WeRateDogs: wrangle_report

> 简要描述你的清洗过程。这可以作为内部文档。

读取数据

1. 读入WeRateDogs推特档案twitter-archive-enhanced.csv，命名该DataFrame为archive。
2. 使用Python的`Requests`和对应的URL来进行编程下载image-predictions.tsv，命名该DataFrame为image_pre。
3. 使用Python Tweepy把每个推特的JSON数据的完整集合存储到一个名为tweet_json.txt的文件中。（需要登外网，直接读入，命名该DataFrame为tweet_df。）

建立清洗副本

```
archive_clean = archive.copy()
image_pre_clean = image_pre.copy()
tweet_df_clean = tweet_df.copy()
```

质量问题

`archive` 表格

- `in_reply_to_status_id`、`in_reply_to_user_id`、`retweeted_status_id`、`retweeted_status_user_id`类型为float64，显示错误。使用`astype`函数改变类型，并且使用`str.pad()`控制字符长度为18位，并`fillna`为‘0’。

```
# 例子：规整in_reply_to_status_id字段
archive_clean.in_reply_to_status_id =
archive_clean.loc[archive_clean.in_reply_to_status_id.notnull() ==
True,'in_reply_to_status_id'].astype(int).astype(str).str.pad(18,fillchar='0')
```

- timestamp和retweeted_status_timestamp为str，应为datetime对象。使用pd.to_datetime()函数将其转换为datetime对象。

```
archive_clean.timestamp =
pd.to_datetime(archive_clean.timestamp,infer_datetime_format=True)
```

- `archive`表格中doggo、floofer、pupper、puppo列为None的缺失值应统一为NaN。

```
archive_clean.loc[archive_clean['name']=='None','name'] = np.nan
```

- `name`列中有‘a’、‘an’、‘the’等无效信息。使用正则化筛选出这些字符，并将其设置为缺失值。

```
wrong_name = archive_clean.name.str.extract('(^[a-z]*)').value_counts().index.tolist()
wrong_name.pop(0)
for name in wrong_name:
    archive_clean.loc[archive.name == name, 'name'] = np.nan
```

- `archive`表格中有name项缺失，可以在text项中分析得到。
  通过正则化得到15个缺失的名字，建立`update_name`，并且update到archive_clean中。

```
update_name = archive_clean[archive_clean.name.isnull() == True].loc[:,['text','name']]
update_name['name'] = update_name.text.str.extract('(\sis\s[A-Z]+[a-z]*)').str[3:]
archive_clean.update(update_name)
```

- `archive`表格中有`rating_denominator`分数不为10，有23条，需要一一检查清洗。

1. 首先提取text中含有**/10分数的项目，进行update。
2. 其他的评分往往是因为图上有大于一只狗（x只狗），所以分母为10x，将这些数据的`rating_numerator`除以x，并规整分母为10。
3. 发现一些误差数据，text中没有分数，而是其他污染信息。tweet_id为832088576586297345的text中的11/15/15是账户创建日期，应修改rating_numerator、rating_denominator为空值。tweet_id为810984652412424192的评分不为24/7，text中的24/7是指狗狗微笑7天24h（每时每刻），应修改rating_numerator、rating_denominator为空值。

```
# 1. 提取text中含有分母为10的字段进行清洗
update_grade = archive_clean[archive_clean.rating_denominator !=10].loc[:,
['text','rating_numerator','rating_denominator']]
update_grade['new_grade'] = update_grade.text.str.extract('(\d+\/10)').str[:-3]
update_grade.loc[update_grade['new_grade'].isnull() == False, 'rating_numerator'] =
update_grade.new_grade
update_grade.loc[update_grade['new_grade'].isnull() == False, 'rating_denominator'] = 10
archive_clean.update(update_grade)
archive_clean.rating_numerator = archive_clean.rating_numerator.astype(int)
archive_clean.rating_denominator = archive_clean.rating_denominator.astype(int)

# 2. 一张照片多张狗狗
archive_clean.rating_numerator =
archive_clean.rating_numerator.truediv((archive_clean.rating_denominator.div(10)))
archive_clean.rating_denominator = 10

# 3. 异常值
archive_clean.loc[archive_clean.tweet_id == 832088576586297345, 'rating_numerator'] =
np.nan
archive_clean.loc[archive_clean.tweet_id == 832088576586297345, 'rating_denominator'] =
np.nan
archive_clean.loc[archive_clean.tweet_id == 810984652412424192, 'rating_numerator'] =
np.nan
archive_clean.loc[archive_clean.tweet_id == 810984652412424192, 'rating_denominator'] =
np.nan
```

- `archive`表格中doggo、pupper、puppo、floofer信息缺失，需要在text列中找到并补全。通过extract方法提取信息，并赋值。

```
archive_clean.loc[archive_clean.text.str.extract('(doggo)').notnull() ==
True,'doggo']='doggo'
```

`image_pre` 表格

- `image_pre`表格中有重复记录，需要去重。使用`drop`方法删除筛选出的duplicated的项。

```
image_pre = image_pre.drop(image_pre[image_pre['jpg_url'].duplicated()].index)
```

清洁度

- source列中冗余信息多，值可简化为'iPhone','Vine','Web','TweetDeck'。

```
archive_clean.loc[archive_clean.source.str.extract('(iPhone)').notnull() ==
True,'source']='iPhone'
archive_clean.loc[archive_clean.source.str.extract('(Vine)').notnull() ==
True,'source']='Vine'
archive_clean.loc[archive_clean.source.str.extract('(Web)').notnull() ==
True,'source']='Web'
archive_clean.loc[archive_clean.source.str.extract('(TweetDeck)').notnull() ==
True,'source']='TweetDeck'
```

- 每条tweet应该有retweet_count和favorite_count将archive和tweet_df表merge在一起。

```
archive_clean = pd.merge(archive_clean, tweet_df_clean, on=['tweet_id'],how = 'left')
archive_clean = pd.merge(archive_clean, image_pre_clean, on=['tweet_id'],how = 'left')
```