گزارش فنی: مدل برای طبقه‌بندی احساسات در توییتر

محمد خورسندی

۱۳ خرداد ۱۴۰۳

۱ مقدمه

این گزارش به بررسی فرآیند پیاده‌سازی و آموزش یک شبکه عصبی بازگشتی برای طبقه‌بندی احساسات در داده‌های توییتر می‌پردازد. هدف از این پروژه، دسته‌بندی توییت‌ها به دو دسته مثبت و منفی است. از داده‌های Sentiment140 استفاده شده و مدل با استفاده از ابزارهای مختلفی مانند PyTorch و Gensim پیاده‌سازی شده است.

۲ پیش‌پردازش داده‌ها

در این بخش، داده‌های اولیه مورد پردازش قرار می‌گیرند تا برای مدل مناسب شوند. مراحل پیش‌پردازش شامل پاک‌سازی متن، توکن‌سازی و لماتیزاسیون است. کد زیر مراحل پیش‌پردازش را نشان می‌دهد:

```
import pandas as pd
import re
from nltk.tokenize import RegexpTokenizer
from nltk.stem import WordNetLemmatizer
import nltk

data: DataFrame = pd.read_csv('/content/HW3-1403/sentiment140.csv')
data.drop(columns=['date', 'user', 'query'], inplace=True)
data['sentiment'] = data['sentiment'].map({0: 0, 4: 1})

nltk.download('wordnet')
nltk.download('punkt')
wnl = WordNetLemmatizer()

def preproc(text: str) -> list[str]:
    text = re.sub(r'@[^\s]+', 'MENTION', text)
    text = re.sub(r'http\S*|www\S*|https\S*', 'URL', text,
        flags=re.MULTILINE)
    text = re.sub(r'#[^\s]+', 'HASHTAG', text)
    return text

def lemmatize_tokens(tokens):
    return [wnl.lemmatize(token) for token in tokens]

data['text'] = data['text'].apply(preproc)
data['text'] = data['text'].apply(lambda x: x.split(" "))
data['text'] = data['text'].apply(lemmatize_tokens)
```

۳ تبدیل متن به دنباله عددی

در این مرحله، واژه‌ها به شاخص‌های عددی تبدیل شده و دنباله‌ها به طول ثابت پر می‌شوند. به دلیل ماهیت شبکه های بازگشتی ساده درصورتی که پدینگ در اول جملات اضافه شود نتایج بسیار بهتر می‌شوند.

```
all_words = [word for tokens in data['text'] for word in tokens]
word_counts = Counter(all_words)
word_to_index = {word: i+1 for i, (word, _) in
    enumerate(word_counts.items())}
word_to_index['<PAD>'] = 0

data['text'] = data['text'].apply(lambda tokens: [word_to_index[word]
    for word in tokens])

max_len = data['text'].apply(lambda x: len(x)).max()
def pad_sequence(seq, max_len):
    return [word_to_index['<PAD>']] * (max_len - len(seq)) + seq

data['text'] = data['text'].apply(lambda x: pad_sequence(x, max_len))
```

۴ تعبیه واژه‌ها

برای استفاده از تعبیه‌های از پیش آموزش‌دیده شده word2vec، ماتریس تعبیه ساخته می‌شود:

```
drive.mount('/content/drive')
model_path = '/content/drive/MyDrive/models/word2vec-google-news-300'

if (True) :
  w2v = gensim.models.KeyedVectors.load(model_path)
else :
  w2v = api.load('word2vec-google-news-300')
  w2v.save(model_path)

embedding_dim = w2v.vector_size
vocab_size = len(word_to_index)
unknown_count = 0
embedding_matrix = torch.zeros((vocab_size, embedding_dim))
for word, index in word_to_index.items():
    if word in w2v:
        embedding_matrix[index] = torch.tensor(w2v[word])
    else:
        embedding_matrix[index] = torch.rand(embedding_dim)
        unknown_count += 1

embedding_matrix.shape
```

۵ آموزش مدل

مدل تعریف و آموزش داده می‌شود. ساختار مدل در کد زیر قابل مشاهده است:

```
class MyRNN(nn.Module):
    def __init__(self):
        super(MyRNN, self).__init__()
        embed_tmp = torch.tensor(embedding_matrix,
            dtype=torch.float32).clone().detach()
        self.embedding = nn.Embedding.from_pretrained(embed_tmp,
            freeze=False, padding_idx=0)
        self.rnn = nn.RNN(input_size=300, hidden_size=32,
            batch_first=True)
        self.dense = nn.Linear(32, 1)

    def forward(self, x):
        embedded = self.embedding(x)
        output, hidden = self.rnn(embedded)
        out = self.dense(output[:, -1, :])
        return out
```

ابر پارامتر ها و لوپ اصلی برنامه:

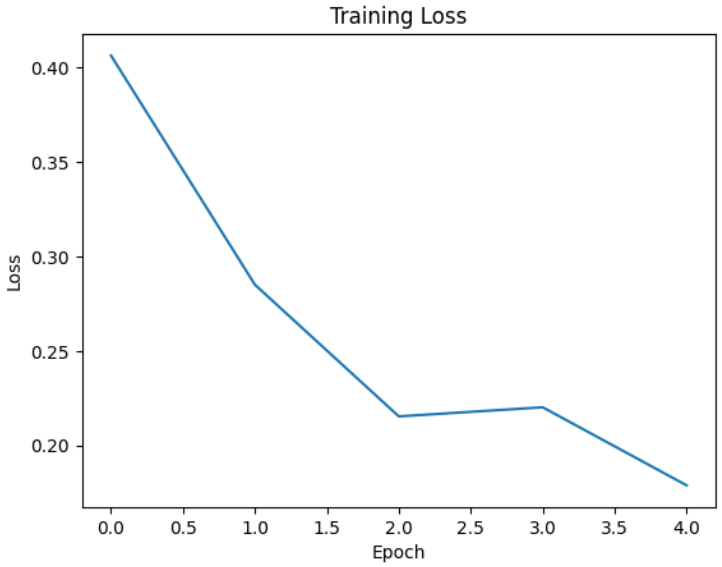
```
model = MyRNN().to(device)
criterion = nn.BCEWithLogitsLoss()
optimizer = optim.Adam(model.parameters())
train_loader = DataLoader(train_dataset, batch_size=512, shuffle=True)

epochs = 5
loss_list = []
for epoch in range(epochs):
    model.train()
    iter = 0
    for inputs, labels in train_loader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs.squeeze(dim=1), labels)
        loss.backward()
        optimizer.step()
        iter += 1
        if iter % 100 == 0:
          print(f"iter {iter}/ {len(train_loader)}")
    loss_list.append(loss.item())

    print(f'Epoch {epoch+1}/{epochs}, Loss: {loss.item()}')
```

۶ ارزیابی مدل و نتیجه‌گیری

در این پروژه، یک مدل شبکه عصبی بازگشتی برای طبقه‌بندی احساسات در داده‌های توییتر پیاده‌سازی و آموزش داده شد. نتایج حاصل نشان می‌دهد که مدل توانایی خوبی در تشخیص احساسات مثبت و منفی دارد.

	precision	recall	f1-score	support
0.0	0.80	0.82	0.81	159920
1.0	0.82	0.79	0.80	160080
accuracy			0.81	320000
macro avg	0.81	0.81	0.81	320000
weighted avg	0.81	0.81	0.81	320000

شکل ۱: ارزیابی



شکل ۲: خطا