تمرین چهارم

دسته‌بندی اخبار بر اساس برچسب خبر

شایان محمدی زاده سماکوش ۹۸۱۰۲۲۷۳

نونا قاضی زاده ۹۸۱۷۱۰۰۷

مقدمه

در این تمرین هدف ما دسته‌بندی اخبار بر اساس برچسب آنها یا همان موضوع خبر است بدین صورت که ابتدا دیتا مورد نیاز که موضوع و متن اخبارها می‌باشد را کراول می‌کنیم و سپس پیش پردازش‌های لازم را روی آن انجام می‌دهیم سپس با دو روش یکی از روش‌های سنتی classification و دیگری از روش‌های مبتنی بر transformer ها این دسته‌بندی را انجام می‌دهیم و سپس برای هر روش accuracy, f1_score(macro), confusion matrix را گزارش می‌کنیم. لازم به ذکر است از آنجا که train کردن روی حجم زیاد داده امکان پذیر نیست در هر دو روش روی ۷۰ درصد داده یادگیری را انجام می‌دهیم.

روش سنتی‌ای که برای دسته‌بندی انتخاب می‌کنیم naive bayes است و برای transformer ها از مدل pars big bert استفاده می‌کنیم و خودمان آن را fine tune می‌کنیم.

پیاده‌سازی

توجه: پیش از گرفتن خروجی از پروژه باید فایل‌های crawler را برای داشتن دیتای عنوان، موضوع متن اخبار ران کنیم.

پیش پردازش

برای این بخش ابتدا متن اخبار را نرمالایز می‌کنیم. سپس با توکنایز کردن متن اخبار آن را به کلمات می‌شکانیم، سپس از stopword هایی که در کلاس داده شده بود استفاده می‌کنیم، اما از آنجا که این stopword ها کامل نیستند و تمام کلمات اضافه و علائم نگارشی را ندارند بنابراین یک فایل دیگر ایجاد می‌کنیم و در این فایل سایر کلمات اضافه و علائم نگارشی که نیاز داریم را می‌افزاییم و بعد از مرحله توکنایزیشن این کلمات را حذف می‌کنیم و سپس با استفاده از stemming یا lemmatization کلمات را به ریشه‌شان می‌بریم.

دسته بندی به روش naive bayes

در این روش پس از پیش پردازش ابتدا به هر یک از موضوعات خبری id یکتایی نسبت می‌دهیم به طور مثال به تمام اخبارهای سیاسی ایدی صفر، به تمام اخبارهای ورزشی ایدی ده و ... را نسبت می‌دهیم

سپس با استفاده از tf-idf که در تمرین سوم پیاده‌سازی شده بود امبدینگ تمام متن‌های اخبارها را به دست می‌آوریم، سپس این را به عنوان بردار x در نظر می‌گیریم و بردار y ما ایدی category ها متناظر با خبرهایمان است.

سپس با استفاده از کتابخانه sklearn دیتامان را به دو دسته train و test تقسیم می‌کنیم. ۷۰ درصد از ۱۵۰۰ خبر را train می‌کنیم و روی ۳۰ درصد باقی آن test می‌کنیم.

در نهایت دیتایی را که برای تست قرار داده بودیم با استفاده از مدلی که حاصل train روی ۷۰ درصد داده بود برچسب گذاری می‌کنیم، بدین معنا که به آنها category مربوطه را نسبت می‌دهیم.

نسبت دادن ایدی یکتا به **subject** های مختلف:

```
target_category = news_df['subject'].unique()
target_category = target_category.tolist()
news_df['subject_id'] = news_df['subject'].factorize()[0]
```

استفاده از **tf-idf** برای امبدینگ متن و مشخص کردن بردار **x** و **y**:

```
vocabulary = set()
for doc in news_df.clean_text:
    vocabulary.update(doc.split(' '))
vocabulary = list(vocabulary)

tfidf = TfidfVectorizer(vocabulary=vocabulary, use_idf = True, dtype=np.float32)
tfidf_tran=tfidf.fit_transform([' '.join(doc) for doc in removed_tokenized_words])

with open('./models/tf_idf/tfidf.pk', 'wb') as f:
    pickle.dump(tfidf, f)

sparse.save_npz("./models/tf_idf/tfidf_tran.npz", tfidf_tran)
```

```
/Users/nonaghazizadeh/opt/anaconda3/lib/python3.9/site-packages/sklearn/feature_extraction/text.py:1322: UserWarning:
Upper case characters found in vocabulary while 'lowercase' is True. These entries will not be matched with any docum
ents
  warnings.warn(
```

```
with open('./models/tf_idf/tfidf.pk', 'rb') as f:
    tfidf_loaded = pickle.load(f)

tfidf_tran_loaded = sparse.load_npz("./models/tf_idf/tfidf_tran.npz")
```

```
x = tfidf_tran_loaded.toarray()
y = np.array(news_df.subject_id.values)
```

تفکیک دیتا برای تست و یادگیری:

لازم به ذکر است که باید پارامتر shuffle را true ست کنیم تا دیتا تست ما روی category خاص بایاس نشود

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state=0, shuffle=True)
```

یادگیری و **predict** کردن روی دیتا تست:

```
clf = MultinomialNB().fit(x_train, y_train)
```

```
with open('./models/naive_bayes/naive_bayes_model.sav', 'wb') as f:
    pickle.dump(clf, f)
```

```
with open('./models/naive_bayes/naive_bayes_model.sav', 'rb') as f:
    clf_loaded = pickle.load(f)
```

```
y_pred = clf_loaded.predict(x_test)
```

شهود بیشتر:

برای دید بهتر یک فایل csv در فولدر result وجود دارد که برچسب واقعی که به یک خبر نسبت داده شده است و برچسب که مدل ما نسبت می‌دهد قرار دارد

```
test_comparision = []
for idx,x in enumerate(x_test):
  pred_res = target_category[y_pred[idx]]
  true_res = target_category[y_test[idx]]
  test_comparision.append([true_res, pred_res])

with open('./result/predicted_naive_bayes.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["true_label", "predicted_label"])
    for row in test_comparision:
      writer.writerow(row)
```

همچنین مقدار accuracy score , f1 score macro, confusion matrix را محاسبه کرده و در بخش بعدی گزارش شده است.

```
confusion_matrix = confusion_matrix(y_test, y_pred)
accuracy_score = accuracy_score(y_test, y_pred) * 100
f1_score = f1_score(y_test, y_pred, average='macro')

print(f"accuracy score: {accuracy_score}")
print('-------------------------------------------------------------------------')
print(f"f1 score: {f1_score}")
print('-------------------------------------------------------------------------')
print(f"confusion matrix:\n {confusion_matrix}")

plt.matshow(confusion_matrix)
plt.title('Confusion matrix')
plt.colorbar()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```

دسته بندی به پایه مدلی مبتنی بر transformers

در این روش پس از پیش پردازش ابتدا به هر یک از موضوعات خبری id یکتایی نسبت می‌دهیم به طور مثال به تمام اخبارهای سیاسی ایدی صفر، به تمام اخبارهای ورزشی ایدی ده و ... را نسبت می‌دهیم

در روش قبلی tf-idf را تحت عنوان بردار x در نظر می‌گیریم حال در اینجا کل متن پیش پردازش شده را تقسیم می‌کنیم، یک تقسیم هم انجام می‌دهیم که validation داشته باشیم تا بتوانیم hyperparameter های مناسبی را انتخاب کنیم. بدین صورت که روی ۷۰ درصد از ۱۵۰۰ خبر یادگیری را انجام می‌دهیم و روی ۳۰ درصد باقی تست و ولیدیشن را انجام می‌دهیم این ۳۰ درصد را بین تست و ولیدیشن به صورت نصف تقسیم می‌کنیم.

در ترنسفرمرها ابتدا ما tokenize می‌کنیم و در نهایت به encoding ای که مد نظرمان است می‌رسیم

سپس یک کلاس برای تبدیل کردن دیتا به فرمت دیتاست pytorch تعریف می‌کنیم و به فرمت دیتاست در می‌آوریم.

هر مدل امکان این که بتوان fine tune کرد را به ما می‌دهد.

سپس با استفاده از همان داده یادگیری که داشتیم از تابع های یادگیری که از پیش تعریف شده است دیتا را train می‌کنیم.

در نهایت دیتایی را که برای تست قرار داده بودیم با استفاده از مدلی که حاصل train روی ۷۰ درصد داده بود برچسب گذاری می‌کنیم، بدین معنا که به آنها category مربوطه را نسبت می‌دهیم.

لینک کولب

لینک درایو

نسبت دادن ایدی یکتا به subject های مختلف:

```
target_category = news_df['subject'].unique()
target_category = target_category.tolist()
news_df['subject_id'] = news_df['subject'].factorize()[0]
```

مشخص کردن بردار x, y:

```
y = np.array(news_df.subject_id.values)
```

```
x_text = []
for k, v in pre_processed_news_dict.items():
  x_text.append(v['clean_text'])
```

تفکیک دیتا train, test, validation:

```
x_train_tr, x_testval_tr, y_train_tr, y_testval_tr = train_test_split(x_text, y, test_size = 0.3, random_state=0, shuffle=True)
x_val_tr, x_test_tr, y_val_tr, y_test_tr = train_test_split(x_testval_tr, y_testval_tr, test_size = 0.5, random_state=0, shuffle=True)
```

توکنایز کردن متن دیتا های train, test, validation:

```
train_encodings = tokenizer(x_train_tr, truncation=True, padding=True)
val_encodings = tokenizer(x_val_tr, truncation=True, padding=True)
test_encodings = tokenizer(x_test_tr, truncation=True, padding=True)
```

پیاده‌سازی کلاسی برای تبدیل کردن دیتا به فرمت دیتاست pytorch و استفاده از آن:

```
class NewsDataset(torch.utils.data.Dataset):
  def __init__(self, encodings, labels):
    self.encodings = encodings
    self.labels = labels
  def __getitem__(self,idx):
    item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
    item['labels'] = torch.tensor([self.labels[idx]])
    return item
  def __len__(self):
    return len(self.labels)
```

```
train_dataset = NewsDataset(train_encodings, y_train_tr)
val_dataset = NewsDataset(val_encodings, y_val_tr)
test_dataset = NewsDataset(test_encodings, y_test_tr)
```

Fine tune کردن:

```
tran_model = AutoModelForSequenceClassification.from_pretrained(MODEL_NAME, num_labels=11).to("cuda")
```

در نهایت با استفاده از تابع‌های یادگیری از پیش تعریف شده و پارامترهایی که در تصویر زیر مشخص است train می‌کنیم.

```
training_args = TrainingArguments(
    output_dir="./results",
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
)

trainer = Trainer(
    model=tran_model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset
)

trainer.train()
```

تابعی برای تعیین برچسب خبر با استفاده از مدلمان و استفاده از آن و تعیین برچسب روی داده تست مان:

```
def get_prediction(text):
    inputs = tokenizer(text, padding=True, truncation=True, return_tensors="pt").to("cuda")
    outputs = tran_model(**inputs)
    probs = outputs[0].softmax(1)
    return probs.argmax().item()
```

```
y_pred_tr = []
for idx,x in enumerate(x_test_tr):
  y_pred_tr.append(get_prediction(x))
```

شهود بیشتر:

برای دید بهتر یک فایل csv در فولدر result وجود دارد که برچسب واقعی که به یک خبر نسبت داده شده است و برچسب که مدل ما نسبت می‌دهد قرار دارد.

```
test_comparision = []
for idx,x in enumerate(x_test_tr):
  pred_res = target_category[get_prediction(x)]
  true_res = target_category[y_test_tr[idx]]
  test_comparision.append([true_res, pred_res])

import csv
with open('predicted_transformers.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["true_label", "predicted_label"])
    for row in test_comparision:
      writer.writerow(row)
```

همچنین مقدار accuracy score , f1 score macro, confusion matrix را محاسبه کرده و در بخش بعدی گزارش شده است.

```
confusion_matrix = confusion_matrix(y_test_tr, y_pred_tr)
accuracy_score = accuracy_score(y_test_tr, y_pred_tr) * 100
f1_score = f1_score(y_test_tr, y_pred_tr, average='macro')

print(f"accuracy score: {accuracy_score}")
print('--------------------------------------------------------------------------')
print(f"f1 score: {f1_score}")
print('--------------------------------------------------------------------------')
print(f"confusion matrix:\n {confusion_matrix}")

plt.matshow(confusion_matrix)
plt.title('Confusion matrix')
plt.colorbar()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```

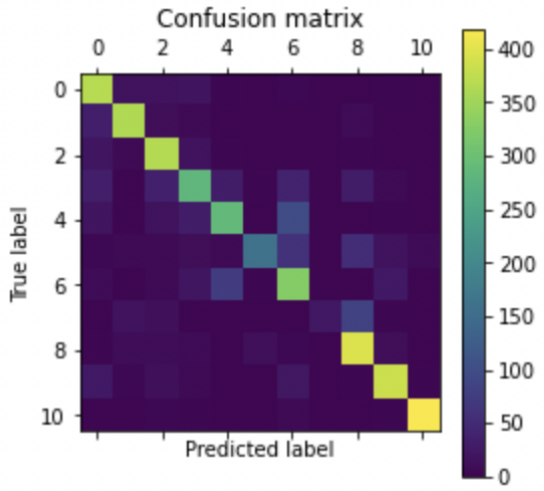
نمونه خروجی

دسته بندی به روش naive bayes

70% train

30% test

```
accuracy score: 75.6665919784898
--------------------------------------------------------------------------------
f1 score: 0.7187042361841592
--------------------------------------------------------------------------------
confusion matrix:
 [[368  15  17  19   3   0   4   0   0   3   1]
 [ 34 363  10   5   0   0   1   0   8   1   0]
 [ 20   6 365  17   2   0   1   0   2   0   0]
 [ 32   1  35 283  31   3  39   0  31   5   0]
 [ 19   0  15  31 288   0  97   0   1   3   0]
 [  4   5   5  11   5 162  60   0  54  17   7]
 [  7   1   6  20  74   2 327   0   0  25   1]
 [  1  17  14   1   0   0   0  24  82   2   1]
 [  3   9   9   7   1  13   4   1 395  10   1]
 [ 25   4  14   5   0   0  22   0   3 383   3]
 [  0   1   4   3   1   0   5   0   1   3 419]]
```



دسته بندی به پایه مدلی مبتنی بر transformers

70 train

30 test + validation (50/50)

```
accuracy score: 84.09498207885304
--------------------------------------------------------------------------------
f1 score: 0.8382849805366882
--------------------------------------------------------------------------------
confusion matrix:
 [[187   1   2   8   1   1   0   0   0   6   0]
 [  3 186   1   0   0   1   0   1   8   1   0]
 [ 11   1 201   3   2   1   0   0   1   0   0]
 [  9   3   7 147  14   2  15   0   3   3   1]
 [  4   1   2   8 170   2  33   0   5   2   0]
 [  2   1   0   7   2 119   8   1   7   7   4]
 [  3   0   0   3  38   1 194   0   1  13   2]
 [  0   0   0   0   0   0   0  54  11   0   0]
 [  0   2   1   2   0  19   2  11 190   0   1]
 [  8   1   0   2   0   7   5   1   4 216   0]
 [  0   1   0   0   0   2   3   0   2   3 213]]
```