特征工程｜文本特征处理之词袋模型+Onehot编码

原创 Thinkgamer 搜索与推荐Wiki 2020-09-14

文本特征在内容平台内使用的场景和方式更多，但并不等于说其在其他形式的平台中无用户之地，比如：电商平台中的商品标题、商品介绍、评论等，商品平台中视频标题、视频介绍、评论等。

利用文本数据可以做的事情很多，包括但不局限于：关键词提取、文本分类、文本聚类、文本情感分析、文本离散表示、文本生成等。在推荐领域，通常是基于文本内容进行挖掘，从而提取出有效特征来表示物品，可以在CTR模型中使用。

> 当然上边提到的主要是如何在CTR模型中利用文本内容，在召回或者物品画像也可以对文本内容进行挖掘，这里不再赘述，在后续的召回系列内容中会涉及，本篇文章主要介绍如何基于文本内容提取特征。

提取文本特征主流的方法有以下几种：

- 词袋模型+OneHot编码
- 词袋模型+关键词权重（TF-IDF、TextRank等）
- 词袋模型+主题模型（LSA、LDA、LDA2Vec等）
- Embedding模型

本文主要介绍词袋模型+OneHot编码。

词袋模型+OneHot编码

1、词袋模型介绍

词袋模型（Bag-of-words）即将文档库中所有词语装进一个袋子里，不考虑词法和语序的问题，即每个词语都是独立的。对于每篇文档中出现的词语映射到词库中，如果出现，则在对应位置加1，否则为0，最终某位置的数字表示的是在该文档中该词语出现的次数。

假设文档库中只有两篇文档，第一篇文档的内容为：我来自北京，我爱中国，第二篇文档的内容为：我爱天安门。按照某种分词规则得到的词库为：[我，来自，北京，爱，中国，天安门]，则第一篇文档的内容可以表示为：[2,1,1,1,1,0]，第二篇文档的内容可以表示为：[1,0,0,1,0,1]。可以看出“我”字在第一篇文档中出现了两次，所以其对应的向量表示中对应位置为2。

使用词袋模型+OneHot编码的形式表达文本，简单直观可解释 ，但其忽略了词语之间的顺序关系、词语之间的相互影响关系，且得到的特征比较离散，特征维度较高。

2、词集模型+OneHot编码

和词袋模型比较接近的是词集模型，同样需要先得到一个词库，然后不同文档中的词语对应到词库中，但其与词袋模型不同的是不进行次数的累加，即如果该词在词库中出现，则对应位置为1，否则为0，在词集模型中不考虑词频。

同样对于上文中的例子，文档1的内容可以表示为：[1,1,1,1,1,0]，文档2可以表示为：[1,0,0,1,0,1]。在日常的工作中，很少使用词集模型，这里可以作为知识了解！

3、使用sklearn中的CountVectorizer构建文本的词集模型表示

CountVectorizer是sklearn中feature_extraction.text中的一个类，更多关于feature_extraction.text的内容可以参考：https://scikit-learn.org/0.22/modules/classes.html#module-sklearn.feature_extraction.text。

CountVectorizer是将文本文档集合转换为词频矩阵，如果不提供先验的词库文档和不使用某种特定的特征提取器，则最终的特征数量将等同于文档库中词汇的大小。该类初始化时支持输入参数较多，其中主要的有：

- input：输入的数据源，支持文件列表、单个文件、内容，其中默认是内容（content）
- lowercase：是否全部转化为小写字母，默认为True，在处理英文文档时需要注意
- stop_words：停用词，如果设置为“english”，则使用默认的停用词，也支持自定义停用词list，默认为None
- analyzer：支持参数值为{'word', 'char', 'char_wb'}，word表示特征由词构成，会过滤掉单个字符的词，char表示特征由单个字符构成，不需要进行空格分割，CountVectorizer会自动按照单字进行分隔统计词频，char_wb和char的区别是会在字符串的两端补两个空格。

- max_features：根据词频对所有词语进行倒排，只保留指定的前多少个词语
- binary：如果设置为True，则所有文档非空的词语次数置为1，相当于是词集模型，默认为False，表示的是词袋模型

提取英文文档的词袋模型表示

```
from sklearn.feature_extraction.text import CountVectorizer

corpus = [
     'This is the first document.',
     'This document is the second document.',
     'And this is the third one.',
     'Is this the first document?',
]
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)
print(vectorizer.get_feature_names())
print(X.toarray())
```

输出为：

```
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

提取中文文档的词袋模型表示

```
from sklearn.feature_extraction.text import CountVectorizer

# 中文文档的词袋表示
docs = [
     "明天 天气 很棒",
     "明天 是 晴天",
     "晴天 很棒"
]

## 默认将所有单个汉字视为停用词 会进行过滤
```

```
vectorizer1=CountVectorizer(token_pattern=r"(?u)\b\w\w+\b")
X1 = vectorizer1.fit_transform(docs)
print(vectorizer1.get_feature_names())
print(X1.toarray())
```

输出为：

```
['天气', '很棒', '明天', '晴天']
[[1 1 1 0]
 [0 0 1 1]
 [0 1 0 1]]
```

The end