Web Scraping Dan Analisa Sentimen Ulasan Aplikasi Livin' By Mandiri Menggunakan Deep Learning Long Short-Term Memory (LSTM)

Kheylina Lidya Situmorang

Maret 2025

Abstrak

Di era yang serba digital, Bank Mandiri menghadirkan aplikasi Livin' by Mandiri sebagai penyempurnaan dari aplikasi Mandiri Online. Aplikasi Livin' by Mandiri adalah Financial Super App Yang memanfaatkan pendekatan artificial intelligence (AI) untuk menciptakan sentuhan unik dan modern dalam akses layanan keuangan, livin berfungsi sebagai alat yang digunakan untuk meingkatkan kemudahan dan kenyamanan nasabah dalam mengakses layanan perbankan. Penelitian ini bertujuan untuk menganalisa sentimen pengguna aplikasi Livin dengan menggunakan algoritma deeplearning LSTM. Dataset yang akan digunakan adalah komentar pengguna di google playstore pada aplikasi livin maka pada tahapan ini menggunakan metode webscraping untuk mengambil data sebagai analisa sentimen pada penilitian ini.

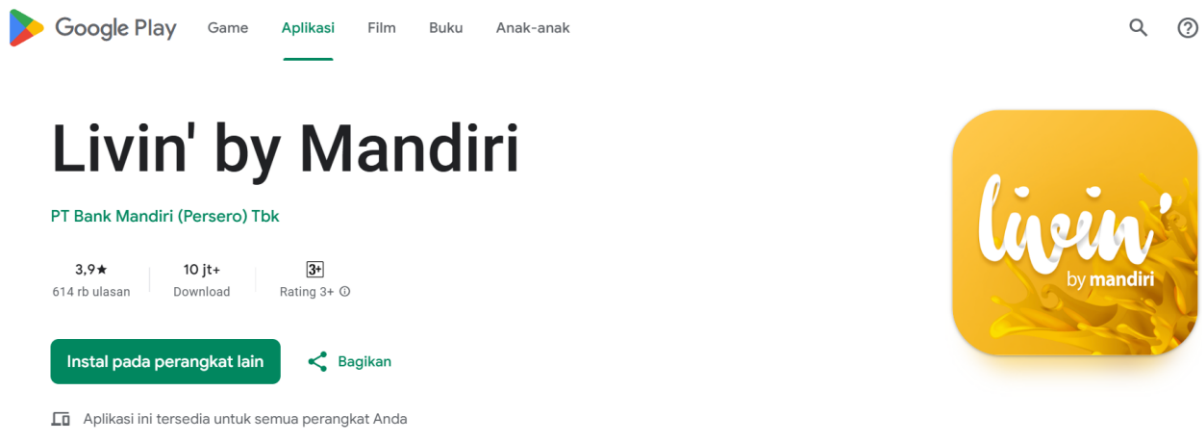
1. **Tujuan**

Tujuan pada penilitian ini adalah :

- Bagaiamana LSTM dapat menganalisis sentimen terhadap ulasan komentar pada aplikasi livin.
- Untuk memberikan Insihgt kepada Perushanan Bank Mandiri melalui Word Cloud dalam peningkatan Aplikasi Livin.

2. **Dataset**

Dataset yang digunakan pada penilitian ini adalah ulasan kometar pengguna livin di App Store/Playstore. Data diambil melalui metode webscraping di Website ini : https://play.google.com/store/apps/details?id=id.bmri.livin



Gambar App Livin Di Google Play Store

3. Livin' By Mandiri

App Livin adalah aplikasi yang digunakan sebagai alat untuk meningkatakan kemudahan dan kenyamanan nasabah dalam mengakses layanan perbankan. Livin juga memiliki banyak feature yang dapat gunakan oleh Pengguna seperti transfer,Bayar/VA, Top-Up,E-money,QR Terima. Aplikasi Livin memanfaatkan penedekatan artificial intelligence (AI) untuk menciptakan sentuhan unik dan lebih modern dalam akses layanan keungan, applikasi Livin juga mendukungan Sistem keamanan yang baik sehingga aman untuk gunakaan dan aplikasi livin terdapat dokumen laporan pajak yang bisa membantu pengguna dalam laporan SPT tahunan.

Gambar user interface pada App Livin

4. Metode

4.1 Text Mining

Text Mining merupakan proses penambangan data berupa teks dan sumber datanya didapat dari dokumen dan bertujuan unrutuk mencari kata-kata yang mewakili isi dari dokumen sehingga dapat dianalisa keterbuhubungan antar dokumen. Tujuan dari Text Mining adalah mengekstrak informasi yang berguna dari sumber data. Jadi, sumber data yang digunakan pada text mining adalah sekumpulan dokumen yang memilki format yang tidak tersutruk melalaui identifikasi dan eksplorasi pola yang menarik. Adapun tahapan pada text mining yang dilakukan pada penlitian ini adalah preprocessing pada data.

4.2 Pre-Processing

Pada Penilitian ini preprocessing dilakukan untuk mengola data yang ada sehingga penliti dapat menghindari gangguan pada data-data yang tidak konsisten. Tujuannya agar hasil output dari klasifikasi memiliki tingkat keakuratan yang tinggi. Tahapan dari preprocessing meliputi case folding, tokenizing, dan filtering. Adapun penjelasan dari masing-masing tahapan tersebut antara lain:

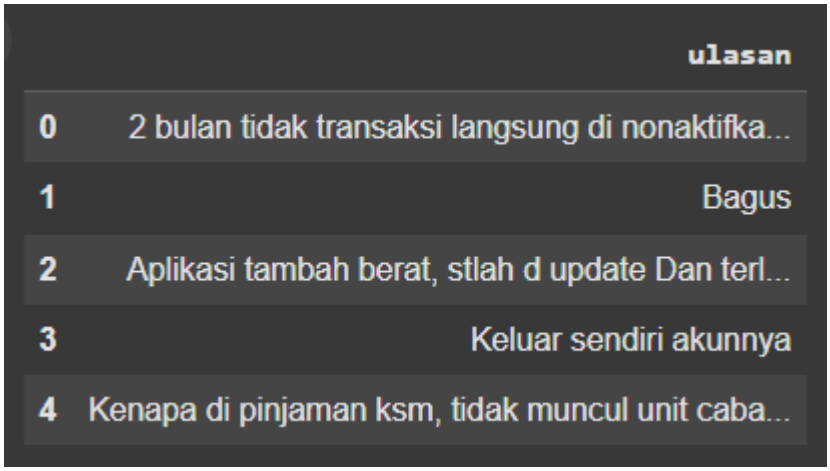
a) Case Folding merupakan tahap mengubah semua huruf yang terdaoat pada komentar menjadi huruf. Hanya huruf "a-z" yang dapat diterima.

Sebelum Case Folding	Sesudah Case Folding
HI NAMA SAYA KHEYLINA	Hi nama saya kheylina
AkU CaYaNg KaMu	Aku cayang kamu

b) Tokenizing adalah tahap pemotongan string input berdasarkan tiap kata yang menyusunnya. Secara garis besar memecah sekumpulan karakter dalam suatu teks ke dalam satuan kata.
c) Filtering adalah tahap mengambil kata-kata yang dianggap penting. Tahapan dalam fitering pada penilitian ini seperti :
    a) Menghapus angka dan tanda baca
    b) Menghapus Stopwords
    c) Normalisasi kata
    d) Steamming

4.3 Scraping

Scraping dilakukan untuk mengumpulkan data yang selanjutnya digunakan sebagai pembelajaran bagi Machine Learning. Alamat url Aplikasi livin di google playstore.



	ulasan
0	2 bulan tidak transaksi langsung di nonaktifka...
1	Bagus
2	Aplikasi tambah berat, stlah d update Dan terl...
3	Keluar sendiri akunnya
4	Kenapa di pinjaman ksm, tidak muncul unit caba...

Gambar hasil web scraping

4.4 Labelisasi dengan Lexicon Based

Klasifikasi sentimen dengan Lexical Based adlaah klasifikasi berdasarakan kata positif, kat negatif ataupun netral yang ada pada ulasan kometar pada aplikasi Livin.

Klasifikasi dicocokan dengan kata- kata yang terdapat dalam kamus Lexicon Bahasa Indonesia. Jika komentar memiliki kata positif, maka akan digolongkan pada sentimen positif, jika komentar miliki kata negatif maka akan digolongkan pada sentiemen negatif, namun jika komentar bernilai sama maka akan digolong pada sentimen netral.

Sentence Sentiment :

- **Positive** if $S_{positive} > S_{negative}$
- **Neutral** if $S_{positive} = S_{negative}$
- **Negative** if $S_{postive} < S_{negative}$

	ulasan	clean_ulasan	sentimen
0	2 bulan tidak transaksi langsung di nonaktifka...	bulan tidak transaksi langsung di nonaktif ke ...	negatif
1	Bagus	bagus	positif
2	Aplikasi tambah berat, stlah d update Dan terl...	aplikasi tambah berat stlah d update dan terla...	netral
3	Keluar sendiri akunnya	keluar sendiri akun	netral
4	Kenapa di pinjaman ksm, tidak muncul unit caba...	kenapa di pinjam ksm tidak muncul unit cabang ...	negatif

Gambar hasil Lebeling dengan Lexicon Based

4.5 Algortima Deep Learning Long Short-Term Memory (LSTM)

LSTM (Long Short-Term Memory) adalah jenis arsitektur jaringan saraf tiruan yang termasuk dalam Recurrent Neural Network (RNN) dan dirancang untuk menangani masalah vanishing gradient dalam RNN konvensional.

a. Struktur LSTM

LSTM memiliki unit yang disebut sebagai sel memori, yang memungkinkan infomrasi untuk dipertahakan dalam jangka waktu yang lebih lama dibandingkan RNN standar. Setiap sel LSTM memiliki tiga gerbang utama:

1. Forget Gate (Gerbang Lupa, ft)
   - Memutuskan informasi mana yang harus dihapus dari sel memori.
   - Jika suatu informasi tidak relevan, akan dihapus dari memori.
   - Menggunakan fungsi sigmoid untuk menghasilkan nilai anatara 0 dan 1

$$f_t=\sigma(Wf\cdot[h_t-1,x_t]+bf)$$

a. Input Gate (Gerbang Input, i_t)
   Memutuskan informasi baru yang akan ditambahkan ke sel memori.
   Memiliki dua bagian :
   - Gerbang Input i_t : Menentukan seberapa banyak informasi baru akan dimasukkan.
   - Kandidat Memori

b. Memperbarui Sel Memori C_t : Menghasilkan informasi baru yang akan ditambahkan ke sel memori.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

c. Output Gate (Gerbang Output, o_t)

Memutuskan apa yang akan dihasilkan sebagai output dari unit LSTM.
Output dihitung menggunakan sel memori yang diperbarui.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

2. Memperbarui Sel Memori

Setelah gerbang forget dan input bekerja, sel memori diperbarui dengan :

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

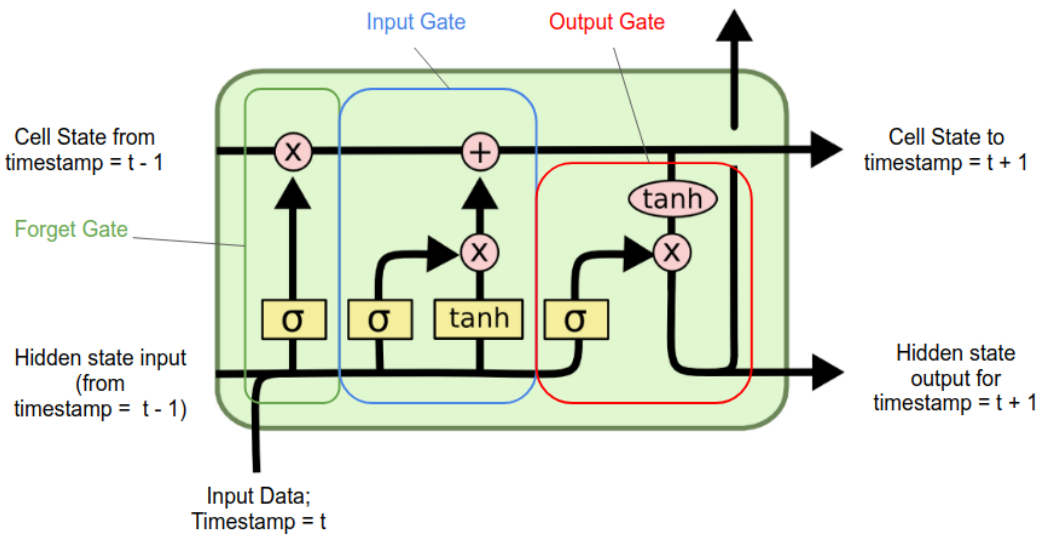
Lalu, output dari LSTM diberikan oleh:

$$h_t = o_t \odot \tanh(C_t)$$

Dimana:

- σ sigmaσ adalah fungsi aktivasi **sigmoid**.
- Tanh tanhtanh adalah fungsi aktivasi **tangens hiperbolik**.

- W dan b adalah bobot dan bias yang dipelajari oleh jaringan.



Hasil dan Pembahasan

a. Hasil Clean Ulasan

Pada tabel ulasan adalah teks yang belum di clean, tabel clean_ulasan adalah teks yang sudah di clean.

	ulasan	clean_ulasan
0	2 bulan tidak transaksi langsung di nonaktifka...	bulan tidak transaksi langsung di nonaktif ke ...
1	Bagus	bagus
2	Aplikasi tambah berat, stlah d update Dan terl...	aplikasi tambah berat stlah d update dan terla...
3	Keluar sendiri akunnya	keluar sendiri akun
4	Kenapa di pinjaman ksm, tidak muncul unit caba...	kenapa di pinjam ksm tidak muncul unit cabang ...

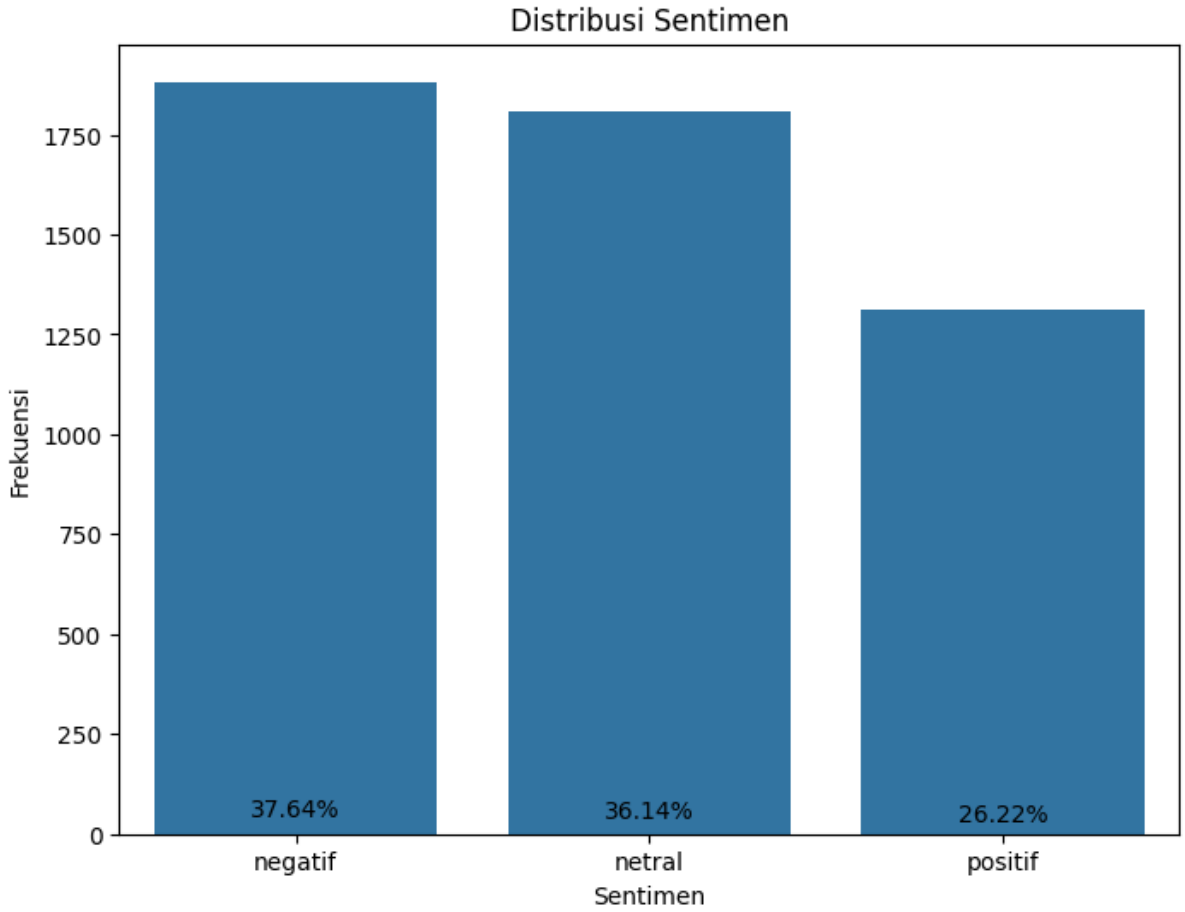
4.6 Visualisasi

a. Presentase sentimen

Setelah di labeling setiap teks menjadi positif, netral dan negatif maka tahapan ini dilakukan perhitungan jumlah pada setiap sentimen.

Sentimen	Frekuensi	Persentase
Negatif	1882	37.64
Netral	1807	36.22

Positif	1311	26.22



Jika dilihat hasil persentasenya maka sentimen terbanyak adalah sentimen negatif kedua sentimen netral ketiga sentimen positif, untuk case ini kita perlu membuat visualisasi world cloud untuk melihat frekuensi kata yang paling banyak muncul.

Word Cloud - Sentimen Positif

Word Cloud - Sentimen Netral

Word Cloud - Sentimen Negatif

- Pada sentimen Positif kata paling banyak muncul adalah **Aplikasi,Bagus,Baik,Sangat,bantu,Mantap.**
- Pada sentimen Netral kalimat yang paling banyak muncul adalah **Saya,Di,Aplikasi,Nya,Kenapa**
- Pada Sentimen Negatif kata yang paling banyak muncul adalah **Tidak,Bisa,Saya,Aplikasi,Kenapa,Susah**

4.7 Acuracy

Pada model LSTM acuracy sebesar 98% , angka ini menunjukkan accuracy yang baik dan bisa diterapkan pada analisa sentimen pada ulasan Aplikasi Livin.

4.8 Kesimpulan

- Berdasarakan hasil analisis sentimen, ulasan dengan sentimen negatif lebih dominan. Kata-kata yang sering muncul dalam ulasan negatif adalah *"tidak"* dan *"bisa"*, yang mengindikasikan bahwa banyak pengguna mengalami kendala dalam login atau kesulitan saat melakukan transaksi. Oleh karena itu, aspek yang perlu dioptimalkan adalah **kecepatan akses saat login** dan **kemudahan dalam proses transaksi**, sehingga pengalaman pengguna dapat ditingkatkan.

  Di sisi lain, pada ulasan dengan sentimen positif, kata-kata yang sering muncul adalah *"bagus"*, *"mantap"*, dan lainnya, yang menunjukkan bahwa aplikasi Livin' dinilai baik dan bermanfaat untuk melakukan transaksi.

Saran Perbaikan:

1. **Optimasi Sistem Login**
   - Meningkatkan performa server agar proses login lebih cepat dan stabil.
   - Menyediakan opsi login alternatif seperti *biometric authentication* atau *single sign-on* untuk mempercepat akses.
2. **Peningkatan Kemudahan Transaksi**
   - Memastikan tampilan antarmuka yang lebih intuitif dan ramah pengguna.
   - Menyederhanakan alur transaksi agar pengguna tidak mengalami kebingungan.
3. **Customer Support yang Lebih Responsif**

- Menyediakan layanan bantuan yang cepat dan efektif untuk pengguna yang mengalami kendala.
- Menyediakan fitur chatbot atau live chat untuk menjawab pertanyaan seputar transaksi dan login.

In [1]:
```
#install google-scraper dikarenakan Playstore Tidak ada API.
!pip install google-play-scraper
```

```
Collecting google-play-scraper
  Downloading google_play_scraper-1.2.7-py3-none-any.whl.metadata (50 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/50.2 kB ? eta -:--:--
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.2/50.2 kB 3.0 MB/s eta 0:00:00
Downloading google_play_scraper-1.2.7-py3-none-any.whl (28 kB)
Installing collected packages: google-play-scraper
Successfully installed google-play-scraper-1.2.7
```

In [2]:
```
pip install wordCloud
```

```
Requirement already satisfied: wordCloud in /usr/local/lib/python3.11/dist-packag
es (1.9.4)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.11/dist-pac
kages (from wordCloud) (2.0.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages
(from wordCloud) (11.1.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packa
ges (from wordCloud) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist
-packages (from matplotlib->wordCloud) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-pac
kages (from matplotlib->wordCloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dis
t-packages (from matplotlib->wordCloud) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dis
t-packages (from matplotlib->wordCloud) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-
packages (from matplotlib->wordCloud) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist
-packages (from matplotlib->wordCloud) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/
dist-packages (from matplotlib->wordCloud) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-package
s (from python-dateutil>=2.7->matplotlib->wordCloud) (1.17.0)
```

In [3]:
```
pip install Sastrawi
```

```
Collecting Sastrawi
  Downloading Sastrawi-1.0.1-py2.py3-none-any.whl.metadata (909 bytes)
Downloading Sastrawi-1.0.1-py2.py3-none-any.whl (209 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 209.7/209.7 kB 6.7 MB/s eta 0:00:00
Installing collected packages: Sastrawi
Successfully installed Sastrawi-1.0.1
```

In [4]:
```
#import Library
import pandas as pd
import numpy as np
import re #digunakan dalam python untuk regular expresion untuk mencari pola yan
#menumukan, memanipulasi teks
import string
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud #untuk membuat wordcloud
from google_play_scraper import reviews
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from tensorflow.keras.losses import sparse_categorical_crossentropy
```

In [5]:
```
#scraping ulasan aplikasi livin
result, _ = reviews(
    'id.bmri.livin', # ID aplikasi Livin di Google Play
    lang='id',
    country='id',
    count=5000 #mengambil 1000 ulasan
)
```

In [6]:
```
#Konversi ke DataFrame
df = pd.DataFrame(result)[['content']]
df.rename(columns={'content':'ulasan'}, inplace=True)
```

In [7]:
```
#Tampilkan Data
df.head()
```

Out[7]:

	ulasan
0	sangat membantu
1	sering matikan hp dulu.kemudiandihidupkan .bar...
2	sangat bagus dan membantu sekali
3	sangat puas dengan aplikasi ini
4	sangat membantu

In [8]:
```
#Preprocessing Data
stemmer = StemmerFactory().create_stemmer()
```

In [9]:
```
#kamus kata slang dan typo
slang_dict = {"bgt":"banget",
              "gpp": "tidak apa-apa",
              "bagusss": "bagus",
              "jd": "jadi",
              "kmu":"kamu",
              "aq":"aku",
              "gw": "saya",
              "loe": "kamu",
              "baper": "terbawa perasaan",
              "mager": "malas gerak",
              "gabut": "tidak ada kerjaan",
              "santuy": "santai",
              "gaje": "tidak jelas",
              "woles": "tenang",
              "otw": "sedang dalam perjalanan",
              "btw": "ngomong-ngomong",
              "kuy": "ayo",
              "gan": "juragan",
              "sis": "kakak perempuan",
              "sist": "kakak perempuan",
```

```
    "bro": "saudara laki-laki",
    "cmiiw": "koreksi jika saya salah",
    "asap": "secepat mungkin",
    "japri": "jalur pribadi",
    "mantul": "mantap betul",
    "rekber": "rekening bersama",
    "bocil": "bocah kecil",
    "vibes": "suasana",
    "spill": "membocorkan",
    "ghosting": "menghilang tanpa kabar",
    "flexing": "pamer",
    "cringe": "memalukan",
    "ngegas": "marah",
    "nolep": "tidak memiliki kehidupan sosial",
    "halu": "halusinasi",
    "wkwk": "tertawa",
    "lol": "tertawa terbahak-bahak",
    "brb": "akan segera kembali",
    "gk" : "tidak",
    "ga" : "tidak",
    "tdk" : "tidak",
    "mager": "malas gerak",
    "gabut": "tidak ada kerjaan",
    "santuy": "santai",
    "mrh": "marah",
    "gpp": "tidak apa-apa",
    "murahhh":"murah",

    # Typo
    "akn": "akan",
    "bnr": "benar",
    "kmn": "kemana",
    "gmna": "bagaimana",
    "mkn": "makan",
    "mnum": "minum",
    "klr": "keluar",
    "bljr": "belajar",
    "tdk": "tidak",
    "bgt": "banget",
    "dngn": "dengan",
    "sya": "saya",
    "trs": "terus",
    "bsa": "bisa",
    "skrg": "sekarang",
    "bkn": "bukan",
    "udh": "sudah",
    "sdh": "sudah",
    "trmks": "terima kasih",
    "smg": "semoga",
    "bbrp": "beberapa",
    "tp": "tapi",
    "krn": "karena",
    "td": "tadi",
    "pgn": "ingin",
    "nyebelin": "menyebalkan",
    "knp": "kenapa",
    "jd": "jadi",
    "dpt": "dapat",
    "mnrt": "menurut",
    "gk": "tidak",
```

```
                "ga": "tidak",
                "jg": "juga",
                "trs": "terus",
                "dlm": "dalam",
                "tp": "tapi",
                "smua": "semua",
                "trsbh": "tersebut",
                "ajh": "aja",
                "sj": "saja",
                "kl": "kalau",
                "trus": "terus",
                "dr": "dari"
                          }
```

In [10]:
```
#Clean teks
def clean_text(text):
  text = text.lower() #Lowercasing
  text = re.sub(r'\d+', '', text)  # Hapus angka)
  text = text.translate (str.maketrans('','', string.punctuation)) #hapus tanda
  text = ' '.join([slang_dict[word] if word in slang_dict else word for word in
  text = stemmer.stem(text) # Stemming
  return text
```

In [11]:
```
df['clean_ulasan'] = df['ulasan'].apply(clean_text)
```

In [12]:
```
df.head()
```

Out[12]:

	ulasan	clean_ulasan
0	sangat membantu	sangat bantu
1	sering matikan hp dulu.kemudiandihidupkan .bar...	sering mati hp dulukemudiandihidupkan baru bis...
2	sangat bagus dan membantu sekali	sangat bagus dan bantu sekali
3	sangat puas dengan aplikasi ini	sangat puas dengan aplikasi ini
4	sangat membantu	sangat bantu

In [13]:
```
#Labeling Sentimen Berdasarkan Kata-kata Positif,Netral dan Negatif
positive_words = ["bagus", "mantap", "cepat", "puas", "senang", "baik", "suka","
                  "lebih mudah","serba mudah","keren","menarik","membantu","prak
                  "cocok",]
netral_words = ["aku","saya","kemarin","pas","debit","kredit","atm","bank","mand
negative_words = ["buruk", "lama", "error", "tidak", "jelek", "lelet", "kecewa",
```

In [14]:
```
#membuat fungsi untuk labeling dengan Lexicon based
def sentimen_label(text):
  # sum (1 for word in text.split() if word in ..... --> code ini menjelaskan un
  pos_count = sum(1 for word in text.split() if word in positive_words)
  netral_count = sum(1 for word in text.split() if word in netral_words)
  neg_count = sum(1 for word in text.split() if word in negative_words)
  # membuat if else jika kata postif bernilai lebih besar dari kata negatif maka
  if pos_count > neg_count:
    return 'positif'
  elif pos_count < neg_count:
    return 'negatif'
```

```
  else :
    return 'netral'
```

In [16]:
```
#implementasi pada feature dengan menggunakan fungsi labeling
df ['sentimen'] = df['clean_ulasan'].apply (sentimen_label)
```

In [17]:
```
#melihat data sebanyak 19 data
df.head(20)
```

Out[17]:

	ulasan	clean_ulasan	sentimen
0	sangat membantu	sangat bantu	positif
1	sering matikan hp dulu.kemudiandihidupkan .bar...	sering mati hp dulukemudiandihidupkan baru bis...	netral
2	sangat bagus dan membantu sekali	sangat bagus dan bantu sekali	positif
3	sangat puas dengan aplikasi ini	sangat puas dengan aplikasi ini	positif
4	sangat membantu	sangat bantu	positif
5	Sangat bagus tolong dung akhir2 ini apk sering...	sangat bagus tolong dung akhir ini apk sering ...	positif
6	sangat baik	sangat baik	positif
7	Ok	ok	netral
8	Kenapa yaa klo mau transfer daftar nama si pen...	kenapa yaa klo mau transfer daftar nama si ter...	netral
9	ok	ok	netral
10	Baik untuk mempermudah nasabah tp ni troble	baik untuk mudah nasabah tapi ni troble	positif
11	sangat baek sya gunakan licin mandiri	sangat baek saya guna licin mandiri	netral
12	tdk bisa Di update	tidak bisa di update	negatif
13	Jelek ah ga bisa pinjam duit lewat online 🤣🤣🤣	jelek ah tidak bisa pinjam duit lewat online	negatif
14	Mudah dibaca	mudah baca	positif
15	bagus lebih mempermudah melakukan transaksi.	bagus lebih mudah laku transaksi	positif
16	👍		netral
17	aplikasinya sangat mempermudah,tapi klw bisa l...	aplikasi sangat mempermudahtapi klw bisa lihat...	negatif
18	Baik Baik	baik baik	positif
19	kecewa dgn aplikasi ini sering macet bahkan sa...	kecewa dgn aplikasi ini sering macet bahkan sa...	negatif

In [18]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   ulasan        5000 non-null   object
 1   clean_ulasan  5000 non-null   object
 2   sentimen      5000 non-null   object
dtypes: object(3)
memory usage: 117.3+ KB
```

In [20]:
```
#mengihung jumlah kata sentimen negatif,netral dan positif
df['sentimen'].value_counts()
```

Out[20]:

sentimen	count
negatif	1882
netral	1807
positif	1311

dtype: int64

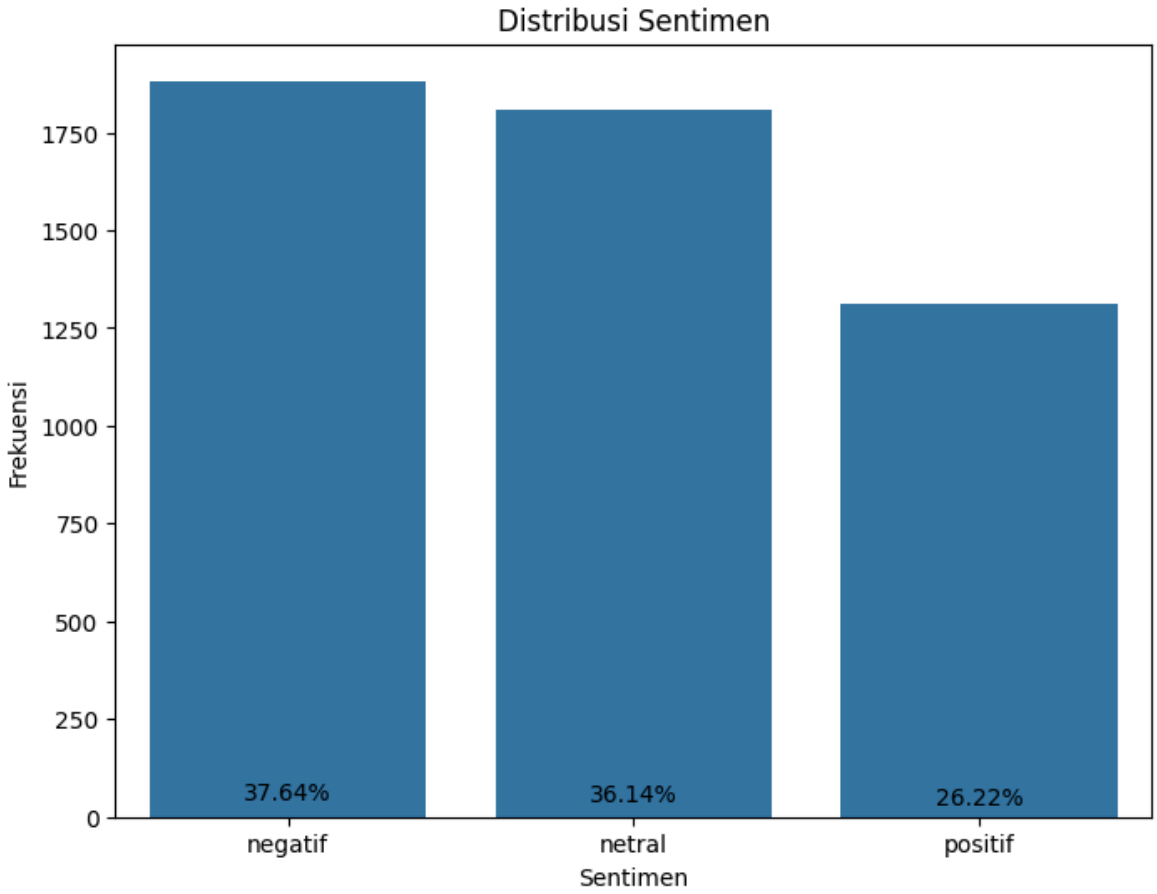
In [22]:
```
#membuat visualisasi pada sentimen
sentimen_counts = df['sentimen'].value_counts()
sentimen_percentage = sentimen_counts / len(df) * 100
sentimen_df = pd.DataFrame({'Sentimen': sentimen_counts.index, 'Frekuensi': sent
print(sentimen_df)

plt.figure(figsize=(8, 6))
sns.barplot(x='Sentimen', y='Frekuensi', data=sentimen_df)
plt.title('Distribusi Sentimen')
plt.xlabel('Sentimen')
plt.ylabel('Frekuensi')

for i, v in enumerate(sentimen_percentage.values):
    plt.text(i, v + 5, f'{v:.2f}%', ha='center')

plt.show()
```

```
  Sentimen  Frekuensi  Persentase
0  negatif       1882       37.64
1   netral       1807       36.14
2  positif       1311       26.22
```



In [23]:
```
#melihat data ke 500 sampai akhir
df.iloc[500:]
```

Out[23]:

	ulasan	clean_ulasan	sentimen
500	tampilannya bagus	tampil bagus	positif
501	Aplikasi nya sangat membantu...canggih semua n...	aplikasi nya sangat membantucanggih semua nya ...	netral
502	Sangat membantu dan mudah digunakan	sangat bantu dan mudah guna	positif
503	Alhamdulillah mempermudah saat Jauh Dari pusat...	alhamdulillah mudah saat jauh dari pusat ramai	positif
504	lancar tdk ada kendala	lancar tidak ada kendala	negatif
...	...	...	...
4995	livin tidak bisa di pakai, aplikasi ga guna	livin tidak bisa di pakai aplikasi tidak guna	negatif
4996	Kenapa tidak bisa masuk livin nya ke HP baru p...		netral
4997	Suport❤️👍	suport	netral
4998	Kenapa livin sering ngelag ya klo mau buka apl...	kenapa livin sering ngelag ya klo mau buka apl...	netral
4999	Tidak bisa buka seharian	tidak bisa buka hari	negatif

4500 rows × 3 columns

In [24]:
```
#konversi teks ke vektor (Tokenisasi & Padding)
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['clean_ulasan'])
X = tokenizer.texts_to_sequences(df['clean_ulasan'])
X = pad_sequences(X, maxlen=50) #Padding Sequences
```

In [25]:
```
# label dimasukkan ke list dan dimapping dengan loop
list_label =[]
for label in df['sentimen']:
    if label == 'positif':
        list_label.append(0)
    elif label == 'netral':
        list_label.append(1)
    else:
         list_label.append(2)
```

In [26]:
```
#konversi label ke numerik
y=np.array(list_label)
```

In [27]:
```
#Split Data Train & Test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [28]:
```
#Melatih Model LSTM
model = Sequential([
    Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=128, input_lengt
    LSTM(128, return_sequences=True),
    Dropout(0.3),
```

```
    LSTM(64),
    Dense(32,activation='relu'),
    Dropout(0.3),
    Dense(3, activation='softmax')
])
# input_dim=len(tokenizer.word_index)+1 --> jumlah kata unik(vocabulary size) da
#output_dim=128 --> setiap kata dikonversi menjadi vektor berdimensi 128
#input_length=50 --> Panjang maksimal dari setiap input teks(padding akan diguna

#fungsi LSTM pertama dengan 128 unit(neuron) pada layer utama untuk Sequence
#return_sequences = True --> mengembalikan urutan lengkap dari setiap langkah wa

#Dropout(0.3) --> fungsi : untuk mengurangi overfitting dengan  membuang 30% neu
                  #tujuan : Mencegah Model terlalu bergantung pada pola tertentu

#LSTM kedua (Layer Akhir untuk Representasi)
#LSTM(64) --> Fungsi : LSTM kedua dengan 64 unit, tanpa return_sequences karena
              #perbedaan dengan LSTM pertama : Hanya Mengembalikan keluaran tera
              #Tujuan: menangkap fitur penting dari ouput LSTM pertama untuk dik

#Dense layer (Fully Connected)
#Dense(32,activation='relu') - Fungsi : layer Fully Connected dengan 32 neuron.
                            # - Aktivasi : relu --> untuk mengangkap fitur non l
                            # - Tujuan : Memproses Informasi dari LSTM untuk di
#Output layer (Prediksi Sentimen)
# Dense(3, activation='softmax') - 3 : Jumlah neuron,sesuai dengan jumlah kelas
                                 # - activation='softmax': fungsi aktivasi softm
                                 #Kelas dengan probabilitas tertinggi akan menja
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: Us
erWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
```

Kesimpulan:

- Model ini mengambil teks, mengonversinya menjadi embedding vektor, lalu menggunakan dua LSTM layer untuk menangkap informasi sekuensial.
- Dropout digunakan untuk regularisasi agar menghindari overfitting.
- Output sigmoid digunakan untuk klasifikasi biner.

In [29]:
```
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam', metrics=[

#metode .compile() digunakan untuk mengonfigurasi model sebelum pelatihan.
# Tiga parameter utama yang digunakan:
#loss --> Fungsi kerugian (Loss Function)
#optimizer --> Algoritma optimasi
# Metrics --> Metode evaluasi kinerja model

# sparse_categorical_crosssentropy --> digunakan karena memiliki 3 label
# rumus : Loss = -Σ(y_true_i * log(y_pred_i))

#optimizer ='adam' --> untuk algoritma optimasi.

#metrics=['accuracy'] --> adalah metrik yang umum digunakan untuk mengevaluasi k
#rumus : Accuracy = (Jumlah prediksi yang benar) / (total jumlah prediksi)
```

In [30]:
```
# Training Model
history = model.fit(X_train, y_train, epochs=5, batch_size=16, validation_data=(

#epochs = 5 : jumlah epoch yang akan dijalankan selama pelatihan.satu epoch bera
#batch_size = 16 : Ukuran batch yang digunakan selam pelatihan. Batch size menen
#validation_data = (X_test, y_test) : Data validasi yang digunakan untuk memanta

#cara kerja :
#1.Inisialisasi : Model diinsiasi dengan bobot acak.
#2.iterasi Epoch : Proses pelatihan diulang sebanyak epoch yang di tentukan.
#3.Iterasi Batch : Dalam setiap epoch, data pelatiahn dibagi menjadi batch-batch
#4.Forward Pass: Model memproses setiap batch data dan menghasilkan prediksi.
#5.Hitung Loss : fungsi kerugian dihitung dnegan membandingkan prediksi model de
#6.Backward Pass : Gradien dihitung untuk setiap bobot model menggunakan backpro
#7.Perbarui Bobot : Optimizer memperbaharui bobot model berdasarkan gradien yang
#8.Evaluasi:Setelah setiap Epoch, model dievaluasi menggunakan data validasi unt
#9.Penyipanan History : Informasi tentang proses pelatihan, seperti loss dan met
```

```
Epoch 1/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 29s 92ms/step - accuracy: 0.5570 - loss: 0.8767 - va
l_accuracy: 0.9070 - val_loss: 0.2349
Epoch 2/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 39s 85ms/step - accuracy: 0.9377 - loss: 0.1668 - va
l_accuracy: 0.9700 - val_loss: 0.0834
Epoch 3/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 42s 89ms/step - accuracy: 0.9737 - loss: 0.0801 - va
l_accuracy: 0.9820 - val_loss: 0.0696
Epoch 4/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 40s 85ms/step - accuracy: 0.9888 - loss: 0.0418 - va
l_accuracy: 0.9750 - val_loss: 0.0689
Epoch 5/5
250/250 ━━━━━━━━━━━━━━━━━━━━ 42s 89ms/step - accuracy: 0.9884 - loss: 0.0413 - va
l_accuracy: 0.9840 - val_loss: 0.0391
```

In [31]:
```
# Evaluasi Model
y_pred_probs = model.predict(X_test)  # Prediksi probabilitas untuk setiap kelas
y_pred = np.argmax(y_pred_probs, axis=-1)  # Mendapatkan prediksi kelas (indeks
print(classification_report(y_test, y_pred))
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 51ms/step
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       282
           1       0.97      0.98      0.98       369
           2       0.99      0.99      0.99       349

    accuracy                           0.98      1000
   macro avg       0.98      0.98      0.98      1000
weighted avg       0.98      0.98      0.98      1000
```

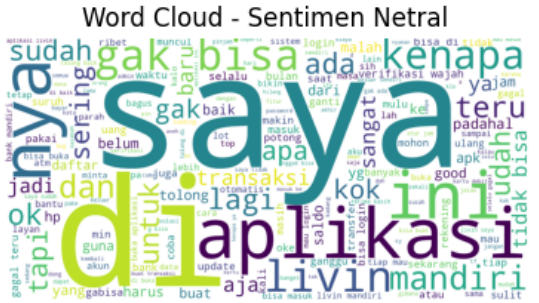
In [32]:
```
#  Visualisasi Word Cloud (Positif & Negatif)
pos_text = ' '.join(df[df['sentimen'] == 'positif']['clean_ulasan'])
net_text = ' '.join(df[df['sentimen'] == 'netral']['clean_ulasan'])
neg_text = ' '.join(df[df['sentimen'] == 'negatif']['clean_ulasan'])

wordcloud_pos = WordCloud(width=2000, height=1000, background_color='white').gen
wordcloud_net = WordCloud(width=2000, height=1000, background_color='white').gen
wordcloud_neg = WordCloud(width=2000, height=1000, background_color='black', col
```

```
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(wordcloud_pos)
plt.axis("off")
plt.title("Word Cloud - Sentimen Positif")

plt.subplot(1, 2, 2)
plt.imshow(wordcloud_net)
plt.axis("off")
plt.title("Word Cloud - Sentimen Netral")
plt.show()

plt.subplot(1, 2, 2)
plt.imshow(wordcloud_neg)
plt.axis("off")
plt.title("Word Cloud - Sentimen Negatif")
plt.show()
```



In [ ]: