# Analisis Data Sentimen Mengunakan Metode Neutral Network Dan LSTM

By: Ahmed Basymeleh, Vincent Arlen Santoso, Nadia Rahmadianty

# PENDAHULUAN

## LATAR BELAKANG

10 tahun lalu, menyampaikan aspirasi, keluhan, dan kritik kepada pejabat negara adalah hal yang awam dikarenakan tidak adanya “alat” yang dapat dijangkau oleh seluruh lapisan masyarakat. Tentu saja, hal ini telah menciptakan sebuah “gap” antara rakyat dan pimpinannya. Namun, hari ini social media menutup “gap” tersebut dan menjelma menjadi sebuah “alat” bagi masyarakat untuk secara tidak langsung dapat berinteraksi dengan “bercuit-cuit” dalam salah satu social media, twitter.

Tingginya antusiasme masyarakat dalam pemilu 2019 dapat dirasakan pengguna twitter, cuitan negatif maupun positif saling bersaut didalamnya. Tanpa disadari cuitan masyarakat menjadi jejak digital bagi para pengguna dan telah menghasilkan data yang menggunung, yang dapat digunakan untuk banyak kepentingan. Cuitan yang hetergogen (penyingkatan kata, karakter tanda baca yang berlebih dalam kalimat) menjadikan analisa data tersebut tidak semudah yang dibayangkan.

Data Science adalah salah satu cabang ilmu digital yang dapat diaplikasikan dalam mengolah data yang dapat digunakan untuk banyak hal seperti perencanaan dan pengambilan keputusan. Pada analisa ini data dari twitter pada saat pemilihan presiden 2019 digunakan sebagai sumber data dan Data Science / Machine Learning diaplikasikan untuk menganalisis sentiment social media apakah codong positive dan negative, maupun netral. Dari analisis ini pengambilan keputusan dapat lebih disesuaikan dengan keadaan sesungguhnya.

# PENDAHULUAN

## RUMUSAN MASALAH

Berangkat dari latar belakang permasalahan, topik yang akan diangkat dari analisis ini adalah membuat sebuah engine/API yang bisa memilah komentar positif, netral, dan negatif dari komentar netizen dari teks non-formal.

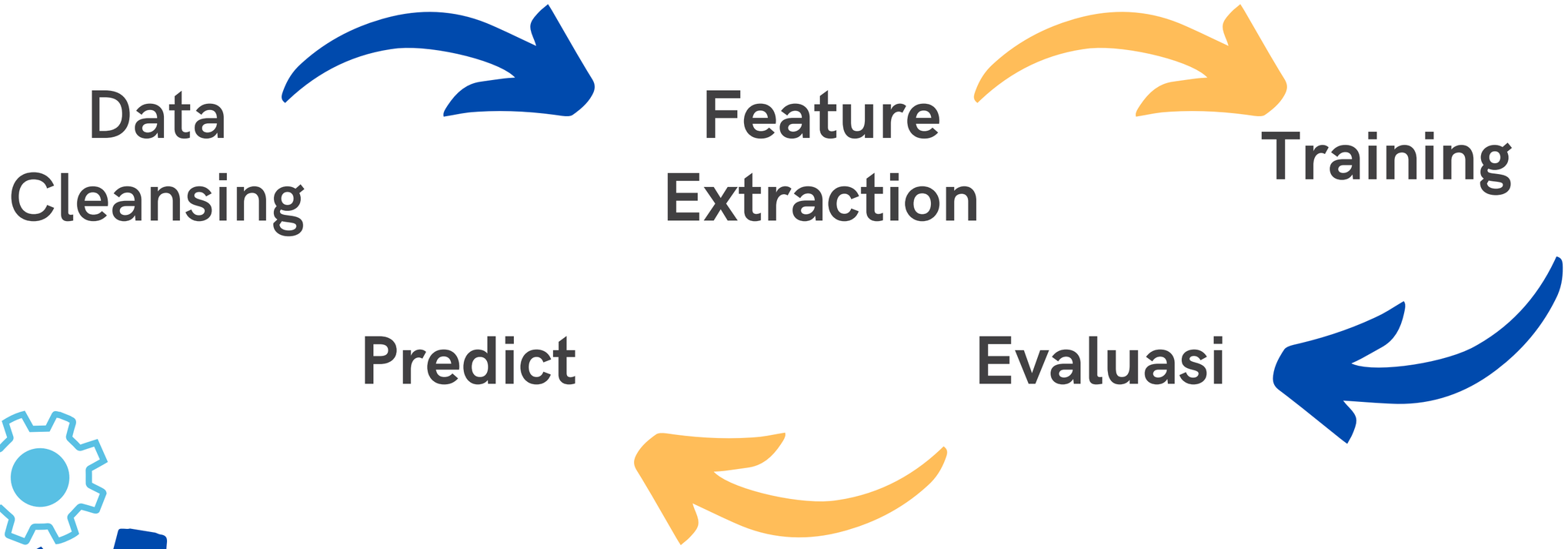
- Feature extraction terbaik apakah yang bisa diaplikasikan dalam analisis ini?
- Berapakah akurasi sentimen analisis menggunakan NN?
- Berapakah akurasi sentimen analisis menggunakan LSTM?
- Model apakah yang paling baik digunakan untuk analisis ini?

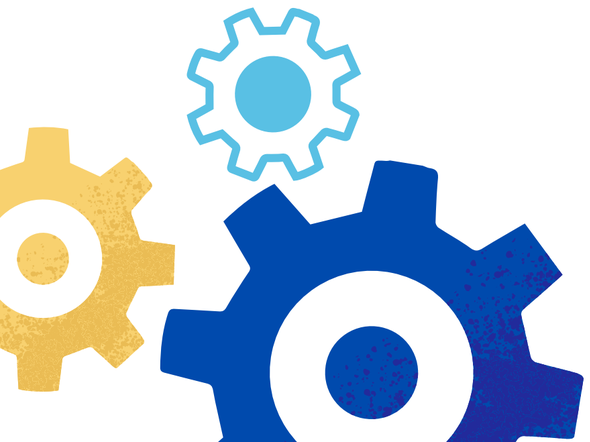
## TUJUAN PENELITIAN

Tujuan dari API yang bisa memilah komentar/sentiment positif, negatif, ataupun neutral dalam penelitian ini adalah untuk membantu end-user dalam mengambil sebuah keputusan dalam perencanaan.

# METODE PENELITIAN

Tahapan-Tahapan Penelitian:



# Data Cleansing

Tahap pertama dalam menganalisa data yang diberikan adalah melakukan "data cleansing", dimana data akan diseragamkan dalam formatnya, seperti :

1. Menghapus tanda baca yang tidak diperlukan/berlebih
2. Merubah kata-kata tidak baku menjadi baku
3. Merubah kata-kata yang disingkat menjadi kata asli/bakunya
4. menghapus stopwords pada data teks

Input kamus yang telah diberikan

```
def fix_word(text):
    return ' '.join([kamus_dict[word] if word in kamus_dict else word for word in text.split(' ')])
```

Menghapus karatakter/ kata dalam kalimat yang tidak diperlukan

```
def remove_unnecessaryChar(text):
    text = re.sub(r'&amp;|amp;|&', 'dan', text)
    text = re.sub(r'\\n+', '', text)
    text = re.sub('&lt;/?[a-z]+&gt;', ' ', text)
    text = re.sub(r'#+','#', text)
    text = re.sub(r'http\S+',' ',text)
    text = re.sub(r'(USER+\s?|RT+\s?|URL+\s?)', ' ', text)
    text = re.sub(r'x[a-zA-Z0-9]+', ' ', text)
    return text
```

Menghapus tanda baca

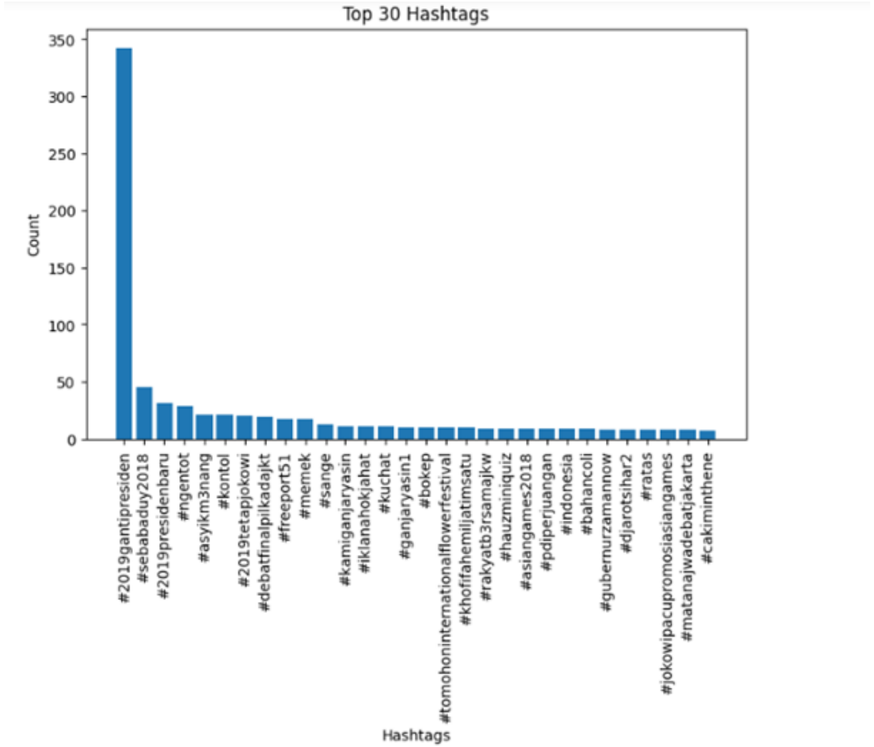
```
def remove_punctuation(text):
    text = re.sub(r'\?', '', text)
    text = re.sub(r'[^a-zA-Z0-9]+', ' ', text)
    text = re.sub(r' +', ' ', text.lower().lstrip("0123456789").strip())
    return text
```

Mengkombinasi 3 aturan diatas untuk data cleansing

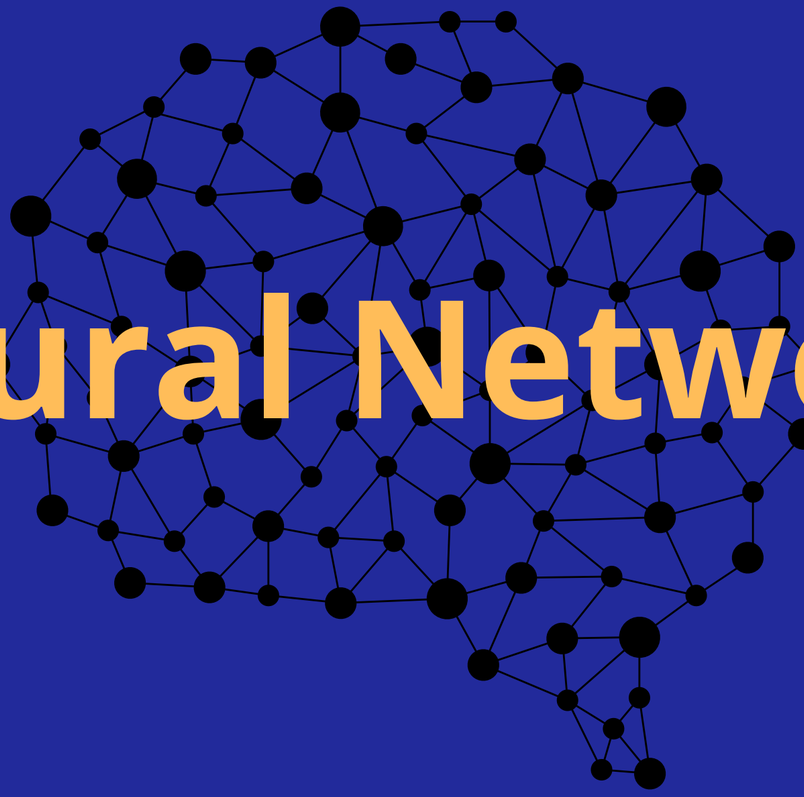
```
def preprocessing(text):
    text = remove_unnecessaryChar(text)
    text = remove_punctuation(text)
    text = fix_word(text)
    return text
```

# Data Cleansing

- METODE STATISTIKA: Descriptive statistics, untuk memberikan gambaran deskriptif mengenai data hashtag, seperti nilai rata-rata, median, modus, dan rentang data.
- EDA (Exploratory Data Analysis):
    1. Countplot: untuk menghitung jumlah hashtag tertentu dan di visualisasikan dengan menggunakan histogram
    2. Wordcloud: untuk melakukan visualisasi pada frekuensi hashtag secara lebih informatif dan memperlihatkan hashtag" yang paling banyak muncul




# Neural Network



- Split dataset untuk train dan test

```
# split dataset
X = df['text']
y = df['sentiment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
```

- Melakukan Feature Extraction menggunakan Bag of Words

```
# feature extraction
vectorizer = CountVectorizer()
X_train_features = vectorizer.fit_transform(X_train)
X_test_features = vectorizer.transform(X_test)
```

- Train dataset menggunakan MLP Classifier

```
# train data
model = MLPClassifier(
    hidden_layer_sizes=(37,),
    solver="adam",
    activation="relu",
    early_stopping=True,
    random_state=27
    )

# Cross-validation
cv_scores = cross_val_score(model, X_train_features, y_train, cv=5)

model.fit(X_train_features, y_train)
```

# Classification Report

- Akurasi Training yang didapatkan sekitar 95%
- Dan akurasi yang didapatkan dari testnya sekitar 88%
- Dan ketika melalui K-fold hasil rata-rata akurasi yang didapatkan sekitar 85%

```
# evaluation
train_accuracy = model.score(X_train_features, y_train)
print("Training Accuracy:", train_accuracy)

y_pred = model.predict(X_test_features)
accuracy = accuracy_score(y_test, y_pred)
report = model(y_test, y_pred, target_names=le.classes_)
print("Validation/Test Accuracy:", accuracy)
print("Classification Report:")
print(report)
```

```
Training Accuracy: 0.9584090909090909
Validation/Test Accuracy: 0.8759090909090909
Classification Report:
              precision    recall  f1-score   support

    negative       0.81      0.85      0.83       678
     neutral       0.88      0.72      0.79       236
    positive       0.91      0.92      0.91      1286

    accuracy                           0.88      2200
   macro avg       0.87      0.83      0.85      2200
weighted avg       0.88      0.88      0.88      2200
```

```
Cross-validation scores: [0.85454545 0.85113636 0.85113636 0.85340909 0.87045455]
Mean cross-validation score: 0.8561363636363636
```

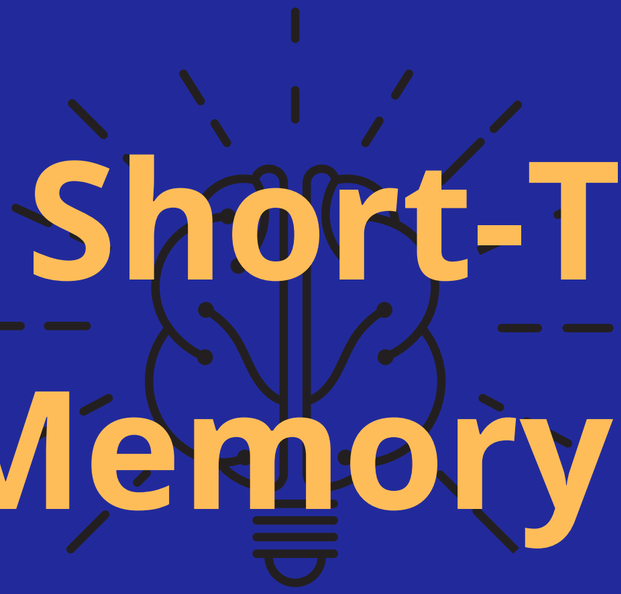
# perbandingan akurasi 3 feature extraction

| | BoW w/o SW | BoW w/ SW | TF-IDF w/o SW | TF-IDF w/ SW | word2vec w/o SW | word2vec w/ SW |
|---|---|---|---|---|---|---|
| **Train Acc** | 95,8% | 95,5% | 95,7% | 94,6% | 78,7% | 81,5% |
| **Test Acc** | 87,5% | 88,1% | 85,9% | 86,9% | 78,0% | 80,7% |
| **5-fold mean Acc** | 85,6% | 87,1% | 85,1% | 86,5% | 78,5% | 81,2% |

dari tabel diatas, Bag of Words mendapatkan akurasi paling tinggi , dan menghapus stopwords mengurangi akurasi walau tidak signifikan

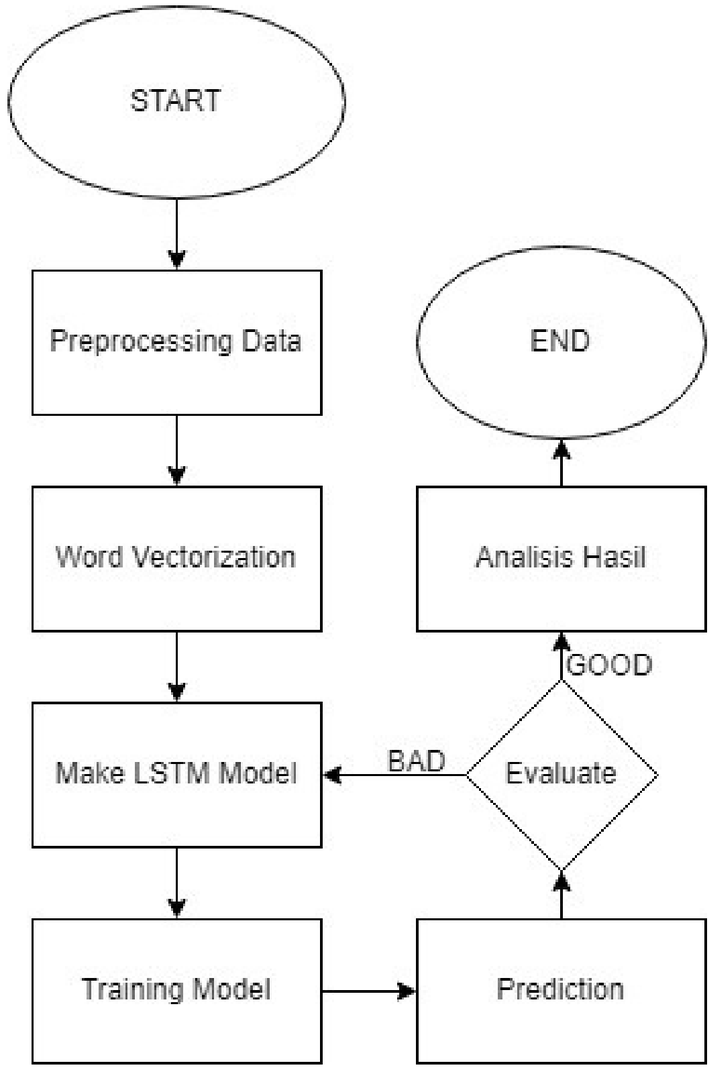
note: sw = stopwords

# Long Short-Term Memory



# Flowchart LSTM

1. Dataset akan dilakukan preprocessing
2. Dataset kemudian akan dirubah kedalam bentuk numerik
3. Membuat LSTM model dan penentuan parameter yang digunakan
4. Melakukan Pelatihan model
5. Melakukan Prediksi dari model yang sudah dilatih
6. Melakukan Evaluasi pada model jika hasil buruk akan dilakukan pembuatan model lagi
7. Melakukan Analisis Hasil dari hasil evaluasi



# tokenizer & word2vec

**Tokenizer** : Tokenisasi kata dengan mengambil 100.000 features

```
tokenizer = Tokenizer(num_words=max_features,split=' ')
tokenizer.fit_on_texts(total_data)
with open('tokenizer.pickle','wb') as handle:
    pickle.dump(tokenizer,handle,protocol=pickle.HIGHEST_PROTOCOL)
    print("tokenizer.pickle has created!")
X = tokenizer.texts_to_sequences(total_data)
# print(X)
```

**Word2Vec** : Menggunakan word2vec untuk melihat kata secara kontekstual

```
sentences = [text.split() for text in total_data]
word2vec_model = Word2Vec(sentences, vector_size=embedding_size, window=5, min_count=1, workers=4)
embedding_matrix = np.zeros((vocab_size+1, embedding_size))
for word, i in tokenizer.word_index.items():
    if word in word2vec_model.wv:
        embedding_matrix[i] = word2vec_model.wv[word]
```

# Model Summary BoW

| Layer (type) | Output Shape | Param # |
|---|---|---|
| lstm (LSTM) | (None, 64) | 4155904 |
| dense (Dense) | (None, 3) | 195 |

Total params: 4,156,099
Trainable params: 4,156,099
Non-trainable params: 0

## Parameter

unit/neuron : 64 unit
dropout: 0,2 / 20%
learning rate = 0.0001
patience = 5
epochs = 200
batch_size = 64

# Model Summary Word2Vec

| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding (Embedding) | (None, 96, 100) | 1619700 |
| lstm (LSTM) | (None, 64) | 42240 |
| dense (Dense) | (None, 3) | 195 |

Total params: 1,662,135
Trainable params: 1,662,135
Non-trainable params: 0

# Parameter

unit/neuron : 64 unit
dropout: 0,2 / 20%
learning rate = 0.0001
patience = 5
epochs = 200
batch_size = 64

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.82 | 0.85 | 0.84 | 693 |
| 1 | 0.85 | 0.73 | 0.79 | 209 |
| 2 | 0.91 | 0.92 | 0.92 | 1285 |
| accuracy | | | 0.88 | 2187 |
| macro avg | 0.86 | 0.83 | 0.85 | 2187 |
| weighted avg | 0.88 | 0.88 | 0.88 | 2187 |

Pakai Stopword

# Classification Report BoW

## Average Accuracy

# 87%

## Tanpa Stopword

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.87 | 0.86 | 693 |
| 1 | 0.85 | 0.76 | 0.80 | 209 |
| 2 | 0.92 | 0.93 | 0.93 | 1285 |
| accuracy | | | 0.89 | 2187 |
| macro avg | 0.87 | 0.85 | 0.86 | 2187 |
| weighted avg | 0.89 | 0.89 | 0.89 | 2187 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.83 | 0.83 | 693 |
| 1 | 0.83 | 0.79 | 0.81 | 209 |
| 2 | 0.91 | 0.92 | 0.92 | 1285 |
| accuracy | | | 0.88 | 2187 |
| macro avg | 0.86 | 0.85 | 0.85 | 2187 |
| weighted avg | 0.88 | 0.88 | 0.88 | 2187 |

Pakai Stopword

# Classification Report Word2Vec

Average Accuracy

**86%**

Tanpa Stopword

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.86 | 0.85 | 693 |
| 1 | 0.79 | 0.83 | 0.81 | 209 |
| 2 | 0.93 | 0.91 | 0.92 | 1285 |
| accuracy | | | 0.89 | 2187 |
| macro avg | 0.86 | 0.87 | 0.86 | 2187 |
| weighted avg | 0.89 | 0.89 | 0.89 | 2187 |

# Hasil K-Fold Word2Vec

Training ke- 0

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.84 | 0.80 | 693 |
| 1 | 0.74 | 0.71 | 0.73 | 209 |
| 2 | 0.93 | 0.88 | 0.90 | 1285 |
| accuracy | | | 0.85 | 2187 |
| macro avg | 0.81 | 0.81 | 0.81 | 2187 |
| weighted avg | 0.86 | 0.85 | 0.85 | 2187 |

Training ke- 1

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.86 | 0.80 | 693 |
| 1 | 0.77 | 0.67 | 0.72 | 209 |
| 2 | 0.93 | 0.88 | 0.91 | 1285 |
| accuracy | | | 0.86 | 2187 |
| macro avg | 0.82 | 0.81 | 0.81 | 2187 |
| weighted avg | 0.86 | 0.86 | 0.86 | 2187 |

Training ke- 2

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.85 | 0.82 | 693 |
| 1 | 0.81 | 0.69 | 0.75 | 209 |
| 2 | 0.92 | 0.91 | 0.91 | 1285 |
| accuracy | | | 0.87 | 2187 |
| macro avg | 0.84 | 0.82 | 0.83 | 2187 |
| weighted avg | 0.87 | 0.87 | 0.87 | 2187 |

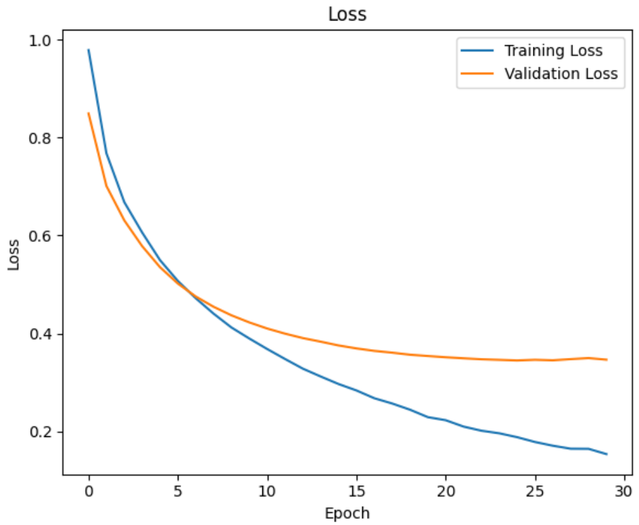
Training ke- 3

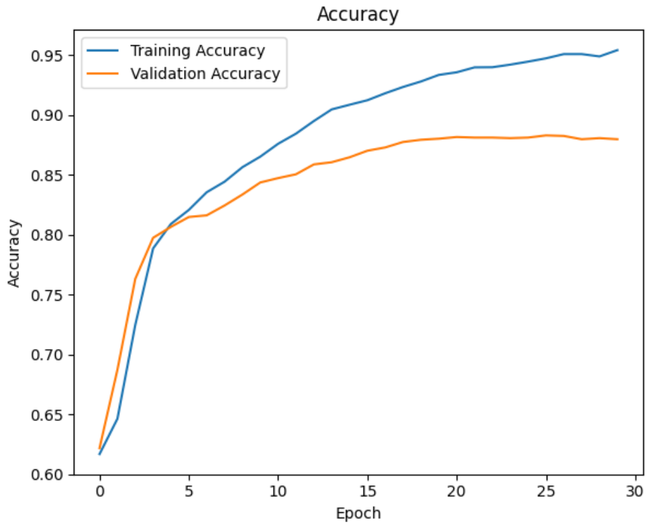
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.83 | 0.81 | 693 |
| 1 | 0.79 | 0.68 | 0.73 | 209 |
| 2 | 0.92 | 0.91 | 0.91 | 1285 |
| accuracy | | | 0.86 | 2187 |
| macro avg | 0.83 | 0.81 | 0.82 | 2187 |
| weighted avg | 0.86 | 0.86 | 0.86 | 2187 |

Training ke- 4

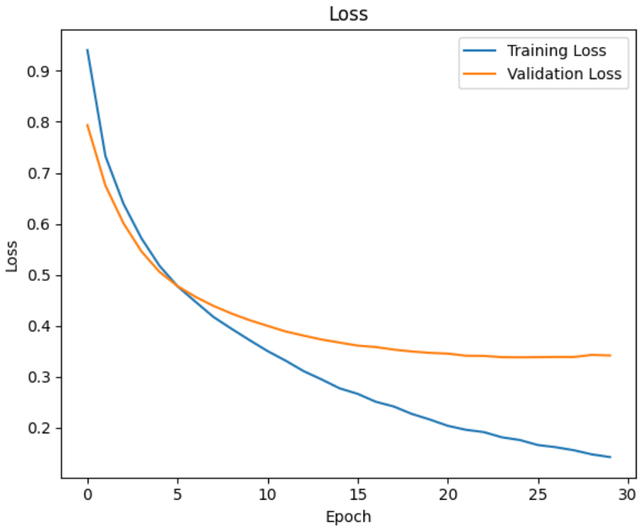
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.80 | 0.84 | 0.82 | 693 |
| 1 | 0.78 | 0.73 | 0.75 | 209 |
| 2 | 0.92 | 0.90 | 0.91 | 1285 |
| accuracy | | | 0.86 | 2187 |
| macro avg | 0.83 | 0.82 | 0.83 | 2187 |
| weighted avg | 0.87 | 0.86 | 0.86 | 2187 |

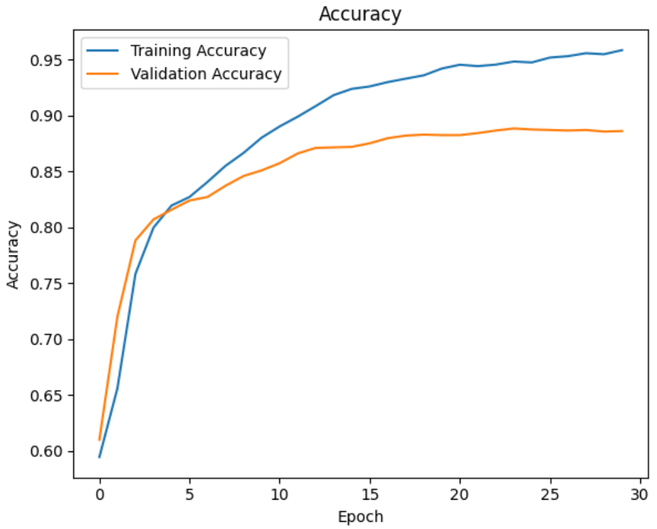
# Bag of Word




Pakai StopWord

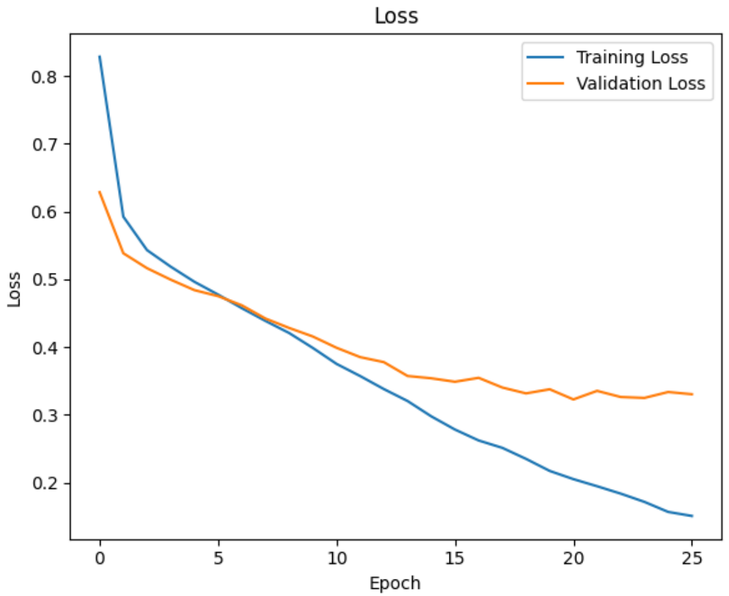


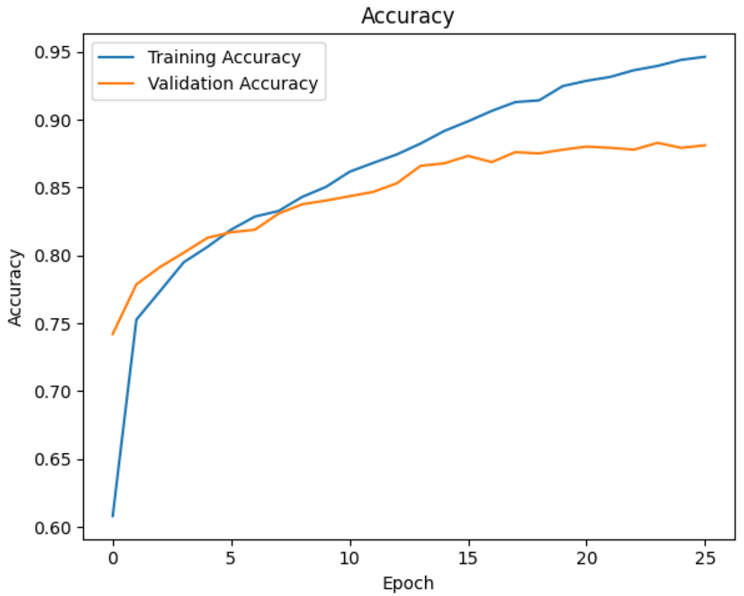




Tanpa StopWord

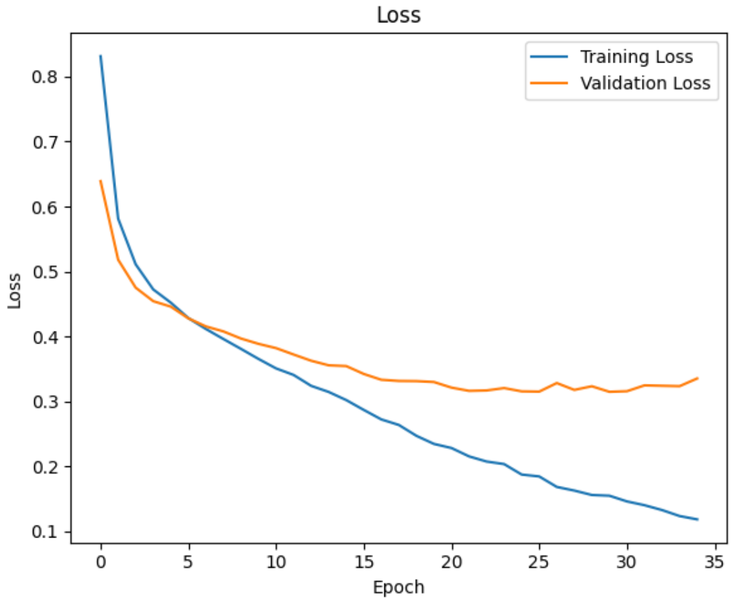
# Word2Vec + embedding

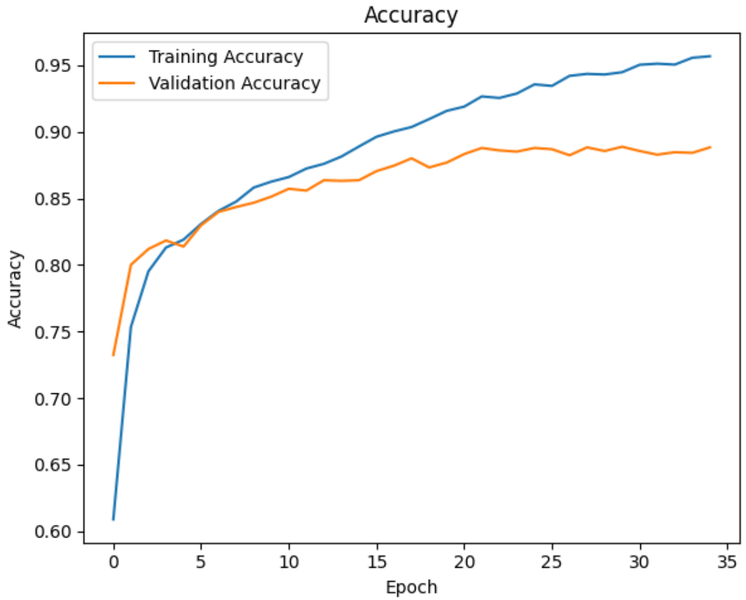
## Pakai Stopword




## Tanpa Stopword



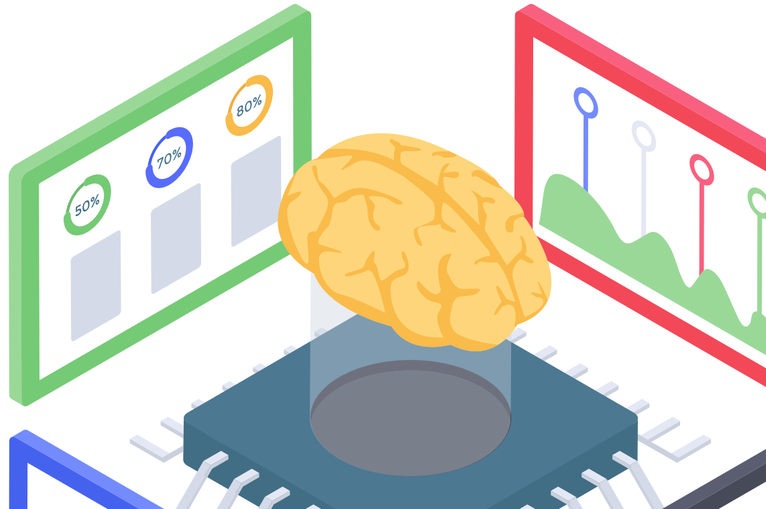




# Hasil Experiment Feature Extraction

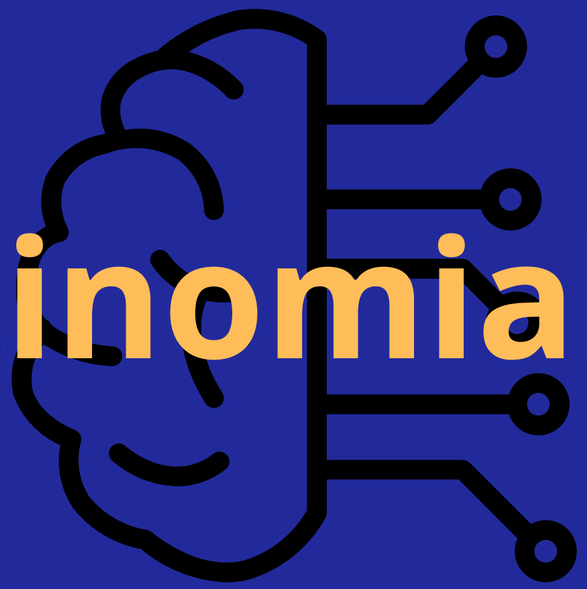
| | BoW w/o SW | BoW w/ SW | TFIDF w/o SW | TFIDF w/ SW | Word2Vec + embedding w/o SW | Word2Vec + embedding w/ SW |
|---|---|---|---|---|---|---|
| Accuracy | 0.89% | 0.88% | 0.59% | 0.59% | 0.89% | 0.88% |
| Average Accuracy | 0.87% | 0.86% | 0.58% | 0.58% | 0.86% | 0.85% |

- Dari hasil experiment yang didapat penggunaan Bag of Word dan Word2vec + embedding memiliki akurasi yang tinggi tetapi untuk akurasi secara keseluruhan setelah dilakukan Cross-Validation maka Bag of Word yang lebih unggul

- Penggunaan Stopword dapat mengurangi hasil akurasi saat dilakukan training dimana perbedaannya berkisar 1%



# MultinomialNB



# Feature  Extraction

**Feature extraction menggunakan Bag-Of-Word dengan menggunakan CountVectorizer dari Sklearn**

```
from sklearn.feature_extraction.text import CountVectorizer

count_vect = CountVectorizer()
count_vect.fit(data_preprocessed)

X = count_vect.transform(data_preprocessed)
print ("Feature Extraction selesai")
```

Pembuatan model dengan membagi data 8:2

80% untuk training dan 20% untuk validasi

Model dibuat dengan menggunakan Multinomial Naive Bayes

```
X_train, X_test, y_train, y_test = train_test_split(X, classes, test_size = 0.2)

from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train, y_train)

print ("Training selesai")
```

# Hasil Training

| - | precision | recall | f1-score | support |
|---|---|---|---|---|
| negative | 0.76 | 0.72 | 0.74 | 76 |
| neutral | 0.85 | 0.58 | 0.69 | 48 |
| positive | 0.75 | 0.93 | 0.83 | 76 |
| accuracy | | | 0.77 | 200 |
| macro avg | 0.79 | 0.75 | 0.76 | 200 |
| weighted avg | 0.78 | 0.77 | 0.76 | 200 |

Hasil Training menunjukan akurasi yang didapat adalah 77% namun ketika dilakukan k-fold validation didapatkan rata" akurasi sebesar 84%

# Hasil K-Fold

Training ke- 1

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| negative | 0.74 | 0.78 | 0.76 | 680 |
| neutral | 0.89 | 0.69 | 0.78 | 239 |
| positive | 0.88 | 0.89 | 0.88 | 1281 |
| accuracy | | | 0.83 | 2200 |
| macro avg | 0.84 | 0.79 | 0.81 | 2200 |
| weighted avg | 0.84 | 0.83 | 0.83 | 2200 |

==========

Training ke- 2

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| negative | 0.76 | 0.79 | 0.78 | 706 |
| neutral | 0.84 | 0.64 | 0.72 | 220 |
| positive | 0.88 | 0.90 | 0.89 | 1274 |
| accuracy | | | 0.84 | 2200 |
| macro avg | 0.83 | 0.78 | 0.80 | 2200 |
| weighted avg | 0.84 | 0.84 | 0.84 | 2200 |

==========

Training ke- 3

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| negative | 0.76 | 0.79 | 0.77 | 682 |
| neutral | 0.90 | 0.68 | 0.78 | 215 |
| positive | 0.89 | 0.91 | 0.90 | 1303 |
| accuracy | | | 0.85 | 2200 |
| macro avg | 0.85 | 0.79 | 0.81 | 2200 |
| weighted avg | 0.85 | 0.85 | 0.85 | 2200 |

Training ke- 4

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| negative | 0.76 | 0.80 | 0.78 | 698 |
| neutral | 0.91 | 0.68 | 0.78 | 229 |
| positive | 0.88 | 0.90 | 0.89 | 1273 |
| accuracy | | | 0.84 | 2200 |
| macro avg | 0.85 | 0.79 | 0.82 | 2200 |
| weighted avg | 0.85 | 0.84 | 0.84 | 2200 |

==========

Training ke- 5

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| negative | 0.76 | 0.84 | 0.80 | 670 |
| neutral | 0.89 | 0.63 | 0.74 | 245 |
| positive | 0.91 | 0.91 | 0.91 | 1285 |
| accuracy | | | 0.86 | 2200 |
| macro avg | 0.85 | 0.79 | 0.81 | 2200 |
| weighted avg | 0.86 | 0.86 | 0.86 | 2200 |

==========

# Kesimpulan

MLPClassifier dengan menggunakan feature extraction Bag-of-Word dan implementasi Stopword terbukti dapat memilah sentiment dengan lebih baik dengan rata" akurasi sekitar 87% dimana akurasi tersebut yang tertinggi diantara beberapa experiment yang sudah dilakukan. MLPClassifier juga lebih unggul dari pada LSTM yang menggunakan metode feature extraction yang sama dengan tingkat rata" akurasi lebih rendah 1% yaitu di angka 86%

# Saran

- Dataset untuk pelatihan diperbanyak agar dapat menggunakan arsitektur NN dan LSTM yang lebih kompleks dan model dapat mengenali pola lebih baik sehingga dapat memilah sentiment dengan lebih baik lagi
- Distribusi Label untuk pelatihan lebih diratakan agar model dapat belajar dengan lebih baik