
Review Sentiment Classification

- Project Overview
- Problem Definition
- Data Collection
- Feature Extraction
- Model Definition
- Model Training
- Model Evaluation
- Conclusion and Outlook



H3Art

Review Sentiment Classification

Project Overview


- **Project Focus:** Development of a sentiment tri-classification model for product reviews, utilizing NLP techniques.
- **ML/DL Models Used:** Three different models: Support Vector Machine (SVM), Long Short-Term Memory (LSTM), and Bidirectional Encoder Representations from Transformers (BERT).
- **Data Collection:** Comments scraped using Selenium from e-commerce platforms and restaurant reviews, categorized into negative, neutral, and positive sentiments.
- **Preprocessing & Feature Extraction:** For SVM and LSTM, used Jieba for Chinese segmentation and Word2Vec for feature extraction. For BERT, employed its native feature extractor with fine-tuning on downstream tasks.
- **Evaluation Metrics:** Model performance assessed using Accuracy, Precision, Recall, and F1 Score.




Defining the Problem


Sentiment Classification in Reviews

- **Objective:** Develop a system to automatically identify and categorize emotional tendencies in text, aiding in areas like product/service reviews and market analysis.
- **Sentiment Categories:** Emotions are typically classified into positive, neutral, and negative categories.
- **Input Data:** User reviews or opinions, primarily in natural language text, sourced from various platforms such as online shopping sites.
- **Preprocessing Requirements:** Text undergoes cleaning (removing irrelevant characters), segmentation using jieba, and stopword removal using Harbin Institute of Technology's stopword list.


**灿**源**
1年前 1000ml蓝色-魔法师

宝贝到手 感觉材质 颜色都是物有所值, 做工 质量看着也不错, 买家的服务也是可以点赞的, 应该是性价比较高的一次购物哦!




**米**w**
15天前 原味20包

特别差, 服务态度差, 包装差, 一团糟

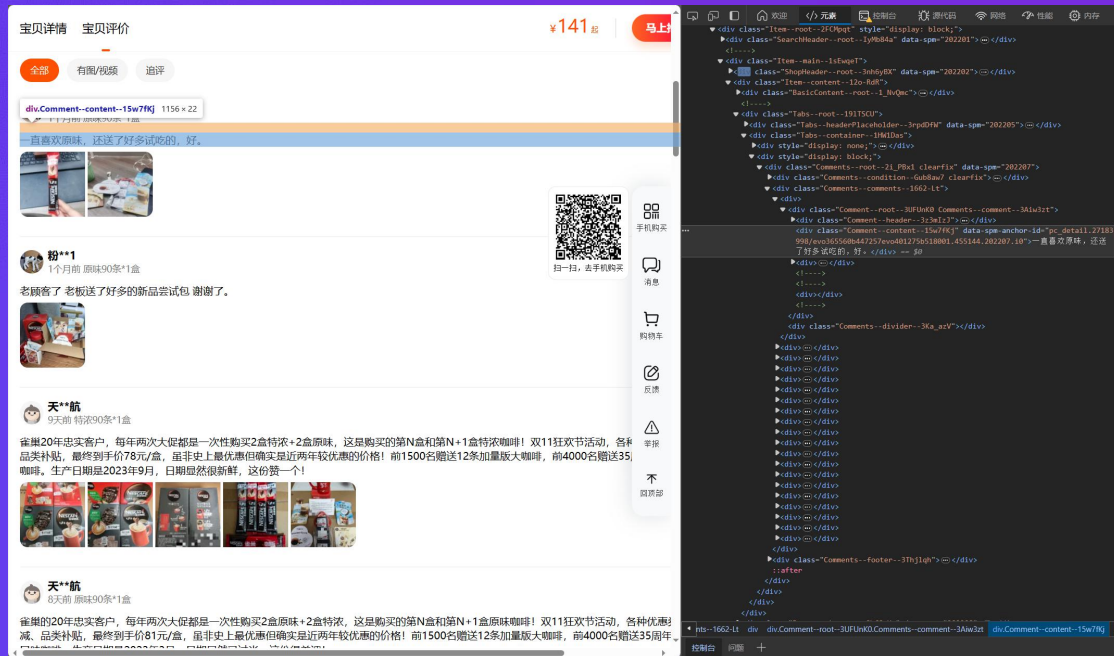
**g**7**
24天前 原味20包

薯条硬硬的 口感不好, 没有之前拍的酥。品控不好

Positive

**s**y**
1年前 T1定制款-挂绳 长款

Neutral
有点宽 拿来挂员工卡的



Data Collection & Preprocessing

Gathering and Preparing Text Data

- **Data Source:** Textual review data collected from e-commerce and restaurant review platforms using Selenium.
- **Challenges in Data Collection:** Addressing sophisticated anti-scraping measures on websites. Manual login through QR code scanning required for automated web scraping.

```
while num < 20:
    # 模拟滚动 (滚动过头会无法点击下一页, 同时天猫限制了只能查看前三页评价, 每次大约爬取50条评论)
    self.scroll()
    # 获取本页面的评价
    review_container = self.driver.find_element_by_class_name('Comments--comments--1662-Lt')
    reviews = review_container.find_elements_by_xpath("//*[contains(@class, 'Comment--content')]")
    for review in reviews:
        text = review.text
        if text != "此用户没有填写评论!" and len(text) > 5:
            review_data_list.append({'text': text, 'ignore_index': True})
            print(text)
    # 翻页
    try:
        self.driver.find_element_by_xpath(
            "//*[[@id='root']/div/div[2]/div[2]/div[2]/div[2]/div[2]/div/div[3]/div/button[2]").click()
        num += 1
    except:
        print("没有更多页面")
        break
```

```
1个用法  H3Art
class Ecommerce_crawler(object):
    H3Art
    def __init__(self):
        self.url = "https://login.tmall.com/?redirectURL=https%3A%2F%2Fdetail.tmall.com%2Fitem.htm%3Fs"
        self.path = "../lib/chromedriver"
        if os.name == 'nt':
            self.path = "../lib/chromedriver.exe"

        options = webdriver.ChromeOptions()
        options.add_experimental_option('excludeSwitches', value=['enable-automation'])
        options.add_experimental_option('useAutomationExtension', value=False)

        self.driver = webdriver.Chrome(executable_path=self.path, options=options)
```

```
1个用法  H3Art
def login(self):
    self.driver.execute_cdp_cmd('Page.addScriptToEvaluateOnNewDocument', {'cmd_args': {
        "source": """
            Object.defineProperty(navigator, 'webdriver', {
                get: () => undefined
            })
        """}
    self.driver.get(url=self.url)
    # 等待登录成功 (天猫的验证比较严格, 暂时没实现直接账号密码登录, 需使用扫码登录)
    input("完成登录后, 在此输入回车 (PRESS ENTER) 以继续...")
```

```
1个用法  H3Art
def get_review(self):
    print("Getting review...")
    # 点击"宝贝评价"
    self.driver.find_element_by_xpath(
        "//*[[@id='root']/div/div[2]/div[2]/div[2]/div[1]/div/div/div[2]/span").click()
    num = 0
```


Data Collection & Preprocessing

Gathering and Preparing Text Data

- **Data Extraction Techniques:** Utilized class or xpath tags to locate review sections. Some platforms provided direct sentiment labels, while others required manual categorization.
- **Data Imbalance:** Collected data exhibited uneven distribution across negative, neutral, and positive categories, leading to potential long-tail issues in model training.

total_neg.csv ×	
8043	很满意 就是安装材料费有点贵花了两百多 棒!
8044	还没装上去,美的货信得过哈哈! 不过物流真心
8045	热水器用了一个星期收到! 和客服沟通了下,现
8046	刚开始说好了,有一个四件套,结果给我忘了发
8047	客服骗人 不是恒温又说是恒温 噪音大 燃气的
8048	厨宝还好吧 但那赠品送了也跟没送差不多
8049	恰逢春节期间,没有安装工人,所以自己动手两
8050	安装我自己花了500多,美的够黑心的,真的是
8051	东西不错,售后太差,安装一个热水器400块钱
8052	碰到最差的、最骗人的卖家,好吧,我倒霉! 这
8053	宝贝不错,物流也不错,售后差,
8054	美的售后太垃圾,其他售后都是两小时回电话,
8055	

total_neu.csv ×	
3140	双十一买的, 还没安装, 但是很满意!
3141	商品不错 送货也快 但是服务员售后真的不
3142	到货速度很快, 宝贝包装完好。虽然还没来
3143	还没拆开, 但是包装的很好。第二次来买了,
3144	价格实惠, 快递也快, 安装也快, 虽然安装
3145	虽然家里用不到 但商家的服务态度 超级棒
3146	已经安装上了, 但没有试, 我相信美的质量,
3147	很不错, 到货很快, 当天给客服打电话, 下
3148	买来放在出租房里的, 所以自己也没试过, 个
3149	整体感觉一般般把, 但是价格不算贵, 口味
3150	对于四个大小伙子来说套餐里的菜有点多, 到
3151	一般般, 常来的店了
3152	

total_pos.csv ×	
⚠ 文件大小(2.61 MB)超出了配置的限制(2.56 MB)。代码洞察功能?	
8717	超级好的卖家! 之前不小心拍错了, 客服非常耐心的帮我解
8718	还没拆货, 物流给力, 送货到家, 一直信赖大品牌, 给32个
8719	挺好的, 卖家态度也很好
8720	挺好的, 今天用了。
8721	很好的热水器! 很实际的功能! 大品牌! 很好! 安装花了9
8722	好评! 好评!
8723	质量不错~~~
8724	东西很好 妈妈很喜欢 大天客服 服务很好 等安装后在来
8725	发货很快, 物流也及时, 热水器包装很好, 已经打电话找师
8726	很不错, 安装很到位哦!
8727	东东不重, 一个人就拉回家了, 找人来安装, 费用也不高,
8728	安装好了 挺好的! 赞
8729	宝贝很好很喜欢 卖家有多努力哦~~~~ 好
8730	

Data Collection & Preprocessing

Gathering and Preparing Text Data

- **Preprocessing Steps:** Used Jieba library for segmentation and Harbin Institute of Technology's stopword list for text cleaning.

jieba

"Jieba" (Chinese for "to stutter") Chinese text segmentation: built to be the best Python Chinese word segmentation module.

Features

- Support three types of segmentation mode:
 1. Accurate Mode attempts to cut the sentence into the most accurate segmentations, which is suitable for text analysis.
 2. Full Mode gets all the possible words from the sentence. Fast but not accurate.
 3. Search Engine Mode, based on the Accurate Mode, attempts to cut long words into several short words, which can raise the recall rate. Suitable for search engines.
- Supports Traditional Chinese
- Supports customized dictionaries
- MIT License

```
hit_stopwords.txt ×
728  之
729  只是
730  只限
731  只要
732  只有
733  至
734  至于
735  诸位
736  着
737  着呢
738  自
739  自从
740  自个儿
741  自各儿
742  自己
743  自家
744  自身
745  综上所述
746  总的来看
747  总的来说
748  总的说来
749  总而言之
```

```
hit_stopwords.txt ×
674  用
675  由
676  由此可见
677  由于
678  有
679  有的
680  有关
681  有些
682  又
683  于
684  于是
685  于是乎
686  与
687  与此同时
688  与否
689  与其
690  越是
691  云云
692  哉
693  再说
694  再者
695  在
```

Data Collection & Preprocessing

Data partition

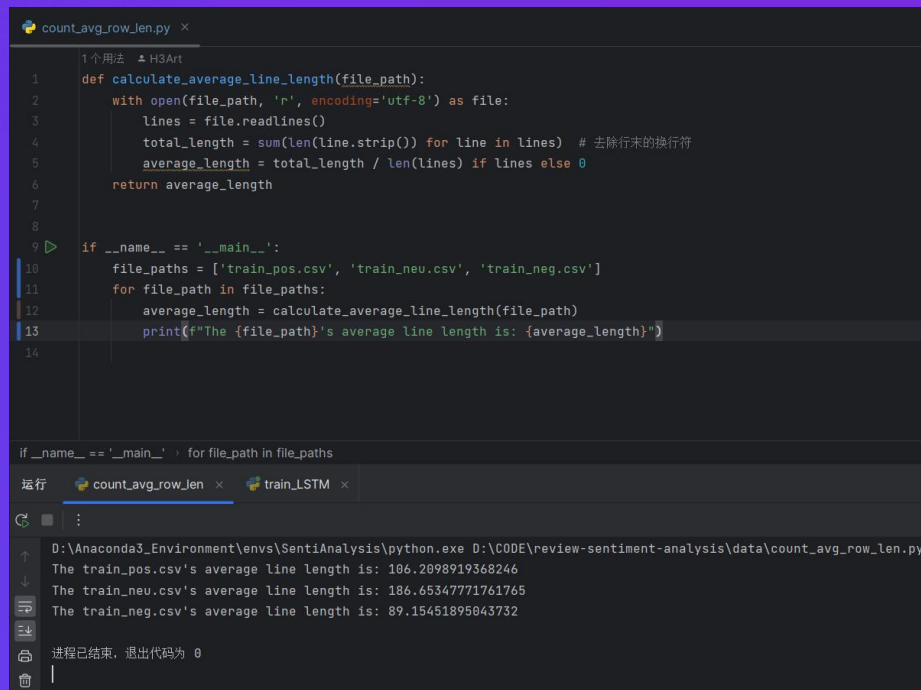
- During the data collection process, the data I obtained actually had a huge bias. This is reflected in the fact that the number of positive, neutral and negative evaluations is not consistent, and even has a large gap. In the entire data, the number of positive evaluations is the largest, while the number of neutral evaluations is far less than the number of positive and negative evaluations.
- Therefore, in the final training, I only used about half of the positive and negative evaluations to ensure that the number was close to the neutral evaluation and prevent the long tail phenomenon. Specifically, for each label, I selected 3,000 reviews and divided the training set and test set at a ratio of 4:1.

```
split_data.py x
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3
4 df = pd.read_csv('./total_pos.csv')
5
6 train_df, test_df = train_test_split(*arrays: df, train_size=2400, test_size=600, random_state=42)
7
8 train_df = train_df.drop_duplicates()
9 test_df = test_df.drop_duplicates()
10
11 train_file_path = './train_pos.csv'
12 test_file_path = './test_pos.csv'
13 train_df.to_csv(train_file_path, index=False)
14 test_df.to_csv(test_file_path, index=False)
15
```


Feature Extraction Techniques

Processing Text Data for ML Models

- I counted the length of each evaluation, and found that the evaluation length in neutral evaluations was significantly longer than that in other categories. This may have an impact on subsequent training, but I did not do further processing. In the content of the subsequent feature extraction. The truncation length of data processing is uniformly set to 100.



```
count_avg_row_len.py x
1 个用法  H3Art
1 def calculate_average_line_length(file_path):
2     with open(file_path, 'r', encoding='utf-8') as file:
3         lines = file.readlines()
4         total_length = sum(len(line.strip()) for line in lines) # 去除行末的换行符
5         average_length = total_length / len(lines) if lines else 0
6     return average_length
7
8
9
10 if __name__ == '__main__':
11     file_paths = ['train_pos.csv', 'train_neu.csv', 'train_neg.csv']
12     for file_path in file_paths:
13         average_length = calculate_average_line_length(file_path)
14         print(f"The {file_path}'s average line length is: {average_length}")
```

运行 count_avg_row_len x train_LSTM x

D:\Anaconda3_Environment\envs\SentiAnalysis\python.exe D:\CODE\review-sentiment-analysis\data\count_avg_row_len.py
The train_pos.csv's average line length is: 186.2898919368246
The train_neu.csv's average line length is: 186.65347771761765
The train_neg.csv's average line length is: 89.15451895043732

进程已结束, 退出代码为 0

Feature Extraction Techniques

Processing Text Data for ML Models

- **For SVM and LSTM:** Used Jieba for Chinese word segmentation and Word2Vec for converting words into vectors. Involved manual data cleaning to remove stop words and special symbols.
- **Dimensionality Reduction:** Word2Vec acts as a dimensionality reduction technique, converting high-dimensional text data into fixed-size vectors. Set a truncation length of 100 words based on average comment length in the dataset.

```
# 过滤emoji
4 个用法  H3Art
def emoji_filter(desstr, restr=' '):
    try:
        co = re.compile(u'[\U00010000-\U0010ffff]')
    except re.error:
        co = re.compile(u'[\uD800-\uDBFF][\uDC00-\uDFFF]')
    return co.sub(restr, desstr)

4 个用法  H3Art
def stopwords_list():
    stopwords = [
        line.strip() for line in open('../review_treatment/hit_stopwords.txt', encoding='utf-8').readlines()
    ]
    return stopwords
```

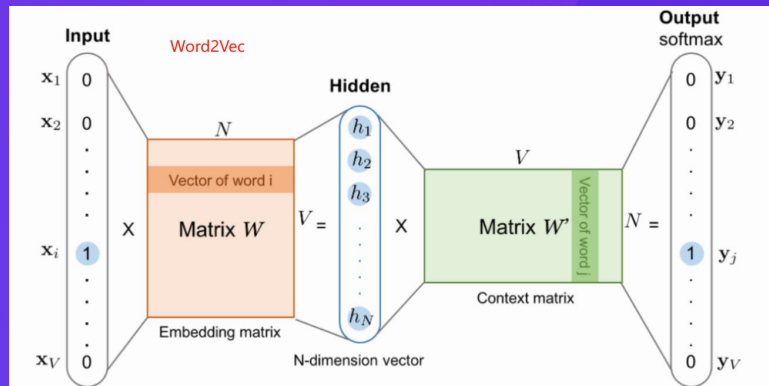
```
# 全模式
seg_list = jieba.cut("中国上海是一座美丽的国际性大都市", cut_all=True)
print("Full Mode: " + " / ".join(seg_list))

# 返回结果
Full Mode: 中国/ 上海/ 是/ 一座/ 美丽/ 的/ 国际/ 国际性/ 大都/ 大都市/ 都市

# 精确模式
seg_list = jieba.cut("中国上海是一座美丽的国际性大都市", cut_all=False)
print("Full Mode: " + " / ".join(seg_list))

# 返回结果
Default Mode: 中国/ 上海/ 是/ 一座/ 美丽/ 的/ 国际性/ 大都市
```

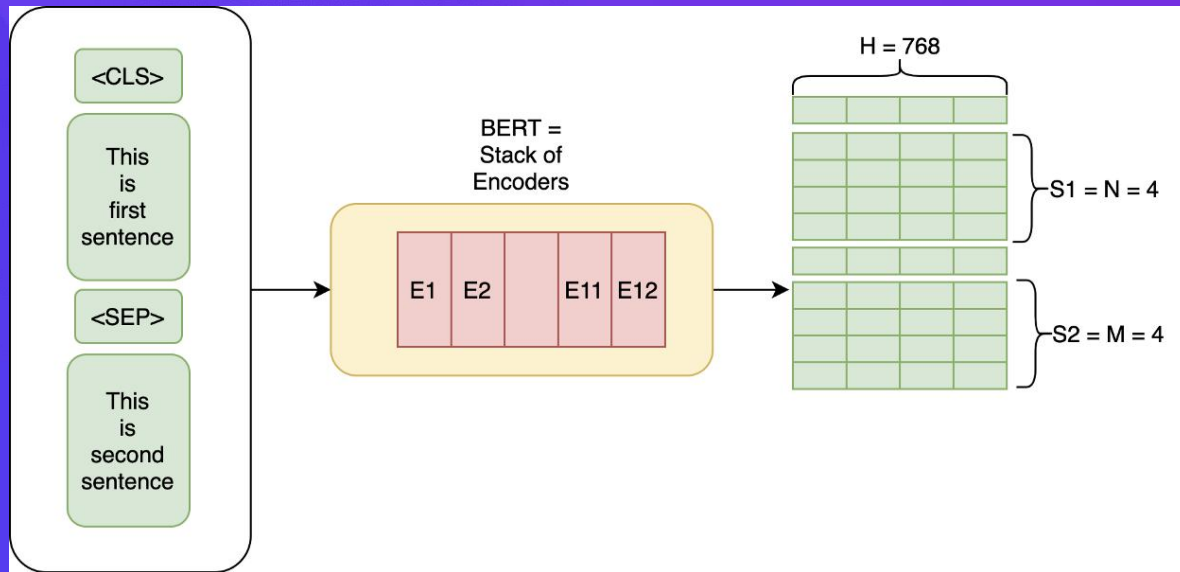
Jieba



Feature Extraction Techniques

Processing Text Data for ML Models

- **BERT Model:** Utilized the pre-trained feature extractor of BERT for context-aware word vector extraction. BERT's ability to understand the context of text reduces the need for extensive preprocessing.



Model Training Details

Training environment

Device

- PC: Zephyrus G14 Laptop
- CPU: AMD Ryzen 9 5900HS
- RAM: 32GB
- GPU: NVIDIA GeForce RTX 3060 Laptop(VRAM: 6GB)

Software environment

- OS: Windows 10
- Python 3.8.18
- CUDA 11.7
- gensim==4.3.0
- jieba==0.42.1

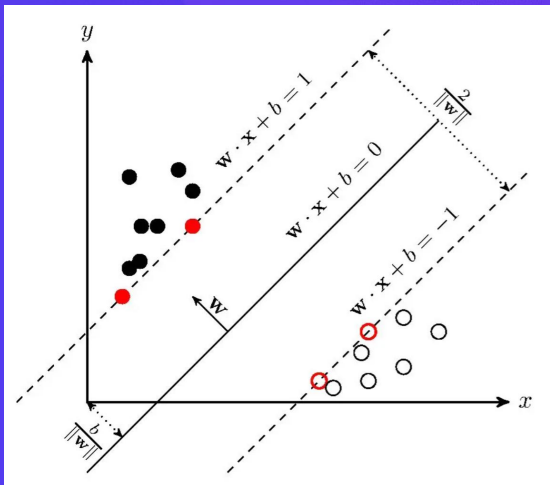
Software environment(cont.)

- joblib==1.2.0
- tensorflow==2.10.1
- keras==2.10.0
- matplotlib==3.7.2
- numpy==1.21.0
- pandas==2.0.3
- scikit-learn==1.3.0
- seaborn==0.12.2
- selenium==3.141.0
- transformers==4.35.2
- tqdm==4.66.1

Model Training Details

Implementing SVM, LSTM, and BERT Models

- **Support Vector Machine (SVM):** Utilized GridSearchCV for model optimization. Optimal parameters: $C = 100$ (regularization), kernel = 'rbf', gamma = 'auto', and this model takes less than 1 minute to train.



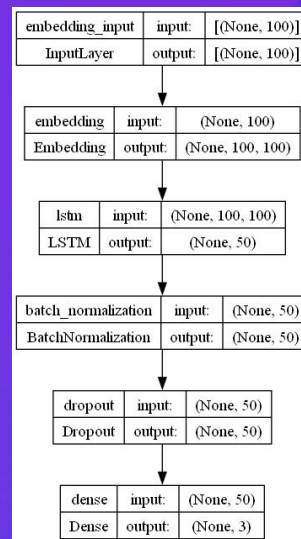
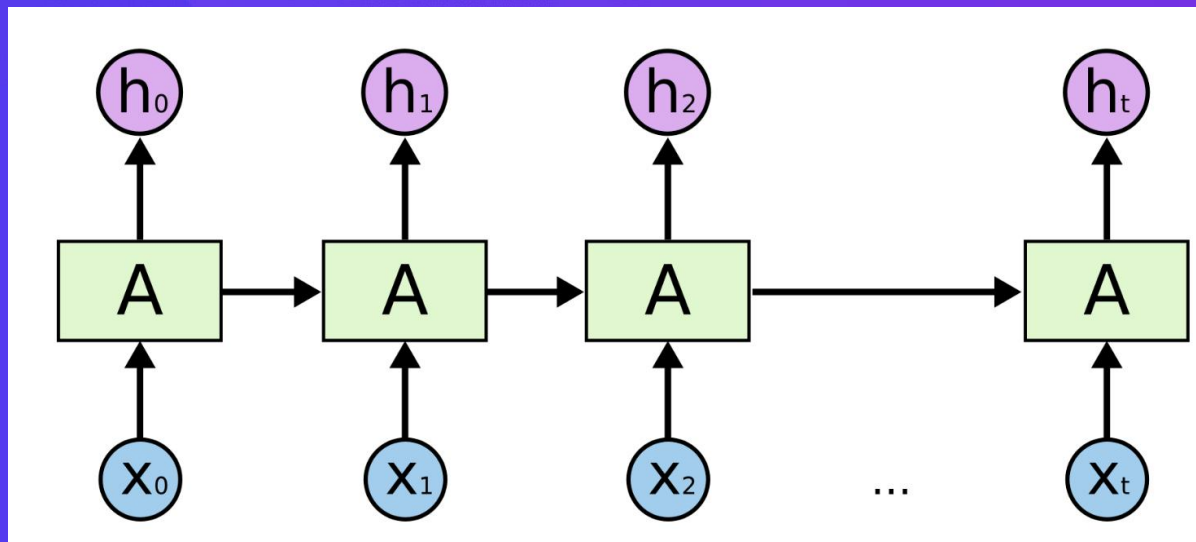
```
# 搜索的参数网格
# param_grid = {
#     'C': [0.1, 1, 10, 100], # 正则化参数
#     'kernel': ['rbf', 'linear', 'poly', 'sigmoid'], # 核函数类型
#     'gamma': ['scale', 'auto'],
# }

# 最佳结果
param_grid = {
    'C': [100], # 正则化参数
    'kernel': ['rbf'], # 核函数类型
    'gamma': ['auto'],
}
```

Model Training Details

Implementing SVM, LSTM, and BERT Models

- **Long Short-Term Memory (LSTM):** It is a variant of Recurrent Neural Network (RNN). Designed to address long-term dependencies in sequence data.



Model Training Details

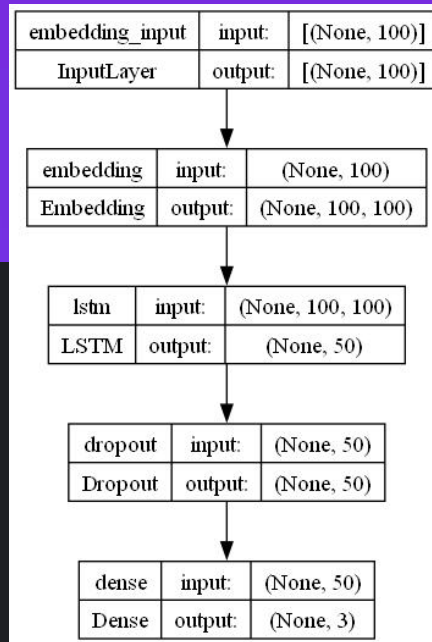
Implementing SVM, LSTM, and BERT Models

- In the model I built, the model structure diagram drawn by calling `keras.utils.vis_utils` and the code details are as shown on the right

Some important parameter:

- `input_length`: 100, the maximum length of the input sequence
- `loss`: `categorical_crossentropy`,
- `learning_rate`: 0.001, which is the default learning rate value of Adam optimizer
- `epoch`: 20, but I set **early_stopping**, which means that the model will be stopped when overfitting begins, and only the one with the latest accuracy during validation will be retained
- `batch size`: 64, it is the maximum value that my device can bear

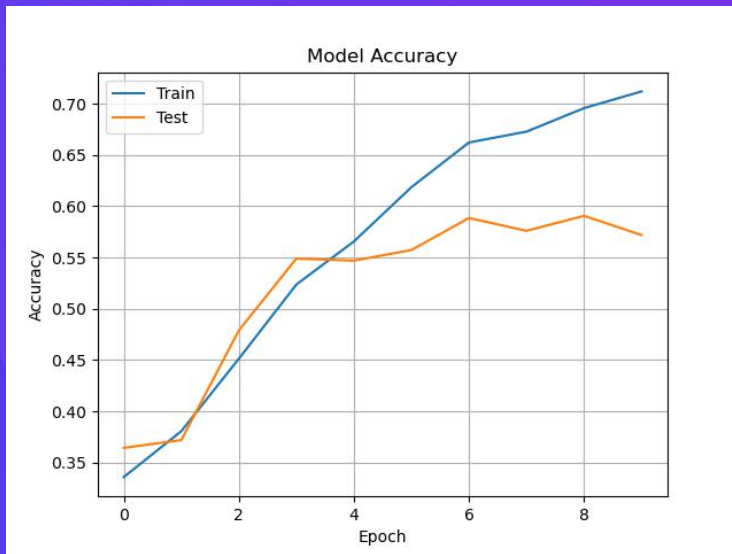
```
model = Sequential([
    Embedding(
        output_dim=vocab_dim,
        input_dim=n_symbols,
        mask_zero=True,
        weights=[embedding_weights],
        input_length=input_length
    ),
    LSTM(units=50, activation='tanh',
        dropout=0.5,
        recurrent_dropout=0.5,
        kernel_regularizer=l2(0.01),
        recurrent_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(units=3, activation='softmax', kernel_regularizer=l2(0.01))
])
```



Model Training Details

Implementing SVM, LSTM, and BERT Models

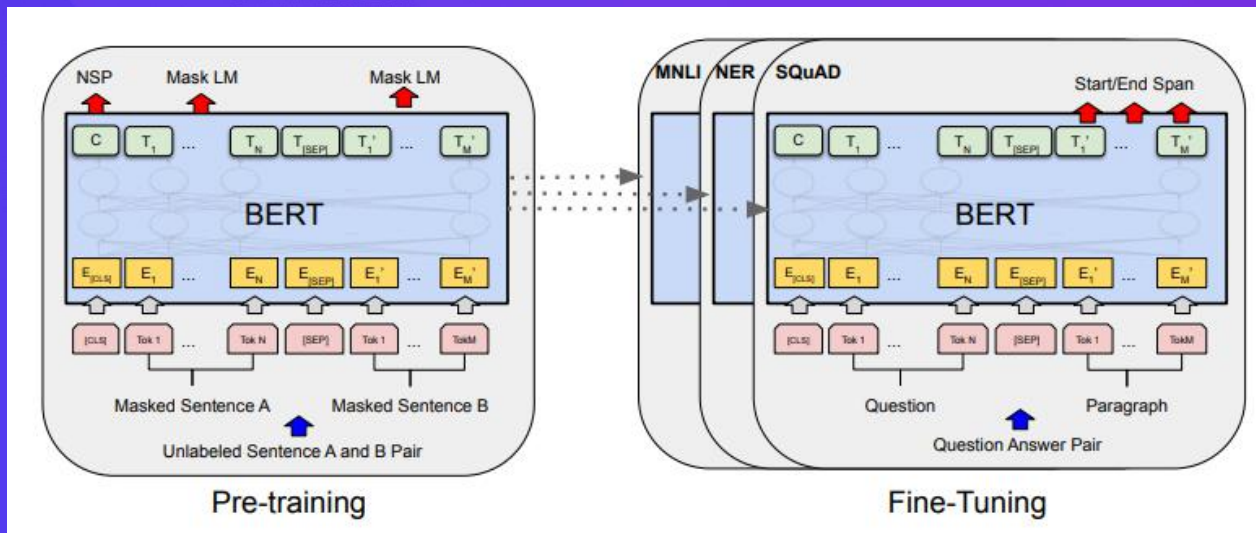
- In the end, the LSTM only stopped after training for 10 epochs. The relationship between the accuracy during training and epoch is as follows:



Model Training Details

Implementing SVM, LSTM, and BERT Models

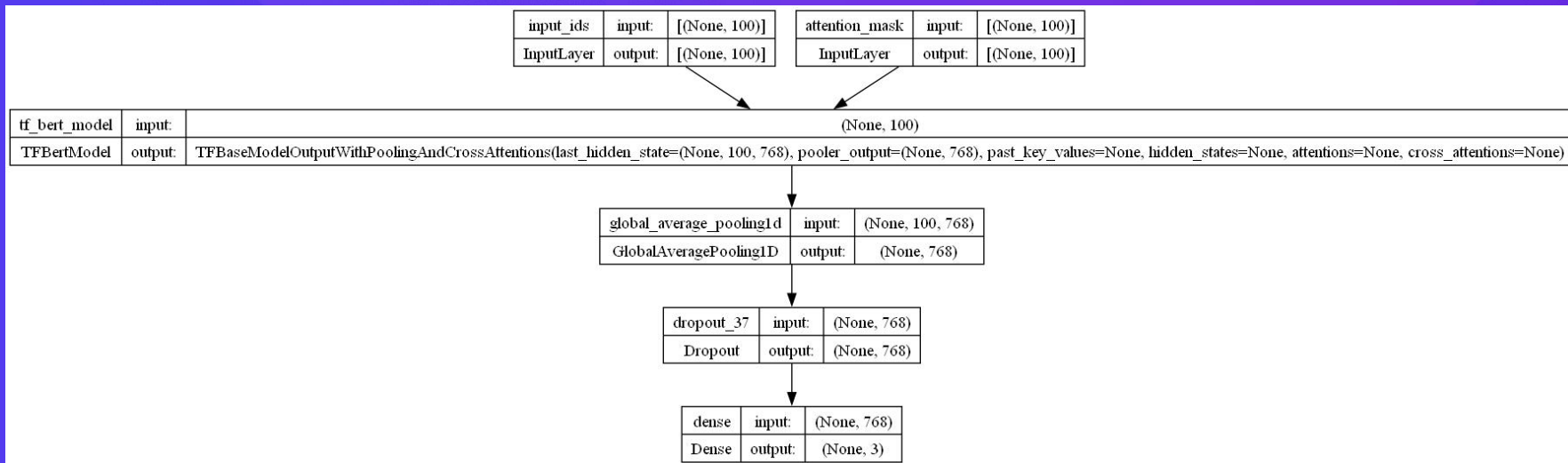
- **Bidirectional Encoder Representations from Transformers (BERT):** Employed BERT-based-Chinese from huggingface for contextually rich feature extraction. Initially used BERT's sequence layer for outputs, later enhanced with pooling, dropout, and additional output layers for improved performance.



Model Training Details

BERT Model fine-tune structure

- In the model I built, the model structure graph drawn by calling `keras.utils.vis_utils` is as follows:



Model Training Details

BERT Model fine-tune structure

- When building the BERT model, I once used **model = TFBertForSequenceClassification.from_pretrained('models/Transformer_model/bert_base_Chinese', num_labels=3)** to directly complete the classification without adding other layers. There seems to be no problem in terms of code, but the training effect is very poor (Acc about 0.33, the model is completely guessing).
- I looked extensively in different places and someone had the same problem as me, but he didn't seem to know how to solve it.
- Finally, I summarized the rules from the correct writing methods of some others, used the feature output of the BERT model, and added pooling, dropout, and fully connected layers myself, and the final model effect became good.

```
bert = TFBertModel.from_pretrained('../models/Transformer_model/bert_base_Chinese')

input_ids = Input(shape=(max_len,), dtype='int32', name="input_ids")
attention_mask = Input(shape=(max_len,), dtype='int32', name="attention_mask")

bert_output = bert(input_ids, attention_mask=attention_mask)[0]
pooled_output = GlobalAveragePooling1D()(bert_output)
dropout = Dropout(0.3)(pooled_output)
output = Dense(units=3, activation='softmax')(dropout)

model = Model(inputs=[input_ids, attention_mask], outputs=output)
model.compile(optimizer=Adam(learning_rate=3e-5), loss='categorical_crossentropy', metrics=['categorical_accuracy'])

plot_model(model, to_file='./model_structure.png', show_shapes=True, show_layer_names=True)

return model
```

Some important parameter:

- max_len: 100, the maximum length of the input sequence
- learning_rate: 3e-5, when fine-tuning the BERT model, the optimal learning rate range is within [5e-5, 3e-5, 2e-5]
- epoch: 1, BERT's fine-tuning only requires a few epochs of training. After experiments, 1 time is the optimal value for this task.
- batch size: 16, it is the maximum value that my device can bear

Model Evaluation

Assessing Model Performance

- **Accuracy:** Measures the proportion of correct predictions to total predictions. Provides an overall effectiveness of the model, but less reliable for imbalanced datasets.
- **Precision:** Indicates the proportion of positive identifications that were actually correct. Crucial when the cost of false positives is high.
- **Recall:** Reflects the proportion of actual positives correctly identified. Important when missing a positive instance is costly.
- **F1 Score:** The harmonic mean of Precision and Recall, balancing the two metrics. Useful when both false positives and false negatives are a concern.
- **Confusion Matrix Analysis:** Used to visualize the performance of each model, highlighting true positives, false positives, true negatives, and false negatives.

```
precision = precision_score(true_labels, predictions, average='weighted')
recall = recall_score(true_labels, predictions, average='weighted')
f1 = f1_score(true_labels, predictions, average='weighted')
accuracy = accuracy_score(true_labels, predictions)

print("Precision: %f" % precision)
print("Recall: %f" % recall)
print("F1 Score: %f" % f1)
print("Accuracy: %f" % accuracy)
```

```
def confusion_matrix_plot(true_labels, predictions):
    cm = confusion_matrix(true_labels, predictions)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='g', cmap='Blues')
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.show()
```


Model Evaluation

Four evaluation indicators for the model

● SVM

Accuracy: 0.611111

Precision: 0.613478

Recall: 0.611111

F1 score: 0.596920

● LSTM

Accuracy: 0.616667

Precision: 0.624210

Recall: 0.616667

F1 score: 0.619433

● BERT

Accuracy: 0.811111

Precision: 0.812916

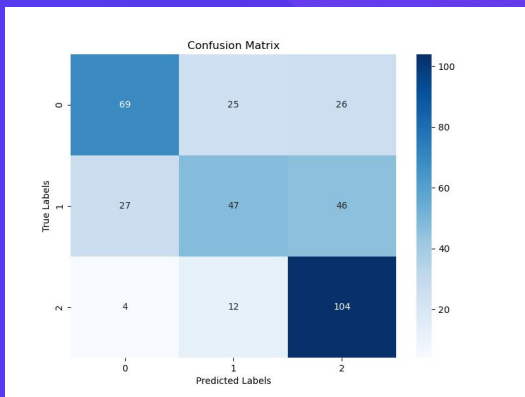
Recall: 0.811111

F1 score: 0.811114

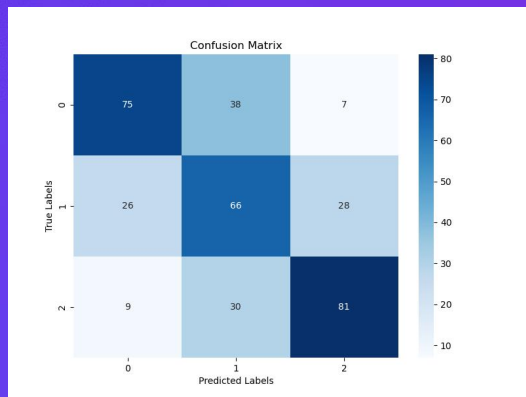
Model Evaluation

Confusion Matrix Analysis

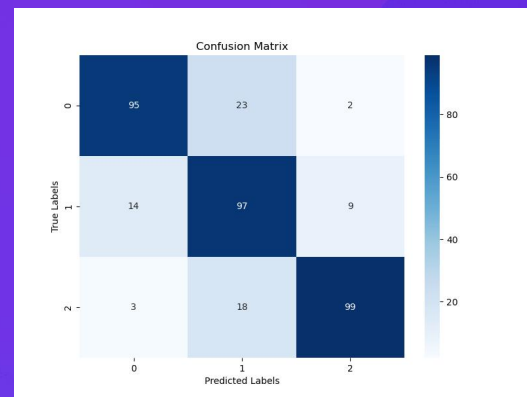
● SVM



● LSTM



● BERT



Conclusion and Outlook

Comparison among above models

- SVM: As a traditional machine learning algorithm, the SVM model has the fastest training speed and the fastest inference speed, but its accuracy and other indicators are low. Even if I use GridSearchCV to find the optimal parameters, its accuracy is not very high.
 - LSTM: The final performance of LSTM is slightly better than the SVM model, but it always has the problem of over-fitting. I failed to solve this problem perfectly, resulting in the accuracy of the model being only usable.
 - BERT: BERT performed best in the construction of this project. Its only shortcoming is that the device requirements are higher. Since the training epochs are fewer, it is still lower than the LSTM model in terms of training time, but in subsequent inference BERT will take up more resources.
-

Conclusion and Outlook

Analysis of causes of deficiencies

- During the data collection process, I decided that my dataset was not an excellent dataset and the information it contained could be misleading because neutrality is a hard-to-define metric. As can be seen in the confusion matrix results of the two worse performing models, their classification accuracy for neutral evaluations is lower than the other two categories.
- For neutral reviews, it was mentioned earlier that their text length is much longer than the other two types of reviews, which may also be the reason for the poor model training effect.
- The content of this data set only uses 6,000 evaluation information, which is not a large data set. If more data is added, the overfitting phenomenon of LSTM and BERT models may be improved, allowing them to learn more features.
- Due to device reasons, I did not complete the training of the deep learning framework on a device with larger memory and video memory. On a more powerful device, I can increase the batch size and the length of the input sequence, which can help the model learn more information.

Conclusion and Outlook

Project Outlook

- In the process of data processing, we can look for a more accurate word list to divide words. At the same time, for BERT, its tokenizer's effect is also good, so we can consider combining the two models with poor performance and BERT's tokenizer to do compare experiments to see if the performance of SVM and LSTM models can be improved.
- I tried to deploy the trained model and implement the interaction between the model and the user.
- Since three classifications can be achieved, an evaluation classification model based on star ratings may also can be constructed.



Thank you

H3Art