In [ ]:
```python
import jieba
import jieba.analyse

# 进行文本预处理
jieba.suggest_freq('牧羊少年', True) #讲牧羊少年看作一个特定名词
# 载入中文常用停用词表（去除一些无意义的词语）采用百度停用词表
jieba.analyse.set_stop_words('stop_words.txt')

# 载入文本
text = open('牧羊少年奇幻之旅.txt', 'r', encoding='utf-8').read()
```

In [ ]:
```python
# 采用TF-IDF提取20个关键词
keywords_tfidf = jieba.analyse.extract_tags(text)
print("TF_IDF", keywords_tfidf)
```

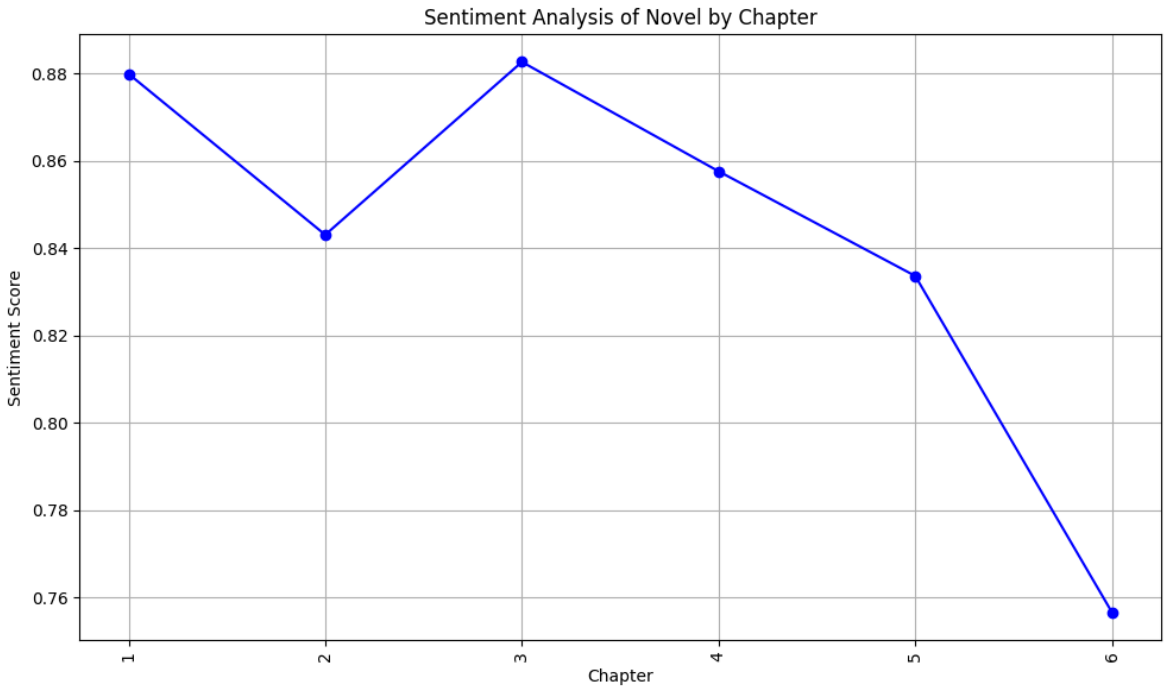
```
TF_IDF ['男孩', '沙漠', '炼金术士', '金术士', '宝藏', '牧羊人', '英国人', '水晶', '老人', '绿洲', '告诉', '预兆', '天命', '商人', '商队', '金字塔', '骆驼', '羊群', '帐篷', '法谛']
```

In [ ]:
```python
# 采用textRank方法提取20个关键词
keywords_textrank = jieba.analyse.textrank(text)
print('Textrank', keywords_textrank)
```

```
Textrank ['男孩', '沙漠', '水晶', '老人', '炼金术士', '商人', '告诉', '绿洲', '金术士', '地方', '帐篷', '东西', '商队', '骆驼', '世界', '发现', '羊群', '离开', '语言', '战争']
```

In [ ]:
```python
import re
import matplotlib.pyplot as plt
from snownlp import SnowNLP

# 分割章节
chapters = re.split(r'(《牧羊少年奇幻之旅》\s*第.*章)', text)
chapter_titles = [chapters[i] for i in range(1, len(chapters), 2)]
chapter_contents = [chapters[i] for i in range(2, len(chapters), 2)]

# 存储每个章节的情感得分
chapter_sentiments = []

for content in chapter_contents:
    paragraphs = content.split('\n\n')  # 将章节按段落分割
    sentiment_scores = []
    for paragraph in paragraphs:
        if len(paragraph) != 0:
            sentiment_scores.append(SnowNLP(paragraph).sentiments)
    average_sentiment = sum(sentiment_scores) / (len(sentiment_scores))
    chapter_sentiments.append(average_sentiment)

# 可视化每个章节的情感变化
chapter_indices = list(range(1, len(chapter_contents) + 1))

plt.figure(figsize=(10, 6))
plt.plot(chapter_indices, chapter_sentiments, marker='o', linestyle='-', color='
plt.xlabel('Chapter')
plt.ylabel('Sentiment Score')
plt.title('Sentiment Analysis of Novel by Chapter')
plt.xticks(ticks=chapter_indices, labels=list(range(1,len(chapter_titles)+1)), r
plt.grid(True)
plt.tight_layout()
```

```
plt.show()

# 打印每个章节的情感得分
for title, score in zip(chapter_titles, chapter_sentiments):
    print(f"{title}: {score}")

# 0到0.5之间：表示负面情感。得分越接近0，负面情感越强
# 0.5：表示中性情感。
# 0.5到1之间：表示正面情感。得分越接近1，正面情感越强。
```



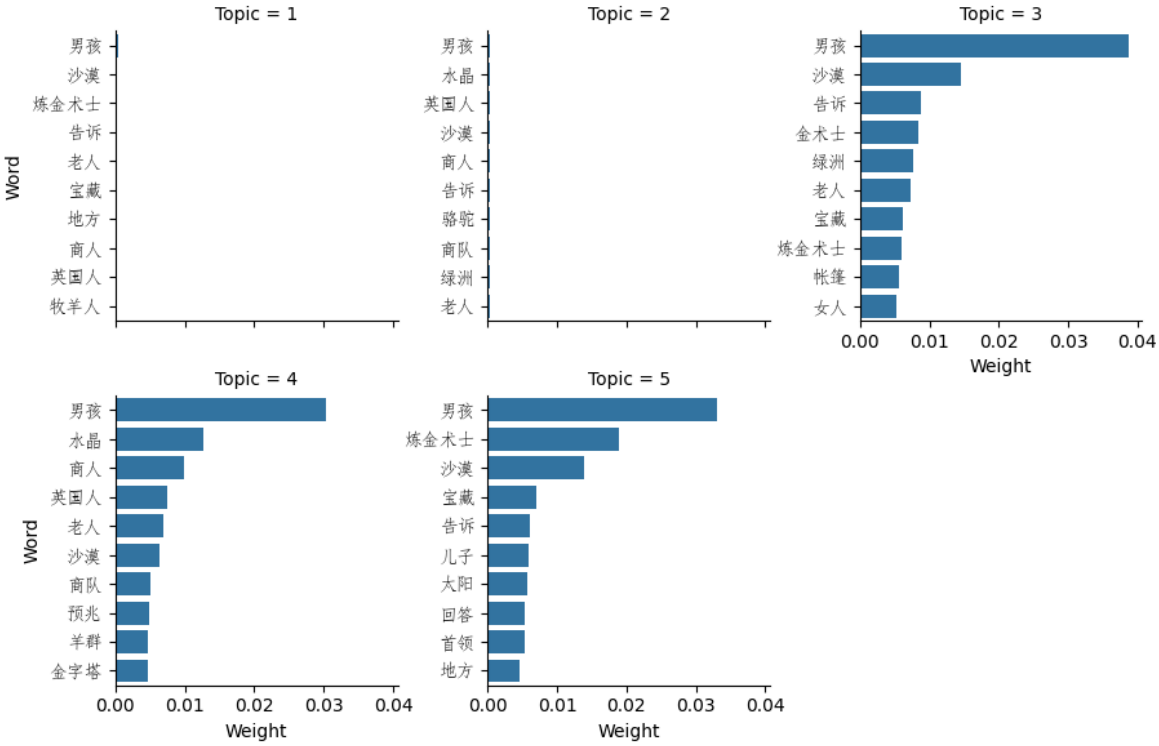
《牧羊少年奇幻之旅》 第一章: 0.8799484991981207
《牧羊少年奇幻之旅》 第二章: 0.8430918357129025
《牧羊少年奇幻之旅》 第三章: 0.8826775061760188
《牧羊少年奇幻之旅》 第四章: 0.8576318453440145
《牧羊少年奇幻之旅》 第五章: 0.8336621447526056
《牧羊少年奇幻之旅》 第六章: 0.7566465574735924

In [ ]:

```
# 主题分析
from gensim import corpora, models
import pyLDAvis.gensim_models as gensimvis
import matplotlib.pyplot as plt
# 加载中文字体
from matplotlib import font_manager
import seaborn as sns
import pandas as pd
import warnings
# 指定字体路径
font_path = 'C:/Windows/Fonts/simfang.ttf'  # 请替换为你系统中中文字体的路径
font_prop = font_manager.FontProperties(fname=font_path)
# 构建词袋模型

# 加载停用词表
stopwords = set()
with open('stop_words.txt', 'r', encoding='utf-8') as f:
    for line in f:
        stopwords.add(line.strip())

# 切词，并且在过程中过滤掉停用词
def preprocess(text):
```

```
    words = jieba.lcut(text)
    words = [word for word in words if word not in stopwords and len(word) > 1]
    return words

# 处理所有章节文本
processed_texts = [preprocess(content) for content in chapter_contents]

dictionary = corpora.Dictionary(processed_texts)
corpus = [dictionary.doc2bow(t) for t in processed_texts]

# 训练LDA模型
lda_model = models.LdaModel(corpus, num_topics=5, id2word=dictionary, passes=15)

# 打印每个主题的关键词
for i, topic in lda_model.print_topics(num_topics=5, num_words=10):
    print(f'Topic {i}: {topic}')

# 可视化主题
def plot_keywords(lda, nb_topics, nb_words, dictionary):
    cols = ['Topic', 'Word', 'Weight']
    df = pd.DataFrame(columns=cols)

    for topic_idx, topic in enumerate(lda.show_topics(num_topics=nb_topics, num_
        for word, weight in topic[1]:
            new_row = pd.Series([int(topic_idx + 1), word, weight], index=cols)
            df = pd.concat([df, new_row.to_frame().T], ignore_index=True)
     # 抑制警告信息
    warnings.filterwarnings("ignore", category=UserWarning)

    g = sns.FacetGrid(df, col='Topic', sharey=False, col_wrap=3)
    g.map_dataframe(sns.barplot, x='Weight', y='Word')
    for ax in g.axes.flat:
        for label in ax.get_yticklabels():
            label.set_fontproperties(font_prop)
    plt.show()

plot_keywords(lda_model, 5, 10, dictionary)
```

```
Topic 0: 0.000*"男孩" + 0.000*"沙漠" + 0.000*"炼金术士" + 0.000*"告诉" + 0.000*"老人" + 0.000*"宝藏" + 0.000*"地方" + 0.000*"商人" + 0.000*"英国人" + 0.000*"牧羊人"
Topic 1: 0.000*"男孩" + 0.000*"水晶" + 0.000*"英国人" + 0.000*"沙漠" + 0.000*"商人" + 0.000*"告诉" + 0.000*"骆驼" + 0.000*"商队" + 0.000*"绿洲" + 0.000*"老人"
Topic 2: 0.039*"男孩" + 0.015*"沙漠" + 0.009*"告诉" + 0.008*"金术士" + 0.008*"绿洲" + 0.007*"老人" + 0.006*"宝藏" + 0.006*"炼金术士" + 0.006*"帐篷" + 0.005*"女人"
Topic 3: 0.030*"男孩" + 0.013*"水晶" + 0.010*"商人" + 0.007*"英国人" + 0.007*"老人" + 0.006*"沙漠" + 0.005*"商队" + 0.005*"预兆" + 0.005*"羊群" + 0.005*"金字塔"
Topic 4: 0.033*"男孩" + 0.019*"炼金术士" + 0.014*"沙漠" + 0.007*"宝藏" + 0.006*"告诉" + 0.006*"儿子" + 0.006*"太阳" + 0.005*"回答" + 0.005*"首领" + 0.005*"地方"
```


Topic = 1
男孩
沙漠
炼金术士
告诉
老人
宝藏
地方
商人
英国人
牧羊人
Word
Topic = 2
男孩
水晶
英国人
沙漠
商人
告诉
骆驼
商队
绿洲
老人
Topic = 3
男孩
沙漠
告诉
金术士
绿洲
老人
宝藏
炼金术士
帐篷
女人
0.00
0.01
0.02
0.03
0.04
Weight
Topic = 4
男孩
水晶
商人
英国人
老人
沙漠
商队
预兆
羊群
金字塔
Word
0.00
0.01
0.02
0.03
0.04
Weight
Topic = 5
男孩
炼金术士
沙漠
宝藏
告诉
儿子
太阳
回答
首领
地方
0.00
0.01
0.02
0.03
0.04
Weight