优达学城毕业项目——文本分类

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之一。本项目所要做的工作，就是运用机器学习，尝试解决文本分类问题。

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，在主题无关和主题近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，就是要训练一个分类器，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，一共存在 20 个不同的新闻主题，所以这是一个典型的多分类问题。而每条新闻对应的主题都是已知的，因此考虑训练一个基于有监督学习的分类器，来解决该问题。

基于以上的基本分析，本文的工作主要分为两部分：特征提取和训练一个有监督分类器。提取到有效的文本特征，是保证分类器性能的前提和关键；而选择合理的分类器，也能进一步提升分类的效果。在本文中，将尝试使用两种不同的方法提取特征，分别为 tf-idf 和 word2vec；在提取到特征后，再使用合适的分类器进行文本分类；最后，对分类的结果和性能做对比分析。

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：

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP})$$

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：

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN})$$

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：

$$\text{F1} = 2 * \text{precision} * \text{recall}/(\text{precision} + \text{recall})$$

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。在分类性能相差不大情况下，运算时间越短的分类器越会得到青睐。

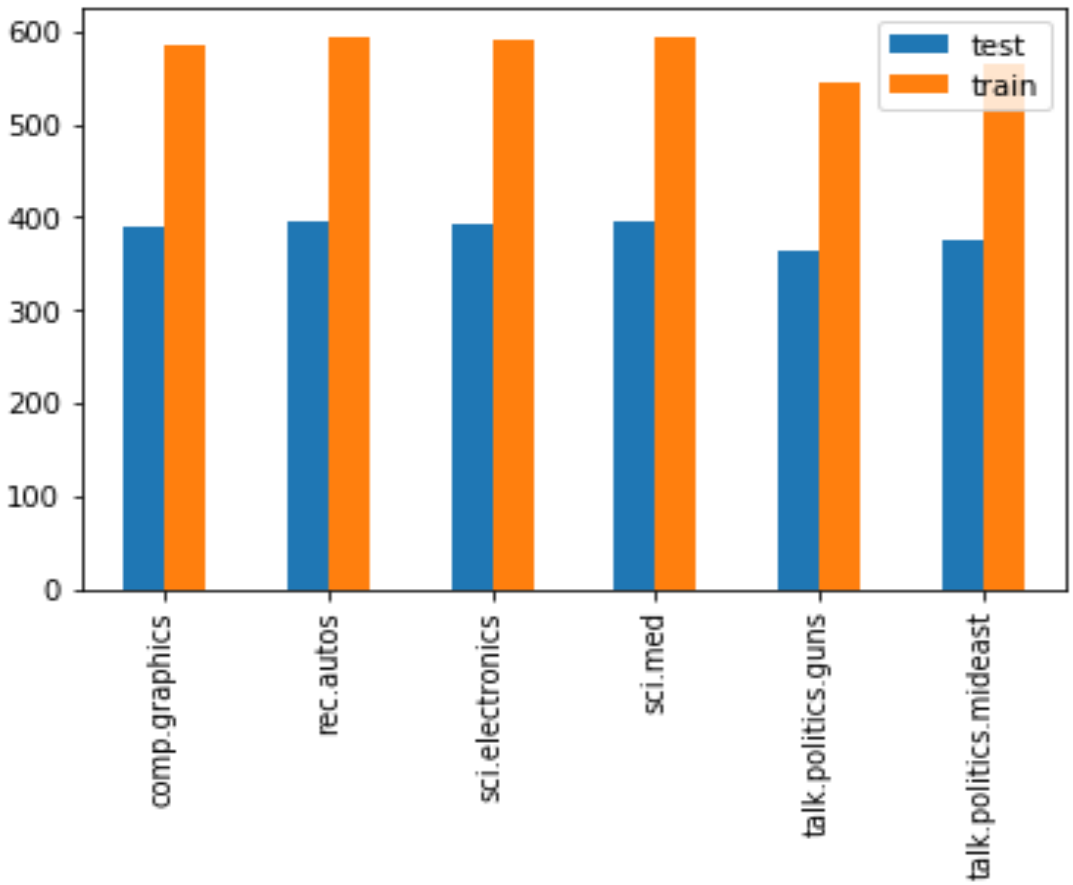
二、问题分析

1 数据集分析

(1) 总体分布

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

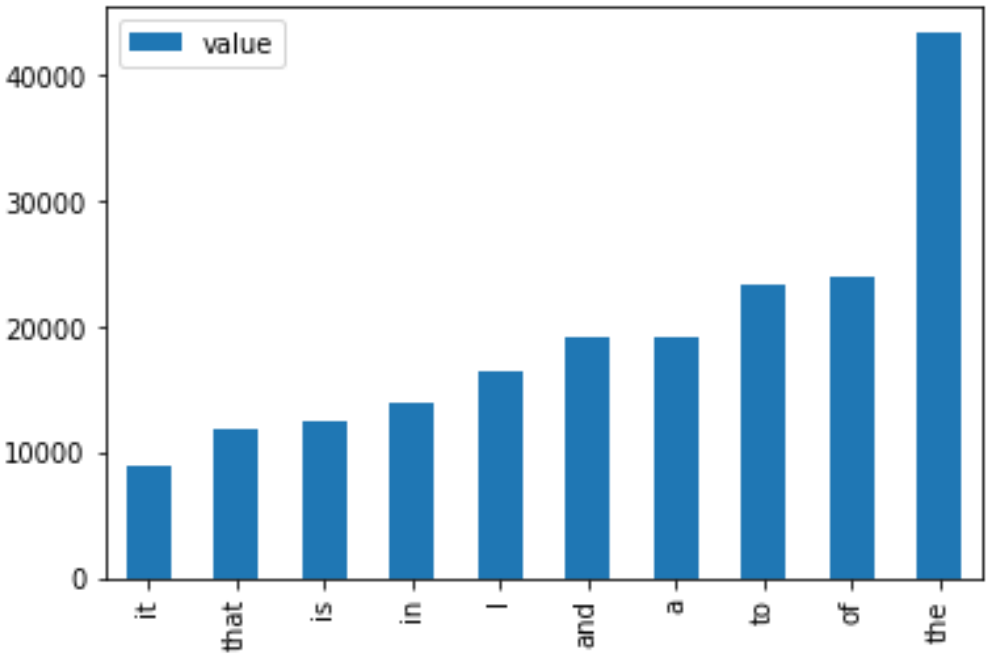


观察可知，训练集中每类新闻的数量大体相等，测试集中也是如此。同时，训练集中数量略多的类别在测试集中也数量略多，训练集中数量略少的类别在测试集中同样数量略少。这说明训练数据中不存在类别不平衡的问题，并且训练集和测试集中的数据是服从同一个分布的。

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无关时的分类效果，还能测试主题可能相关时的分类效果。

(2) 高频词

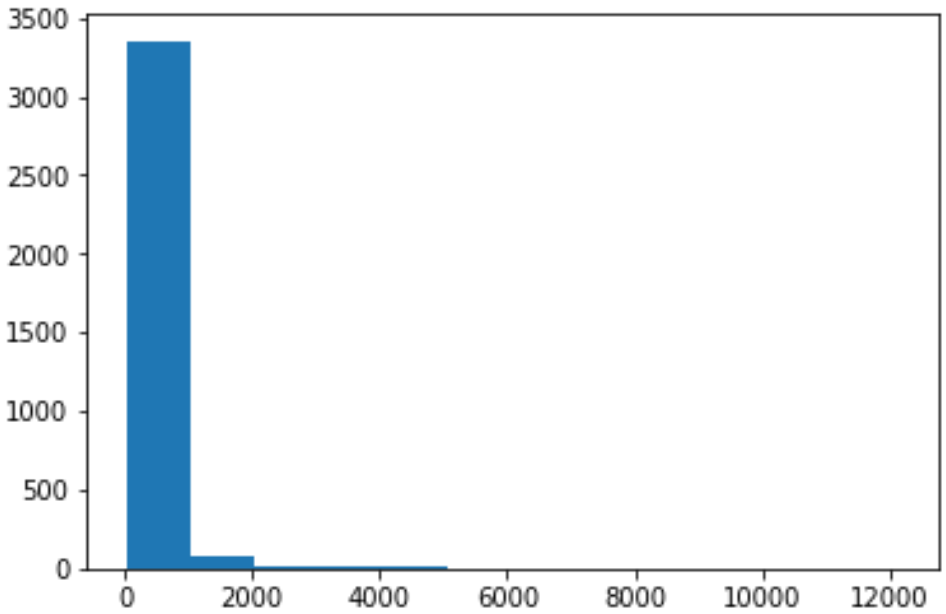
接下来，在所选的六类数据的测试集中统计词频，看看高频词能否较好的用来区分文本。在去掉标点后，统计使用频率最高的 10 个词语，得到的结果如下图：



可以发现以上词汇都属于英文停用词，非常常用却无法提供足够有用的信息，对文本分类并没有实际的作用。因此在数据预处理的过程中，文本将去除这些词而只保留真正有用的词语。

(3) 文本长度

在去除标点后，对测试集中文本的长度做了统计，得到结果如下：最短文本长度为 18 个词，最长文本长度为 12172 个词，平均值为 321 个词。具体分布如下图：



大部分文件的长度都小于 1000 个词，而绝大部分文件的长度都小于 2000

个词（不包括标点）。在此对较短的文本进行观察，看看是否有必要删除这些数据。下图为一个词数很少的文件的全部内容：

```
From: hl7204@eehp22 (H L)
Subject: Re: Graphics Library Package
Organization: University of Illinois at Urbana
Lines: 2
```

可以发现它没有任何正文，但由于存在 Subject 等信息，因此依然可找到用于区分文本类别的词语（subject 中的 graphics）。很短的文本也可能可以被区分，鉴于这样的情况，本文没有再去删除那些长度很小的文本个体。

(3) 其他

观察具体的文本后可知，除了之前提到的问题外，还存在着一些其他需要处理的细节，比如统一词语的大小写等，这些工作都应该在数据预处理步骤中完成。

2 算法介绍

在本项目中使用了两种不同的文本特征提取方法，以及一种有监督分类方法，分别介绍如下：

(1)tf-idf：

tf-idf 特征包含了两个因子：tf 和 idf。其中，tf 统计了每个文件中每个单词出现的次数[3]。而 idf 称为逆向文件频率，表征了一个词在类别区分上的重要性，其计算公式如下：

$$\mathrm{idf_i} = \log \frac{\text{文档总数}}{\text{包含词 i 的文档数} + 1}$$

在计算出所有的 idf 值后，再将文本中每个词的 tf 值和其 idf 值相乘，进而得到一个向量。该向量便可用来表征文本的特征，并用于分类工作了。

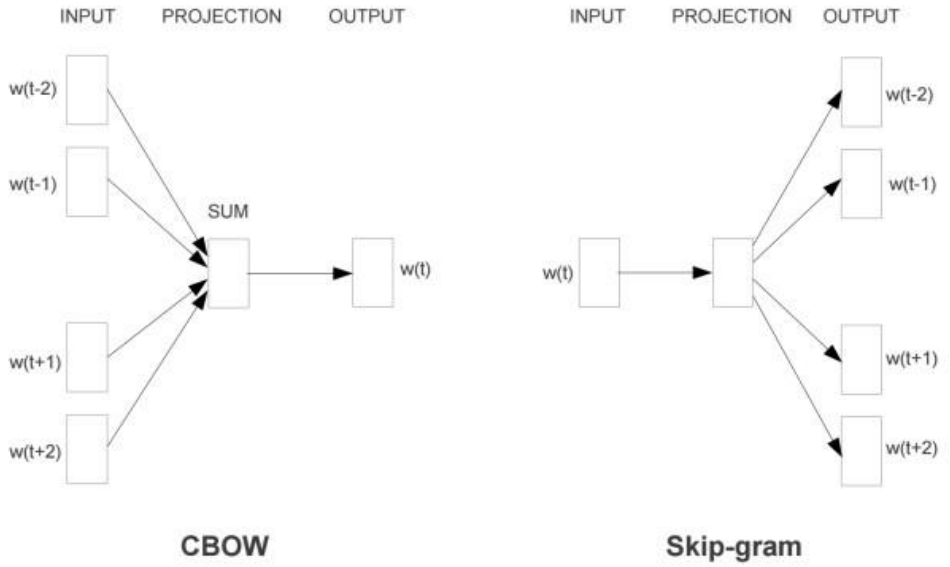
另外，除了统计单个词的频率外，还可以统计 n 个连续词组成的子序列的频率，即 n 元语法。当 n 大于 1 时，相比统计单个单词，它能更好的抓取长文本的某些细化特征，比如写作的惯用手法、习惯短语等。

不过在本项目中，n 的取值依然为 1。原因是本数据集中的文本大多数是短文本，不是长文本，并不适合将 n 设大。另外本项目主要是根据主题分类，不用过多涉及一些更细化的文本特征。

(2)word2Vec:

word2Vec 的基本思想，是把自然语言中的每一个词都表示成一个统一意义统一维度的稠密短向量[4]。然后使用该短向量，便能轻松实现单词间的语义比较、聚类分析等工作。

要把词转化为短向量，需要使用到神经网络的相关技术，具体主要有 Skip-Gram 和 CBOW 两种模型。这两种模型的特点如下图所示[5]：



无论是 CBOW 还是 Skip-Gram，其每个输入都是一个类似 one-hot 的长词向量。不同的是上图中 CBOW 有多个输入一个输出而 Skip-Gram 有一个输入多个输出。这就是这两个模型的核心不同点：CBOW 模型根据上下文预测当前单词，而 Skip-Gram 根据当前单词预测其上下文。可以发现这里也有 n-gram 的思想，周围的 n 个单词即为当前单词的上下文。

在本项目中，使用了 gensim 来完成词向量的训练。查看相关文档可知[6]，只需修改 sg、window 等参数，便可方便的实现切换 CBOW 和 Skip-Gram 模型、修改上下文的长度等训练需求。

在获取到所有的词向量后，还需要使用适当的方法提取表示文档的短向量。在此做了两种尝试，一种是直接求和后再求均值；另一种是基于 idf 给每个词向量赋予权重，然后再求和求均值。

(3)SVM:

在获取到文本的有效特征后，下一步便是进行数据训练了。在此，我选择了支持向量机作为本项目的分类器。它是一种经典的分类算法，通过定义和计算间隔、构造划分平面，进而实现对数据的二分类。在此基础之上，结合一对多或层次分类的思想，便可将 SVM 应用到多分类任务上。

选择它的原因是它能较好的应对高维空间，并且鲁棒性不错，适合处理

当前的文本分类问题。

3 基准测试

在本项目中，我一共使用了 6 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%。这个准确率显然太低了，无法作为基准值使用。因此继续查看其他资料[7]，发现在进行 20 分类问题的处理时，大部分分类器的准确率都能达到 0.8 左右，而性能较好的分类器，其准确率能达到 0.85 左右。考虑到我仅仅是处理 6 分类问题，因此我把准确率的基准值定为了 0.85，即本项目的分类器的总体准确率必须达到 0.85 以上。另外，分类器不仅要有不错的准确率，还应该有不错的召回率。因此，最后把模型的召回率和 F1 值的基准值也都定为了 0.85。

另外，是否发生过拟合也是需要关注的重点。因此在得到结果后，需要对比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

良好的数据预处理，不仅能为之后的工作提供必要的数据，而且还有助于提升整个模型的性能，因此也是很重要的一步。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，我总共对数据做了以下五种处理：

(1)统一单词的大小写，将所有的大写字母转化为了小写。

(2)分词并使用正则表达式删除标点符号。

(3)删除了所有停用词。

(4)删除了过短的单词，如果长度小于 2，则删除这些词。

(5)删除分词后不包含任何字母的词，这些词一般也不具备和主题相关的信息。

其中 1 到 3 步骤是建立初始词向量模型时便已使用的；而 4 和 5 步骤是在之后优化模型的过程中添加的，起到了改善模型性能的效果。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

本项目的整个实施过程分可为以下四个步骤：

(1)基准 tf-idf 模型训练和测试文本分类器。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。其中，n_features 设置为了 20000,即文档向量的大小为 20000 维；ngram 设置为了 1（原因在之前的算法介绍中提及）；其他参数都使用了默认值。在得到向量后，便可训练、调优和测试分类器了。

(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，给予预处理过的训练数据，便可得到一个词向量的模型了。其中，我将 min_count 设置为了 3，即出现 3 次或以上的词才会具有词向量；并分别尝试了 COW 和 Skip-Gram 模型（通过 sg 参数）；而其他参数都使用了默认值。在使用默认值的情况下，词向量的维数为 100 维。在训练得到词向量模型后，可以直接从模型中获取所有词的词向量。

(3)根据词向量生成文档的向量。本文做了两种不同的尝试，对于求和再求均值的方式，可直接计算得到文档的向量。而对于结合 idf 的方式，需要首先统计训练集中每个词的 idf。无论是训练集还是测试集，在生成文档向量时，都统一使用这个 idf 作为词向量的权重。无论使用哪种方法，由于之前的词向量维度只有 100 维，因此此处得到的文档向量也仅有 100 维。

(4)基于文档向量训练和测试文本分类器。在得到文档向量后，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,感觉训练时间较长。然后根据 sklearn 的官方文档[8]，又尝试了 LinearSVC。发现 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）	0.89	0.88	0.89	8.19 秒	4.47 秒
LinearSVC	0.91	0.91	0.91	0.14 秒	0.002 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)改进数据的预处理。基于数据分析阶段的大体结果，一开始我只对分词后的

数据做了大小写转换、删标点符号和删停用词的处理。之后查看生成的词向量模型，发现仍存在大量的无分类意义的词。因此我又对预处理工作做了一些改进，包括：删除过短的词（长度小于 2）、删除不包含字母的词（比如纯数字、纯下划线等）。之后重新生成词向量模型，在其他条件不变的情况下，分类器的性能因此提升了约 2%。

(3)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至少 5%。另外，min_count 决定了一个词至少需要在训练集出现多少次，才会具有词向量，设置的过小或过大也会影响到分类器的性能。

(4)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：

$$\text{文档向量} = \frac{\sum \text{词向量}}{\text{文本长度}}$$

之后我查看了一些论文[9]，在论文的启发下尝试将 tf、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：

$$\text{文档向量} = \frac{\sum(\text{词向量} * \text{idf})}{\sqrt{\sum \text{idf}^2}}$$

经过这样的调整，如下表所示，分类器的性能提升了约 3%：

	准确率	召回率	F1 值
直接求均值方案	0.87	0.87	0.87
基于 idf 的方案	0.90	0.90	0.90

三、结果讨论

1 结果展示

(1)性能结果

对于基于词向量的 LinearSVC 分类器，使用 GridSearchCV 对参数 C 进行了调整。C 是一个惩罚参数，可对训练中发生的错分，调节惩罚的强度。C 过小时会引起欠拟合，而过大时会引起过拟合。最后的搜索结果为 C=1，此时分类器性能如下：

	precision	recall	f1-score	support
comp.graphics	0.85	0.90	0.87	389
rec.autos	0.91	0.92	0.91	396
sci.electronics	0.84	0.80	0.82	393
sci.med	0.92	0.90	0.91	396
talk.politics.guns	0.94	0.95	0.94	364
talk.politics.mideast	0.97	0.96	0.96	376
avg / total	0.90	0.90	0.90	2314

观察上图可知，分类器的总体 F1 值达到了 0.9。它在 sci.electronics 上表现最差，而在 talk.politics.mideast 上表现最好。对于具有相关主题的子类，比如 talk.politics.guns 和 talk.politics.mideast、sci.electronics 和 sci.med，分类器的性能并没有表现出变差的迹象。

(2)错误分析

在此，寻找那些分类错误的数据，尝试做一些分析。首先从总体分布着手，寻找一些规律性的趋势。统计发现，发生分类错误的文本的平均长度为 216 个单词，而测试集文本的平均长度为 324 个单词，可见相对而言较短的文本更可能发生分类错误。

其次由个体文本入手，寻找具体的错误原因。第一个发生错误的文本的主要内容截图如下：

```
Subject: **** And now serious: E-Magazine *****
Summary: How about starting a group where scientific articles can be pre-publish
Keywords: Scientific papers, Electronic magazine
Nntp-Posting-Host: duteela.et.tudelft.nl
Organization: Delft University of Technology, Dept. of Electrical Engineering
Lines: 22


For some time I've been thinking about the possiblity of starting a group
where scientific articles can be published (or perhaps just summaries).
Possible advantages would be:
* Free disribution
* Fast acceptance
* Online discussion between authers and readers

This would be possible with one group with a moderator for publishing the
articles and one perhaps without for discussion.

The best thing would be if all the articles would be in a standard format which
would make it possible to print or view the documents camera ready. Perhaps
Postscript or Rich Text Format?
```

观察可以发现，scientific、electronic 可以说是该文本的重要单词。但奇怪的是，它并没有被分入 sci.electronics，而被错误分到了 sci.med。因此对

这两个单词做进一步的测试，发现当仅保留 scientific 和 electronic 两个单词时，该文本依然被归为了 med。而当文本中只保留 electronic 时，它才会被归类到 electronics。可以认为，模型在此处的分类行为是存在瑕疵的。可能也正是类似的原因，导致 sci.electronics 的召回率成为了所有数据中最差的那个。

第二个发生错误的文本的主要内容截图如下：

```
Subject: VASCAR
Organization: Freshman, Biology, Carnegie Mellon, Pittsburgh, PA
Lines: 16
NNTP-Posting-Host: po3.andrew.cmu.edu

I know this is the wrong place to post this, but I couldn't find any
relevant newsgroups in my area.

For those of you who are from PA, where is VASCAR (where the cops
measure your speed from the time it takes you to cross the distance
between two white lines on the road, right?) most commonly used?  I'm
especially interested in the Pittsburgh area (specific locations, prior
experiences, if possible).  For those PA and non-PA, if they use VASCAR
in your state, is it most common in rural, city, highway areas, etc.
What I'm interested in mainly is where I can speed with the least risk
of being caught.  You can always detect radar, but there's no way to
fight VASCAR unless you know where all the white lines are.
```

可以说，VASCAR 是该文本的一个重要单词。不过通过查阅发现，VASCAR 和 vascar 并不是同一个单词，前者和汽车相关而后者和录音机相关。也就是说，大小写转换在此时起到了反作用，把两个不同的词混淆为了同一个词。最后，本该归类为 autos 的该文本，被错归为了 electronics。

由以上两个个体的错误分析大体可知，一方面模型还存在进一步改进的空间；而另一方面一些特殊的个体问题也会致使文本的分类发生错误。

(3)过拟合情况

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略好，但性能差距并不大，可以认为该分类器并不存在明显的过拟合问题。

(4)结果对比

最后，对比一下 tf-idf 方案和词向量方案的分类性能：

	准确率	召回率	F1 值
tf-idf 方案	0.91	0.91	0.91
词向量方案	0.90	0.90	0.90

由表格可见，这两个方案的结果都超出了基准值 0.85，达到了预期的目标。另外，后者在调整了词向量模型的参数和文档向量的生成方式后，各个指标都已接近 tf-idf 方案的结果。可见基于词向量进行文本分类，也是一个完全可行的方案。

2 思考和展望

通过本项目，我主要的收获有以下几点：

(1)实践了传统的 tf-idf 模型在文本分类上的使用，结合 sk-learn，使用起来简洁而高效，并且性能不错。

(2)通过 gensim，实践了词向量模型的训练。

(3)为了将词向量和传统分类器结合起来使用，我初步探索了通过词向量生成文本向量的方式，此时生成的文档向量维数（100 维）远低于 tf-idf 的文档向量维数（20000 维）。

对于文本分类的问题，本项目只是做了一定程度上的尝试，进一步可以深入的地方还有不少，主要有以下几个方面：

(1)虽然 SVM 在本项目上有着不错的效果，但也还可以尝试其他的有监督分类器。

(2)本项目中，使用了词向量和有监督分类器结合的方式进行文本分类。其实词向量也可以结合深度神经网络来使用，本文没有做这方面的探索。

(3)最后，本文中用来训练词向量模型的语料还是比较少的，仅使用了 comp.graphics、rec.autos 等 6 类新闻组的测试数据，如果使用更多的语料或许会有更好的效果。

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] https://blog.csdn.net/sangyongjia/article/details/52440063

[4] http://www.sohu.com/a/128794834_211120

[5] https://www.leiphone.com/news/201706/PamWKpfRFEI42McI.html 一文详解 Word2Vec 之 Skip-Gram 模型

[6] https://radimrehurek.com/gensim/models/word2vec.html

[7] https://blog.csdn.net/yangliuy/article/details/7400984

[8] http://scikit-learn.org/
[9] 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）