**2022-2023学年第二学期**

**《大数据分析与领域建模》**

**期末课程报告**

**班 级 大数据2004**

**姓 名 蒋俊涛**

**学 号 2011020063**

**指导教师 汤小月**

**日 期 2023年6月9日**

动漫数据集及评论分析和可视化报告

蒋俊涛 大数据2004 2011020063

摘要

本报告对“MyAnimeList - Anime Dataset with Reviews”数据集进行了深入分析和可视化。本报告描述了使用 Apache Spark 对大规模动漫评论数据进行分析的过程和结果。目标是通过分析评论内容来获取用户对不同动漫的情感倾向，以及探索动漫的流行趋势和观众的偏好。通过 Spark 的分布式计算能力，使得我能够高效地处理和分析大规模数据集。这个数据集来源于Kaggle，包含S.no，标题，评论日期，用户名和文本五列。S.no 是每条评论的唯一标识符，标题是被评论的动漫名称，评论日期是评论发布的日期，用户名是发布评论的用户，而文本则包含用户的评论内容。我首先使用Python对数据进行预处理，然后使用Hadoop分布式文件系统（HDFS）存储处理后的数据。接下来，利用Spark对数据进行分析，包括词频统计和聚合。最后，使用Python的matplotlib，seaborn和wordcloud库进行数据可视化。

**Abstract**

This report conducts an in-depth analysis and visualization of the “MyAnimeList - Anime Dataset with Reviews” dataset. It describes the process and results of using Apache Spark to analyze large-scale anime review data. The objective is to understand users' sentiment towards different animes by analyzing the content of the reviews and to explore trends and preferences in anime. The distributed computing capabilities of Spark enable me to efficiently handle and analyze a large volume of data. This dataset, sourced from Kaggle, comprises five columns: S.no, title, review date, username, and text. S.no is a unique identifier for each review, title refers to the name of the anime being reviewed, review date indicates when the review was posted, username is the user who posted the review, and text contains the content of the user's review. I initially used Python for data preprocessing and then stored the processed data in the Hadoop Distributed File System (HDFS). Subsequently, I employed Spark for data analysis, including word frequency counting and aggregation. Finally, I utilized Python's matplotlib, seaborn, and wordcloud libraries for data visualization.

# 一、 数据预处理

## （1）关于数据集的信息

使用的数据集名称是MyAnimeList - Anime Dataset with Reviews来自kaggle，<https://www.kaggle.com/datasets/ansh0007/myanimelist-anime-dataset-with-reviews>

图形用户界面, 文本, 表格, 电子邮件

描述已自动生成

数据集包含五列信息，即 S.no、标题、评论日期、用户名和文本。S.no 列包含数据集中每条评论的唯一标识符，而标题列包含正在审核的动漫的名称。“评论日期”列指示发布评论的日期，而“用户名”列显示发布评论的人员的用户名。最后，文本列包含用户对相关动漫留下的实际评论或评论。在数据集中拥有大量评论也意味着代表各种各样的意见和观点。这对于想要探索动漫社区内意见多样性或调查不同因素（例如流派或发布日期）如何影响用户意见的研究人员非常有用。

## （2）数据清洗

数据预处理是清洗和结构化数据以进行分析的关键步骤。在此报告中，使用Python进行数据预处理。加载原始数据并删除不必要的列。然后对数据进行清洗，以删除任何包含缺失或损坏数据的记录。

本数据集中包含了从2021年9月到2023年2月的31993条数据，首先我需要一个完整的年份跨度，方便后续将数据按照季度来划分数据，所有首先筛选了从2022年到目前2023年的24396条数据，同时由于用户评论中包含有大量文本信息，所以对于每条数据我截断了每位用户的前30个词，同时由于有的用户会多次评论同一部动漫作品，所以为了保持统计的客观性我只选取每位用户的第一次评论的动漫数据，评论数据中由于会保留有emoji所以需要晒去，同时对于动漫评论数据只保留tag标签为推荐的动漫数据，经过以上筛选后保留有最后的19696条用户评论的动漫数据

# 二、 数据分析

Spark是处理大数据的强大工具，可以高效地进行分布式处理。我利SparkCore和SparkSQL组件对数据进行处理。通过Scala语言编写程序，执行了词频统计，对数据集进行分析，包括分类和聚合。

# 三、 可视化

使用Python的matplotlib，seaborn和wordcloud库创建可视化图表：

1. 评论关键词的词云图 - 通过词云图展示评论中最常用的单词。

2. 动漫标题的词云图 - 通过词云图展示最受欢迎的动漫标题。

3. 每月评论数量的柱状图 - 通过柱状图展示每个月的评论数量的变化。

4. 最受欢迎的动漫标题的柱状图 - 显示最受欢迎的动漫及其评论数量。

5. 每个季度评论占比饼图

6. 不同季度评论前20动漫的柱状图

7. 2021-2023评论数前20的动漫排名

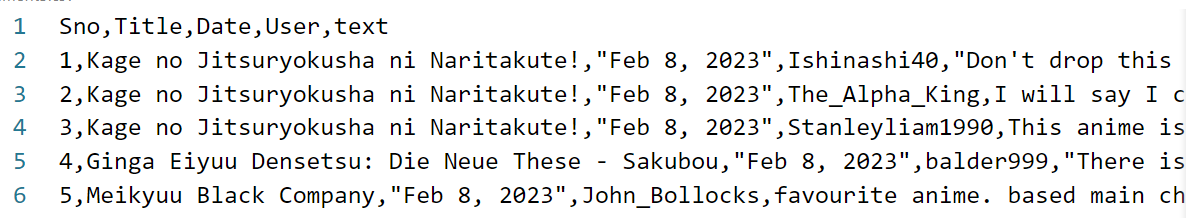
# 四、 实验过程以及相关结果

## 1、 对原始数据的清洗

原始数据中包含了部分不完整的2021年评论数据所以选择删除2021占了少部分的不完整数据，保留2023年2个月的数据是因为2023年的数据具有时效性，更能反映用户对于动漫的评价，而由于每位用户会多次评论同一作品所以只保留对于该作品的第一次评论的数据，而对于数据中tag为recommend的作品则保留，其余作品数据则删除，由于text中是用户对于作品的评论词，充斥有无意义的表情包和大量的空格，所以选择删除包含emoji的数据同时每个评论截选前30个词。进行数据清洗的python代码如下所示：

|  |
| --- |
| import pandas as pd  import re  # 定义一个函数检查文本中是否包含emoji  def contains\_emoji(text):      # 定义一个正则表达式匹配emoji字符      emoji\_pattern = re.compile("["          u"\U0001F600-\U0001F64F"  # emoticons          u"\U0001F300-\U0001F5FF"  # symbols & pictographs          u"\U0001F680-\U0001F6FF"  # transport & map symbols          u"\U0001F1E0-\U0001F1FF"  # flags (iOS)                             "]+", flags=re.UNICODE)      # 返回文本是否包含emoji      return bool(emoji\_pattern.search(text))  # 定义一个函数修剪评论文本  def trim\_comment(text):      # 去除多余的空白符      text = " ".join(text.split())      # 截断至30个词      text = " ".join(text.split()[:30])      return text  # 加载Excel数据文件  df = pd.read\_excel('anime\_reviews.xlsx')  # 过滤数据，仅保留"Recommended"标签的评论  df = df[df['Tag'] == 'Recommended']  # 删除包含emoji的评论  df = df[~df['text'].apply(contains\_emoji)]  # 应用trim\_comment函数，截断评论至30个词并去除空行  df['text'] = df['text'].apply(trim\_comment)  # 删除重复项，每位用户对每部作品只能评论一次，如果有多次评论，保留第一条评论  df = df.drop\_duplicates(subset=['User', 'Title'], keep='first')  # 删除不需要的列，这里删除'Tag'列  df = df.drop(columns=['Tag'])  # 仅保留前24396条记录  df = df.head(24396)  # 修改列名，将'S.no'重命名为'Sno'  df.rename(columns={'S.no': 'Sno'}, inplace=True)  # 重新编排Sno列，使其成为一个连续的序列  df['Sno'] = range(1, len(df) + 1)  # 将清理后的数据保存为CSV文件  df.to\_csv('comment3.csv', index=False) |

进行清洗过后的数据文件comment3.csv如下所示：

包含Sno,Title,Date,User,text  


## 2、 将其上传至hdfs中

文本

描述已自动生成

#### 2.1 用户评论词云图

采用scala语言进行spark编程对下面hdfs的/目录下的数据文件commment3.csv按照text属性去除虚词和无意义的词进行词频统计，并在根目录下产生一个wordcount.csv文件储存有用户评论的词频统计的结果，以方便我来后期用python的wordcloud来生成词云图

代码如下所示：



这段Scala代码主要用于Apache Spark的数据处理，具体来说，它读取了一份存在于HDFS上的comment.csv文件，然后对其中的"text"字段进行了词语划分（tokenization），删除了停用词（stop words），最后计算了每个词的出现次数，并按次数降序排序，最后将结果写入wordcount.csv文件。

程序运行的结果如下：  
图形用户界面, 应用程序

描述已自动生成

#### 2.1.1 将产生的结果文件从hdfs中下载到本地并且重名名为result.csv，产生用户评论的词云图

result.csv的部分数据如下所示：

图形用户界面, 文本, 应用程序

描述已自动生成

上面表示的信息如anime被用户提起过1116次

2.1.2 生成用户评论热词词云图

Python代码如下所示：

|  |
| --- |
| import matplotlib.pyplot as plt  from wordcloud import WordCloud  import pandas as pd  # 读取CSV文件  data = pd.read\_csv('result.csv', error\_bad\_lines=False)  # 转换数据到字典格式，用于词云的输入  word\_freq = dict(zip(data['word'], data['count']))  # 定义要移除的无意义词汇  stopwords = set(["much", "lot", "one", "say"])  # 创建词云对象  wordcloud = WordCloud(width=800, height=800,                        background\_color='white',                        stopwords=stopwords,                        min\_font\_size=10).generate\_from\_frequencies(word\_freq)  # 绘制词云图  plt.figure(figsize=(8, 8), facecolor=None)  plt.imshow(wordcloud)  plt.axis("off")  plt.tight\_layout(pad=0)  # 显示图  plt.show() |
|  |



由以上图片可知用户多次提及anime,show,really等词汇，表示了用户对于动漫相关作品的强烈喜爱。

#### 2.2 对动漫的标题进行词频统计，并将统计的词汇结果生成动漫标题词云图

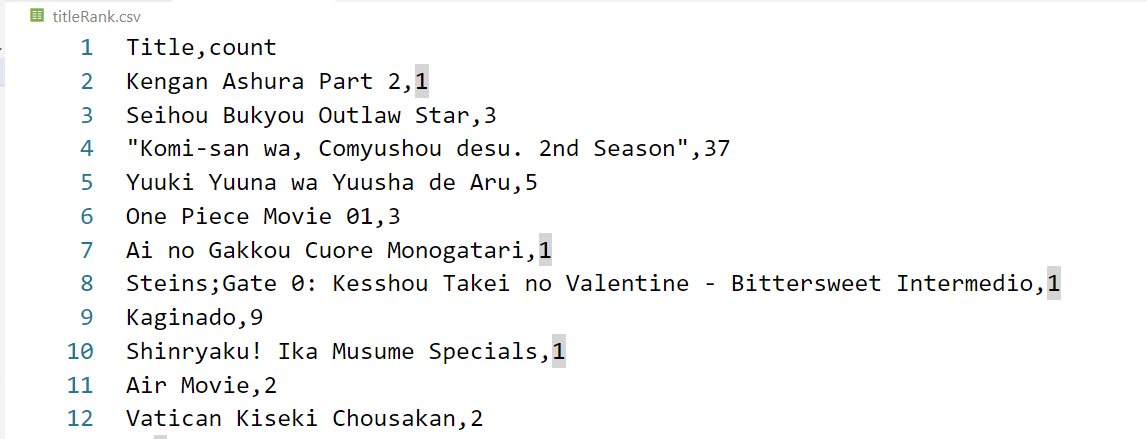
#### 2.2.1 编写scala代码用以对动漫标题词频进行统计，并将结果保存在集群的hdfs中

代码如下所示：

|  |
| --- |
| import org.apache.spark.sql.functions.\_  import org.apache.spark.ml.feature.{Tokenizer, StopWordsRemover}  import org.apache.spark.sql.\_  // 读取comment.csv  val inputPath = "hdfs://hadoop01:9000/comment3.csv"  val df = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load(inputPath)  // Tokenize文本  val tokenizer = new Tokenizer().setInputCol("Title").setOutputCol("words")  val tokenized = tokenizer.transform(df)  // 删除停止词  val remover = new StopWordsRemover().setInputCol("words").setOutputCol("filteredWords")  val filtered = remover.transform(tokenized)  // 计算词频  val wordCounts = filtered.select(explode(col("filteredWords")).as("word")).groupBy("word").count().orderBy(desc("count"))  // 写入wordcount.csv  // 将数据集的分区数减少到1  // 改成输出目录，而不是具体的文件名  val outputPath = "hdfs://hadoop01:9000/Animationwordcount\_output"  wordCounts.coalesce(1).write.format("csv").option("header", "true").mode("overwrite").save(outputPath) |

#### 2.2.2 将hdfs中生成的结果下载至本地命名为titleRank.csv方便生成词云图

titleRank.csv的部分数据如下所示:title表示动漫的名称，count表示评论的数量

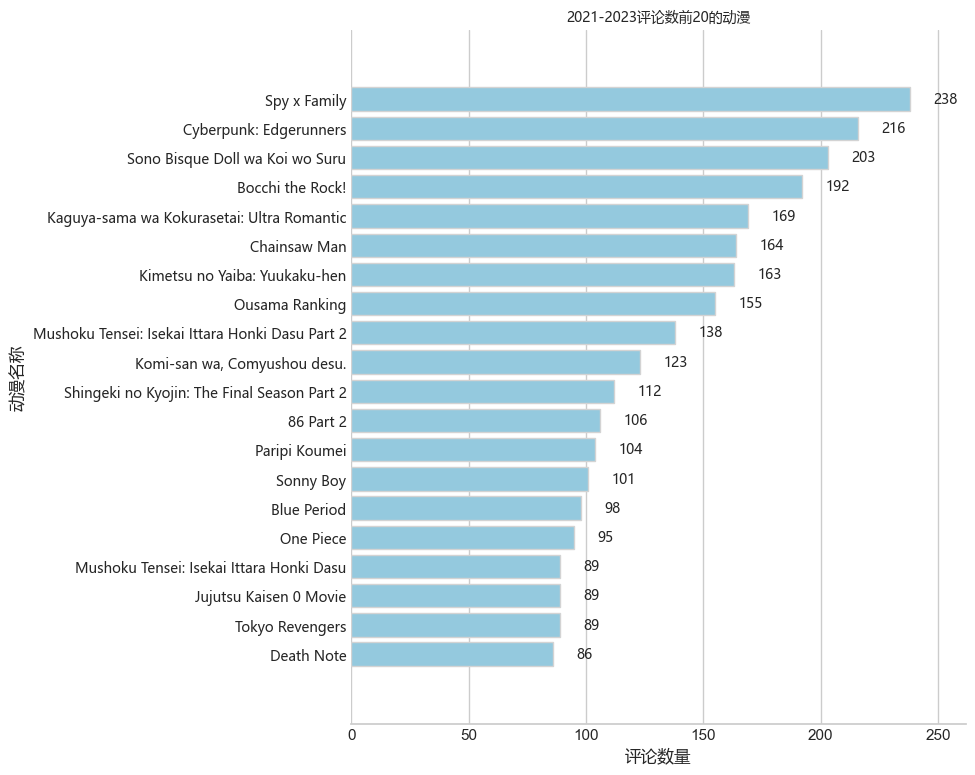


#### 2.2.3 根据titleRank.csv使用python进行可视化处理

使用Matplotlib和Seaborn库绘制一个水平条形图，展示了2021-2023年评论数排名前20的动漫

代码如下所示：

|  |
| --- |
| import matplotlib.pyplot as plt  import matplotlib.font\_manager as font\_manager  import seaborn as sns  # 设置中文字体  font\_path = 'C:\\Windows\\Fonts\\msyh.ttc' # 或者使用原始字符串：r'C:\Windows\Fonts\msyh.ttc'  # 创建字体属性对象  font\_prop = font\_manager.FontProperties(fname=font\_path)  # 设置字体  plt.rcParams['font.family'] = font\_prop.get\_name()  sns.set(font=font\_prop.get\_name())  # 读取CSV文件  data = pd.read\_csv('titleRank.csv')  # 按照count列降序排序  sorted\_data = data.sort\_values(by='count', ascending=False)  # 获取排名前20的动漫  top\_20 = sorted\_data.head(20)  # 设置Seaborn风格  sns.set\_style('whitegrid')  sns.set\_palette('pastel')  # 创建水平条形图  plt.figure(figsize=(10, 8))  sns.barplot(x='count', y='Title', data=top\_20, color='skyblue', edgecolor='lightgray')  # 添加标题和标签  plt.title('2021-2023评论数前20的动漫', fontsize=16, fontproperties=font\_prop)  plt.xlabel('评论数量', fontsize=12, fontproperties=font\_prop)  plt.ylabel('动漫名称', fontsize=12, fontproperties=font\_prop)  # 调整Y轴标签的位置和间距  plt.yticks(range(len(top\_20)), top\_20['Title'], fontsize=10, fontproperties=font\_prop)  plt.tight\_layout()  # 添加数值标签  for i, v in enumerate(top\_20['count']):  plt.text(v + 10, i, str(int(v)), va='center', fontsize=10, fontproperties=font\_prop)  # 调整图表边距  plt.margins(0.1)  # 调整边框样式  sns.despine(left=True)  # 调整刻度线样式  plt.tick\_params(axis='both', which='both', length=0)  # 展示图表  plt.show() |

运行结果如下图所示：  


根据上表数据显示，过去两年中，用户评论次数最多的前20部动漫包括：《间谍过家家》，《赛博朋克：边缘行者》，《更衣人偶坠入爱河》，《孤独的摇滚》，《辉夜大小姐想让我告白：究极浪漫》，《电锯人》，《鬼灭之刃》，《漂流少年》以及其他一些在国外排名靠前的动漫。通过这些数据，我们可以初步了解用户对这20部动漫的喜爱程度。为了深入挖掘这些数据的价值，我们可以继续分析这20部作品的特点和亮点，如故事情节的吸引力、角色的深度、动画的质量和风格等。同时，对比不同动漫的评分和评论，可以进一步揭示不同类型和主题的受欢迎程度。此外，动漫制作公司和从业者可以通过对这些数据的深入分析，更加准确地把握观众的兴趣和偏好。例如，如果某种类型的动漫或特定题材在评论中频繁出现正面评价，那么制作公司可以考虑投入更多资源开发相关题材的作品。同样地，制作公司还可以关注评论中提及的不足之处，以便在未来的作品中进行改进。这种数据驱动的方法有助于制作公司制定更加精准的市场策略，满足观众的需求，从而创作出更加精彩和深受喜爱的动漫作品。

#### 2.2.4 利用spark编程对comment3.csv进一步处理生成对动漫标题的词频统计的词云图

代码如下所示：

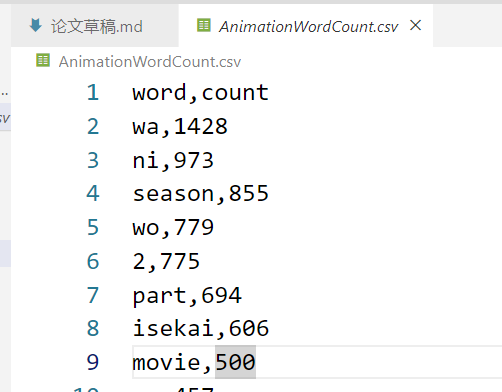
|  |
| --- |
| import org.apache.spark.sql.functions.\_  import org.apache.spark.ml.feature.{Tokenizer, StopWordsRemover}  import org.apache.spark.sql.\_  // 读取comment.csv  val inputPath = "hdfs://hadoop01:9000/comment3.csv"  val df = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load(inputPath)  // Tokenize文本  val tokenizer = new Tokenizer().setInputCol("Title").setOutputCol("words")  val tokenized = tokenizer.transform(df)  // 删除停止词  val remover = new StopWordsRemover().setInputCol("words").setOutputCol("filteredWords")  val filtered = remover.transform(tokenized)  // 计算词频  val wordCounts = filtered.select(explode(col("filteredWords")).as("word")).groupBy("word").count().orderBy(desc("count"))  // 写入wordcount.csv  // 将数据集的分区数减少到1  // 改成输出目录，而不是具体的文件名  val outputPath = "hdfs://hadoop01:9000/Animationwordcount\_output"  wordCounts.coalesce(1).write.format("csv").option("header", "true").mode("overwrite").save(outputPath) |

图形用户界面, 应用程序, 表格

描述已自动生成

将生成的结果下载到本地并重命名为AnimationWordCount.csv

部分数据如下所示：



利用wordcloud对生成的结果生成词云图

python代码如下所示：

|  |
| --- |
| import pandas as pd  from wordcloud import WordCloud  import matplotlib.pyplot as plt  import re  # 读取数据  try:  data = pd.read\_csv('AnimationWordCount.csv', error\_bad\_lines=False, quoting=3)  except Exception as e:  print("读取文件时出错:", e)  exit()  # 检查'word'列中的值是否为字符串类型，如果是，则去除数字，否则保持不变  data['word'] = data['word'].apply(lambda x: re.sub(r'\d+', '', x) if isinstance(x, str) else x)  # 去除无意义的单词  meaningless\_words = ['wa', 'ni', 'wo', 'ga', 'x', 'de', 'na','season','part','movie','nd','sekai']  data = data[~data['word'].isin(meaningless\_words)]  # 删除'word'列中为NaN的行  data = data.dropna(subset=['word'])  # 创建词云图，设置较大的宽度和高度，以及最大字体大小和最大词汇数量  wordcloud = WordCloud(width=1200, height=800, background\_color='white', max\_font\_size=300, max\_words=50).generate\_from\_frequencies(data.set\_index('word')['count'])  # 显示词云图，设置较大的图像尺寸  plt.figure(figsize=(12, 8))  plt.imshow(wordcloud, interpolation='bilinear')  plt.axis('off')  plt.show() |



通过上图可知过去两年(isekai)异世界，间谍（spy）,kaguya-sama(辉夜)一直是动漫界绕不开的话题，表明含有异世界，间谍等元素的动漫作品深受国外观众的喜爱。

#### 2.3 使用spark对comment3.csv数据进行进一步的处理

一般根据日本动漫爱好者的习惯动漫一般分为1月新番也就是1-3月份播出的动画，4月新番是4-7月播放的动画，7月番剧是7-12月播出的动画，同时日本的动画制作公司也是将新推出的作品放到1月4月和7月进行放映，为了更加准确的分析数据需要将comment根据以上的月份信息分为三个季度，以下通过spark编程继续将comment3.csv处理成三个数据集分别是firstSeason.csv表示第一季度（包含1-3月）动画以及用户评论的信息，secondSeason.csv表示第二季度（包含4-7月）的信息，thirdSeason.csv第三季度（7-12月）的动画和用户评论信息,以下是具体实现的scala代码：

|  |
| --- |
| import org.apache.spark.sql.SparkSession  // 创建SparkSession对象  val spark = SparkSession.builder().appName("CSV Processing").getOrCreate()  // 读取CSV文件并创建DataFrame  val df = spark.read.option("header", "true").option("inferSchema", "true").csv("hdfs://hadoop01:9000/comment3.csv")  import org.apache.spark.sql.functions.\_  // 添加日期列，并将日期格式化为"MMM d, yyyy"格式  val dfWithDate = df.withColumn("Date", to\_date(col("Date"), "MMM d, yyyy"))  // 定义各个季度的起始日期和结束日期  val firstSeasonStart = to\_date(lit("Jan 1"), "MMM d")  val firstSeasonEnd = to\_date(lit("Mar 31"), "MMM d")  val secondSeasonStart = to\_date(lit("Apr 1"), "MMM d")  val secondSeasonEnd = to\_date(lit("Jul 31"), "MMM d")  val thirdSeasonStart = to\_date(lit("Aug 1"), "MMM d")  val thirdSeasonEnd = to\_date(lit("Dec 31"), "MMM d")  // 过滤出第一季度（1月至3月）或年份大于2023的数据，并将结果写入CSV文件  dfWithDate.filter((month(col("Date")) >= 1 && month(col("Date")) <= 3) || (year(col("Date")) > 2023))  .coalesce(1)  .write.csv("hdfs://hadoop01:9000/firstSeason.csv")  // 过滤出第二季度（4月至7月）的数据，并将结果写入CSV文件  dfWithDate.filter(month(col("Date")) >= 4 && month(col("Date")) <= 7)  .coalesce(1)  .write.csv("hdfs://hadoop01:9000/secondSeason.csv")  // 过滤出第三季度（8月至12月）的数据，并将结果写入CSV文件  dfWithDate.filter(month(col("Date")) >= 8 && month(col("Date")) <= 12)  .coalesce(1)  .write.csv("hdfs://hadoop01:9000/thirdSeason.csv") |

表格

描述已自动生成

2.3.1 在spark-shell中运行上面的代码后将产生的对应文件下载到本地

分别命名为firstSeason.csv,secondSeason.csv,thirdSeason.csv

firstSeason.csv的部分数据如下

图片包含 日历

描述已自动生成

secondSeason.csv的部分数据如下

文本

描述已自动生成

secondSeason.csv的部分数据如下

文本

描述已自动生成

#### 2.3.1 使用python数据可视化

##### （一）、不同季度评论前20的动漫

使用pandas和matplotlib.pyplot库创建并显示一个包含三个子图的条形图，

根据不同季度的数据集，绘制了每个季度评论数量最多的前20个动漫的条形图，并以子图的形式展示在同一个图表中。python代码如下所示

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  # 设置字体和解决负号显示问题  plt.rcParams['font.sans-serif'] = ['SimHei']  plt.rcParams['axes.unicode\_minus'] = False  # 读取每个季度的数据集  first\_season\_df = pd.read\_csv("firstSeasonWordcount.csv")  second\_season\_df = pd.read\_csv("secondSeasonWordcount.csv")  third\_season\_df = pd.read\_csv("thirdSeasonWordcount.csv")  # 提取标题和评论数量，并按评论数量降序排序  first\_season\_sorted = first\_season\_df.sort\_values(by="count", ascending=False).head(20)  first\_season\_titles = first\_season\_sorted["title"]  first\_season\_counts = first\_season\_sorted["count"]  second\_season\_sorted = second\_season\_df.sort\_values(by="count", ascending=False).head(20)  second\_season\_titles = second\_season\_sorted["title"]  second\_season\_counts = second\_season\_sorted["count"]  third\_season\_sorted = third\_season\_df.sort\_values(by="count", ascending=False).head(20)  third\_season\_titles = third\_season\_sorted["title"]  third\_season\_counts = third\_season\_sorted["count"]  # 创建子图  fig, axs = plt.subplots(3, 1, figsize=(10, 15))  # 颜色  colors = ['#1f77b4', '#ff7f0e', '#2ca02c']  # 绘制第一个季度的条形图  axs[0].barh(first\_season\_titles, first\_season\_counts, color=colors[0])  axs[0].set\_title("1-3月", fontsize=16)  axs[0].set\_xlabel("评论数量", fontsize=14)  axs[0].set\_ylabel("动漫名称", fontsize=14)  # 绘制第二个季度的条形图  axs[1].barh(second\_season\_titles, second\_season\_counts, color=colors[1])  axs[1].set\_title("4-7月", fontsize=16)  axs[1].set\_xlabel("评论数量", fontsize=14)  axs[1].set\_ylabel("动漫名称", fontsize=14)  # 绘制第三个季度的条形图  axs[2].barh(third\_season\_titles, third\_season\_counts, color=colors[2])  axs[2].set\_title("8-12月", fontsize=16)  axs[2].set\_xlabel("评论数量", fontsize=14)  axs[2].set\_ylabel("动漫名称", fontsize=14)  # 调整子图之间的间距  plt.tight\_layout()  # 添加整体标题  plt.suptitle('动漫评论数量分析', fontsize=20, y=1.02)  # 显示图表  plt.show() |

图表, 直方图

描述已自动生成

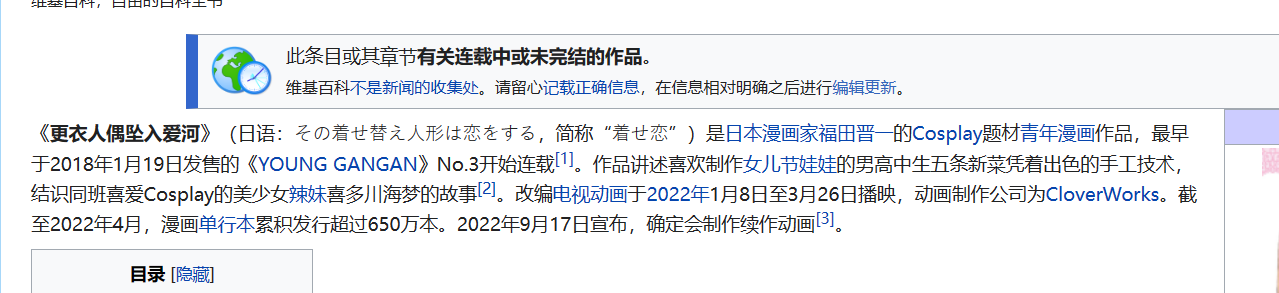
图表

描述已自动生成

图表, 直方图

描述已自动生成

从上述图表中我们不仅可以观察到用户评论的数量，还能清晰地看到各个季度受欢迎的动漫作品。在第一季度（1-3月），《更衣人偶坠入爱河》的评论数量最多，这使我们可以推断该作品在当季度深受观众的喜爱。通过进一步检索维基百科，我们发现《更衣人偶坠入爱河》的播出时间正好是2022年1月8日至3月26日，与我们的统计结果时间段完美吻合。

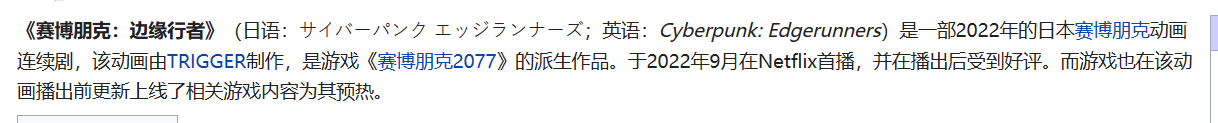


而在第二季度（4-6月），图表显示《间谍过家家》成为了观众的最爱。同样，我们查阅百科资料，发现《间谍过家家》的播出时间恰好与第二季度相符。

文本

描述已自动生成

进入第三季度，图表中的数据表明《赛博朋克边缘行者》成为了最受欢迎的动漫。再次查询维基百科，我们可以确认《赛博朋克边缘行者》的播出时间是在第三季度。



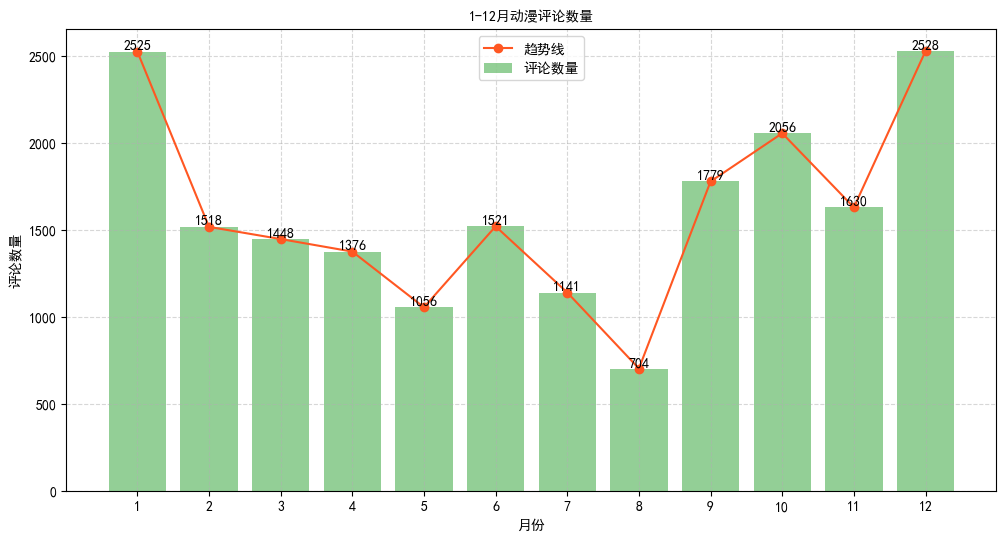
这一系列的数据分析不仅揭示了每个季度观众最喜爱的动漫，还为我们提供了一种可靠的方法来预测或分析动漫作品的播出季度与其受欢迎程度之间的关系。即便是跨越不同年份，我们仍然可以准确地通过评论数量分析出动漫作品的播出时间和受欢迎程度。值得注意的是，评论数量可能受多种因素影响，包括作品的宣传力度、口碑、以及社交媒体的讨论热度等。因此，在分析时还可以结合其他数据和信息，如播放量、收视率和社交媒体上的讨论，以获得更全面和深入的了解。

总的来说，通过精细化的数据分析和结合多个信息来源，我们能够洞察不同季度内受欢迎的动漫作品及其背后的原因，从而为动漫产业的市场策略和内容创新提供数据支持。

##### （二）、 根据每个月的评论量绘制折线统计图

从三个季度的CSV文件中读取评论数据，并计算每个月的评论数量。然后，它使用matplotlib库创建一个柱状图来可视化这些数据，并在柱状图上添加折线图来展示python代码如下所示：

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  from datetime import datetime  # 定义一个函数来读取CSV文件  # 参数：file\_name - 要读取的CSV文件的名称  # 返回：一个DataFrame，包含CSV文件的内容  def read\_csv(file\_name):  # error\_bad\_lines=False，这个参数意味着在读取时忽略错误行  return pd.read\_csv(file\_name, error\_bad\_lines=False)  # 定义主函数  def main():  # 评论文件列表  comment\_files = ['firstSeason.csv', 'secondSeason.csv', 'thirdSeason.csv']    # 初始化一个列表，包含12个元素，每个元素都是0  # 用于存储每个月的评论数量  monthly\_comments = [0] \* 12    # 遍历每个评论文件，计算每个月的评论数量  for comment\_file in comment\_files:  # 读取评论数据  data = read\_csv(comment\_file)  # 遍历数据的每一行  for index, row in data.iterrows():  # 将日期字符串转换为datetime对象  comment\_date = datetime.strptime(row['Date'], '%Y-%m-%d')  # 获取月份  month = comment\_date.month  # 累加每个月的评论数量  monthly\_comments[month - 1] += 1    # 创建一个新的图表，并设置大小  plt.figure(figsize=(12, 6))  # 月份范围 (1-12)  months = range(1, 13)  # 创建柱状图表示每个月的评论数量  bars = plt.bar(months, monthly\_comments, color='#4CAF50', alpha=0.6, label='评论数量')    # 创建折线图表示评论数量的趋势  plt.plot(months, monthly\_comments, color='#FF5722', marker='o', label='趋势线')    # 为每个柱子添加评论数量标签  for bar, comment\_count in zip(bars, monthly\_comments):  height = bar.get\_height()  # 在每个柱子上方添加评论数量  plt.text(bar.get\_x() + bar.get\_width() / 2, height + 0.5, str(comment\_count),  ha='center', va='bottom')    # 添加x轴和y轴的标签以及图表的标题  plt.xlabel('月份')  plt.ylabel('评论数量')  plt.title('1-12月动漫评论数量', fontproperties='SimHei')  plt.xticks(months)  plt.legend()  plt.grid(True, linestyle='--', alpha=0.5)    # 显示图表  plt.show()  # 当脚本作为主程序运行时，执行主函数  if \_\_name\_\_ == "\_\_main\_\_":  main() |



从上面的折线柱状图中，我们可以清晰地观察到1月和12月份的动漫评论数量是最多的，这很可能与冬季是动漫新作品发布的热门时期有关。更为详细地观察，我们还可以发现，在1月、3月和7月附近，评论数量也相对较多。这种现象与动漫产业的常规做法是一致的，因为很多动漫公司通常会选择在每季度初发布新的作品。值得一提的是，从图表中我们还可以看出，第一季度和第三季度的评论数量普遍比第二季度要多。这一发现提示我们，用户可能更偏爱第一和第三季度的动漫作品。这可能是由于在第一季度，新年伊始，很多人寻找新的娱乐内容来开始一年；而在第三季度，夏季假期为很多人提供了额外的时间来观看动漫。同时，动漫公司可能也在第一和第三季度发布更多备受期待的作品，以吸引更广泛的观众。这种策略可能是基于对市场和观众行为的深入了解。对于动漫制作公司和相关从业者来说，了解这些数据和趋势是非常重要的。通过精确地分析评论数量及其与播出时间的关系，他们可以更加明智地制定发布计划和市场策略。例如，如果第一和第三季度的观众参与度较高，公司可能会考虑在这些时段发布重要的新作品，并在第二季度集中精力进行宣传和市场推广。

##### （三）、 每个月最受欢应的动漫及其评论数量

分析和可视化每个月评论最多的动漫及其评论数量。代码首先读取三个CSV文件中的评论数据，然后计算每个月每个动漫的评论数量并存储在一个嵌套的 defaultdict 中。接下来，代码遍历每个月并找出评论最多的动漫以及其评论数量。最后，使用matplotlib库创建一个水平条形图来可视化这些数据，并在每个条形旁边添加动漫名称和评论数量的标签。图表的y轴表示月份，x轴表示评论数量，并显示网格线。Python代码如下：

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  from datetime import datetime  from collections import defaultdict  # 定义一个函数来读取CSV文件  # 参数：file\_name - 要读取的CSV文件的名称  # 返回：一个DataFrame，包含CSV文件的内容  def read\_csv(file\_name):  # error\_bad\_lines=False，这个参数意味着在读取时忽略错误行  return pd.read\_csv(file\_name, error\_bad\_lines=False)  # 定义主函数  def main():  # 评论文件列表  comment\_files = ['firstSeason.csv', 'secondSeason.csv', 'thirdSeason.csv']    # 初始化一个嵌套的 defaultdict，用于存储每个月每个动漫的评论数量  # 默认值为int，即0  monthly\_anime\_comments = defaultdict(lambda: defaultdict(int))    # 遍历每个评论文件，计算每个月每个动漫的评论数量  for comment\_file in comment\_files:  # 读取评论数据  data = read\_csv(comment\_file)  # 遍历数据的每一行  for index, row in data.iterrows():  # 将日期字符串转换为datetime对象  comment\_date = datetime.strptime(row['Date'], '%Y-%m-%d')  # 获取月份  month = comment\_date.month  # 获取动漫标题  anime\_title = row['Title']  # 累加每个动漫的评论数量  monthly\_anime\_comments[month][anime\_title] += 1    # 找出每个月评论最多的动漫  popular\_animes = []  max\_comments = []  # 遍历每个月  for month, animes in monthly\_anime\_comments.items():  # 找出该月评论最多的动漫  popular\_anime = max(animes, key=animes.get)  # 将找到的动漫添加到列表中  popular\_animes.append(popular\_anime)  # 将最多评论数量添加到列表中  max\_comments.append(animes[popular\_anime])  # 创建一个水平条形图  plt.figure(figsize=(12, 8))  months = range(1, 13)  # 绘制每个月评论最多的动漫的评论数量  bars = plt.barh(months, max\_comments, color='#4CAF50', alpha=0.6)    # 添加动漫名称和评论数量标签  for bar, popular\_anime, comment\_count in zip(bars, popular\_animes, max\_comments):  width = bar.get\_width()  # 在每个条形旁边添加动漫名称和评论数量  plt.text(width + 0.5, bar.get\_y() + bar.get\_height() / 2,  f"{popular\_anime} ({comment\_count})", ha='left', va='center')    # 添加y轴和x轴的标签以及图表的标题  plt.ylabel('月份')  plt.xlabel('评论数量')  plt.title('每个月最受欢迎的动漫及其评论数量', fontproperties='SimHei')  # 设置y轴的刻度标签  plt.yticks(months, [f'月份 {month}' for month in months])  # 显示网格  plt.grid(True, linestyle='--', alpha=0.5)    # 显示图表  plt.tight\_layout()  plt.show()  if \_\_name\_\_ == "\_\_main\_\_":  main() |

日程表

描述已自动生成

从上述的数据表格中，我们可以清楚地看到每个月最受欢迎的动漫作品及其评论数量。令人瞩目的是，《赛博朋克边缘行者》在11月份的评论数量远超其他月份作品的评论，这表明它在那个月有着极高的受欢迎程度和观众关注度。这可能是由于《赛博朋克边缘行者》的剧情吸引力、动画质量或是其宣传策略的成功。

进一步观察数据，我们发现《间谍过家家》在4月、6月和12月份成为了当月评论最多的动漫作品。这意味着《间谍过家家》在这三个月里吸引了大量的观众。这可能是由于该动漫在这些月份发布了重要的剧情节点，或者与观众的偏好和当时的市场趋势相吻合。

而对于《孤独摇滚》，我们看到它在2月和8月成为了当月评论最多的动漫作品。这可能意味着《孤独摇滚》在这两个月中有着特别吸引人的内容或者与特定的节日、事件有关联，从而引发了观众的热情。

此外，对于动漫制作公司和分析师来说，了解这些评论的分布和趋势是非常有价值的。它们可以借此洞察不同作品在不同时间段的受欢迎程度，以制定更为精准的市场推广策略和内容规划。

例如，如果某部动漫在特定月份的评论数量激增，那么制作公司可以分析是什么因素导致了这一增长，并在未来的作品中利用类似的策略。此外，也可以通过对比不同动漫在相同时间段的表现，来发现可能的市场趋势和观众喜好。

##### （四）、 每个季度评论占比

分析和可视化每个季度的动漫评论数量占比。首先，代码读取三个CSV文件中的评论数据，然后计算每个季度的评论数量并存储在一个列表中。然后，使用matplotlib库创建一个饼图来可视化这些数据。饼图显示了每个季度的评论数量占总评论的占比，python代码如下所示：

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  from matplotlib.font\_manager import FontProperties  # 定义一个函数来读取CSV文件  # 参数：file\_name - 要读取的CSV文件的名称  # 返回：一个DataFrame，包含CSV文件的内容  def read\_csv(file\_name):  # error\_bad\_lines=False，这个参数意味着在读取时忽略错误行  return pd.read\_csv(file\_name, error\_bad\_lines=False)  # 定义主函数  def main():  # 评论文件列表  comment\_files = ['firstSeason.csv', 'secondSeason.csv', 'thirdSeason.csv']    # 初始化季度评论数量列表  quarterly\_comments = [0, 0, 0]    # 计算每个季度的评论数量  for comment\_file in comment\_files:  # 读取评论数据  data = read\_csv(comment\_file)  # 遍历数据的每一行  for index, row in data.iterrows():  # 将日期字符串转换为datetime对象  date = pd.to\_datetime(row['Date'])  # 判断月份，根据月份增加对应季度的评论数量  if date.month <= 3:  quarterly\_comments[0] += 1  elif date.month <= 6:  quarterly\_comments[1] += 1  else:  quarterly\_comments[2] += 1    # 定义季度标签  quarters = ['第一季度(1-3月)', '第二季度(4-7)', '第三季度(7-12)']    # 导入中文字体，这是为了在图表中正确显示中文  font = FontProperties(fname='C:\\Windows\\Fonts\\msyh.ttc')    # 创建一个新的图表窗口  plt.figure(figsize=(8, 8))  # 绘制饼图，显示每个季度的评论数量占比  plt.pie(quarterly\_comments, labels=quarters, autopct='%1.1f%%',  colors=['#4CAF50', '#FF5722', '#2196F3'], startangle=90, textprops={'fontproperties': font})  # 添加图表标题  plt.title('每季度动漫评论数占比', fontproperties=font)    # 设置图表的坐标轴比例，使其保持相等，这样饼图将是一个完整的圆  plt.axis('equal')  # 显示图表  plt.show()  # 当脚本作为主程序运行时，执行主函数  if \_\_name\_\_ == "\_\_main\_\_":  main() |

图表, 饼图

描述已自动生成

上述饼图以直观的方式展示了各个季度的动漫评论数占比。我们可以明显看到，在第一季度，评论数占总数的28.5%，这一比例在第二季度略有下降，为20.5%。然而，令人惊讶的是，第三季度的评论数占比达到了51.0%，超过了前两个季度的评论数之和。这一数据显示，第三季度对于动漫作品的观众参与度和兴趣具有显著的重要性。这种显著的增长可能受到几个因素的影响。首先，第三季度包含了暑假期间，很多学生和上班族会有更多的空余时间来观看动漫，从而导致评论数的增加。其次，动漫制作公司可能会选择在这个时期发布一些备受期待的新作或关键剧情，以吸引更多的观众。此外，一些特定的文化活动或者动漫展也可能会在第三季度举行，这些活动可能会激发观众的兴趣，进而增加对动漫作品的讨论和评论。

# 五、 总结与思考

在本项目中，我学习了如何使用Hadoop和Spark处理大规模数据，以及如何使用Python进行数据可视化。通过整合这些工具，能够从大量数据中提取有价值的信息。我对“MyAnimeList - Anime Dataset with Reviews”数据集进行了深入的探讨和分析。使用Apache Spark，能够有效地处理和分析大规模的动漫评论数据。本次分析的目标是了解用户对不同动漫的情感倾向，探索动漫的流行趋势，以及分析观众的偏好。值得一提的是，某些动漫在特定的月份和季度中获得了显著的评论数。例如，《赛博朋克边缘行者》在11月份的评论数远超其他月份，而《间谍过家家》在4月、6月和12月成为了评论最多的动漫。通过进一步的调查和分析，我们发现评论数量的增加往往与动漫的播出时间、剧情的吸引力，以及市场营销策略等因素有关。此外，还发现第三季度的评论数量占总评论的51.0%，超过了第一季度和第二季度的总和。这可能是由于第三季度包含暑假，人们有更多的空余时间观看动漫，以及制作公司倾向于在这个时期发布新作品。

这些洞察对动漫制作公司和市场营销人员非常有价值，因为它们可以根据这些数据制定更有效的内容策略和市场推广计划。总的来说，通过对大规模的动漫评论数据进行分析，我们能够洞悉动漫作品的流行趋势和观众偏好，为动漫产业提供宝贵的策略建议和市场洞察。对于未来的项目，可以考虑使用更先进的机器学习算法，以提供更深入的洞察。

# 参考文献

1. "Apache Spark", [https://spark.apache.org/](<https://spark.apache.org/>)
2. "Hadoop", [https://hadoop.apache.org/](<https://hadoop.apache.org/>)

3."Kaggle-MyAnimeList数据集", [https://www.kaggle.com](<https://www.kaggle.com>)

4."Matplotlib:Visualization with Python",[https://matplotlib.org/](<https://matplotlib.or/>)

5."Seaborn:StatisticalDataVisualization",[https://seaborn.pydata.org/](<https://seaborn.pydata.org/>)

6.“WordCloudforPythondocumentation”,[https://amueller.github.io/word\_cloud/](<https://amueller.github.io/word_cloud/>)

7. McKinney, W. (2012). Python for Data Analysis. O'Reilly Media, Inc.