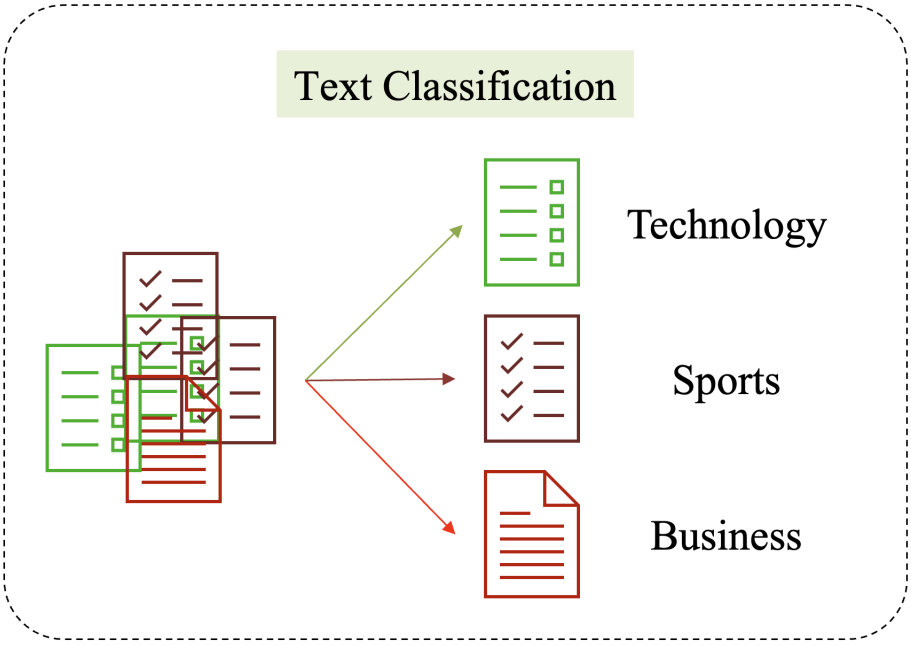
AI VIET NAM – COURSE 2024

Sentiment Analysis - Project

Ngày 13 tháng 9 năm 2024

Phần I: Giới thiệu

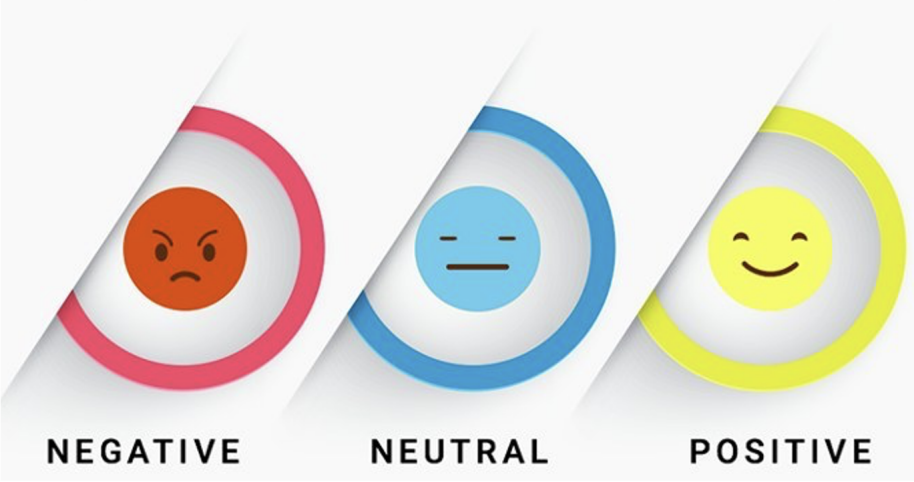
Trong bài tập này, chúng ta sẽ thực hành các nội dung về phân tích cảm xúc của khách hàng với các bình luận đánh giá phim dựa vào các phương pháp tiếp cận cho bài toán phân loại văn bản điển hình. Bài toán phân loại được lấy ví dụ như hình dưới. Với mỗi đơn vị văn bản sẽ được phân loại với một nhãn cụ thể thuộc tập hợp các nhãn cho trước.



Hình 1: Text Classification.

Với mỗi đơn vị văn bản có thể thuộc vào tập các nhãn như: 'Technology', 'Sports', 'Business',...

Sentiment Analysis là nhóm các bài toán con thuộc vào phân loại văn bản. Với mục tiêu phân tích và đánh giá các bình luận của khách hàng cho các sản phẩm và tích cực, tiêu cực hay trung tính.



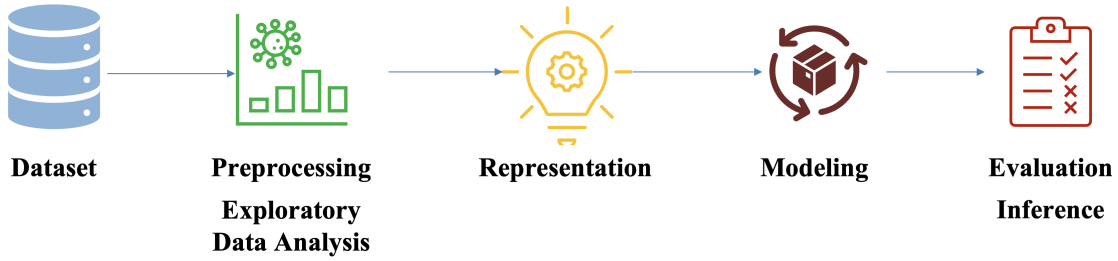
Hình 2: Sentiment Analysis.

Phần project này sẽ tập trung giải quyết cho bài phân tích cảm xúc trên bộ dữ liệu IMDB - Đánh giá phim.

Phần II: Bài tập

A. Phần lập trình

Trong phần này, chúng ta sẽ cài đặt và huấn luyện hai mô hình Decision Tree và Random Forest để giải quyết bài toán phân tích cảm xúc gồm các bước như hình sau:



Hình 3: Các bước huấn luyện mô hình phân loại.

1. **Tải bộ dữ liệu:** Các bạn tải bộ dữ liệu IMDB-Dataset.csv tại đây.

2. **Đọc bộ dữ liệu:** Sử dụng thư viện pandas, chúng ta sẽ đọc file .csv lên như sau:

```
# Load dataset
import pandas as pd

df = pd.read_csv('./IMDB-Dataset.csv')

# Remove duplicate rows
df = df.drop_duplicates()
```

Ở đây chúng ta sẽ thực hiện làm sạch dữ liệu thông qua các bước như: xoá thẻ html, xoá dấu câu, xoá số, xoá các icon,...

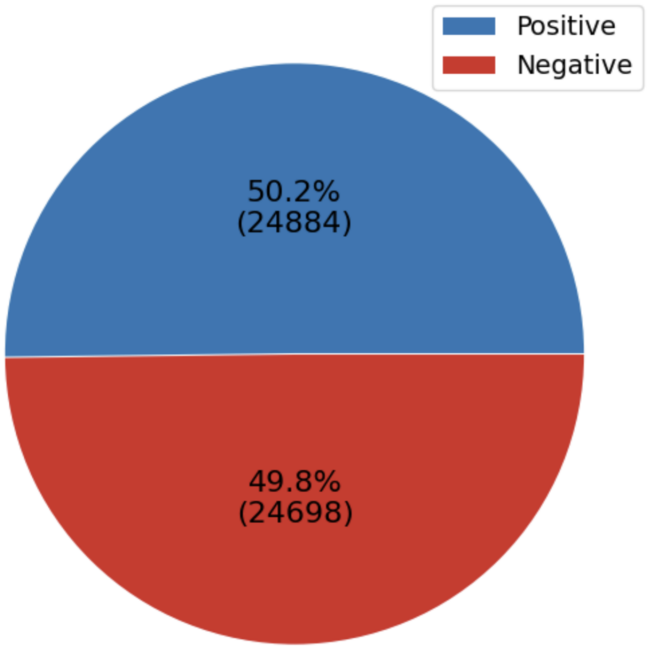
```
import re
import string
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from bs4 import BeautifulSoup
import contractions

stop = set(stopwords.words('english'))

# Expanding contractions
def expand_contractions(text):
    return contractions.fix(text)

# Function to clean data
def preprocess_text(text):

    wl = WordNetLemmatizer()

    soup = BeautifulSoup(text, "html.parser") # Removing html tags
```

```
    text = soup.get_text()
    text = expand_contractions(text) # Expanding chatwords and contracts clearing contractions
    emoji_clean = re.compile("["
                             u"\U0001F600-\U0001F64F"  # emoticons
                             u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                             u"\U0001F680-\U0001F6FF"  # transport & map symbols
                             u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                             u"\U00002702-\U000027B0"
                             u"\U000024C2-\U0001F251"
                             "]+", flags=re.UNICODE)
    text = emoji_clean.sub(r'',text)
    text = re.sub(r'\.(?=\S)', '. ',text) #add space after full stop
    text = re.sub(r'http\S+', '', text) #remove urls
    text = "".join([
        word.lower() for word in text if word not in string.punctuation
    ]) #remove punctuation and make text lowercase
    text = " ".join([
        wl.lemmatize(word) for word in text.split() if word not in stop and word.isalpha()]) #lemmatize
    return text

df['review'] = df['review'].apply(preprocess_text)
```

3. **Phân tích dữ liệu:** Thống kê số lượng các nhãn trong bộ dữ liệu:

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Creating autocpt arguments
def func(pct, allvalues):
    absolute = int(pct / 100.*np.sum(allvalues))
    return "{:.1f}%\n({:d})".format(pct, absolute)

freq_pos = len(df[df['sentiment'] == 'positive'])
freq_neg = len(df[df['sentiment'] == 'negative'])

data = [freq_pos, freq_neg]

labels = ['positive', 'negative']
# Create pie chart
pie, ax = plt.subplots(figsize=[11,7])
plt.pie(x=data, autopct=lambda pct: func(pct, data), explode=[0.0025]*2, pctdistance=0.5, colors=[sns.color_palette()[0],'tab:red'],textprops={'fontsize': 16})
# plt.title('Frequencies of sentiment labels', fontsize=14, fontweight='bold')
labels = [r'Positive', r'Negative']
plt.legend(labels, loc="best", prop={'size': 14})
pie.savefig("PieChart.png")
plt.show()
```

Kết quả thu được:

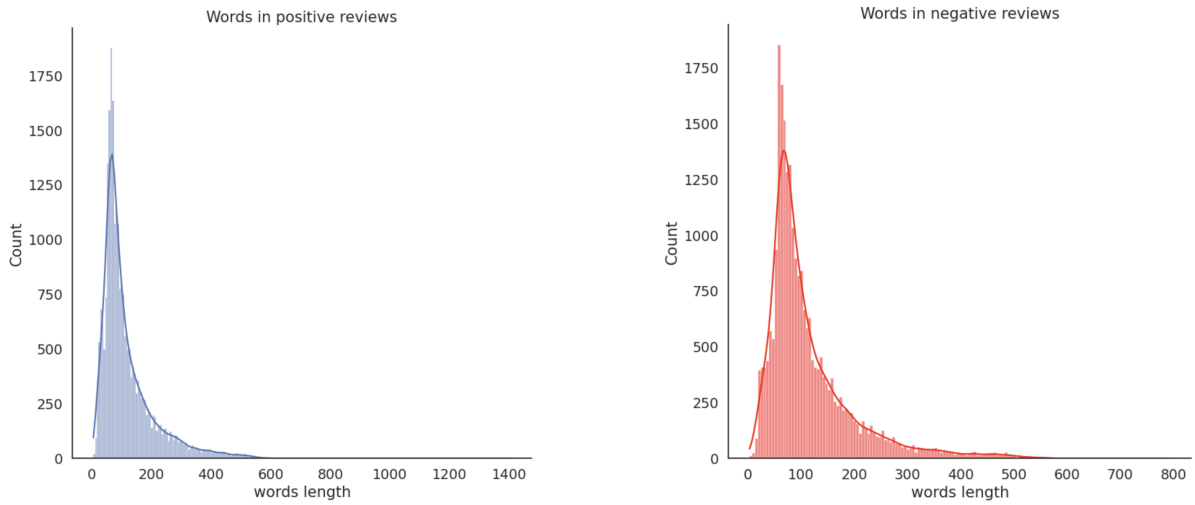


Hình 4: Số lượng các nhãn trong bộ dữ liệu IMDB.

Thống kê độ dài của các mẫu cho mỗi class.

```
words_len = df['review'].str.split().map(lambda x: len(x))
df_temp = df.copy()
df_temp['words length'] = words_len

hist_positive = sns.displot(
    data=df_temp[df_temp['sentiment'] == 'positive'],
    x="words length", hue="sentiment", kde=True,  height=7, aspect=1.1, legend=
    False
).set(title='Words in positive reviews')
plt.show(hist_positive)

hist_negative = sns.displot(
    data=df_temp[df_temp['sentiment'] == 'negative'],
    x="words length", hue="sentiment", kde=True,  height=7, aspect=1.1, legend=
    False,  palette=['red']
).set(title='Words in negative reviews')
plt.show(hist_negative)

plt.figure(figsize=(7,7.1))
kernel_distibution_number_words_plot  = sns.kdeplot(
    data=df_temp, x="words length", hue="sentiment", fill=True,  palette=[sns.
    color_palette()[0],'red']
).set(title='Words in reviews')
plt.legend(title='Sentiment', labels=['negative', 'positive'])
plt.show(kernel_distibution_number_words_plot)
```

Kết quả thu được:



Hình 5: Số lượng các nhãn trong bộ dữ liệu IMDB.

4. **Chia tập train và test:**

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder

label_encode = LabelEncoder()
y_data = label_encode.fit_transform(df['sentiment'])

x_train, x_test, y_train, y_test = train_test_split(
    x_data, y_data, test_size=0.2, random_state=42
)
```

5. **Biểu diễn văn bản thành vector:**

```
tfidf_vectorizer = TfidfVectorizer(max_features=10000)
tfidf_vectorizer.fit(x_train, y_train)

x_train_encoded = tfidf_vectorizer.transform(x_train)
x_test_encoded = tfidf_vectorizer.transform(x_test)
```

6. **Huấn luyện và đánh giá mô hình:**

Ta thực hiện huấn luyện mô hình với bộ dữ liệu train. Để huấn luyện mô hình Decision Tree, các bạn sẽ sử dụng DecisionTreeClassifier():

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

dt_classifier = DecisionTreeClassifier(
    criterion='entropy',
    random_state=42
)
dt_classifier.fit(x_train_encoded, y_train)
y_pred = dt_classifier.predict(x_test_encoded)
accuracy_score(y_pred, y_test)
```

Để huấn luyện mô hình Random Forest, các bạn sẽ sử dụng RandomForestClassifier():

```
rf_classifier = RandomForestClassifier(
    random_state=42
)
rf_classifier.fit(x_train_encoded, y_train)
y_pred = rf_classifier.predict(x_test_encoded)
accuracy_score(y_pred, y_test)
```

Bên cạnh các mô hình như Decision Tree hoặc Random Forest, chúng ta có thể mở rộng đánh giá và so sánh với các mô hình học máy khác như AdaBoost, Gradient Boosting và XGBoost. Từ đó xác định xem các mô hình nào sẽ hiệu quả và có độ chính cao trên bài toán phân loại với bộ dữ liệu IMDB.

B. Phần trắc nghiệm

1. Độ đo đánh giá mô hình cho bài toán phân loại văn bản là?

   (a) Accuracy
   (c) ROUGE
   (b) BLEU
   (d) COMET

2. Bước nào không thuộc vào mô hình phân loại ở phần 2?

   (a) Modeling
   (c) Text Representation
   (b) Text Preprocessing
   (d) Data Crawling

3. Mô hình nào có thể được sử dụng làm bộ phân loại cho bài toán phân loại văn bản?

   (a) Decision Tree Classifier
   (c) TF-IDF
   (b) Decision Tree Regressor
   (d) BoW

4. Phương pháp biểu diễn văn bản thành vector được sử dụng trong phần thực nghiệm là?

   (a) One-hot Encoding
   (c) TF-IDF
   (b) Bag Of Word
   (d) Bag Of N-gram

5. Bộ dữ liệu được sử dụng cho bài toán phân loại trong phần thực nghiệm là?

   (a) IMDB
   (c) Play Tennis
   (b) IRIS
   (d) OPUS

6. Số lượng sample trùng lặp trong bộ dữ liệu phân loại được sử dụng trong phần thực nghiệm là?

   (a) 418
   (c) 518
   (b) 318
   (d) 218

7. Kích thước bộ từ điển sau bước tiền xử lý trong phần thực nghiệm xấp xỉ với kết quả nào sau đây?

   (a) 100000 từ
   (c) 140000 từ
   (b) 120000 từ
   (d) 160000 từ

8. Khai báo nào sau đây là đúng?

   (a) TfidfVectorizer(max_features=10000)
   (c) TfidfVectorizer(max_feature=10000)
   (b) TfidfVector(max_features=10000)
   (d) TfidfVector(max_features=10000)

9. Kết quả độ chính xác accuracy của mô hình Random Forest Classifier trên xấp xỉ với kết quả nào sau đây?

   (*a*) 64
   (*b*) 74
   (*c*) 84
   (*d*) 94

10. Lớp LabelEncoder có hàm nào sau đây?

    (*a*) inverse_transform()
    (*b*) inverse_transforms()
    (*c*) fit_transforms()
    (*d*) fit_data()

- Hết -