I. 问题的定义

项目概述

自然语言处理（后面简称 NLP）是机器学习技术重要应用范畴之一，从手机上的智能语音助理如 Siri，到移动、联通的自动语音服务，再到具有理解、推理能力的 IBM Waston，最近亚马逊也相应推出了可提供高级语音识别及自然语言理解功能的 Lex，这些都是自然语言处理技术应用前沿产品实例。 试想，如果有朝一日人类完全解决自然语言处理瓶颈，实现计算机对自然语言完全理解、分析，那么出现在科幻片如《机械姬》、《西部世界》里面的机器人与人类无障碍沟通及情感交流的情景很可能出现。

但是现实中自然语言处理技术层面还面临诸多挑战，其中之一就是词、语句以及文章的表达。统计语言处理里面，利用符号来描述概率模型，比如 ngram 模型 。对句子或者文章的表示，可以采用词袋子模型。

近几年来，借助深度学习概念和性能强劲的硬件平台，Geofrey Hinton, Tomas Mikolov, Richard Socher 等学者深入开展了针对词向量的研究，进行了大量鼓舞人心的实验，将自然语言处理推向了新的高度。以词向量为基础，可以方便引入机器学习模型对文本进行分类、情感分析、预测、自动翻译等。

本项目目的就是利用上述自然语言处理技术结合所学机器学习知识对文档进行准确分类。

分类文本数据可以使用经典的 20 类新闻包，里面大约有 20000 条新闻，比较均衡地分成了 20 类，是比较常用的文本数据之一。既可以从官方网站下载，也可利用 sklearn 工具包下载，具体请参见说明文档。

此外，词向量的训练也需要大量数据，如果感觉 20 类新闻数据样本量不足以训练出较好的词向量模型，可以采用 Mikolov 曾经使用过的 text8 数据包进行训练。

问题陈述

如何使用机器学习算法进行文本分类。

首先面临的问题是如何对文档进行处理，比如词、语句以及文章的表达。统计语言处理里面，比较容易利用符号来描述概率模型，比如 ngram 模型 ，计算两个单词或者多个单词同时出现的概率，但是这些符号难以直接表示词与词之间的关联，也难以直接作为机器学习模型输入向量。

对句子或者文章的表示，可以采用词袋子模型，即将段落或文章表示成一组单词。使用词袋子模型来表示每篇文档，常见的一种思路是首先将文本进行分词，也就是将一个文本文件分成单词的集合，建立词典，每篇文档表示成特征词的频率向量或者加权词频 TF-IDF 向量，这样可以得到熟悉的特征表。接下来，就可以方便利用机器学习分类模型进行训练。

以词向量为基础，也可以方便引入机器学习模型对文本进行分类、情感分析、预测、自动翻译等。Mikolov、Socher 等人提出了 Word2Vec、GloVec 等词向量模型。用维数较少的向量表达词以及词之间的关联性。利用 Word2Vec 方式即词向量模型表示每篇文档，这里面包含两部分主要工作：

利用文本数据对词向量进行训练，将每个单词表示成向量形式。词向量训练后需要进行简单评测，比如检验一些单词之间相似性是否符合逻辑。

探讨怎样用文档中每个词的向量来表达整个文档。

然后分别在词袋子、词向量表达基础上采用机器学习模型对文本分类。

最后要达到对文本分类的预测的准确率要高，也就是能尽可能正确预测文本的分类。

评价指标

针对不同的目的，多种文本分类器性能评价方法被提出，包括召回率、精确率和 F1-score。设定 TP 表示分类器将输入文本正确分类到某个类别的个数；FP 表示分类器将

输入文本错误分类到某个类别的个数；FN 表示分类器将输入文本错误地排除在某个类别之外的个数；TN 表示分类器将输入文本正确地排除在某个类别之外的个数。

该分类器的准确率 A、召回率、精确率和 F1-score 分别采用以下公式计算：

准确率 A = TP / (TP + TN + FN + FP) * 100%

召回率 R = TP/ (TP+ FN) * 100%

精确率 P = TP / (TP + FP) * 100%

F1-score F = (2 * P * R) /(P + R)

由于在分类结果中，对应每个类别都会有一个召回率和精确率，因此，可以根据每个类别的分类结果评价分类器的整体性能，通常方法有两种：微平均和宏平均。微平均是根据精确率和召回率计算公式直接计算出总的精确率和召回率值。宏平均是指首先计算出每个类别的精确率和召回率，然后对精确率和召回率分别取平均得到总的精确率和召回率。不难看出，宏平均平等对待每一个类别，所以它的值主要受到稀有类别的影响，而微平均平等考虑文档集中的每一个文档，所以它的值受到常见类别的影响比较大。

我会使用宏平均和微平均作为项目的评估标准。

II. 分析

数据的探索

数据集可以利用 sklearn 工具包下载。数据集包含了经典的 20 类新闻包，里面有 18000 条新闻，比较均衡的分成了 20 类。

数据集被分成训练集和测试集，训练集包含 11314 个样本，测试集包含 7532 个样本。每个样本都有标签。下面将展示一个具体的文本。

From: halat@pooh.bears (Jim Halat)
Date: 1 Apr 93 21:24:35 GMT
Reply-To: halat@pooh.bears (Jim Halat)
Sender: news@bear.com

References: <16BA1E927.DRPORTER@SUVM.SYR.EDU>
Lines: 24

In article <16BA1E927.DRPORTER@SUVM.SYR.EDU>, DRPORTER@SUVM.SYR.EDU (Brad Porter) writes:
>
> Science is wonderful at answering most of our questions. I'm not the type
>to question scientific findings very often, but... Personally, I find the
>theory of evolution to be unfathomable. Could humans, a highly evolved,
>complex organism that thinks, learns, and develops truly be an organism
>that resulted from random genetic mutations and natural selection?

[...stuff deleted...]

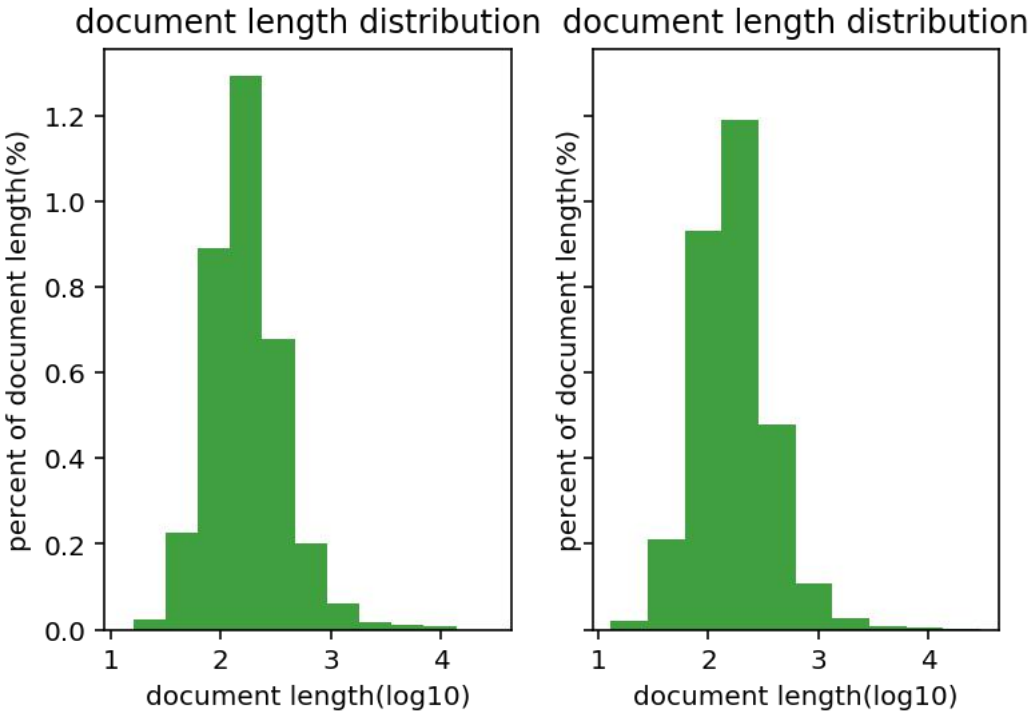
Computers are an excellent example...of evolution without "a" creator.
We did not "create" computers. We did not create the sand that goes
into the silicon that goes into the integrated circuits that go into
processor board. We took these things and put them together in an
interesting way. Just like plants "create" oxygen using light through
photosynthesis. It's a much bigger leap to talk about something that
created "everything" from nothing. I find it unfathomable to resort
to believing in a creator when a much simpler alternative exists: we
simply are incapable of understanding our beginnings -- if there even
were beginnings at all. And that's ok with me. The present keeps me
perfectly busy.

-jim halat

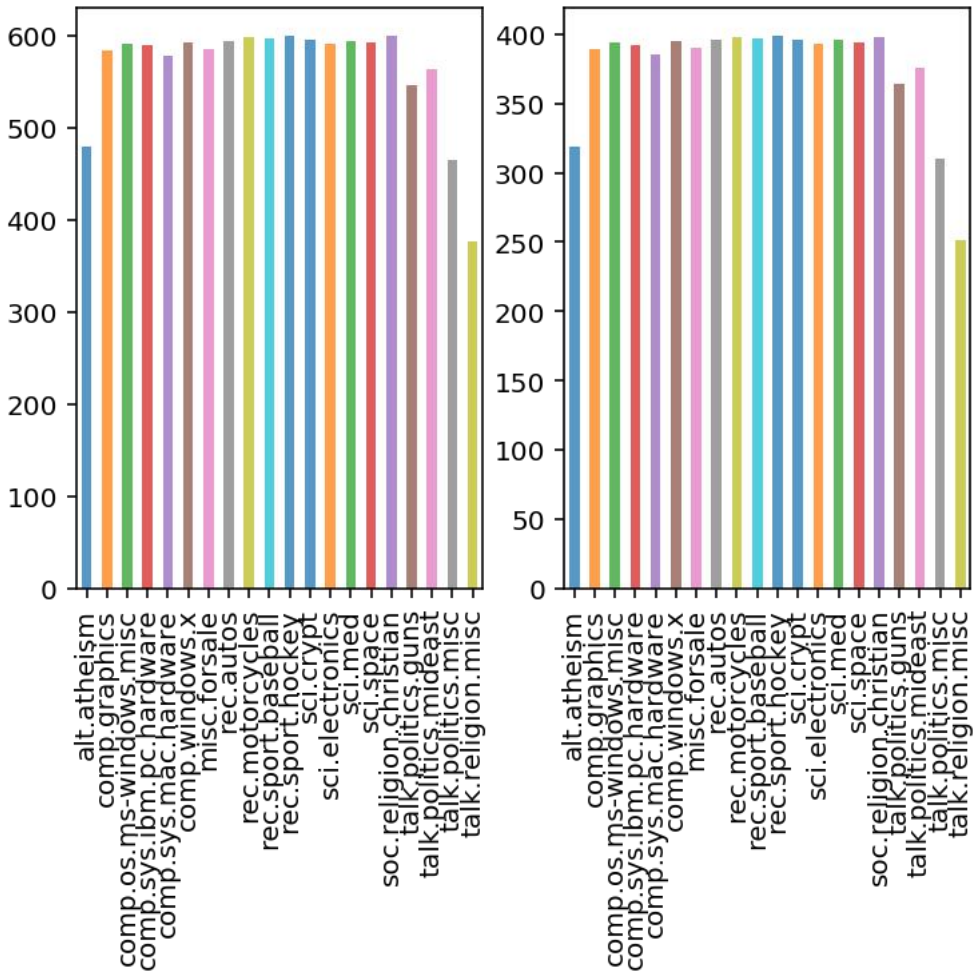
首先表明来自哪（halat@pooh.bears (Jim Halat)），主题（Re:There must be a creator! (Maybe)），消息 ID（ <30066@ursa.bear.com> ），参考文献（<16BA1E927.DRPORTER@SUVM.SYR.EDU>），日期（1 Apr 93 21:24:35 GMT），正文行数（24），最后就是正文内容。

很明显，有些词语对于文本分类是很有用的，比如 evolution，random genetic mutations，natural selection 等。一些标点，数字对文本分类也没有用处，也应该删除。

探索性可视化



文档包含的单词有几个也有几万个，集中在几百到几千。



每个类型的文档分布比较均匀，训练集和测试集都是。

算法和技术

数据预处理

去除标点符号、数字和特殊字符。

单词全部转换成小写。

探索文本表示的方式

使用词袋子模型来表示每篇文档，常见的一种思路是首先将文本进行分词，也就是将一个文本文件分成单词的集合，建立词典，每篇文档表示成特征词的频率向量或者加权词频 TF-IDF 向量，这样可以得到熟悉的特征表。接下来，就可以方便利用机器学习

分类模型进行训练。如下面所示意：

	She	He	loves	cats	dogs	too
"She loves cats."	1	0	1	1	0	0
"He loves cats too."	0	1	1	1	0	1
"She loves dogs."	1	0	1	0	1	0

利用 Word2Vec 方式即词向量模型表示每篇文档，这里面包含两部分主要工作：

利用文本数据对词向量进行训练，将每个单词表示成向量形式。词向量训练后需要进行简单评测，比如检验一些单词之间相似性是否符合逻辑。

word2vec 可以利用两种模型架构中的任何一种来产生分布式的词语表达：连续词袋（CBOW）或 skip-gram。

在 CBOW 中，模型从周围环境词的窗口中预测当前词。 上下文词语的顺序不影响预测（词袋假设）。

在 skip-gram 结构中，模型使用当前词来预测环境词的周围窗口。 skip-gram 体系结构权衡附近的上下文词语比更远的上下文词语更重。CBOW 速度更快，而 skip-gram 速度更慢，但对于不常见的词语做得更好。[18]

探讨怎样用文档中每个词的向量来表达整个文档。采用文档的词向量平均值作为文档的向量。

文档向量 Doc2Vec

Word 可以表示成向量形式，文档也可以。

段落向量，一个无监督的学习连续分布向量的框架文本片段的表示。 从句子到文档，文本可以是可变长度。

Doc2vec 向量表示被训练成是用于预测段落中的单词。 更确切地说，从一个段

落中，将段落向量与几个词向量连接起来，去预测给定的上下文中接下来的单词。 词向量和段向量都是通过随机梯度下降和反向传播进行训练。段落向量是在段落中独一无二，词向量是共享的。 在预测时间，段落向量通过修复来推断单词向量和训练新的段落向量直到收敛。[14]

Doc2Vec 有两种实现：

dbow(distributed bag of words)

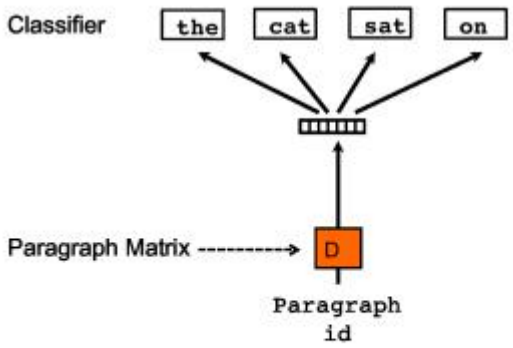


Figure 3. Distributed Bag of Words version of paragraph vectors. In this version, the paragraph vector is trained to predict the words in a small window.

dm(distributed memory)

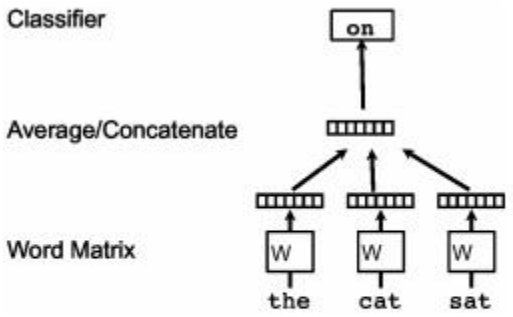


Figure 1. A framework for learning word vectors. Context of three words ("the," "cat," and "sat") is used to predict the fourth word ("on"). The input words are mapped to columns of the matrix W to predict the output word.

分别在词袋子、词向量表达基础上采用适当的模型对文本分类，优化模型并分析其稳健性。

决策树模型

决策树是一个预测模型；他代表的是对象属性与对象值之间的一种映射关系。树中每个节点表示某个对象，而每个分叉路径则代表的某个可能的属性值，而每个叶结点则对应从根节点到该叶节点所经历的路径所表示的对象的值。决策树仅有单一输出，若欲有复数输出，可以建立独立的决策树以处理不同输出。

决策树易于理解和解释，可以可视化分析，容易提取出规则。

可以同时处理标称型和数值型数据。

测试数据集时，运行速度比较快。

ID3（Iterative Dichotomiser 3）由 Ross Quinlan 于 1986 年开发。该算法创建一个多路树，为每个节点（即以贪婪的方式）找到将为分类目标产生最大信息增益的分类特征。树木生长到最大尺寸，然后通常采用修剪步骤来提高树的推广能力，以推广到看不见的数据。

C4.5 是 ID3 的继承者，并通过动态定义将连续属性值划分为离散的一组间隔的离散属性（基于数值变量），消除了要素必须分类的限制。C4.5 将训练树（即 ID3 算法的输出）转换成若干组 if-then 规则。然后评估每个规则的这些准确性，以确定它们应用的顺序。如果规则的准确性没有提高，修剪通过移除规则的先决条件来完成。

CART（分类和回归树）与 C4.5 非常相似，但它不同之处在于它支持数值目标变量（回归）并且不计算规则集。CART 使用在每个节点处产生最大信息增益的特征和阈值构造二叉树。

scikit-learn 使用 CART 算法的优化版本。

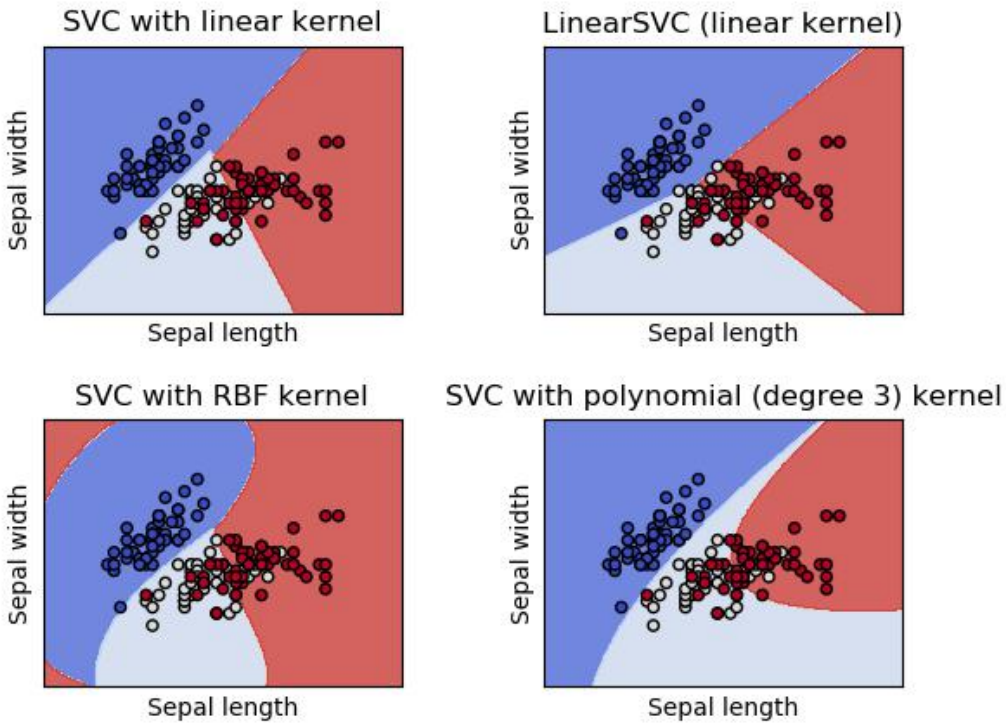
支持向量机模型

SVM 方法是通过一个非线性映射 p，把样本空间映射到一个高维乃至无穷维的特征空间中（Hilbert 空间），使得在原来的样本空间中非线性可分的问题转化为在特征

空间中的线性可分的问题．简单地说，就是升维和线性化．升维，就是把样本向高维空间做映射。

可以很好的处理高维数据集。

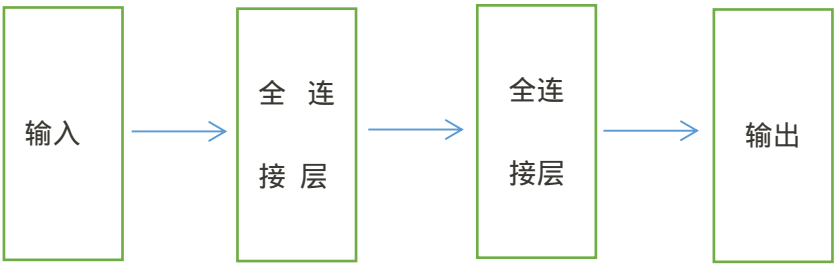
泛化能力比较强。



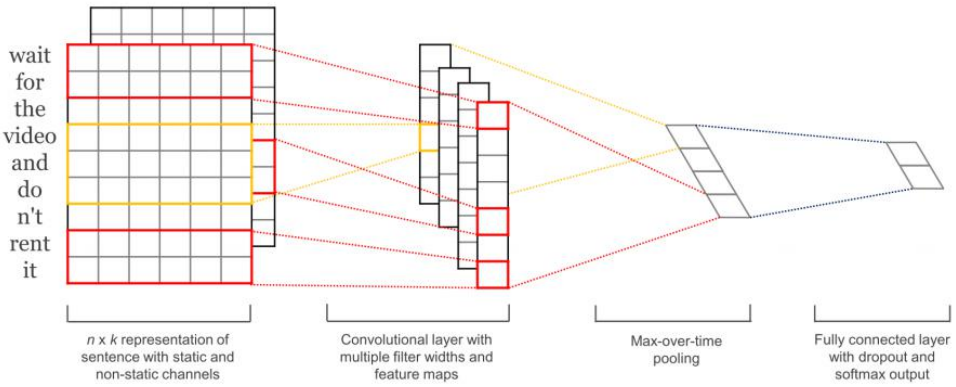
神经网络模型

分类准确度高，学习能力极强。

对噪声数据鲁棒性和容错性较强。



textCNN



第一层将单词嵌入到低维矢量中。 下一层使用多个不同大小的滤波器对嵌入的单词向量执行卷积。 例如，一次滑动 3，4 或 5 个单词。 接下来，我们将卷积层的结果最大化为长特征向量，添加丢失正则化，并使用 softmax 层对结果进行分类。

基准模型

斯坦福大学的一个小组使用斯坦福分类器对文本进行分类，他们在 20 个类别上的微平均 F1_score 在 81.7%左右，宏平均 F1_score 在 81.2%左右。

他们主要是通过正则表达式去分词，比如去除空格，或者单个字母，或者美元符，或者百分之数字。然后来提高文档分类的准确率。

III. 方法

数据预处理

使用的数据是 20 类新闻包，里面大约有 20000 条新闻，比较均衡地分成了 20 类。这些数据是文本数据，包含了标点符号和数字，需要进行处理。还有就是单词大小写。

对于数字和标点符号，直接去掉，不考虑他们的意义。

对于单词大小写，不考虑区分单词大小写，直接转换成小写。

将 20 类已经分成训练集和测试集的数据，分别分成特征和类别，特征就是文档的内容，

类别就是目录名。

然后对文档的内容进行处理，比如去掉数字，去掉一些停用词，把非字符数字作为分词，单词转化为小写。

数据预处理前：

[b"From: mattf@cac.washington.edu (Matthew Freedman)\nSubject: Non-Roman Font Availability\nArticle-I.D.: shelley.1rmgleINNa0g\nDistribution: world\nOrganization: U.W. Information Systems\nLines: 16\nNNTP-Posting-Host: elvis.cac.washington.edu\n\nCan anybody tell me anything about the availibility of non-Roman fonts\nfor X-Windows? Especially Unicode and/or han idiographic fonts.\n\nAlso, how about conversion tools for getting PC/Macintosh fonts into a\nformat suitable for X? I would assume it is not too difficult for\nbitmap fonts.\n\nThe FAQ's for this group and comp.fonts are not very helpful on these\nquestions. \n\n-=-\n= Matthew M. Freedman =\n= U. of Washington Information Systems mattf@cac.washington.edu =\n= 4545 15th Ave. NE; 3rd Floor (206) 543-5593 =\n= Seattle, WA 98105 =\n-=-\n"]

数据预处理后：

['from', 'afielden', 'cbnewsb', 'cb', 'att', 'com', 'andrew', 'j', 'fielden', 'subject', 'x', 'benchmarks', 'keywords', 'benchmark', 'organization', 'at', 't', 'lines', 'we', 'are', 'process', 'evaluating', 'x', 'terminals', 'this', 'includes', 'running', 'xremote', 'over', 'a', 'serial', 'line', 'i', 'would', 'like', 'run', 'some', 'x', 'benchmarks', 'determine', 'comparative', 'performance', 'has', 'anyone', 'written', 'any', 'such', 'benchmarks', 'or', 'know', 'any', 'useful', 'programs', 'on', 'net', 'i', 'heard', 'a', 'program', 'called', 'xstone', 'but', 'i', 'couldn', 't', 'locate', 'it', 'using', 'archie', 'please', 'reply', 'afielden', 'mlsma', 'att', 'com', 'as', 'i', 'don', 't', 'get', 'read', 'this', 'newsgroup', 'much', 'thanks', 'advance', 'any', 'help', 'andrew', 'fielden', 'at', 't', 'network', 'systems', 'uk', 'tel', 'information', 'systems', 'group', 'sun', 'support', 'email', 'afielden', 'mlsma', 'att', 'com']

Tfidf 文档库

基于 20 类新闻数据样本，使用 gensim 的 Dictionary 生成词典，使用 Dictionary 中的 doc2bow 创建词袋子模型，最后生成 tfidf 文档库。

词典中有 107010 个单词。

Word2vec 词向量

Word2vec 模型的准备，使用了 gensim 中的 word2vec，基于 20 类新闻数据样本生成了词向量模型。

Word2vec 主要有以下参数：

大小（size），特征向量的维度，设置的是 200 维。

窗口（window），当前单词与预测单词之间的距离，设置的是 5。

最小数（min-count），设置最低频率，默认是 5，如果一个词语在文档中出现的次数小于 5，那么就会丢弃，设置的是默认值 5。

处理器（workers）：是训练的进程数，默认是当前运行机器的处理器核数，设置的是 4。

文档的表达

Word2vec 词向量的平均值

文档表达为 word2vec 词向量的平均值。

Doc2vec

生成文档向量，使用 gensim 中的 doc2vec，生成训练、验证和测试集。

执行过程

决策树

使用 tfidf 来分词，在这之前，先将 tfidf 转换为稀疏矩阵。这里选择调节了两个参数 ’max_depth’ 和 ’min_samples_split’, 使用网格搜索，最终最佳参数是 {'max_depth': 90, 'min_samples_split': 40}.

平均词向量，取文档中单词词向量的平均值作为文档的向量表达。这里选择调节了

两个参数’max_depth’和’min_samples_split’,使用网格搜索,最终最佳参数是{'max_depth': 120, 'min_samples_split': 40}.

Doc2vec。这里选择调节了两个参数’max_depth’和’min_samples_split’,使用网格搜索,最终最佳参数是{'max_depth': 125, 'min_samples_split': 45}.

支持向量机

使用 tfidf 来分词，在这之前，先将 tfidf 转换为稀疏矩阵。这里主要调节了’C’这一个参数,最佳参数是{'C': 2}.

平均词向量，取文档中单词词向量的平均值作为文档的向量表达。

Doc2vec。

神经网络模型

全连接网络

平均词向量，取文档中单词词向量的平均值作为文档的向量表达。

Doc2vec。

测评

使用了 sklearn 中的 f1_score 函数。

完善

关于词向量的训练，我直接使用 20 类新闻数据样本去训练的，相似性检测结果表明，训练的效果不够好，直接影响了后面的文档分类效果。我可以尝试使用 text8 数据包进行训练。

IV. 结果

模型的评价与验证

决策树（F1 Macro/F1 micro）

	训练集	验证集	测试集
tfidf	**0.9999/0.9999**	**0.6353/0.6398**	**0.5457/0.5512**
Word2vec200	**0.9999/0.9999**	**0.2486/0.2510**	**0.2157/0.2179**
Doc2vec200	**1.000/1.000**	**0.1610/0.1648**	**0.1523/0.1539**

Tfidf 的训练集 f1_score 非常高，但是验证集只有 63%，测试集只有 54%，明显是过拟合了。

Word2vec200，Doc2vec200 也是同样情况。

支持向量机（F1 Macro/F1 micro）

	训练集	验证集	测试集
tfidf	**0.9932/0.9932**	**0.9163/0.9156**	**0.8252/0.8297**
Word2vec200	**0.5972/0.6097**	**0.5260/0.5368**	**0.4654/0.4766**
Doc2vec200	**0.8587/0.8582**	**0.5675/0.5700**	**0.5349/0.5447**

Tfidf 模型表现的非常好，测试集的 f1_score 是 82%，超过了基准模型 81%。

Word2vec200 的结果是 47%，在意料之中，首先在训练词向量之后，相似性测试中‘computer’和‘pc’的相似性仅为 37%，说明训练的数据不够多。其次使用词向量的平均值来表示整个文档，不够准确。

Doc2vec200 的训练集达到 85%，但是验证集和测试集只有 50%多，可能是过拟合了。

Doc2vec200

深度神经网络（F1 Macro/F1 micro）

	训练集	验证集	测试集
Word2vec200	**0.5902/0.6112**	**0.4883/0.5101**	**0.4460/0.4679**
Doc2vec200	**0.9516/0.9513**	**0.6333/0.6360**	**0.5848/0.5935**

和支持向量机情况一样，Word2vec200 很可能是词向量训练的数据不够，同时平均

值没法很好的表达整个文档。

Doc2vec200 的训练集分数非常高，但是验证集和测试集并没有基准模型高，很可能是过拟合了。

textCNN（F1 Macro/F1 micro）

	训练集	测试集
Glove	0.9875/0.9875	0.8316/0.8374

合理性分析

Svm，决策树和深度学习在 Word2vec 词向量的平均值上表现不好，首先是词向量的训练问题，其次平均值不能够很好的表达文档。

20 类新闻数据样本是按照时间划分训练集和测试集的，数据本身可能存在偏差。

V. 项目结论

对项目的思考

我已经总结了项目的整个流程。从一开始的数据预处理，到数据分析，到文档的表达，最后再进行分类。

Svm 和 tfidf 的组合在一开始就超过了基准模型，表现非常棒。

词向量训练不够的问题，过拟合的问题。

最终模型和结果符合我对这个问题的期望，有模型能超过基准模型。它可以在通用的场景下解决这些类型的问题，例如推荐系统，垃圾信息的检测。

需要作出的改进

词向量的训练可以完成的更好。

深度学习方面可以试试 LSTM 和 RNN。词向量方面尝试比如 GloVec。

如果将最终模型作为新的基准，可以在分词方面做的更好。

工具

建议使用的工具包：

gensim，可以方便快捷地训练 Word2Vec 词向量。

GloVec，可以用来训练 GloVec 词向量。

sklearn，功能强大的机器学习包，包含有常用的分类工具。

tensorflow，可以逐步定义词向量训练过程，也可以建立深度学习建模。

参考文献

1.维基百科，自然语言处理，https://en.wikipedia.org/wiki/Natural_language_processing

2.我爱自然语言处理，中英文维基百科语料上的 Word2Vec 实验，http://www.52nlp.cn/tag/word2vec

3.优达学城，机器学习毕业项目说明，https://github.com/nd009/machine-learning

4.牛津大学，自然语言处理和深度学习课程，https://github.com/oxford-cs-deepnlp-2017/lectures

5.斯坦福大学，自然语言处理和深度学习课程，http://cs224d.stanford.edu/

6.哥伦比亚大学，自然语言处理课程，http://www.cs.columbia.edu/~mcollins/

7.优达学城，机器学习图像分类项目，https://github.com/udacity/cn-deep-learning/

8.Matplotlib，https://matplotlib.org/users/pyplot_tutorial.html

9.Stanford，https://nlp.stanford.edu/wiki/Software/Classifier/20_Newsgroups

10.N-Gram，http://blog.csdn.net/ahmanz/article/details/51273500

11.词袋子模型，http://www.cnblogs.com/platero/archive/2012/12/03/2800251.html

12.Word2vec ，

http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

13.GloVec，https://nlp.stanford.edu/pubs/glove.pdf

14.Doc2vec，https://arxiv.org/pdf/1405.4053v2.pdf

15.20Newsgroups，http://www.qwone.com/~jason/20Newsgroups/

16.Tfidf，https://baike.baidu.com/item/tf-idf/8816134?fr=aladdin

17.Word2vec，https://radimrehurek.com/gensim/models/word2vec.html

18.Word2vec，https://en.wikipedia.org/wiki/Word2vec

19.Text8，http://mattmahoney.net/dc/text8.zip

20.https://github.com/dennybritz/cnn-text-classification-tf/blob/master/
21.textCNN
picture:http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/
22.textCNN:https://www.kaggle.com/yekenot/textcnn-2d-convolution
23.textCNN:https://www.kaggle.com/quoniammm/textcnn-baseline-glove-dropout/code