ANALISA SENTIMEN

Menggunakan NN & LSTM

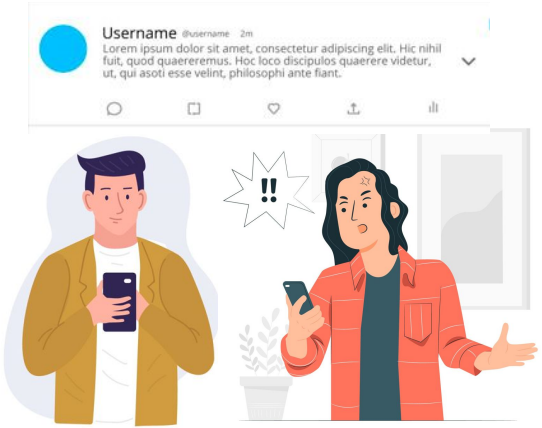
Kelompok 4

- Fyalisia Amanda Putri
- Ikhlasul Amal
- M Fauzi Wikantyoso

BINAR Data Science Challenge

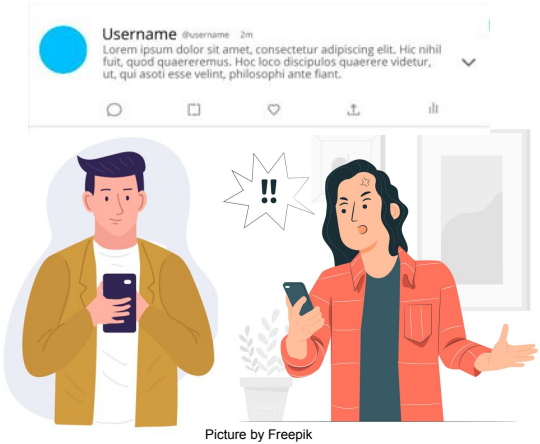
Platinum Level

Pendahuluan



Picture by Freepik

- Twitter merupakan salah satu platform media sosial yang paling banyak digunakan di Indonesia.
- Pengguna twitter biasanya membuat *tweet* untuk meluapkan isi hatinya terhadap sesuatu, baik itu orang, tempat bahkan kejadian
- *Tweet* tersebut dapat digunakan untuk menganalisa reaksi pengguna Twitter terhadap suatu hal, yang kemudian dapat dimanfaatkan untuk kepentingan promosi, development suatu produk dan lain lain
- Dalam melakukan Analisa Sentimen, ada banyak parameter dan fungsi yang dapat meningkatkan akurasi model



Picture by Freepik

Rumusan Masalah

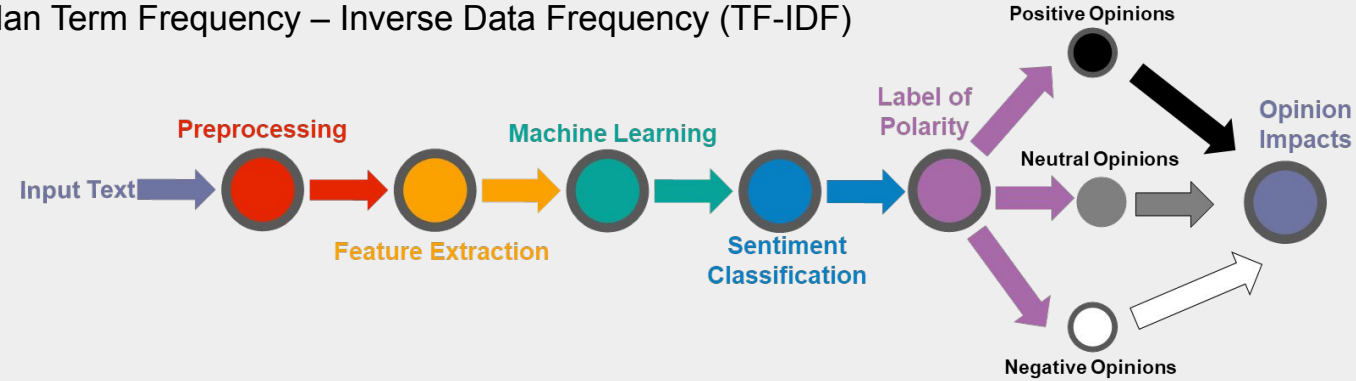
1. Apakah dengan menambahkan Stemming dan Stop Words dapat meningkatkan akurasi Sentimen Analisis menggunakan Neural Network?
2. Apakah ada perbedaan akurasi antara penggunaan Feature Extraction BoW dan TF-IDF pada model Neural Network dan LSTM?

Tujuan Penelitian

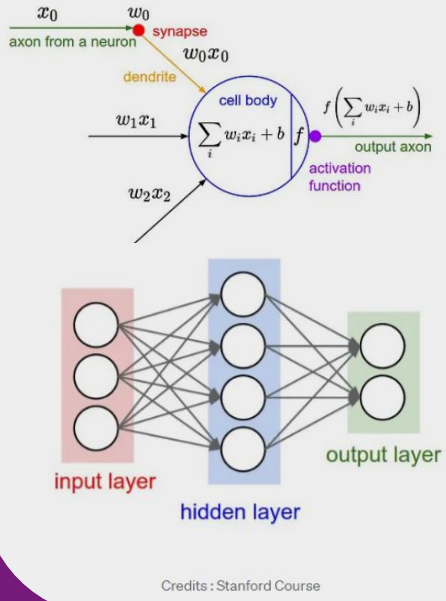
1. Melihat peningkatan akurasi jika dilakukan stemming dan Stop Words Removal pada tahap preprocessing menggunakan Neural Network
2. Membandingkan tingkat akurasi pada Neural Network yang menggunakan BoW dan TF-IDF sebagai fungsi feature extraction nya
3. Melihat perbedaan akurasi antara Neural Network dan LSTM jika menggunakan feature extraction TF-IDF dan BoW

Sentiment Analysis

- Analisa Sentimen sudah digunakan sebagai bagian dari *Natural Language Processing* untuk memproses bahasa dengan berbagai tingkat ketelitian. Mulai dari pengklasifikasian pada dokumen, kalimat bahkan yang paling terbaru pada tingkat frasa.
- Contohnya pada sosial media Twitter dengan total 368 juta pengguna aktif, pengguna yang merasa dapat langsung mengekspresikan pendapat maupun reaksi mereka terhadap postingan atau *tweet* pengguna lain secara cepat dan mudah membuat *tweets* tersebut dapat diekstraksi dan dimanfaatkan
- Beberapa metode yang populer digunakan untuk membangun model analisa sentimen diantaranya Naive Bayes, Support Vector Machines, Neural Network dan Long Short Term Memory (LSTM). Untuk mendukung metode tersebut terkadang diperlukan fungsi feature extraction seperti Bag of Words (BOW) dan Term Frequency – Inverse Data Frequency (TF-IDF)



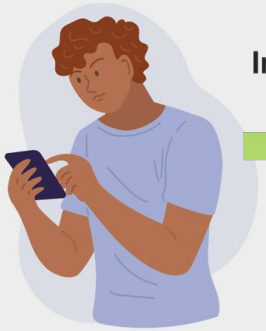
Neural Network



- Neural Network menyerupai sistem syaraf manusia yang saling terhubung satu dan lainnya, perbedaanya Neural network menggunakan activation function untuk meneruskan ataupun tidak meneruskan output ke neuron selanjutnya.
- Activation function yang di gunakan adalah Rectified Linear Unit (ReLU)
- Diperlukan Feature Extraction sehingga di dapatkan input berupa matriks (ukuran matriks menyesuaikan jenis data), metode feature extraction yang di gunakan adalah Bag of Words (BoW) dan Term Frequency - Inverse Data Frequency (TF-IDF)

BINAR

Neural Network



Picture by Freepik

Input Tweet dg Label

Pre Processing

Pre Processing / Data Preparation

- Text Cleaning
- Check w/ Kamus Alay
- Stemming

Feature Extraction

Method:

- Bag of Words **(BOW)**
- Term Frequency – Inverse Data Frequency **(TF-IDF)**

Feature Extraction

Train Process

Model Training

Neural Network MLPClassifier from Scikit Learn

- Learning Rate Constant

Cross Validation

Compare to Data Test using Kfold function

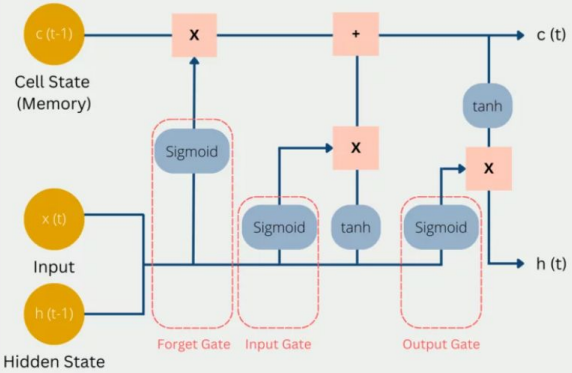
Cross Validation

Trained & Validated Model

Model ready to do Sentimen Analisis

Picture by Freepik

How LSTM works



Pada setiap tahap perhitungan, digunakan input x(t), short-term memory tahap sebelumnya c(t-1), dan hidden state sebelumnya h(t-1)

Gates yang harus di lewati oleh ketiga nilai tersebut untuk digunakan pada cell berikutnya maupun hidden state berikutnya adalah:

1. Forget Gate
2. Input Gate
3. Output Gate

- *Long Short-Term Memory (LSTM) networks* merupakan *Recurrent Neural Networks*. Pada *Recurrent Neural Networks*, informasi dari neuron sebelumnya disimpan pada *short-term memory* untuk digunakan pada neuron selanjutnya. Hal tersebut menjadi kelemahan ketika *sequences* terlalu banyak, sehingga untuk menanggulangi hal tersebut muncul LSTM
- Permasalahan pada *Recurrent Neural Networks* adalah *short-term memory runs out* dimana memori tersebut menyimpan data dari neuron sebelumnya, sehingga data terlama akan di *overwrite.* Sehingga LSTM menggunakan *long-term memory* untuk menyimpan informasi yang dibutuhkan melalui *gates*. Tempat penyimpanan tersebut dinamakan *Cell State* dan tetap memiliki *Hidden Sate* untuk menyimpan *short-term memory* seperti Neural Network biasa.

BINAR

Long Short Term Memory (LSTM)

Input Tweet

Picture by Freepik

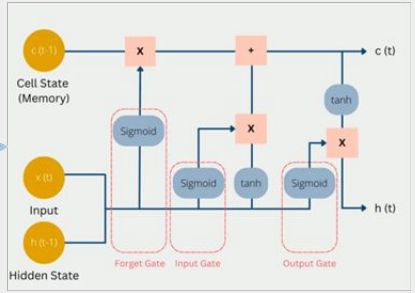
Pre Processing

- Text Cleaning
- Check w/ Kamus Alay
- Stemming

Feature Extraction

Method:
- Tokenizer
- Pad Sequences

Train Process



LSTM Keras from TensorFlow
- Activation function Softmax
- Optimizer Adam learning rate 0,001%
- Drop out 2%

Validating using data validation

Trained & Validated Model

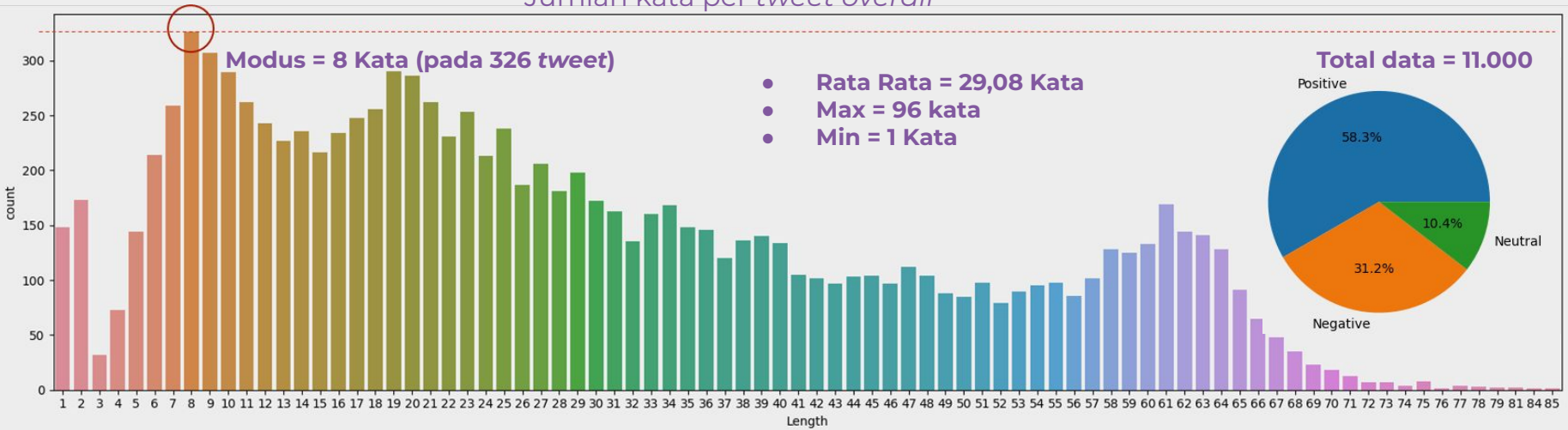
Model ready to do Sentimen Analisis

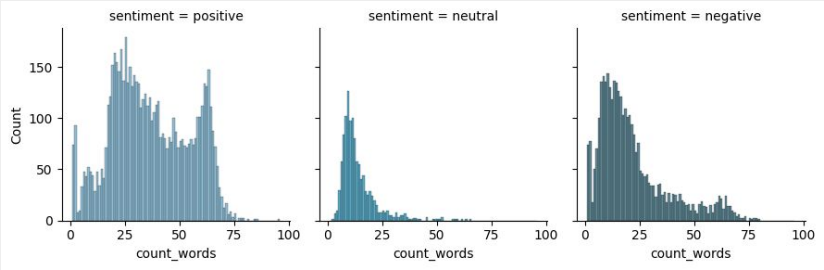
Picture by Freepik

BINAR

Data Training dan Validation yang di gunakan

Jumlah kata per *tweet overall*



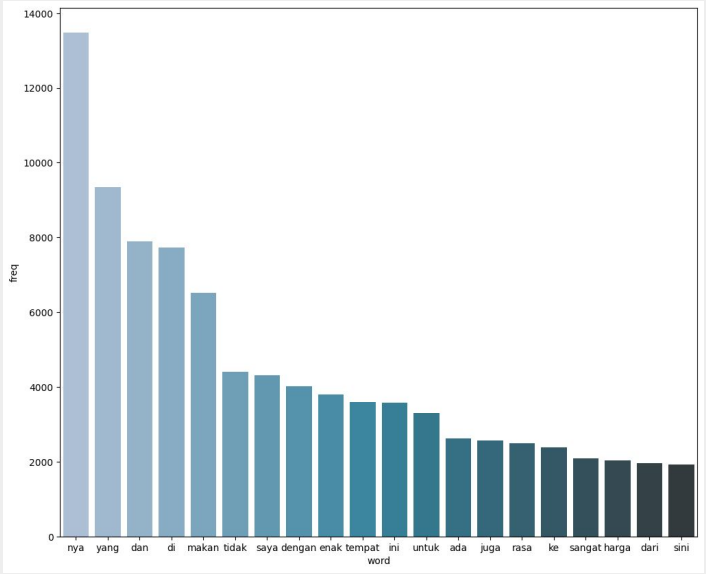


Per Sentimen

```
Descriptive stats for count_words
-----------------------------------
           count       mean        std  min   25%   50%   75%   max
sentiment
negative  3436.0  21.247672  16.051194  1.0  10.0  17.0  27.0  79.0
neutral   1148.0  13.396341   8.163150  2.0   8.0  11.0  16.0  65.0
positive  6416.0  36.456359  17.866700  1.0  23.0  34.0  52.0  96.0
```

Data Set yang di Gunakan

The most frequent words

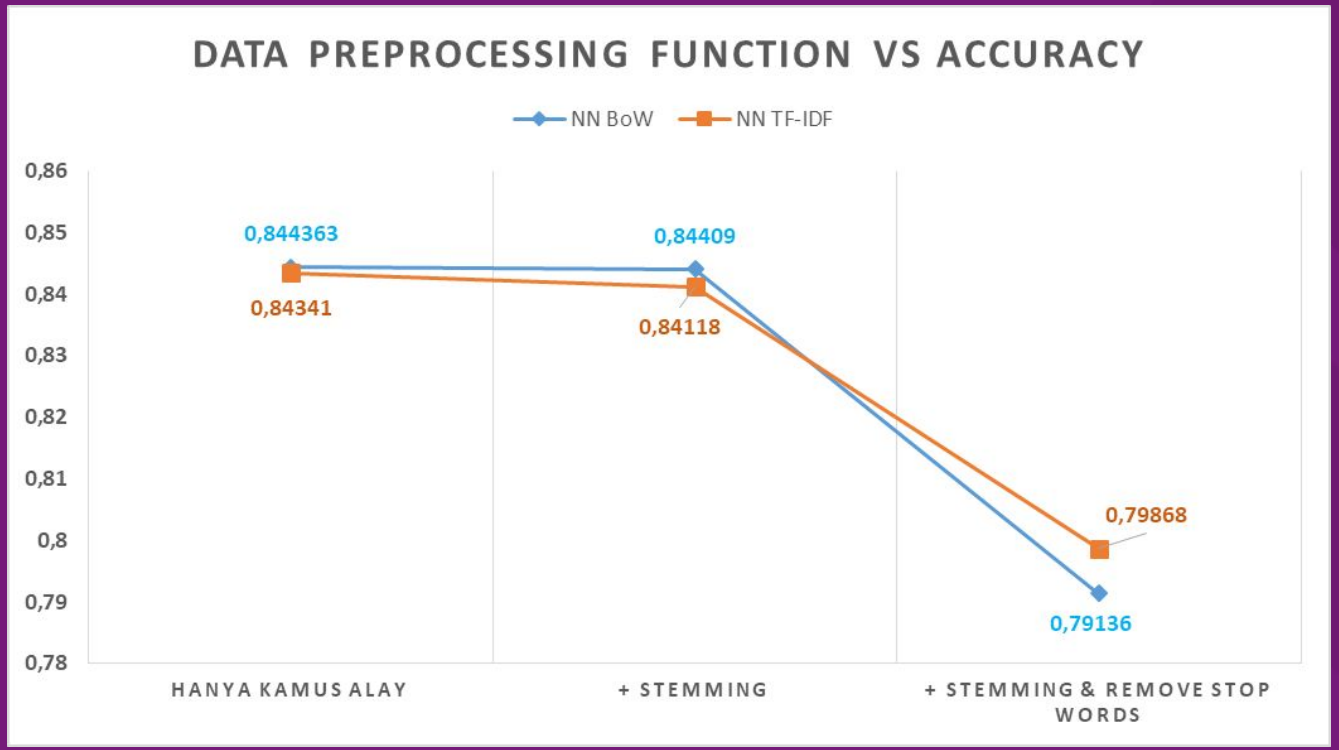


Top 10 Words

1. nya
2. yang
3. dan
4. di
5. makan
6. tidak
7. saya
8. dengan
9. enak
10. tempat

Result NN Data Preprocessing Function

Perbandingan Akurasi, jika Neural Network dilakukan pencocokan dengan Kamus Alay saja, ditambahkan stemming text dan ditambahkan stemming dan remove stop words pada saat data preprocessing



Result LSTM Compare Feature Extraction

Perbandingan Akurasi, jika Neural Network dilakukan feature extraction menggunakan BoW dan TF-IDF (data preprocessing : cleaning text, compare to kamus alay, stemming)

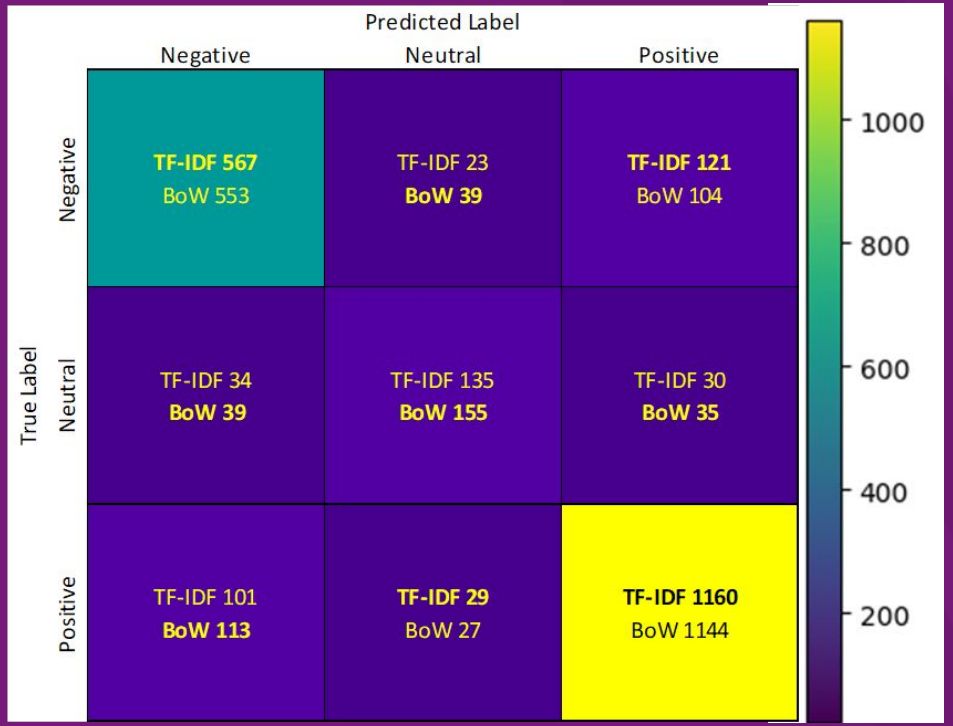
True Positive & Negative
TF-IDF > BoW

False Positive
TF-IDF > BoW

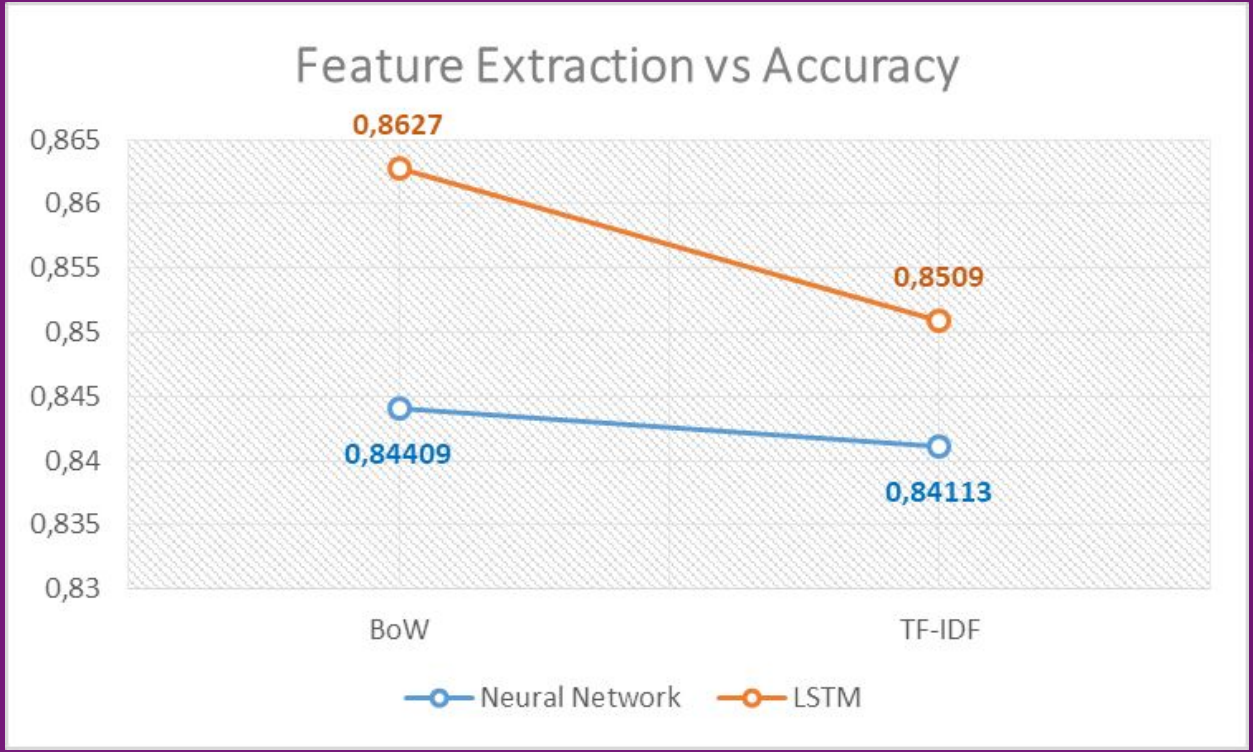
True Neutral
TF-IDF < BoW

False Negative
TF-IDF < BoW

% Accuracies
TF-IDF < BoW
84,40% 84,11%



Result NN & LSTM Compare Feature Extraction

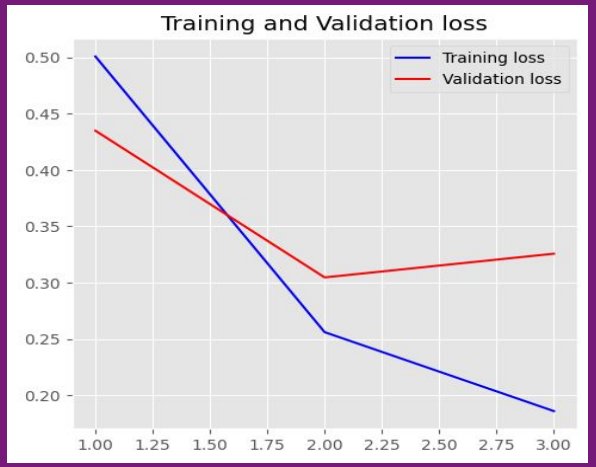


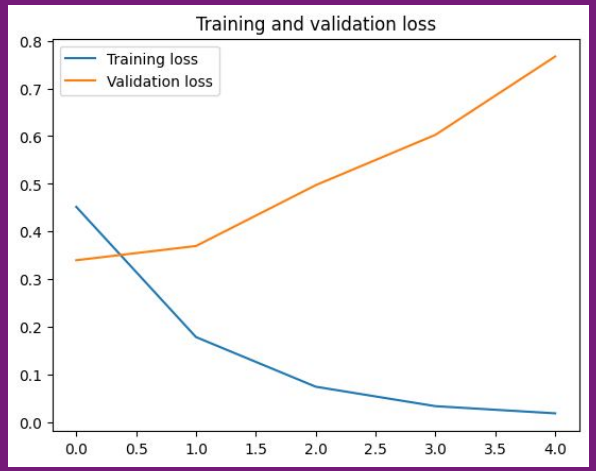
Neural Network & LSTM dilakukan feature extraction menggunakan BoW dan TF-IDF
(data preprocessing : cleaning text, compare to kamus alay, stemming)

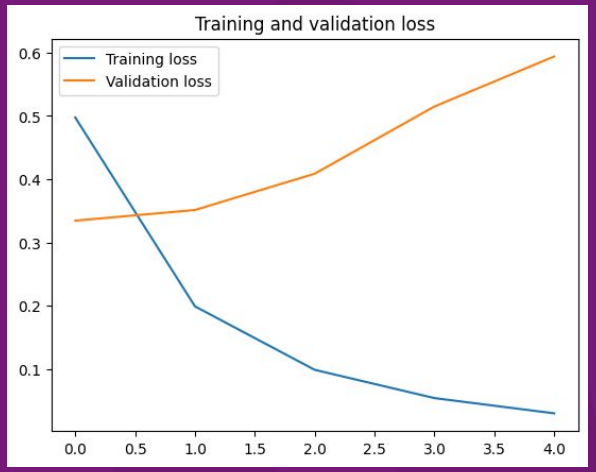
Result LSTM Compare Feature Extraction

	TFIDF	BOW	word Embed
Accuracy	0.8509	0.8732	0.8754
Percentage	85.09%	87.32%	87,54%

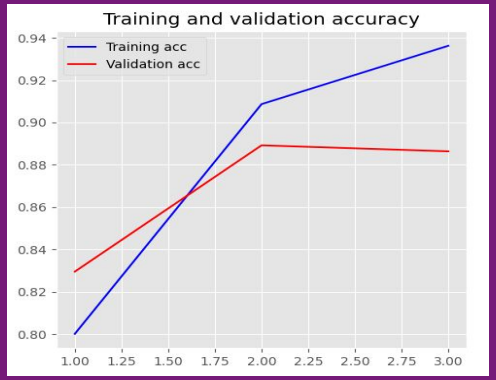
Result LSTM



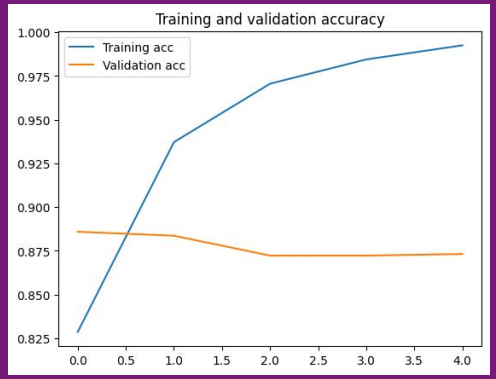




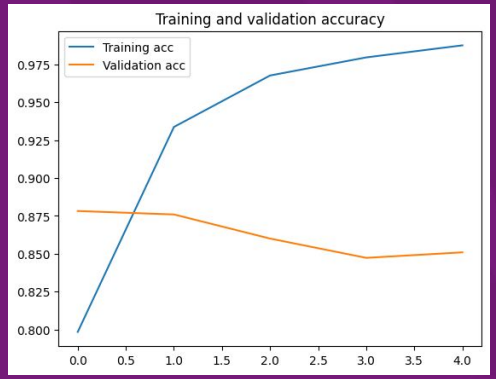
Word Embedding



Bag of Words



TF-IDF



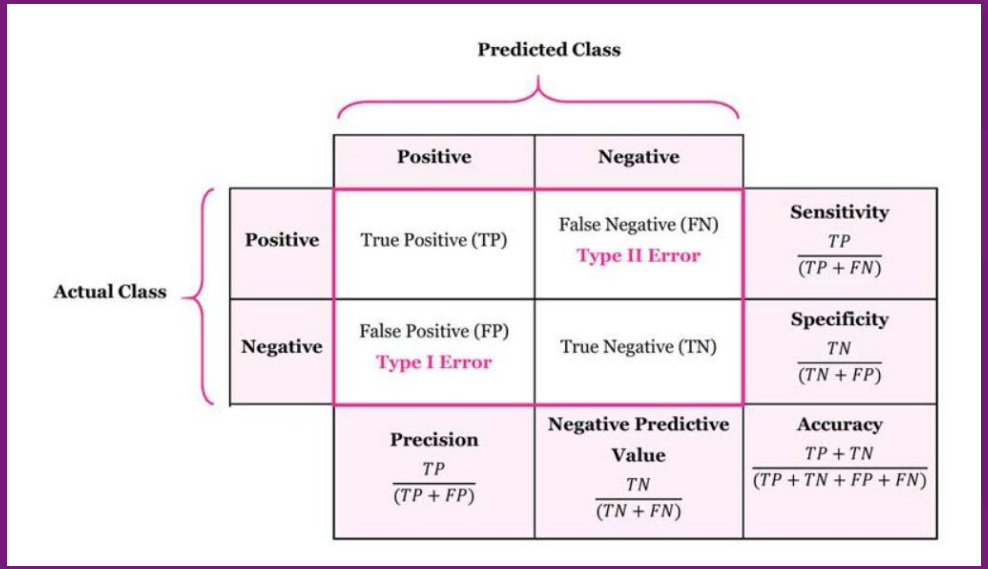
Kesimpulan

- Penggunaan Stemming Text dan Stop Words dapat menurunkan akurasi model analisa sentimen menggunakan neural network, salah satu penyebabnya terdapat beberapa homograph pada bahasa indonesia, sehingga akan ikut terhapus oleh stop words, contohnya Bapak jualan tahu. Karena ‘tahu’ merupakan stop words, maka kata ‘tahu’ yang memiliki arti berbeda tetap terhapus.
- Pada LSTM maupun NN penggunaan feature extraction BoW menghasilkan akurasi yang lebih tinggi jika di bandingkan dengan TF-IDF pada model Analisa Sentimen.
- Bahkan pada LSTM akurasi model dengan feature extraction BoW menghasilkan akurasi yang lebih tinggi daripada TF-IDF namun word embedding (dalam platform Keras) menghasilkan tingkat akurasi yang lebih tinggi
- Hasil akurasi LSTM masih lebih tinggi jika dibandingkan dengan Neural Network

BINAR

Terimakasih

Lampiran



Cara menghitung confusion matrix

BINAR

Neural Network menggunakan Kamus alay dan process Stopwords

	precision	recall	f1-score	support
negative	0.70	0.71	0.71	673
neutral	0.73	0.62	0.67	229
positive	0.85	0.87	0.86	1298
accuracy			0.79	2200
macro avg	0.76	0.73	0.75	2200
weighted avg	0.79	0.79	0.79	2200

Accuracy sekitar 79%

```
# prediction
original_text = '''
saya suka makan di restoran itu karena makanannya enak enak
'''

# Feature Extraction
text = vectorizer.transform([cleaning_data(original_text)])

#print(text) #0, 14410

# predict the sentiment
result = model.predict(text)[0]
print("Sentiment:")
print()
print(result)
```

```
Sentiment:

positive
```

Test Sentimen dengan frasa “saya suka makan di restoran itu karena makanannya enak enak” menghasilkan sentimen positif

Result NN using BOW

Neural Network menggunakan Kamus alay dan process Stemm

	precision	recall	f1-score	support
negative	0.78	0.80	0.79	687
neutral	0.73	0.68	0.70	229
positive	0.89	0.89	0.89	1284
accuracy			0.84	2200
macro avg	0.80	0.79	0.80	2200
weighted avg	0.84	0.84	0.84	2200

Accuracy sekitar 84%

```
# prediction
original_text = '''
saya suka makan di restoran itu karena makanannya enak enak
'''

# Feature Extraction
text = vectorizer.transform([cleaning_data(original_text)])

#print(text) #0, 14410

# predict the sentiment
result = model.predict(text)[0]
print("Sentiment:")
print()
print(result)
```

```
Sentiment:

positive
```

Test Sentimen dengan frasa “saya suka makan di restoran itu karena makanannya enak enak” menghasilkan sentimen positif

Result NN using TF-IDF

	precision	recall	f1-score	support
negative	0.78	0.78	0.78	679
neutral	0.80	0.71	0.76	228
positive	0.89	0.91	0.90	1293
accuracy			0.85	2200
macro avg	0.82	0.80	0.81	2200
weighted avg	0.85	0.85	0.85	2200

Accuracy maksimum sekitar 85%

Test Sentimen dg Frasa “saya suka makan di restoran itu karena makanannya enak enak”

```
# prediction
original_text = '''
saya suka makan di restoran itu karena makanannya enak enak
'''

# Feature Extraction
text = vectorizerr.transform([cleaning_data(original_text)])

#print(text) #0, 14410

# predict the sentiment
result = model.predict(text)[0]
print("Sentiment:")
print()
print(result)
```

```
Sentiment:

positive
```

Hasil analisa sentimen berupa sentimen positif

Result NN using TF-IDF

	precision	recall	f1-score	support
negative	0.81	0.80	0.80	711
neutral	0.72	0.68	0.70	199
positive	0.88	0.90	0.89	1290
accuracy			0.85	2200
macro avg	0.80	0.79	0.80	2200
weighted avg	0.85	0.85	0.85	2200

Accuracy maksimum sekitar 85%

Test Sentimen dg Frasa “saya suka makan di restoran itu karena makanannya enak enak”

```
# prediction
original_text = '''
saya suka makan di restoran itu karena makanannya enak enak
'''

# Feature Extraction
text = vectorizerr.transform([cleaning_data(original_text)])

#print(text) #0, 14410

# predict the sentiment
result = model.predict(text)[0]
print("Sentiment:")
print()
print(result)
```

```
Sentiment:

positive
```

Hasil analisa sentimen berupa sentimen positif

Result NN using TF-IDF

BINAR

	precision	recall	f1-score	support
negative	0.70	0.75	0.72	683
neutral	0.70	0.52	0.60	206
positive	0.86	0.86	0.86	1311
accuracy			0.79	2200
macro avg	0.75	0.71	0.73	2200
weighted avg	0.79	0.79	0.79	2200

Accuracy maksimum sekitar 79%

Test Sentimen dg Frasa “saya suka makan di restoran itu karena makanannya enak enak”

```
# prediction
original_text = '''
saya suka makan di restoran itu karena makanannya enak enak
'''

# Feature Extraction
text = vectorizerr.transform([cleaning_data(original_text)])

#print(text) #0, 14410

# predict the sentiment
result = model.predict(text)[0]
print("Sentiment:")
print()
print(result)
```

```
Sentiment:

positive
```

Hasil analisa sentimen berupa sentimen positif

Result LSTM using Word Embedding

BINAR

	precision	recall	f1-score	support
0	0.87	0.77	0.81	680
1	0.74	0.80	0.77	239
2	0.89	0.93	0.91	1281
accuracy			0.87	2200
macro avg	0.83	0.83	0.83	2200
weighted avg	0.87	0.87	0.87	2200

Accuracy maksimum sekitar 87%

Test Sentimen dengan Frasa “saya suka makan di restoran itu karena makanannya enak enak”

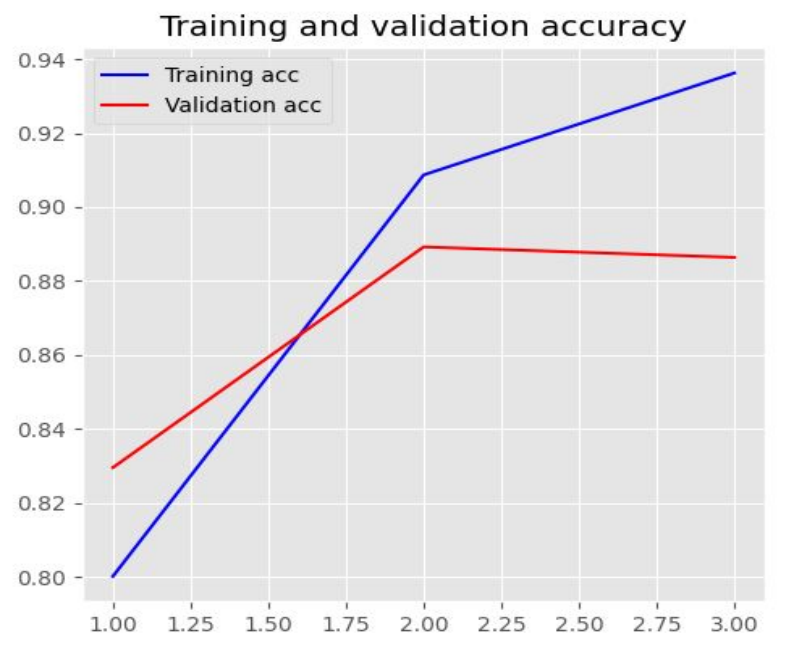
```
original_text =  '''
saya sangat suka makan di restaurant itu karena makanannya enak enak
'''
def cleansing(string):
  string = cleaning_data(string)

  return string


sentiment = ['negative', 'neutral', 'positive']

text = [cleansing(original_text)]
predicted = tokenizer.texts_to_sequences(text)
# Padding change token to matrix/vector
choice = pad_sequences(predicted)

model=load_model('model.h5')
prediction = model.predict(choice)
polarity = np.argmax(prediction[0])

print("Text :", text[0])
print("Sentiment :", sentiment[polarity])
```

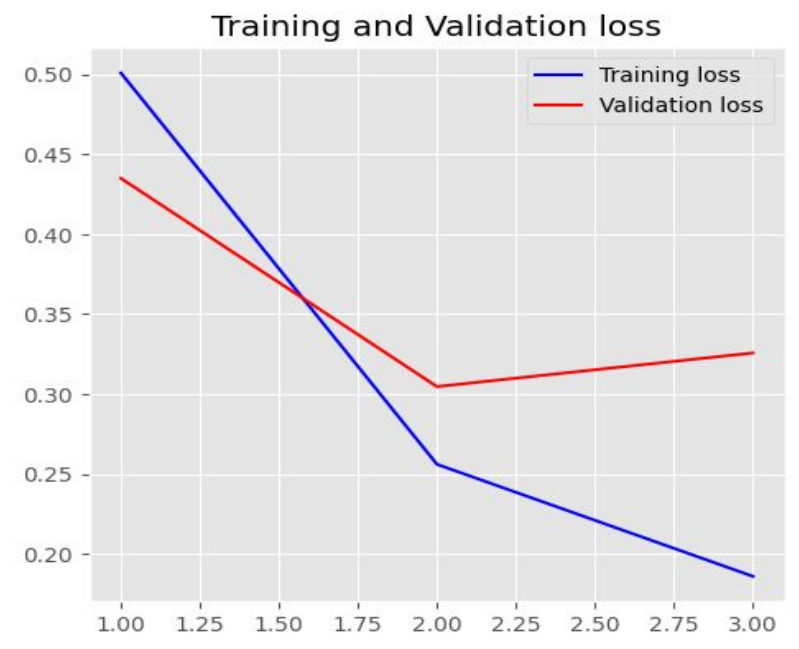
```
1/1 [==============================] - 1s 516ms/step
Text :  saya sangat suka makan di restaurant itu karena makanannya enak enak
Sentiment : positive
```

Hasil analisa sentimen berupa sentimen positif

Result LSTM using Word Embedding

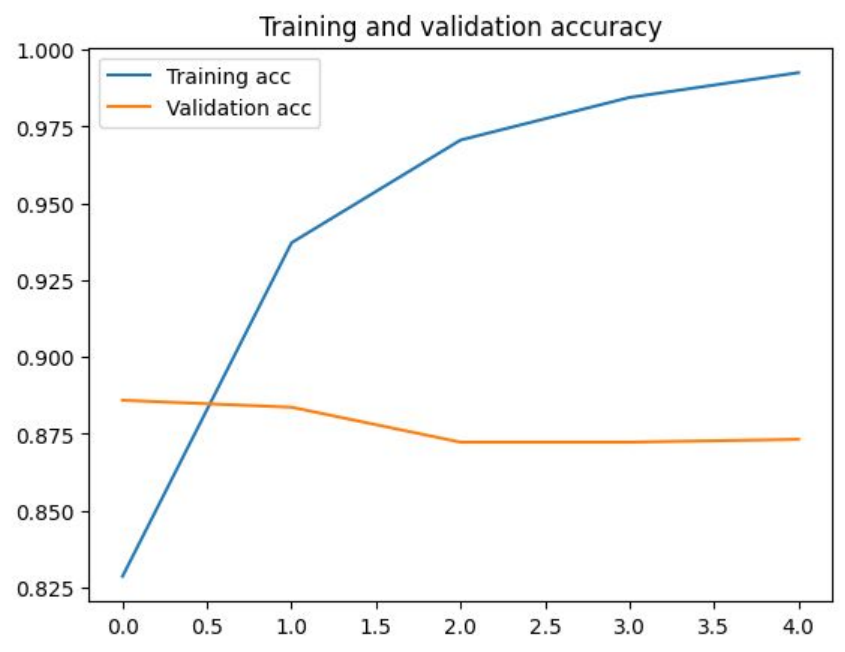
```
Average accuracy : 0.8754545454545454
```

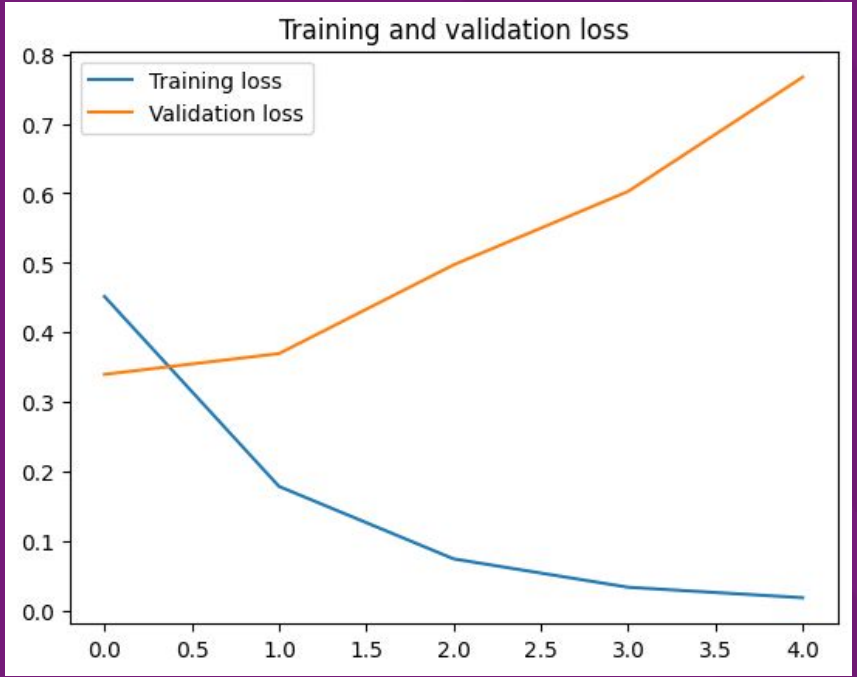




Result LSTM using BOW

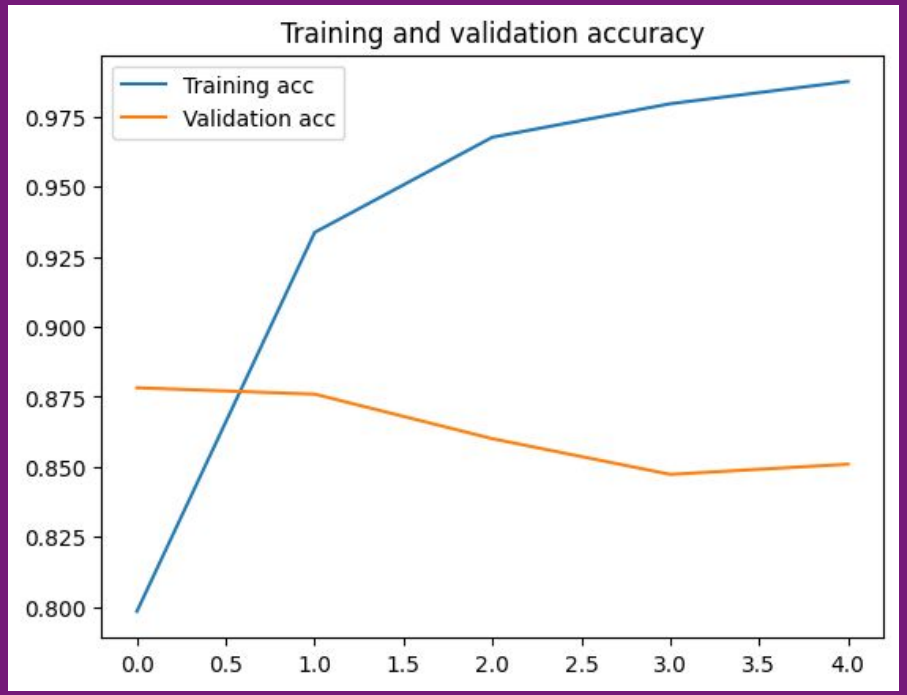
```
550/550 [==============================] - 3s 5ms/step - loss: 0.0189 - accuracy: 0.9925 - val_loss: 0.7668 - val_accuracy: 0.8732
```

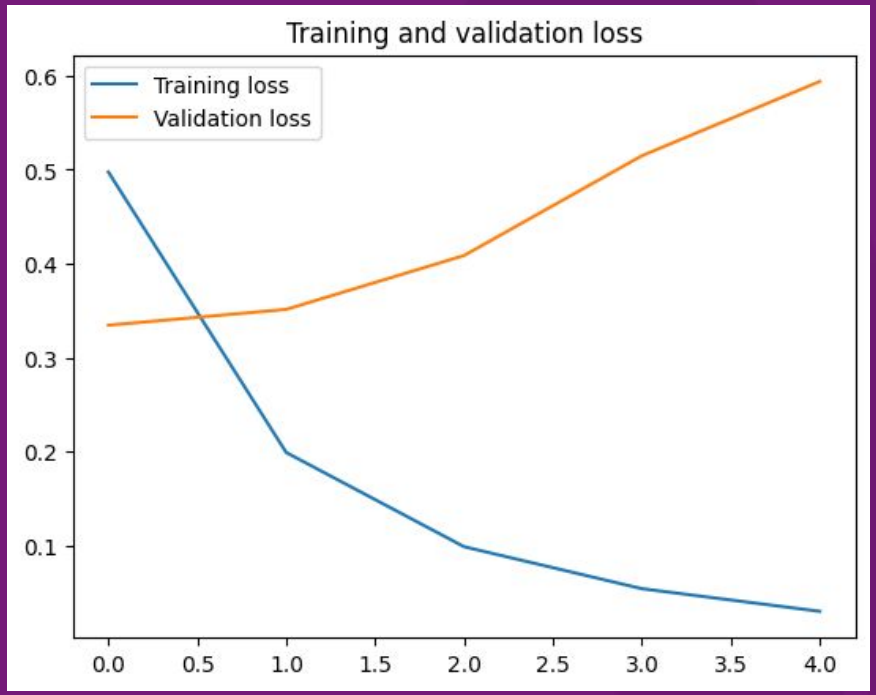




Result LSTM using TFIDF

```
54s 99ms/step - loss: 0.0300 - accuracy: 0.9875 - val_loss: 0.5938 - val_accuracy: 0.8509
```





	NN - BOW - Kamus Alay	NN - BOW - Kamus Alay dan Process Stemm	NN - BOW - Kamus Alay dan Process Stopwords
Accuracy	0.844363	0.84409	0.79136
Percentage	84,44%	84.40%	79.10%
Hasil	Positive	Positive	Positive

	NN - TFIDF- Kamus Alay	NN - TFIDF - Kamus Alay dan Process Stemm	NN - TFIDF - Kamus Alay dan Process Stopwords
Accuracy	0.84341	0.84118	0.79868
Percentage	84.34%	84.11%	79.87%
Hasil	Positive	Positive	Positive