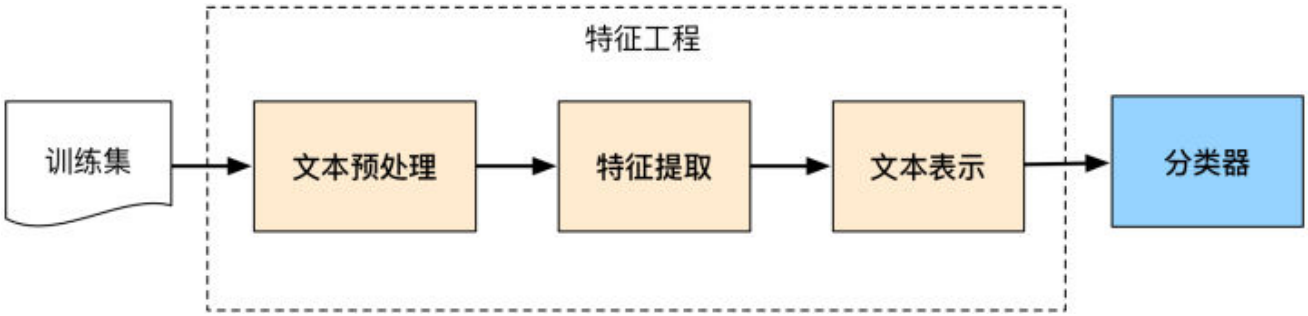
NLP 文本分类

一、传统文本分类方法

文本分类问题算是自然语言处理领域中一个非常经典的问题了，相关研究最早可以追溯到上世纪50年代，当时是通过专家规则（Pattern）进行分类，甚至在80年代初一度发展到利用知识工程建立专家系统，这样做的好处是短平快的解决top问题，但显然天花板非常低，不仅费时费力，覆盖的范围和准确率都非常有限。

后来伴随着统计学习方法的发展，特别是90年代后互联网在线文本数量增长和机器学习学科的兴起，逐渐形成了一套解决大规模文本分类问题的经典玩法，这个阶段的主要套路是人工特征工程+浅层分类模型。训练文本分类器过程见下图：



整个文本分类问题就拆分成了特征工程和分类器两部分。

1.1 特征工程

特征工程在机器学习中往往是最耗时耗力的，但却极其的重要。抽象来讲，机器学习问题是把数据转换成信息再提炼到知识的过程，特征是“数据-->信息”的过程，决定了结果的上限，而分类器是“信息-->知识”的过程，则是去逼近这个上限。然而特征工程不同于分类器模型，不具备很强的通用性，往往需要结合对特征任务的理解。

文本分类问题所在的自然语言领域自然也有其特有的特征处理逻辑，传统分本分类任务大部分工作也在此处。文本特征工程分位文本预处理、特征提取、文本表示三个部分，最终目的是把文本转换成计算机可理解的格式，并封装足够用于分类的信息，即很强的特征表达能力。

1）文本预处理

文本预处理过程是在文本中提取关键词表示文本的过程，中文文本处理中主要包括文本分词和去停用词两个阶段。之所以进行分词，是因为很多研究表明特征粒度为词粒度远好于字粒度，其实很好理解，因为大部分分类算法不考虑词序信息，基于字粒度显然损失了过多“n-gram”信息。

具体到中文分词，不同于英文有天然的空格间隔，需要设计复杂的分词算法。传统算法主要有基于字符串匹配的正向/逆向/双向最大匹配；基于理解的句法和语义分析消歧；基于统计的互信息/CRF方法。近年来随着深度学习的应用，WordEmbedding + Bi-LSTM+CRF方法逐渐成为主流，本文重点在文本分类，就不展开了。而停止词是文本中一些高频的代词连词介词等对文本分类无意义的词，通常维护一个停用词表，特征提取过程中删除停用表中出现的词，本质上属于特征选择的一部分。

经过文本分词和去停止词之后，一个句子就变成了下图“ / ”分割的一个个关键词的形式：

```
夏装 / 雪纺 / 条纹 / 短袖 / t恤 / 女 / 春 / 半袖 / 衣服 / 夏天 / 中长款 / 大码 / 胖mm / 显瘦 / 上衣 / 夏
```

2）文本表示和特征提取

文本表示：

文本表示的目的是把文本预处理后的转换成计算机可理解的方式，是决定文本分类质量最重要的部分。传统做法常用词袋模型（BOW, Bag Of Words）或向量空间模型（Vector Space Model），最大的不足是忽略文本上下文关系，每个词之间彼此独立，并且无法表征语义信息。词袋模型的示例如下：

```
( 0, 0, 0, 0, .... , 1, ... 0, 0, 0, 0)
```

一般来说词库量至少都是百万级别，因此词袋模型有个两个最大的问题：高纬度、高稀疏性。词袋模型是向量空间模型的基础，因此向量空间模型通过特征项选择降低维度，通过特征权重计算增加稠密性。

特征提取：

向量空间模型的文本表示方法的特征提取对应特征项的选择和特征权重计算两部分。特征选择的基本思路是根据某个评价指标独立的对原始特征项（词项）进行评分排序，从中选择得分最高的一些特征项，过滤掉其余的特征项。常用的评价有文档频率、互信息、信息增益、χ^2统计量等。

特征权重主要是经典的TF-IDF方法及其扩展方法，主要思路是一个词的重要度与在类别内的词频成正比，与所有类别出现的次数成反比。

3）基于语义的文本表示

传统做法在文本表示方面除了向量空间模型，还有基于语义的文本表示方法，比如LDA主题模型、LSI/PLSI概率潜在语义索引等方法，一般认为这些方法得到的文本表示可以认为文档的深层表示，而word embedding文本分布式表示方法则是深度学习方法的重要基础，下文会展现。

1.2 分类器

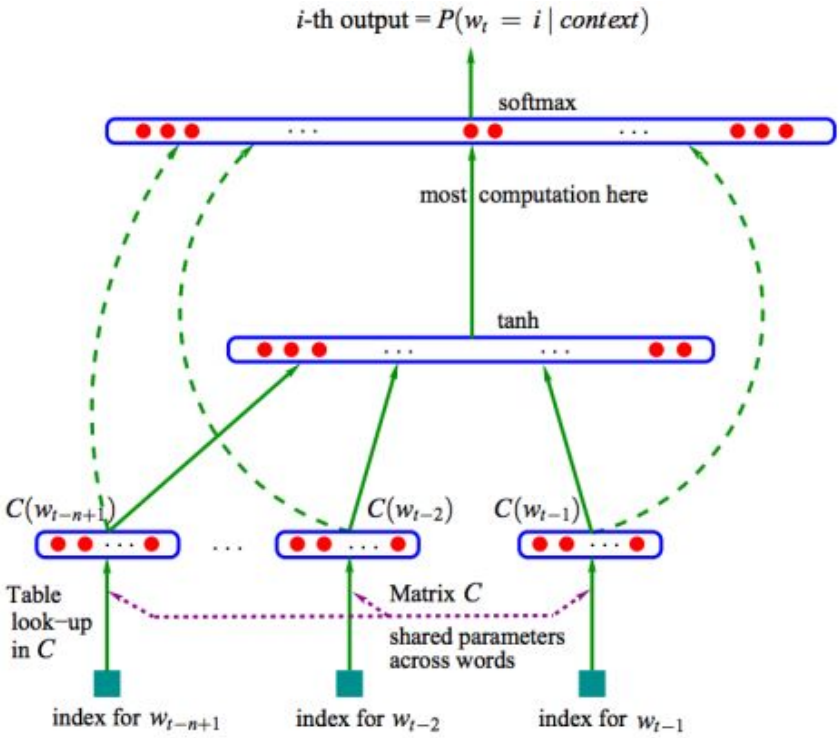
分类器基本都是统计分类方法了，基本上大部分机器学习方法都在文本分类领域有所应用，比如朴素贝叶斯分类算法（Naïve Bayes）、KNN、SVM、最大熵和神经网络等等。

二、深度学习 文本分类方法

上面介绍了传统的文本分类做法，传统做法主要问题的文本表示是高纬度高稀疏的，特征表达能力很弱，而且神经网络很不擅长对此类数据的处理；此外需要人工进行特征工程，成本很高。而深度学习最初在之所以图像和语音取得巨大成功，一个很重要的原因是图像和语音原始数据是连续和稠密的，有局部相关性，。应用深度学习解决大规模文本分类问题最重要的是解决文本表示，再利用CNN/RNN等网络结构自动获取特征表达能力，去掉繁杂的人工特征工程，端到端的解决问题。接下来会分别介绍：

2.1 文本的分布式表示：词向量（**word embedding**）

分布式表示（Distributed Representation）其实Hinton 最早在1986年就提出了，基本思想是将每个词表达成 n 维稠密、连续的实数向量，与之相对的one-hot encoding向量空间只有一个维度是1，其余都是0。分布式表示最大的优点是具备非常powerful的特征表达能力，比如 n 维向量每维 k 个值，可以表征 k^n 个概念。事实上，不管是神经网络的隐层，还是多个潜在变量的概率主题模型，都是应用分布式表示。下图是03年Bengio在 A Neural Probabilistic Language Model的网络结构：



这篇文章提出的神经网络语言模型（NNLM，Neural Probabilistic Language Model）采用的是文本分布式表示，即每个词表示为稠密的实数向量。NNLM模型的目标是构建语言模型：

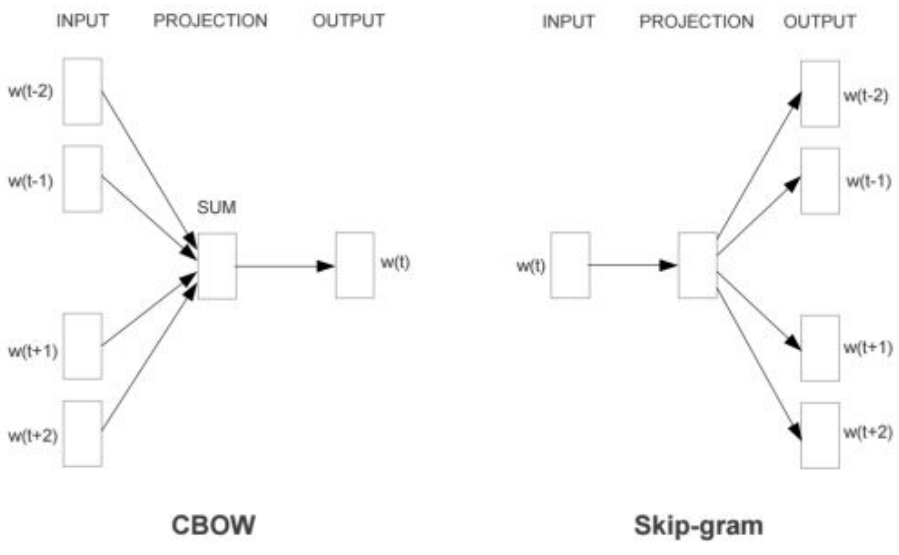
$$\text{The objective is to learn a good model } f(w_t,\cdots,w_{t-n+1}) = \hat{P}(w_t|w_1^{t-1}),$$

词的分布式表示即词向量（word embedding）是训练语言模型的一个附加产物，即图中的Matrix C。

尽管Hinton 86年就提出了词的分布式表示，Bengio 03年便提出了NNLM，词向量真正火起来是google Mikolov 13年发表的两篇word2vec的文章 Efficient Estimation of Word Representations in Vector Space 和 Distributed Representations of Words and Phrases and their Compositionality

更重要的是发布了简单好用的 **word2vec**工具包

在语义维度上得到了很好的验证，极大的推进了文本分析的进程。下图是文中提出的CBOW 和 Skip-Gram两个模型的结构，基本类似于NNLM，不同的是模型去掉了非线性隐层，预测目标不同，CBOW是上下文词预测当前词，Skip-Gram则相反。



除此之外，提出了Hierarchical Softmax 和 Negative Sample两个方法，很好的解决了计算有效性，事实上这两个方法都没有严格的理论证明，有些trick之处，非常的实用主义。实际上word2vec学习的向量和真正语义还有差距，更多学到的是具备相似上下文的词，比如“good”“bad”相似度也很高，反而是文本分类任务输入有监督的语义能够学到更好的语义表示，有机会后续系统分享下。

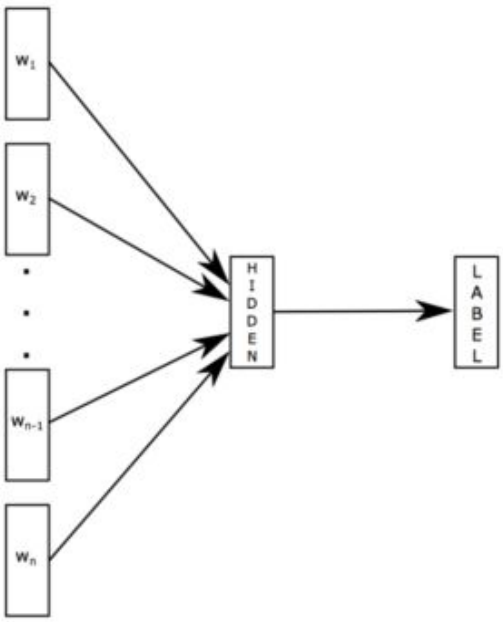
至此，文本的表示通过词向量的表示方式，把文本数据从高纬度高稀疏的神经网络难处理的方式，变成了类似图像、语音的的连续稠密数据。深度学习算法本身有很强的数据迁移性，很多之前在图像领域很适用的深度学习算法比如CNN等也可以很好的迁移到文本领域了。

2.2 深度学习文本分类模型

词向量解决了文本表示的问题，文本分类模型则是利用CNN/RNN等深度学习网络及其变体解决自动特征提取（即特征表达）的问题。

1）fastText

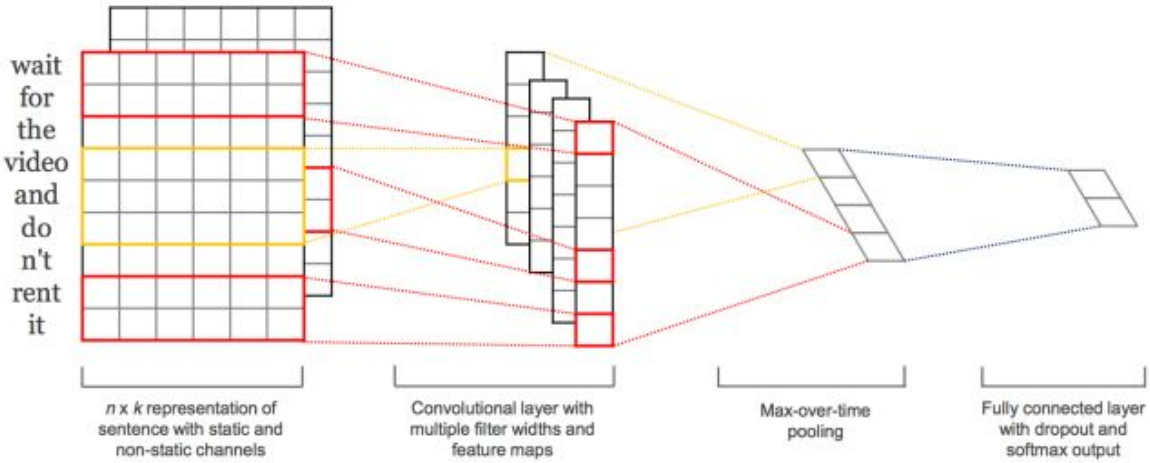
fastText 是上文提到的 word2vec 作者 Mikolov 转战 Facebook 后16年7月刚发表的一篇论文 Bag of Tricks for Efficient Text Classification。 fastText 它极致简单，模型图见下：



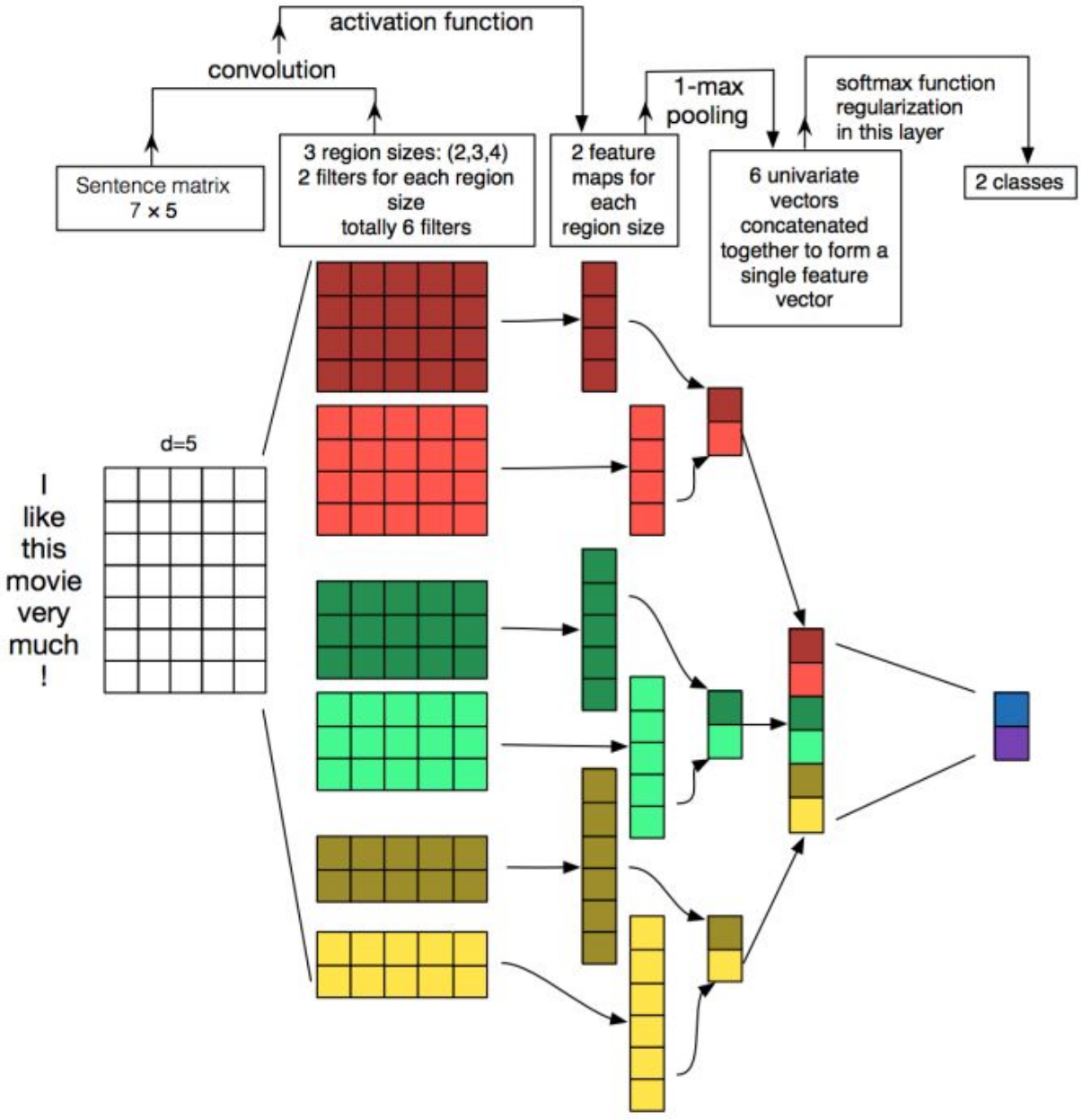
原理是把句子中所有的词向量进行平均（某种意义上可以理解为只有一个avg pooling特殊CNN），然后直接接 softmax 层。其实文章也加入了一些 n-gram 特征的 trick 来捕获局部序列信息。文章倒没太多信息量，算是“水文”吧，带来的思考是文本分类问题是有一些“线性”问题的部分，也就是说不必做过多的非线性转换、特征组合即可捕获很多分类信息，因此有些任务即便简单的模型便可以搞定了。

2）TextCNN

fastText 中的网络结果是完全没有考虑词序信息的，而它用的 n-gram 特征 trick 恰恰说明了局部序列信息的重要意义。卷积神经网络（CNN Convolutional Neural Network）最初在图像领域取得了巨大成功，核心点在于可以**捕捉局部相关性**，具体到文本分类任务中可以利用CNN来提取句子中类似 n-gram 的关键信息。



TextCNN的详细过程原理图见下：



TextCNN详细过程：第一层是图中最左边的7乘5的句子矩阵，每行是词向量，维度=5，这个可以类比为图像中的原始像素点了。然后经过有 filter_size=(2,3,4) 的一维卷积层，每个filter_size 有两个输出 channel。第三层是一个1-max pooling层，这样不同长度句子经过pooling层之后都能变成定长的表示了，最后接一层全连接的 softmax 层，输出每个类别的概率。

特征：这里的特征就是词向量，有静态（static）和非静态（non-static）方式。static方式采用比如word2vec预训练的词向量，训练过程不更新词向量，实质上属于迁移学习了，特别是数据量比较小的情况下，采用静态的词向量往往效果不错。non-static则是在训练过程中更新词向量。推荐的方式是 non-static 中的 fine-tunning方式，它是以预训练（pre-train）的word2vec向量初始化词向量，训练过程中调整词向量，能加速收敛，当然如果有充足的训练数据和资源，直接随机初始化词向量效果也是可以的。

通道（Channels）：图像中可以利用 (R, G, B) 作为不同channel，而文本的输入的channel通常是不同方式的embedding方式（比如 word2vec或Glove），实践中也有利用静态词向量和fine-tunning词向量作为不同channel的做法。

一维卷积（conv-1d）：图像是二维数据，经过词向量表达的文本为一维数据，因此在TextCNN卷积用的是一维卷积。一维卷积带来的问题是需要设计通过不同 filter_size 的 filter 获取不同宽度的视野。

Pooling层：将 pooling 改成 (dynamic) k-max pooling ，pooling阶段保留 k 个最大的信息，保留了全局的序列信息。比如在情感分析场景，举个例子：

> “ 我觉得这个地方景色还不错，但是人也实在太多了 ”

虽然前半部分体现情感是正向的，全局文本表达的是偏负面的情感，利用 k-max pooling能够很好捕捉这类信息。

3）TextRNN

尽管TextCNN能够在很多任务里面能有不错的表现，但CNN有个最大问题是固定 filter_size 的视野，一方面无法建模更长的序列信息，另一方面 filter_size 的超参调节也很繁琐。CNN本质是做文本的特征表达工作，而自然语言处理中更常用的是递归神经网络（RNN, Recurrent Neural Network），能够更好的表达上下文信息。具体在文本分类任务中，Bi-directional RNN（实际使用的是双向LSTM）从某种意义上可以理解为可以捕获变长且双向的的 "n-gram" 信息。

RNN算是在自然语言处理领域非常一个标配网络了，在序列标注/命名体识别/seq2seq模型等很多场景都有应用，[，下图LSTM用于网络结构原理示意图，示例中的是利用最后一个词的结果直接接全连接层softmax输出了。

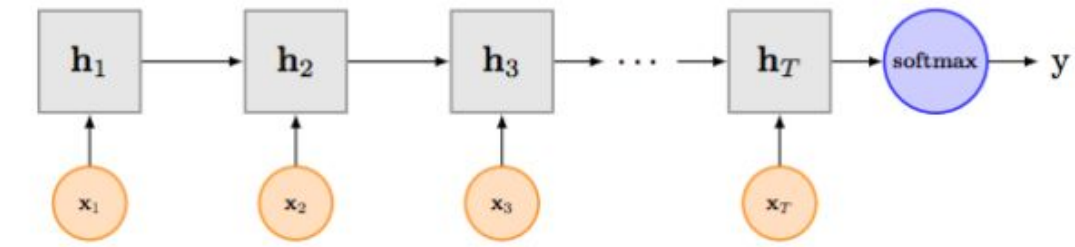


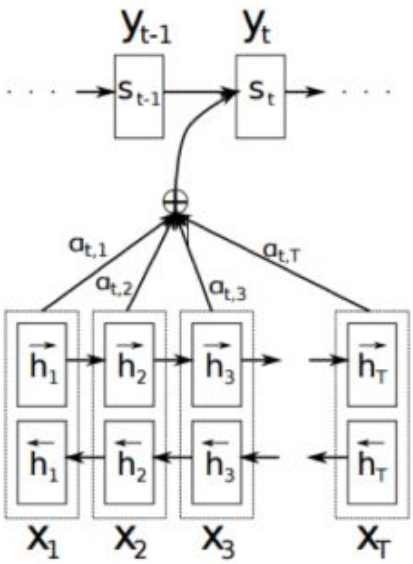
Figure 1: Recurrent Neural Network for Classification

4）TextRNN + Attention

CNN和RNN用在文本分类任务中尽管效果显著，但都有一个不足的地方就是不够直观，可解释性不好，特别是在分析badcase时候感受尤其深刻。而注意力（Attention）机制是自然语言处理领域一个常用的建模长时间记忆机制，能够很直观的给出每个词对结果的贡献，基本成了Seq2Seq模型的标配了。实际上文本分类从某种意义上也可以理解为一种特殊的Seq2Seq，所以考虑把Attention机制引入近来。

Attention机制介绍：

以机器翻译为例简单介绍下，下图中 x_t 是源语言的一个词，y_t 是目标语言的一个词，机器翻译的任务就是给定源序列得到目标序列。翻译 y_t 的过程产生取决于上一个词 y_{t-1} 和源语言的词的表示 h_j（x_j 的 bi-RNN 模型的表示），而每个词所占的权重是不一样的。比如源语言是中文 “我 / 是 / 中国人” 目标语言 “i / am / Chinese”，翻译出“Chinese”时候显然取决于“中国人”，而与“我 / 是”基本无关。下图公式, $/alpha_{ij}$ 则是翻译英文第 i 个词时，中文第 j 个词的贡献，也就是注意力。显然在翻译“Chinese”时，“中国人”的注意力值非常大。

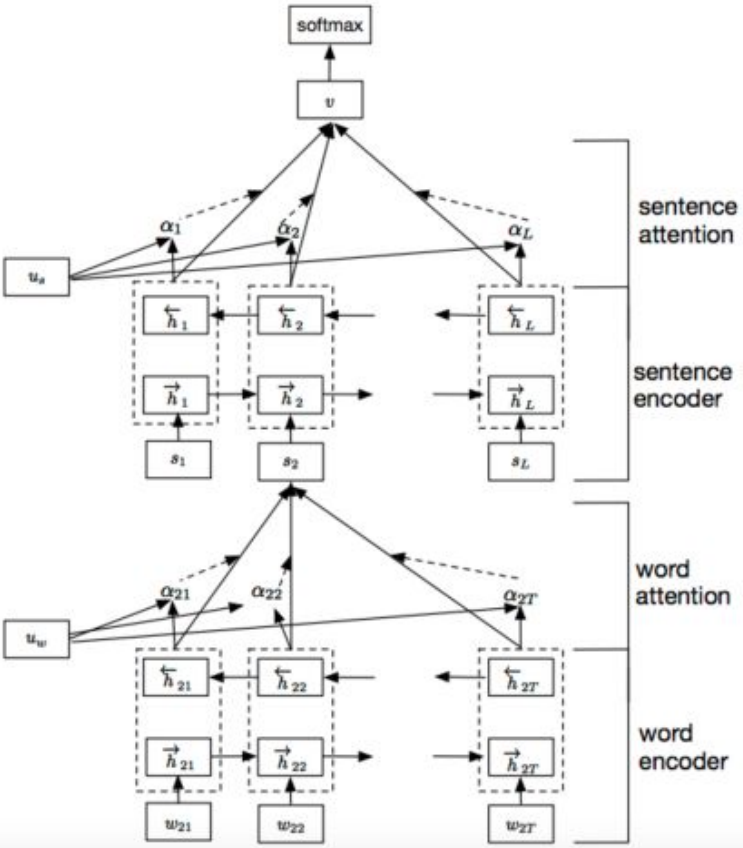


$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \qquad (5)$$

Attention的核心point是在翻译每个目标词（或 预测商品标题文本所属类别）所用的上下文是不同的，这样的考虑显然是更合理的。

TextRNN + Attention 模型：

下图是模型的网络结构图，它一方面用层次化的结构保留了文档的结构，另一方面在word-level和sentence-level。



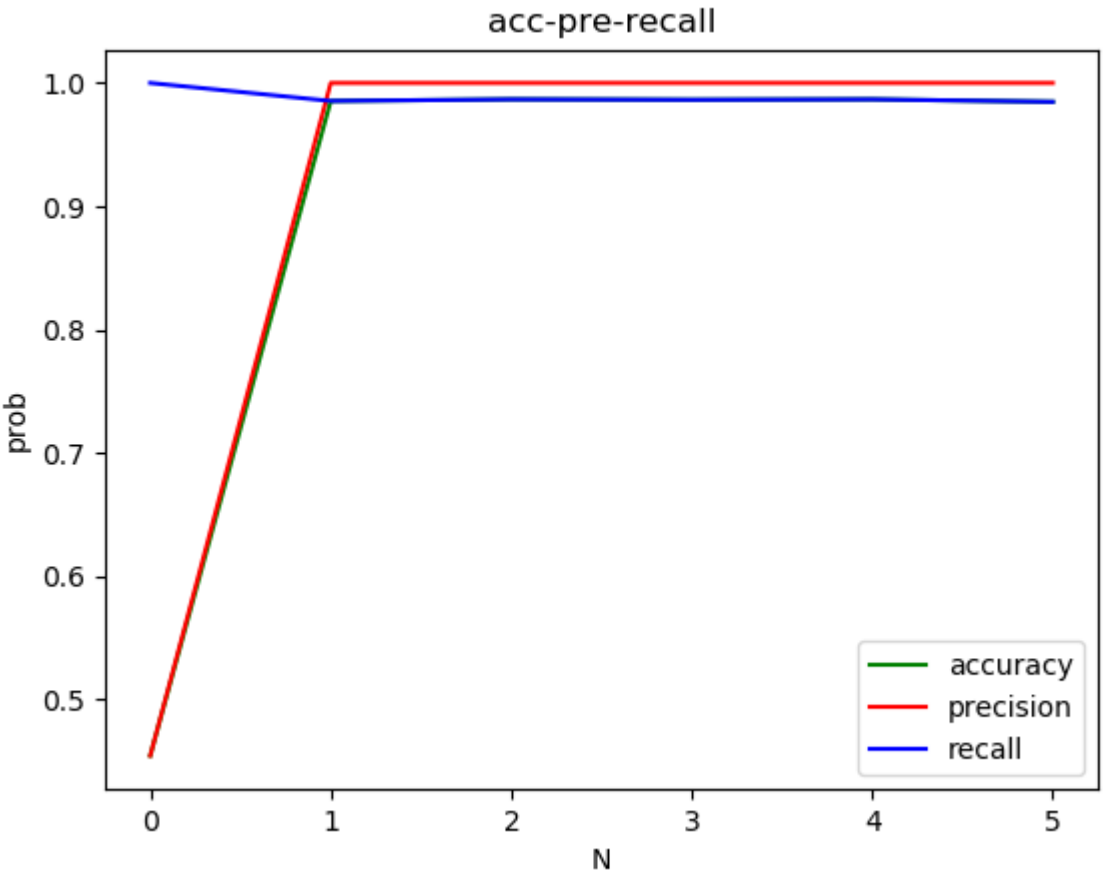
加入Attention之后最大的好处自然是能够直观的解释各个句子和词对分类类别的重要性。

三.实验报告说明

> 在本次实验中，我们爬取了旅游评论数据集，总共有35706条数据，正样本为好评，负样本为差评，其中正样本有32460条，负样本有3246条。对于文本表示，我们使用VSM和word2vec模型。一共做了四种测试：
>
> 1. VSM + RandomForestClassifier
> 2. VSM + BernoulliNB
> 3. word2vec + TextRNN
> 4. word2vec + TextRNN + Attention

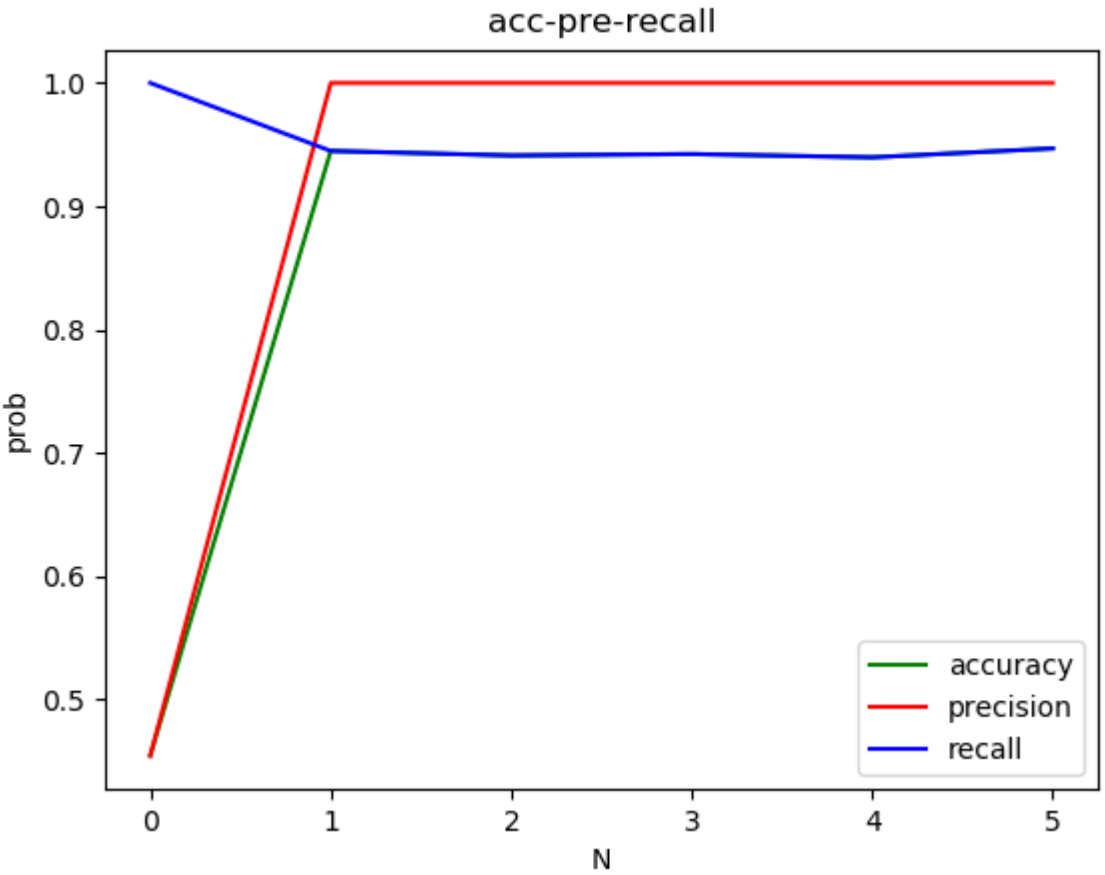
1.VSM + RandomForestClassifier

实验采用6-fold验证，accuracy',precision,recall如下所示：



2.VSM + BernoulliNB

实验采用6-fold验证，accuracy',precision,recall如下所示：



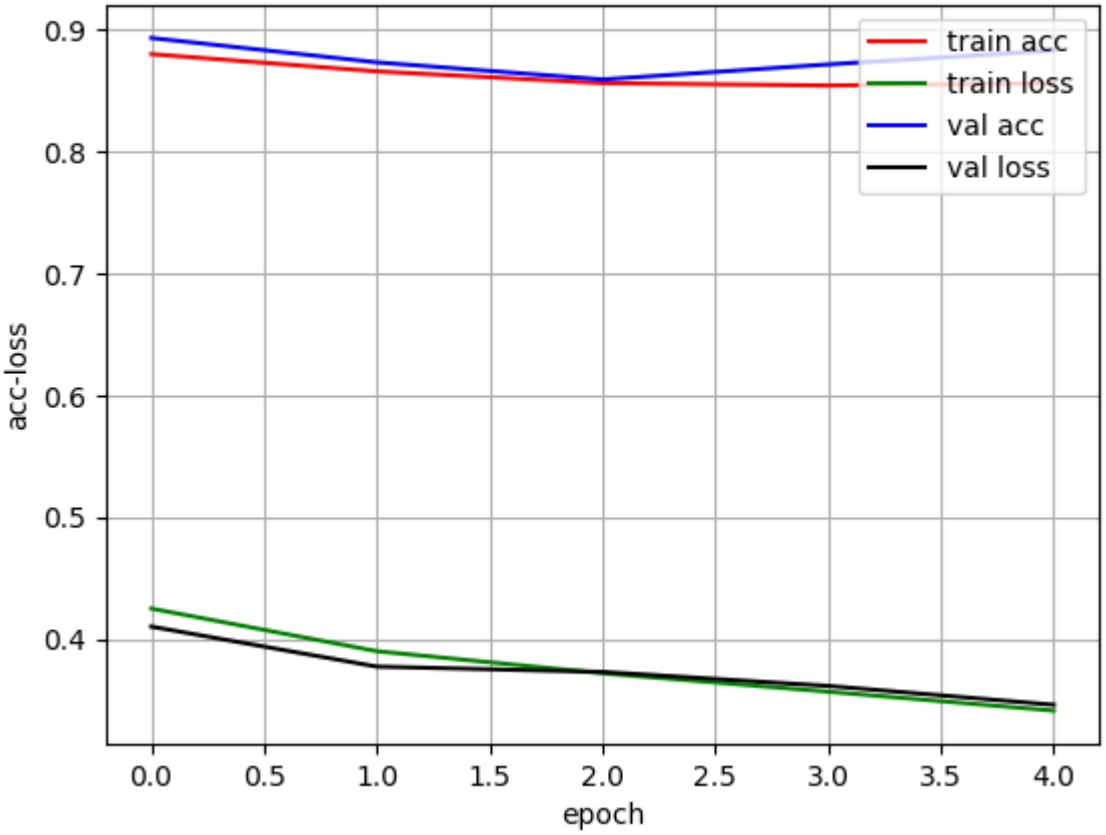
3.word2vec + TextRNN

双向GRU结构，模型结构参数如下：

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 50)	0	
embedding_1 (Embedding)	(None, 50, 300)	9000000	input_1[0][0]
spatial_dropout1d_1 (SpatialDrop	(None, 50, 300)	0	embedding_1[0][0]
bidirectional_1 (Bidirectional)	(None, 50, 512)	855552	spatial_dropout1d_1[0][0]
conv1d_1 (Conv1D)	(None, 49, 128)	131200	bidirectional_1[0][0]
global_average_pooling1d_1 (Glob	(None, 128)	0	conv1d_1[0][0]
global_max_pooling1d_1 (GlobalMa	(None, 128)	0	conv1d_1[0][0]
concatenate_1 (Concatenate)	(None, 256)	0	global_average_pooling1d_1[0][0] global_max_pooling1d_1[0][0]
dense_1 (Dense)	(None, 1)	257	concatenate_1[0][0]

Total params: 9,987,009
Trainable params: 9,987,009
Non-trainable params: 0

实验结果如下所示：



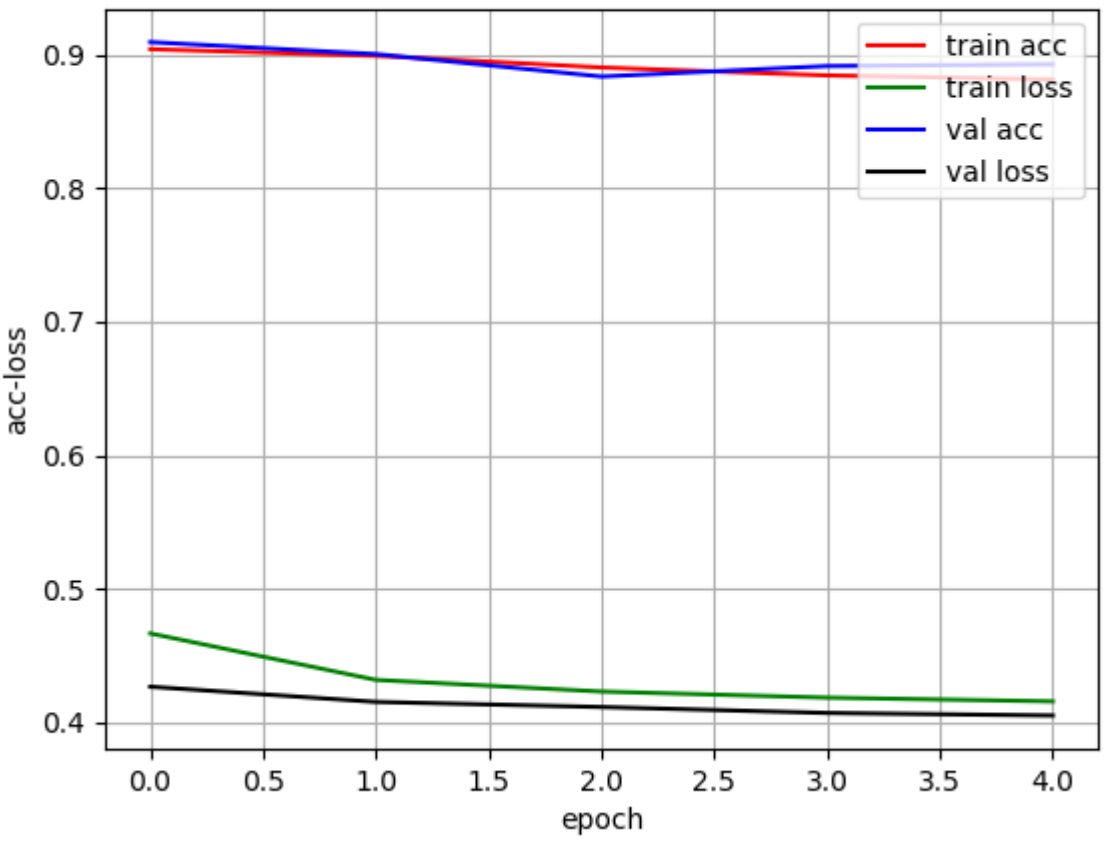
4.word2vec + TextRNN + Attention

LSTM加Attention模型，模型参数如下：

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 50)	0
embedding_1 (Embedding)	(None, 50, 300)	9000000
bidirectional_1 (Bidirection	(None, 50, 600)	1442400
dropout_1 (Dropout)	(None, 50, 600)	0
attention_1 (Attention)	(None, 600)	650
dense_1 (Dense)	(None, 256)	153856
dropout_2 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 1)	257

Total params: 10,597,163
Trainable params: 1,597,163
Non-trainable params: 9,000,000

实验结果如下所示：



从上面四个实验，我们可以看出，传统的文本分类模型，效果还是不错的，其中RandomForestClassifier的效果要比BernoulliNB效果更好一些。深度学习的方法，acc大约也在0.9左右，训练集和测试集的acc相差不大。

参考

深度学习（CNN RNN Attention）解决大规模文本分类问题