NATURAL LANGUAGE PROCESSING (NLP)

AI MASTERY

ANALISIS SENTIMEN PILKADA DKI JAKARTA 2017 PADA TWITTER MENGGUNAKAN MACHINE LEARNING

Ditulis oleh Giselle Halim (Kelas Tensor)

AI MASTERY

NATURAL LANGUAGE PROCESSING (NLP)

Materi Pembahasan

Latar Belakang
Rumusan Masalah
Data yang digunakan
Persebaran Data
Text Preprocessing
Feature Extraction
Modelling
Performa dan Evaluasi Model
Kesimpulan

02

Latar Belakang

Ungkapan-ungkapan pada sosial media dapat mencerminkan opini dan perasaan dari masyarakat mengenai suatu hal. Jika dapat diolah dengan tepat, maka data-data seperti komentar dan postingan berbentuk teks dapat menjadi alat bantu dalam pengambilan keputusan.

Salah satu sosial media yang sering digunakan dan digemari masyarakat adalah Twitter. Pada Twitter, kita dapat menggunakan data tweet atau retweet dari suatu user untuk mengetahui sentimen masyarakat mengenai suatu topik. Kegiatan ini dinamakan sentiment analysis atau analisis sentimen.

Sentiment analysis adalah kegiatan yang bertujuan untuk menganalisa data untuk mengetahui sentimen (positive, negative, atau netral) mengenai suatu hal. Pada penelitian ini, akan dilakukan sentiment analysis mengenai topik Pilkada DKI Jakarta 2017 di Twitter.

Rumusan Masalah

Rumusan masalah pada penelitian ini adalah:

- Bagaimana cara memproses text tweet agar menjadi data yang dapat dianalisa oleh model?
- Bagaimana cara menerapkan machine learning untuk melakukan sentiment analysis?
- Bagaimana hasil dan performa algoritma yang telah digunakan?

Data yang digunakan

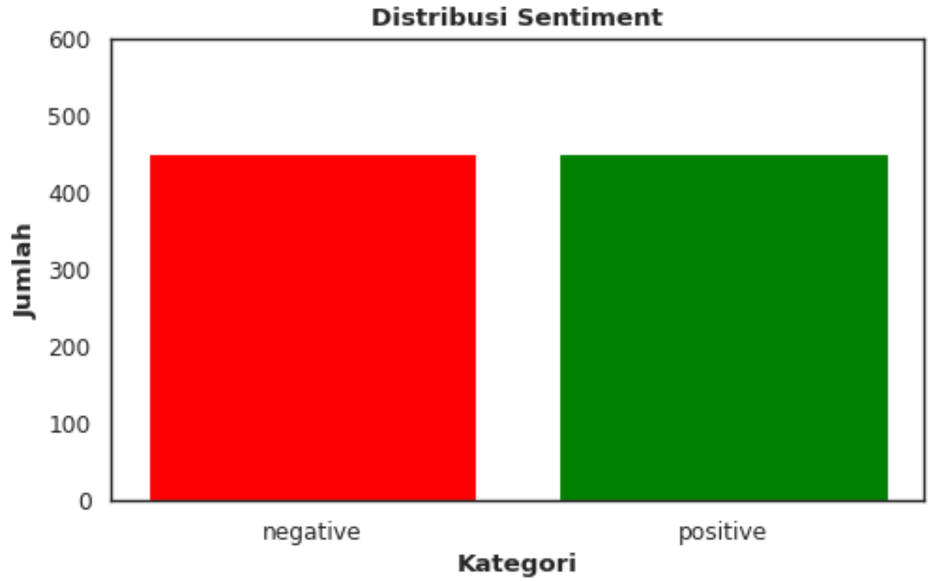
Dataset yang digunakan pada tugas ini adalah dataset sentimen mengenai pelaksanaan Pilkada DKI Jakarta pada tahun 2017 di Twitter. Dataset ini memiliki 900 dokumen tweet yang berisi cuitan mengenai 3 pasangan calon, yaitu Agus-Sylvi, Ahok-Djarot, dan Anies-Sandi. Dataset ini memiliki 2 kelas sentimen, yaitu positive dan negative.

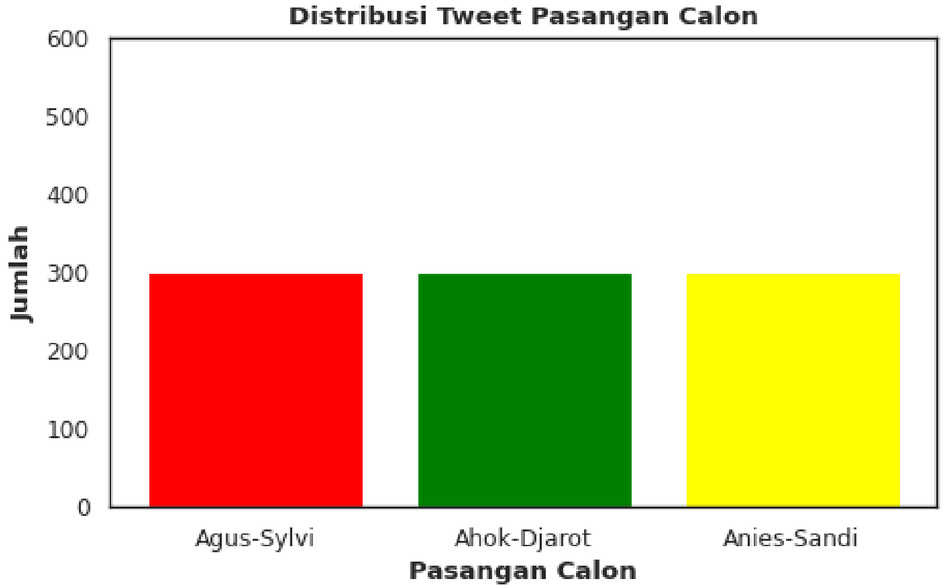
Dataset terdiri dari kolom Id, Sentiment, Pasangan Calon, dan Text Tweet.

Untuk melakukan pelatihan model, dataset didefinisikan polaritasnya dengan nilai positive sebesar 1 dan nilai negative sebesar -1.

Persebaran Data

Dataset memiliki data yang seimbang dengan sentiment positive dan negative yang masing-masing berjumlah 450. Data tweet terkait calon pasangan juga seimbang dengan masing-masing sebanyak 300 data.
Persebaran data dapat dilihat pada kedua grafik disamping.





NATURAL LANGUAGE PROCESSING (NLP)

AI MASTERY

Text Preprocessing

Case Folding

Pengubahan teks menjadi seluruhnya lowercase, menghapus angka, URL, dan tanda baca.

Stopwords

Menghapus stopwords pada data tweet sesuai dengan list stopwords yang telah ditentukan dalam file csv khusus.

Stemming

Proses pengubahan kata berimbuhan menjadi kata dasar. Karena dataset berbahasa Indonesia maka digunakan library sastrawi yang hanya bisa melakukan stemming.

Pipeline

Penerapan fungsi-fungsi text preprocessing yang telah dibuat sebelumnya pada dataset dan menambahkan kolom baru untuk melihat data tweet yang sudah bersih.

```python
import re

#Casefolding
def casefolding(text):
  text = text.lower()                                  #Change all text to lowercase
  text = re.sub(r'https?://\S+|www\.\S+', '', text) #Delete URLs
  text = re.sub(r'[-+]?[0-9]+', '', text)              #Delete numbers
  text = re.sub(r'[^\w\s]','', text)                   #Delete punctuations
  text = text.strip()
  return text
```

- Casefolding (kiri atas)
- Stemming (kiri bawah)
- Remove stopwords (kanan bawah)

```python
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

factory = StemmerFactory()
stemmer = factory.create_stemmer()

#Stemming
def stemming(text):
  text = stemmer.stem(text)
  return text
```

```python
#Removing stopwords

def remove_stop_words(text):
  clean_words = []
  text = text.split()
  for word in text:
      if word not in stopwords_ind:
          clean_words.append(word)
  return " ".join(clean_words)
```

Feature Extraction

Pada tahap feature extraction, dilakukan pemisahan fitur (X) dan target (y). Data tweet yang telah dibersihkan ditentukan menjadi X dan label data sentimen ditentukan menjadi y.

Digunakan TF-IDF dan n-gram untuk mengubah kata menjadi vektor, dengan n-gram range (1,1). Setelah itu, digunakan Chi Square untuk melakukan feature selection.

Pada feature selection, dipilih 1000 fitur dari 2795 fitur.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

#Ten features with highest chi-squared statistics are selected
chi2_features = SelectKBest(chi2, k=1000)
X_kbest_features = chi2_features.fit_transform(X, y)

#Reduced features
print('Original feature number:', X.shape[1])
print('Reduced feature number:', X_kbest_features.shape[1])
```

Pengurangan feature dengan Chi Square

Modelling

Model yang digunakan untuk melakukan sentiment analysis pada kasus ini adalah model yang dibuat dengan algoritma Naive Bayes dan Support Vector Machine (SVM). Seluruh algoritma ini termasuk dalam algoritma klasifikasi yang merupakan supervised learning. Penggunaan kedua algortima tersebut ditujukan untuk melakukan perbandingan akurasi dalam melakukan sentiment analysis.

Pada algoritma Naive Bayes, digunakan tipe BernoulliNB yang merupakan algoritma Naive Bayes yang lebih banyak digunakan untuk text classification.

Sebelum membuat model, dilakukan pembagian data training dan data testing dengan rasio 80:20.

Algoritma Naive Bayes (BernoulliNB)

```
from sklearn.naive_bayes import BernoulliNB
#Training the model
algorithm = BernoulliNB()
model = algorithm.fit(X_train, y_train)
```

Algoritma SVM (SVC)

```
from sklearn.svm import SVC
#Training the model
algorithm2 = SVC(gamma=0.01, C=100, kernel='rbf')
model2 = algorithm2.fit(X_train, y_train)
```

Performa dan Evaluasi

Pada hasil percobaan di data testing, kedua algoritma menunjukkan angka akurasi pengujian yang berbeda, walau perbedaannya tidak terlalu jauh. Metode cross validation yang digunakan adalah StratifiedShuffleCross.

Naive Bayes memiliki akurasi testing sebesar 87.7% dengan jumlah prediksi benar sebanyak 158 dan prediksi salah sebanyak 22. Saat dilakukan cross validation, akurasi rata-ratanya menjadi 85.7%.

SVM memiliki akurasi testing sebesar 88.8% dengan jumlah prediksi benar sebanyak 160 dan prediksi salah sebanyak 20. Saat dilakukan cross validation, akurasi rata-rata menjadi 84.4%.

Performa dan Evaluasi

Pada classification report, nilai precision, recall, dan F1 score kedua algoritma cukup baik.

Algoritma Naive Bayes memiliki perbedaan sekitar 12% pada nilai precision dan recall untuk label positive dan negative, F1 score sebesar 88% dan akurasi sebesar 88%.

Algoritma SVM memiliki perbedaan yang sedikit pada nilai precision, recall, dan F1 scorenya untuk label positive dan negativenya, dengan perbedaan sekitar 1-2%. Nilai akurasi yang didapatkan adalah sebesar 87%.

Evaluasi Performa Algoritma Naive Bayes (BernoulliNB)

```python
#Predicted vs Actual Label
prediksi_benar = (model_pred == y_test).sum()
prediksi_salah = (model_pred != y_test).sum()

print('Jumlah prediksi benar\t:', prediksi_benar)
print('Jumlah prediksi salah\t:', prediksi_salah)

accuracy = prediksi_benar / (prediksi_benar + prediksi_salah)*100
print('Akurasi pengujian\t:', accuracy, '%')
```

```
Jumlah prediksi benar   : 158
Jumlah prediksi salah   : 22
Akurasi pengujian       : 87.77777777777777 %
```

```python
# Cross Validation
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.model_selection import cross_val_score

cv = StratifiedShuffleSplit(n_splits=10, test_size=0.2, random_state=42)

cv_accuracy = (cross_val_score(model, X_kbest_features, y, cv=cv, scoring='accuracy'))
avg_accuracy = np.mean(cv_accuracy)

print('Akurasi setiap split:', cv_accuracy, '\n')
print('Rata-rata akurasi pada cross validation:', avg_accuracy)
```

```
Akurasi setiap split: [0.88333333 0.83333333 0.81111111 0.88888889 0.87777778 0.82777778
 0.86111111 0.83888889 0.86111111 0.87222222]

Rata-rata akurasi pada cross validation: 0.8555555555555555
```

```python
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, model_pred)
print('Confusion matrix:\n', cm)
```

```
Confusion matrix:
 [[78 17]
 [ 5 80]]
```

```
Classification report:
              precision    recall  f1-score   support

          -1       0.94      0.82      0.88        95
           1       0.82      0.94      0.88        85

    accuracy                           0.88       180
   macro avg       0.88      0.88      0.88       180
weighted avg       0.89      0.88      0.88       180
```

Evaluasi Performa Algoritma SVM (SVC)

```
prediksi_benar = (model_pred2 == y_test).sum()
prediksi_salah = (model_pred2 != y_test).sum()

print('Jumlah prediksi benar\t:', prediksi_benar)
print('Jumlah prediksi salah\t:', prediksi_salah)

accuracy = prediksi_benar / (prediksi_benar + prediksi_salah)*100
print('Akurasi pengujian\t:', accuracy, '%')
```

```
Jumlah prediksi benar   : 157
Jumlah prediksi salah   : 23
Akurasi pengujian       : 87.22222222222223 %
```

```
# Cross Validation

from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.model_selection import cross_val_score

cv = StratifiedShuffleSplit(n_splits=10, test_size=0.2, random_state=42)

cv_accuracy = (cross_val_score(model2, X_kbest_features, y, cv=cv, scoring='accuracy'))
avg_accuracy = np.mean(cv_accuracy)

print('Akurasi setiap split:', cv_accuracy, '\n')
print('Rata-rata akurasi pada cross validation:', avg_accuracy)
```

```
Akurasi setiap split: [0.83888889 0.81111111 0.80555556 0.88333333 0.86111111 0.77777778
 0.84444444 0.85555556 0.87222222 0.86111111]

Rata-rata akurasi pada cross validation: 0.8411111111111109
```

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, model_pred2)
print('Confusion matrix:\n', cm)
```

```
Confusion matrix:
 [[84 11]
 [12 73]]
```

```
Classification report:
              precision    recall  f1-score   support

          -1       0.88      0.88      0.88        95
           1       0.87      0.86      0.86        85

    accuracy                           0.87       180
   macro avg       0.87      0.87      0.87       180
weighted avg       0.87      0.87      0.87       180
```

Kesimpulan

- Perbedaan kedua algoritma secara performa tidak terlalu jauh, tetapi pada kasus ini, algoritma Naive Bayes lebih baik untuk melakukan sentiment analysis karena akurasinya yang sedikit lebih tinggi.

- Text preprocessing yang tepat akan mempengaruhi akurasi prediksi machine learning.

- Tipe algoritma yang digunakan juga berpengaruh pada akurasi, sehingga pada kasus ini digunakan Naive Bayes tipe Bernoulli yang umum digunakan untuk text classification dan untuk SVM digunakan SVC (khusus klasifikasi).

THANK YOU!

Google Colab Link