先介绍一下模型的含义吧：

SARIMAX 是一种时间序列预测模型，它是 SARIMA 模型的扩展。SARIMA 模型是一种用于预测时间序列数据的统计模型，它考虑了数据中的季节性和趋势性

SARIMAX 模型在 SARIMA 模型的基础上增加了外生变量（exogenous variable）的考虑。外生变量是指与时间序列数据相关但不受其影响的变量。通过引入外生变量，SARIMAX 模型可以更好地捕捉到时间序列数据中的复杂关系

SARIMAX 模型的应用范围广泛，适用于各种需要考虑季节性和趋势性的时间序列预测问题。例如，它可以用于经济学、金融学、气象学、销售预测等领域

基于这个基础我们采用了这个模型来进行时间预测

这里我们首先通过获取到的数据，对整的一个数据集进行划分，划分为个个学科

```
    #读取文件
    df = pd.read_excel('处理结果.xlsx')
    #判断关键词的数量
    new_df = df['关键词en'].value_counts()
    #生成关键词种类
    x_data = list(new_df.index)
    for x in tqdm(x_data):
        demo(x)
```

这个就是对整的文件进行学科划分

接着我们根据发布时间对该列进行划分，按月来划分

```
data_month = data.resample("M").sum()
```

可以得知每一个月发帖的总数多少，然后对该列表进行时间预测

列表的格式：

发表日期	博文id	回复数量	转发数量	...	粉丝数量	关注数量	发帖数量
				...			
2019-02-28	1095012056091090944	1163	332	...	47397	1485	1
2019-03-31	0	0	0	...	0	0	0
2019-04-30	0	0	0	...	0	0	0
2019-05-31	0	0	0	...	0	0	0
2019-06-30	0	0	0	...	0	0	0
2019-07-31	0	0	0	...	0	0	0
2019-08-31	0	0	0	...	0	0	0
2019-09-30	0	0	0	...	0	0	0
2019-10-31	0	0	0	...	0	0	0
2019-11-30	0	0	0	...	0	0	0
2019-12-31	0	0	0	...	0	0	0

如果数据量很少的就变成这样，因此为了排除这些不符合规则的列表，我们采用了一阶差分

一阶差分是指离散函数中连续相邻两项之差。当自变量从 x 变到 x+1 时，函数 y=y(x) 的改变量 Δyx=y(x+1)-y(x)，(x=0，1，2，...) 称为函数 y(x) 在点 x 的一阶差分1。

一阶差分的主要用途是消除时间序列数据中的线性趋势因素，得到平稳序列2。如果一阶差分后数据平稳，称之为一阶单整

实现代码如下：

```
#对数据进行一阶差分处理
    data1 = data_month['发帖数量'].diff().dropna()
    data1 = data1.diff().dropna()
```

接着，我们采用auto_arima函数，去自动筛选最优的参数值

```
best_fit = auto_arima(data1, trace=True, error_action='ignore',
suppress_warnings=True, stepwise=True)
```

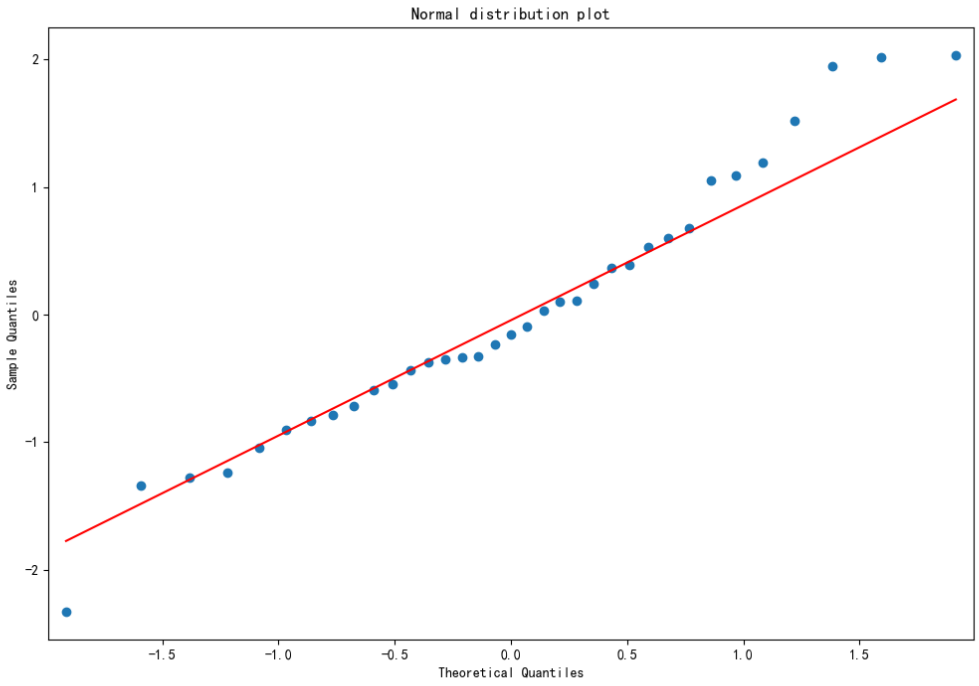
通过找出最优的参数值，我们去构建相关模型，再根据模型进行拟合

```
#根据前面找到的最优数值来输出最优模型
model = SARIMAX(data1, order=best_fit.order,
seasonal_order=best_fit.seasonal_order)
#对模型进行拟合
model_fit = model.fit(disp=False)
```

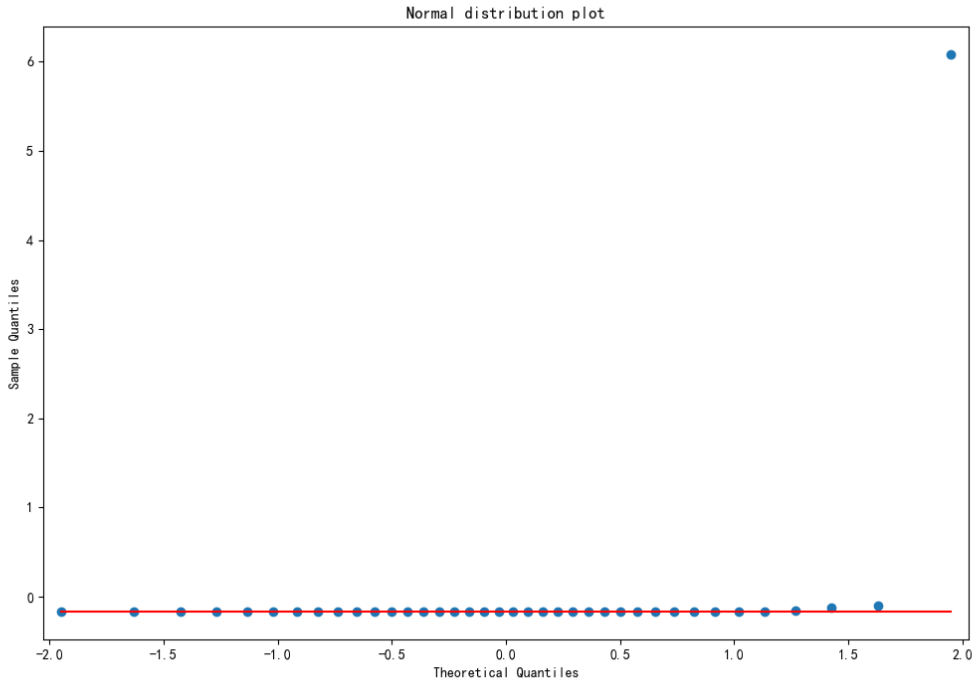
根据找到的最优模型，我们进行残差处理

将残差标准化为正态分布，有助于检查模型是否正确地捕获了数据中的信息。这是一种常见的做法，可以帮助我们更好地理解模型的性能

一般如果好的正太分布图，应该如下：



呈一条斜线，并且斜线上面的点都是围绕着斜线进行分布的状态



像这种的画，就是不太好的表现

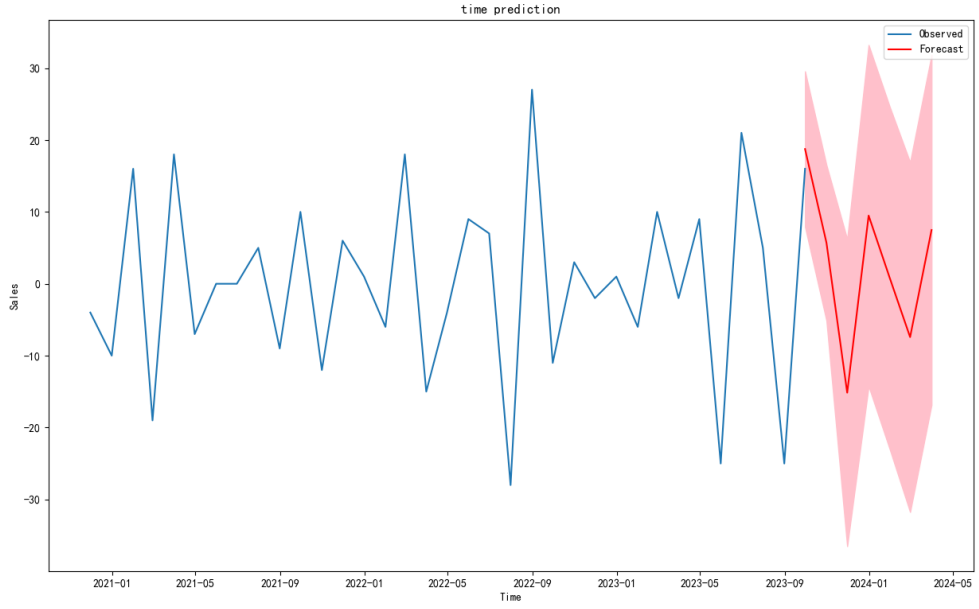
接着我们再来查看拟合效果，代码如下：

```
# 查看拟合效果
delta = model_fit.fittedvalues - data1
# 它的值在0-1之间，越接近1，拟合效果越好
score = 1 - delta.var() / data1.var()
with open('./{}/拟合得分.txt'.format(name),'w',encoding='utf-8-sig')as f:
    f.write("拟合得分为：{}".format(score))
```

接着我们进行预测

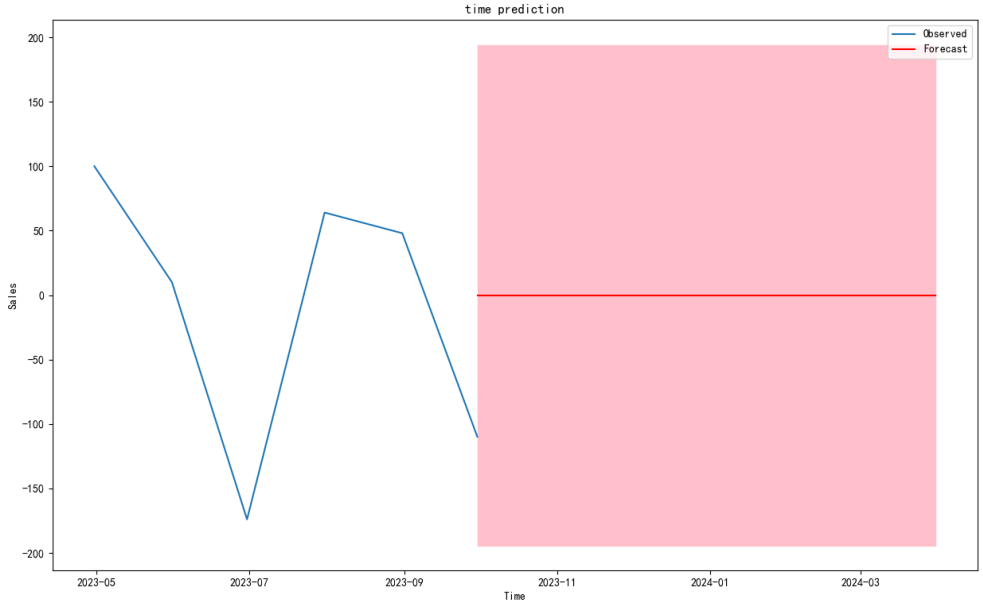
```
# 预测后面时间发布频率的趋势，预测接下来五个月的发帖趋势
prediction = model_fit.get_prediction(start=list(data1.index)[-1],
end=len(data1)+5, dynamic=False, full_results=True)
predicted_mean = prediction.predicted_mean
predicted_confidence_intervals = prediction.conf_int()
#画图
plt.figure(figsize=(15,9))
#原先的数值图
plt.plot(data1, label='Observed')
#预测的数值图
plt.plot(predicted_mean.index, predicted_mean, color='r', label='Forecast')
#把两者结合在一起，生成预测效果图
plt.fill_between(predicted_confidence_intervals.index,
predicted_confidence_intervals.iloc[:, 0], predicted_confidence_intervals.iloc[:,
1], color='pink')
plt.xlabel('Time')
plt.ylabel('Sales')
plt.legend()
plt.title('time prediction')
plt.savefig('./{}/time prediction.png'.format(name))
```

```
plt.show()
```



像这种有明显的趋势变化的话，这种就说明预测的效果不错，一般效果最好的，就是蓝线和红线完全连接在一起，这种的效果就是预测最好的

如果像这种



完全变成一条直线的话，就说明预测的效果并不好，这个预测的好与坏，和数据有关，如果数据太少，就容易影响预测的内容，如果文件夹是空的，说明那一整个数据表格都不能用，所以文件夹为空值

以上便是整体的分析内容