**优达学城毕业项目——文本分类**

2018年5月5日

肖志铭

**一、问题定义**

**1 项目概览**

文本分类问题是自然语言处理中最常见、最经典的问题之一，相关研究最早可追溯到上世纪50年代，当时主要是通过专家规则（Pattern）进行分类[1]。但到了上世纪80年代末，人们开始越来越多地关注工程化、实用化的解决方法，很多人开始研究和关注基于大规模语料的统计机器学习方法及其在自然语言处理中的应用[2]。到了现在，机器学习已成为解决文本分类问题的主要工具，并且存在多种不同的解决思路和解决方案。本项目所要做的工作，就是基于机器学习进行文本分类，并在这个过程中做一些尝试和探索。

本项目所使用的数据集为20 newsgroups数据集。该数据集收集了近20000条新闻组文档，分为了20个不同主题的新闻组集合。其中一些新闻组的主题比较相似，还有一些却几乎无关。这样的特性能有效的测试文本分类方法的性能，在主题无关和主题类似两种不同情况下的分类效果。

**2 问题描述**

文本分类是指，使用电脑对文本集按照一定的标准进行的自动分类。具体到本文的工作，就是训练一个分类器，将新闻尽可能正确的自动分类到其所属的主题之下。在20 newsgroups数据集中，每条新闻对应的主题都是已知的，因此我将训练一个基于监督学习的分类器来对文本进行分类，比如贝叶斯分类器、支持向量机等。而在此之前，一个可能更关键的问题是，如何提取到有效的文本特征，来用于监督学习分类器的训练和分类。

在本文中，将尝试使用不同的特征提取方法进行文本分类，并在最后对结果做对比分析，主要会涉及到词袋模型、word2vec等。

**3 评估指标**

对于本模型的评估指标，首先需要考虑准确率。它表述了预测属于某一类的个体的数量，与这些个体中实际属于该类的数量之比。其定义如下：

其中，TP代表了被正确分入该类的个体的数量，FP代表被错误分入该类的个体的数量。

另外，还需要考虑到召回率。它表述了实际属于某一类的个体的数量，与这些个体中被正确预测的数量之比。其定义如下：

TP同样代表了被正确分入该类的个体的数量，而FN代表被错误分出该类的个体的数量。由于需要同时考虑以上两个指标，因此使用F1值来进行均衡的考量。F1值的定义如下：

其中precision代表了准确率，而recall代表了召回率。在此没有调整precision和recall的比重关系，而认为它们具有同样的重要性。

本项目中使用的数据集有多个子类别。对项目中使用到的子类别，须分别计算它们的F1值。并在此基础上，计算它们的宏平均值。

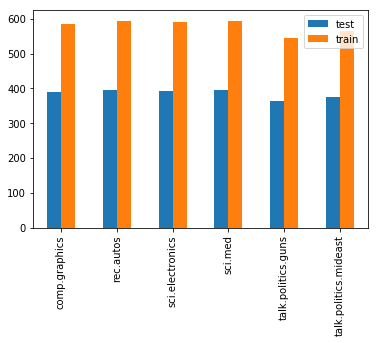
另外，运算时间也是需要关注的指标，过长的运算时间是不能接受的。在分类性能相差不大情况下，运算时间越短的分类器越应得到青睐。

**二、问题分析**

**1 数据集的分析**

通过sklearn提供的接口,可以方便的直接获取20 newsgroups数据集的数据。在本项目中，通过设置categories属性，仅使用了‘comp.graphics'、‘rec.autos’、‘sci.electronics'、‘sci.med'、‘talk.politics.guns'和‘talk.politics.mideast'六类新闻。统计这六类新闻的数据量，得到如下的图表和表格结果：

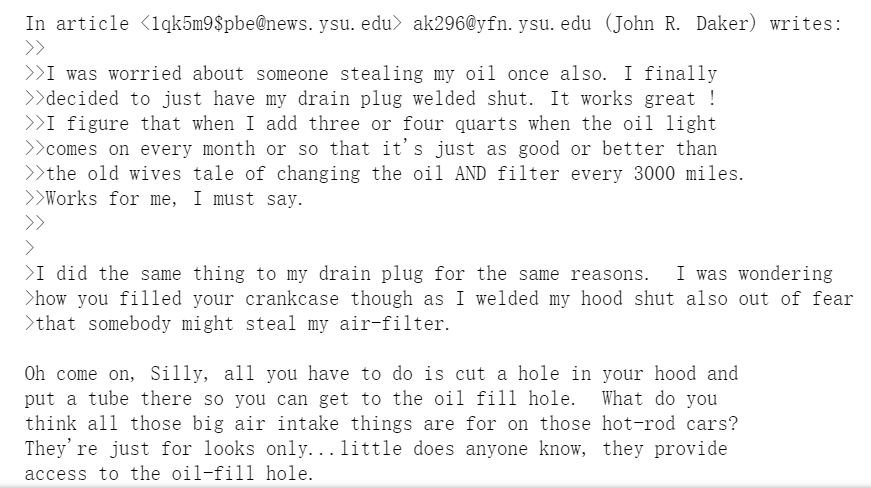
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | comp.  graphics | rec.  autos | sci.  electronics | sci.med | talk.politics.guns | talk.politics.mideast |
| 训练集数量 | 584 | 594 | 591 | 594 | 546 | 564 |
| 测试集数量 | 389 | 396 | 393 | 396 | 364 | 376 |



观察可知，训练集中每类新闻的数量大体相等，测试集中也是如此。同时，训练集中数量略多的类别在测试集中也数量略多，训练集中数量略少的类别在测试集中同样数量略少。这说明训练数据中不存在类别不平衡的问题，并且训练集和测试集中的数据是服从同一个分布的。

另外，在这六个类别中‘sci.electronics'和‘sci.med'属于同一大类，而‘talk.politics.guns'和‘talk.politics.mideast'也属于同一大类。这样的选择不仅能测试主题完全无关时的分类效果，还能测试主题可能相关时的分类效果。

在确定了使用的数据类别后，可以通过data属性查看数据集中的具体文本。比如训练集中的第一条文本部分如下：



查看该文本，注意到文章中存在着一些可能需要处理的地方，比如：去标点符号、修改大小写、删除停用词等。停用词即指一些词频较高，但却没有什么实际含义的词，比如英文中的‘the’、‘to’、‘is’等。这些细节都需要在之后的数据预处理过程中得到妥善解决。

**2 算法介绍**

本项目中涉及到的算法分为两类，一类用于提取特征而另一类用于最后的分类。

特征提取涉及到的算法有两个，以下作简要介绍：

1. tf-idf：

tf即词袋模型，统计了数据集中每个单词的出现次数。最后得到一个矩阵,它的每一行代表了数据集中的一篇文档[3]。idf

1. word2Vec:

对于最后的分类器选择，我使用了支持向量机。因为它能很好的应对高维特征的情况，比较适合处理文本分类的问题。

**3 基准测试**

在本项目中，我一共使用了六类新闻数据，在纯随机猜测的情况下，分类的准确率约为：1/6 ≈ 16.67%。因此如果总体准确率显著大于该数字，就说明分类器至少是有一定效果的。

但是，仅仅大于16.67%又显然是不够的。因此本文尝试了两个不同的分类模型，一个基于词袋而另一个基于word2Vec。其中前者更为经典和成熟，作为基准模型来使用；后者是本项目中的另一个尝试，看看是否能得到类似于前者的分类性能。

另外，是否发生过拟合也是需要关注的重点。因此在项目中会测试分类器在训练集和测试集上的性能差，查看是否存在过拟合问题。

**三、方案实施**

**1 数据预处理**

对于基准模型tf-idf而言，sklearn本身已经提供了相当完整的支持。不仅不需要手工去处理分词等工作，甚至连文档向量都可以使用工具类直接生成。因此，更多的数据预处理工作是用于生成词向量模型的。在本项目中，我对数据做了如下的预处理：

1. 分词并使用正则表达式删除标点符号。
2. 删除了所有停用词。
3. 删除了分词后不包含任何字母的词。
4. 删除了单词长度只有1的词。

在完成以上步骤后，每个文档都被转换为了一个单词列表。然后使用gensim提供的方法，便可以训练词向量模型了。

**2 实施过程**

整个实施过程分可为以下四个步骤：

1. tf-idf模型的训练和测试。sklearn对此提供了比较完善的支持，通过使用HashVectorizer，只需几行代码便可以得到文本的tf-idf向量。在得到向量后，便可训练和测试分类器了。
2. 词向量模型的建立和获取词向量。在本项目中，我使用了gensim模块来建立词向量模型，建立模型时仅使用训练数据。整个过程也比较简单，调用对应的接口，接受预处理过的训练数据，便可得到一个词向量的模型了。在获取到词向量模型后，可以直接从模型获取到所有的词向量。
3. 根据词向量生成文档的向量。如何由词向量生成合理的文档向量，是本项目的关键问题之一。本文做了两种尝试，第一种直接将词向量相加求平均值，即为文档向量；第二种结合idf，得出文档向量。

(4) 训练和测试基于文档向量的文本分类器。在得出文档向量后，分类器本身的训练和测试并不是特别复杂，类似于第一个步骤即可。

**３ 方案改进**

在整个方案的实施过程中，从基准模型到词向量，微调和改进的地方不少，主要有以下这些方面：

(1)改进基于tf-itf模型的分类器。我首先尝试了线性核的SVC分类器。在感觉到训练时间较长后，根据sklearn的官方文档[4]，又尝试了LinearSVC。意外的是，LinearSVC不仅大幅缩短了训练和预测的时间，而且还提升了分类性能，得到的结果如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 准确率 | 召回率 | F1 | 训练时间 | 预测时间 |
| SVC（Linear核） | 0.88 | 0.87 | 0.88 | 17.82秒 | 10.34秒 |
| LinearSVC | 0.90 | 0.90 | 0.90 | 0.28秒 | 0.005秒 |

可见，选用LinearSVC分类器几乎在各个方面都得到了更好的性能。

(2)改进词向量模型的参数。在训练词向量模型时，可以设置min\_count、sg等参数。其中sg决定了词向量模型使用的算法模型：CBOW或Skip-gram。在本项目中sg设置为了1，也就是选用Skip-gram模型，文本分类器的性能因此提升了5%。另外，min\_count决定了一个词至少需要出现多少次才会被词向量模型使用，设置的过小也会影响到分类器的性能。

(3)改进文档向量的生成方式。在得到词向量模型后，还需要得到每篇文档的文档向量。首先尝试了词向量相加再除以文档总词数的方式，即：

之后我查看了一些论文[5]，在论文的启发下尝试将tf和idf与词向量结合。经过多次尝试，发现在只使用idf的情况下，分类器性能不错。此时生成文档向量的方式如下：

经过这样的调整，分类器的性能提升了约3%。

**三、结果讨论**

**1 结果展示**

基准模型if-idf（features数取10000）和词向量方案的分类结果对比如下：

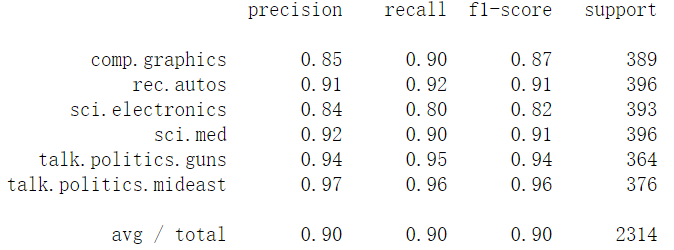
|  |  |  |  |
| --- | --- | --- | --- |
|  | 准确率 | 召回率 | F1值 |
| Tf-idf方案 | 0.90 | 0.90 | 0.90 |
| 词向量方案 | 0.90 | 0.90 | 0.90 |

由表格可见，在调整了词向量模型参数和文档向量的生成方式后，后者的分类性能同样达到了90%，已经有了不错的性能。另外，基于词向量方案的分类器在测试集和训练集上的测试效果如下：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 准确率 | 召回率 | F1值 |
| 训练集测试结果 | 0.94 | 0.94 | 0.94 |
| 测试集测试结果 | 0.90 | 0.90 | 0.90 |

分类器在训练集上的效果比在测试集上的效果好4%。这样的性能差并不大，可以认为该分类器并不存在明显的过拟合问题。

最后得到的词向量方案的分类器性能如下：



观察上图可以看到，分类器的总体F1值达到了0.9。其中，它在‘sci.electronics'上表现最差，在‘talk.politics.mideast'上表现最好。

**2 思考和展望**

根据本项目的实践结果，可以发现tf-idf模型在文本分类上依然有着可观的性能，使用起来简洁而高效。相比而言，词向量模型可以包含更为丰富的语义信息；可以避免文档向量维数过高的问题；同样有不错的文本分类性能。不过用它来做文本分类，需要根据词向量来生成文档向量。如何生成合理的文档向量，是必须要面对的问题，本项目中的方案只是做了一个尝试，还有更多不同的方案可以选择和探索。

另外对于文本分类的问题，除了本项目中的尝试外，还有不少其他的可行方案。比如基于Doc2Vec的文本分类，基于LDA模型的文本分类等。如果想要对文本分类问题有更深入更全面的了解，还需要在以后进行更多的尝试和研究。

**参考资料和文献**

[1] https://www.cnblogs.com/sxron/p/7742692.html

[2] 宗成庆 《统计自然语言处理》（第2版） 2003

[3] Robert Layton 杜春晓《Python数据挖掘入门与实践》2016

[4] http://scikit-learn.org/

[5] http://www.docin.com/p-1715862349.html 基于Word2Vec的一种文档向量表示 （计算机科学 2016.06）