**《自然语言处理导论》实验报告**

学院（系）：信息科学技术学院 专业：智能科学与技术 班级：2022-2

姓 名： 孙伟豪 学号： 2220223194 实验项目：情感词典实验

目录

[1 实验目的和要求 2](#_Toc184413206)

[2 实验原理和内容 2](#_Toc184413207)

[3 实验设备 2](#_Toc184413208)

[4 实验核心代码 2](#_Toc184413209)

[5 实验结果 6](#_Toc184413210)

[6 实验结果讨论分析 7](#_Toc184413211)

# 实验目的和要求

本实验旨在通过情感词典（使用SentiWordNet）进行文本情感分析，验证情感词典在自然语言处理中的有效性和局限性。旨在引导学生学会使用nltk工具中情感分析的预处理模型和用法，对单词，短语，段落进行有效的分类和处理。

学会情感分类是自然语言处理中必不可少的一环，可以培养同学们对于单词预处理，文本分类等任务的动手能力。

通过情感词典对文本进行情感分析，并评估不同情感词典（如 SentiWordNet、情感词典、百度情感词典等）在情感分析任务中的表现。主要比较以下方面：情感词典对情感分类（正面/负面）的准确性影响。不同情感词典对模型性能的影响。对情感强度的衡量和应用。

# 实验原理和内容

通过实验探索情感词典如何影响情感分析结果，以及它在不同情感表达上的准确性。通过对文本中的每个词汇进行情感得分的计算，理解如何从单一词汇的情感评分到整个文本的情感分类。目标是了解情感分析中的情感词汇对文本整体情感的贡献，以及如何通过聚合词汇得分来获得整体情感评价。探讨否定词（如“not”）对情感分析结果的影响，验证情感词典在面对否定词时是否能正确反转情感得分，以及如何改进情感分析算法以提高准确性。通过比较情感分析的结果与人工标签数据集的标注，评估基于情感词典的情感分类模型的准确性，验证词典方法的有效性和局限性。

在vader中文本预处理：对文本进行标准化处理，包括分词、去停用词、词干化等。情感词典匹配：查找文本中的每个词是否在情感词典中，若存在则根据词典中的情感值进行加权计算。情感值计算：对文本中所有情感词的情感值进行加总（可以考虑加权平均），最终得出该文本的情感倾向。情感分类：根据情感值的正负判断文本的情感倾向。如果情感值大于0，则为正面情感；小于0则为负面情感。

# 实验设备

在本地的3060laptop进行实验，实际上，没有使用机器学习和深度学习模型进行计算的话，无相关device赋值代码，使用的是本地的CPU（i7 12700h）。

# 实验核心代码

* SentiWordNet

import nltk

from nltk.corpus import sentiwordnet as swn

from nltk.tokenize import word\_tokenize

import pandas as pd

# 函数：获取单词的情感分数

def get\_sentiment\_score(word):

# 使用SentiWordNet查找单词的情感得分

senti\_synsets = list(swn.senti\_synsets(word))

if senti\_synsets:

# 计算所有同义词集的平均情感得分

total\_pos\_score = 0

total\_neg\_score = 0

for senti\_synset in senti\_synsets:

total\_pos\_score += senti\_synset.pos\_score()

total\_neg\_score += senti\_synset.neg\_score()

# 返回正面情感分数 - 负面情感分数的平均值

avg\_pos\_score = total\_pos\_score / len(senti\_synsets)

avg\_neg\_score = total\_neg\_score / len(senti\_synsets)

return avg\_pos\_score - avg\_neg\_score

else:

return 0 # 如果没有找到同义词集，返回0

# 示例：计算单词情感得分

word = 'good'

word1 = 'wrong'

print(f"Sentiment score for '{word}': {get\_sentiment\_score(word)}")

print(f"Sentiment score for '{word1}': {get\_sentiment\_score(word1)}")

# 函数：计算文本情感得分

def analyze\_sentiment(text):

words = word\_tokenize(text) # 将文本分词

sentiment\_score = 0

for word in words:

word\_score = get\_sentiment\_score(word.lower()) # 计算单词情感得分（转换为小写以匹配词典）

sentiment\_score += word\_score # 累加情感得分

return sentiment\_score

# 示例：分析文本情感

text = "The movie was really good, I enjoyed it a lot!"

print(f"Sentiment score for the text: {analyze\_sentiment(text)}")

# 函数：情感分类

def classify\_sentiment(text):

score = analyze\_sentiment(text)

if score > 0:

return "Positive"

elif score < 0:

return "Negative"

else:

return "Neutral"

# 示例：情感分类

print(f"Sentiment classification for the text: {classify\_sentiment(text)}")

# 假设我们有一个IMDb数据集（假设为csv格式）

data = pd.DataFrame({

'text': ["I love this movie!", "This is a terrible movie.", "It was okay,although a little ordinary."],

'label': ['Positive', 'Negative', 'Positive']

})

# 准确性验证

correct = 0

for \_, row in data.iterrows():

predicted = classify\_sentiment(row['text'])

actual = row['label']

if predicted == actual:

correct += 1

accuracy = correct / len(data)

print(f"Accuracy: {accuracy \* 100:.2f}%")

* VADER

import pandas as pd

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

from textblob import TextBlob

from sklearn.metrics import accuracy\_score, classification\_report

# 示例数据集（评论，标签：1正面，0负面）

data = pd.DataFrame({

'text': ['I love this movie!', 'I hate this movie!', 'This is an amazing book.', 'This book is terrible.'],

'label': [1, 0, 1, 0]

})

X = data['text']

y = data['label']

# VADER情感分析函数

def vader\_sentiment(text):

analyzer = SentimentIntensityAnalyzer()

score = analyzer.polarity\_scores(text)

return score['compound'] # compound是综合得分，范围[-1,1]，大于0为正面，小于0为负面

# TextBlob情感分析函数

def textblob\_sentiment(text):

blob = TextBlob(text)

return blob.sentiment.polarity # Polarity在[-1,1]之间，大于0为正面，小于0为负面

# 自定义情感词典（示例）

sentiment\_dict = {

'love': 0.8,

'amazing': 0.7,

'terrible': -0.7,

'hate': -0.8

}

# 使用自定义情感词典进行情感分析

def custom\_dict\_sentiment(text):

words = text.lower().split()

sentiment\_score = 0

count = 0

for word in words:

if word in sentiment\_dict:

sentiment\_score += sentiment\_dict[word]

count += 1

if count == 0:

return 0 # 如果没有匹配到情感词，则认为情感中性

return sentiment\_score / count # 返回情感词的平均值

# 应用情感分析方法

X\_vader = X.apply(vader\_sentiment)

X\_textblob = X.apply(textblob\_sentiment)

X\_custom = X.apply(custom\_dict\_sentiment)

# 根据情感分析得分进行情感分类（大于0为正面，否则为负面）

y\_pred\_vader = (X\_vader > 0).astype(int)

y\_pred\_textblob = (X\_textblob > 0).astype(int)

y\_pred\_custom = (X\_custom > 0).astype(int)

# 评估模型性能

print("VADER Sentiment Analysis")

print("Accuracy:", accuracy\_score(y, y\_pred\_vader))

print(classification\_report(y, y\_pred\_vader))

print("TextBlob Sentiment Analysis")

print("Accuracy:", accuracy\_score(y, y\_pred\_textblob))

print(classification\_report(y, y\_pred\_textblob))

print("Custom Dictionary Sentiment Analysis")

print("Accuracy:", accuracy\_score(y, y\_pred\_custom))

print(classification\_report(y, y\_pred\_custom))

# 实验结果

由于本实验程序的输出较为简单，通常是分数，lable和提示语，故将其集成到一整个程序中直接运行输出。

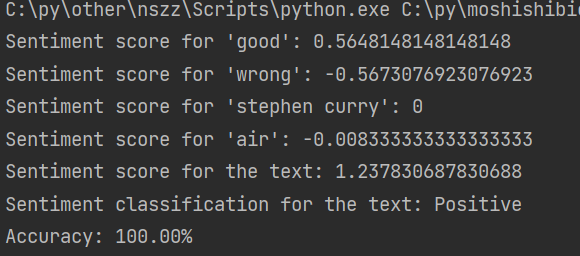
以下输出依次是：

Good,wrong,stephrn curry,air单词的情感得分。

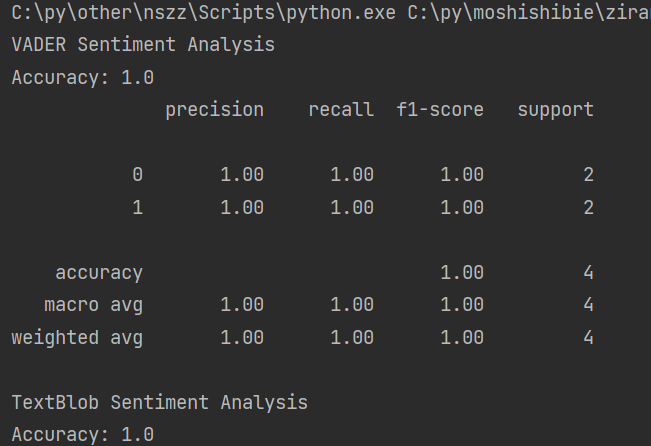
The movie was really good, I enjoyed it a lot!的情感得分。

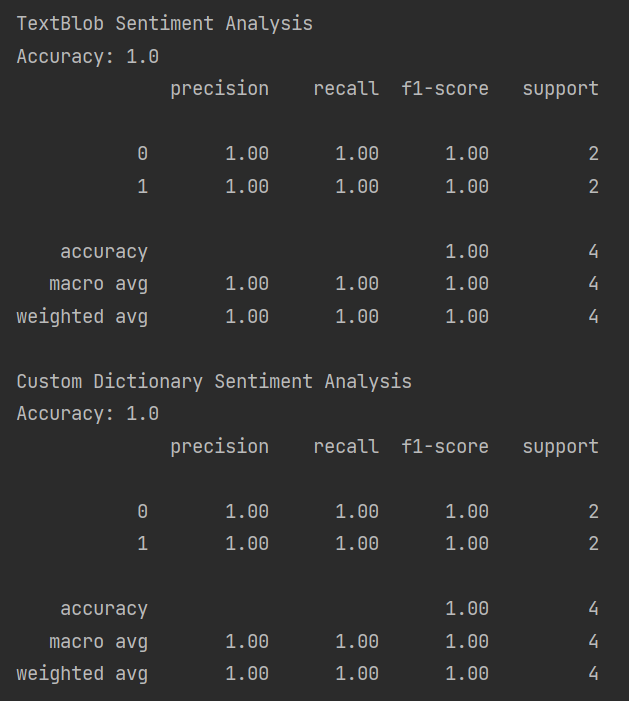
The movie was really good, I enjoyed it a lot!的情感分类。

"I love this movie!", "This is a terrible movie.", "It was okay,although a little ordinary."的情感标签预测结果（全对）。



使用vador的验证效果：





# 实验结果讨论分析

基于SentiWordNet情感词典，使我大致了解了情感分析的一些基本用法，原理和使用技巧。但本实验未涉及更多情感词典或预训练语言模型的对比分析，且文本数据集较小，未来可以通过多种情感词典或先进模型的结合来进一步提升情感分析的准确性。情感词典在情感分析中的应用为我们提供了一个简便的分析工具，但由于其局限性，不能处理所有复杂的情感表达。实验表明，尽管基于词典的情感分析方法在某些情境下有效，但对于包含否定词、复杂语法结构和多义词的文本，情感词典的准确性存在明显问题。通过引入上下文感知的方法和优化情感词典的处理方式，可以提高情感分析的效果。

VADER情感分析通常能提供较高的准确性，尤其是对于社交媒体文本或较短的评论，它在处理情感强度时表现较好。