Analisis Klasifikasi Data
Pada Model N-Gram Dan Neural Based

Dipersiapkan oleh

Kurniadi Ahmad Wijaya (1301194024)
Priyan Fadhil Supriyadi (1301190442)

Program Studi S1 Informatika – Fakultas Informatika
Universitas Telkom
Jalan Telekomunikasi Terusan Buah Batu,
Bandung, Indonesia

Eksperimen

1. Dataset Yang Digunakan

Data yang penulis gunakan pada tugas eksperimen ini adalah dataset analisis sentimen twitter [1]. Dataset ini adalah kumpulan data analisis sentimen tingkat entitas dari twitter. Terdapat 4 kolom dalam dataset ini yaitu Tweet ID, Entity, Sentiment, dan Tweet Content. Ada tiga kelas dalam dataset ini yang terdapat dalam kolom sentimen yaitu: Positif, Negatif dan Netral. Penulis menganggap pesan yang tidak relevan dengan entitas (yaitu Tidak relevan) sebagai Netral. Dataset yang digunakan mengandung 69491 nilai unik, dan memiliki 4 kategori sentimen pada dataset ini yang masing-masing diberikan label angka dari 0 sampai dengan 3 yaitu 0 sebagai sentimen negatif, 1 sebagai sentimen netral, 2 sebagai sentimen positif, dan 3 sebagai sentimen yang tidak relevan. Dataset tersebut dibagi menjadi 2 bagian yaitu data train yang akan digunakan untuk melatih model yang akan digunakan dan data test yang akan digunakan untuk melakukan evaluasi.

Adapun tabel dari data train yang penulis gunakan adalah sebagai berikut:

	text	sentiment
12459	Microsoft start posting Cortana for Mobile, In...	1
11763	<unk>	1
70303	5 Games I Love.. 1. Call of Duty Modern Warfar...	2
82017	New new GPU coming in soon (1660), soon afte...	2
23872	you love	2

Sedangkan tabel dari data test yang penulis gunakan adalah sebagai berikut:

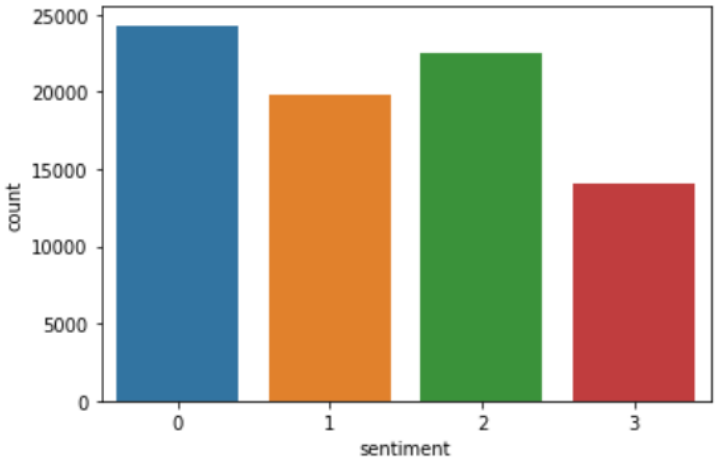
	text	sentiment
803	@googlechrome Every time I open Google Keep, i...	0
689	Ahh man, I have zero friends on\nTwitter so I ...	3
848	"you're still not diamond yet?" #ApexLegends #...	1
53	@BeverlyCitizen Ronald Bellanti is a resident ...	2
831	Today's the day that early access to Black Ops...	1

Kemudian penulis melakukan preprocessing untuk dataset yang akan digunakan sebelum melatih model. Pre-processing yang penulis lakukan diantaranya adalah mengubah setiap kata menjadi lowercase, menghapus link Dengan Pattern http/https dan www, menghapus tag HTML, menghapus karakter selain huruf a-z dan A-Z, mengganti baris baru (enter) dengan spasi, menghapus spasi yang lebih dari satu, memisahkan kata singkatan, menghapus kata yang mengandung judul topik dan kata yang terdapat pada stopwords bahasa inggris NLTK, dan melakukan lematisasi yang merupakan teknik untuk mereduksi kata menjadi Lemma atau bentuk dasar dari

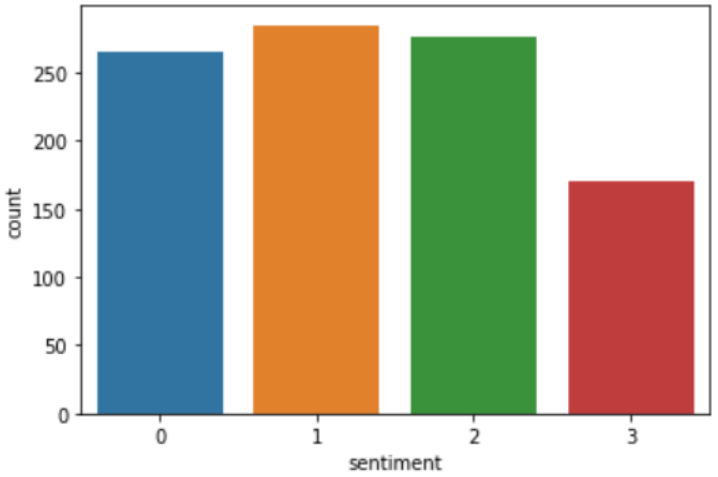
sebuah kata. Selain itu penulis juga menghapus data null pada dataset. Berikut adalah algoritma yang diimplementasikan pada kode python,

```
def preprocessing(text):
    text = str(text)
    # Mengubah setiap kata menjadi lowercase
    text = text.lower()

    # Menghapus Link Dengan Pattern http/https dan www
    text = re.sub(r'http\S+', '', text)
    text = re.sub('(@\w+|#\w+)', '', text)

    # Menghapus Tag HTML
    text = re.sub('<.*?>', '', text)

    # Menghapus Karakter Selain Huruf a-z dan A-Z
    text = re.sub('[^a-zA-Z]', ' ', text)

    # Mengganti baris baru (enter) dengan spasi
    text = re.sub("\n", " ", text)

    # Menghapus Spasi Yang Lebih Dari Satu
    text = re.sub('(s{2,})', ' ', text)

    # Memisahkan Kata Singkatan (Abbreviaton). Contoh won't -> will not
    text = re.sub(r"n\'t", " not", text)
    text = re.sub(r"\'re", " are", text)
    text = re.sub(r"\'s", " is", text)
    text = re.sub(r"\'d", " would", text)
    text = re.sub(r"\'ll", " will", text)
    text = re.sub(r"\'t", " not", text)
    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"\'m", " am", text)

    # Menghapus kata yang mengandung judul topik
    # dan kata yang terdapat pada stopwords bahasa inggris NLTK
    final_text = ''
    words = word_tokenize(text)

    # Lematisasi adalah teknik  untuk mereduksi kata menjadi Lemma atau
    # bentuk dasar dari sebuah kata.
    # Contoh:  better -> good
    for w in words:
        final_text += lemmatizer.lemmatize(w) + ' '

    # Mengembalikan Hasil Tokenizing Text
    return final_text.rstrip()
```

Adapun visualisasi menggunakan bar plot terhadap penyebaran 4 kategori sentimen yang ada pada data train adalah sebagai berikut :



Sedangkan visualisasi menggunakan bar plot terhadap penyebaran 4 kategori sentimen yang ada pada data test adalah sebagai berikut :



2. Klasifikasi N-Gram Cepet

Proses klasifikasi pada tugas eksperimen ini dilakukan dengan menggunakan teknik n-gram yang melatih model dengan rentang n-gram yang berbeda. Penulis telah menggunakan kerangka fitur bag-of-word (kumpulan kata-kata) untuk mengimplementasikan algoritma machine learning [2].

Untuk studi kasus ini, teks akan diubah menjadi model tas-kata-kata dengan objek CountVectorizer di modul sklearn sebelum digunakan untuk melatih pengklasifikasi machine learning. CountVectorizer adalah alat hebat yang disediakan oleh perpustakaan scikit-learn dengan Python [3]. CountVectorizer digunakan untuk mengubah teks yang diberikan menjadi vektor berdasarkan frekuensi (jumlah) setiap kata yang muncul di seluruh teks. Berikut merupakan beberapa contoh teks dari dokumen (masing-masing sebagai elemen list):

document = [“One Guy helps Two Guys”, “Two Guys help Four Guys”, “Each Guy helps many other Guys at Gym.”]

CountVectorizer membuat matriks di mana setiap kata unik diwakili oleh kolom matriks, dan setiap sampel teks dari dokumen adalah baris dalam matriks. Nilai setiap sel tidak lain adalah jumlah kata dalam sampel teks tertentu. Hal ini dapat divisualisasikan dengan tabel sebagai berikut:

	at	each	four	guy	guys	gym	help	helps	many	one	other	two
document[0]	0	0	0	1	1	0	0	1	0	0	0	1
document[1]	0	0	1	0	2	0	1	0	0	0	0	1
document[2]	1	1	0	1	1	1	0	1	1	0	1	0

Key Observations:

1. Ada 12 kata unik dalam document, direpresentasikan sebagai kolom tabel.
2. Ada 3 sampel teks dalam document, masing-masing direpresentasikan sebagai baris tabel.
3. Setiap sel berisi nomor, yang mewakili jumlah kata dalam teks tertentu.
4. Semua kata telah diubah menjadi huruf kecil.
5. Kata-kata dalam kolom telah diatur menurut abjad.

Di dalam CountVectorizer, kata-kata ini tidak disimpan sebagai string. Sebaliknya, mereka diberi nilai indeks tertentu. Dalam hal ini, 'at' akan memiliki indeks 0, 'each' akan memiliki indeks 1, 'four' akan memiliki indeks 2 dan seterusnya.

Model N-Gram adalah metode untuk memeriksa 'n' kata secara terus menerus dari urutan teks atau ucapan yang diberikan. Model ini membantu untuk memprediksi item berikutnya secara berurutan. Dalam analisis sentimen, model n-gram membantu menganalisis sentimen teks. Unigram mengacu pada n-gram ukuran 1 sedangkan bigram mengacu pada n-gram ukuran 2. N-gram yang lebih tinggi mengacu pada empat gram, lima gram, dan seterusnya. Metode n-gram dapat dijelaskan dengan menggunakan contoh berikut:

Sebuah contoh dari kalimat seperti “The game is not a good one”.

- Unigram: “‘The’,‘game’,‘is’, ‘not’, ‘a’, ‘good’,‘one’”.
- Bigram: “‘The game’,‘game is’, ‘is not’, ‘not a’, ‘a good’, ‘good one’”.

Model n-gram yang penulis latih yaitu model dengan rentang n-gram yang berbeda dimana N yang digunakan mempunyai rentang dari 1 sampai dengan 7.

Penulis menggunakan pengklasifikasi Naive Bayes untuk model multinomial karena Pengklasifikasi Naive Bayes multinomial cocok untuk klasifikasi dengan fitur diskrit. Distribusi multinomial biasanya membutuhkan jumlah fitur integer. Metode

supervised learning ini merupakan metode pembelajaran probabilitas. Metode Naive Bayes adalah tools yang kuat untuk menganalisis input teks dan memecahkan masalah dengan banyak kelas. Teorema Naive Bayes didasarkan pada teorema Bayes. Teorema Bayes, yang dikembangkan oleh Thomas Bayes, memperkirakan kemungkinan terjadinya berdasarkan pengetahuan sebelumnya tentang kondisi peristiwa.

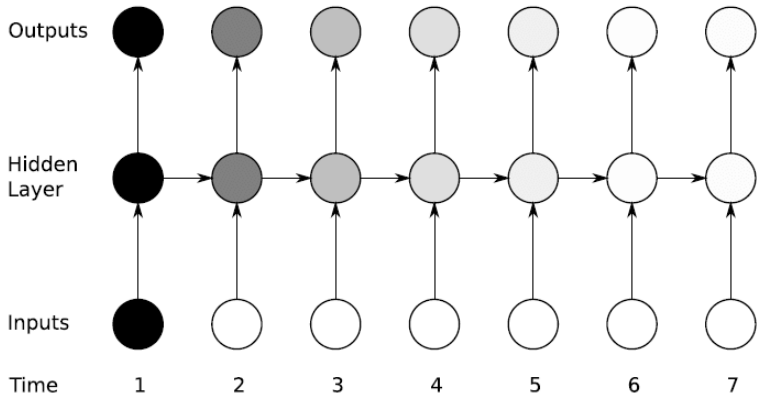
Adapun algoritma yang kami implementasikan di kode python adalah sebagai berikut:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer

# melatih model dengan rentang n-gram yang berbeda
for N in range(1, 8):

    # Mengubah data train menjadi representasi N-Gram sesuai perulangan
    cv = CountVectorizer(analyzer = 'word', ngram_range=(1,N))
    X_train_cv = cv.fit_transform(df_train['text_clean'])
    X_test_cv = cv.transform(df_test['text_clean'])

    # melatih model dan menghasilkan prediksi
    clf = MultinomialNB()
    clf.fit(X_train_cv, df_train['sentiment'])
    y_pred = clf.predict(X_test_cv)

    print(f'{N}-Gram\n')

    cm = confusion_matrix(y_pred, df_test['sentiment'])
    show_confusion_matrix(cm)
    print('')

    print(classification_report(y_pred, df_test['sentiment'],
                                target_names=["Negative", "Neutral", "Positive", "Irrelevant"]))
    print('')
```

3. Klasifikasi Neural-Based

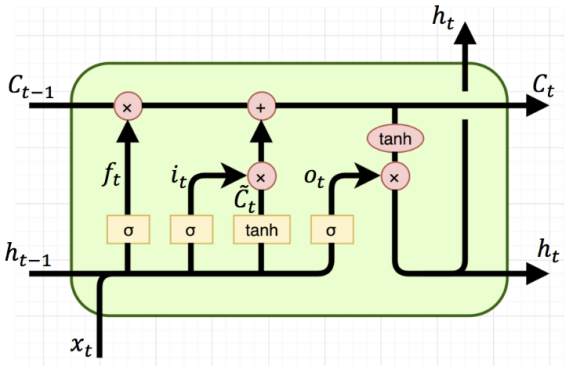
- Arsitektur Model Neural Based

Deep Learning adalah seperangkat algoritma dan teknik yang terinspirasi oleh cara kerja otak manusia, yang disebut jaringan saraf. Arsitektur deep learning menawarkan manfaat besar untuk klasifikasi teks karena kinerjanya dengan akurasi super tinggi dengan rekayasa dan komputasi tingkat rendah. Terdapat 2 jenis model arsitektur deep learning untuk klasifikasi teks diantaranya CNN (Convolutional Neural Networks) and Recurrent Neural Network (RNN).



Pada tugas eksperimen 1 ini akan digunakan model arsitektur RNN yaitu LSTM (Long Short Term Memory). Model berbasis RNN melihat teks sebagai urutan kata, dan dimaksudkan untuk menangkap dependensi kata dan teks struktur. Namun, model dasar RNN tidak berkinerja baik, dan sering kali berkinerja buruk di feed-forward neural network. Long Short-Term Memory (LSTM) adalah arsitektur yang paling populer yang dirancang untuk menghilangkan ketergantungan jangka panjang dengan lebih baik. LSTM mengatasi *vanishing gradient* yang dialami oleh RNN murni dengan memperkenalkan sel memori untuk mengingat nilai selama waktu yang sewenang-wenang interval, dan tiga gerbang (gerbang input, gerbang keluaran, gerbang lupa) untuk mengatur arus informasi masuk dan keluar sel neural network.

- Klasifikasi Dengan LSTM (Long Short Term Memory)



FORGET GATE

Gerbang pertama dalam LSTM disebut dengan forget gate. Mudahnya, gerbang ini bertugas untuk melupakan beberapa informasi yang tidak relevan dan sudah tidak diperlukan oleh sebuah sistem sehingga LSTM dapat menyajikan kumpulan informasi yang lengkap, tetapi tetap aktual sesuai dengan kebutuhan.

INPUT GATE

Berikutnya, ada gerbang kedua, yakni input gate yang bertugas untuk memasukkan informasi yang berguna untuk mendukung keakuratan data. Tugas input gate adalah untuk menambahkan informasi yang sebelumnya telah diseleksi terlebih dahulu melalui gerbang forget gate. Gerbang ini tidak dimiliki oleh RNN yang hanya memungkinkan satu input data untuk satu output data.

Dalam input gate kemudian dikenal istilah input modulation gate yang sering tidak ditulis dalam beberapa ulasan tentang LSTM. Sesuai namanya, input modulation gate berfungsi untuk memodulasi informasi yang ada, sehingga dapat mengurangi kecepatan konvergensi dari data zero-mean.

OUTPUT GATE

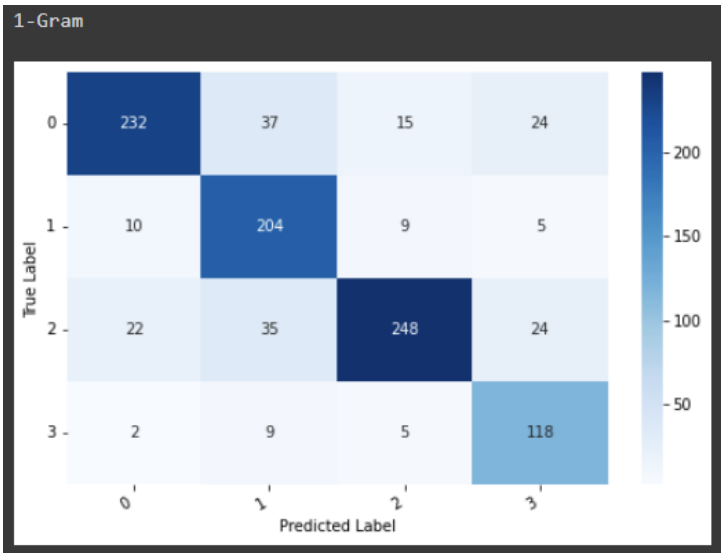
Terakhir adalah output gate yang menjadi gerbang terakhir untuk menghasilkan informasi data yang komplit dan aktual. Gerbang ini bisa menjadi yang terakhir atas sebuah informasi atau hanya menjadi bagian dari tahap pertama saja, sebelum akhirnya informasi akan diproses lewat input gate di sel berikutnya.

Analisis hasil eksperimen

1. Klasifikasi N-Gram

Dalam melakukan analisis hasil eksperimen dari model klasifikasi N-gram, penulis menggunakan matrik evaluasi agar dapat mengukur kualitas dari model klasifikasi yang telah dilatih. Metrik evaluasi yang digunakan pada penelitian ini adalah, Confusion Matrix yang merupakan konsep machine learning, dan mengandung informasi mengenai klasifikasi fakta dan prediksi. Precision yang digunakan untuk melakukan pengukuran pola positif yang dapat diprediksi benar dari hasil pola prediksi dalam kelas positif itu sendiri. Recall yang digunakan untuk melakukan pengukuran berupa pecahan dan berasal dari pola positif yang dilakukan klasifikasi dengan benar. f-1 score yang merepresentasikan mean harmonik dari nilai recall dan juga nilai precision, serta accuracy yang diukur sebagai rasio estimasi terhadap total estimasi. Hasil implementasi dari kode python untuk evaluasi pada model ini dapat dilihat pada gambar di bagian 2 dengan sub judul klasifikasi n-gram.

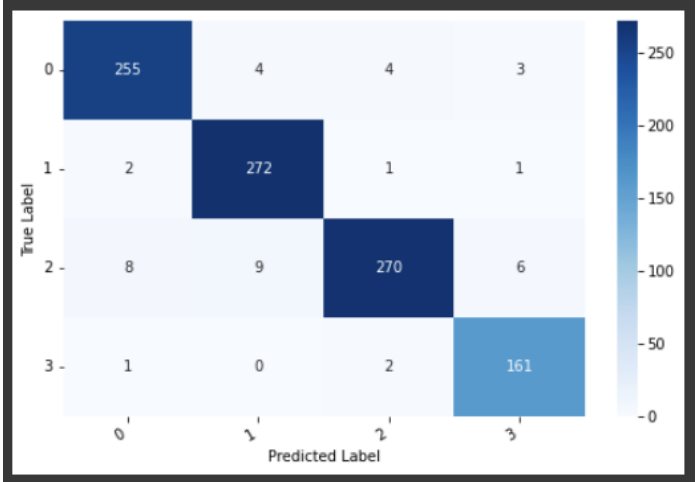
Adapun hasil dari evaluasi model ini, untuk N = 1 yaitu sebagai berikut:



	precision	recall	f1-score	support
Negative	0.87	0.75	0.81	308
Neutral	0.72	0.89	0.80	228
Positive	0.90	0.75	0.82	329
Irrelevant	0.69	0.88	0.77	134
accuracy			0.80	999
macro avg	0.79	0.82	0.80	999
weighted avg	0.82	0.80	0.80	999

Untuk N = 2 yaitu sebagai berikut:

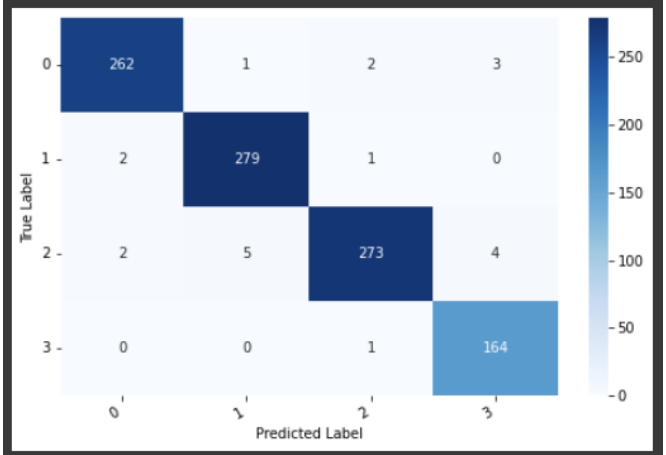
2-Gram



	precision	recall	f1-score	support
Negative	0.96	0.96	0.96	266
Neutral	0.95	0.99	0.97	276
Positive	0.97	0.92	0.95	293
Irrelevant	0.94	0.98	0.96	164
accuracy			0.96	999
macro avg	0.96	0.96	0.96	999
weighted avg	0.96	0.96	0.96	999

Untuk N = 3 yaitu sebagai berikut:

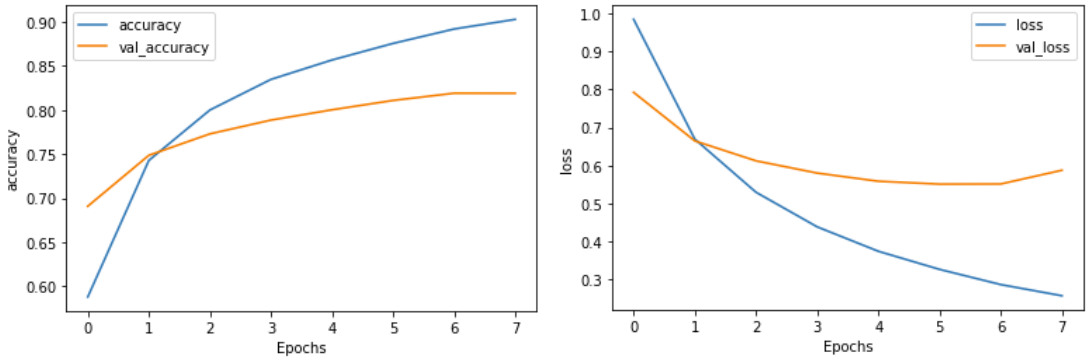
3-Gram



	precision	recall	f1-score	support
Negative	0.98	0.98	0.98	268
Neutral	0.98	0.99	0.98	282
Positive	0.99	0.96	0.97	284
Irrelevant	0.96	0.99	0.98	165
accuracy			0.98	999
macro avg	0.98	0.98	0.98	999
weighted avg	0.98	0.98	0.98	999

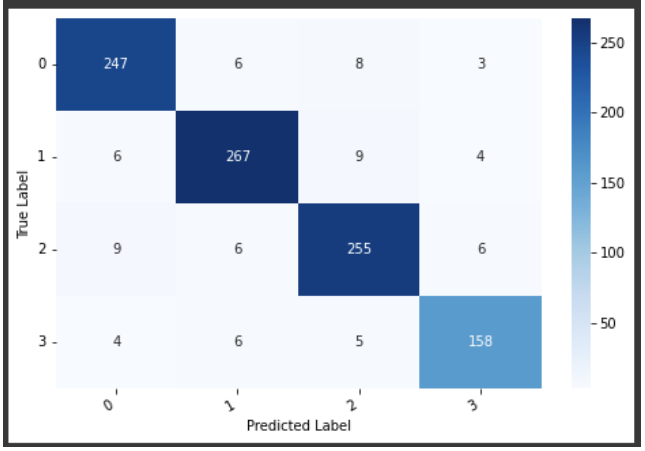
2. Klasifikasi Neural-Based

Berdasarkan model serta parameter yang telah dilatih untuk BiLSTM didapatkan hasil accuracy dan loss beserta metrik evaluasi untuk data tes sebagai berikut.



Berdasarkan grafik diatas didapatkan hasil epoch berhenti pada iterasi ke 7 setelah model memperhatikan callback untuk Early Stopping sehingga model berhenti ketika val loss sudah naik pada proses iterasi kedua kalinya.

	precision	recall	f1-score	support
Negative	0.93	0.94	0.93	264
Neutral	0.94	0.93	0.94	286
Positive	0.92	0.92	0.92	276
Irrelevant	0.92	0.91	0.92	173
accuracy			0.93	999
macro avg	0.93	0.93	0.93	999
weighted avg	0.93	0.93	0.93	999



Pada hasil prediksi data tes diatas didapatkan hasil rata rata untuk metrik evaluasi F1-Score, Accuracy, Precision, dan Recall untuk model BiLSTM sebesar 0.93. Pada model Neural Based hasil training model dilakukan dengan jangka waktu 1.5 menit.

```
stop = timeit.default_timer()
print('Waktu Training: ', (stop - start) / 60, 'Menit')
```

```
Waktu Training:  1.528201657883335 Menit
```

Kesimpulan

Dalam penulisan laporan ini, penulis melakukan eksplorasi dan eksperimen pemodelan bahasa dengan N-gram dan neural-based. Menggunakan korpus yang sama untuk N-gram dan neural based dimana dataset yang digunakan adalah dataset analisis sentimen twitter. Ada beberapa temuan eksperimental:

1. Semakin besar N yang digunakan pada klasifikasi N-Gram maka hasil dari matrik evaluasi seperti precision, recall, f1-score, accuracy juga akan semakin baik, dimana nilai maksimalnya ditunjukkan ketika N = 4 yaitu sebesar 0.98.

2. Pada model Neural Based BiLSTM didapatkan hasil prediksi data test sebesar 0.93 untuk rata-rata hasil evaluasi model serta waktu training sebesar 1.5 menit untuk 7 epoch.

Dengan temuan diatas, penulis menyimpulkan bahwa untuk dataset yang digunakan pada eksperimen ini, model bahasa dengan N-gram memiliki nilai akurasi dan waktu latih yang lebih baik dibandingkan dengan pemodelan bahasa neural based BiLSTM

Daftar referensi

[1] https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis
[2] https://towardsdatascience.com/basics-of-countvectorizer-e26677900f9c
[3] https://www.sciencedirect.com/science/article/pii/S095741741630118X
[4] https://www.tensorflow.org/text/tutorials/text_classification_rnn