一、情感分析与文本量化

1. emo.py简介

在emo.py中，我们实现了对中文新闻数据的情感分析和关键词频率的量化。主要目标是通过分析新闻文本中的情感倾向和相关关键词的频率，提供对新闻情感的量化指标。这些指标包括情感得分、情感标签、情感倾向、情感强度等，并进一步利用这些信息进行情感分析，来预测与股票相关的情感变化。

1.1 数据来源

emo.py使用了两份包含新闻内容的数据集：2020_2021data.csv和2022data.csv（由原新闻数据提取必要的列得到，具体过程见preprocess.py）。每份数据集包含了多条新闻记录，主要内容为各类金融新闻。

2. 数据预处理

2.1 数据加载与清洗

首先，我们读取CSV文件并加载新闻数据：

```
data = pd.read_csv(file_path)
```

接着，我们去除所有缺失新闻内容的记录：

```
data.dropna(subset=['NewsContent'], inplace=True)
```

2.2 提取关键词频率

我们使用CountVectorizer对新闻内容中的关键词进行提取。关键词是事先定义好的与股票涨跌相关的词汇，便于后续预测时加强特定新闻对股票涨跌的影响。

```
up_keywords = [
    '增长', '盈利', '业绩', '扩张', '收购', '创新', '突破', '发展', '市场需求', '投资', '股东回报', '利润','利好', '好消息', '强劲', '飙升', '红利', '增长率', '回报', '股市繁荣', '开盘上涨','并购', '盈利能力', '股东奖励', '业绩增长', '产品创新', '市场领先', '财务改善', '营收增长', '战略成功','行业繁荣', '市场份额提升', '行业扩张', '需求增加', '行业前景', '市场前景', '经济复苏', '外资进入','政策支持', '利率降低', '政策利好', '增长动力', '市场信心'
]
down_keywords = [
    '亏损', '裁员', '下滑', '跌幅', '亏损', '危机', '负面', '下降', '停滞', '下降', '萎缩', '负债', '风险','报告不佳', '低迷', '倒闭', '萎靡', '亏损预期', '盈利下滑', '业绩亏损', '负债累累', '财务危机', '销售下降','裁员计划', '经营困难', '产品召回', '行业衰退', '市场份额丧失', '需求下降', '行业疲软', '市场不景气','经济衰退', '通货紧缩', '失业率上升', '政策收紧', '货币紧缩', '金融危机', '外资撤出', '股市暴跌'
]
all_keywords = list(set(up_keywords + down_keywords))
```

然后使用CountVectorizer提取新闻文本中的这些关键词频率：

```
vectorizer = CountVectorizer(vocabulary=all_keywords)
keyword_matrix = vectorizer.transform(data['NewsContent'])
keyword_features = pd.DataFrame(keyword_matrix.toarray(), columns=vectorizer.get_feature_names_out())
```

2.3 处理文本分块

由于BERT模型对输入文本的长度有限制，我们将文本划分为多个块，确保每块的长度不超过512个tokens：

```
def split_text_into_chunks(text, max_length=512):
    max_chunk_length = max_length - 2  # 留出2个位置给[CLS]和[SEP]
    tokens = tokenizer.tokenize(text)
    chunks = [tokens[i:i + max_chunk_length] for i in range(0, len(tokens), max_chunk_length)]
    return [tokenizer.convert_tokens_to_string(chunk) for chunk in chunks]
```

3. 情感分析

3.1 使用BERT进行情感分析

我们使用预训练的中文BERT模型来分析新闻文本的情感倾向。通过BertTokenizer和BertForSequenceClassification，我们加载了BERT模型并将其转移到GPU：

```
model_name = "bert-base-chinese"
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForSequenceClassification.from_pretrained(model_name)
model = model.to(device)
```

3.2 情感得分计算

通过BERT模型，我们对每个新闻文本进行情感分析，并计算其情感得分。我们使用Softmax函数来获得每个文本的情感分类概率：

```
def analyze_sentiment_batch(texts):
    inputs = tokenizer(texts, return_tensors="pt", padding=True,
truncation=True, max_length=512).to(device)
    with torch.no_grad():
        outputs = model(**inputs)
        logits = outputs.logits
        sentiment_scores = torch.softmax(logits,
dim=-1).cpu().numpy()
    return sentiment_scores[:, 1]  # 返回正面情感的概率
```

3.3 批量处理情感分析

由于数据量较大，我们对新闻数据进行批量处理，这里设置批量的大小为16：

```
def process_batches(data, batch_size=16):
    sentiments = []
    for i in tqdm(range(0, len(data), batch_size)):
        batch = data.iloc[i:i + batch_size]
        text_batch = batch['NewsContent'].tolist()
        sentiment_batch = analyze_sentiment_batch(text_batch)
        sentiments.extend(sentiment_batch)
    return sentiments
```

3.4 调整情感得分

在情感得分计算的基础上，我们根据关键词频率对情感得分进行调整。正面关键词（如‘盈利’,‘增长’）和负面关键词（如‘亏损’,‘危机’）的频率会影响最终的情感得分：

```
def adjust_sentiment_with_keywords(row):
    up_count = sum([row[key] for key in up_keywords if key in row])
    down_count = sum([row[key] for key in down_keywords if key in
row])
    sentiment_score = row['SentimentScore']
    sentiment_adjusted_score = sentiment_score + 0.1 * (up_count -
down_count)  # 增加正面关键词的影响，减少负面关键词的影响
    return sentiment_adjusted_score
```

3.5 确定情感标签

基于调整后的情感得分，我们可以为每条新闻确定情感标签（正面、负面、中立）：

```
def sentiment_label(score):
    if score > 0.5:
        return 'POSITIVE'
    elif score < 0.5:
        return 'NEGATIVE'
    else:
        return 'NEUTRAL'
```

4. 结果与保存

最终，我们将关键词频率、情感标签、情感倾向等数据合并，并保存为CSV文件：

```
data = pd.concat([data[['DeclareDate', 'Symbol']],
keyword_features, data[['SentimentLabel', 'SentimentScore',
'SentimentTendency', 'SentimentStrength']]], axis=1)
data.to_csv('news_text_quantification.csv', index=False)
```

文本量化部分结果如下：SentimentScore大于0.5表示积极情感，越大表示情感越强烈。

	A	B	C	D	E	F	G	H	I
1	DeclareDat	Symbol	SentimentScore						
2	2020/1/1	300146	0.683905						
3	2020/1/1	300773	0.576567						
4	2020/1/1	600083	0.671845						
5	2020/1/1	892	0.639538						
6	2020/1/1	601077	0.573105						
7	2020/1/1	601965	0.657451						
8	2020/1/1	601916	0.674856						
9	2020/1/1	300199	0.566095						
10	2020/1/1	300139	0.543247						
11	2020/1/1	300264	0.690384						
12	2020/1/1	601916	0.684735						
13	2020/1/1	2291	0.738315						
14	2020/1/1	651	0.574621						
15	2020/1/1	300176	0.626558						
16	2020/1/1	1	0.703061						
17	2020/1/1	625	0.727569						
18	2020/1/1	625	0.722006						
19	2020/1/1	300498	0.61617						
20	2020/1/1	300006	0.616441						
21	2020/1/1	1	0.726814						
22	2020/1/1	600116	0.651195						
23	2020/1/1	799	0.619384						
24	2020/1/1	625	0.74021						
25	2020/1/1	2415	0.593839						
26	2020/1/1	300142	0.700117						
27	2020/1/1	603109	0.719021						
28	2020/1/1	600029	0.717566						

5. 总结

通过emo.py，我们成功地对中文金融新闻进行了情感分析，量化了新闻文本中的情感倾向，并根据相关关键词频率对情感得分进行了调整。通过这种方式，我们可以更好地理解新闻对股市的潜在影响。此外，使用BERT模型进行情感分析保证了情感分析的高准确性和稳定性。

二、基于LSTM的股价预测与情感分析

1.predict.py简介

predict.py旨在通过结合LSTM（长短期记忆网络）和情感分析，对股票价格进行预测。通过情感得分对股价进行加权调整，以模拟新闻情感对股市的影响。LSTM模型利用历史股价数据和情感得分预测未来的股价，并通过情感得分加权进一步调整预测结果。本报告详细描

述了如何使用LSTM模型进行股价预测，并解释了在该过程中如何处理情感得分。

1.1 数据来源与预处理

predict.py的数据来源于merged_data.csv（这是用前面的bert预训练模型提取的文本量化特征和股票688981的2020年到2022年的历史数据处理得到的数据集，具体处理过程见feature.py和feature1.py），该文件包含了股票的日收盘价和对应的情感得分（SentimentScore）。我们将数据集划分为训练集（2020年7月16日到2022年6月6日）和测试集（2022年6月6日到2022年12月30日），并对特征进行标准化处理。标准化后的数据为LSTM模型的训练和预测提供了一个更稳定的输入。

1.2 目标

1. 使用LSTM模型对股价进行预测。
2. 利用情感分析结果调整股价预测。
3. 评估情感得分对股价预测的影响，并将预测结果与实际股价进行对比。

2. 关键函数解析

2.1 process_sentiment函数

process_sentiment函数用于填充情感得分中的缺失值，并确保时间序列的平滑过渡。该函数通过前一个时间点的情感得分填充缺失值，且通过线性变化避免情感得分的剧烈波动。具体实现如下：process_sentiment函数的主要算法过程是对情感得分（SentimentScore）中的缺失值进行平滑填充。首先，函数检查是否提供了训练集参考数据train_ref，如果提供，则使用参考数据中的最后一个有效情感得分填充当前数据集中的缺失值。接着，对于每个缺失的情感得分，算法会根据前一个有效值进行填充。具体地，如果前一个情感得分大于0.5，缺失值会逐渐衰减至0.5，每次减小0.003；如果前一个情感得分小于0.5，缺失值会逐渐递增至0.5，每次增加0.003。这样处理确保了情感得分在时间序列中的平滑变化，避免了突变或剧烈波动，最终返回处理后的DataFrame。

```
def process_sentiment(df, train_ref=None):
    df = df.copy()
    sentiment_values = df['SentimentScore'].values

    if train_ref is not None:
        df['SentimentScore'] =
df['SentimentScore'].fillna(train_ref['SentimentScore'].iloc[-1])
```

```
    for i in range(1, len(sentiment_values)):
        if np.isnan(sentiment_values[i]):
            if sentiment_values[i - 1] > 0.5:
                sentiment_values[i] = max(sentiment_values[i - 1] -
0.003, 0.5)
            else:
                sentiment_values[i] = min(sentiment_values[i - 1] +
0.003, 0.5)
        else:
            last_valid_value = sentiment_values[i]

    df['SentimentScore'] = sentiment_values
    return df
```

- 功能：填补缺失的情感得分，并确保其平滑过渡。
- 应用：用于训练集和测试集的情感得分填充。

2.2 predict_steps函数

predict_steps函数基于LSTM模型进行未来股价的多步预测，并结合情感得分对预测结果进行加权调整。情感得分（SentimentScore）直接影响股价预测结果，使得模型能够模拟市场情感对股市的影响：在predict_steps函数中，情感得分通过加权调整对股价预测产生影响。当情感得分大于0.5时，表示市场情感积极，股价预测值会增加，调整幅度与情感得分与0.5的差值成正比，调整系数由sentiment_weight控制；反之，当情感得分小于0.5时，表示市场情感消极，股价预测值会减少，调整幅度同样与情感得分与0.5的差值成正比。每次预测时，股价预测值会根据情感得分进行调整，并作为新的特征与股价一起进入下一个时间步的预测中，从而在未来的预测中继续影响股价。通过这种方式，情感得分直接影响股价预测的方向和幅度，使得股价预测不仅基于历史价格，还考虑了市场情绪的潜在影响。

```
def predict_steps(model, initial_sequence, steps, test_sentiments,
sentiment_weight):
    model.eval()
    current_sequence = initial_sequence.clone().detach()
    predictions = []

    with torch.no_grad():
        for i in range(steps):
            output = model(current_sequence.unsqueeze(0))
            pred = output.item()
```

```
            sentiment = test_sentiments[i].item() if
isinstance(test_sentiments, np.ndarray) else test_sentiments[i]

            if sentiment > 0.5:
                pred += sentiment_weight * (sentiment - 0.5)
            else:
                pred -= sentiment_weight * (0.5 - sentiment)

            new_features = torch.tensor([[pred, float(sentiment)]],
dtype=torch.float32).to(device)
            current_sequence = torch.cat([current_sequence[1:],
new_features])
            predictions.append(pred)

    return np.array(predictions)
```

- **功能**：基于情感得分对股价预测进行加权调整，模拟情感对股价的影响。
- **应用**：在预测阶段，通过情感得分加权调整每一步的股价预测。

2.3 LSTM模型定义

LSTM模型是本项目的核心，用于从历史股价数据中学习股市的时序模式。LSTM具有记忆能力，能够有效捕捉序列数据中的长期依赖性。下面是LSTM模型的定义：

```
class LSTMModel(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers=2):
        super(LSTMModel, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers,
batch_first=True, dropout=0.2)
        self.dropout = nn.Dropout(0.2)
        self.linear = nn.Linear(hidden_size, 1)

    def forward(self, x):
        out, _ = self.lstm(x)
        out = self.dropout(out[:, -1, :])
        return self.linear(out)
```

- **LSTM层**：由多个LSTM单元组成，能够捕捉输入序列中的时序特征。
- **Dropout层**：为了防止过拟合，我们在LSTM输出后加入了Dropout层。

- **全连接层**：将LSTM层的输出转化为单一的股价预测值。

3. 模型训练与预测

3.1 模型训练

在训练过程中，我们使用了均方误差（MSE）作为损失函数，并使用Adam优化器对模型进行优化。训练过程中，损失函数的值会不断下降，说明模型在学习股价的预测能力。

```
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

for epoch in range(num_epochs):
    model.train()
    total_loss = 0
    for inputs, targets in train_loader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs.squeeze(), targets)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()

    avg_loss = total_loss / len(train_loader)
    print(f'Epoch [{epoch + 1}/{num_epochs}], Loss:
{avg_loss:.4f}')
```

3.2 股价预测与情感加权调整

在预测阶段，使用LSTM模型生成未来股价的预测，并结合情感得分进行加权调整（sentiment_weight设置为0.035）。情感得分越高，表示市场情感越积极，股价预测上调得越多；情感得分越低，股价预测下调得越多。

```
predicted_scaled = predict_steps(model, initial_sequence,
                                 len(test_sentiments),
                                 test_sentiments,
sentiment_weight=0.035)
```

3.3 反标准化预测结果

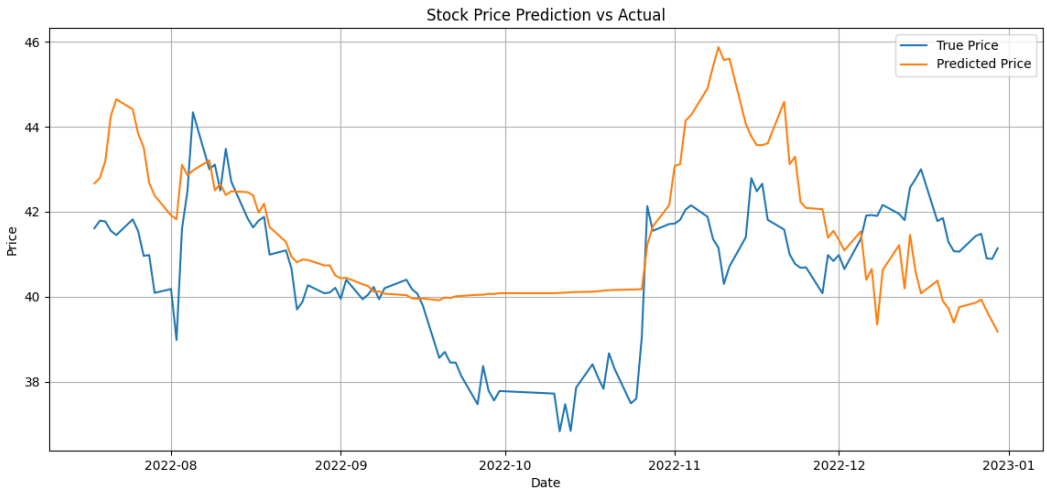
最终，我们将预测的股价通过反标准化转换为实际股价值，并与真实股价进行对比。

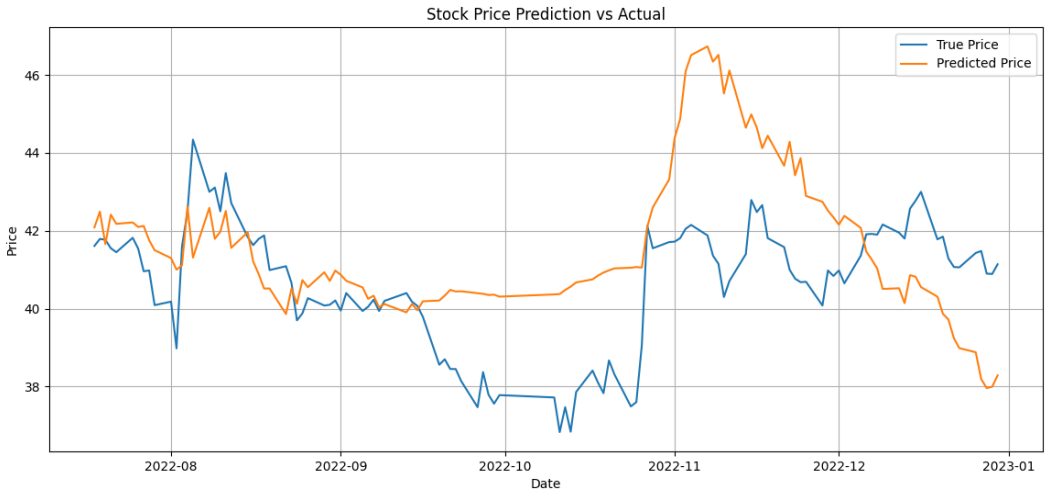
```
dummy_matrix = np.zeros((len(predicted_scaled), 2))
dummy_matrix[:, 0] = predicted_scaled
dummy_matrix[:, 1] = test_sentiments
predicted_prices = scaler.inverse_transform(dummy_matrix)[:, 0]
```

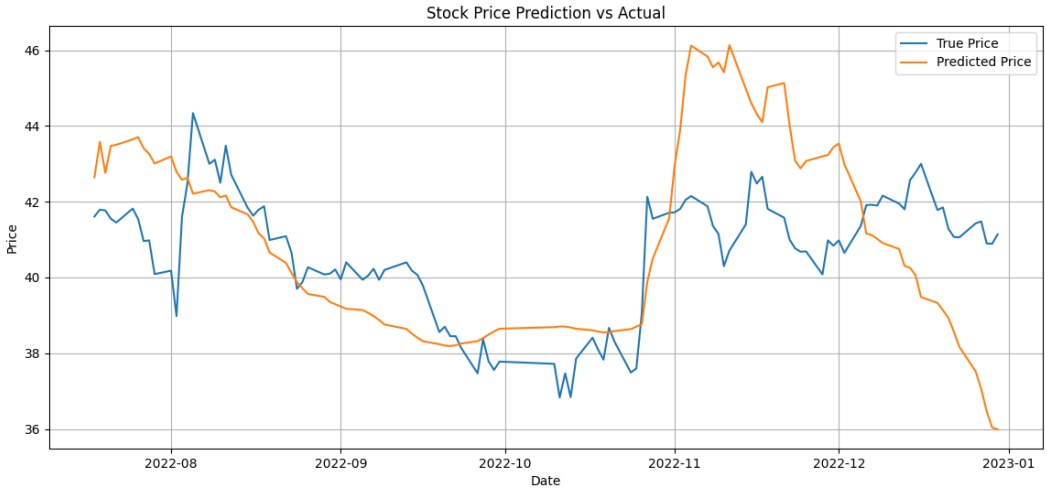
4. 结果分析与可视化

4.1 预测结果与实际股价对比

运行predict.py，并将预测结果与真实的股价进行了对比，绘制了预测值与实际值的折线图。可以调节num_epochs和sentiment_weight等参数改变模型性能，这里我设置num_epochs=4000，sentiment_weight=0.035跑了三次，结果如下：







4.2 模型性能

通过观察预测结果与真实股价的对比图，我们可以评估模型的性能：从图中可以看出，预测的股价（橙色曲线）与实际股价（蓝色曲线）大致呈现出相同的趋势，尤其是在前面的时期，预测结果与真实股价的吻合度较高。然而，在一些波动较大的阶段，预测值与实际股价之间存在较为明显的差距，特别是在2022年9月和2022年11月之间。总体而言，模型能够较好地捕捉股价的趋势，但在股市剧烈波动时，预测精度有所下降，这表明模型在面对突发市场情绪和快速变化的股市环境时表现不够理想。