机器学习纳米学位

毕业项目报告

Flyzhg 优达学城
2018/9/20

文档分类

I. 问题的定义

项目概述

自然语言处理是（NLP: natural language processing）是当前信息时代中最为重要的技术之一。 是人工智能领域中占有举足轻重的地位。是研究计算机科学与人类的语言学相互转换和识别的科学。NLP 的应用无处不在，因为人们用语言进行大部分沟通：网络搜索，广告，电子邮件，客户服务，语言翻译，发布学报告等等。人类想让机器， 尤其是具有智能的计算机能力的机器，能做到具有人类能达到的地步。 在 NLP 应用的背后，就是有大量的基础任务和机器学习模型， 这些都是我们需要持续投入和研究的。

自然语言处理是人类交互方式的巨大进步。 人类革命的每一次进步都是交互方式变革引起的，尤其是以电子计算机为代表的第三次信息技术革命最能体现这个特点。 从人工手动敲击键盘输入指令，到计算机能自动识别人类的语音并呈现出来，这些都是交互方式的重大改善，改进了人与机器之间的交往模式，变得更简单，更高效，更快速。

在自然语言处理过程中，需要大量的数据作为燃料，为我们的机器学习算法这个引擎进行消耗，这样才能训练出更好的分类结果。目前阶段， 为了很多自然语言处理算法的评估，也考虑到很多人学习使用的方便性， 最初有 Ken Lang 在他的论文 Newsweeder：学习过滤 netnews 专门收集的， 尽管没有明确提出。 20 个新闻组数据集是大约 20,000 个新闻组文档的集合，在 20 个不同的新闻组中均匀分区（几乎）。目前，20 个新闻组集合已经成为机器学习技术的文本应用实验的流行数据集，例如文本分类和文本聚类。[1]

问题陈述

该项目的主要研究问题是通过训练的模型，对未分类的数据进行进行分类，看能否分类的比较正确。

在实际的过程中，需要了解输入的数据是 20NewsGroups 的新闻数据集。通过代码 log 如下：

```
['alt.atheism',
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware',
 'comp.sys.mac.hardware',
 'comp.windows.x',
 'misc.forsale',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.electronics',
 'sci.med',
 'sci.space',
 'soc.religion.christian',
 'talk.politics.guns',
 'talk.politics.mideast',
 'talk.politics.misc',
 'talk.religion.misc']
newsgroups_data category numbers: 20
```

总共有 20 类的数据，输入一个数据后， 我们想得到的结果是，通过分类模型，这个数据属于这 20 个数据集中的那一个。就是所谓的多分类的问题。

解决问题的方法就是，参考如下的分类模型

- 决策树模型
- 支持矢量机(SVM)模型
- 朴素贝叶斯模型
- 神经网络模型

进行分类。 因为文本信息不是直接的标签数据，需要我们对 20NewsGroups 进行处理，先使用词袋子模型或者词向量模型对文本进行数学化的表示, 以便后续的分类器处理，训练和预测等。

评价指标

我们评价模型和结果的指标采用 accuracy_score； accuracy_score 可理解为分类准确率，是所有的分类正确的样本的数量与总样本数量的比值；这个概念优点就是比较容易实施计算，比较容易理解。缺点就是相应值的潜在分布不能得到。分类准确率的定义如下：

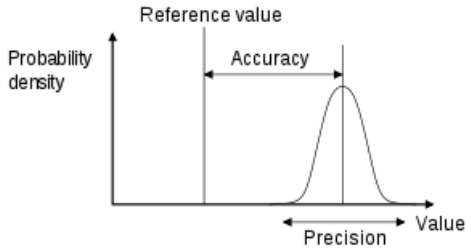
Accuracy=（TP+TN）/(TP+FP+TN+FN)

相反的就是错误率

Error = （FP+FN）/(TP+FP+TN+FN)

相关的就是 Precision

Precision = TP/(TP+FP)



图：Accuracy 是测量结果与真实值的接近度，precision 是重复性测量误差。

II. 分析

数据的探索

所使用的 20NewsGroups 数据集的信息如下大小：

数据集共有 18846 条记录
训练集有 11314 条记录 测试集共有 18846 条记录
新闻类别如下；

```
['alt.atheism',
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware',
 'comp.sys.mac.hardware',
 'comp.windows.x',
 'misc.forsale',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.electronics',
 'sci.med',
 'sci.space',
 'soc.religion.christian',
 'talk.politics.guns',
 'talk.politics.mideast',
 'talk.politics.misc',
 'talk.religion.misc']
```

数据集中，target_names 表示，新闻类别。
数据集中，target 表示：每个新闻的类别 id;
数据集中，data 表示：每个新闻包含的内容，分为 headers， footers，quotes，正式内容这四部分
数据集中，filenames 表示： 每个新闻的文件名
根据输出：当前新闻有 20 个类别；
subset，子类类型，用来设定数据类型：可以设置 train，test，all。
category，类别，就是数据集新闻的类型； 如果设置成 None， 会加载所有的 categories。
remove， 删除新闻的部分内容，主要包括 headers， footers， quotes，这三部分
shuffle 和 random_state 这两个要结合起来用。当前项目 random_state=42

所谓的离群值，就是某些数据远远大于或小于其他的数据，显的不太和群，称之为离群值，或者异常值，极端等。 我们选择的 20newsgroups 最初是由 ken lang 收集的，应该都是筛选过的，不需要我们进行异常筛选了。所以这些(分类变数，缺失数据，离群值等）就不用实施了。

探索性可视化

展示一下第一条新闻的内容：

From: Mamatha Devineni Ratnam <mr47+@andrew.cmu.edu>
Subject: Pens fans reactions
Organization: Post Office, Carnegie Mellon, Pittsburgh, PA
Lines: 12
NNTP-Posting-Host: po4.andrew.cmu.edu

I am sure some bashers of Pens fans are pretty confused about the lack
of any kind of posts about the recent Pens massacre of the Devils. Actually,
I am bit puzzled too and a bit relieved. However, I am going to put an end
to non-PIttsburghers' relief with a bit of praise for the Pens. Man, they
are killing those Devils worse than I thought. Jagr just showed you why
he is much better than his regular season stats. He is also a lot
fo fun to watch in the playoffs. Bowman should let JAgr have a lot of
fun in the next couple of games since the Pens are going to beat the pulp out of Jersey
anyway. I was very disappointed not to see the Islanders lose the final
regular season game.   PENS RULE!!!

第二条新闻内容如下：

From: mblawson@midway.ecn.uoknor.edu (Matthew B Lawson)
Subject: Which high-performance VLB video card?
Summary: Seek recommendations for VLB video card
Nntp-Posting-Host: midway.ecn.uoknor.edu
Organization: Engineering Computer Network, University of Oklahoma, Norman, OK, USA
Keywords: orchid, stealth, vlb
Lines: 21

 My brother is in the market for a high-performance video card that supports
VESA local bus with 1-2MB RAM. Does anyone have suggestions/ideas on:

 - Diamond Stealth Pro Local Bus

 - Orchid Farenheit 1280

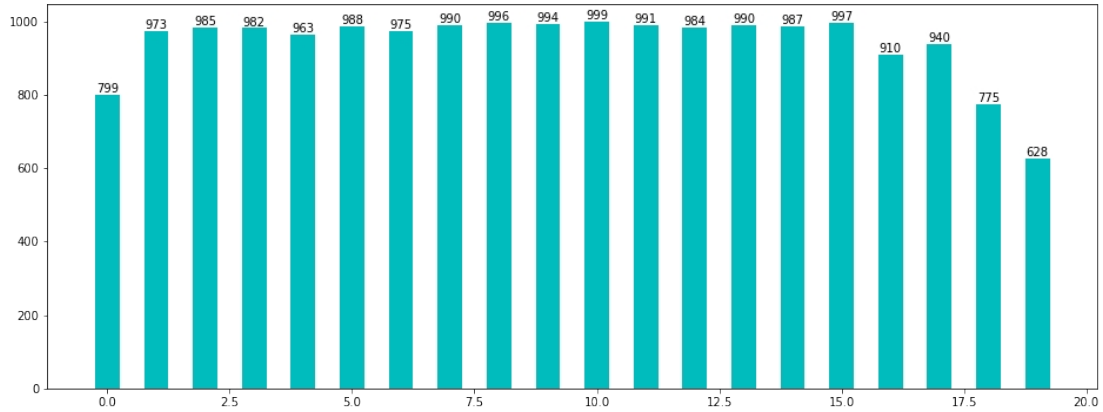
 - ATI Graphics Ultra Pro

 - Any other high-performance VLB card

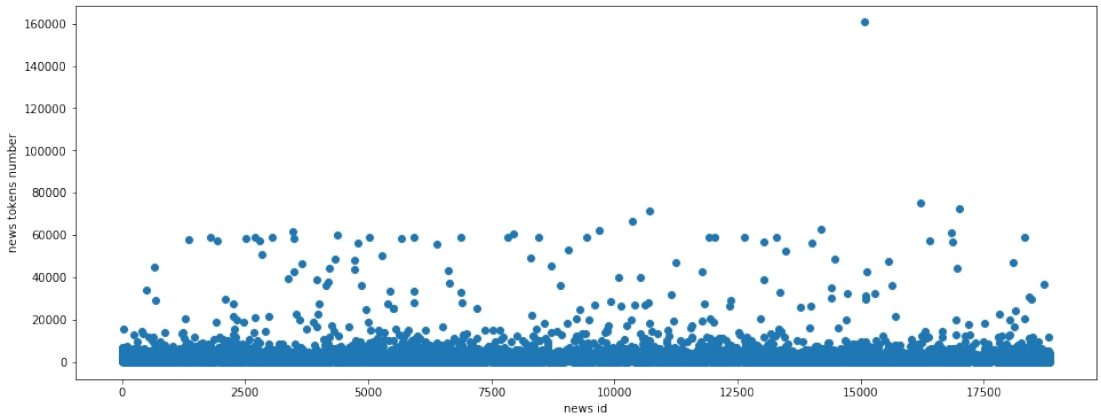
```
Please post or email.  Thank you!

  - Matt

--
    |  Matthew B. Lawson <-------------> (mblawson@essex.ecn.uoknor.edu)  |
  --+-- "Now I, Nebuchadnezzar, praise and exalt and glorify the King  --+--
    |   of heaven, because everything he does is right and all his ways  |
    |   are just." - Nebuchadnezzar, king of Babylon, 562 B.C.           |
```

分析这 2 条新闻数据可以发现，有些共同点，比如都包好 From，subject，NNTP-Posting-Host， lines 这个几个部分等等。从整体上说，分为 headers，footers， quotes，正文，这四部分。Headers 表示新闻的头部信息，footers 尾部信息，中间正文部分，还有 quotes 引用部分的内容。

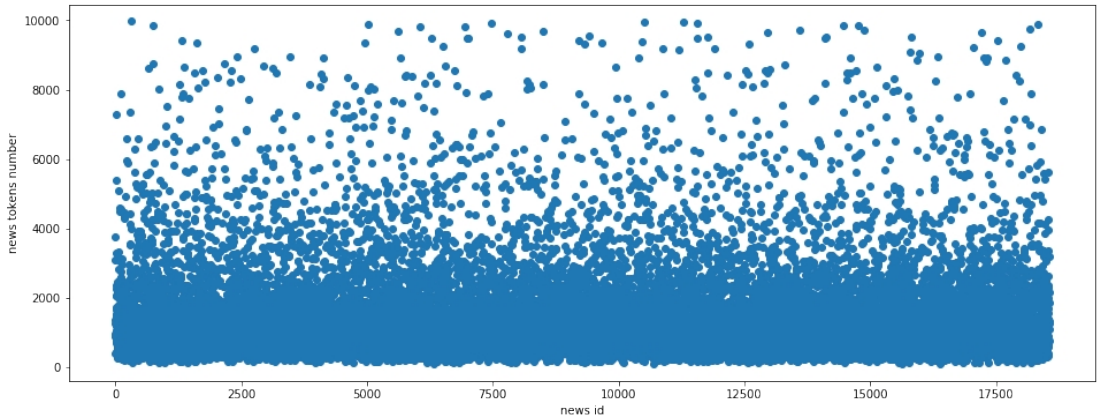
整个新闻集每一个类的新闻数量都差不多，第一类和最后两类数量能偏少一些， 数据如下所示：
0 : 799， 1 : 973， 2 : 985， 3 : 982， 4 : 963， 5 : 988， 6 : 975， 7 : 990，
8 : 996， 9 : 994， 10 : 999， 11 : 991， 12 : 984， 13 : 990， 14 : 987， 15 : 997， 16 : 910， 17 : 940， 18 : 775， 19 : 628
更形象化的展示，如下图所示；



所有新闻中，字数最小的新闻是有 115 个单词； 字数最大的新闻有 160616 个单词，平均字数的新闻是 1902，中位数是 1175 个单词。整个数据的分布情况如下图：



通过图像发现大部分的数据都在 10000 一下， 然后我们剔除 10000 以上的 291 个数据，占的比重仅有 1.5%。 重新散点图展示一下如下：



发现大部分的数据都在 2000 以下，这个和之前的平均数基本是一致的。

算法和技术

数据预处理

本项目的文本分类主要的是实施对象是英文文本，因此，在数据的预处理时，也要偏向于英文文本的处理，比如，单复数，大小写等，这个在中文坏境中时不用考虑的。 中英文共同的要处理的部分，包括停用词的处理，标点符号，词性等处理。

在 Phthon 的自然语言处理中，NTLK（Natural Language Toolkit） 是一个比较著名工具包。我们采用该包中的工具来进行数据预处理。

最先处理的，应该删除 footer 和 quotes 相关数据，我认为这些数据对正文影响不大，而且容易包含各种特殊符号，因此要通过 remove 先删除掉。 为什么不删除 headers，主要原因是 headers 中包含了新闻中有 keywords，keyword 对于区分新闻的类别很重要，是一个非常重要的特征，所有 header 不能删除。

通过数据分析，决定 footer，和 quotes 也不能删除，详细见报告内的数据分析处理部分。

接下来，目前的新闻集都是一条一条新闻存在的，我们需要对他们进行分词，word_tokenize 就是对新闻内容进行分词用的。处理完分词后，一些标点符号，空行，日期等等对于本文的分类没有多大的相关性，因此也要删除，如果是考虑情感分析，则需要单独考虑这些符号。删除标点符号后，就需要调整字母的大小写，对词根进行处理，最后就是要删除停用词了。
这里还有一个需要考虑的问题，就是在要删除在整个 20Newsgroups 语料库中出现次数为 1 的低频词，这个对于分类的准确度也会有很大帮助。

文本表示的方式

文本表示的思路首先是要文本进行分词，也就是将新闻集或者文档改造成单词的集合，建立一种计算机便于运算的方式来存储的集合。

词袋子模型

词袋子模型，即 Bag-of-words model (BoW model) 最早出现在 NLP 和 IR 领域. 该模型忽略掉文本的语法和语序, 用一组无序的单词(words)来表达一段文字或一个文档. 近年来, BoW 模型被广泛应用于计算机视觉中. 与应用于文本的 BoW 类比, 图像的特征(feature)被当作单词(Word). [2]

词袋子模型即将段落或文章表示成一组单词，例如两个句子：”She loves dogs.“、”He loves dogs too.“ ,这样我们可以构建一个词频字典：{"She": 1, "He": 1, "loves": 2 "dogs": 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。[3]

词袋子模型，主要的实现函数是 CountVectorizer 和 TfidfVectorizer，这两个函数都来自于 sklearn.feature_extraction.text 这个子模块；该子模块的作用就是
收集实用程序从文本文档建立特征向量。主要包含 4 个函数，如下所示：
feature_extraction.text.CountVectorizer（[...]）
将文本文档集合转换为令牌计数矩阵
feature_extraction.text.HashingVectorizer（[...]）
将文本文档集合转换为令牌出现的矩阵
feature_extraction.text.TfidfTransformer（[...]）
将计数矩阵转换为标准化的 tf 或 tf-idf 表示
feature_extraction.text.TfidfVectorizer（[...]）
将原始文档集合转换为 TF-IDF 特征矩阵。[4]

这里只介绍一下两个类的使用方法，CountVectorizer 与 TfidfVectorizer，这两个类都是特征数值计算的常见方法。对于每一个训练文本，CountVectorizer 只考虑每种词汇在该训练文本中出现的频率，而 TfidfVectorizer 除了考量某一词汇在当前训练文本中出现的频率之外，同时关注包含这个词汇的其它训练文本数目的倒数。相比之下，训练文本的数量越多，TfidfVectorizer 这种特征量化方式就更有优势。

在介绍 sklearn.feature_extraction.text.TfidfVectorizer 这个函数之前，先要介绍一下 tf-idf； tf-idf 等于 tf x idf 的乘积； tf 的定义是某一个训练数据中，某一个词出现的次数， 有时候也称之为词频 （Term Frequency）； idf 的英文是 Inverse Document Frequency，常常称之为逆文档频率，用来调整 tf 的权重，被称之为于权重调整系数。 IDF 的公式如下：

$$IDF(t) = log\frac{m}{n+1}$$

其中 m 为所有文档的总数量， n 为包含单词 t 的所有文档的总数量。n+1 是为了考虑某一个文档不包含该单词，分母不能为 0；加 1 的影响也不会影响太大。数学上的解释就是如果一个词越常见，那么分母就越大，逆文档频率就越小越接近 0，也就是该权重系数很小。 log 表示对得到的值取对数（此处为自然对数）。

通俗的解释，就是 如果一个单词在很多文档中都出现，比如 and，那么说明这个词非常的常用，对于区分谋篇文章或者新闻贡献的作用就越小，也就是权重值很低。

词向量模型

最近，随着深度学习概念的兴起及硬件计算能力的提升，很多学者提出了词向量的概念。 词向量作为基础，可以方便的引入机器学习的模型以便对文档进行分类，分析，预测等。最简单的词向量表达方式是独热编码，缺点是不能表达关联性，并且维度容易增加太快，给存储和计算带来了很多的问题。

词向量简单的说就是将新闻文档中的单词转化成一个密集向量，对于其中相似的词，其对应的词向量也非常相近。其优点是能表达出词之间的相似关系，还可以包含更多的信息。

目前，转向于 Word2Vec 模型。 Word2Vec 模型是一群用来产生词向量的相关模型， 通过这些模型，实现词向量模型表示每篇文档。操作过程就是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。另外，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 算法简单说来就是一个隐含层的神经元网络，输入层是新闻集的词汇表，输出层也是一个词汇表。实现的过程就是对于新闻样本中的每一个单

词，就在词汇表中相应的位置设置成 1， 相反设置成 0. 用大量的样本训练这个网络之后， 当收敛时， 把输入层到最后的隐藏层的那些权重，作为每一个词汇表中的词向量。词的个数，就是词汇表向量的维度。有了每个词的有限维度的向量，就可以用到其它的应用中，因为它们就像图像，有了有限维度的统一意义的输入[5]

文本分类器与分类技术

词袋模型分类器与分类技术

词袋子模型的训练的主要使用如下 3 种模型分类器

1. 决策树模型

   决策树 Decision Tree，一种基于倒树结构进行决策的模型。本质上是通过一系列的规则判断而对数据样本进行分类的过程。实现方式上的理解上类似于很多的 if else 函数，数学的概念上的理解就是一系列的分段函数。这种模型在很早之前就已产生，是一种经典的分类方法。 经常分为两个步骤来实现，第一构建决策树，通过对训练样本的训练生成决策树； 第二步，就是 决策树的剪枝，对第一步生成的决策树进行修改，校正的过程。 整个过程就是，决策树通过数据集学过的模型，寻找最佳的划分特征对新的样本进行分类的过程。

2. 支持向量机 SVM 模型分类器

   支持向量机 Support Vector Machine, 简称 SVM。 也是一种比较典型的分类方法，在深度学习出现之前，曾经有很不错的表现。 具体过程如下： 给定一个训练样本集 如下：

   D={(x1→,y1),(x2→,y2),…,(xn→,yn)}D={(x1→,y1),(x2→,y2),…,(xn→,yn)}，

   yi∈{+1,−1}yi∈{+1,−1}，

   i 表示第 i 个样本，n 表示样本容量。分类学习最基本的想法就是基于训练集 D 在特征空间中找到一个最佳划分超平面将正负样本分开，而 SVM 算法解决的就是如何找到最佳超平面的问题。超平面可通过如下的线性方程来描述：

$$\vec{w}^T\vec{x}+b=0$$

   其中 w⃗ 表示法向量，决定了超平面的方向；表示偏移量，决定了超平面与原点之间的距离。 [6]

3. 朴素贝叶斯模型分类器

   朴素贝叶斯也是经典的机器学习算法之一，也是为数不多的基于概率伦的分类算法。常用于文本分类，垃圾邮件过滤等。 算法的核心思想是通过考虑

特征概率来预测分类。从数学的理论上讲，朴素贝叶斯的核心思想就是贝叶斯法则，而贝叶斯法则的理论基础就是条件概率。 贝叶斯法则如下：

$$p(c_i \mid x,y) = \frac{p(x,y \mid c_i)\,p(c_i)}{p(x,y)}$$

简单版本就是 P(A|B)=P(B|A)P(A)/P(B)

条件概率的理解要从先验概率和后验概率说起，以生产某一个产品部件为例，根据以往的经验，或者抽查很多数据，发现部件良好的概率是 p1， 这个就是先验概率。而后验概率是通过之前的判断比如满足某一个条件或者是合格的条件下，进行重新修改校正后的概率。由于涉及关于所有属性的联合概率的问题， 为了计算联合概率的方便，朴素贝叶斯分类器采用了“属性条件独立性假设”:对已知类别，假设所有属性相互独立。通俗的说，就是，每个属性对分类结果的影响都是独立的，互不相干的， 这个就是假设的前提。

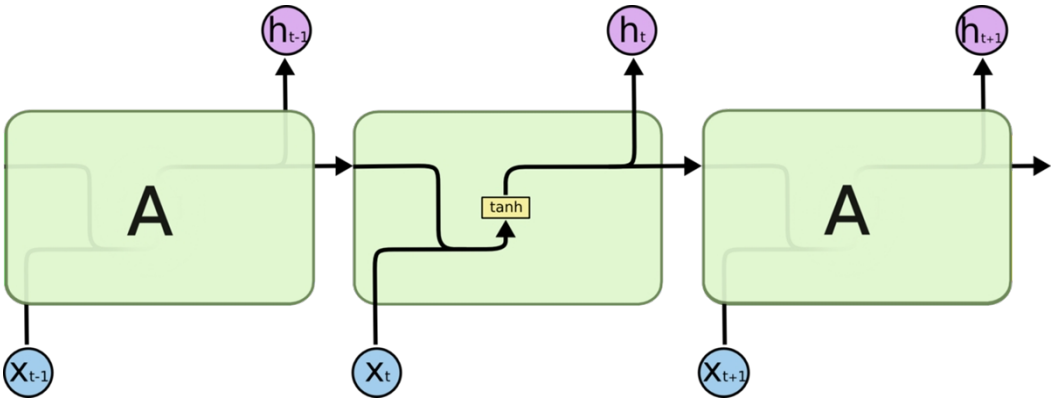
词向量模型分类器与分类技术

词向量模型分类器主要采用 2 种方案
本文主要采用 Keras 的 LSTM 来实现

Keras 是一个高层神经网络 API，Keras 由纯 Python 编写而成并基 Tensorflow、Theano 以及 CNTK 后端。Keras 为支持快速实验而生，能够把你的 idea 迅速转换为结果，如果你有如下需求，请选择 Keras：

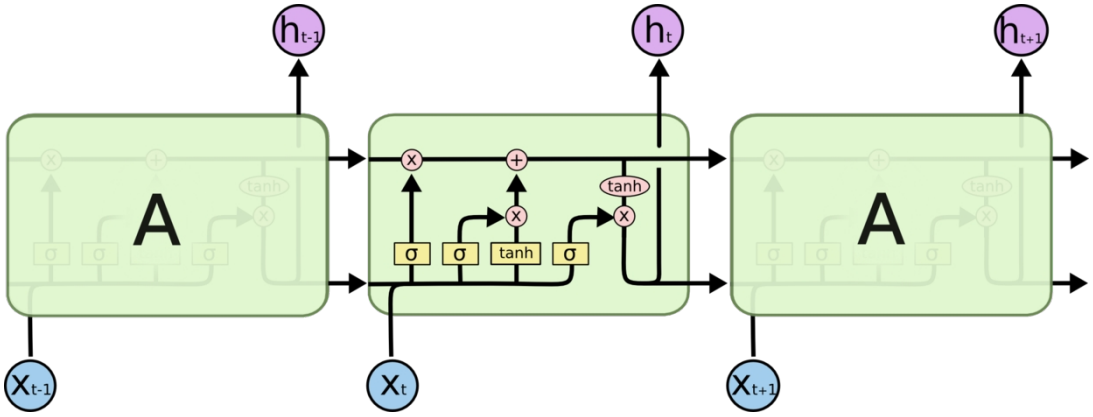
1. 简易和快速的原型设计（keras 具有高度模块化，极简，和可扩充特性）
2. 支持 CNN 和 RNN，或二者的结合
3. 无缝 CPU 和 GPU 切换 [7]

LSTM 网络
Long Short Term Memory 网络—— 一般就叫做 LSTM ——是一种 RNN 特殊的类型，可以学习长期依赖信息。LSTM 由 Hochreiter & Schmidhuber (1997)提出，并在近期被 Alex Graves 进行了改良和推广。在很多问题，LSTM 都取得相当巨大的成功，并得到了广泛的使用。
LSTM 通过刻意的设计来避免长期依赖问题。记住长期的信息在实践中是 LSTM 的默认行为，而非需要付出很大代价才能获得的能力！
所有 RNN 都具有一种重复神经网络模块的链式的形式。在标准的 RNN 中，这个重复的模块只有一个非常简单的结构，例如一个 tanh 层。



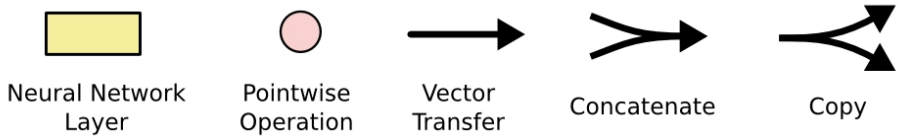
标准 RNN 中的重复模块包含单一的层

LSTM 同样是这样的结构，但是重复的模块拥有一个不同的结构。不同于 单一神经网络层，这里是有四个，以一种非常特殊的方式进行交互。



LSTM 中的重复模块包含四个交互的层

不必担心这里的细节。我们会一步一步地剖析 LSTM 解析图。现在，我们先来熟悉一下图中使用的各种元素的图标。



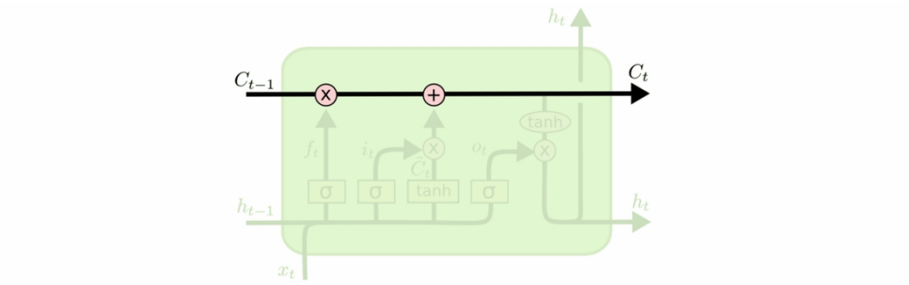
LSTM 中的图标

在上面的图例中，每一条黑线传输着一整个向量，从一个节点的输出到其他节点的输入。粉色的圈代表 pointwise 的操作，诸如向量的和，而黄色的矩阵就是学习到的神经网络层。合在一起的线表示向量的连接，分开的线表示内容被复制，然后分发到不同的位置。

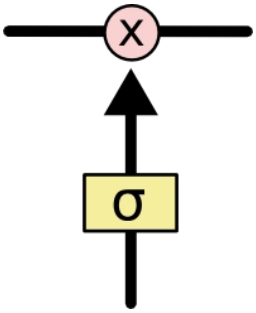
LSTM 的核心思想

LSTM 的关键就是细胞状态，水平线在图上方贯穿运行。

细胞状态类似于传送带。直接在整个链上运行，只有一些少量的线性交互。信息在上面流传保持不变会很容易。



LSTM 有通过精心设计的称作为“门”的结构来去除或者增加信息到细胞状态的能力。门是一种让信息选择式通过的方法。他们包含一个 sigmoid 神经网络层和一个 pointwise 乘法操作。



Sigmoid 层输出 0 到 1 之间的数值，描述每个部分有多少量可以通过。0 代表“不许任何量通过”，1 就指“允许任意量通过”！
LSTM 拥有三个门，来保护和控制细胞状态。 [8]

基准模型

项目中给定了如下 4 个模型作为参考

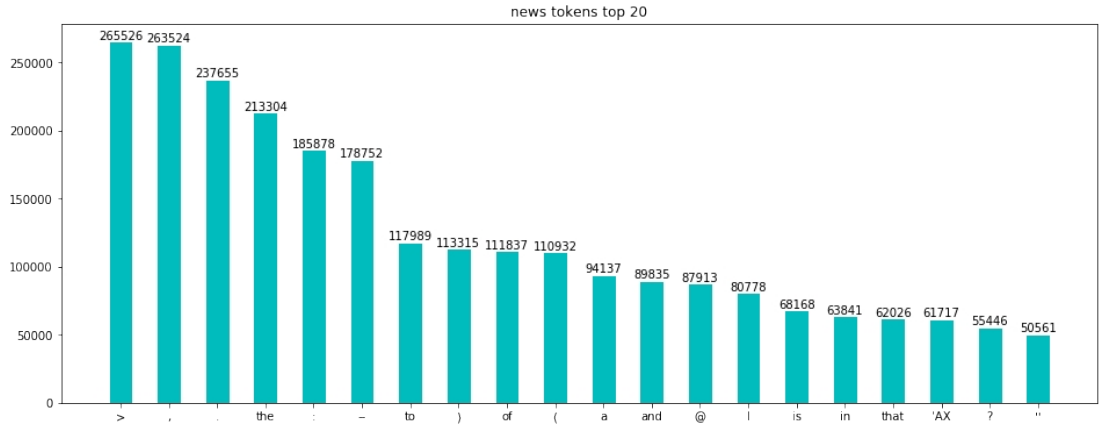
- 决策树模型
- 支持矢量机(SVM)模型
- 朴素贝叶斯模型
- 神经网络模型

前 3 种模型都是深度学习之前就已经存在的模型了，对于计算机的性能要求不是很高，尤其是对于我的笔记本电脑来说还可以承受， 神经网络模型做了几个回合，发现需要的时间几十倍的提高，这个对于毕业来说，已然成为很大的问题。因此项目中主要设定的目标对采用非机器学习的算法来实现，识别率超过 80%就可以。 深度学习的模型能超过 70%就可以。

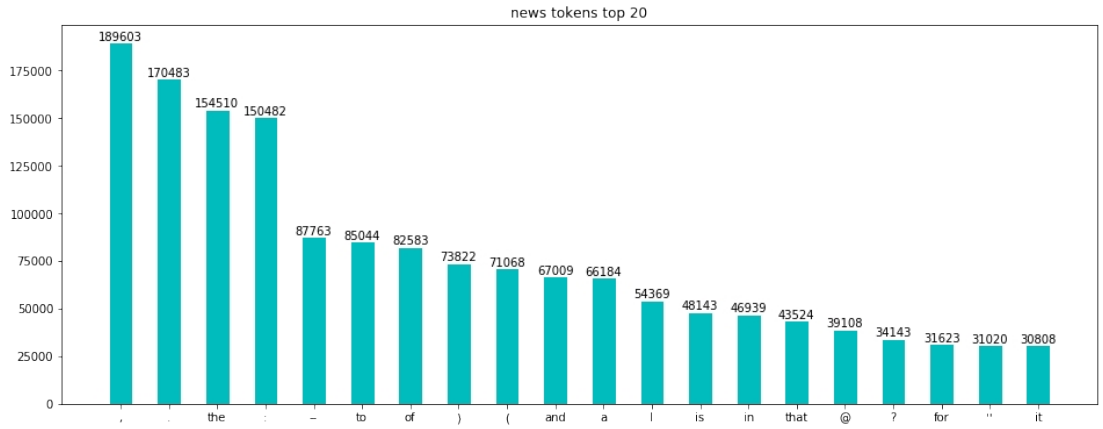
III. 方法

数据预处理

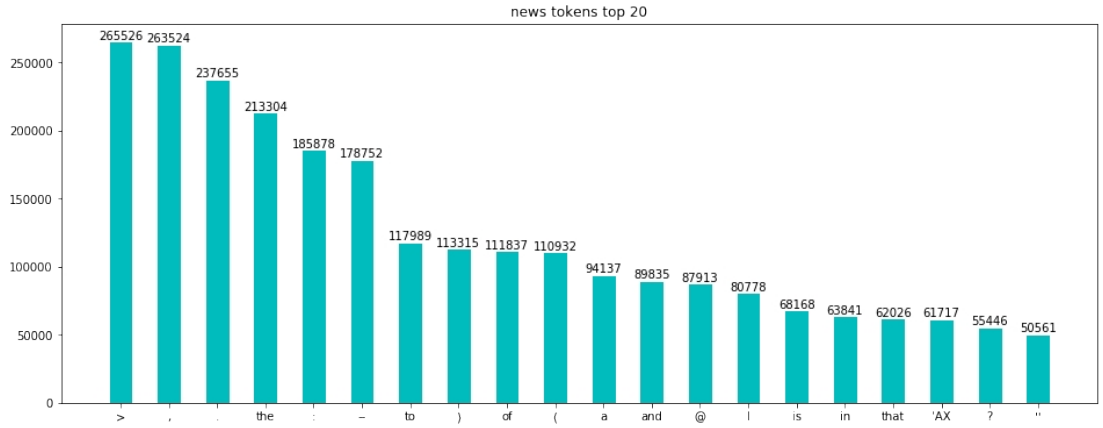
开始时 全部的特征数如下图



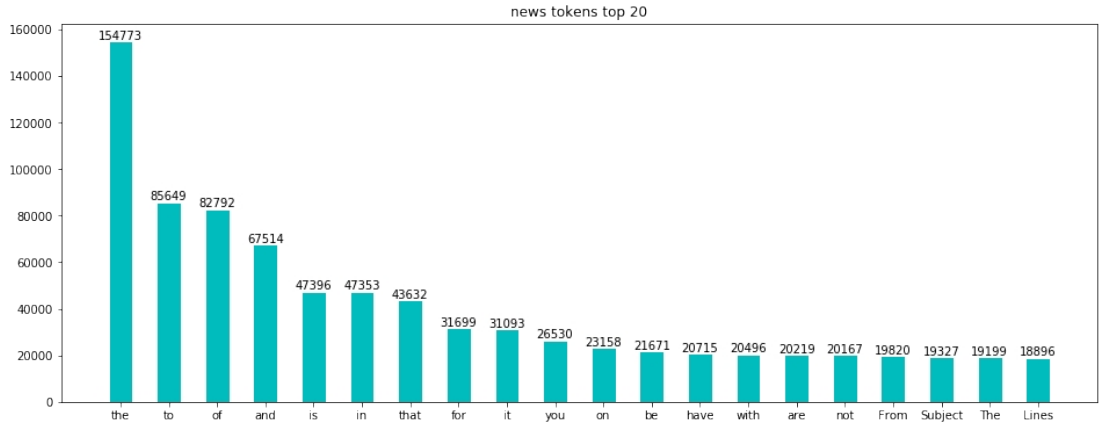
删除 footer 和 quotes 后，剩余的特征数，如下图：



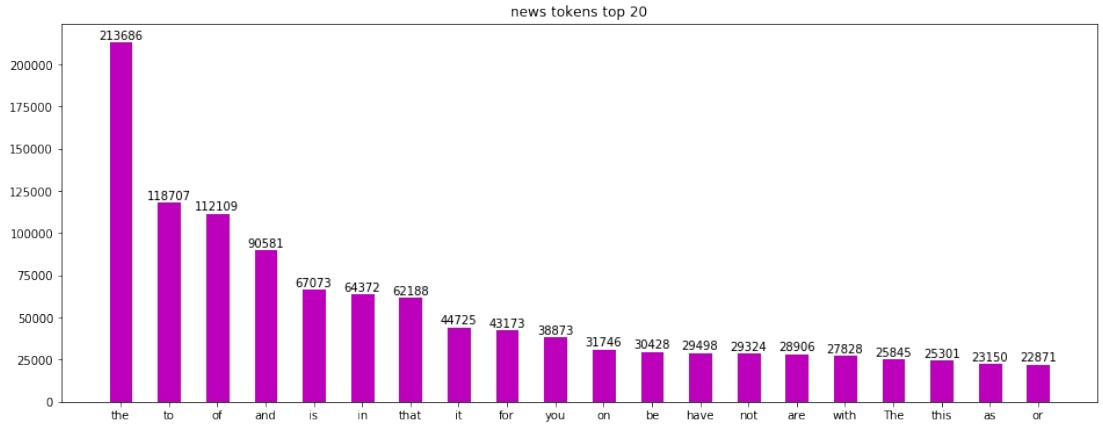
不删除 footer 和 quotes， 剩余的特征数（和开始的一样）如下所示：



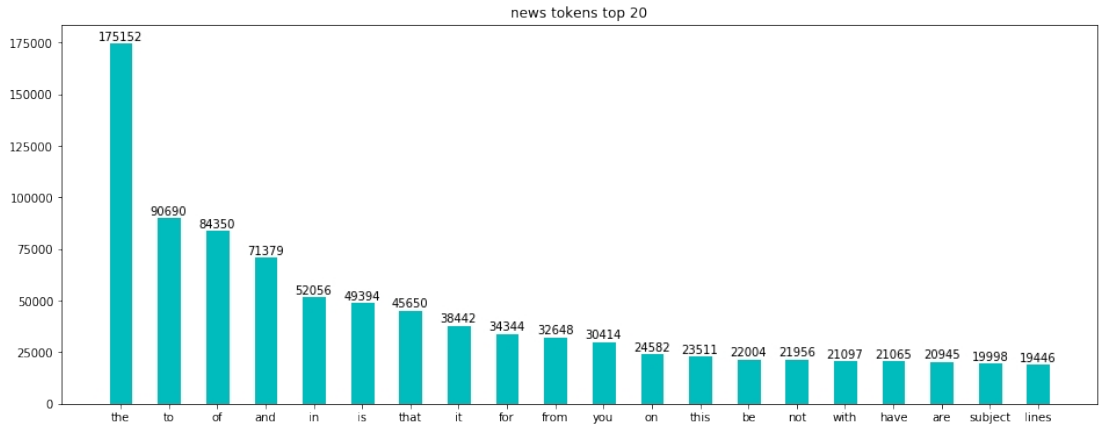
删除 footer 和 quotes，再删除标点符号，特殊符号等后， 剩余特征数如下图



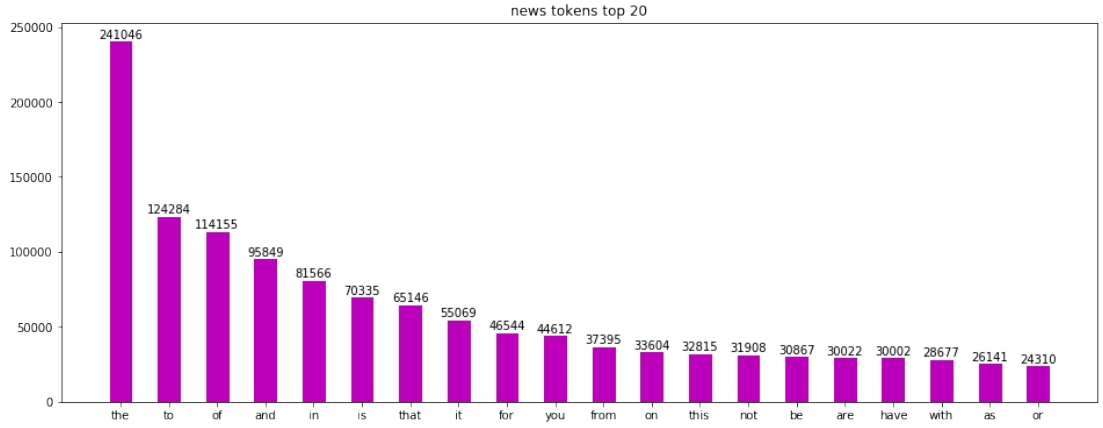
不删除 footer 和 quotes，再删除标点符合，特殊符号等后， 剩余特征数如下图



删除 footer&quotes，再字母大小写处理：剩余特征数如下图



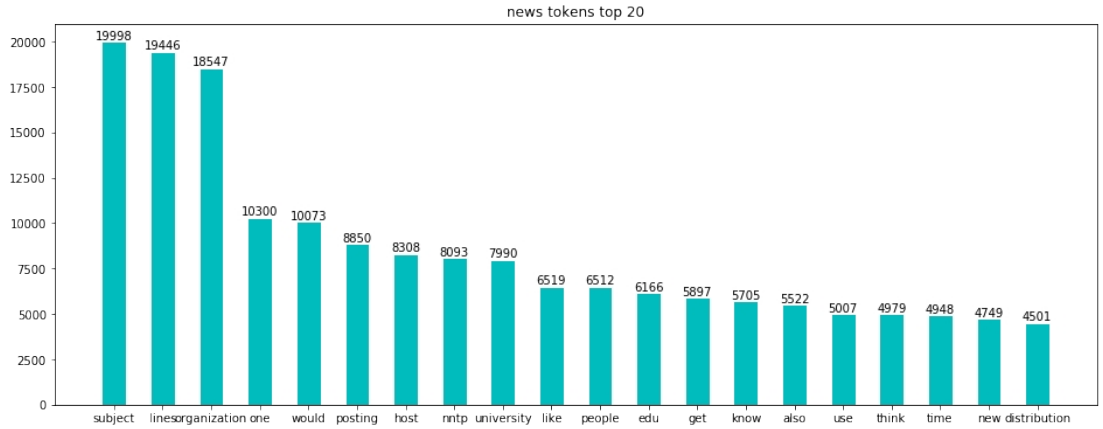
不删除 footer&quotes，再字母大小写处理：剩余特征数如下图



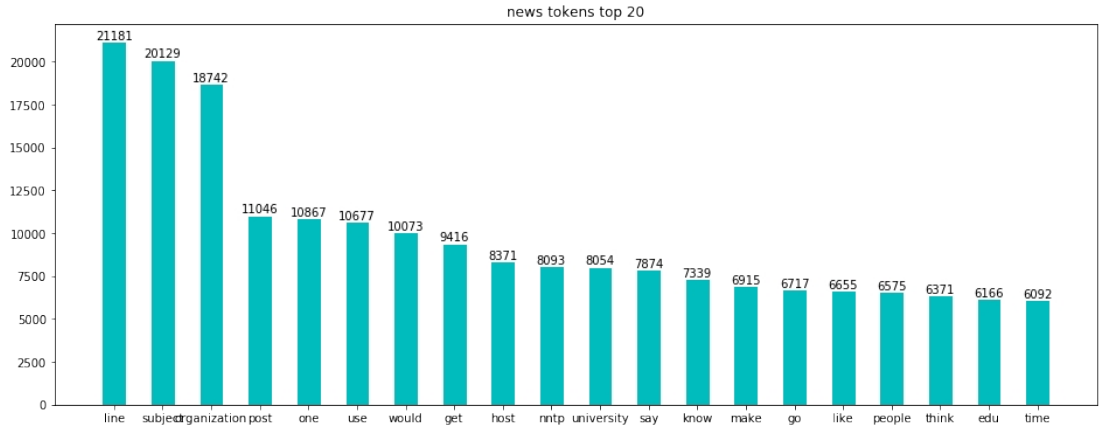
删除停用词后特征数如下图：

停用词列表如下：

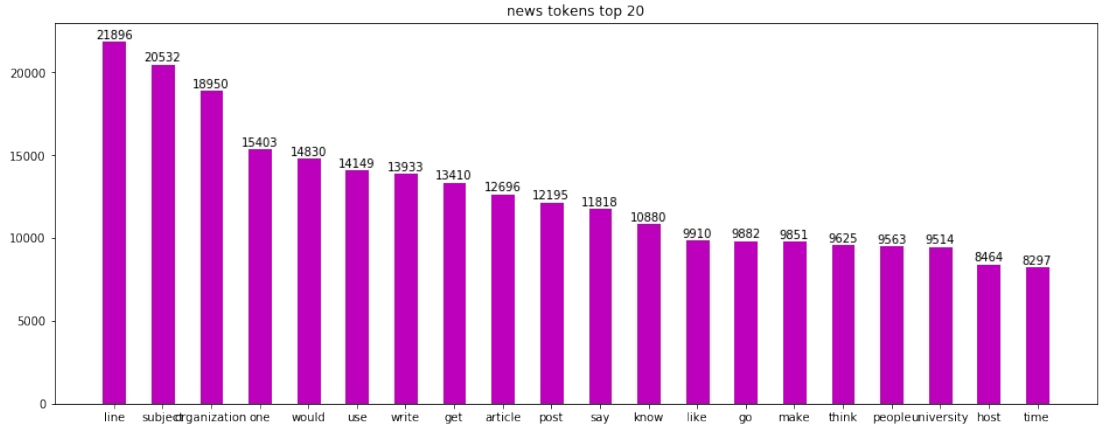
```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll",
"you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's",
'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themse
lves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is',
'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did',
'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at',
'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after'
, 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again',
'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same',
'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've",
'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',
"didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn'
t", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn
', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```



删除 footer&quotes，再词根还原后 特征数如下图：



不删除 footer&quotes，词根还原后 特征数如下图



本文也尝试了删除出现频率是 1 的单词，但在实际实施的时候，发现运行时间太长，因而在本项目中，暂且没有使用这个方法。

另外，我也对比了只删除 footer& quotes，和不删除 header，footer，quotes 的所有处理过程中的对比特征图，最后处理后，2 种方式的特征数都明显地变少了，但 不删除 header，footer，quotes 剩余的特征更多一下。
如下图所示：

图中对应的处理步骤 如下：

0： 开始
1： 删除 header,footer， quotes 数据
2： 数据清理
3： 转小写
4： 去掉停用词
5： 词根统一

执行过程

词袋子模型执行分类算法

首先，用函数 train_test_split 对数据集进行划分，数据集划分，将数据集按照 1：4 的比例划分，也就是 1/5 = 0.2； 80%的训练数据，20%的测试数据；训练数据中，也按照 1：4 划分，分成 4 份的训练数据和验证数据， 划分的结果如下： 训练集 12060， 验证集 3016，测试集 3770；

使用词袋子模型中的 CountVectorizer 对数据进行表示， 然后通过 fit_transform 函数，先拟合和标准化， 然后分别使用 DecisionTreeClassifier（DTC），支持向量机模型 SVC，朴素贝叶斯模型 MNB 分别进行训练和评估。

接下来使用词袋子模型中的 TfidfVectorizer 对数据进行特征表示， 同样使用上面 3 种模型分别进行训练和评估

词向量模型执行分类算法

例句:
Jane wants to go to Shenzhen.
Bob wants to go to Shanghai.

词向量模型是考虑词语位置关系的一种模型。通过大量语料的训练，将每一个词语映射到高维度（几千、几万维以上）的向量当中，通过求余弦的方式，可以判断两个词语之间的关系，例如例句中的 Jane 和 Bob 在词向量模型中，他们的余弦值可能就接近 1，因为这两个都是人名，Shenzhen 和 Bob 的余弦值可能就接近 0，因为一个是人名一个是地名。

现在常用 word2vec 构成词向量模型，它的底层采用基于 CBOW 和 Skip-Gram 算法的神经网络模型。

1. CBOW 模型

CBOW 模型的训练输入是某一个特征词的上下文相关的词对应的词向量，而输出就是这特定的一个词的词向量。比如上面的第一句话，将上下文大小取值为 2，特定的这个词是"go"，也就是我们需要的输出词向量,上下文对应的词有 4 个，前后各 2 个，这 4 个词是我们模型的输入。由于 CBOW 使用的是词袋模型，因此这 4 个词都是平等的，也就是不考虑他们和我们关注的词之间的距离大小，只要在我们上下文之内即可。

这样我们这个 CBOW 的例子里，我们的输入是 4 个词向量，输出是所有词的 softmax 概率（训练的目标是期望训练样本特定词对应的 softmax 概率最大），对应的 CBOW 神经网络模型输入层有 4 个神经元，输出层有词汇表大小个神经元。隐藏层的神经元个数我们可以自己指定。通过 DNN 的反向传播算法，我们

可以求出 DNN 模型的参数，同时得到所有的词对应的词向量。这样当我们有新的需求，要求出某 4 个词对应的最可能的输出中心词时，我们可以通过一次 DNN 前向传播算法并通过 softmax 激活函数找到概率最大的词对应的神经元即可。

2.Skip-Gram 模型

Skip-Gram 模型和 CBOW 的思路是反着来的，即输入是特定的一个词的词向量，而输出是特定词对应的上下文词向量。还是上面的例子，我们的上下文大小取值为 2， 特定的这个词"go"是我们的输入，而这 4 个上下文词是我们的输出。

这样我们这个 Skip-Gram 的例子里，我们的输入是特定词， 输出是 softmax 概率排前 4 的 4 个词，对应的 Skip-Gram 神经网络模型输入层有 1 个神经元，输出层有词汇表大小个神经元。隐藏层的神经元个数我们可以自己指定。通过 DNN 的反向传播算法，我们可以求出 DNN 模型的参数，同时得到所有的词对应的词向量。这样当我们有新的需求，要求出某 1 个词对应的最可能的 4 个上下文词时，我们可以通过一次 DNN 前向传播算法得到概率大小排前 4 的 softmax 概率对应的神经元所对应的词即可。

词向量模型突出特点：

在词向量模型中，词向量与词向量之间有这非常特殊的特性。例如现在存在国王、男生、女人、皇后四个词向量，那么一个完善的词向量模型，就存在“国王-男人+女人=皇后”这样的关系。［9］

完善

词袋子模型，报告中只考虑了论文指导中给出的决策树分类器 DecisionTreeClassifier DTC，支持向量机分类器 SVMC，朴素贝叶斯分类器 MultinomialNB MNB 等，其实参考 slearn 还有很多其他的算法可以在词袋子模型上进行尝试，比如 RandomForestClassifier, Ridge Classifier 等， 这里由于时间关系，就不做其他的尝试了，可以参考如下链接：
http://scikit-learn.org/stable/auto_examples/text/plot_document_classification_20newsgroups.html#sphx-glr-auto-examples-text-plot-document-classification-20newsgroups-py

在对词向量进行训练时，只使用了 LSTM 网络，keras 还支持其他的 CNN, RNN 网络，由于硬件设备的原因跑不出结果，也可能时算法存在问题，在本项目中，只使用了 LSTM 网络，这个工作还需要继续完善。另外，LSTM 模型也需要继续进行调优工作，包括调整卷积层个数，dropout 的比例， epoch 的训练次数，都需要进行探索。

整体上结果基本上达到了预期的目标。

IV. 结果

模型的评价与验证

词袋子模型进行预测的结果如下：

	DTC	SVC	MNB
count_accuracy_score	0.6021	0.776	0.83
count_train_time	13.342	266.476	0.093
count_loss	13.23	0.891	1.89
tfidf_accuracy_score	0.60	0.868	0.838
tfidf_train_time	21.402	654.052	0.087
tfidf_loss	13.742	0.4478	1.1879

通过观察对比，发现: DTC 决策树模型表现很差，loss 很高，accuracy_score 很低； 而 svc 模型 loss 很低，accuracy_score 很高，表现最好。SVC 虽然表现最好，但是用的时间也是最多。 MNB 模型比 SVC 稍微差一些。accuracy_score 和 loss 都稍微低一些，但是用的时间确实最短的，这个效果是最好的。用最短的时间得到一个基本上很好的结果。 另外，采用 TfidfVectorizer【图中实线表示】 表示文本，要比 countVectorizer 【用虚线表示的】要好，tfidf 的 accuracy_score 要比 count 要高。 loss 低一些。

决策树表现的不是很好，原因是文本中的特征量比较大，决策树在处理大量特征的数据时容易发生过拟合的现象。

之前的结果是在删除 footer 和 quotes 的基础上得到的，现在不删除 footer 和 quotes 后，得到的结果如下图所示：

	DTC	SVC	MNB
count_accuracy_score	0.6611	0.852	0.8700
count_train_time	14.923	349.951	0.183
count_loss	11.70	0.634	1.85
tfidf_accuracy_score	0.63	0.9157	0.8730
tfidf_train_time	19.434	731.577	0.105
tfidf_loss	12.505	0.3076	0.9509

对比发现，删除 footer 和 quotes，是错误的，保留 header，footer 和 quotes 的情况下，DTC，SVC，MNB 各个模型的 accuracy_score 都比删除 footer

和 quotes 要高很多。 所以数据处理非常重要，当你选择某些部分或者不选择某些部分的时候，一定要做一个对比，这样在以后的处理过程中，才能得到更好的结果。

删除 footer 和 quotes 时，采用 MNB 朴素贝叶斯分类算法进行优化，优化的数据如下截图所示：

```
优化前-    accuracy_score    0.8382    Log_loss    0.4487
优化后-                      0.8657                0.4487
```

通过 GridSearchCV 网格搜索调优后，发现 accuracy_score 有了稍微的提高，但提高的幅度不大。 没有变化。

不删除 footer 和 quotes， 即保留 header，footer 和 quotes 所有的数据时，得到的结果如下：

采用 SVC linear 类算法并进行 gridsearchcv 优化，优化的数据如下截图所示：

在验证集上， svc 优化前后的数据如下：

```
优化前------
accuracy_score on validation data: 0.9158
Log_loss_val    on the validation data: 0.3049
优化后-----
accuracy_score on the validation data: 0.9171
log_loss_val  on the validation data: 0.3049
now best_params_: {'C': 10}
```

虽然在验证集上，svc 有所提高，但是

```
优化前------
Accuracy_score on test data: 0.9019
Log_loss_val on the test data: 0.3376
优化后------
Accuracy_score on test data: 0.9019
Log_loss_val on the test data: 0.3376
```

测试数据集上，svc（c=1） 和 优化后的 svc（c=10）准确率都是 90.19%, Log_loss 为 0.3376, 没有提高；
svc 运行速度过慢，所以参数只选了一个 c=10，没选择更多的参数测试。

并且，测试集上的结果不如验证集的结果更好，可能发生了过拟合。

在验证集上， MNB 优化前后的数据如下：

```
grid train_time: 19.093 s
优化前------
accuracy_score on validation data: 0.8730
Log_loss_val    on the validation data: 0.3028
优化后-----
accuracy_score on the validation data: 0.9108
log_loss_val  on the validation data: 0.3028
now best_params_: {'alpha': 0.01, 'fit_prior': False}
```

对比发现，在 MNB 验证集上，优化前后还是有所提高的。
可喜的是，验证集上提高了，在测试集上的结果也提高了，如下：

```
优化前------
Accuracy_score on test data: 0.8568
Log_loss_val on the test data: 0.9748
优化后-----
Accuracy_score on test data: 0.9066
Log_loss_val on the test data: 0.3170
```

相比而言，MNB 测试集的效果。
但是，测试集上的结果不如验证集的结果更好，也可能发生了过拟合。

最后，总结，通过包含 header，footer，和 quotes 数据， 分别使用 SVM 和 MNB 分类模型， 最后在测试集上，都达到了 90%以上的结果。
这远远超过了当初的基准模型 DTC 决策树分类模型。
不过，通过分析，决策树模型 在数据处理后，得到的效果也很不理想，这也是导致基准模型和这个结果差距比较大的原因。

词向量模型进行预测的结果如下

	accuracy_score	log_loss
测试集	0.7488	0.7875
验证集	0.7599	0.7492

在验证集上，accuracy_score 达到了 0.7599，log loss 损失为 0.7492。 在测试集上，accuracy_score 是 0.7488， log_loss 为 0.7875，要比测试结果低一些，硬件设备限制，没有进行其他方案的尝试， 相对来说， 这个结果还可以。

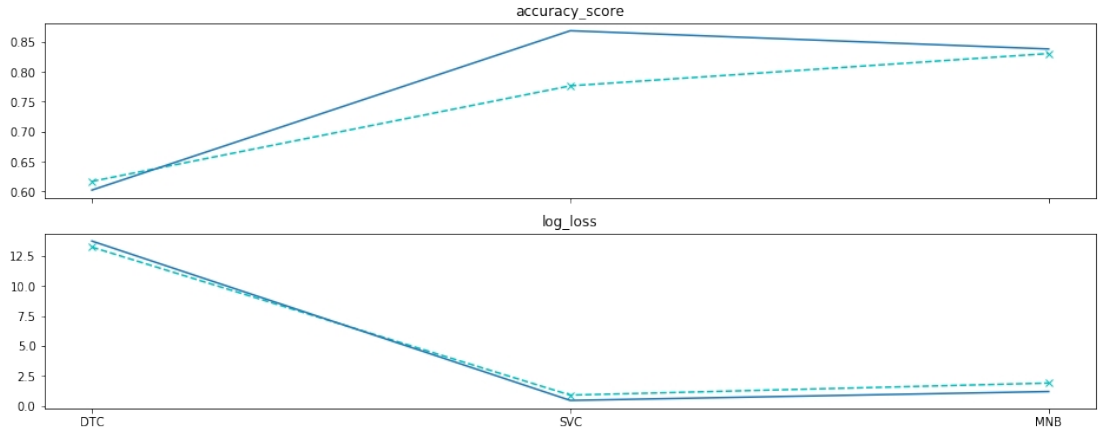
合理性分析

从词袋子模型 和词词向量模型的比较可以看出，在传统的分类算法领域，词袋子模型的分类结果还是非常不错的。虽然在本文中，神经网网络训练的结果不如传统分类算法，可能主要是 由于本文中无法进行更多 epoch 的训练有关。 如果平时实验要快速得出结果，可以考虑词袋子模型和传统的分类算法。如果在实际的工程中，既要追究精度，又要追求速度，那么可能需要把这两种模型和分类算法整合到一起，分别发挥各自的优势。

V. 项目结论

结果可视化

词袋子模型，使用常规的分类器 DTC 决策树，SVC 支持向量机，MNB 朴素贝叶斯等，对结果进行了如下的对比。



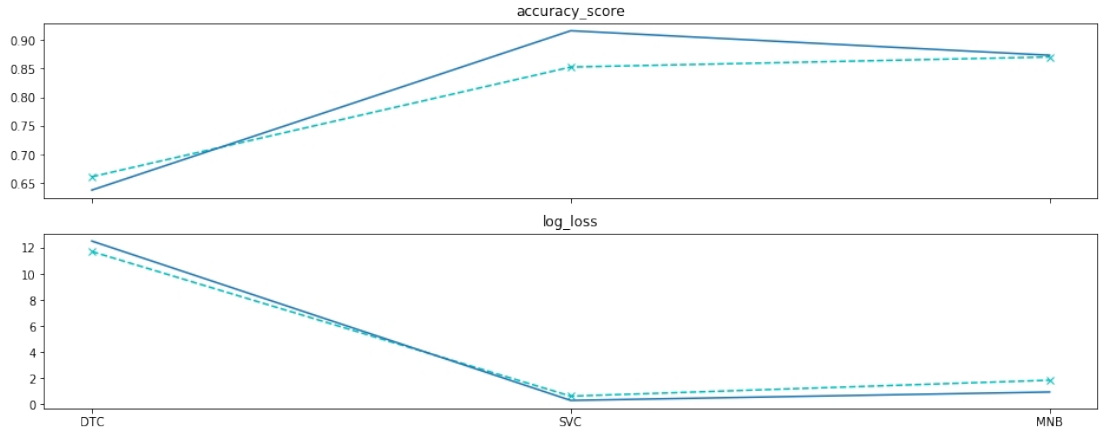
	DTC	SVC	MNB
count_accuracy_score	0.6021	0.776	0.83
count_train_time	13.342	266.476	0.093
count_loss	13.23	0.891	1.89
tfidf_accuracy_score	0.60	0.868	0.838
tfidf_train_time	21.402	654.052	0.087
tfidf_loss	13.742	0.4478	1.1879

通过观察对比，发现：DTC 决策树模型表现很差，loss 很高，accuracy_score 很低； 而 svc 模型 loss 很低，accuracy_score 很高，表现最好。 MNB 模型比 SVC 稍微差一些。 另外，采用 TfidfVectorizer【图中实线表示】 表示文本，要比 countVectorizer 【用虚线表示的】要好，tfidf 的 accuracy_score 要比 count 要高。 loss 低一些。

删除 footer 和 quotes 时，优化后的模型，

优化前-	accuracy_score	0.8382	Log_loss	0.4487
优化后-		0.8657		0.4487

不删除 footer 和 quotes 后，即保留 header，footer 及 quotes 得到的数据如下：



	DTC	SVC	MNB
count_accuracy_score	0.6611	0.852	0.8700
count_train_time	14.923	349.951	0.183
count_loss	11.70	0.634	1.85
tfidf_accuracy_score	0.63	0.9157	0.8730
tfidf_train_time	19.434	731.577	0.105
tfidf_loss	12.505	0.3076	0.9509

	指标	验证集	测试集
优化前 MNB	accuracy_score	0.873	0.8568
	Log_loss_val	0.3028	0.9748
优化前 SVC	accuracy_score	0.9158	0.9019
	Log_loss_val	0.3049	0.3376
优化后 MNB	accuracy_score	0.9108	0.9066
	Log_loss_val	0.3028	0.317
优化后 SVC	accuracy_score	0.9171	0.9019
	Log_loss_val	0.3049	0.3376

和删除 footer 和 quotes 之前的比较， 优化后的 accuracy_score 又得到了提高。

总结如下，通过包含 header，footer，和 quotes 数据， 分别使用 SVM 和 MNB 分类模型，并且采用 gridsearchcv 优化， 最后在测试集上，都达到了 90%以上的结果。

这个结果远远超过了当初的基准模型 DTC 决策树分类模型。不过，通过分析，决策树模型 在数据处理后，得到的效果也很不理想，这也是导致基准模型和这个结果差距比较大的原因。

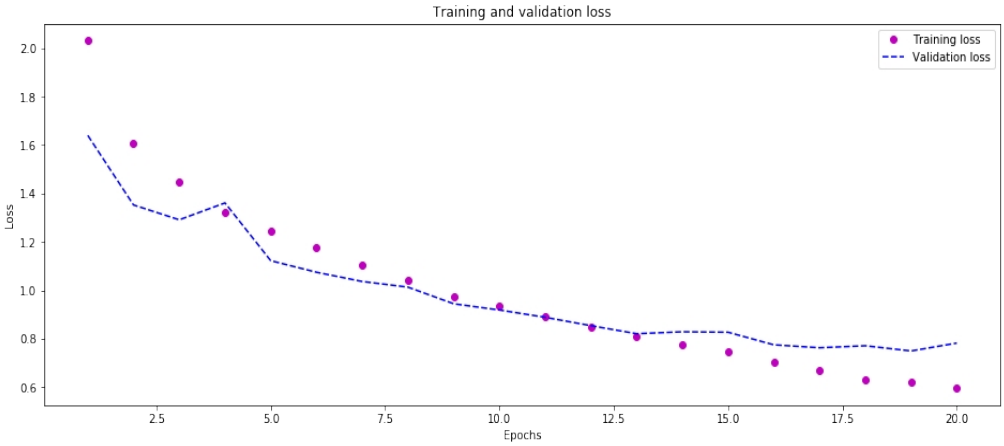
词向量模型: 采用 LSTM 神经网络分类算法

使用 LSTM 分类算法的 summary 函数，得到的 LSTM 模型的结构信息如下：

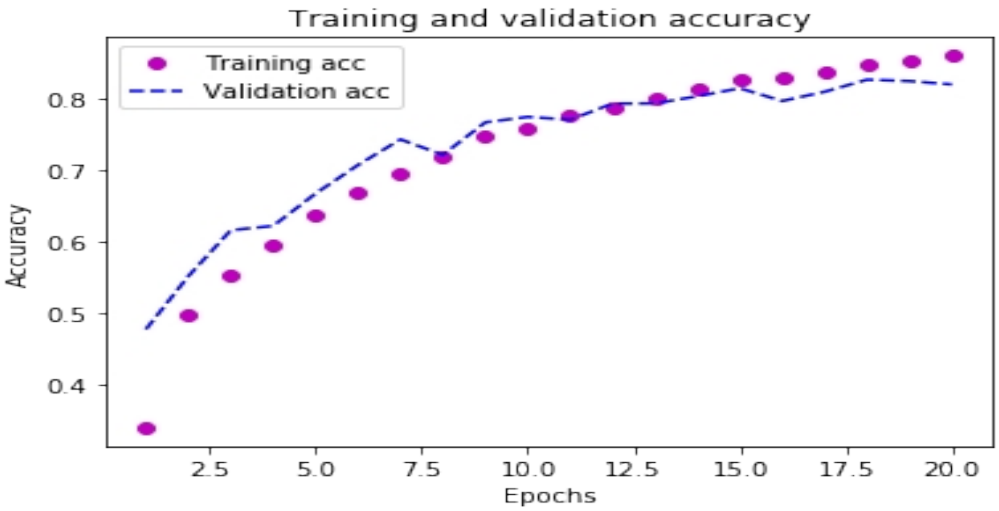
Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 500, 120)	11781840
lstm_1 (LSTM)	(None, 200)	256800
dropout_1 (Dropout)	(None, 200)	0
dense_1 (Dense)	(None, 20)	4020

Total params: 12,042,660
Trainable params: 260,820
Non-trainable params: 11,781,840

Loss 曲线的变化 如下图所示：



准确率 Accuracy_score 曲线在测试集和验证集上的变化 如下图所示：



通过图像分析，LSTM 模型在前 15 轮的表现很不错；accuracy_score 在测试集和验证集上在上升， 而 log loss 都在下降；15 轮之后，验证集开始出现过拟合， loss 不在明显变化。

删除 footer 和 quotes 后， 最后词向量表示，用 LSTM 进行测试和预测，得到的结果如下图：

	accuracy_score	log_loss
测试集	0.7488	0.7875
验证集	0.7599	0.7492

不删除 footer，quotes 时，得到的结果如下：

	accuracy_score	log_loss
不包含：footer，quotes		
测试集	0.7488	0.7875
验证集	0.7599	0.7492
包含：header，footer，quotes		
测试集	0.5727	0.8074
验证集	0.5357	0.8246

在删除 footer 和 quotes 后， 测试集上，accuracy_score 是 0.7488，log_loss 为 0.7875，

在包含 footer 和 quotes 的结果如下：

accuracy_score 是 0.8074， log_loss 为 0.5727，

总结： header，footer 和 quotes 都不能删除。

另外，在不删除 footer 和 quotes 的情况下， 测试结果要比验证的结果低一些，硬件设备限制，没有进行其他方案的尝试， 相对来说， 这个结果还可以。超过了当时的基准目标 70%。

对项目的思考

通过本项目的实施，对 NLP 的整个流程有了非常深入的理解。 对文本数据的处理有一些体会。 比如，英文中，要考虑大小写， 词根是否统一等。 这个在中文中就不需要考虑。 另外对文本数据的预处理要讲究先后顺序，举个例子，你不能先去掉停用词，要把去掉停用词放在单词的大小写处理之后， 因为停用词 中都时小写表示，所以，要先变成统一的小写，再去去掉停用词。 类似的道理，词根的统一，标点符号的删除等，都要细细琢磨。 本文只是简单的分类，涉及到情感分析，如果要考虑情感分析的化，标点符号等的处理规则估计要重新设计。

其次，还对文本数据集的模型比如，词袋子模型还是 词向量模型有了更深层次的理解。 之前，这些模型的理解只是表面上的，没有深入的追究为什么什么叫词袋子，为什么叫词向量，现在感觉很有道理。

另外，感觉高纬度的文本处理和图片的处理有些类似，只是基本的元素不一样，一个是像素，一个时单独的词汇。从这个角度说，是不是处理图片的某些算法可以修改或形象化的用到文本的分类上，或者文本的分类的是否反过来可以用到图像的识别上，这些非常值得去探索。

需要作出的改进

文本分类，不管时词袋子模型还词向量模型，都存在一个特征维度很大的问题，这个在本文中，在数据处理之前数据的维度已经很大了，后来做了一些数据处理，把数据集的特征维度降下来了， 但我认为数据集的特征维度是否可以继续减少？ 这个问题从一开始就困扰着我，这个需要再继续研究研究。

这个问题，有了新的认识，通过添加 footer 和 quotes 这 2 个部分的数据，accuracy_score 反而提高了，虽然特征数也提高了，但是 score 也提高了，在这种情况下看， 数据集的特征维度高也是可以接受的。

另外，本项目没有尝试其他的 RNN 或 CNN 网站，这个至少训练一个其他类型的神经网络进行对比和观察，这样才能更好的方便以后做出选择。

在助教的指引下，了解了 Attention 机制，这个非常不错，以后要用起来。

Attention 的作用如下：在 RNN 的文本分类模型中，可以把 RNN 看成一个 encoder，将需要被分类的文本表示为一个 dense vector，再使用全连接层与

softmax 输出各类别的概率。

在具体的文本的表示上，可以将 RNN 最后一个时刻的输出作为文本的表示，也可以综合考虑每个时刻的的输出，将它们合并为一个向量。在 tagging 与 classication 的任务中常用双向 RNN(下文写作 BIRNN)，每个时刻的输出向量可以理解为这个时刻的输入词在上下文的语境中对当前任务的一个贡献。

根据人类的阅读习惯进行思考，我们在阅读的时候，注意力通常不会平均分配在文本中的每个词。再回到上面的文本表示，如果直接将每个时刻的输出向量相加再平均，就等于认为每个输入词对于文本表示的贡献是相等的，但实际情况往往不是这样，比如在情感分析中，文本中地名、人名这些词应该占有更小的权重，而情感类词汇应该享有更大的权重。

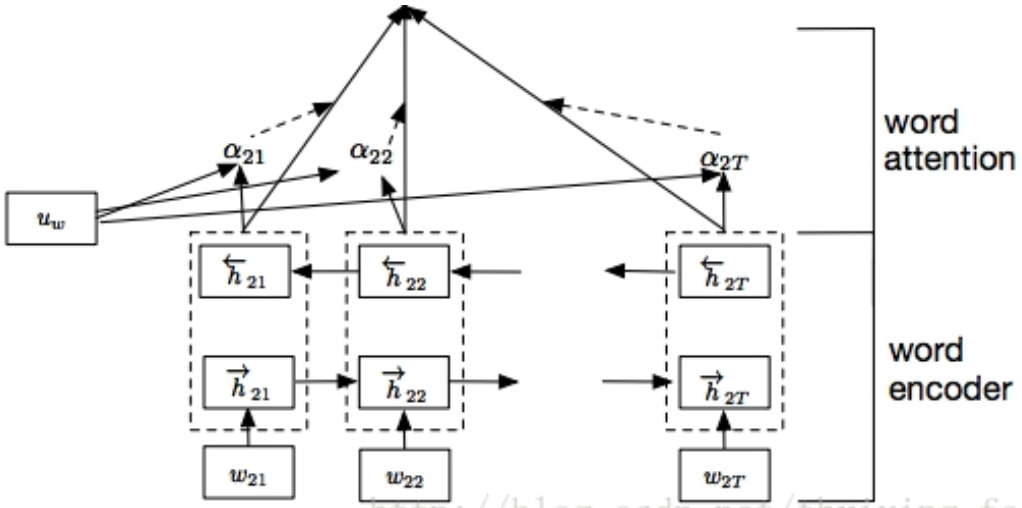
所以在合并这些输出向量时，希望可以将注意力集中在那些对当前任务更重要的向量上。也就是给他们都分配一个权值，将所有的输出向量加权平均。假设输出向量为 ht，权值为 α t，则合并后的表示为

$$s = \sum_t \alpha_t h_t$$

上文所说的为 BIRNN 的每个输出向量分配不同权值，使得模型可以将注意力集中在重点几个词，降低其他无关词的作用的机制就是 Attention 机制。使用了 Attention 机制可以使得文本表示的结果在当前的任务中更合理。

Attention 原理

上文说到需要给 BIRNN 的每个输出分配权重，如何分配就是 Attention 的原理，用一张结构图加三个公式应该就可以解释清楚了。



$$u_t = \tanh(W_w h_t + b_w) \tag{1}$$

$$\alpha_t = \frac{\exp(u_t^T u_w)}{\sum_t \exp(u_t^T u_w)} \tag{2}$$

$$s = \sum_t \alpha_t h_t \tag{3}$$

公式(1)中的 Ww 与 bw 为 Attention 的权重与 bias，在实现的时候也要设置 attention 的 size，不过也可以简单的令它们等于 BIRNN 的输出向量的 size。

公式(2)中的 uw 也是需要设置的权重，公式(2)其实也就是对所有 uTtuw 结果的 softmax。

公式(3)即是将计算出的 α t 作为各时刻输出的权值，对它们加权求和表示为一个向量。[10]

参考文献

1. http://qwone.com/~jason/20Newsgroups/
2. http://www.cnblogs.com/platero/archive/2012/12/03/2800251.html
3. udacity 毕业指导文档
4. sklearn 文档 http://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_extraction.text
5. https://www.sohu.com/a/128794834_211120
6. https://blog.csdn.net/liugan528/article/details/79448379
7. https://keras-cn.readthedocs.io/en/latest/
8. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
9. https://blog.csdn.net/sinat_36521655/article/details/79993369?utm_source=copy
10. https://blog.csdn.net/thriving_fcl/article/details/73381217?utm_source=copy