机器学习纳米学位

NLP 文档归类毕业报告

顾雪明
2017/11/24

一、项目背景

自然语言处理[1]（后面简称 NLP）是机器学习技术重要应用范畴之一，也是最具挑战的领域，包含了从语音识别、语音合成、语义分析、人机交互。最近几年，自然语言处理突飞猛进，涌现了很多新的技术和公司，如中国的科大讯飞[14]、百度语音[15]在语音识别、语音合成都取得了很大的成绩，包括在语音输入、自动翻译以及教育应用上都取得很大成就。国外苹果的 Siri，亚马逊的 Lex[16]，都是自然语言处理技术应用前沿产品的优秀实例。
尽管在语音识别、语音合成业界取得很大的成就，由于语义表达的复杂性、模糊型，我们在语义分析、人机交互还有很长的路要走，目前业界还处于简单交互和娱乐水平，如对命令、简短语句的回应和控制，如最新的暴风 TV[17]；或机械的应答，如翻译；或娱乐，包括各种语音交互玩具。
文本分类是按需求把文本划分为不同类别，它是语义分析和人机交互的基础，其本身就在行业中有直接的应用价值，比如新闻稿的自动分类、论坛发言审核、垃圾邮件检查等等。基于不同应用领域，同一篇文本也可以划分到不同的类别，同一篇文本我们既可以按照行业划分，如技术、体育、军事，也可以年代划分，或作者划分或文章风格划分等等。
以前文本分类一般是通过人手工完成，需要大量人力，而且并且缺少客观性、准确度。在现在互联网数据爆发的情况下，天量的数据靠人工分类根本无法完成，如垃圾邮件的检测发现，论坛发言的审核。这部分必须借助于机器学习技术，来实现文档的自动分类，我们甚至看到了网站安全管理员与恶意用户发言间的机器学习算法的激烈竞争:一方面维护网站次序，删除恶意、不良信息，一方面躲开审查发布广告等垃圾、不良信息。近几年来，Geofrey Hinton, Tomas Mikolov, Richard Socher 等学者深入开展了针对词向量的研究，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
我选择这个课题基于对 NLP 和人机交互报浓厚的兴趣，曾经与伙伴合作研制了一款车载机器人，基于 AIML[8]实现了车载特殊场景的人机交互。AIML 人机交互主要解决两个问题：一个是中文分词问题，这部分我们基于结巴分词，并对具体对话场景进行词库优化；一个构写对话规则，针对具体场景编写会话语料库。会话语料库是手工编写，一定程度可实现人机交互，但灵活性不够，适宜特定场景的简单应用。如果能够通过会话自动学习，不但效率提高，而且用户体验也会更自然。 这个课题虽然目标只是文档分类，但词向量的的神奇效果，非常吸引人 - 基于语料库可以学习到一个单词、一个短语或一篇文章特征矢量，也就可以描述他们在文字组成的虚拟世界中的位置。那稍微扩展一下，我们很容易基于某人的说话习惯，同样可以生成个人表达的特征矢量，那么我们是否可以基于每个人的特征优化人机交互的效果？

二、问题描述

文本分类以监督学习为主，通过对既有文档的手工识别归类，来引导机器学习。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。本项目将使用经典的 20 类新闻包官方网站[20]来验证文本分类，数据为 20 组已分好类的英文文本。 需要解决以下几个关键问题：

- **文本分词**， 单词是语义表达的最基本单位，英文分词相对中文要简单很多，空格就能基本实现分词的效果。然而，在文本中存在各种非文字符号，如标点符号，数字，简单去除这些符号会失去相应表达的信息，而包含这些信息，又导入大量低效、重复的单词。
- **文本语义**的表达方式是文本分类最关键的部分之一。
    - 一种是以单词或符号概率来表达文本的语义，比如采用词袋子模型[19]，即将段落或文章表示成一组单词，例如两个句子：”She loves cats.“、”He loves cats too.“ 我们可以构建一个词频字典：{"She": 1, "He": 1, "loves": 2 "cats": 2, "too": 1}。根据这个字典，我们能将上述两句话重新表达为下述两个向量: [1, 0, 1, 1, 0]和[0, 1, 1, 1, 1]，每 1 维代表对应单词的频率。
    - 以词与词的关系来表达单词、文本的语义 - 也就是词向量。比如有三个单词“man"、”husband“、”dog“，将之分别表示为[0,0,1]，[0,1,0]，[1,0,0]。Mikolov、Socher 等人提出了 Word2Vec[20]、GloVec[21]等词向量模型，能够比较好的解决这个问题。
    - 直接生成文本向量，称之为 Doc2Vec，有兴趣的可以拜读 Mikolov 的论文《Distributed Representations of Sentences and Documents》[22]
- **机器学习模型选择**， 从朴素贝叶斯、支持向量机(SVM)，到深度学习中的全连接神经网络、LSTM 循环网络、卷积网络等都可以用于文档文类，本项目将分别分析比较各模型的文档分类效果。参考 Karl Moritz Hermann 基于深度学习文档分类[10]。

三、评估标准

一个分类器最主要的评测指标就是**查准率（正确率）**和**查全率（召回率）**，**综合指标 F1** 是综合考虑了正确率和查全率: *2*正确率*召回率/（正确率+召回率）*。对于多类别分类器，主要指标是**微平均 F1（micro-averaging）**和**宏平均 F1（macro-averaging ）**两种指标，宏平均 F1 着重每个类别分类结果的平均，微平均 F1 则是对每篇文档分类结果的平均。参考文档[9]

本项目将采用微平均 F1（micro-averaging）和宏平均 F1（macro-averaging ）两种指标对文档分类结果分别作评估。分别对不同分词模型和机器学习模型做出评估。

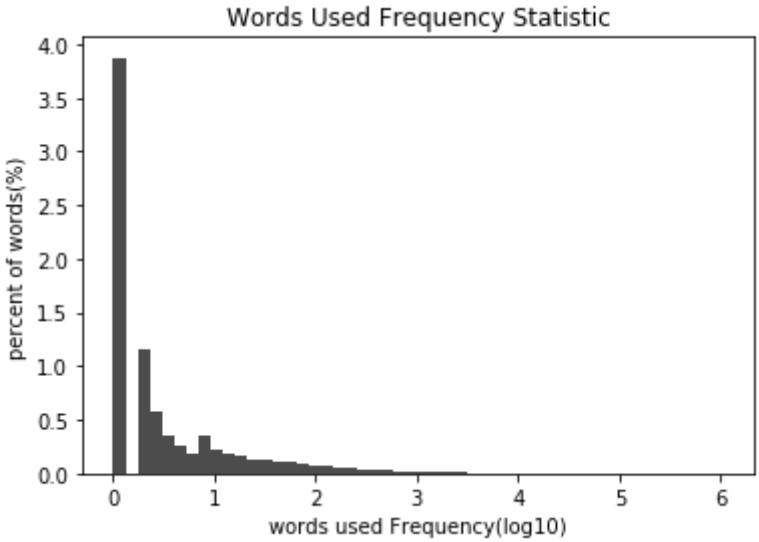
四、数据或输入

本项目将采用两部分数据：

- **词向量训练数据**，本项目采用 Mikolov 曾经使用过的 text8[23]数据包进行词向量的训练。text8 是经过整理干净的单词列表，不需要进一步处理。文本内容如下:“anarchism originated as a term of abuse first used against early working class

radicals including the diggers of the English revolution and the sans culottes of the French revolution whilst the term is still used in a pejorative way to describe any act...”

Text8 文档长度 17005208，一共包含了 253854 个不同的单词。从下图我们可以看出，text8 每个单词出现的频率宽度很大，从一次到数百万次。



最常出现的 30 个单词如下，都是我们最常见辅助词，稍显以外的是数字”nine”, “two”, ”eight”, “five”等排名非常靠前。这些常见单词虽然使用频率很高，但从意义来看，这些单词没法区分文档的属性，对文档分类帮助不大。所以在文档分类中如何弱化这部分常用但不重要的单词显得极为重要。

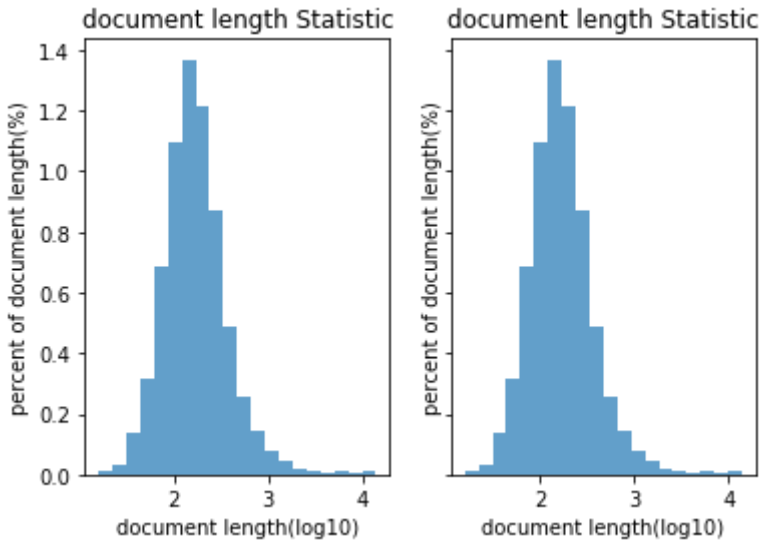
```
[('the', 1061396), ('of', 593677), ('and', 416629), ('one', 411764), ('in', 37
2201), ('a', 325873), ('to', 316376), ('zero', 264975), ('nine', 250430), ('tw
o', 192644), ('is', 183153), ('as', 131815), ('eight', 125285), ('for', 11844
5), ('s', 116710), ('five', 115789), ('three', 114775), ('was', 112807), ('by',
 111831), ('that', 109510), ('four', 108182), ('six', 102145), ('seven', 9968
3), ('with', 95603), ('on', 91250), ('are', 76527), ('it', 73334), ('from', 72
871), ('or', 68945), ('his', 62603)]
```

- **分类文本数据**，本项目使用经典的 20 类新闻包（以下称“新闻 20”），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，可以从官方网站[20]下载。这些数据是未经处理原始数据，如"b'From: mathew mathew@mantis.co.uk\nSubject: Alt.Atheism FAQ: Atheist Resources\nSummary: Books, addresses, music -- anything related to atheism\nKeywords: FAQ, atheism, books, music, fiction, addresses, contacts\nExpires: Thu, 29 Apr 1993 11:57:19 GMT\nDistribution: world\nOrganization: Mantis Consultants, Cambridge. UK.\nSupersedes"。
    - 需要统一解码为 Unicode，新闻 20 数据为 ANSI 编码，然而包括了部分非法字符，需要忽略非法字符才能完整解码数据。

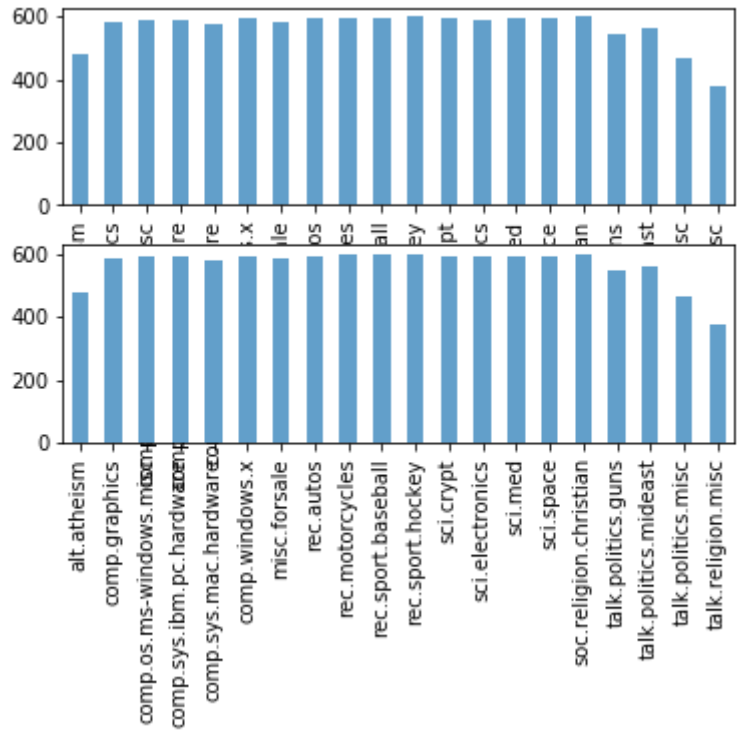
- 文本中包含大量未经处理字符，需要经过预处理，本项目采用小写字母，保留字母和字母与数字的组合，其他符号都作为分隔符。
- 新闻 20 已经包括了训练与测试两部分数据，本项目将把训练数据继续分成两部分用作机器学习，而新闻 20 测试部分数据将用于最终评估模型的成绩。

分析数据包，我们可以看到训练集有文档 11314 份，测试集有 7532 份。

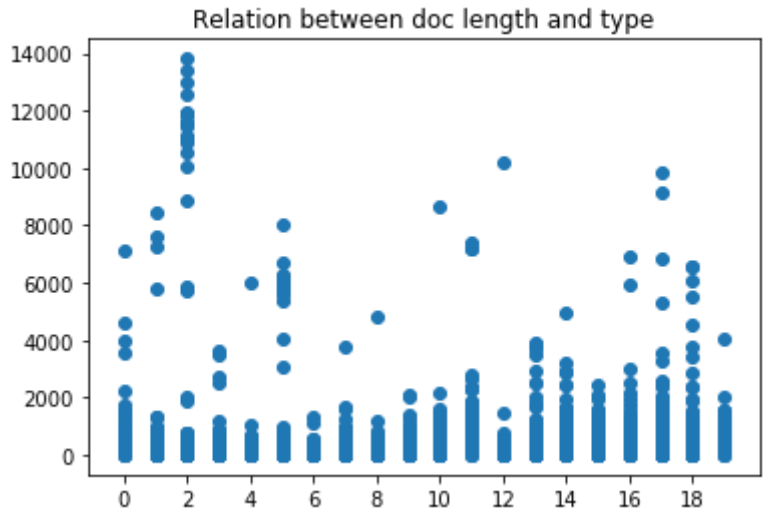
文档长度分布的长度分布范围较广，从几个单词到一万个单词，但训练集于测试集分布范围类似。



从文档类型看，新闻 20 数据包中，各类不同新闻基本平均分布，每类文档在 600 篇左右，训练集和测试集相似。



文档长度长度与于文档类型关系如下，从中可以看出文档长度与类型关系不明显，除了类型 2- 似乎大于 10000 个单词以上的新闻稿，都属于类型 2。



五、算法和技术

本项目探索文档的分类方法关键算法和技术，包括三部分：

（1）**文档预处理**：本项目遵循最小预处理的原则：

- 不考虑单词大小写。本项目暂不考虑单词的大小写，虽然部分单词大写往往有特殊含义。
- 单复数形式保持不变。一方面处理较为复杂，另一方面词向量本身就有助于查找到单复数的关系，不用在预处理中考虑单复数形式。
- 去除标点符号，所有标点符号将用作分隔符。
- 数字，本项目将去除所有单纯的数字，但把字母与数字的结合作为独立的单词。

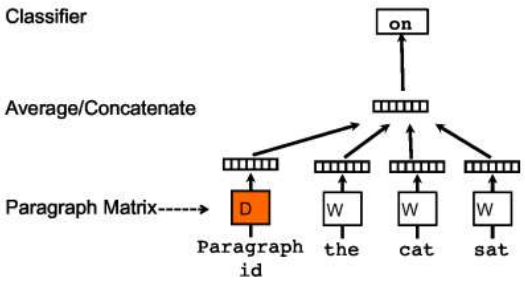
（2）探索文本语义的表达方式

- **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[24]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：

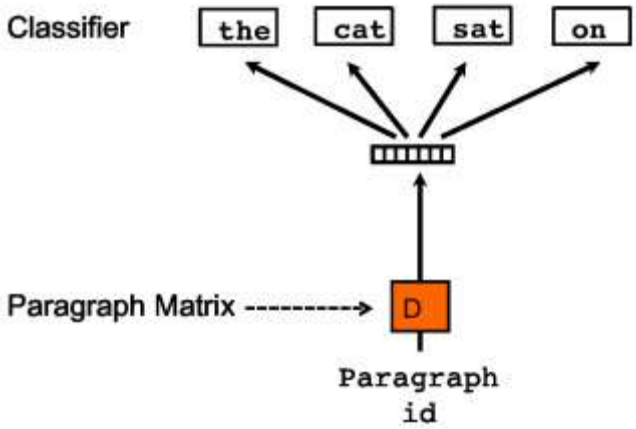
```
                    She    He    loves cats dogs    too
"She loves cats."       1     0     1      1    0     0
"He loves cats too."    0     1     1      1    0     1
"She loves dogs."       1     0     1      0    1     0
```

这部分功能的实现将基于 gensim[25]的 TfIdf model API 来实现

- **利用 Word2Vec/Glovec 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 使用 gensim 中的 models.word2vec，分别基 skip-gram/CBOW。Glovec 使用[GloVec]的库来训练。
  - 探讨怎样用文档中每个词的向量来表达整个文档。本项目采用以下几种方式：
    - 文档所有词向量的平均值作为文档的向量
      a) 本项目分别基于 100/300/1000 三个维度进行测试。
    - 采用文档的词向量和 TF-IDF 加权平均值作为文档的向量
      a) 本项目分别基于 100/300 两个维度进行测试。
- **文档向量 Doc2VEC**
  - 与词向量相对应，文档也可以用直接训练自己的向量来表示文档的含义。
  - 一种是通过分布内存文档向量（PV-DM），通过文档向量与词向量的组合，来预测下一个单词。
  - 一种是词袋子模型，直接通过词向量来预测文档中出现的单词（PV-DBOW）。
  - 本项目采用 gensim 库中 doc2vec 默认的 PV-DM 模型



PV-DM 模型

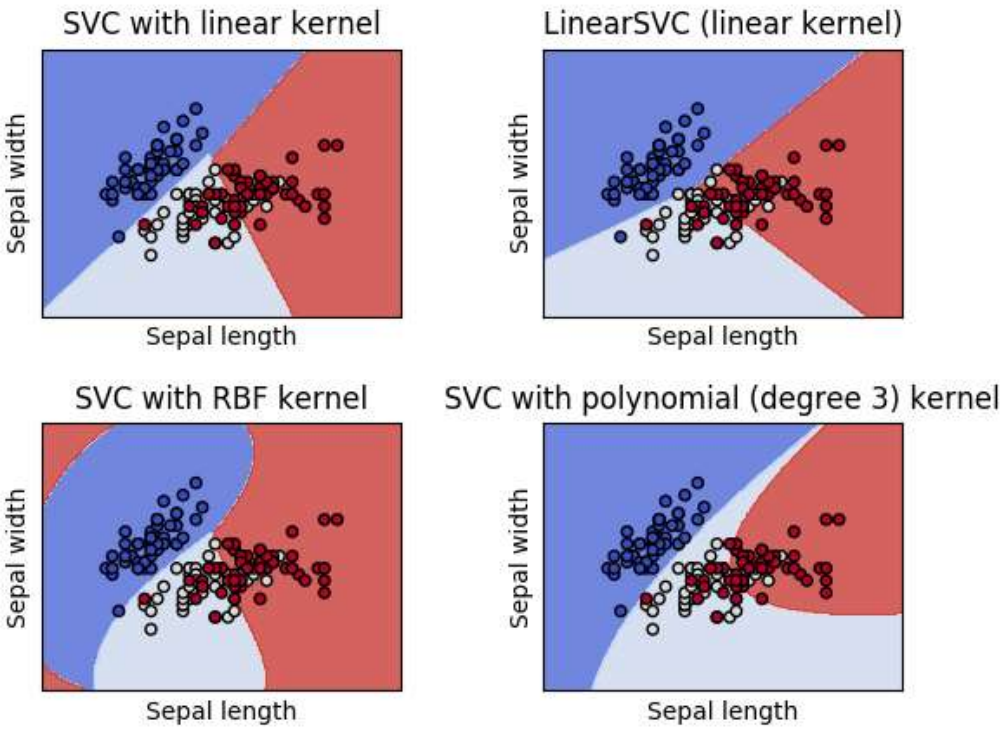


- PV-DBOW 模型

（3）**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，本项目使用以下模型：

- **支持矢量机(SVM)模型**

  支持向量机是一种高效的监督学习模型，在高维度空间对对象进行分类，基于目标到各空间分割面的距离，划分目标类型，可以进行多类别分类。采用不同的内核，可以起到不同的分类效果，如线性（Linear），径向（RBF）等。（下图来源于 http://scikit-learn.org/stable/index.html）



本项目采用线性内核，分别给予文档的 TFIDF 特性，WORD2VEC 的 100/300/1000 维特性，和联合 TFIDF/WORD2VEC 特性进行文档分类。

- **朴素贝叶斯模型**

  朴素贝叶斯模型是基于贝叶斯理论的一种可监督学习模型，对于

  $$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

  在考虑各个条件相互独立的情况下，可以简化为

  $$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

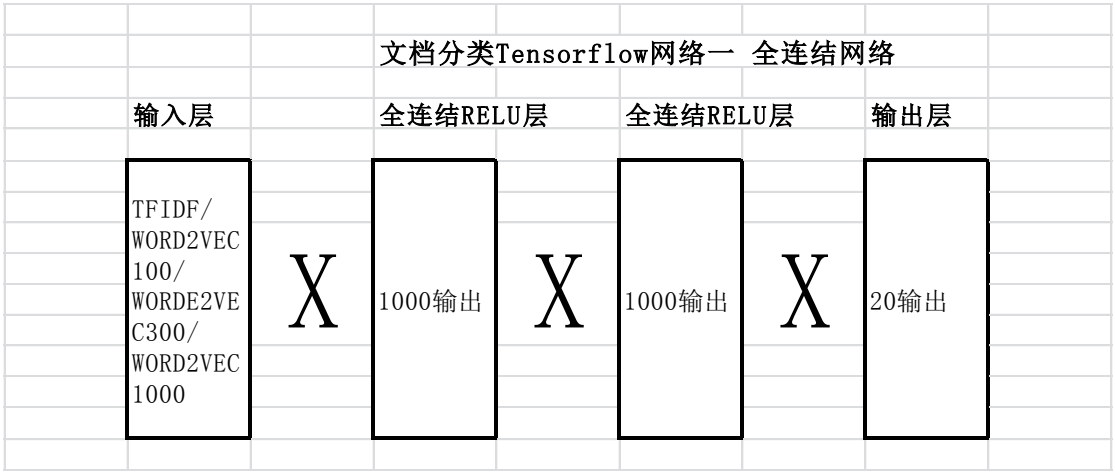
  对于文档分类来说，如果文档可以表示为各个独立的特性，那采用朴素贝叶斯也是一种简单有效的方法。本项目分别基于 word2vec 100 和 300 个特征进行分类。

- **神经网络深度学习模型**

  Tensorflow 是目前使用最广的深度学习工具包，广泛的应用于各种监督分类。

  本项目基于文档分类的特点，分别构建了三种深度学习模型：

  - **全连结 RELU 网络**

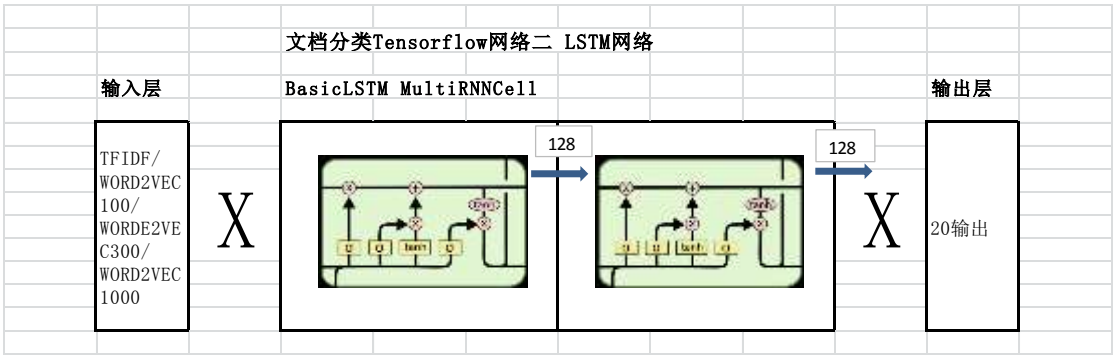


成本函数：本项目采用**交叉熵成本函数**，交叉熵函数可以避免神经元学习速度变慢的问题。

交叉熵：假设神经元的输出为 a=σ(z)a=σ(z)，这里 z=∑jwjxj+bz=∑jwjxj+b。神经元的交叉熵代价函数为 C=−1n∑x[ylna+(1−y)ln(1−a)]，当目标值和预测值都为 1 或 0 时，成本函数为 0，反之成本函数会迅速变大。较好地避免了学习速度变慢的问题。具体参考[28]

优化函数：本项目采用 Adam 优化算法，Adam 算法即自适应时刻估计方法（Adaptive Moment Estimation），能计算每个参数的自适应学习率，是随机梯度下降算法的一种，结合了 Momentum（动量更新方法，综合了上次梯度下降的方向，可以更快地收敛）和 AdaGrad 优化算法（可以基于下降速度自己调整学习速率）的优点，可以又快又好地收敛。

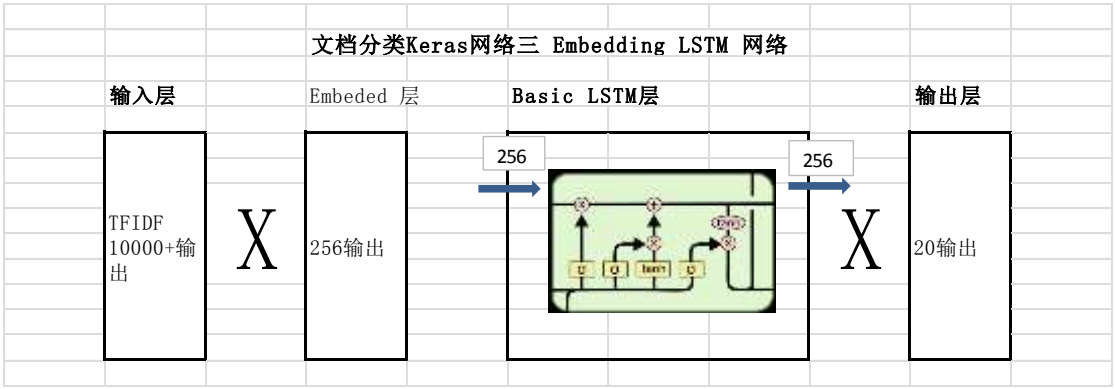
- **多层 RNN 网络**



成本函数：本项目采用交叉熵成本函数。

优化函数：本项目采用 Adam 优化函数。

- **Embedding RNN 网络**



成本函数：本项目采用交叉熵成本函数。

优化函数：本项目采用 Adam 优化函数。

模型三与模型二的区别在于，在模型三种文档单词的 Onehot 高度稀疏表示通过 Embedding 层动态映射为 256 维的输出。它不同于模型二，文档已经已经在预处理中转换为低维度的 word2vec 输出。模型三可通过文档分类结果动态调整文档的矢量维度输出。

六、基准模型

本项目采用斯坦福大学 20 类新闻分类模型作为基准模型（简称斯坦福 20）：
https://nlp.stanford.edu/wiki/Software/Classifier/20_Newsgroups
斯坦福 20 采用线性回归模型，重点通过优化分词来提高文档分类效果，主要采用了

- 正则表达式分词
  - 字符开头包含若干字符或数字
  - 或美元$
  - 或百分比数字
  - 或单个字母
  - 去除空格
  - 去除 uuencode 文本编码特殊信息”M[-]{60}”，uuencode unix 系统中文档传送格式，见[11]
- NGram 是大词汇连续语音识别中常用的一种语言模型 - P(wn|w1,w2…wn-1)，第 n 个单词可以由前面若干单词预测，参考[12]。 这可以识别由若干意义不明确的单词或符号组成的有意义短词，斯坦福 20 最后采用 NGram 模型生成最多 4 个单词组成的令牌。其缺点是处理需要更多的内存。

斯坦福 20 最终训练集的分辨得分为：
Micro-averaged accuracy/F1: 0.90361
Macro-averaged F1: 0.90277
最终测试集的分类得分为:
Micro-averaged accuracy/F1: 0.81731
Macro-averaged F1: 0.81158

七、方法

数据预处理

1. Text8 数据预处理

Text8 是已经经过预处理的词库，用空格即可以有效分词，本项目用于生成 word2vec 词向量。

2. 新闻 20 数据预处理

准备分类文本数据，本项目使用经典的 20 类新闻包，里面大约有 20000 条新闻，比较均衡地分成了 20 类。打开文件可以看到，这些文本数据是原始新闻数据，包含标点符号等数据，需要进行预处理。本项目

- 大小写，统一转换成小写字符。虽然大写的文本往往包含特殊含义，但也有部分纯粹语法要求，没有实际意义，在本项目文档分类中，暂不考虑这部分的影响。

- 标点符号，本项目去掉所有的标点符。标点符号在文档表达中有其特定的意义，但本文档中只考虑单词的意义对文档的影响，不考虑标点符号。
- 数字，去除所有独立的数字，但字符与数字的结合认为是一个单词。如"abc 123"中"123"会被去除，"abc123"则会认作一个单词。

分别读取 news20 中 train 和 test 数据，保存到变量，并以其中新闻目录作为文档分类的目标值

- document_train_x, 训练文档初始数据
- document_train_y，训练文档目标分类
- document_test_x， 测试文档初始数据
- document_test_y， 测试文档目标分类

文档预处理，采用 refine_corpus 函数对文档进行处理

- lower()把文档字符全部专成小写
- re.split(r'W+,str)则把所有非字符数字作为分词
- 通过 stop list 把一些无用的关键词给滤出了
- 通过 re.match(r'[0-9]+')把纯数字也滤出了

转换并保存 news20 数据

- document_train_x_refined， 整理过的训练数据
- document_test_x_refined， 整理过的测试数据
- document_train_y_lb, 训练文档分类编码
- document_test_y_lb， 测试文档分类编码

整理后的数据如下，已经是干净的单词列表：
['from', 'mathew', 'mathew', 'mantis', 'co', 'uk', 'subject', 'alt', 'atheism', 'faq', 'atheist', 'resources', 'summary', 'books', 'addresses', 'music', 'anything', 'related'…]

词向量的准备

生成 word2vec 模型，本项目采用 gensim.models.Word2Vec 库，基于 text8 生成词向量模型

1. model_word2vec，模型主要与一下因素：
    - 矢量大小，本项目分别选取 100/300/1000 三个维度作测试。
    - 窗口大小，当前字符与预测字符之间的距离，本项目选取默认的 5。
    - 模式 CBOW or Skip-gram, 这两个都是词向量训练方式，方式类似但相反，本项目采用默认的 CBOW。
        - CBOW 连续词袋子预测模型，是根据周边字符来预测当前字符。
        - Skip-gram，通过当前字符预测周边的字符。

- 输出层优化策略，使用 hierarchical softmax 或 negative sampling，本项目使用默认的 negative sampling，出现次数低于 5 的单词将被忽略。优化策略参考文档[31]
  - **Hierarchical softmax** 是一种对输出层进行优化的策略，输出层从原始模型的利用 softmax 计算概率值改为了利用 Huffman 树计算概率值
  - **Negative sampling**，是把语料中的一个词串的中心词替换为别的词，构造语料 D 中不存在的词串作为负样本。因此在这种策略下，优化目标变为了：最大化正样本的概率，同时最小化负样本的概率。

2. 生成模型分别保存于：
   - model_word2vec100
   - model_word2vec300
   - model_word2vec1000
3. 对 word2vec 的有效性进行验证，部分结果如下，能够较准确的反映单词的相似性：

“The similarity of 'woman' and 'man' is: 0.749896299054
The similarity of 'computer' and 'pc' is: 0.623348012708
The similarity of 'computer' and 'hp' is: 0.42880044116”

生成 TFIDF 语料库

1. Step1 - 基于 News20，使用 gensim.corpora.Dictionary 生成词典
2. Step 2 - 创建文档词袋子模型，使用 Dictionary.doc2bow 函数转换
   - corpus_bow_train_x
   - corpus_bow_test_x
3. step 3 - 生成 TFIDF 文档库
   - corpus_tfidf_train tfidf 训练集
   - corpus_tfidf_test tfidf 测试集

训练集 TFIDF 生成数据如下，本项目语料库共归纳了 107010 个单词：

```
[(17, 0.027342727271743677), (26, 0.0004485957594131995), (30,
3.648844482866216e-05), (38, 0.022024848266869842), (47,
0.015128262389073695), (49, 0.007058059188505339), (55,
0.726128393725912),。。。。]
```

文档的表达

文档可以有多种表达方式，最简单直接的，直接用连续词袋子模型（CBOW）表示，但由于不能排除大量使用但缺少表达意义的单词，文档分类效果较差，本项目不采用。
本项目分别使用 TFIDF、基于平均词向量、TFIDF 加权词向量、TFIDF 扩展词向量和文档向量

（DOC2VEC）。

1. TFIDF 文档表达

文档表达为每个单词 TFIDF 的列表，本项目将分别生成 TFIDF 训练集、验证集和测试集，用于以后不同模型的输入。

```
corpus_tfidf_train_train = [tfidf[x] for x in corpus_bow_train_x_train]
corpus_tfidf_train_test = [tfidf[x] for x in corpus_bow_train_x_test]
corpus_tfidf_test    = [tfidf[x] for x in corpus_bow_test_x]
```

2. 基于词向量平均值生成文档向量¶

文档表达为文档中每个单词 word2vec 的加权平均矢量，分别生成训练集、验证集和测试集。

document_train_x_train_average_word_vec（100/300/1000）

document_train_x_test_average_word_vec（100/300/1000）

document_test_x_average_word_vec（100/300/1000）

3. 生成基于 TFIDF 的加权词向量文档库

每个单词的 word2vec 词向量，再乘以 TFIDF 值，以期更好的反映单词重要性。

document_train_x_train_average_word_vec_tfidf（100/300/1000）

document_train_x_test_average_word_vec_tfidf（100/300/1000）

document_test_x_average_word_vec_tfidf （100/300/1000）

4. 生成基于 TFIDF 的扩展词向量文档库

基于文档的 TFIDF，但每个单词的值替换以 word2vec 值表达，以期更好的结合 TFIDF 与 word2vec 优点，反映单词重要性，但所需存储空间更加巨大。文档可以是基于 TFIDF 值选取部分或全部单词。文档转换方式为:*[TFIDF1, TFIDF2, ….]* ➔ *[VEC1, VEC2, ….]*。

document_train_x_train_extend_word_vec_tfidf（100/300）

document_train_x_test_extend_word_vec_tfidf（100/300）

document_test_x_extend_word_vec_tfidf （100/300）

5. 生成 doc2vec 文档库

与词向量类似，也可以直接生成文档向量，本项目采用 gensim.models.doc2vec，分别生成训练、验证和测试集。

document_train_x_train_doc2vec（100/300）

document_train_x_test_doc2vec（100/300）

document_test_x_doc2vec （100/300）

机器学习模型

本项目分别基于 SVM, NB 和深度学习模型进行分析

1. SVM:

本项目使用 Sklearn.SVM, Kernal 选择"linear"。分别基于不同的输入，评测新闻 20 的分类效果。

- TFIDF:

稀疏矩阵转变，由于 TFIDF 单词表达的维度高达 10 万，需要大量的内存空间，本项目定义了公共函数 *sparse_gensim2matrix（）*，利用用了 scipy.sparse 的 csr_matrix 库把 TFIDF 转换为 SVM 支持的稀疏矩阵格式。

- 平均 WORD2VEC:

取文档中词向量的平均值作为文档的向量表达。

- TFIDF 加权平均 WORD2VEC:

取文档中词向量的 TFIDF 加权平均值作为文档的向量表达。

- TFIDF 扩展 WORD2VEC

文档表达为词袋子的 TFIDF 稀疏矩阵，但 TFIDF 值使用 word2vec 向量表示。由于所需存储巨大，本项目尝试了 word2vec 100/300 向量。

- Doc2VEC

文档向量以 Doc2VEC 表达。

2. 朴素贝叶斯（Naive Bayers）

由于朴素贝叶斯算法不支持稀疏矩阵，本项目没有支持 TFIDF 的输入。分别基于以下输入:

- 平均 WORD2VEC:
- TFIDF 加权平均 WORD2VEC:
- Doc2VEC

3. 深度学习模型

本项目将分别尝试以下深度学习模型

- 全连结 Relu 网络：

本项目分别以 word2vec 100/300 和最高 TFIDF 值的词向量组合为文档向量。

- Multi LSTM RNN 两层网络

由于所需存储空间很大，本项目仅以 word2vec100 为输入。

- Embedding LSTM RNN

以每篇文档最高 TFIDF 值得单词为输入

测评函数：

本项目对每个模型的运行结果，都进行了 f1 评分，以对各模型分类效果客观评测。
F1 评分采用 sklearn.metrics f1_score，封装在自定义 get_f1_score(clf, input_data, target_data) 函数中。

八、结果

模型的评价与验证

本项目分别基于 SVM, Naïve Bayes 和深度学习模型对新闻 20 分类比较，以 f1 score 为评价模型的基准。

- SVM （F1 Macro / F1 Micro）

	训练集	验证集	测试集
TFIDF	0.994/0.994	0.906/0.908	0.821/0.826
WORD2VEC100	0.704/0.710	0.601/0.610	0.556/0.565
WORD2VEC300	0.776/0.781	0.650/0.658	0.580/0.590
WORD2VEC1000	0.774/0.779	0.650/0.657	0.578/0.587
WORD2VEC100 TFIDF	0.354/0.417	0.341/0.394	0.345/0.403
WORD2VEC300 TFIDF	0.356/0.420	0.341/0.399	0.348/0.407
DOC2VEC100	0.768/0.771	0.610/0.623	0.572/0.584
DOC2VEC300	0.916/0.914	0.535/0.547	0.516/0.525

基于 TFIDF 的结果相当的好，测试集文档分类 f1 达到 82%，超过了基准模型的 80%。TFIDF 模型与基准模型的主要区别在于降低了大量高频使用常用字的权重，而突出了在少数文档中使用的单词的全重。可以更好的体现文档的属性。
Word2VEC 在 SVM 中，表现符合预期，在 300 维度下, f1 准确度达到 58%，1000 维度对文档分类表现与 300 区别不大。这一方面与词向量表达意义方面还不够精确，相似性测试中也仅为 60%，另一方面本项目采用的词向量的平均值表示文档又牺牲了单词间的关系，仅保留平均统计值属性。

Word2Vec 加权 TFIDF，本意在词向量上进一步加上 TFIDF 全重，效果却更差，f1 准确率仅为 34%。研究后发现，本项目 word2vec 默认模式 negative sampling 已经考虑了单词使用频率的全重，再加上 TFIDF 是结果失真。
文档向量(Doc2VEC)，没有带来惊喜，测试集 f1 准确度为 57%，而 300 维度在训练集的得分提高，但测试集的分数反而降低，这明显是过拟合了。

- 朴素贝叶斯 （F1 Macro / F1 Micro）

	训练集	验证集	测试集
WORD2VEC100	0.420/0.427	0.395/0.403	0.372/0.383
WORD2VEC300	0.435/0.440	0.406/0.414	0.376/0.384
WORD2VEC TFIDF100	0.389/0.395	0.384/0.391	0.363/0.370
WORD2VEC TFIDF300	0.388/0.395	0.379/0.387	0.358/0.366
DOC2VEC100	0.372/0.342	0.391/0.367	0.385/0.367
DOC2VEC300	0.325/0.293	0.291/0.279	0.274/0.272

一方面由于 TFIDF 朴素贝叶斯的结论都不理想，这可以认为由词向量或文档向量表达的各向量间并非完全独立， 与朴素贝叶斯的基本前提不符。

- 深度神经网络 （F1 Macro / F1 Micro）

	训练集	验证集	测试集
RELU Network on WORD2VEC 100 (F1 Micro)	0.966/0.966	0.606/0.614	0.526/0.534
RELU Network on WORD2VEC 300(F1 Micro)	0.969/0.971	0.654/0.663	0.565/0.576
LSTM RNN WORD2VEC100(F1 Micro)	0.977/0.977	0.710/0.710	0.621/0.614
Embedding RNN	0.999/0.999	0.826/0.830	0.707/0.710

深度学习网络在本项目中初步结果还不令人满意，在测试集的准确度都接近或达到了 0.6 的以上。但在 LSTM RNN 网络，在每篇文档以 100 个最常用(TFIDF) 的单词 WORD2VEC100 表示，准确率达到了 0.707，比较接近基准模型和本项目的 SVM TFIDF 模型。

九、完善

初次运行深度学习网络即获得不错的效果，然而还没有达到基准网络和 SVM TFIDF 模型，观察初次结果可以看到基于静态词向量 + LSTM RNN 和 Embedding+LSTM 效果明显好于另外几个模型。本项目将分别从几个方面去调试完善这两个模型，在项目后期则重点深入研究了过拟合问题的优化，**并获得了高达 94%的分辨率**。

基于静态词向量 + 两层 LSTM 网络

本项目以初期 LSTM RNN WORD2VEC100 模型为基准，通过调整不同的网络参数，与基准网络模型比较，确定最优的模型。网络可调的参数较多，本项目从以下几个角度优化：

- RNN 内核函数，比较 LSTM 和 GRU 内核
- RNN 网络大小，比较 128 和 256 两种状态
- Batch Size，比较 32 和 128 两种大小
- Drop Rate，比较 LSTM 网路 0.6 和 0.8 两种 Drop Rate
- Output Size，比较不同的 LSTM 输出大小 - 1000， 256， 128
- 文档单词长度，分别尝试不同的文档输入长度 100，64，48， 32

通过比较最优的配置为：128 RNN Size + 32 Batch Size + 0.8 Drop Rate + 128 Oupput Size + 64 文档长度。**但最终结果仍不及预期，最优 F1 仅为 0.64.**

	F1 Macro	F1 Micro	训练时间	预测时间
初始配置(LSTM with 128 RNN size + 32 Batch Size + 0.8 Drop Rate + 1000 out size + 100 Doc Len)	0.612	0.622	1:30:13	00:02:30
RNN 内核函数	与 RNN GRU 内核作比较			
GRU	0.56	0.57	1:48:34	00:05:20
RNN Size	比较不同的 RNN Size			
256	0.586	0.586	3:26:28	00:06:28
Batch Size	比较不同的 Bath Size			
128	0.597	0.603	1:05:24	00:03:24
Drop Rate	增加 Drop Rate 从 0.8 到 0.6			
0.6	0.594	0.601	1:26:24	00:01:51
Output Size	测试不同的 Output Size			
256	0.624	0.626	1:09:28	00:02:21
128	0.629	0.637	1:21:42	00:03:05
Document Length	不同的文档长度			
64	0.632	0.646	00:52:16	00:02:19
48	0.614	0.621	00:36:05	00:01:14
32	0.611	0.619	00:24:59	00:00:53
综合最优配置	LSTM + 128 RNN Size + 32 Batch Size + 0.8 Drop Rate + 128 Oupput Size + 64 文档长度			

基于 Embedding + LSTM 网络的初步优化

本项目分别从几个方面考虑优化网络，。

- Batch Normalization(BN)层的引入和比较
- 单层与双层 LSTM 网络的比较。
- 不同学习率的比较。
- 调整初始化权重。
- 文档长度，本项目尝试了默认的 100 和 64 两种文档长度。默认的 100 效果更好。
- LSTM Size，本项目尝试了默认的 128 和 64 两种长度。默认的 128 效果跟好。
- 不同的 Drop Out， 分别尝试了 0.2、0.5 和 0.6。0.5 的时候测试效果最好。

最后本网络取得相对不错的文档分类效果 0.82，超过基准模型。配置为 Doc Len 100 + Embedding 128 + LSTM 128 + Drop Out 0.5 + Batch Normalization

	F1 Macro	F1 Micro
初始配置(Doc Len 100 + Embedding 128+ LSTM 128 + Drop Out 0.2echos 30)	**0.670**	**0.674**
Add BatchNormalization after ebemedding	0.794	0.798
Add another BatchNormalization after LSTM	0.809	0.816
Double LSTM	0.786	0.794
New Learning Rate	0.657	0.677
Adapt the initialization stddev = 0.01	0.804	0.812
Doc Len		
64	0.805	0.816
LSTM Size with Normalization 64 + Embedding Output Size 64	0.798	0.807
Drop Rate		
0.6	0.670	0.676
0.5	0.814	0.822
综合配置	**Doc Len 100 + Embedding 128+ LSTM 128 + Drop Out 0.5 + Batch Normalization**	

对过拟合的进一步优化

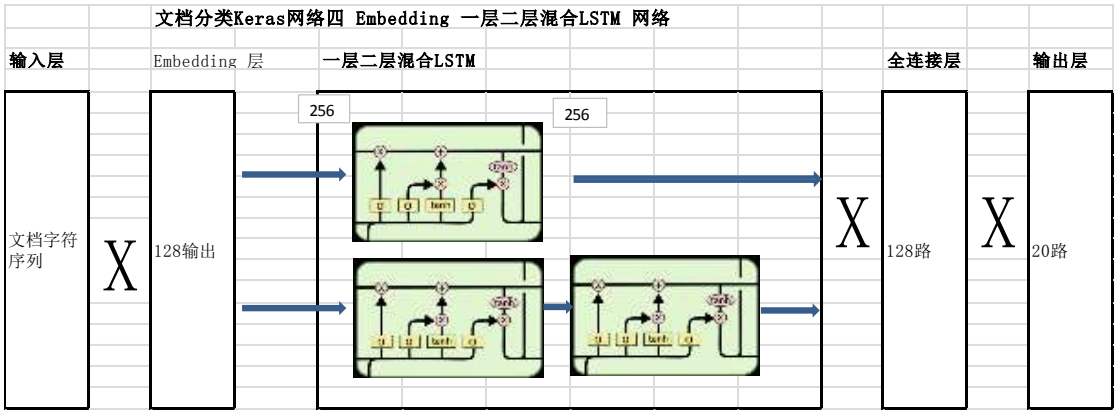
针对本项目出现的深度学习过拟合问题，本项目作了大量的实践和深入的学习，克服深度学习中玄之又玄的调参是关键， 主要从以下几方面入手。

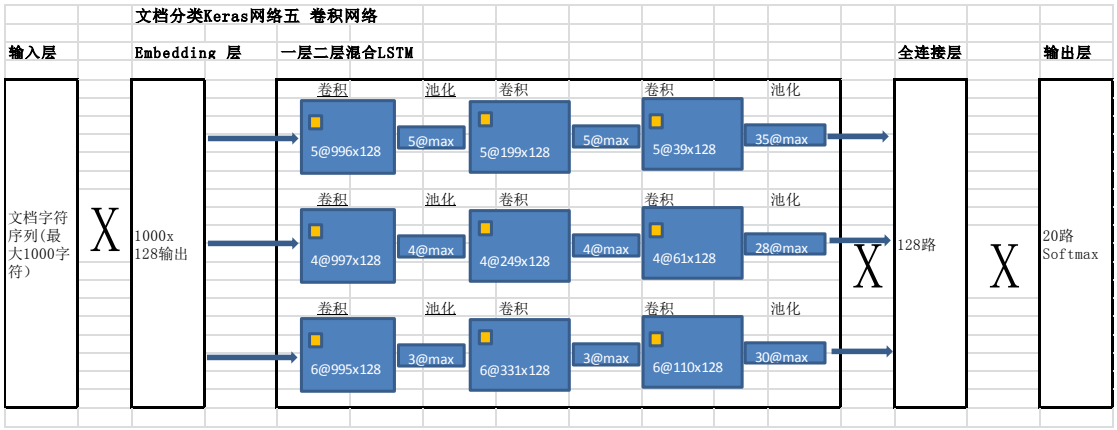
- 输入优化

  本项目还探索了短语(phases)、Glove 作为词向量输入的影响。

- 模型优化探索

  进一步探索了单双层混合 LSTM 网络和卷积网络。





（卷积模型参考 http://blog.sina.com.cn/s/blog_d76227260102wz1e.html）

- L2 规则化的探索

  L2 规则化是处理过拟合的最重要手段，本项目通过对学习率与 L2 规则化的大量实践观察，找到又快又好的模型收敛组合。

- 对输入数据的再整理

  本项目的数据来自 News20-bydate，训练数据和测试数据是按时间切割的。为了排除不同阶段数据的固有误差，本项目最后尝试把训练数据和测试数据混合再分配。

  在 Embedding + LSTM 模型中，准确率从 80+% 升到 94%。

  在 SVM 模型中，准确率从 82%升到了 92%。

十、合理性分析

本项目分析了各种文档的表达方式和机器学习模型。

- 输入数据的固有时间偏差。尽管服务万分努力优化模型，本项目使用新闻 20 的原始训练集数据和测试集数据作训练和测试时，测试集的准确率最高为 82+%，与验证集差距较大。当把原始训练和测试数据混合，再分配，测试集的准确率可达 94+%。这很可能是不同时间段新闻的关键词、风格引起的固有偏差。在训练时需要避免以时间划分训练和测试集。
- 不同模型的对未知数据的适应性不同。LSTM 深度学习模型在混合输入数据时，表现出出色的效果(94%), 好于 SVM+TFIDF(92%)；而在以时间划分的情况下，SVM + TFIDF 模型略强于深度学习模型, 表现出更强的兼容性。
- 基于词向量的 SVM 模型，文档分类结果比基准模型稍差，但由于文档表达的维度比 TFIDF 小很多，从 10 万多维度到 300 维度，信息在这过程中有所损失。
- 神经网络深度学习显示了优秀的性能，特别是 Embedding + LSTM RNN 网络，在以 100 个最高 TFIDF 值的单词为文档输入下，初步便获得了 82.2% f1 准确率。
- 在文档分类中，可输入的特性可以很多，以每个单词一个维度计，可达 25 万多维。**深度学习极易过拟合，要进一步提高准确率非常困难**，优化关键难在调参，从网络模型、数据输入、学习率、正则化等有无数的可能性。**本项目在大量的探索和实践中，逐渐总结出了合适的模型、学习率、L2 正则化等参数，最终获得了 94%的文档分类精度**。

十一、项目结论

本项目对各文档分类模型作了详尽的分析和比较，特别是对深度学习模型做了大量探索和实践，最终克服了模型的过拟合问题， **获得了 94%分类准确率**。

- 本项目中基于 TFIDF 的 SVM 达到了 82.6%的准确率，超过了基准模型 81.8%的准确率，作为本项目一开始便获得的结果，不失为文档分类快捷的一种方式，也是作为评价深度学习模型的一个重要参考和指引。
- 本项重点研究的深度学习模型 Embedding + LSTM 网络准确率初始运行就可以达到与 SVM 几乎相同的效果 82.2%。也超过了基准模型的准确率，各类文档分类效果总体比较平衡，最差的为 Religion 和 Electronics, f1 分类得分也都达到了 64%。
- 新闻 20 训练数据和测试数据间有较大固有时间偏差，通过混合训练和测试数据，再重新分配训练和测试集，Embedding+LSTM 测试集可获得 94%准确率，而 SVM 获得 92%准确率。
- 在训练和测试数据分布一致的情况下，深度学习模型可以比传统及其学习算法获得更高的准确率；而 SVM 表现出对未知数据更强适应性。
- 文档分类深度学习**关键问题在过拟合，关键解决要点在学习率和 L2 正则化参数**，本项目先采用较大的学习率和正则化参数获得较优模型，可以获得较快的收敛速度和准确率。
- 较少的关键词输入在深度学习模型中分类效果反而更好，**本项目自定的以 TFIDF 优选词向量为文档输入，获得理想的效果**，并且在减小模型输入，提高速度方面有明显优势。

十二、对项目的思考

由于本项目尝试系统的比较多种文档表达方式和多种文档分类算法，此项目的复杂性和所花的时间大大的超出初始的估计，尤其在后期，对深度学习中过拟合问题的优化进行了深度的探索，然而也正因为这付出，对自然语言处理、机器学习、对深度学习模型和网络优化有了深刻的理解。

- 项目实施流程
  - Notebook 项目实施结构合理清晰，从数据预处理、分析和最终的模型分析、优化都比较合理。
  - 本报告详尽的描述了整个过程
- *项目里有哪些比较有意思的地方？*

  本项目的实施有不少意外之喜
  - 最终获得测试集 94%的准确率。本项目长时间深度学习只能获得 82+%准确率，非常难以突破。在考虑了训练集和测试集的固有偏差时，重新分配训练集和测试集则深度学习模型可以获得 94%的准确率。虽然没有从模型上解决固有偏差，但从原理上理解了这偏差的重要性，在准备训练集和测试集时，应该考虑采用相同分布的数据。
  - 基于 TFIDF SVM 模型，一开始就给人惊喜，测试数据机文档分类 f1 指标轻松达到 82.6%，超过基准模型 81%的精度。
- *项目里有哪些比较困难的地方*
  - 新闻 20 训练集和测试集数据间固有时间偏差难以消除。尽管尝试了各种模型和优化方法，训练及和测试集固有时间偏差难以消除。
  - 深度学习过拟合问题非常难以解决，而模型调参也是玄之又玄，需要对模型、输入、各种参数进行探索。
  - *部分模型对内存需求巨大。比如：*
    - *基于 Naïve Bayes 的 TFIDF，由于不允许稀疏矩阵输入，所需内存巨大，只能放弃验证。*
  - *部分模型训练时间较长，如 Embedding + LSTM 网络，一次训练时间需要 3-4 小时。*
- *最终模型和结果是否符合你对这个问题的期望？它可以在通用的场景下解决这些类型的问题吗？*
  - 最终的模型和结果达到了我对问题的期望。文档是带有大量冗余、噪音信息的，察看 Region.Misc（这部分准确最低）中文档，一部分文档只有几十个单词，并没有显著宗教相关单词。要准确分类很困难，一开始认为能超过 85%就是不错的成绩。
  - 模型优化的难度和效果都超出了我的预期，花了大量时间探索模型的合理参数，我一旦掌握了模型参数，深度学习模型表现出的魔力也着实让人吃惊。
  - 基于 TFIDF+词向量的输入，获得理想的结果，和我一开始的期望相符。
  - 基于 LSTM 模型，获得了理想的结果，特别后期探索出一层双层混合模型，可以综合文档不同层次的有用信息，大大加速了模型训练收敛的速度。
  - 最终模型对类似文档分类问题具有一定的普适性，比如垃圾信息监测、文档情态分析等都适用，特别是模型优化的方法可以是通用的。

十三、需要作出的改进

本项目已经系统的考虑了文档分类各种技术，特别对深度学习模型进行了深入探索和研究，并取得了理想的成绩。

- *是否可以有算法和技术层面的进一步的完善？*
    - *词向量表达，在相似性测试中，结果为60%准确性，还不是很理想。包括Glove 等算法，还没机会去完善。*
- *是否有一些你了解到，但是你还没能够实践的算法和技术？*
    - *本项目实践了各种主要的算法和技术，对卷积深度学习模型由于运行速度较慢，只做了初步的探索。*
- *如果将你最终模型作为新的基准，你认为还能有更好的解决方案吗？*
    - *当前模型已经获得较为理想的效果，还可以进一步优化参数，作局部优化。*

工具

本项目将使用至少以下工具包：

- 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. http://alice.pandorabots.com/
8. http://www.alicebot.org
9. http://blog.csdn.net/xiaoyu714543065/article/details/8559741
10. 基于深度学习的文本分类 https://github.com/oxford-cs-deepnlp-2017/lectures/blob/master/Lecture%205%20-%20Text%20Classification.pdf
11. Uuencoding https://en.wikipedia.org/wiki/Uuencoding
12. N-Gram 用于文本分类 http://blog.csdn.net/shiwei1003462571/article/details/43482881
13. 文本分类课题 https://github.com/nd009/capstone/tree/master/document_classification
14. 科大讯飞 http://www.iflytek.com/audioengine/index.html
15. 百度语音 http://ai.baidu.com/
16. Lex https://aws.amazon.com/cn/lex/

17. 暴风 TV http://www.bftv.com/
18. 20 类新闻包 http://www.qwone.com/~jason/20Newsgroups/
19. 词袋子模型 http://www.cnblogs.com/platero/archive/2012/12/03/2800251.html
20. Word2Vec
    http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf
21. Glove http://nlp.stanford.edu/pubs/glove.pdf
22. Doc2Vec: https://arxiv.org/pdf/1405.4053v2.pdf
23. Text8: http://mattmahoney.net/dc/text8.zip
24. TF-IDF:
    http://baike.baidu.com/link?url=toXJqDyZ1smDK2HpzusBzUnWX6YlKffU9bigEa5DHEOHmF0pL6XsDlhbzF10sijRGPeeml5Ze3cOtGAIHLXT0_
25. Genism: http://radimrehurek.com/gensim/
26. 斯坦福深度学习课程第二弹：词向量内部和外部任务评价
    https://yq.aliyun.com/articles/82105
27. Wordsim353, Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin, **"Placing Search in Context: The Concept Revisited"**, *ACM Transactions on Information Systems, 20(1):116-131, January 2002* [Abstract / PDF]
    http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/wordsim353.html
28. 有关交叉熵的理论
    https://hit-scir.gitbooks.io/neural-networks-and-deep-learning-zh_cn/content/chap3/c3s1.html
29. ADM 算法 https://zhuanlan.zhihu.com/p/27449596
30. 关于 LSTM/MULTI RNN CELL
    https://r2rt.com/recurrent-neural-networks-in-tensorflow-ii.html
31. 词向量输出优化算法 Hierarchical Softmax vs Negative Sampling
    http://www.cnblogs.com/Determined22/p/5807362.html
32. Batch Normalization http://blog.csdn.net/hjimce/article/details/50866313
33. 卷积网络用于文档分类
    http://blog.csdn.net/u010223750/article/details/53334313?locationNum=7&fps=1