1、数据预处理

代码处理逻辑及步骤说明：

1. 初始化准备阶段

- **加载自定义词典**：`jieba.load_userdict("custom_dict.txt")` 确保分词时能识别专业词汇
- **导入停用词表**：从 `stopwords_cn.txt` 加载停用词，用于后续过滤
- **依赖库导入**：包括数据处理（pandas/numpy）、正则处理（re）、分词（jieba）、进度条（tqdm）

2. 数据清洗阶段

- **原始数据读取**：`pd.read_excel('data.xlsx')`
- **内容合并**：将"标题"和"详情"列合并为新列"内容"
- **关键清洗步骤：

```
Pythondf['内容'].apply(preprocess_word)  # 去除#标签、@用户、英文、数字等
df['内容'].apply(emjio_tihuan)     # 去除表情符号和特殊格式
df['内容'].apply(yasuo)            # 机械压缩重复文本
```

- **空值处理**：`dropna` 两次过滤空内容

3. 分词处理阶段

- **词性标注分词**：`pseg.cut` 实现带词性的分词
- **词性过滤**：保留名词(n/Ng)、形容词(a/Ag/an/ad)、动词(v)
- **二次过滤**：剔除停用词、单字词
- **异常处理**：`try-except` 保证流程稳定性

4. 结果输出阶段

- 删除中间列"内容"，保留分词结果
- 输出到 `new_data.xlsx`，不保留索引

关键处理逻辑图示

```
原始数据 → 合并文本 → 正则清洗 → 机械压缩 → 词性过滤分词 → 停用词过滤 → 输出结果
```

2、LDA建模

代码处理逻辑及步骤说明：

1. 初始化与数据准备阶段

```
Pythondf = pd.read_excel('new_data.xlsx')
df['发布时间'] = pd.to_datetime(df['发布时间'])  # 时间列标准化
df['quarter'] = df['发布时间'].dt.to_period('Q')  # 按季度分组
```

- **输入数据**：读取预处理后的`new_data.xlsx`
- **时间处理**：将时间列转换为季度周期对象（如`2024Q2`）
- **循环处理**：对每个季度数据调用`lda()`函数

2. LDA模型核心处理流程

```
Pythondef lda(df,name):
    # 停用词二次加载
    # 数据再过滤（长度≥2且非停用词）
    # 构建词典和词袋模型
    dictionary = corpora.Dictionary(train)
    corpus = [dictionary.doc2bow(text) for text in train]
```

- **二次过滤**：虽然数据已预处理，仍重新过滤停用词和短词
- **语料构建**：将分词结果转换为gensim要求的`(词ID, 词频)`格式

3. 主题数调优与模型评估

```
Pythonfor i in tqdm(range(2, 16)):
    lda_model = gensim.models.ldamodel.LdaModel(...)
    # 计算困惑度（perplexity）和一致性（coherence）
    # 生成双指标折线图
```

- **遍历测试**：尝试2-15个主题数
- 评估指标
  - 困惑度：模型对数据的解释能力（越小越好）
  - 一致性：主题内词的语义相关性（越大越好）
- **可视化保存**：保存`困惑度和一致性.png`及CSV数据

4. 最佳模型训练与可视化

```
Pythonoptimal_z = max(z_data)  # 选择一致性最高的主题数
lda = gensim.models.ldamodel.LdaModel(...)  # 训练最终模型
pyLDAvis.save_html(data1, f'./{name}/lda.html')  # 交互式可视化
```

- **自动选择**：根据最大一致性确定最佳主题数
- **可视化输出**：生成可交互的LDA主题分布图

5. 主题分析与结果输出

```
Pythondf['主题概率'] = list3  # 各文档的主题概率分布
df['主题类型'] = list2  # 最大概率对应的主题ID
# 生成主题词分布表
# 绘制主题相似性热力图
```

- **主题分配**：记录每个文档的所属主题及概率
- **关键词提取**：提取每个主题的前20个关键词及其权重
- **热力图**：使用Plotly生成主题间余弦相似度矩阵

关键流程示意图

```
原始数据 → 季度分割 → 语料构建 → 主题数调优 → 模型训练 → 可视化输出
      ↓              ↓              ↓              ↓
   时间处理      停用词过滤     困惑度/一致性计算   主题词提取
                                    ↓              ↓
                                最佳主题选择    相似性热力图
```

3、arima数据建模

代码处理逻辑及步骤说明：

1. 数据加载与预处理

- **读取数据**：`pd.read_excel('new_data.xlsx')`
- **时间标准化**：`发布时间`列转datetime并提取月份（`dt.to_period('M')`）
- 数值清洗

  ：

  ```
  Pythondef data_process(x):  # 处理"万"单位（1.2万→12000）
  df['笔记热度'] = 点赞+评论+分享  # 创建目标变量
  ```

- **月度聚合**：`groupby('month').mean()`
- **时间序列补全**：生成完整月份范围并填充缺失值为0

2. 模型构建阶段

```
Pythonts = merged_df.set_index('month').to_timestamp(freq='M')  # 时间序列转换
diff = ts.diff().dropna()  # 一阶差分
adf_test(diff)  # 平稳性检验
plot_acf/pacf(diff)  # 自相关图分析
```

- **参数调优**：网格搜索p(0-2)、q(0-2)、d=1的最优组合（AIC最小化）
- **模型训练**：`ARIMA(ts, order=best_order)`
- **模型诊断**：残差QQ图、残差分布等可视化

3. 预测与输出

```
Pythonforecast = final_results.get_forecast(steps=1)  # 预测下月数值
# 可视化实际值 vs 预测值（带置信区间）
merged_df.to_excel('./ARIMA_DATA/月份热度原始数据.xlsx')  # 数据持久化
```

关键处理流程图示

```
 原始数据 → 时间标准化 → 热度计算 → 月度聚合 → 缺失补零 → 差分平稳化 → ACF/PACF分析 → 参数搜索 → 模型训练 → 预测输出
```

4、机器学习预测

1. 数据整合与初始化

```
Pythondf1 = pd.read_csv('./2024Q2/lda_data.csv')  # 读取各季度主题分析结果
df = pd.concat([df1,df2,df3,df4,df5])  # 纵向合并跨季度数据
```

- **数据来源**：整合LDA主题分析后的季度数据文件
- **时间处理**：提取日期粒度（`dt.to_period('D')`），按天排序

2. 特征工程与清洗

```
Pythondef data_process(x):  # 处理"万"单位换算
df['类型'] = label_encoder.fit_transform(...)  # 标签编码
```

- **数值清洗**：统一"万"单位、缺失值填充0
- 特征选择
  - 文本特征：`fenci`（已分词的文本）
  - 数值特征：标题字数、图片数量、社交指标等
  - 目标变量：`点赞数`

3. 特征处理管道

```
Pythonpreprocessor = ColumnTransformer([
    ('text1', TfidfVectorizer(max_features=100), 'fenci'),
    ('num', StandardScaler(), numeric_features)
])
```

- **文本处理**：TF-IDF向量化（限制100个关键词）
- **数值处理**：标准化缩放

4. 模型训练与调优

```
Pythonmodels = {
    'RandomForest': Pipeline + GridSearchCV,
    'RidgeRegression': Pipeline + GridSearchCV
}
```

- **模型选择**：随机森林与岭回归对比
- 参数搜索
  - 随机森林：树数量（100/200）、深度（无限制/10）
  - 岭回归：正则化强度（0.1/1.0）
- **评估指标**：R^2、MSE、MAE

5. 结果分析与可视化

```
Pythonresults_df.to_excel('model_performance.xlsx')  # 保存评估结果
sns.barplot(...)  # 模型性能对比
sns.heatmap(...)  # 数值特征相关性
```

关键流程示意图

```
数据加载 → 时间处理 → 单位换算 → 特征编码 → 管道处理 → 模型训练 → 评估对比
              ↓              ↓              ↓              ↓
          日期标准化      万→整数转换     标签/文本处理    超参调优
```