NLP实验六

实验报告

姓名:王雨静

班级 ： 1期NLP训练营

日期:2021年4月24日

目录

一、疫情相关谣言

1.0 数据读取

数据格式:

- date: 时间
- explain: 谣言类型
- tag: 谣言标签
- abstract: 用以验证谣言的内容
- title: 标题
- rumor: 谣言

结合使用json.load和pandas.DataFrame，将数据集读取到一个DataFrame中。

谣言类型包括'伪常识'，'伪科学'，'尚无定论'和'确实如此'，分别有1, 61, 43和19条。
其中‘伪常识’和‘伪科学’判定为谣言，设标签1；‘尚无定论’判定为不确定，设标签0；‘确实如此’判定为不是谣言，设标签-1。

1.1 判断是否为谣言

1.1.1 特征提取

调用预训练roberta模型'hfl/chinese-roberta-wwm-ext-large' ,对tag、abstract、title和rumor分别进行特征提取，分别得到1024维的向量。

部分代码：

```
max_length = 512
abstracts = []

for i in range(0, 124):
  text = df['abstract'][i][:max_length]
  encoded_input = tokenizer(text, return_tensors='pt').to('cuda')
  output = model(**encoded_input)
  feature = output[0][:,0,:].cpu().detach().numpy() #取出需要的特征，1024维向量
  abstracts.append(feature)
```

1.1.2 分类

2:1划分训练集和验证集。

分别使用每一个特征，使用多层感知机进行分类。

部分代码：

```
X_train, X_test, y_train, y_test = train_test_split(title_array, y,
test_size=0.33, random_state=2021)
clf = MLPClassifier(random_state=1, max_iter=20, verbose = True,
                    early_stopping = True)
clf.fit(X_train, y_train)
print("train score:", clf.score(X_train, y_train))
print("test score:", clf.score(X_test, y_test))
```

Output:

```
train score: 0.6746987951807228
test score: 0.6585365853658537
```

1.1.3 实验结果

第一次实验结果：

使用的特征	train_accuracy	dev_accuracy
tag	0.614	0.390
abstract	0.737	0.60
title	0.675	0.659
rumor	0.687	0.634
Abstract + title + rumor	0.831	0.682

(由于tag的表现较差，没有一起放入全部特征中）

不意外得，综合考量Abstract、title和rumor取得的效果更好。

进行五折交叉验证：

部分代码：

```
from sklearn.model_selection import cross_val_score
clf =  MLPClassifier(random_state=1, max_iter=20, verbose = False,
                     early_stopping = True)
scores = cross_val_score(clf, all_array, y, cv=5)
scores.mean()
```

使用的特征	dev_accuracy（mean）
abstract	0.604
title	0.525
rumor	0.531
Abstract + title + rumor	0.637

使用abstract的信息对于预测是否是谣言的帮助最大。这可能是因为abstract中包含了更多的有用的语义信息，对于做出最后的推断有建设性的作用。

但是加入title和rumor的信息对最终的效果也有帮助。

1.2 谣言tag分析

对确定是谣言的数据的tag进行统计。

代码：

```
rumor_tag_dict = {}
tag_set = set()
for i in range(0, 124):
  if df.loc[i, 'label_'] == 1:
    tags = df.loc[i, 'tag'].split(', ')
    for tag in tags:
      tag_set.add(tag)
      if tag in rumor_tag_dict.keys():
        rumor_tag_dict[tag] += 1
      else:
        rumor_tag_dict[tag] = 1
rumor_tag_dict
```

并绘制词云：

代码：

```
from wordcloud import WordCloud, ImageColorGenerator
from imageio import imread
back_coloring = imread("/content/covid2.png")

font_path = '/content/字体圈欣意吉祥宋.ttf'

wc = WordCloud(font_path=font_path, background_color="white",
max_words=2000, mask=back_coloring,
               max_font_size=100, random_state=42, width=1000,
height=860, margin=2,)

wc.generate_from_frequencies(rumor_tag_dict)
image_colors_byImg = ImageColorGenerator(back_coloring)

plt.figure(dpi = 200)
plt.imshow(wc.recolor(color_func=image_colors_byImg),
interpolation="bilinear")

plt.imshow(wc, interpolation="bilinear")
# plt.imshow(covid_mask, interpolation="bilinear")
plt.axis("off")
plt.show()
```

Fig1. 谣言Tag词云

其中'口罩'出现了12词，'传染病'出现了3词， 含有‘消毒’的tag出现了4词。

大部分标签都和疫情防护相关，关于预防和消毒的方法和防疫物资如口罩、防护服等。

疫情期间，口罩成为了大家的日常用品，而关于怎么使用口罩、鉴别口罩、自制口罩等方面，**十分容易出**现谣言，例如用尿不湿代替口罩等。

而关于怎么消毒的谣言也较多，例如用‘白酒’、放‘烟花爆竹’、用‘吹风机’、‘盐水漱口’**等**这些都是常见的谣言。

1.3 谣言词云

对abstract、title、rumor分别绘制词云：

Fig2. 谣言abstract词云

Fig3. 谣言title词云

Fig4. 谣言rumor词云

从图中可以发现，在标题中，新型冠状病毒出现的频率最高，而在abstract和rumor中口罩和病毒两个词占了最大的比重。由此可见，关于口罩的谣言在要严重占了主导地位，

二、疫情相关中文新闻

2.0 数据读取

数据格式:

新闻以天为单位存储在json格式的文档中。

文档名是日期，文档内容是每一条新闻的具体信息，例如：

{'meta': {'content': '**大宗商品将**创下自2016年以来的最佳年度表现，原油到铜的**年度收**…… SF107',
'description': '**担**忧情绪消散 商品有望迎来2016年来表现最好的一年',
'keyword': [],
'title': '**担**忧情绪消散 商品有望迎来2016年来表现最好的一年',
'type': 'news'},
'time': '01-01 00:00',
'title': '**担**忧情绪消散 商品有望迎来2016年来表现最好的一年',
'url':
'https://finance.sina.com.cn/stock/usstock/c/2020-01-01/doc-iihnzahk1229877.shtml'}

接下来的实验将针对‘meta’下的‘title’，即新闻的标题展开。

2.1 标题词云视频

对每一天的标题去掉stopword后，使用jieba工具进行分词，统计出当日出现频率最高的20个词。

以三天为单位绘制词云。（更新频率词典：加入新的一天的高频词，去掉三天前的高频词）

视频每秒25帧，每10帧为1天。

Fig5. 01-01新闻标题词云

图中可以发现，在新年初始，大家更多关心的是辞旧迎新。

视频链接：

https://drive.google.com/file/d/1Sv6Rg-Iz47-me1SDFEjiHcpmCWEhxVCz/view?usp=sharing

从词云的变化中，可以观察到国人关心的话题。

2.2 标题情感分布

对每一天的标题调用cnsenti进行情感提取。

'好', '乐', '哀', '怒', '惧', '恶', '惊'七种具体的情绪，以及'pos'和'neg两种倾向分别进行情感词的统计和计算。

（cnsenti主要是字级别的，其中pos和neg考虑到副词和否定词等）

之后对每一天的七种情绪和正负面情绪分别进行归一化（除以当天的总和）,并绘制柱状图。

制成视频，视频每秒25帧，每10帧为1天。

视频链接：

https://drive.google.com/file/d/1SvrLAdidsBkJJ_JTLKi30x_p3SsKoqdC/view?usp=sharing

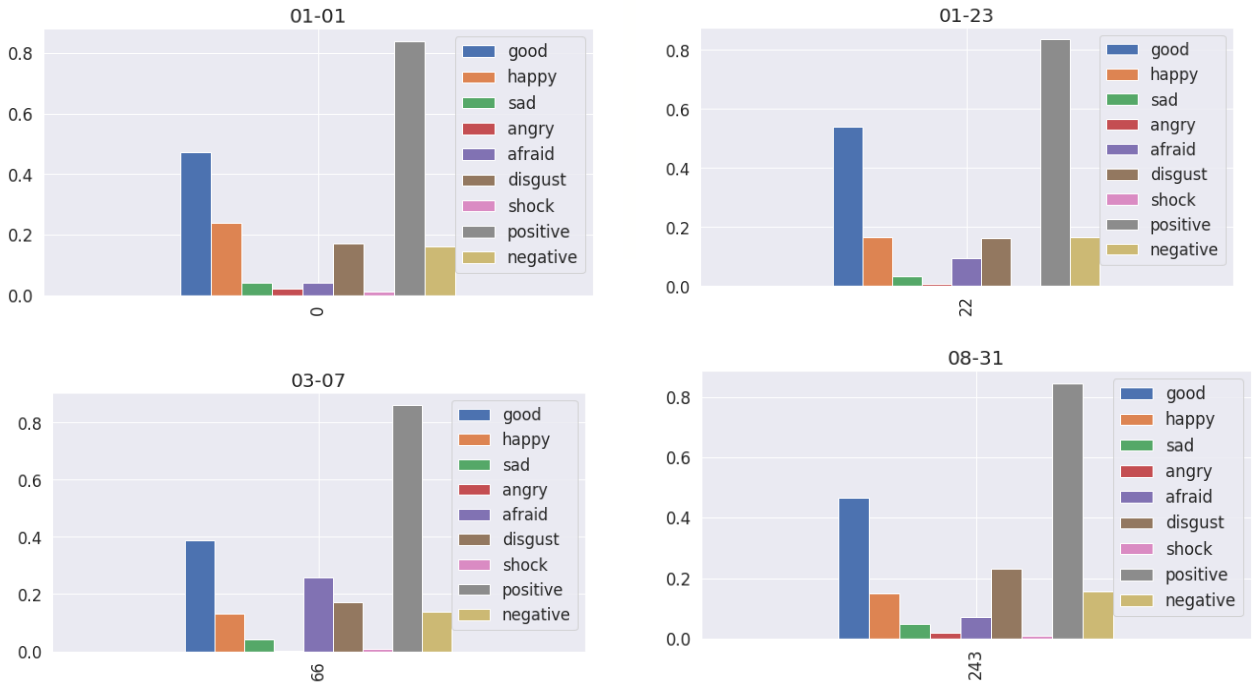

Fig6. 新闻标题情感提取

从视频中可以中可以观察到，新闻标题中正面词一直远多于负面词，占比一直都在80%左右。而七种情绪中，‘好’总是最多的。（这可能是国人多喜欢‘报喜不报忧’？）

值得注意的是‘惧’（图中afraid）这种情绪。在新年伊始时，这种情绪几乎不存在于新闻标题中，但是1月22日后疫情的消息传播开来后，这种情绪逐渐变多，在三四月份（可能是全球疫情形势严峻）时达到顶峰，在七八月份国内疫情逐渐稳定时又回到较低的值。惧这种情绪，伴随着疫情起起落落，可见人们对疫情主要还是惧怕的。

2.3 部分代码

第二部分的一些代码。

2.3.1 特征提取

```
def extract_news_features(txt, top_n):
  #提取情感部分
  result0 = emotion.emotion_count(txt)
  result1 = senti.sentiment_count(txt)
  result2 = senti.sentiment_calculate(txt)

  string_data = txt
  pattern = re.compile(u'\t|\n|\.|-|:|;|\)|\(|\?|"')
  string_data = re.sub(pattern, '', string_data)      # 将符合模式的字符去除

  seg_list_exact = jieba.cut(string_data, cut_all=False, HMM=True)
  object_list = []

  for word in seg_list_exact:          # 循环读出每个分词
      if word not in stopwords:        # 如果不在去除词库中
          object_list.append(word)     # 分词追加到列表

  word_counts = collections.Counter(object_list)        # 对分词做词频统计
  word_counts_top = word_counts.most_common(top_n)     # 获取前number个最高频的词

  return result0, result1, result2, word_counts_top

def file_to_df(i, date, df):
  file_path = '/content/drive/MyDrive/Colab Notebooks/xtzx/nlp6/news/news/data/' + date +  '.txt'
  with open(file_path) as f:
    data = json.loads(f.read())

  title_txt = ''
  for j, news in enumerate(tqdm(data)):
    title_txt += ' '
    title_txt += news['meta']['title']

  r0, r1, r2, word_counts_top = extract_news_features(title_txt, top_n)
```

```
    df.loc[i, 'date'] = date
    df.loc[i, 'num'] = len(data)
    df.loc[i, '好'] = r0['好']
    df.loc[i, '乐'] = r0['乐']
    df.loc[i, '哀'] = r0['哀']
    df.loc[i, '怒'] = r0['怒']
    df.loc[i, '惧'] = r0['惧']
    df.loc[i, '恶'] = r0['恶']
    df.loc[i, '惊'] = r0['惊']
    df.loc[i, '情感sum'] = r0['好'] + r0['乐'] +  r0['哀'] + r0['怒'] +
r0['惧'] + r0['恶'] + r0['惊']
    df.loc[i, 'pos1'] = r1['pos']
    df.loc[i, 'neg1'] = r1['neg']
    df.loc[i, 'pos2'] = r2['pos']
    df.loc[i, 'neg2'] = r2['neg']
    df.loc[i, 'sum1'] = r1['pos'] + r1['neg']
    df.loc[i, 'sum2'] = r2['pos'] + r2['neg']

    return df, word_counts_top



def t2dic(top_words):
    top_dict = {}
    for t in top_words:
        top_dict[t[0]] = t[1]
    return top_dict

def add_dic(dict1, dict2):
    comb_dict = {}
    for key in dict1.keys():
        comb_dict[key] = dict1[key]

    for key in dict2.keys():
        if key not in comb_dict.keys():
            comb_dict[key] = dict2[key]
        else:
            comb_dict[key] += dict2[key]
    return comb_dict

def del_dic(dict1, dict2):
    comb_dict = {}
    for key in dict1.keys():
```

```
    comb_dict[key] = dict1[key]

  for key in dict2.keys():
    if key not in comb_dict.keys():
      continue
    else:
      comb_dict[key] -= dict2[key]
      if comb_dict[key] <= 0:
        del comb_dict[key]

  return comb_dict


def extract_news_features(txt, top_n):
  #提取情感部分
  result0 = emotion.emotion_count(txt)
  result1 = senti.sentiment_count(txt)
  result2 = senti.sentiment_calculate(txt)

  string_data = txt
  pattern = re.compile(u'\t|\n|\.|-|:|;|\)|\(|\?|"') # 定义正则表达式匹配
模式（空格等）
  string_data = re.sub(pattern, '', string_data)      # 将符合模式的字符去
除

  seg_list_exact = jieba.cut(string_data, cut_all=False, HMM=True)     #
精确模式分词+HMM
  object_list = []

  for word in seg_list_exact:          # 循环读出每个分词
      if word not in stopwords:        # 如果不在去除词库中
          object_list.append(word)     # 分词追加到列表

  word_counts = collections.Counter(object_list)         # 对分词做词频统计
  word_counts_top = word_counts.most_common(top_n)     # 获取前number个最
高频的词

  return result0, result1, result2, word_counts_top

def file_to_df(i, date, df):
  file_path = '/content/drive/MyDrive/Colab
Notebooks/xtzx/nlp6/news/news/data/' + date +  '.txt'
  with open(file_path) as f:
    data = json.loads(f.read())
```

```
  title_txt = ''
  for j, news in enumerate(tqdm(data)):
    title_txt += ' '
    title_txt += news['meta']['title']

  r0, r1, r2, word_counts_top = extract_news_features(title_txt, top_n)

  df.loc[i, 'date'] = date
  df.loc[i, 'num'] = len(data)
  df.loc[i, '好'] = r0['好']
  df.loc[i, '乐'] = r0['乐']
  df.loc[i, '哀'] = r0['哀']
  df.loc[i, '怒'] = r0['怒']
  df.loc[i, '惧'] = r0['惧']
  df.loc[i, '恶'] = r0['恶']
  df.loc[i, '惊'] = r0['惊']
  df.loc[i, '情感sum'] = r0['好'] + r0['乐'] +  r0['哀'] + r0['怒'] +
r0['惧'] + r0['恶'] + r0['惊']
  df.loc[i, 'pos1'] = r1['pos']
  df.loc[i, 'neg1'] = r1['neg']
  df.loc[i, 'pos2'] = r2['pos']
  df.loc[i, 'neg2'] = r2['neg']
  df.loc[i, 'sum1'] = r1['pos'] + r1['neg']
  df.loc[i, 'sum2'] = r2['pos'] + r2['neg']

  return df, word_counts_top
```

2.3.2 视频部分

```
import cv2
import time

videoWriter = cv2.VideoWriter('/content/drive/MyDrive/Colab
Notebooks/xtzx/nlp6/wordcloud.mp4', cv2.VideoWriter_fourcc(*'MJPG'),
25, (1000,860), isColor = True)
# - fps为25,即每秒25张图片
# - 视频尺寸大小为1000, 860
# - isColor可以为true,flase选择是否有颜色

for i in range(0,244):
```

```
    img  = cv2.imread('/content/drive/MyDrive/Colab
Notebooks/xtzx/nlp6/wordclouds/'+str(i)+'.jpg')
    img = cv2.resize(img,(1000, 860))

    # 如下让每张图显示1秒，具体与fps相等
    a = 0
    while a < 10:
        videoWriter.write(img)
        a += 1

videoWriter.release()
```

三、小结

在此次实验中，再次使用预训练模型RoBERTa辅助了谣言的判定，并分析了数据的不同部分对谣言判定的影响。尝试了绘制词云，观察了谣言高频词的分布。对新闻标题进行了情感的提取和高频词词云绘制，观察到疫情带给社会关注点和情绪的变化。