مشروع لمقرر التعلم الآلي

رمز المقرر CS461

تصنيف الرسائل الإلكترونية للكشف عن البريد المزعج

Categorize spam detection emails

إعداد: سليمان حسن القره مانلى

ربيع - 2023

Background:

في العالم الحديث، أصبح التصنيف التلقائي للنصوص من الأنشطة الهامة في مجال استخدامات التعلم الآلي. يستخدم تصنيف النصوص لتحديد فئة أو تصنيف لنص معين استنادًا إلى محتواه. من بين التطبيقات الشائعة لتصنيف النصوص تصنيف البريد الإلكتروني إلى الرسائل الإلكترونية المزعجة (السبام) والرسائل الإلكترونية العادية.

Method:

تم استخدام مجموعة من النماذج لتنفيذ تصنيف البريد الإلكتروني في هذا الكود. تضمن ذلك نموذجين رئيسيين: نموذج Naive Bayes ونموذج Logistic Regression. تم استخدام تقنية Bag-of-Words مع تحويل النصوص إلى تمثيل رقمي باستخدام CountVectorizer.

Experiment:

قام البرنامج بقراءة بيانات البريد الإلكتروني من ملف CSV وتنظيفها من القيم المفقودة. ثم قام بتدريب كلٍ من النموذجين باستخدام البيانات النصية المتاحة. تم قياس دقة النماذج باستخدام مصفوفة الارتباك (Confusion Matrix) وحساب الدقة (Accuracy Score).

Conclusion:

توصل البرنامج إلى أن نموذج Naive Bayes حقق متوسط دقة يبلغ حوالي 98.22%، بينما حقق نموذج Logistic Regression متوسط دقة يبلغ حوالي 99.84%. يمكن استخدام هذه النتائج لتصنيف البريد الإلكتروني بشكل فعال وتحديد ما إذا كان البريد الإلكتروني مزعجًا (سبام) أم لا.

References:

- Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
- A Comparison of Event Models for Naive Bayes Text Classification", Mccallum, A., Nigam, K. in AAAI-98 Workshop on Learning for Text Categorization, 1998.
- Logistic Regression", Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Logistic_regression . Accessed: Mar. 20, 2024.

توثيق الكود الخاص بتصنيف البريد الإلكتروني

المقدمة:

يعتبر كود تصنيف البريد الإلكتروني برنامجًا مكتوبًا بلغة Python يهدف إلى تصنيف الرسائل الإلكترونية إلى فئتين: الرسائل المزعجة والرسائل غير المزعجة باستخدام نموذجين في تعلم الآلة: نموذج نايف بايز ونموذج الانحدار اللوجستي. يهدف هذا الوثائق إلى توثيق الكود بشكل مفصل لفهم كيفية عمله والاعتمادات المستخدمة.

الاعتمادات:

- pandas: لتلاعب البيانات وقراءة ملفات CSV.
- scikit-learn: لتنفيذ خوارزميات تعلم الآلة وتقييم أدائها.

بنية الكود:

يتكون الكود من classes التالية:

1. نموذج NaiveBayesModel:
   - يعرف هذا النموذج نموذج نايف بايز المستخدم لتصنيف البريد الإلكتروني.
   - يبدأ بتهيئة نموذج Multinomial Naive Bayes وكائن CountVectorizer.
   - الطرق:
     - `train(X_train, y_train)`: يدرب النموذج نايف بايز على البيانات التدريبية المقدمة.
     - `predict(X_test)`: يقوم بعمل توقعات على بيانات الاختبار باستخدام النموذج المدرب.

2. نموذج LogisticRegressionModel:
   - يعرف هذا النموذج نموذج الانحدار اللوجستي المستخدم لتصنيف البريد الإلكتروني.
   - يبدأ بتهيئة نموذج الانحدار اللوجستي مع حد أقصى لعدد الدورات يساوي 1100 وكائن CountVectorizer.
   - الطرق:
     - `train(X_train, y_train)`: يدرب نموذج الانحدار اللوجستي على البيانات التدريبية المقدمة.
     - `predict(X_test)`: يقوم بعمل توقعات على بيانات الاختبار باستخدام النموذج المدرب.

3. فئة EmailClassifier:

- تدير هذه الفئة عملية تصنيف البريد الإلكتروني بأكملها.

- يتم تهيئتها مع مسار الملف الذي تحتوي عليه البيانات و يتم تهيئة Object من NaiveBayesModel و LogisticRegressionModel.

- الطرق:

- `read_data()`: يقرأ البيانات البريد الإلكتروني من ملف CSV محدد بواسطة مسار البيانات.

- `clean_data()`: يقوم بتنظيف البيانات عن طريق ملء القيم المفقودة وإزالة الصفوف ذات قيم نصية مفقودة.

- `train_models()`: يقوم بتدريب نموذجي النايف بايز والانحدار اللوجستي على البيانات المنظفة وطباعة مصفوفة الارتباك لكل نموذج.

- `predict_single_text(text)`: يتنبأ بتسميات الفئة (مزعجة أو غير مزعجة) لنص واحد مدخل من قبل المستخدم باستخدام النموذجين.

4. الكتلة الرئيسية للتنفيذ:

- في الكتلة الرئيسية، يتم إنشاء مثيل لفئة EmailClassifier، ثم يتم قراءة البيانات وتنظيفها، وتدريب النماذج، وأخيرًا تقديم التنبؤات لنص مدخل من قبل المستخدم.

التنفيذ:

لتنفيذ الكود:

1. تأكد من تثبيت الاعتماديات المطلوبة (pandas وscikit-learn).
2. حدد مسار ملف البيانات الخاص بالبريد الإلكتروني.
3. قم بتشغيل البرنامج، وسيتم تدريب النماذج، وطباعة مقاييس أدائها، ومن ثم يطلب من المستخدم إدخال نص للتنبؤ به.

الاستنتاج:

يوضح كود تصنيف البريد الإلكتروني تنفيذ نموذجين شهيرين في تعلم الآلة لتصنيف الرسائل الإلكترونية. من خلال استخدام خوارزميات نايف بايز والانحدار اللوجستي، يقوم الكود بفعالية بتصنيف الرسائل الإلكترونية إلى فئتين: الرسائل المزعجة والرسائل غير المزعجة، مما يبرز إمكانيات تعلم الآلة في تطبيقات تصفية البريد الإلكتروني.

البرنامج:

```python
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score

class NaiveBayesModel:
    def __init__(self):
        self.model = MultinomialNB()
        self.vectorizer = CountVectorizer()

    def train(self, X_train, y_train):
        X_train = self.vectorizer.fit_transform(X_train)
        self.model.fit(X_train, y_train)

    def predict(self, X_test):
        X_test = self.vectorizer.transform(X_test)
        return self.model.predict(X_test)

class LogisticRegressionModel:
    def __init__(self):
        self.model = LogisticRegression(max_iter=1100)
        self.vectorizer = CountVectorizer()

    def train(self, X_train, y_train):
        X_train = self.vectorizer.fit_transform(X_train)
        self.model.fit(X_train, y_train)

    def predict(self, X_test):
        X_test = self.vectorizer.transform(X_test)
```

```
20  class LogisticRegressionModel:
29      def predict(self, X_test):
30          X_test = self.vectorizer.transform(X_test)
31          return self.model.predict(X_test)
32
33  class EmailClassifier:
34      def __init__(self, data_path):
35          self.data_path = data_path
36          self.data = None
37          self.models = {'Naive Bayes': NaiveBayesModel(),
38                         'Logistic Regression': LogisticRegressionModel()}
39
40      def read_data(self):
41          self.data = pd.read_csv(self.data_path)
42
43      def clean_data(self):
44          self.data = self.data.fillna('')
45          self.data = self.data[self.data['text'].notna()]
46
47      def train_models(self):
48          results = {}
49          for model_name, model in self.models.items():
50              accuracies = []
51              X = self.data['text']
52              y = self.data['label']
53              model.train(X, y)
54              y_pred = model.predict(X)
55              accuracy = accuracy_score(y, y_pred)
56              accuracies.append(accuracy)
```

```
33  class EmailClassifier:
47      def train_models(self):
61
62          return results
63
64      def predict_single_text(self, text):
65          nb_model = self.models['Naive Bayes']
66          nb_prediction = nb_model.predict([text])[0]
67
68          lr_model = self.models['Logistic Regression']
69          lr_prediction = lr_model.predict([text])[0]
70
71          return nb_prediction, lr_prediction
72
73
74  if __name__ == "__main__":
75      data_path = "C:\\Users\\sulim\\Downloads\\DataSets_for_Spam-Emails\\combined_data.csv"
76      classifier = EmailClassifier(data_path)
77      classifier.read_data()
78      classifier.clean_data()
79      results = classifier.train_models()
80      for model_name, accuracies in results.items():
81          print(f"\nModel: {model_name}")
82          print("Average Accuracy:", sum(accuracies) / len(accuracies))
83
84      user_input = input("Enter the text to predict: ")
85      nb_prediction, lr_prediction = classifier.predict_single_text(user_input)
86      print("Naive Bayes Prediction:", nb_prediction)
87      print("Logistic Regression Prediction:", lr_prediction)
88
```

النتائج:

```
PS C:\Users\sulim> python -u "c:\Users\sulim\Downloads\MyGitHub\ML\Project\Code\v4.py"

Confusion Matrix for Naive Bayes:
[[39013   525]
 [  960 42950]]

Confusion Matrix for Logistic Regression:
[[39411   127]
 [    7 43903]]

Model: Naive Bayes
Average Accuracy: 0.982204486626402

Model: Logistic Regression
Average Accuracy: 0.9983942095676349
Enter the text to predict: Send me your information to win a million dollars.
Naive Bayes Prediction: 1
Logistic Regression Prediction: 1
PS C:\Users\sulim>
```

ملاحظات:

عند استعمال جزء قليل من الداتا ست تقوم Naïve Bayes بالتعلم بدقة اكتر من Logistic Regression.

وهذا بسبب ان Naïve Bayes سريعة التعلم.

```
PS C:\Users\sulim> python -u "c:\Users\sulim\Downloads\MyGitHub\ML\Project\Code\SulimanAlgaramanli.py"

Model: Naive Bayes
Average Accuracy: 0.8691431995206711
Confusion Matrix:
[[7448  490]
 [1694 7058]]

Model: Logistic Regression
Average Accuracy: 0.8240263630916717
Confusion Matrix:
[[6492 1446]
 [1491 7261]]
PS C:\Users\sulim> python -u "c:\Users\sulim\Downloads\MyGitHub\ML\Project\Code\SulimanAlgaramanli.py"

Model: Naive Bayes
Average Accuracy: 0.9527261833433194
Confusion Matrix:
[[7752  186]
 [ 603 8149]]

Model: Logistic Regression
Average Accuracy: 0.9383463151587778
Confusion Matrix:
[[7227  711]
 [ 318 8434]]
PS C:\Users\sulim> python -u "c:\Users\sulim\Downloads\MyGitHub\ML\Project\Code\SulimanAlgaramanli.py"

Model: Naive Bayes
Average Accuracy: 0.9689035350509287
Confusion Matrix:
[[7783  155]
 [ 364 8388]]

Model: Logistic Regression
Average Accuracy: 0.9772318753744758
Confusion Matrix:
[[7700  238]
 [ 142 8610]]
PS C:\Users\sulim>
```

عند استعمال 100 صف فقط.

عند استعمال 1,000 صف فقط.

عند استعمال 10,000 صف فقط.

ولكن عند استعمال الداتا كاملة 80,000 سطر صف تقريبا. تكون نتيجة الدقة متقاربة وغالبا ما تتفوق Logistic Regression.

```
PS C:\Users\sulim> python -u "c:\Users\sulim\Downloads\MyGitHub\ML\Project\Code\SulimanAlgaramanli.py"

Model: Naive Bayes
Average Accuracy: 0.982204486626402
Confusion Matrix:
[[39013   525]
 [  960 42950]]

Model: Logistic Regression
Average Accuracy: 0.9983942095676349
Confusion Matrix:
[[39411   127]
 [    7 43903]]
PS C:\Users\sulim>
```

عند استعمال 80,000 صف فقط.