiri dari beberapa kata bertipe data string diubah menjadi *list* dengan jummlah anggota sebanyak jumlah kata pada kalimat.

(v) Stopword Removal

```
def filtering_stopwords(value):
  list_stopwords = set(stopwords.words('indonesian'))
  filtered = []
  for text in value:
    if text not in list_stopwords:
      filtered.append(text)
  return filtered
```

Gambar 5. 5 Penggalan Kode *Stopword Removal*

Gambar 5.5 merupakan penggalan kode untuk menghilangkan *stopword*. Sumber data *stopword* bahasa indonesia berasal dari *library NLTK*. *List* kata dalam kalimat dilakukan perulang dengan kondisi apabila kata tidak terdaftar di sumber data *stopword* maka kata tersebut dimasukkan ke dalam *list* kata baru dan dikembalikan.

Tabel 5. 5 Hasil *Stopword Removal*

Sebelum	Sesudah
['setuju', 'ini', 'sama', 'usulan', 'baru', 'mengenai', 'etle', 'ini', 'dari', 'pak', 'agar', 'tidak', 'terjadi', 'lagi', 'hal', 'yang', 'berbau', 'pungutan', 'liar', 'oleh', 'oknum', 'polisi', 'yang', 'hanya', 'ingin', 'memperkaya', 'diri']	['setuju', 'usulan', 'etle', 'hal', 'berbau', 'pungutan', 'liar', 'oknum', 'polisi', 'memperkaya']

Tabel 5.5 merupakan tabel perbandingan hasil *stopword removal*. Kata yang termasuk dalam kategori *stopword* dihilangkan karena dianggap tidak relevan seperti ini, sama, baru, yang, oleh, dsb.

(vi) Stemming

```
def stemming_text(value):
  factory = StemmerFactory()
  stemmer = factory.create_stemmer()
  return [stemmer.stem(word) for word in value]
```

Gambar 5. 6 Penggalan Kode *Stemming*

Gambar 5.6 merupakan penggalan kode untuk melakukan stemming atau mengubah kata yang mengandung imbuhan mendasi kata dasar. *Library* yang digunakan pada penggalan kode ini adalah *PySastrawi*. Langkah awal untuk menggunakan *stemmer* yaitu inisiasi *objek factory* di variabel. Langkah selanjutnya yaitu membuat *stemmer* dari *factory*. Langkah terakhir yaitu *stem* setiap kata.

Tabel 5. 6 Hasil *Stemming*

Sebelum	Sesudah
['setuju', 'usulan', 'etle', 'hal', 'berbau', 'pungutan', 'liar', 'oknum', 'polisi', 'memperkaya']	['tuju', 'usul', 'etle', 'hal', 'bau', 'pungut', 'liar', 'oknum', 'polisi', 'kaya']

Tabel 5.6 merupakan perbandingan hasil stemming kata. Seluruh kata yang ada pada list diubah menjadi kata dasar dengan menghilangkan prefiks, sufiks, dan infiks. Perbedaan terlihat pada kata setuju menjadi tuju, usulan menjadi usul, berbau menjadi bau, memperkaya menjadi kaya.

(vii) Pelabelan Data

```
def lexicon_reader(path: str) -> dict:
    with open(path, 'r') as csv_data:
        bucket = dict()
        reader = list(csv.reader(csv_data, delimiter =","))
        for row in reader[1:]:
        # print(row[0])
            bucket[row[0]] = int(row[1])
    return bucket

def sentiment_lexicon_indonesia(value):
    score = 0
    for word in value:
        word = word.replace("'","")
        if word in lexicon_positive:
            score += lexicon_positive[word]
        if word in lexicon_negative:
            score += lexicon_negative[word]
    if score > 0:
        polarity = 'positif'
    elif score < 0:
        polarity = 'negatif'
    else:
        polarity = 'netral'
    return score, polarity

lexicon_positive = lexicon_reader('positive.csv')
lexicon_negative = lexicon_reader('negative.csv')
```

Gambar 5. 7 Penggalan Kode Pelabelan Data Otomatis

Gambar 5.7 merupakan penggalan kode untuk menilai polaritas kalimat. Langkah pertama yaitu membaca kamus leksikon yang berisi pasangan kata dan nilai polaritasnya. Adapun cara menghitung polaritas kalimat yaitu menjumlahkan polaritas setiap kata dalam kalimat. Kalimat dianggap negatif jika nilai total polaritas kurang dari 0. Hal tersebut berlaku untuk netral jika nilai 0 dan positif jika nilai 1.

```
polarity_encode = {'negatif':0, 'netral':1, 'positif':2 }
# y = df_etle['polarity'].map(polarity_encode).values

y = pd.get_dummies(df_etle['polarity']).values
```

Gambar 5. 8 Penggalan Kode *Mapping Values*

Gambar 5.8 merupakan penggalan kode untuk mengubah kalimat pada nilai y atau target di *dataset*. Pemetaannya yaitu sentimen negatif diubah menjadi angka 0, sentimen netral diubah menjadi angka 1, dan sentimen positif diubah menjadi angka 2. Sentimen yang telah diubah menjadi angka kemudian diubah menjadi nilai biner. Angka 0 direpresentasikan dengan biner [0 0 0], angka 1 direpresentasikan dengan biner [0 1 0], dan angka 2 direpresentasikan dengan biner [1 0 0].

(viii) Pembuatan Model

```
def model_initiator(nn_type: str = "LSTM"):
    model = Sequential()
    model.add(Embedding(input_dim=max_feature+1, output_dim=64, input_length=X.shape[1]))
    model.add(Dropout(0.5))
    if nn_type == 'GRU':
        model.add(GRU(units=128, dropout=0.2, recurrent_dropout=0.2))
    else:
        model.add(LSTM(units=128, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dropout(0.5))
    model.add(Dense(units=3, activation="softmax"))
    model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
    model.summary()
    plot_model(model, to_file=f'{nn_type}_diagram.png', show_shapes=True, show_layer_names=True)
    return model
```

Gambar 5. 9 Penggalan Kode Inisiasi Model LSTM

Gambar 5.9 merupakan penggalan kode untuk membuat desain jaringan dari arsitektur *LSTM* untuk melakukan klasifikasi sentimen. Model terdiri dari modul *embedding* untuk memasukkan data yang telah dikonversi menjadi angka dan disesuaikan ukurannya ke layer *LSTM*. Layer LSTM yang digunakan pada penelitian ini memiliki unit sebanyak 128. *Layer* dropout digunakan untuk mencegah *overfitting* atau akurasi model mencapai 99% dengan cara menonaktifkan beberapa jaringan *LSTM*. *Layer dense* dengan jumlah unit 3 digunakan sebagai *output* atau luaran hasil pengolahan jaringan dengan aktifasi *softmax*.

5.1.2 Hasil Pemodelan NLP

Tabel 5. 7 Hasil Pemodelan

Batch Size	Epoch	Rasio Data	Akurasi
32	10	80:20	77,4%
64	10	80:20	70,3%
128	10	80:20	71,7%

32	10	70:30	64,4%
64	10	70:30	68,8%
128	10	70:30	73,4%
32	10	60:40	71,4%
64	10	60:40	72,2%
128	10	60:40	71,6%

Tabel 5.7 menggambarkan hasil uji pengaruh *hyperparameter* terhadap akurasi yang diperoleh. Hasil di atas merupakan olahan dataset yang telah dilakukan *balancing* dengan teknik *random downsampling* karena dataset awal yang bersifat *imbalance dataset*. *Imbalance dataset* yaitu keadaan dimana *dataset* memiliki kesenjangan antar kelas. *Hyperparameter* yang diuji yaitu *batch size* dan rasio pembagian data. *Epoch* yang digunakan pada pengujian adalah 10.

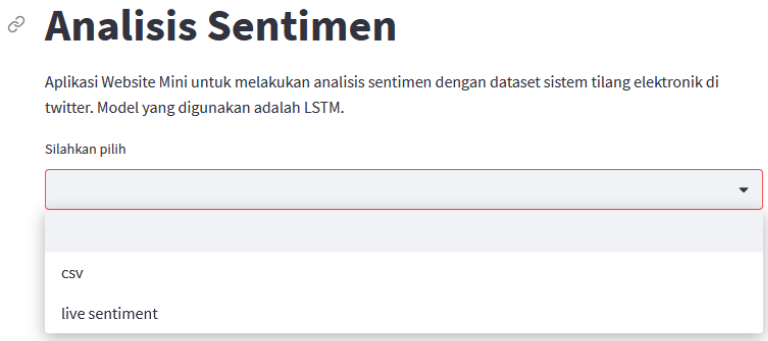
Epoch digambarkan sebagai setiap sampel dalam *dataset* pelatihan memiliki kemungkinan untuk memperbarui parameter model internal satu kali selama satu epoch. Seluruh *batch* data yang telah melalui model dihitung 1 *epoch*. Jumlah *epoch* dapat berjumlah 1 atau lebih. *Batch size* digambarkan sebagai data latih yang dipotong kecil-kecil menjadi beberapa bagian. *Batch* yang lebih kecil ini dapat dengan mudah dimasukkan ke dalam model pembelajaran mesin untuk melatihnya. *Batch size* memiliki ukuran maksimal sesuai jumlah data latih dan minimal 1.

Rasio pembagian data 80:20 memiliki akurasi yang lebih tinggi dibandingkan rasio 70:30 dan 60:40 dengan rata-rata akurasi 73,1%. Rasio pembagian data 70:30 memiliki rata-rata akurasi terendah yaitu 0,688. Rasio pembagi data 60:40 memiliki rata-rata akurasi 71,7%. Akurasi tertinggi diperoleh dengan nilai 77,4%.

Pada pengujian *hyperparameter batch size* dengan ukuran 32 memiliki akurasi rata-rata 71%. *Batch size* dengan ukuran 64 memiliki akurasi rata-rata 70,4%. *Batch size* dengan ukuran 128 memiliki akurasi rata-rata sebesar 72,2%.

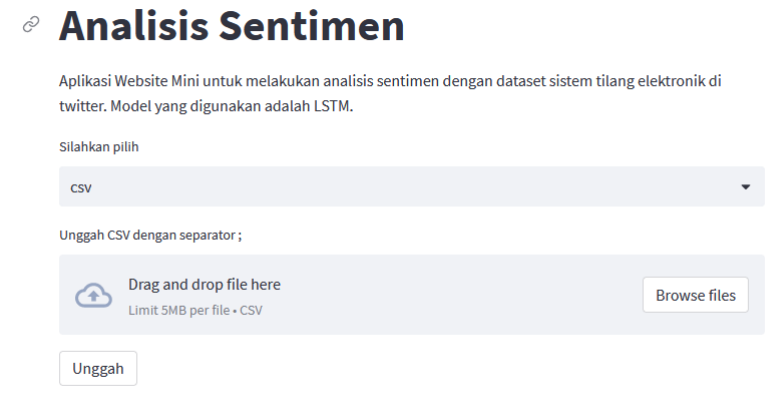
5.1.3 Hasil Implementasi Perancangan Antarmuka

Aplikasi analisis sentimen dengan model NLP berhasil diimplementasikan sesuai dengan rancangan yang telah dibuat pada bab sebelumnya. Berikut merupakan hasil implementasi perancangan antarmuka.



Gambar 5. 10 Hasil Implementasi *Homepage*

Gambar 5.10 merupakan tampilan awal jika *user* mengakses aplikasi. *User* dapat memilih beberapa opsi yaitu *live sentiment* dan unggah *csv*. Ketika memilih opsi upload csv, *user* akan ditampilkan *upload box*. Ketika memilih opsi *live sentiment*, *user* akan ditampilkan *input form*.

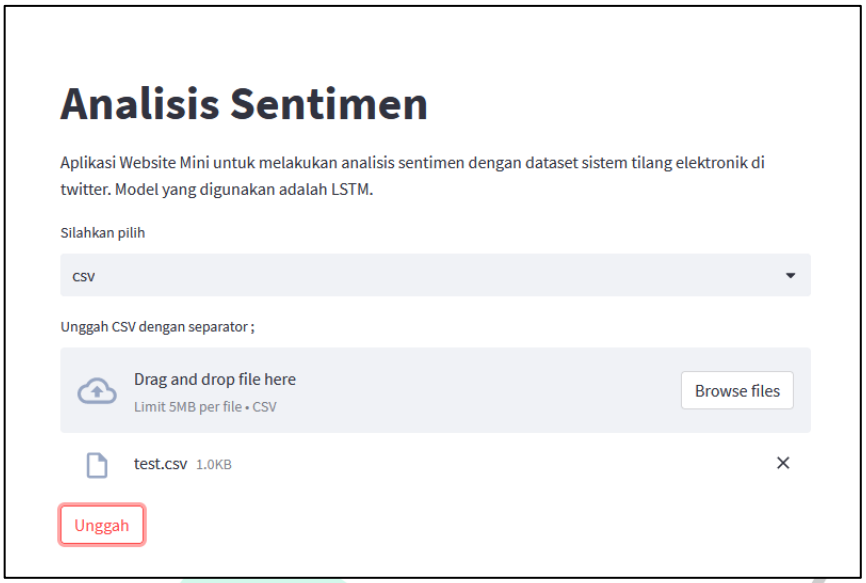


Gambar 5. 11 Hasil Implementasi Opsi *Upload csv*

Gambar 5.11 merupakan halaman tampilan untuk mengunggah file *csv*. Pada bagian ini terdapat tombol yang berfungsi untuk memilih file yang ingin diunggah. File yang akan diunggah harus memiliki format *csv* dengan *delimiter* titik koma dan ukuran maksimal 5MB. Aturan tersebut digunakan untuk meminimalisasi galat akibat format eksekusi sistem yang tidak sesuai dengan file yang diunggah.

Gambar 5. 12 Hasil Implementasi Opsi *Live Sentiment*

Gambar 5.12 merupakan tampilan untuk menu *live sentiment*. Form pada opsi ini berisi input *text query* dengan tipe data *string*, input tanggal mulai bertipe data *date*, dan input tanggal akhir dengan tipe data *date*. Tombol *submit* dapat ditekan jika *user* sudah mengisi seluruh *input form*.

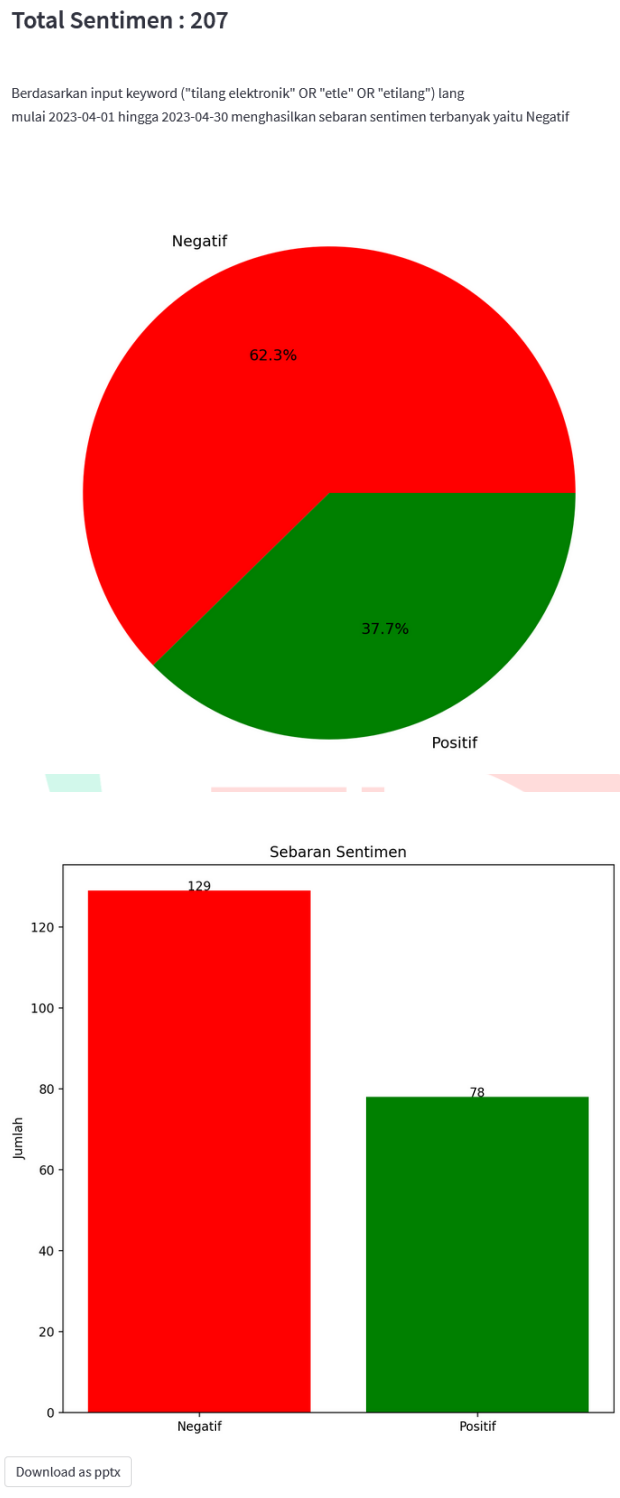


Gambar 5. 13 Tampilan File yang Telah Terunggah

Gambar 5.13 merupakan tampilan setelah *user* memilih file. Nama file dan ukuran file akan muncul di bawah kolom unggah. Tombol silang pada samping kanan berfungsi untuk menghapus unggahan file jika ingin mengubah file.

Gambar 5. 14 Tampilan Animasi Pengolahan Data

Gambar 5.14 merupakan tampilan sistem setelah *user* menekan tombol unggah. Animasi *loading* akan muncul sebagai tanda bahwa sistem tidak *error* atau sedang melakukan proses analisis sentimen.



Gambar 5. 15 Tampilan Statistik Hasil Pengolahan Sentimen

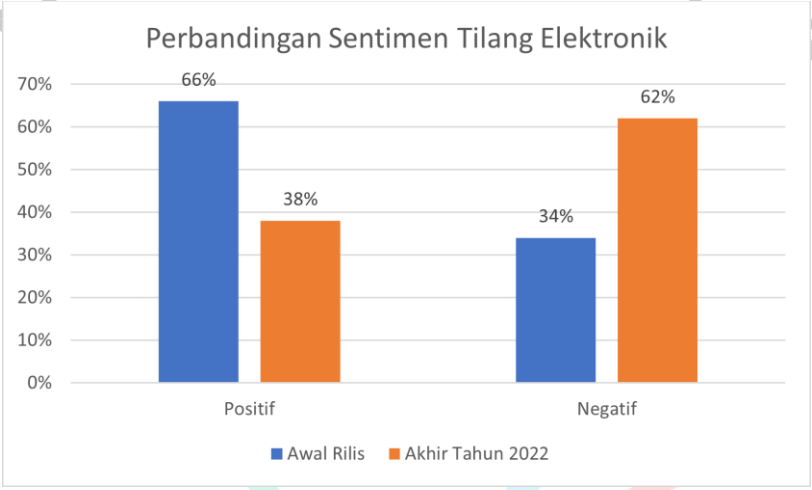
Gambar 5.15 adalah tampilkan akhir berupa statistik sentimen. Adapun statistic yang digunakan yaitu total sentimen, sentimen terbanyak, diagram venn,

dan diagram batang. Data statistik berasal dari file yang diunggah atau *live sentiment* kemudian dibaca sistem dan model melakukan prediksi.

5.2 Pembahasan

Pembahasan merupakan hasil dari evaluasi sistem yang telah dikembangkan dan mengetahui kinerja atau fungsi dari aplikasi analisis sentimen terhadap sistem tilang elektronik.

5.2.1 Perbandingan sentimen saat awal rilis dan akhir tahun 2022



Gambar 5. 16 Visualisasi Perbandingan Sentimen

Gambar 5.16 menggambarkan perbandingan sentimen saat awal rilis dan dan akhir tahun 2022. Pada awal rilis tilang elektronik, sentiment positif memiliki persentase lebih besar daripada sentimen negatif yaitu 66% berbanding 34%. Sebaran sentimen pada akhir tahun 2022 mengalami perubahan yaitu sentimen negatif naik menjadi 62%, sentimen positif turun menjadi 38%. Pergeseran mayoritas sentimen berubah dari positif menjadi negatig pada akhir tahun 2022.

5.2.2 Perbandingan pelabelan manual dan otomatis

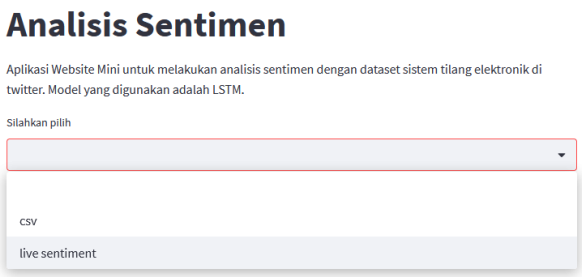
Pelabelan otomatis dan manual mengalami perbedaan. Pelabelan manual mendapatkan sentimen negatif sebanyak 935, positif sebanyak 532, dan netral sebanyak 4882. Sedangkan pelabelan otomatis mendapatkan sentiment negatif sebanyak 3815, positif sebanyak 1960, dan netral sebanyak 574. Hasil kedua

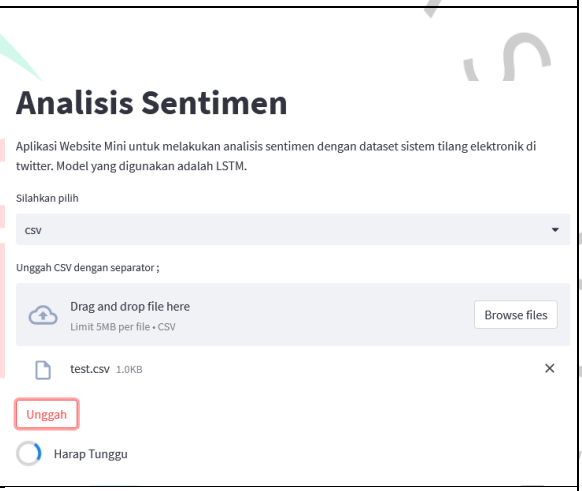
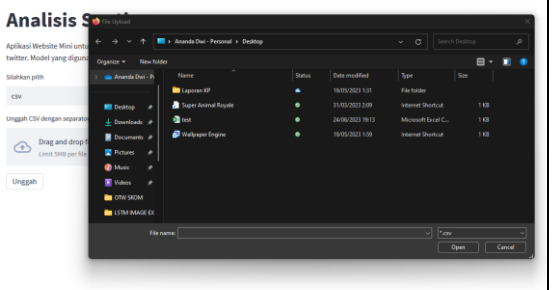
metode tersebut berbeda karena pelabelan otomatis menggunakan tipe *lexicon*, yaitu pembobotan berdasarkan kata kemudian dihitung total dari penambahan bobot antar kata di setiap kalimat. Pelabelan manual mendapatkan nilai netral lebih banyak karena mayoritas *tweet* yang didapatkan berupa berita mengenai pelanggaran yang dideteksi oleh tilang elektronik. *Tweet* tersebut adalah netral karena hanya berisikan informasi atau berita, sedangkan pelabelan otomatis menghitung polaritas berdasarkan total bobot yang dimiliki oleh kalimat tersebut. Jika total bobot bernilai positif maka polaritas sentimen adalah positif. Jika total bobot bernilai negatif maka polaritas sentimen adalah negatif.

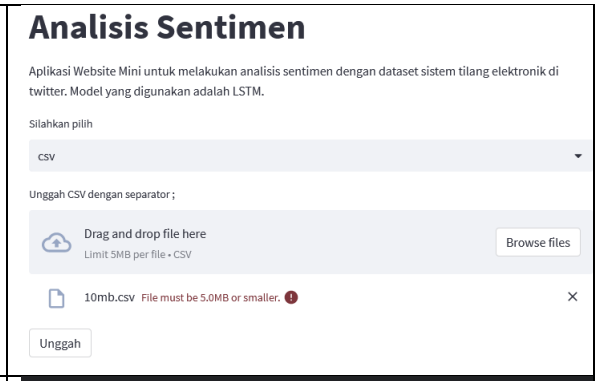
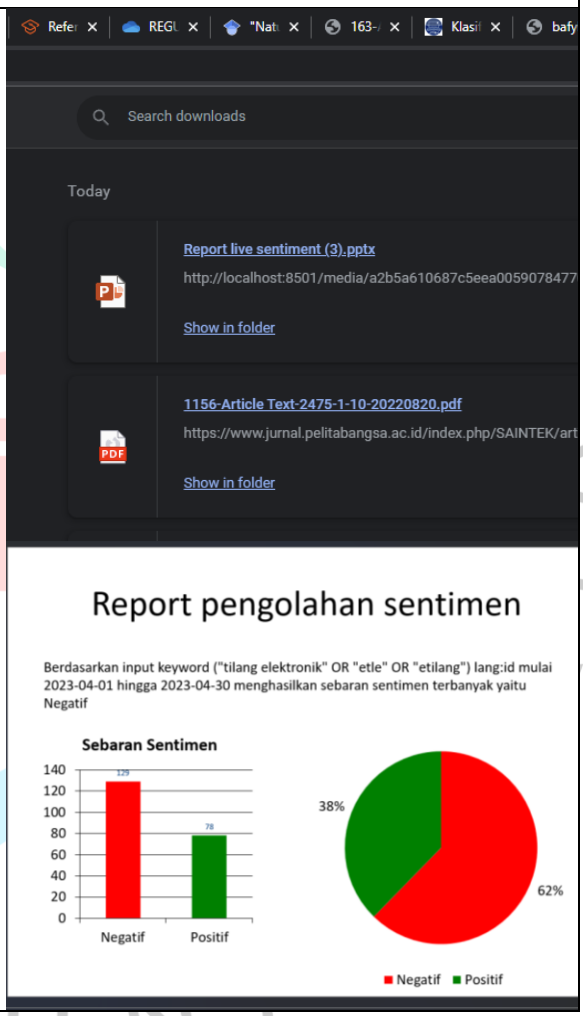
5.2.3 Pengujian *Black box*

Pengujian *black box* berfokus pada pengujian kualitas fitur yang telah dikembangkan. Pengujian dilakukan untuk memastikan bahwa fungsi dari fitur berjalan dengan baik.

Tabel 5. 8 Skenario Pengujian *Black Box*

No	Fitur yang diuji	Hasil yang diharapkan	Hasil Pengamatan
1	Menu Dropdown	*User* dapat memilih dan masuk ke dalam halaman *live sentiment* atau *upload csv*.	

2	Kolom *Form*	Sistem menampilkan isi *form* yang diisi oleh *user*.	
3	*Input*	Sistem menampilkan animasi *loading* sebagai tanda bahwa sistem sedang memproses *input* dari *user*.	
4	Tombol Pemilihan *File*	Sistem menampilkan jendela *file explorer* untuk memilih *file*.	
5	File *Limiter*	Sistem menampilkan notifikasi bahwa *file* melebihi batas.	

			
6	Tombol Simpan	*Browser* menyimpan *file pptx* yang diberikan oleh sistem.	

Tabel 5.8 merupakan skenario pengujian *black box*. Pada pengujian akses halaman utama, user dapat memilih opsi *csv* atau live sentiment. Pada opsi *csv*, *user* dapat mengunggah file yang memenuhi syarat dan tombol dapat ditekan. Pada opsi *Live sentiment*, *user* dapat mengisi *keyword* pencarian *tweet*, tanggal awal, dan tanggal akhir. Jika semua *form* telah terisi, *user* dapat menekan tombol *submit*. Opsi *upload csv* dan *live sentiment* mampu menyajikan hasil klasifikasi sentimen dalam

bentuk kalimat dan diagram. Tombol *download* untuk opsi *pptx* juga berjalan dengan baik. Seluruh skenario pengujian *black box* berjalan dengan baik dan tidak terdapat masalah.

5.2.4 Pengujian *White box*

Pengujian *white box* berfokus pada pengujian struktur kode yang telah dikembangkan. Pengujian dilakukan untuk memastikan bahwa kode berjalan dengan baik. Tabel 5.3 menunjukkan beberapa skenario pengujian dengan berisi hasil yang dirapkan, kode program dan hasil pengujian.

Tabel 5. 9 Skenario Pengujian *Whitebox*

No	Hasil yang diharapkan	Kode Program	Hasil Pengujian
Proses *Preprocessing Data*			

1	Aplikasi memuat model dan tokenizer ke dalam cache browser	1 usage anandadwii *
		@st.cache_resource
		def tokenizer_loader():
		"""fungsi untuk load tokenizer ke dalam cache website"
		with open('manual_tokenizer.pickle', 'rb') as handler:
		tokenizer = pickle.load(handler)
		return tokenizer
		1 usage anandadwii *
		@st.cache_resource
		def model_loader():
		""" fungsi untuk load model ke dalam streamlit
		return load_model('balanced_manual_model.h5')
			Valid

2	Melakukan eliminasi emoji pada kalimat	def remove_emoji(): """menghilangkan emoji""" emojis = sorted(emoji.EMOJI_DATA, key=len, reverse=True) pattern = u'(' + u'|'.join(re.escape(u) for u in emojis) + u')' return re.compile(pattern)	Valid

<table>
<tr><td>3</td><td>Menyamakan tipe huruf menjadi kecil, menghapus hyperlinks, mention, hastag, emoji, dan angka</td><td><pre>def cleaning_text(value):
    result = value.lower().strip()
    # result = remove_three_same_char(result)
    result = ' '.join(result.split())
    result = re.sub(r'(@|https?)\S+|#[A-Za-z0-9_]+',
                    '', result).replace("&", "dan")
    result = re.sub(r'RT[\s]+', '', result)
    result = begone_emoji.sub(repl='', string=result)
    result = re.sub(r'[0-9]+', '', result)
    result = result.replace('\n', ' ')
    result = result.translate(
        str.maketrans('', '', string.punctuation))
    return result</pre></td><td>Valid</td></tr>
<tr><td>4</td><td>Menghilangkan stopword pada kalimat</td><td><pre>def filtering_stopwords(value):
    """ menghilangkan stopword pada kalimat
    list_stopwords = set(
        stopwords.words('indonesian'))
    list_stopwords.remove("tidak")
    filtered = []
    for text in value:
        if text not in list_stopwords:
            filtered.append(text)
    return filtered</pre></td><td>Valid</td></tr>
<tr><td colspan="4">Proses Prediksi</td></tr>
<tr><td>5</td><td>Word Embedding input, melakukan prediksi, dan encode output menjadi kata.</td><td><pre>df = pd.DataFrame(_result, columns=['processed'])
X = df['processed']
X = tokenizer.texts_to_sequences(X.values)
X = pad_sequences(X, maxlen=57)
y_pred = model.predict(X, verbose=0)
y_pred = np.argmax(y_pred, axis=1)
df['pred'] = y_pred
polarity_encode = {0: 'Negatif',
                   1: 'Netral',
                   2: 'Positif'}
df['pred'] = df['pred'].map(polarity_encode).values
labels = ['Negatif', 'Positif', 'Netral']
color_bar = ['red', 'green', 'gray']
sizes = [0, 0, 0]
for key, value in df['pred'].value_counts().items():
    if key == 'Positif':
        sizes[1] = value
    elif key == 'Negatif':
        sizes[0] = value
    elif key == 'Netral':
        sizes[2] = value</pre></td><td>Valid</td></tr>
<tr><td colspan="4">Output</td></tr>
</table>

6	Menampilkan diagram venn	def pie_chart(x, label, color): """ to create pie chart and return figure and ax """ fig, ax = plt.subplots(figsize=(8, 8)) explode = (0.1, 0, 0) ax.pie(x=x, labels=label, colors=color, autopct='%1.1f%%', textprops={'fontsize': 11}) return fig, ax	Valid

7	Menampilkan diagram batang	3 usages anandadwii *
		def bar_chart(x, height, sizes, color):
		"""
		function to create bar chart and return figur
		"""
		fig, ax = plt.subplots(figsize=(8, 8))
		ax.bar(x, height, color=color)
		for i in range(len(height)):
		ax.text(i, sizes[i], sizes[i], ha='center')
		# ax.set_xlabel('Nilai Prediksi')
		ax.set_ylabel('Jumlah')
		ax.set_title('Sebaran Sentimen')
		return fig, ax
			Valid

8	Menampilkan tombol *download*	st.download_button(label="Download as pptx", data=create_ppt(dict_data, paragraph), file_name=f"Report {str(selectbox)}.pptx")	Valid

Tabel 5.9 merupakan skenario pengujian *white box*. Pengujian *white box* dimulai saat aplikasi melakukan loading model dan *tokenizer* ke *session web* browser. Hasilnya aplikasi mampu melakukan *loading* model dan *tokenizer*. Kemudian menguji kode program untuk melakukan eliminasi *emoji*, *stopwrod removal*, dan *cleaning text*. Kode berjalan dengan baik dan output kode sesuai dengan harapan. Pengujian selanjutnya yaitu kode untuk melakukan *word embedding*, *padding*, melakukan prediksi dan *encode* output dari bentuk numerik menjadi kata. Skenario terakhir yaitu pengujian kode untuk menampilkan diagram dan tombol *download*. kode dapat menampilkan diagram dan tombol *download*.