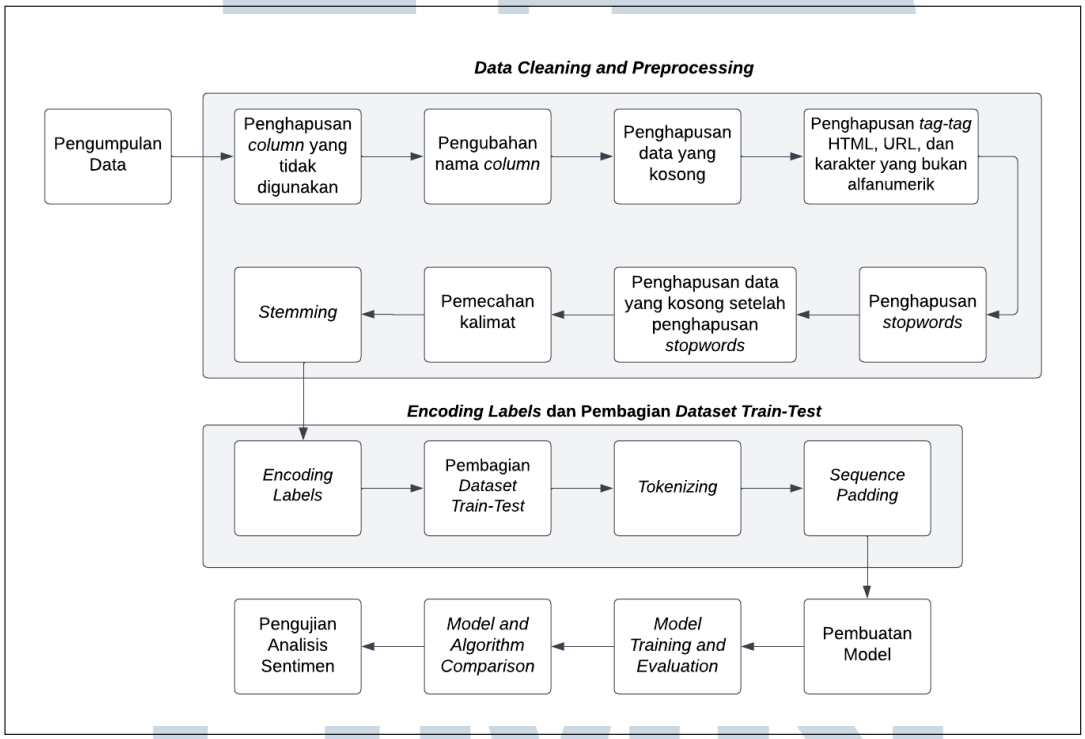
BAB 3
METODOLOGI PENELITIAN

Pada penelitian ini, terdapat beberapa tahap yang dilakukan untuk mengimplementasikan algoritma Bi-LSTM untuk analisis sentimen *review* (*feedback*) pelanggan. Pada Gambar 3.1 menunjukkan beberapa tahapan yang dilakukan pada penelitian ini.



Gambar 3.1. Tahapan-tahapan yang dilakukan pada penelitian

3.1 Pengumpulan Data

Data yang digunakan dalam penelitian ini yaitu data yang didapatkan dari *website* Mendeley Data, yaitu *Product Reviews Dataset for Emotions Classification Tasks - Indonesian*. *Dataset* tersebut terpilih karena merupakan *dataset* yang memiliki statistik label yang seimbang (*balance*). Idealnya, *dataset* harus seimbang karena *dataset* yang sangat tidak seimbang akan menyulitkan pemodelan dan akan menyebabkan keakurasian kurang tepat. Kode 3.1 menunjukkan semua *packages* dan *libraries* yang digunakan pada penelitian ini. Pada data yang akan digunakan di penelitian ini, terdapat

sebesar 42.75% data dengan label positif dan 52.24% data dengan label negatif. Total data yang digunakan adalah sebanyak 5400 data. Gambar 3.2 merupakan sampel data yang akan digunakan pada penelitian ini.

```
import re
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences
import keras
from sklearn.metrics  import classification_report
from sklearn.metrics import accuracy_score
import math
import nltk
import matplotlib.pyplot as plt
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from nltk.tokenize import WhitespaceTokenizer
from nltk.probability import FreqDist
import warnings
# nltk.download('wordnet')
# nltk.download('omw-1.4')
# pip install Sastrawi
```

Kode 3.1: *Packages* dan *libraries* yang digunakan

Library 're' (*Regular Expression*) berfungsi untuk *function-function* seperti *compare* kata, *tokenizer*, dan sebagainya. *Library* 'pandas' berfungsi untuk memanipulasi data, seperti *read* data, *plot* data, *drop* data, dan sebagainya. *Library* 'numpy' befungsi untuk *function-function* data *array*. *Library* 'sklearn' berfungsi untuk *transform* data, menunjukkan hasil akurasi, dan sebagainya. *Library* 'keras' digunakan untuk mengimplementasikan *deep learning* (*neural network*). *Library* 'math' berfungsi untuk menjalankan *task* yang berhubungan dengan matematika. *Library* 'matplotlib' digunakan untuk membuat plot data. *Library* 'Sastrawi' digunakan untuk melakukan *stemming* dalam Bahasa Indonesia. *Library* 'nltk' digunakan untuk melakukan *function-function* NLP, seperti *tokenize*, *stopwords*, dan sebagainya. *Library* 'warnings' digunakan untuk menghapus (*ignore*) *warning-warning* yang didapatkan.

```
#IMPORT DATA
datas = pd.read_csv('PRDECT-ID Dataset (1).csv')
datas
```

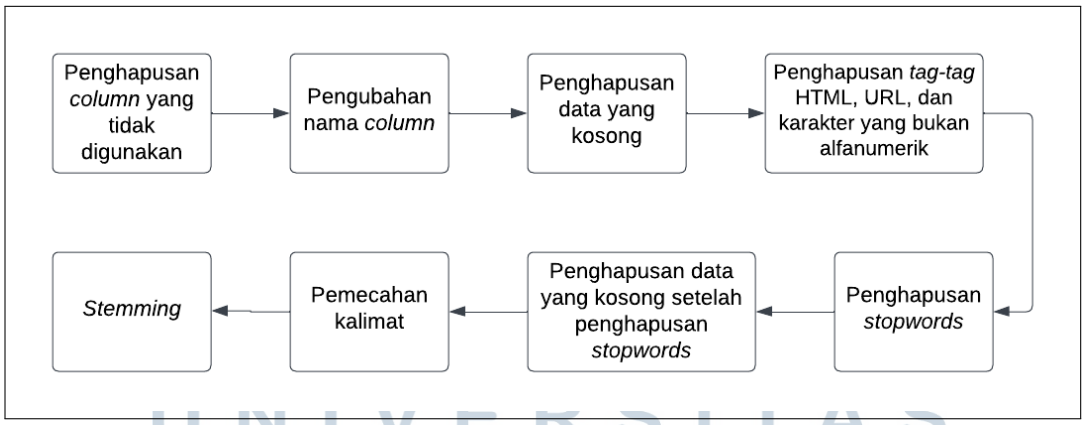
	Category	Product Name	Location	Price	Overall Rating	Number Sold	Total Review	Customer Rating	Customer Review	Sentiment	Emotion
0	Computers and Laptops	Wireless Keyboard i8 Mini TouchPad Mouse 2.4G ...	Jakarta Utara	53500	4.9	5449	2369	5	Alhamdulillah berfungsi dengan baik. Packaging...	Positive	Happy
1	Computers and Laptops	PAKET LISENSI WINDOWS 10 PRO DAN OFFICE 2019 O...	Kota Tangerang Selatan	72000	4.9	2359	1044	5	barang bagus dan respon cepat, harga bersaing ...	Positive	Happy
2	Computers and Laptops	SSD Midasforce 128 Gb - Tanpa Caddy	Jakarta Barat	213000	5.0	12300	3573	5	barang bagus, berfungsi dengan baik, seler ram...	Positive	Happy
3	Computers and Laptops	ADAPTOR CHARGER MONITOR LCD LED TV LG merek LG...	Jakarta Timur	55000	4.7	2030	672	5	bagus sesuai harapan penjual nya juga ramah. t...	Positive	Happy
4	Computers and Laptops	ADAPTOR CHARGER MONITOR LCD LED TV LG merek LG...	Jakarta Timur	55000	4.7	2030	672	5	Barang Bagus, pengemasan Aman, dapat Berfungsi...	Positive	Happy
...	...	...	...	...	...	...	...	...	...	...	...

Gambar 3.2. Data yang digunakan

Sumber: Mendeley Data

3.2 Data Cleaning and Preprocessing

Pada tahap ini, dilakukan pemrosesan data, seperti penghapusan beberapa elemen dari *dataset* yang tidak memiliki *value* atau akan mengganggu proses analisis. Gambar 3.3 menggambarkan tahapan-tahapan yang dilakukan pada proses *data cleaning* dan *preprocessing* di penelitian ini.



Gambar 3.3. Tahapan-tahapan yang dilakukan pada *data cleaning* dan *preprocessing*

Berikut ini akan dijelaskan secara lebih terperinci dan detail mengenai proses *data cleaning* dan *preprocessing* di penelitian ini. Mulai dari proses pada tahapan tersebut, sampai hasilnya.

1. Penghapusan beberapa *column* yang tidak digunakan menggunakan *function* drop. Gambar 3.4 menunjukkan data *column* yang akan digunakan pada penelitian ini, dimana data aslinya dapat dilihat pada Gambar 3.2.

```
datas.drop(datas.columns[[0, 1, 2, 3, 4, 5, 6, 7, 10]], axis=1, inplace=True)
datas.head()
```

	Customer Review	Sentiment
0	Alhamdulillah berfungsi dengan baik. Packaging...	Positive
1	barang bagus dan respon cepat, harga bersaing ...	Positive
2	barang bagus, berfungsi dengan baik, seler ram...	Positive
3	bagus sesuai harapan penjual nya juga ramah. t...	Positive
4	Barang Bagus, pengemasan Aman, dapat Berfungsi...	Positive

Gambar 3.4. *Column* data yang akan digunakan

2. Pengubahan nama *column* menggunakan *function* rename. Dapat dilihat hasilnya pada Gambar 3.5 *column* yang sebelumnya ”Customer Review” menjadi ”review” dan *column* ”Sentiment” menjadi ”sentiment”.

```
data = datas.rename(columns={"Customer Review": "review", "Sentiment": "sentiment"})
data.head()
```

	review	sentiment
0	Alhamdulillah berfungsi dengan baik. Packaging...	Positive
1	barang bagus dan respon cepat, harga bersaing ...	Positive
2	barang bagus, berfungsi dengan baik, seler ram...	Positive
3	bagus sesuai harapan penjual nya juga ramah. t...	Positive
4	Barang Bagus, pengemasan Aman, dapat Berfungsi...	Positive

Gambar 3.5. Penamaan *column* yang akan digunakan

3. Penghapusan data yang kosong menggunakan *function* dropna. Pada data yang digunakan, dapat dilihat pada Gambar 3.6 bahwa tidak ada data yang kosong.

```
#HAPUS DATA YANG NULL
data.dropna(subset=['review'], inplace=True)
data
```

	review	sentiment
0	Alhamdulillah berfungsi dengan baik. Packaging...	Positive
1	barang bagus dan respon cepat, harga bersaing ...	Positive
2	barang bagus, berfungsi dengan baik, seler ram...	Positive
3	bagus sesuai harapan penjual nya juga ramah. t...	Positive
4	Barang Bagus, pengemasan Aman, dapat Berfungsi...	Positive
...	...	...
5395	Harga bersaing, barang sesuai pesanan. Saya na...	Positive
5396	Beli ini krn Anak & Istri mau liburan di Jakar...	Positive
5397	pengemasan barang baik, kondisi barang jg utuh...	Positive
5398	Mungil tapi bekerja dng baik. Dan murahh terja...	Positive
5399	Produk sesuai deskripsi, packing aman terlindu...	Positive

5400 rows × 2 columns

Gambar 3.6. Penghapusan data yang kosong

4. Penghapusan *tag-tag* HTML, URL, dan karakter yang bukan alfabet. Hal tersebut dilakukan dengan bantuan fungsi dari *library* Regex, seperti yang telah ditampilkan di Kode 3.2.

```
#MENGHAPUS ELEMEN SEPERTI TAG HTML, URL, DAN KARAKTER NON ALFABET
def remove_tags(string):
    removelist = ""
    result = re.sub('<.*?>', '', string)          # tag HTML
    result = re.sub('https://.*', '', result)     # URL
    result = re.sub(r'[^a-zA-Z'+removelist+']', ' ', result)
      # karakter non-alfabet
    result = result.lower()     # mengubah semua kalimat menjadi menggunakan huruf kecil (lowercase)
    return result

data['review'] = data['review'].apply(lambda cw: remove_tags(cw))
data
```

Kode 3.2: Penghapusan *tag* HTML dan URL serta karakter non-alfabet

Contoh dari penggunaan Kode 3.2 yang menghapus *tag-tag* HTML, URL, dan karakter yang bukan alfabet. dapat dilihat pada Tabel 3.1.

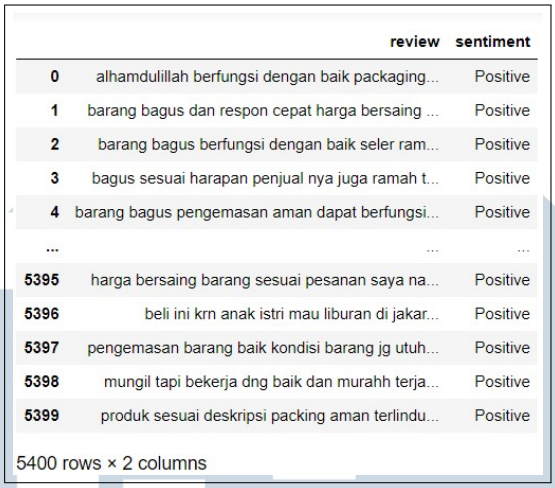
Tabel 3.1. Hasil penghapusan *tag-tag* HTML, URL, dan karakter yang bukan alfabet

Tipe Penghapusan	**Sebelum**	**Sesudah**
Tag-tag HTML	** **Penjual Ramah** ** Melayani Pembeli Dengan Sabar dan Memberikan Berbagai Saran Yang Pembeli Tidak Tahu Mantap Lah	Penjual Ramah Melayani Pembeli Dengan Sabar dan Memberikan Berbagai Saran Yang Pembeli Tidak Tahu Mantap Lah
URL	mantap kipasnya kenceng **https://www.tokopedia.com/**, barangnya berkualitas sesuai sama harga	mantap kipasnya kenceng, barangnya berkualitas sesuai sama harga
Karakter yang bukan alfabet	Kualitas barang bagus, harga relatif murah, kiriman super cepat, response penjual baik, recommended seller, bintang **5**, terima kasih yah, semoga SUKSES usaha-nya, aamiin **??????**	Kualitas barang bagus harga relatif murah kiriman super cepat response penjual baik recommended seller bintang terima kasih yah semoga SUKSES usahanya aamiin

Selain itu, dilakukan juga pengubahan semua kata menjadi huruf kecil (*lowercase*) menggunakan *function* 'lower'. Tabel 3.2 menunjukkan contoh penggunaan *function* 'lower' yang mengubah semua kata menjadi huruf kecil (*lowercase*). Gambar 3.7 menjukkan hasil akhir dari dilakukannya penghapusan *tag-tag* HTML, URL, dan karakter yang bukan alfabet, serta mengubah semua kata menjadi *lowercase*.

Tabel 3.2. Hasil pengubahan semua kata menjadi huruf kecil (*lowercase*)

Sebelum	**Sesudah**
Penjual Ramah Melayani Pembeli Dengan Sabar dan Memberikan Berbagai Saran Yang Pembeli Tidak Tahu Mantap Lah	penjual ramah melayani pembeli dengan sabar dan memberikan berbagai saran yang pembeli tidak tahu mantap lah



	review	sentiment
0	alhamdulillah berfungsi dengan baik packaging...	Positive
1	barang bagus dan respon cepat harga bersaing ...	Positive
2	barang bagus berfungsi dengan baik seler ram...	Positive
3	bagus sesuai harapan penjual nya juga ramah t...	Positive
4	barang bagus pengemasan aman dapat berfungsi...	Positive
...	...	...
5395	harga bersaing barang sesuai pesanan saya na...	Positive
5396	beli ini krn anak istri mau liburan di jakar...	Positive
5397	pengemasan barang baik kondisi barang jg utuh...	Positive
5398	mungil tapi bekerja dng baik dan murahh terja...	Positive
5399	produk sesuai deskripsi packing aman terlindu...	Positive

5400 rows × 2 columns

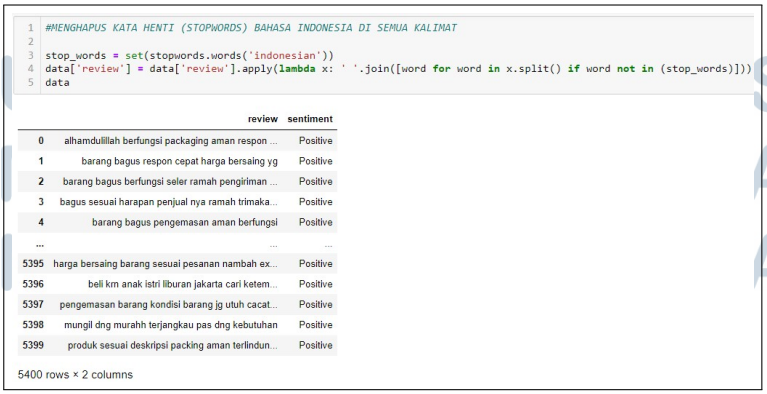
Gambar 3.7. Data yang digunakan

Sumber: Mendeley Data

5. Penghapusan kata henti (*stopwords*) dari korpus, seperti kata ’yang’, ’di’, dan sebagainya. Hal ini dilakukan untuk tidak memberikan arti yang istimewa pada sebuah kalimat. Tahap ini menggunakan *library* dari sebuah NLP yaitu NLTK. Tabel 3.3 menunjukkan proses penghapusan kata henti (*stopwords*). Pada Gambar 3.6 menunjukkan data sebelum dilakukan diproses pada tahap ini, sedangkan Gambar 3.8 menunjukkan hasil dari data yang sudah dilakukan penghapusan kata henti.

Tabel 3.3. Hasil penghapusan kata henti (*stopwords*)

Sebelum	**Sesudah**
barangnya **sangat** bagus	barangnya bagus
saya sangat kecewa **dengan** barang **ini**	kecewa barang



```
#MENGHAPUS KATA HENTI (STOPWORDS) BAHASA INDONESIA DI SEMUA KALIMAT

stop_words = set(stopwords.words('indonesian'))
data['review'] = data['review'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_words)]))
data
```

	review	sentiment
0	alhamdulillah berfungsi packaging aman respon ...	Positive
1	barang bagus respon cepat harga bersaing yg	Positive
2	barang bagus berfungsi seler ramah pengiriman ...	Positive
3	bagus sesuai harapan penjual nya ramah trimaka...	Positive
4	barang bagus pengemasan aman berfungsi	Positive
...	...	...
5395	harga bersaing barang sesuai pesanan nambah ex...	Positive
5396	beli krn anak istri liburan jakarta cari ketem...	Positive
5397	pengemasan barang kondisi barang jg utuh cacat...	Positive
5398	mungil dng murahh terjangkau pas dng kebutuhan	Positive
5399	produk sesuai deskripsi packing aman terlindun...	Positive

5400 rows × 2 columns

Gambar 3.8. Hasil penghapusan kata henti (*stopwords*) Bahasa Indonesia

6. Setelah dilakukan (*stopwords*), terdapat kemungkinan ada data review yang kosong. Maka dari itu, dilakukan penghapusan data yang kosong setelah dilakukannya penghapusan kata henti (*stopwords*) menggunakan *function* drop. Gambar 3.9 menunjukkan hasil dari penghapusan data yang kosong setelah dilakukan penghapusan *stopwords* Bahasa Indonesia.

```
#MENGHAPUS DATA YANG NULL SETELAH DILAKUKAN PENGHAPUSAN STOPWORDS
data.drop(data[data['review'].str.strip() == ''].index, inplace=True)
data
```

	review	sentiment
0	alhamdulillah berfungsi packaging aman respon ...	Positive
1	barang bagus respon cepat harga bersaing yg	Positive
2	barang bagus berfungsi seler ramah pengiriman ...	Positive
3	bagus sesuai harapan penjual nya ramah trimaka...	Positive
4	barang bagus pengemasan aman berfungsi	Positive
...	...	...
5395	harga bersaing barang sesuai pesanan nambah ex...	Positive
5396	beli krn anak istri liburan jakarta cari ketem...	Positive
5397	pengemasan barang kondisi barang jg utuh cacat...	Positive
5398	mungil dng murahh terjangkau pas dng kebutuhan	Positive
5399	produk sesuai deskripsi packing aman terlindun...	Positive

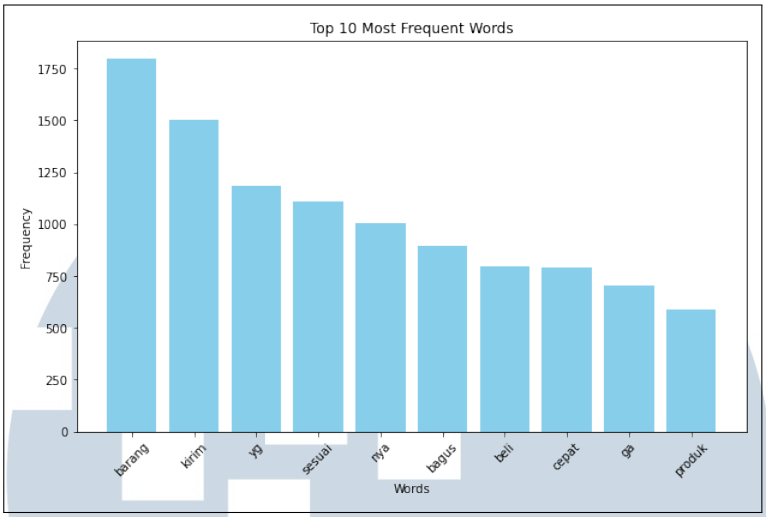
5398 rows × 2 columns

Gambar 3.9. Hasil dari penghapusan data kosong setelah dilakukan penghapusan *stopwords*

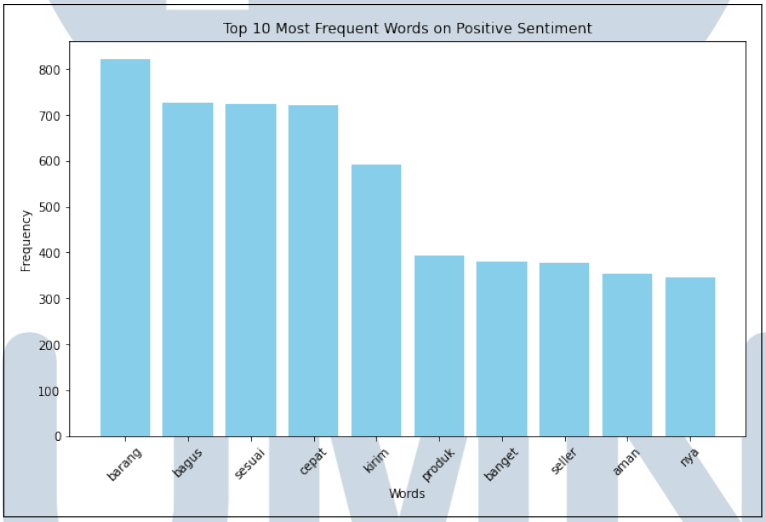
7. Pemecahan kalimat menjadi per kata dengan menggunakan *function* dari library NLTK yaitu WhitespaceTokenizer(). Hal ini dilakukan untuk memudahkan tahapan proses selanjutnya, yaitu proses *stemming*. Tabel 3.4 menunjukkan contoh dari proses pemecahan kalimat menjadi per kata menggunakan WhitespaceTokenizer(). Gambar 3.10 menunjukkan *ranking* 10 besar dari semua kata-kata di berbagai label. Gambar 3.11 menunjukkan *ranking* 10 besar dari semua kata-kata di label positif. Gambar 3.12 menunjukkan *ranking* 10 besar dari semua kata-kata di label negatif.

Tabel 3.4. Hasil pemecahan kalimat menjadi per kata

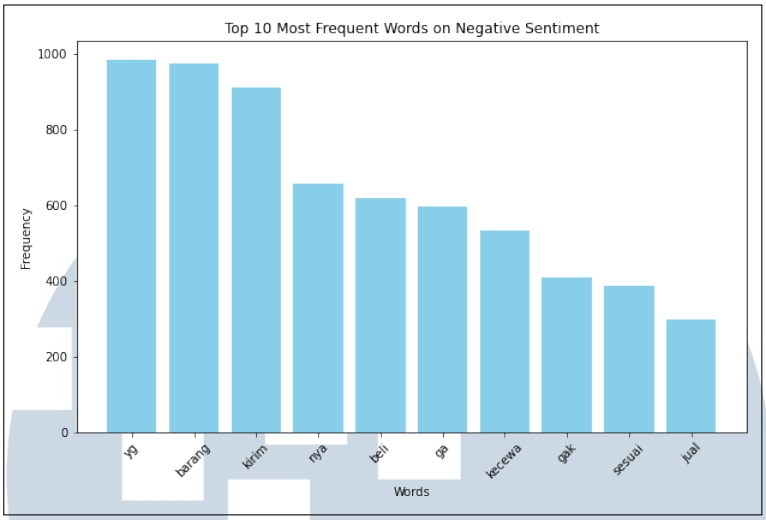
Sebelum	**Sesudah**
barangnya sangat bagus	{barangnya}, {sangat}, {bagus}
saya sangat kecewa dengan barang ini	{saya}, {sangat}, {kecewa}, {dengan}, {barang}, {ini}



Gambar 3.10. *Top* 10 kata yang sering disebutkan di semua label



Gambar 3.11. *Top* 10 kata yang sering disebutkan di label positif



Gambar 3.12. *Top* 10 kata yang sering disebutkan di label negatif

8. Proses *stemming*, yaitu teknik yang berfungsi untuk mengambil akar dari kata, atau biasa disebut dengan lema. Hal ini dilakukan untuk membantu mengurangi komputasi yang tidak diperlukan dalam mencoba menguraikan seluruh kata, yang dimana arti dari sebagian besar kata diekspresikan dengan baik oleh lema-lema yang terpisah. Tahap ini menggunakan *library* dari Sastrawi.Stemmer.StemmerFactory. Tabel 3.5 menunjukkan contoh proses *stemming*. Gambar 3.13 menunjukkan hasil setelah dilakukannya *stemming*.

Tabel 3.5. Hasil pengubahan semua kata menjadi huruf kecil (*lowercase*)

Sebelum	**Sesudah**
barang bagus **berfungsi** seler ramah **pengiriman** cepat	barang bagus fungsi seler ramah kirim cepat
barang mengecewakan	barang kecewa

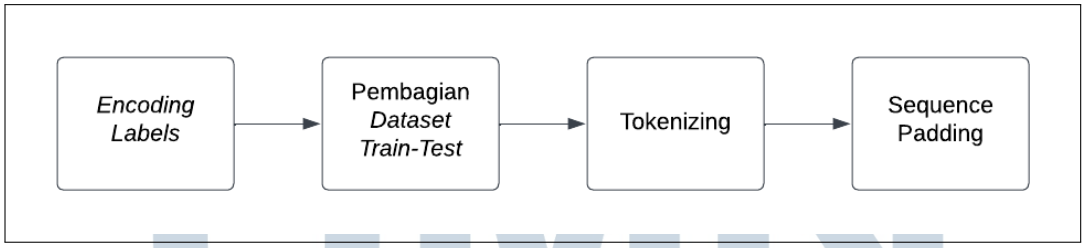
	review	sentiment
0	alhamdulillah fungsi packaging aman respon cep...	Positive
1	barang bagus respon cepat harga saing yg	Positive
2	barang bagus fungsi seler ramah kirim cepat	Positive
3	bagus sesuai harap jual nya ramah trimakasih l...	Positive
4	barang bagus emas aman fungsi	Positive
...	...	...
5395	harga saing barang sesuai pesan nambah extra p...	Positive
5396	beli krn anak istri libur jakarta cari ketemu ...	Positive
5397	emas barang kondisi barang jg utuh cacat fungs...	Positive
5398	mungil dng murahh jangkau pas dng butuh	Positive
5399	produk sesuai deskripsi packing aman lindung k...	Positive

5398 rows × 2 columns

Gambar 3.13. Hasil dari dilakukannya *stemming*

3.3 Encoding Labels dan Pembagian Dataset Train-Test

Setelah dilakukan *data cleaning* dan *preprocessing*, tahapan selanjutnya dapat dilihat pada Gambar 3.14. Dapat dilihat bahwa pada tahap ini akan dilakukan *encoding labels* dan pembagian *dataset train-test*. Lalu akan dilanjutkan dengan proses *tokenizing* dan *sequences padding*.



Gambar 3.14. Tahapan-tahapan yang dilakukan pada *encoding labels* dan pembagian *dataset train-test*

Encoding merupakan metode pengubahan sebuah *value* satu menjadi *value* yang lain, kebanyakan dari tipe *string* menjadi angka. Pada tahap kesempatan ini dilakukan dengan cara mengubah label 'positive' dan 'negative' menjadi angka '1' dan '0'. Hal ini dilakukan dengan *function* LabelEncoder() dari *library* sklearn.preprocessing seperti yang dapat dilihat pada potongan Kode 3.3.

```
#PERSIAPAN DATA

#EXTRACTING DATA
reviews = data['review'].values
```

```
labels = data['sentiment'].values

#ENCODING KATEGORIKAL LABEL
encoder = LabelEncoder()
encoded_labels = encoder.fit_transform(labels)
```

Kode 3.3: Melakukan *encoding* kategorikal label

Setelah itu, dilakukan pembagian *dataset* menjadi bagian *train* dan test sebesar 80% dan 20% [21]. Hal ini dilakukan dengan menggunakan *function* train_test_split dari *library* sklearn.model_selection yang dapat dilihat pada potongan Kode 3.4. Setelah dilakukan pembagian *train* dan *test*, dilakukan *tokenizing* dan *sequence padding* seperti pada potongan Kode 3.5.

```
#PEMBAGIAN DATA TRAINING DAN TESTING (80:20)
train_sentences, test_sentences, train_labels, test_labels =
    train_test_split(reviews, encoded_labels, stratify =
    encoded_labels)
```

Kode 3.4: Pembagian data

Terdapat beberapa parameter yang digunakan pada penelitian ini, yaitu `vocab_size`, `oov_tok`, `embedding_dim`, `max_length`, `padding_type`, dan `trunc_type`. Parameter `vocab_size` merupakan parameter yang menyatakan banyaknya kamus yang ingin dibuat. Hal tersebut berdasarkan total kata-kata unik ketika dilakukan *tokenize*. Parameter `oov_tok` atau yang memiliki kepanjangan *Out-of-Vocabulary* mewakili kata-kata yang tidak ditemukan dalam kosakata dan digantikan selama *tokenize*. Parameter `embedding_dim` merupakan dimensi dari *embedding space* pada kata. Hal ini menentukan ukuran representasi vektor untuk kata-kata. Parameter `max_length` merupakan parameter yang menentukan maksimal panjang suatu kalimat. Parameter `padding_type` merupakan parameter yang menentukan *padding* di suatu kalimat, pada penelitian ini digunakan tipe *padding* 'post', dimana *padding* ditambahkan setelah akhir kalimat. Sedangkan parameter `trunc_type` merupakan parameter yang digunakan untuk mengurangi jumlah kata pada suatu kalimat apabila kalimat tersebut melebihi jumlah parameter `max_length`. Pada penelitian ini digunakan tipe *trunc* 'post', yang dimana kata yang akan dipotong berada di akhir kalimat.

```
#PARAMETER DARI MODEL
vocab_size = 3000 # KURANG LEBIH 4 - 5 KALI DARI TOTAL STOPWORDS
oov_tok = '' #OUT-OF-VOCABULARY
embedding_dim = 100
max_length = 200 #MAKSIMAL DARI PANJANG KALIMAT
```

```
padding_type='post'
trunc_type='post'
#TOKENIZE KALIMAT YANG AKAN DIGUNAKAN
tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(train_sentences)
word_index = tokenizer.word_index
#MEMASUKKAN DATA TRAINING KE DALAM TRAIN_SEQUENCES (SEQUENCES) DAN
    TRAIN_PADDED (PAD SEQUENCES)
train_sequences = tokenizer.texts_to_sequences(train_sentences)
train_padded = pad_sequences(train_sequences, padding='post',
    maxlen=max_length)
#MEMASUKKAN DATA TESTING KE DALAM TEST_SEQUENCES (SEQUENCES) DAN
    TEST_PADDED (PAD SEQUENCES)
test_sequences = tokenizer.texts_to_sequences(test_sentences)
test_padded = pad_sequences(test_sequences, padding='post', maxlen
    =max_length)
```

Kode 3.5: Persiapan data sebelum dimasukkan ke model

Setelah parameter-parameter tersebut ditentukan, dilakukan *tokenize* pada kata dan diubah menjadi urutan dari *integers*, yang dimana menggunakan parameter `vocab_size` dan `oov_tok`. *Tokenizer* ini membangun kosakata berdasarkan frekuensi kata dalam data *training* ('training_sentences'). Setelah itu digunakan *function* texts_to_sequences, yang dimana metode ini mengkonversi kalimat di data *training* dan *testing* menjadi urutan *integers* menggunakan *tokenizer*. Lalu digunakan juga *function* pad_sequences, yang menambahkan *padding* untuk menyesuaikan panjang dari setiap kalimat pada data *training* dan *testing*.

3.3.1 Tokenizing

Tokenizing merupakan metode pembagian kalimat ke dalam kata-kata dan membuat kamus dari semua kata unik yang ditemukan dan semua kata tersebut diberikan *integer* yang unik juga. Setiap kalimat diubah menjadi sebuah *array* dari *integer* yang mewakilkan semua kata yang terdapat di dalamnya. Pada penelitian ini, digunakan API *tokenizer* dari *library* Keras. Tabel 3.6 menunjukkan contoh proses *tokenizing*.

Tabel 3.6. Proses *Tokenizing*

Review	**Hasil Tokenizing**
barangnya bagus	kamus[0]: {barangnya}, kamus[1]: {bagus}
kecewa barang	kamus[2]: {kecewa}, kamus[3]: {barang}

3.3.2 Sequence Padding

Pada tahap ini, setiap *array* mewakili setiap kalimat yang ada di dalam *dataset* diisi dengan nol dari akhir kalimat untuk membuat ukuran *array* menjadi 200 dan membuat semua kalimat memiliki panjang yang sama. Tabel 3.7 menunjukkan contoh proses dari *sequence padding* dengan maksimal 200 *array* dan metode 'post'.

Tabel 3.7. Proses *Sequence Padding*

Review	**Hasil Sequence Padding**
barangnya bagus	[{barangnya}, {bagus}, {0}, {0}, ..., {0}]
kecewa barang	[{kecewa}, {barang}, {0}, {0}, ..., {0}]

3.4 Pembuatan Model

Pada tahap ini, model dibuat dengan *library* Keras dan menggunakan metode Bi-LSTM yang akan mengeluarkan *output* dari probablitias dari kalimat yang positif jika label adalah '1'. Pada model ini akan dilakukan kompilasi menggunakan *binary cross-entropy loss* dan *optimizer* Adam. Hal ini dikarenakan pada data yang ada terdapat klasifikasi *binary* dan *optimizer* Adam menggunakan *stochastic gradient descent* untuk melakukan *training* model *deep learning*, serta untuk melakukan komparasi dari setiap *probabilities* yang diprediksi menjadi kelas label (0 atau 1). Keakurasian digunakan sebagai *metric* utama.

3.5 Model Training and Evaluation

Model di penelitian ini di *training* dengan menggunakan lima (5) *epoch*. Setelah itu, model dievaluasi dengan cara dihitung keakurasiannya. Keakurasiannya dihitung dengan cara membagi jumlah prediksi yang benar dengan jumlah total prediksi.

3.6 Model and Algorithm Comparison

Terdapat beberapa model yang dicoba pada penelitian ini, baik itu model yang menggunakan data traning dan testing dengan pembagian 70 dan 30, model yang menggunakan *epoch* enam (6), sepuluh (10), dan lima belas (15), model yang menggunakan unit LSTM sebesar 32 dan 128, serta model yang tidak menggunakan aktivasi relu (*Rectified Linear Unit*). Penelitian ini juga membandingkan algoritma Bi-LSTM dengan algoritma Uni-LSTM, GRU, CNN, dan *Simple*-RNN, yang dimana sebagian besar algoritma tersebut merupakan algoritma yang digunakan untuk pada perbandingan di penelitian terdahulu, yaitu penelitian [9], [10], dan [11].

3.7 Pengujian Analisis Sentimen

Pengujian ini dilakukan dengan model yang sama dengan yang sebelumnya dibuat dengan Bi-LSTM. Disini pengguna dapat melakukan *input string* berupa kalimat dan dapat menginput *file* juga. *Output* yang dikeluarkan untuk pengguna yang melakukan *input* kalimat adalah hasil prediksi sentimen ('Positive' atau 'Negative'), sedangkan untuk pengguna yang melakukan *input file* akan mendapatkan *output* berupa diagram yang menggambarkan persentase dari total prediksi sentimen yang 'Positive' dan 'Negative', serta tren analisis sentimen pada periode tertentu, yaitu per bulan. Pengguna yang melakukan *input file* akan mendapatkan *file* hasil prediksi tersebut beserta tanggalnya. Data pengujian yang digunakan untuk inputan *string* adalah inputan secara manual, sedangkan data pengujian yang digunakan untuk inputan *file* adalah data ulasan restoran dari Kaggle yang diberikan tanggal secara manual.

3.8 Spesifikasi Sistem

Pada penelitian ini, digunakan satu perangkat keras (*hardware*), yaitu laptop. Lalu digunakan juga satu sistem lunak (*software*) utama untuk membantu proses analisis sentimen, yaitu Jupyter Notebook.

3.8.1 Laptop

Hardware yang digunakan pada penelitian ini adalah laptop dengan *merk* Dell, yang memiliki spesifikasi sebagai berikut.

- Sistem operasi: Windows 10 Pro 64-bit (10.0, Build 19045)
- *Language*: English
- *System Manufacturer*: Dell Inc.
- *System Model*: Latitude E6320
- BIOS: Default System BIOS
- *Processor*: Intel(R) Core(TM) i5-2520M CPU @2.50GHz (4CPU), 2.5GHz
- *Memory*: 8192MB RAM
- DirectX *Version*: DirectX 12

3.8.2 Jupyter Notebook

Software yang dibutuhkan adalah Anaconda (Jupyter Notebook). Jupyter Notebook adalah singkatan dari tiga bahasa pemrograman yaitu Ju (Julia), Py (Python), dan R [22]. Jupyter Notebook merupakan *tools* yang populer digunakan untuk melakukan pengolahan data bagi seorang *data scientist* yang memungkinkan untuk mengintegrasikan antara kode dengan *output* di dalam satu dokumen secara interaktif yang berisi *live code*, persamaan, visualisasi dan teks naratif yang kaya [23]. Kemudahan aspek penulisan dan berbagi teks maupun *code* dalam aplikasi ini juga membuatnya cocok digunakan untuk berkolaborasi. Pada penelitian ini digunakan versi notebook 6.4.8 dan python 3.9.12.