In [3]:

```
import pandas as pd
import seaborn as sns
import re, string
import matplotlib.pyplot as plt
import nltk
#nltk.download("stopwords")
plt.style.use('ggplot')
```

In [4]:

```
train_data = pd.read_csv("/home/lzj/桌面/nlp/train.csv")
test_data = pd.read_csv("/home/lzj/桌面/nlp/test.csv")
```

In [5]:

```
#训练数据有7613个观察值和5个特征，包括目标(我们想要预测的标签)。
print("train_data shape:", train_data.shape)
train_data.head()
```

```
train_data shape: (7613, 5)
```

Out[5]:

| | id | keyword | location | text | target |
|---|---|---|---|---|---|
| **0** | 1 | NaN | NaN | Our Deeds are the Reason of this #earthquake M... | 1 |
| **1** | 4 | NaN | NaN | Forest fire near La Ronge Sask. Canada | 1 |
| **2** | 5 | NaN | NaN | All residents asked to 'shelter in place' are ... | 1 |
| **3** | 6 | NaN | NaN | 13,000 people receive #wildfires evacuation or... | 1 |
| **4** | 7 | NaN | NaN | Just got sent this photo from Ruby #Alaska as ... | 1 |

In [6]:

```
#测试数据有3263个观察值和四个特征
print("test_data shape:",  test_data.shape)
test_data.head()
```

```
test_data shape: (3263, 4)
```

Out[6]:

| | id | keyword | location | text |
|---|---|---|---|---|
| **0** | 0 | NaN | NaN | Just happened a terrible car crash |
| **1** | 2 | NaN | NaN | Heard about #earthquake is different cities, s... |
| **2** | 3 | NaN | NaN | there is a forest fire at spot pond, geese are... |
| **3** | 9 | NaN | NaN | Apocalypse lighting. #Spokane #wildfires |
| **4** | 11 | NaN | NaN | Typhoon Soudelor kills 28 in China and Taiwan |

In [7]:

```
# 统计 trian_data 里面各类 twitter 数量
Real_len = train_data[train_data['target'] == 1].shape[0]
Not_len = train_data[train_data['target'] == 0].shape[0]
Real_len, Not_len
```

Out[7]:

```
(3271, 4342)
```

In [8]:

```
# 先来看看 灾难 twitter 的样子
disaster_tweets = train_data[train_data['target']==1]['text']
disaster_tweets.values[1:5]
```

Out[8]:

```
array(['Forest fire near La Ronge Sask. Canada',
       "All residents asked to 'shelter in place' are being notified b
y officers. No other evacuation or shelter in place orders are expecte
d",
       '13,000 people receive #wildfires evacuation orders in Californ
ia ',
       'Just got sent this photo from Ruby #Alaska as smoke from #wild
fires pours into a school '],
      dtype=object)
```

In [9]:

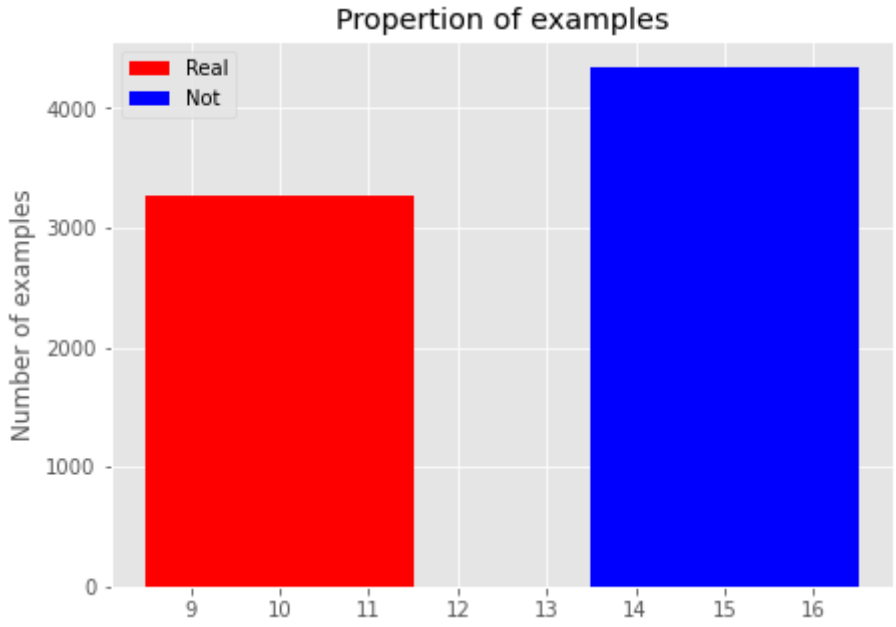
```
# 不是灾难的 twitter 的样子
non_disaster_tweets = train_data[train_data['target']==0]['text']
non_disaster_tweets.values[1:5]
```

Out[9]:

```
array(['I love fruits', 'Summer is lovely', 'My car is so fast',
       'What a gooooooooaaaaaal!!!!!!'], dtype=object)
```

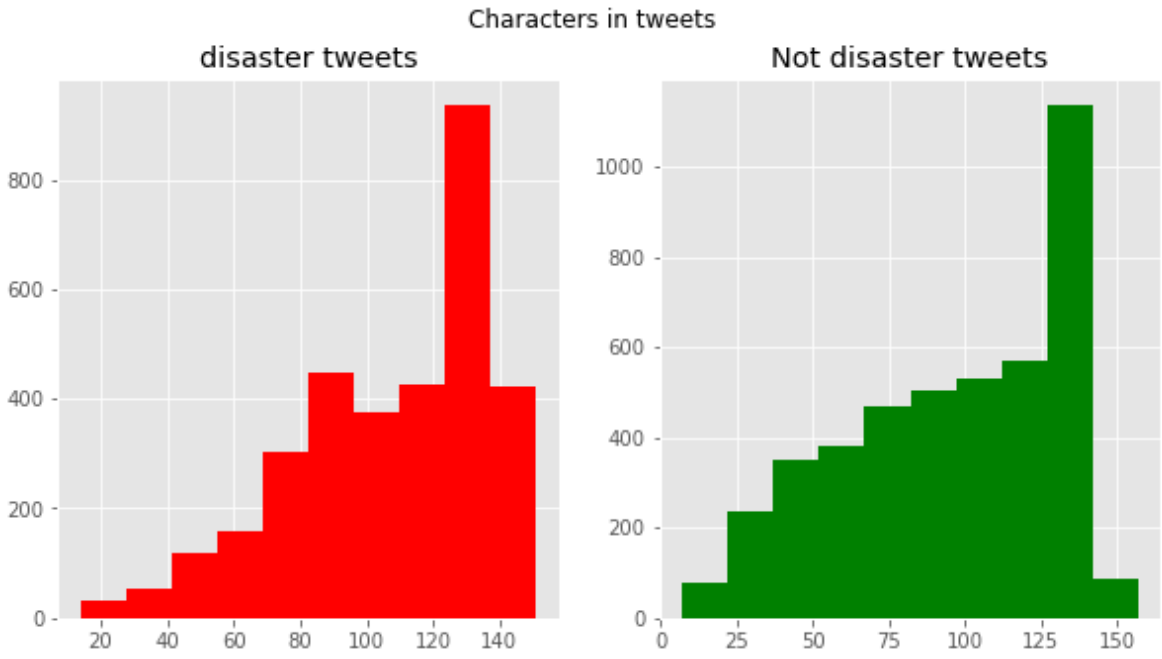
In [10]:

```
# 画出两类 twitter 数量统计直方图

plt.rcParams['figure.figsize'] = (7, 5)
plt.bar(10,Real_len,3, label="Real", color='red')
plt.bar(15,Not_len,3, label="Not", color='blue')
plt.legend()
plt.ylabel('Number of examples')
plt.title('Propertion of examples')
plt.show()
```



In [11]:

```
# 统计 字符数量与 最终目标的关系

fig,(ax1,ax2)=plt.subplots(1,2,figsize=(10,5))
tweet_len=train_data[train_data['target']==1]['text'].str.len()
ax1.hist(tweet_len,color='red')
ax1.set_title('disaster tweets')
tweet_len=train_data[train_data['target']==0]['text'].str.len()
ax2.hist(tweet_len,color='green')
ax2.set_title('Not disaster tweets')
fig.suptitle('Characters in tweets')
plt.show()
#从统计直方图可以看出， 无论是disaster 还是 No disaster tweets 大致字符范围是 120 ～ 140. 相
```

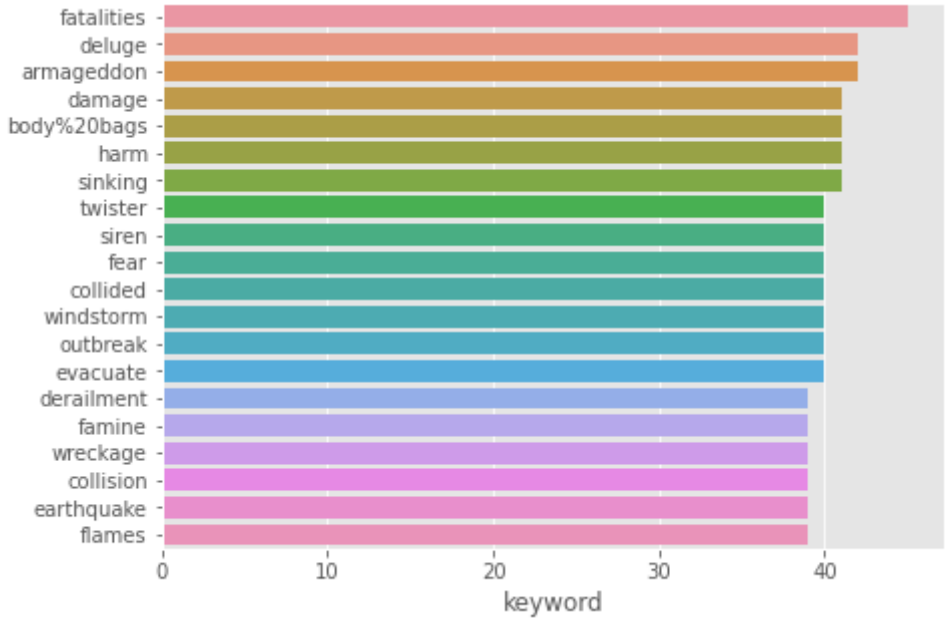


In [12]:

```
# 特征“keyword”分析
#让我们先来看看 前 20位 keyword 是什么样的

sns.barplot(y=train_data['keyword'].value_counts()[:20].index,x=train_data['keyword
            orient='h')
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa92eef2c70>
```




In [13]:

```
#包含 词语"disaster" 的twitter 和 "target" 的关系

train_data.loc[train_data['text'].str.contains('disaster', na=False, case=False)].t
```

Out[13]:

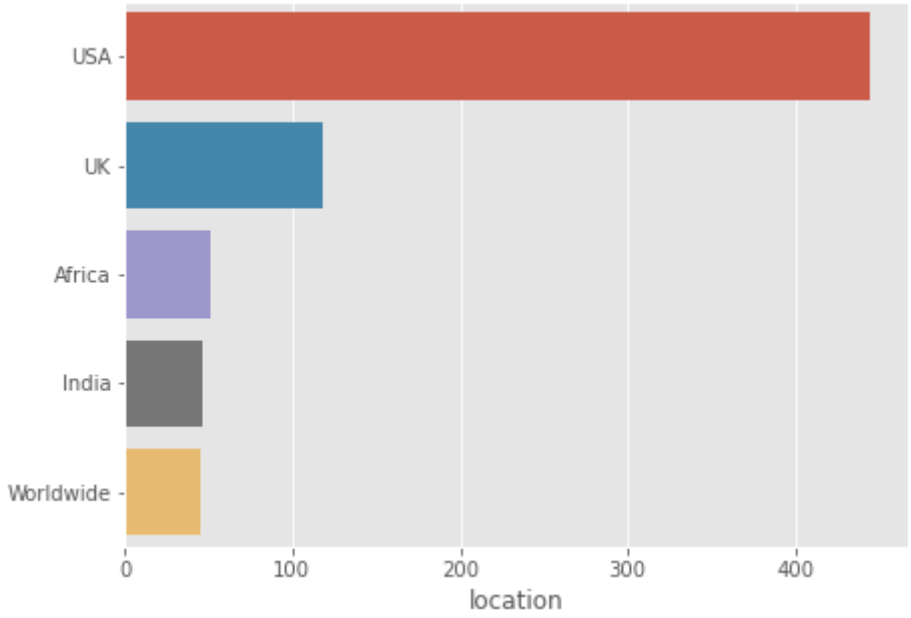
```
1    102
0     40
Name: target, dtype: int64
```

In [14]:

```
# 特征 “location” 分析
# 用简称 替换 标准名称， 并统计前 5 位
train_data['location'].replace({'United States':'USA',
                                'New York':'USA',
                                 "London":'UK',
                                 "Los Angeles, CA":'USA',
                                 "Washington, D.C.":'USA',
                                 "California":'USA',
                                  "Chicago, IL":'USA',
                                  "Chicago":'USA',
                                 "New York, NY":'USA',
                                 "California, USA":'USA',
                                 "FLorida":'USA',
                                 "Nigeria":'Africa',
                                 "Kenya":'Africa',
                                 "Everywhere":'Worldwide',
                                 "San Francisco":'USA',
                                 "Florida":'USA',
                                 "United Kingdom":'UK',
                                 "Los Angeles":'USA',
                                 "Toronto":'Canada',
                                 "San Francisco, CA":'USA',
                                 "NYC":'USA',
                                 "Seattle":'USA',
                                 "Earth":'Worldwide',
                                 "Ireland":'UK',
                                 "London, England":'UK',
                                 "New York City":'USA',
                                 "Texas":'USA',
                                 "London, UK":'UK',
                                 "Atlanta, GA":'USA',
                                 "Mumbai":"India"},inplace=True)

sns.barplot(y=train_data['location'].value_counts()[:5].index,x=train_data['locatio
            orient='h')
```

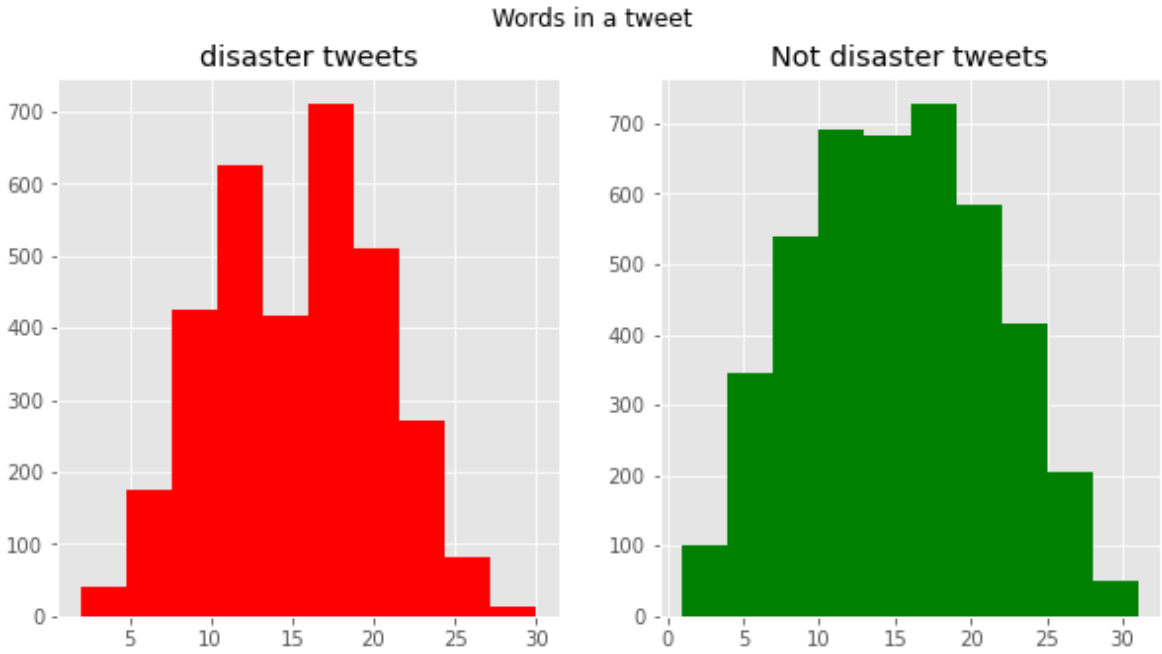
Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa92edd9910>
```




In [15]:

```
# 现在开始 "text" 特征分析， 这是最重要的特征

fig,(ax1,ax2)=plt.subplots(1,2,figsize=(10,5))
tweet_len=train_data[train_data['target']==1]['text'].str.split().map(lambda x: len
ax1.hist(tweet_len,color='red')
ax1.set_title('disaster tweets')
tweet_len=train_data[train_data['target']==0]['text'].str.split().map(lambda x: len
ax2.hist(tweet_len,color='green')
ax2.set_title('Not disaster tweets')
fig.suptitle('Words in a tweet')
plt.show()
```

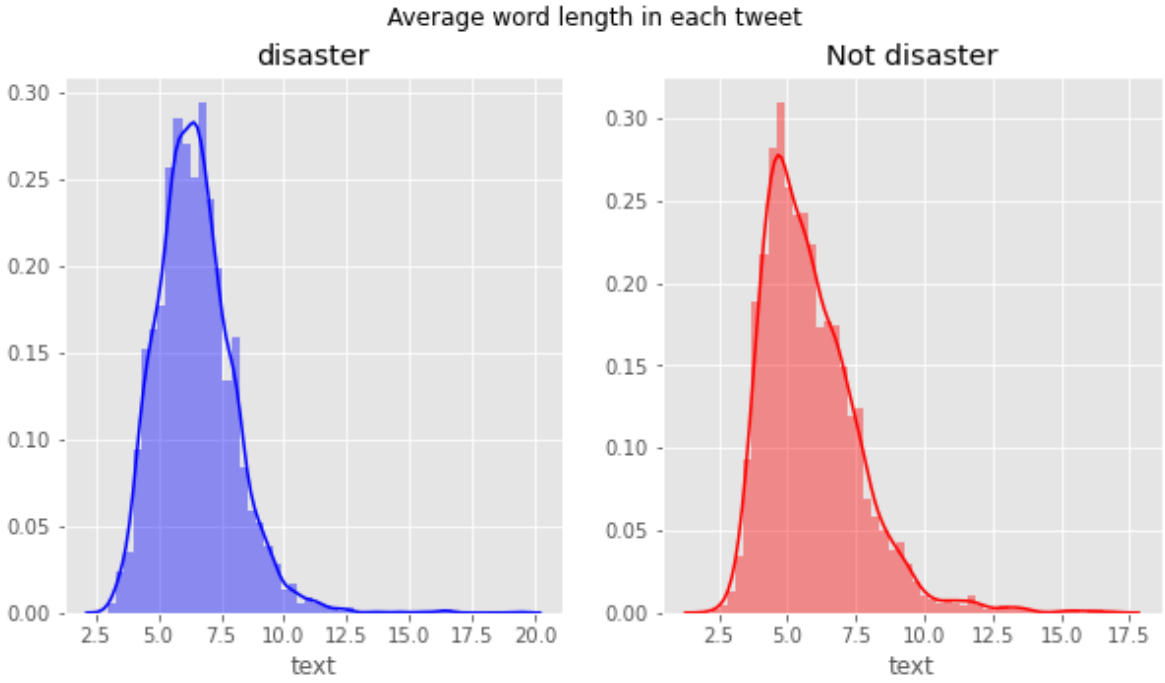


In [16]:

```
import numpy as np
# 统计平均 twitter 长度


fig,(ax1,ax2)=plt.subplots(1,2,figsize=(10,5))
word=train_data[train_data['target']==1]['text'].str.split().apply(lambda x : [len(
sns.distplot(word.map(lambda x: np.mean(x)),ax=ax1,color='blue')
ax1.set_title('disaster')
word=train_data[train_data['target']==0]['text'].str.split().apply(lambda x : [len(
sns.distplot(word.map(lambda x: np.mean(x)),ax=ax2,color='red')
ax2.set_title('Not disaster')
fig.suptitle('Average word length in each tweet')
```

Out[16]:

```
Text(0.5, 0.98, 'Average word length in each tweet')
```



In [17]:

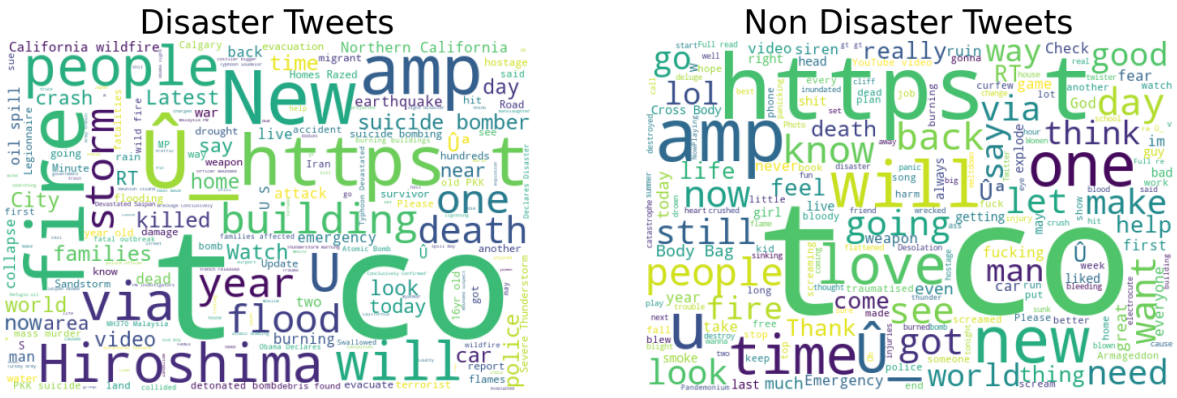
```
# 在将数据传递给模型之前，需要对数据进行预处理， 比如转换数据到一个矩阵或向量;
# 一些基本的文本预处理技术包括，大小写，标点符号去除，转义字符等等;
# 现定义函数， 文本清理函数

def clean_text(text):
    '''Make text lowercase, remove text in square brackets,remove links,remove punc
    and remove words containing numbers.'''
    text = text.lower() # 全部小写
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
```

In [18]:

```
# 同时对 train_data 和 test_data 的 "text" 类作清理
train_data['text'] = train_data['text'].apply(lambda x: clean_text(x))
test_data['text'] = test_data['text'].apply(lambda x: clean_text(x))
```

In [19]:

```
# 词云， just for fun
from wordcloud import WordCloud
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=[26, 8])
wordcloud1 = WordCloud( background_color='white',
                        width=600,
                        height=400).generate(" ".join(disaster_tweets))
ax1.imshow(wordcloud1)
ax1.axis('off')
ax1.set_title('Disaster Tweets',fontsize=40);

wordcloud2 = WordCloud( background_color='white',
                        width=600,
                        height=400).generate(" ".join(non_disaster_tweets))
ax2.imshow(wordcloud2)
ax2.axis('off')
ax2.set_title('Non Disaster Tweets',fontsize=40);
```



In [20]:

```
# 对文本分词
tokenizer = nltk.tokenize.RegexpTokenizer(r'\w+')
train_data['text'] = train_data['text'].apply(lambda x: tokenizer.tokenize(x))
test_data['text'] = test_data['text'].apply(lambda x: tokenizer.tokenize(x))
train_data['text'].head()
```

Out[20]:

```
0    [our, deeds, are, the, reason, of, this, earth...
1        [forest, fire, near, la, ronge, sask, canada]
2    [all, residents, asked, to, shelter, in, place...
3    [people, receive, wildfires, evacuation, order...
4    [just, got, sent, this, photo, from, ruby, ala...
Name: text, dtype: object
```

In [21]:

```
test_data["text"].head()
```

Out[21]:

```
0              [just, happened, a, terrible, car, crash]
1    [heard, about, earthquake, is, different, citi...
2    [there, is, a, forest, fire, at, spot, pond, g...
3           [apocalypse, lighting, spokane, wildfires]
4    [typhoon, soudelor, kills, in, china, and, tai...
Name: text, dtype: object
```

In [22]:

```
# 从新将 token List 转成文本
def combine_text(list_of_text):
    combined_text = ' '.join(list_of_text)
    return combined_text

train_data['text'] = train_data['text'].apply(lambda x : combine_text(x))
test_data['text'] = test_data['text'].apply(lambda x : combine_text(x))
train_data['text']
train_data.head()
```

Out[22]:

| | id | keyword | location | text | target |
|---|---|---|---|---|---|
| **0** | 1 | NaN | NaN | our deeds are the reason of this earthquake ma... | 1 |
| **1** | 4 | NaN | NaN | forest fire near la ronge sask canada | 1 |
| **2** | 5 | NaN | NaN | all residents asked to shelter in place are be... | 1 |
| **3** | 6 | NaN | NaN | people receive wildfires evacuation orders in ... | 1 |
| **4** | 7 | NaN | NaN | just got sent this photo from ruby alaska as s... | 1 |

In [23]:

```
# 为了构建词典， 把 train_data 和 test_data合并
combine_data = pd.concat([train_data, test_data], axis=0, ignore_index=True)
combine_data
```

Out[23]:

| | id | keyword | location | text | target |
|---|---|---|---|---|---|
| **0** | 1 | NaN | NaN | our deeds are the reason of this earthquake ma... | 1.0 |
| **1** | 4 | NaN | NaN | forest fire near la ronge sask canada | 1.0 |
| **2** | 5 | NaN | NaN | all residents asked to shelter in place are be... | 1.0 |
| **3** | 6 | NaN | NaN | people receive wildfires evacuation orders in ... | 1.0 |
| **4** | 7 | NaN | NaN | just got sent this photo from ruby alaska as s... | 1.0 |
| **...** | ... | ... | ... | ... | ... |
| **10871** | 10861 | NaN | NaN | earthquake safety los angeles ûò safety fasten... | NaN |
| **10872** | 10865 | NaN | NaN | storm in ri worse than last hurricane my harde... | NaN |
| **10873** | 10868 | NaN | NaN | green line derailment in chicago | NaN |
| **10874** | 10874 | NaN | NaN | meg issues hazardous weather outlook hwo | NaN |
| **10875** | 10875 | NaN | NaN | cityofcalgary has activated its municipal emer... | NaN |

10876 rows × 5 columns

In [24]:

```
combine_data["text"].tolist()
```

Out[24]:

```
['our deeds are the reason of this earthquake may allah forgive us a
ll',
 'forest fire near la ronge sask canada',
 'all residents asked to shelter in place are being notified by offi
cers no other evacuation or shelter in place orders are expected',
 'people receive wildfires evacuation orders in california',
 'just got sent this photo from ruby alaska as smoke from wildfires
pours into a school',
 'rockyfire update california hwy closed in both directions due to l
ake county fire cafire wildfires',
 'flood disaster heavy rain causes flash flooding of streets in mani
tou colorado springs areas',
 'im on top of the hill and i can see a fire in the woods',
 'theres an emergency evacuation happening now in the building acros
s the street',
 'im afraid that the tornado is coming to our area',
 'three people died from the heat wave so far',
 'haha south tampa is getting flooded hah wait a second i live in so
```

In [25]:

```
# 定义分词语函数

data = [review.split(" ") for review in combine_data["text"].tolist()]
len(data), data
```

Out[25]:

```
(10876,
 [['our',
   'deeds',
   'are',
   'the',
   'reason',
   'of',
   'this',
   'earthquake',
   'may',
   'allah',
   'forgive',
   'us',
   'all'],
  ['forest', 'fire', 'near', 'la', 'ronge', 'sask', 'canada'],
  ['all',
   'residents',
   'asked',
```

In [26]:

```
# 开始构建词典
import collections
from mxnet import contrib

counter = collections.Counter([tk for st in data for tk in st])
vocab = contrib.text.vocab.Vocabulary(counter,reserved_tokens = ['<pad>'])
print("词典中共有词语: %d 个" %len(vocab))
```

词典中共有词语: 20830 个

In [27]:

```
# 因为每条评论的长度不一致， 所以不能直接组合成小批量， 现在对每条评论 截断或是 补 “<pad>”符号，
#长度固定成 50
from mxnet import nd
import mxnet
def preprocess_twitter(data, vocab):
    max_l = 50

    def pad(x):
        return x[:max_l] if len(x) > max_l else x+[vocab.token_to_idx['<pad>']]*(ma


    features = nd.array([pad(vocab.to_indices(x)) for x in  data[0:len(train_data)]

    labels = nd.array(train_data["target"].tolist())
    return features, labels



# 创建数据 迭代器， 每次返回一个小批量的数据
batch_size = 32
train_set = mxnet.gluon.data.ArrayDataset(*preprocess_twitter(data[0:len(train_data
train_iter = mxnet.gluon.data.DataLoader(train_set, batch_size, shuffle = True)

#
test_iter = train_iter

# 打印第一个小批量数据的形状以及训练中小批量的个数
for x, y in train_iter:
    print("x", x.shape, "y", y.shape)
    break
"batches:", len(train_iter),

features, labels = preprocess_twitter(data[0:len(train_data)], vocab)
features.shape, labels.shape
```

```
x (32, 50) y (32,)
```

Out[27]:

```
((7613, 50), (7613,))
```

In [28]:

```
# 定义网络模型
# 在这个模型中， 每个词先通过 嵌入层 得到 词的特征向量。 然后双向循环神经网络对 特征序列 编码 得到
# 最后 将得到的 编码信息 通过 全连接层 变换曾成 输出。 将双向长短期记忆 在最初时间步 和 最终时间
# 连接， 作为特征的表征， 传递到 输出层 分类。

import mxnet as mx
from mxnet.gluon import nn, rnn
class BiRNN(nn.Block):
    def __init__(self, vocab, embed_size, num_hiddens, num_layers, **kwargs):
        super(BiRNN, self).__init__(**kwargs)
        self.embedding = nn.Embedding(len(vocab), embed_size)

        self.encoder = rnn.LSTM(num_hiddens, num_layers=num_layers, bidirectional=T
        self.decoder = nn.Dense(units=2)


    def forward(self, inputs):
        embeddings = self.embedding(inputs.T)
        outputs = self.encoder(embeddings)
        encoding = nd.concat(outputs[0], outputs[-1])
        outs = self.decoder(encoding)
        return outs

# 向量
def try_all_gpus():
    ctxes = []
    try:
        for i in range(5):
            ctx = mx.gpu(i)
            _ = nd.array([0], ctx=ctx)
            ctxes.append(ctx)
    except mx.base.MXNetError:
        pass
    if not ctxes:
        ctxes = [mx.cpu()]
    return ctxes

# 创建一个 含有 2个 隐藏层的 双向循环神经网络.
embed_size, num_hiddens, num_layers, ctx = 100, 100,2, mx.gpu()
net = BiRNN(vocab, embed_size, num_hiddens, num_layers)
net.initialize(init=mxnet.init.Xavier(), ctx=ctx)

# 加载预训练的词向量
glove_embedding = contrib.text.embedding.create('glove', pretrained_file_name = 'gl

net.embedding.weight.set_data(glove_embedding.idx_to_vec)
net.embedding.collect_params().setattr('grad_seq', 'null')
```

In [29]:

```
# 由于训练数据集的样本不多，使用 k折 交叉验证方法， 调节模型超参数

# 定义函数， 他返回第 i 折 交叉验证时 训练集 和 验证集

def get_k_fold_data(k, i, X, y):
    assert k > 1
    fold_size = X.shape[0] // k
    X_train, y_train = None, None
    for j in range(k):
        idx = slice(j * fold_size, (j + 1) * fold_size)
        X_part, y_part = X[idx, :], y[idx]
        if j == i:
            X_valid, y_valid = X_part, y_part
        elif X_train is None:
            X_train, y_train = X_part, y_part
        else:
            X_train = nd.concat(X_train, X_part, dim=0)
            y_train = nd.concat(y_train, y_part, dim=0)
    return X_train, y_train, X_valid, y_valid
```

In [30]:

```
get_k_fold_data(5,1,features, labels) #第一次切分全部训练集， 返回的训练集 和 验证集
```

Out[30]:

```
(
 [[1.010e+02 6.002e+03 2.200e+01 ... 1.000e+00 1.000e+00 1.000e+00]
  [1.560e+02 4.300e+01 1.950e+02 ... 1.000e+00 1.000e+00 1.000e+00]
  [4.200e+01 1.579e+03 1.433e+03 ... 1.000e+00 1.000e+00 1.000e+00]
  ...
  [7.601e+03 3.660e+02 1.128e+03 ... 1.000e+00 1.000e+00 1.000e+00]
  [1.060e+02 8.020e+02 1.439e+03 ... 1.000e+00 1.000e+00 1.000e+00]
  [5.579e+03 7.712e+03 2.000e+00 ... 1.000e+00 1.000e+00 1.000e+00]]
 <NDArray 6088x50 @cpu(0)>,

 [1. 1. 1. ... 1. 1. 1.]
 <NDArray 6088 @cpu(0)>,

 [[1.7670e+03 2.1000e+01 2.0000e+00 ... 1.0000e+00 1.0000e+00 1.0000e+
00]
  [1.8840e+03 3.4000e+01 1.2600e+02 ... 1.0000e+00 1.0000e+00 1.0000e+
00]
  [6.7600e+02 1.8800e+02 2.2600e+02 ... 1.0000e+00 1.0000e+00 1.0000e+
00]
  ...
  [1.7855e+04 5.3100e+02 1.4794e+04 ... 1.0000e+00 1.0000e+00 1.0000e+
00]
  [1.5010e+03 7.3400e+02 2.3200e+02 ... 1.0000e+00 1.0000e+00 1.0000e+
00]
  [2.3200e+02 5.1800e+02 1.8100e+02 ... 1.0000e+00 1.0000e+00 1.0000e+
00]]
 <NDArray 1522x50 @cpu(0)>,

 [1. 0. 0. ... 1. 1. 0.]
 <NDArray 1522 @cpu(0)>)
```

In [31]:

```
from mxnet import autograd
from mxnet.gluon import utils as gutils
import time

def _get_batch(batch, ctx):# 将当前小批量数据分发至多个cpu，数据并行;
    features, labels = batch
    if labels.dtype != features.dtype:
        labels = labels.astype(features.dtype)
    return (gutils.split_and_load(features, ctx),
            gutils.split_and_load(labels, ctx), features.shape[0])

def evaluate_accuracy(data_iter, net, ctx):# 准确率统计函数
    if isinstance(ctx, mx.Context):
        ctx = [ctx]
    acc_sum, n = nd.array([0]), 0
    for batch in data_iter:
        features, labels, _ = _get_batch(batch, ctx)
        for X, y in zip(features, labels):
            y = y.astype('float32')
            acc_sum += (net(X).argmax(axis=1) == y).sum().copyto(mx.cpu())
            n += y.size
        acc_sum.wait_to_read()
    return acc_sum.asscalar() / n
```

In [32]:

```
# 定义训练函数
a, b, c = [],[],[]
def train(train_iter, test_iter, net, loss, trainer, ctx, num_epochs):
    print('training on', ctx)
    if isinstance(ctx, mx.Context):
        ctx = [ctx]
    for epoch in range(num_epochs):
        train_l_sum, train_acc_sum, n, m, start = 0.0, 0.0, 0, 0, time.time()
        for i, batch in enumerate(train_iter):
            Xs, ys, batch_size = _get_batch(batch, ctx)
            with autograd.record():
                y_hats = [net(X) for X in Xs]
                ls = [loss(y_hat, y) for y_hat, y in zip(y_hats, ys)]
            for l in ls:
                l.backward()
            trainer.step(batch_size)
            train_l_sum += sum([l.sum().asscalar() for l in ls])
            n += sum([l.size for l in ls])
            train_acc_sum += sum([(y_hat.argmax(axis=1) == y).sum().asscalar()
                                  for y_hat, y in zip(y_hats, ys)])
            m += sum([y.size for y in ys])
        test_acc = evaluate_accuracy(test_iter, net, ctx)
        print('epoch %d, loss %.4f, train acc %.3f, test acc %.3f, '
              'time %.1f sec'
              % (epoch + 1, train_l_sum / n, train_acc_sum / m, test_acc,
                 time.time() - start))
    if num_epochs == 1:
        a.append(train_l_sum/n),  b.append(train_acc_sum/m), c.append(test_acc)
```

In [33]:

```
lr, num_epochs, k = 0.01, 1, 5
trainer = mxnet.gluon.Trainer(net.collect_params(), 'adam', {'learning_rate': lr} )
loss = mxnet.gluon.loss.SoftmaxCrossEntropyLoss()



for i in range(5):
    data = get_k_fold_data(k, i, features, labels)
    cur_train_data_set, cur_valid_data_set = mxnet.gluon.data.ArrayDataset(data[0],
    cur_train_data_iter = mxnet.gluon.data.DataLoader(cur_train_data_set, batch_siz
    cur_valid_data_iter = mxnet.gluon.data.DataLoader(cur_valid_data_set, batch_siz

#   a, b, c =
    print("%d-th fold"%i)
    train(cur_train_data_iter, cur_valid_data_iter, net, loss, trainer, ctx, num_ep
#   train_loss_sum.append(a), train_acc_sum.append(b), test_acc_sum.append(c)
s1, s2, s3 = sum(a), sum(b),  sum(c)
print("avg loss %f, avg train acc %f, avg test acc %f"%(s1/5, s2/5, s3/5))

# print('%d-fold validation:avg train loss %f, avg train rmse %f, avg valid rmse %f
#       % (k, sum(train_loss_sum)/k, sum(train_acc_sum)/k, sum(test_acc_sum)/k))
```

```
0-th fold
training on gpu(0)
epoch 1, loss 0.4873, train acc 0.778, test acc 0.783, time 2.3 sec
1-th fold
training on gpu(0)
epoch 1, loss 0.2979, train acc 0.881, test acc 0.920, time 2.2 sec
2-th fold
training on gpu(0)
epoch 1, loss 0.1542, train acc 0.943, test acc 0.949, time 2.2 sec
3-th fold
training on gpu(0)
epoch 1, loss 0.0995, train acc 0.964, test acc 0.974, time 2.2 sec
4-th fold
training on gpu(0)
epoch 1, loss 0.0713, train acc 0.973, test acc 0.984, time 2.2 sec
avg loss 0.222024, avg train acc 0.907786, avg test acc 0.922208
```

In [35]:

```
lr, num_epochs = 0.01, 10
trainer = mxnet.gluon.Trainer(net.collect_params(), 'adam', {'learning_rate': lr} )
loss = mxnet.gluon.loss.SoftmaxCrossEntropyLoss()



#开始模型训练

train(train_iter, test_iter, net, loss, trainer, ctx, num_epochs)
```

```
training on gpu(0)
epoch 1, loss 0.0410, train acc 0.982, test acc 0.983, time 3.6 sec
epoch 2, loss 0.0418, train acc 0.980, test acc 0.984, time 3.5 sec
epoch 3, loss 0.0368, train acc 0.982, test acc 0.985, time 3.5 sec
epoch 4, loss 0.0335, train acc 0.983, test acc 0.984, time 3.4 sec
epoch 5, loss 0.0390, train acc 0.982, test acc 0.984, time 3.7 sec
epoch 6, loss 0.0397, train acc 0.981, test acc 0.984, time 3.4 sec
epoch 7, loss 0.0373, train acc 0.980, test acc 0.984, time 3.4 sec
epoch 8, loss 0.0349, train acc 0.982, test acc 0.985, time 3.5 sec
epoch 9, loss 0.0310, train acc 0.985, test acc 0.986, time 3.5 sec
epoch 10, loss 0.0292, train acc 0.984, test acc 0.985, time 3.5 sec
```

In [33]:

```
#定义预测函数
def predict_sentiment(net, vocab, sentence):
    sentence = nd.array(vocab.to_indices(sentence), ctx = ctx)
    label = nd.argmax(net(sentence.reshape((1, -1))), axis=1)
    return 1 if label.asscalar() == 1 else 0

predict_sentiment(net, vocab, 'I love fruits')
```

Out[33]:

```
0
```

In [41]:

```
# test_data 预测
test_data["target"] = test_data["text"].apply(lambda x: predict_sentiment(net, voca
```

In [42]:

```
test_data
```

Out[42]:

| | id | keyword | location | text | target |
|---|---|---|---|---|---|
| **0** | 0 | NaN | NaN | just happened a terrible car crash | 0 |
| **1** | 2 | NaN | NaN | heard about earthquake is different cities sta... | 0 |
| **2** | 3 | NaN | NaN | there is a forest fire at spot pond geese are ... | 0 |
| **3** | 9 | NaN | NaN | apocalypse lighting spokane wildfires | 0 |
| **4** | 11 | NaN | NaN | typhoon soudelor kills in china and taiwan | 0 |
| **...** | ... | ... | ... | ... | ... |
| **3258** | 10861 | NaN | NaN | earthquake safety los angeles ûò safety fasten... | 0 |
| **3259** | 10865 | NaN | NaN | storm in ri worse than last hurricane my harde... | 0 |
| **3260** | 10868 | NaN | NaN | green line derailment in chicago | 0 |
| **3261** | 10874 | NaN | NaN | meg issues hazardous weather outlook hwo | 0 |
| **3262** | 10875 | NaN | NaN | cityofcalgary has activated its municipal emer... | 0 |

3263 rows × 5 columns

In [ ]:

```
#预测结果提交

def submission(submission_file_path,model,test_vectors):
    sample_submission = pd.read_csv(submission_file_path)
    sample_submission["target"] = model.predict(test_vectors)
    sample_submission.to_csv("submission.csv", index=False)


submission_file_path = "/home/lzj/桌面/nlp/sample_submission.csv"
test_vectors=test_tfidf
submission(submission_file_path,clf_NB_TFIDF,test_vectors)
```