自然语言处理系列（4）：深度学习解决大规模文本分类问题

这篇文章总结了文本分类领域特别是应用深度学习解决文本分类的相关思路、做法和部分实践的经验。转载自知乎清凇撰写的文章用深度学习（CNN RNN Attention）解决大规模文本分类问题 - 综述和实践

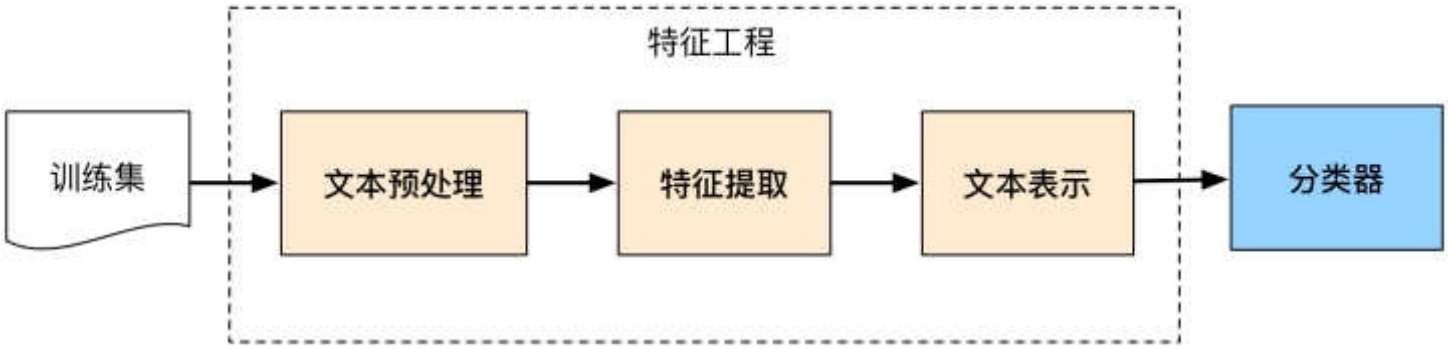
> 业务问题描述：

淘宝商品的一个经典的例子见下图，图中商品的标题是“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”。淘宝网后台是通过树形的多层的类目体系管理商品的，覆盖叶子类目数量达上万个，商品量也是10亿量级，我们的任务是根据商品标题预测其所在叶子类目，示例中商品归属的类目为“女装/女士精品>>蕾丝衫/雪纺衫”。很显然，这是一个非常典型的短文本多酚类问题。接下来分别会介绍下文本分类传统的和深度学习的做法，最后简单梳理下实践的经验。

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里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监理专家系统，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，覆盖的范围和准确性都非常有限。

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+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，玩机器学习的同学对此自然再熟悉不过了。

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，但却及其的重要。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，二分类器是“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处理领域也有其特有的特征处理逻辑，传统文本分类任务大部分工作也在此处。文本特征工程分为文本预处理、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。

1.1.1 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，基于字粒度显然随时了过多“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。而停用词是本文中一些高频的带刺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词表中出现的词，本质上属于特征选择的一部分。

经过文本分词和去停用词之后淘宝商品示例标题变成了下图“/”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
```

1.1.2 文本表示和特征提取

> 文本表示

文本表示的目的是把预处理后的文本转化成计算机可以理解的方式，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），最大的不足是忽略上下文关系，没歌词之间彼此独立，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)
```

一般来说词库至少都是百万级别，因此词袋模型有两个最大的问题：高维度、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纬度，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。

> 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/PLS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

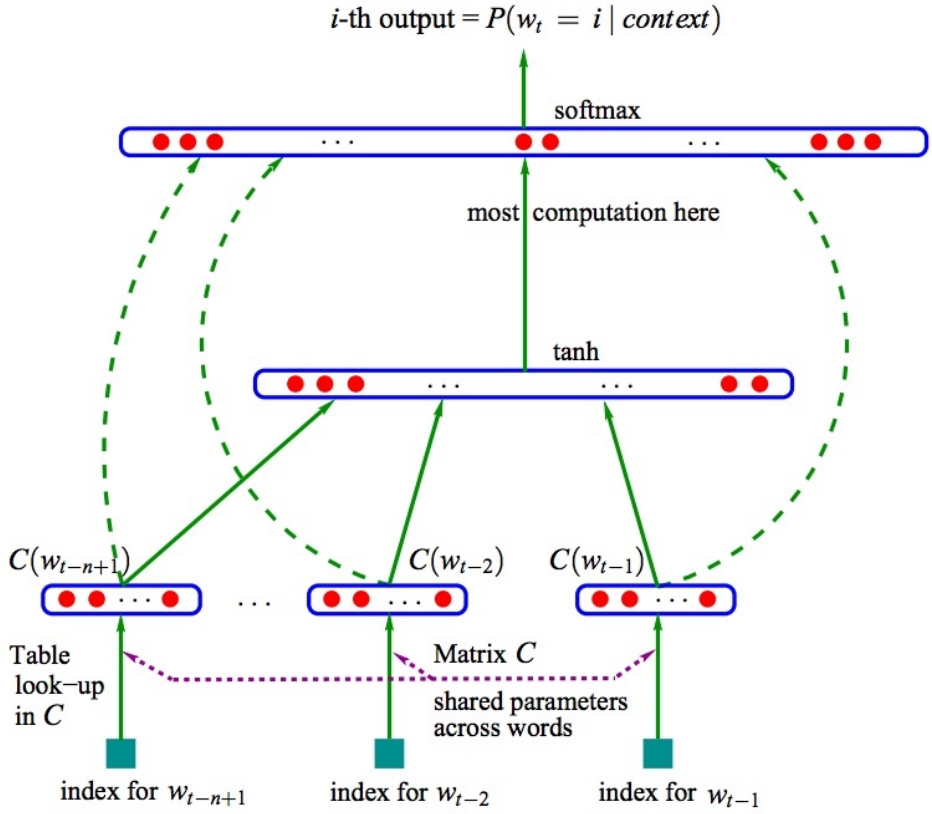
分类器都是统计分类方法了，基本上大部分机器学习方法都在本文分类领域有所应用，比如朴素贝叶斯分类算法、KNN、SVM、最大熵和神经网络等等，传统分类模型不是本文重点，在这里就不展开了。

二、深度学习文本分类方法

上文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维度高稀疏性的，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。而深度学习最初之所以在图像和语音取到巨大的成功，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，基本思想是将没歌词表达成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维向量每一维k个值，就可以表征k^n个概念了。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

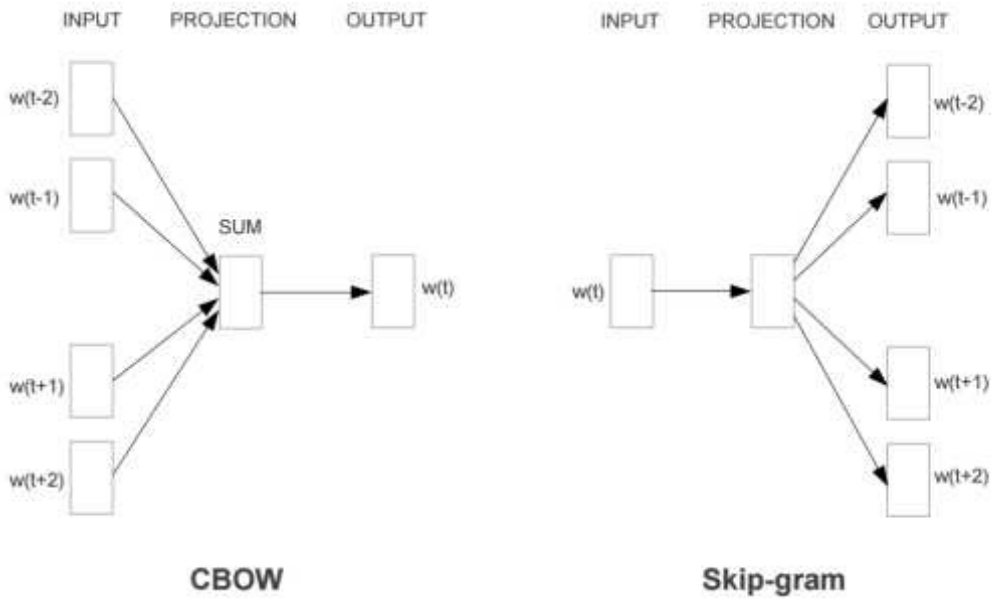


这篇文章提出的神经网络语言模型（NNLM，Neural Probabilistic Language Model），采用的是文本分布式表示，即每个词表示为稠密的实数向量。NNLM模型的目标是构建语言模型：

$$f(w_t, \cdots, w_{t-n+1}) = P(w_t|w_1^{t-1})$$

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NNLM，词向量真正火起来是google的Mikolov在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地验证，极大的推动了文本分析的进程。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，很好地解决了计算有效性，事实上这两个方法都没有严格的理论证明，有一些trick之处，非常的实用主义。详细的过程不再阐述了，有兴趣深入理解Word2Vec的，推荐读读这篇不错的paper：word2vec Parameter Learning Explained。额外多提一点，实际上WordVec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。

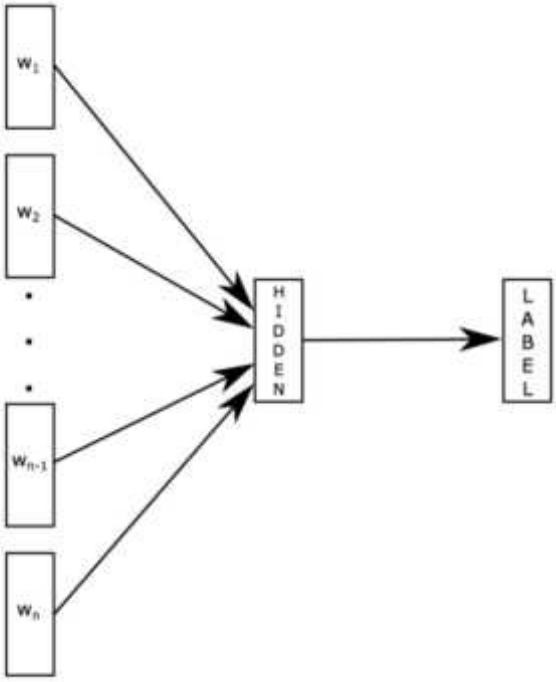
至此，文本的表示通过词向量的表示方法，把文本数据从高维度高稀疏性的神经网络难处理的方式，变成了类似图像、语言的连续稠密数据。深度学习算法本身有很强的数据迁移性，很多之前在图像领域很适用的深度学习算法比如CNN等也可以很好的迁移到文本领域了，下面一小节具体阐述下文本分类邻域深度学习的方法。

2.2 深度学习文本分类模型

2.2.1 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把fastText放在此处并非因为他是文本分类的主流做法，而

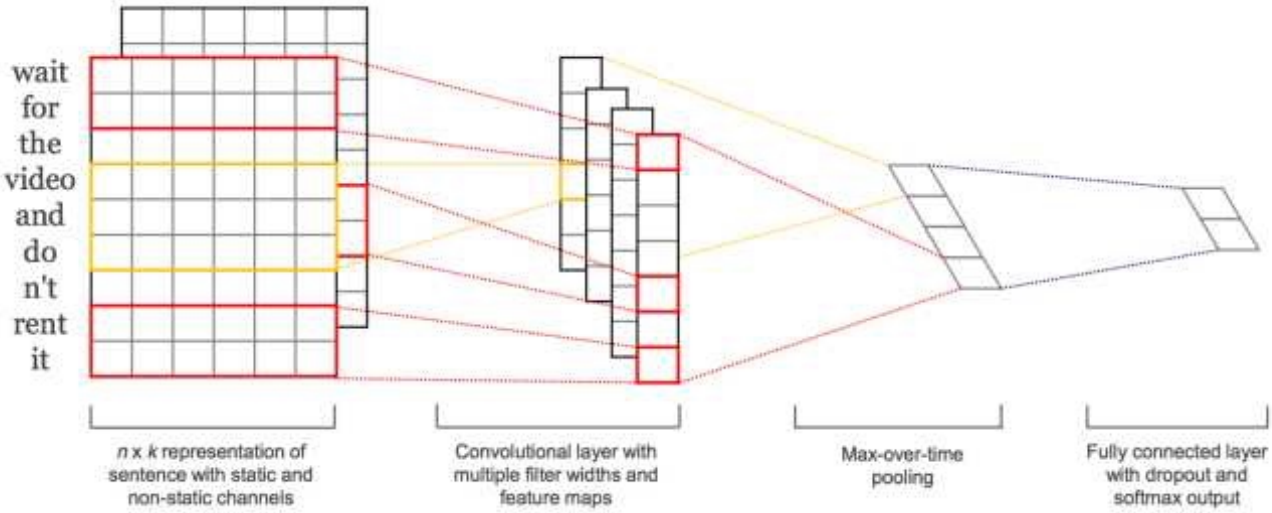
是他及其简单，模型图见下：



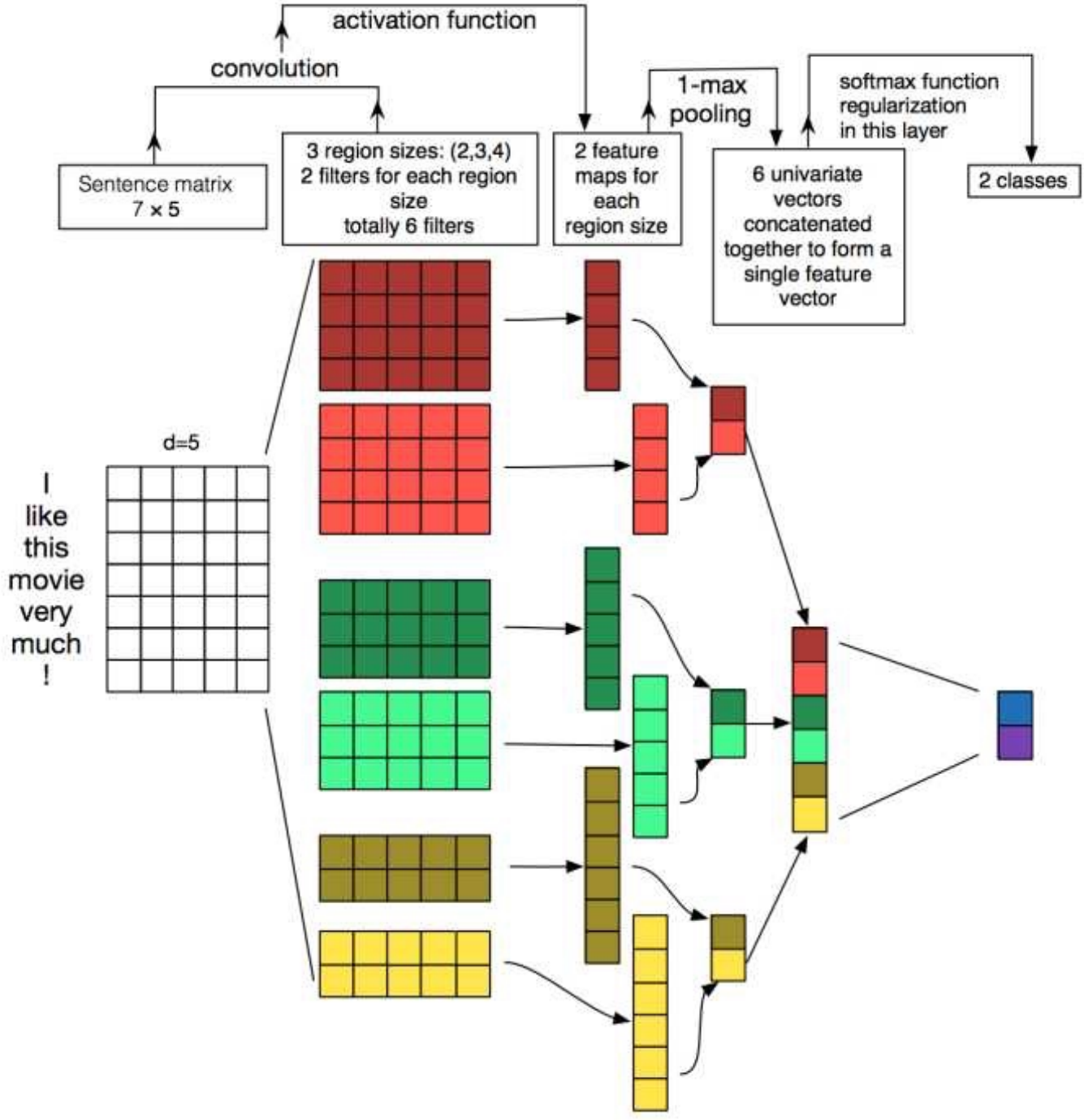
原理是把句子中所有的词向量进行平均（某种意义上可以理解为只有一个Avg Pooling的特征CNN），然后直接接SoftMax层。其实文章也加入了一些n-gram特征的tric来捕获局部序列信息。文章到没有太多的信息量，算是“水文”吧，带来的思考是文本分类问题是有一些“线性”问题的部分，也就是说不必做过多的非线性转换、特征组合即可捕获很多分类信息，因此有些任务即便简单的模型便可以搞定了。

2.2.2 TextCNN

本篇文章的题图选用的就是14年这篇文章提出的TextCNN的结果（见下图）。fastText中的网络结构是完全没有考虑词序信息的，而他用的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，CNN原理就不讲了，核心点在于可以捕捉局部相关性，具体到文本分类任务重可以利用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：利用CNN解决文本分类问题的文章还是很多的，比如这篇 A Convolutional Neural Network for Modelling Sentences 最有意思的输入是在 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。

2.2.3 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，Recurrent Neural Network for Text Classification with Multi-Task Learning文中介绍了RNN用于分类问题的设计，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。

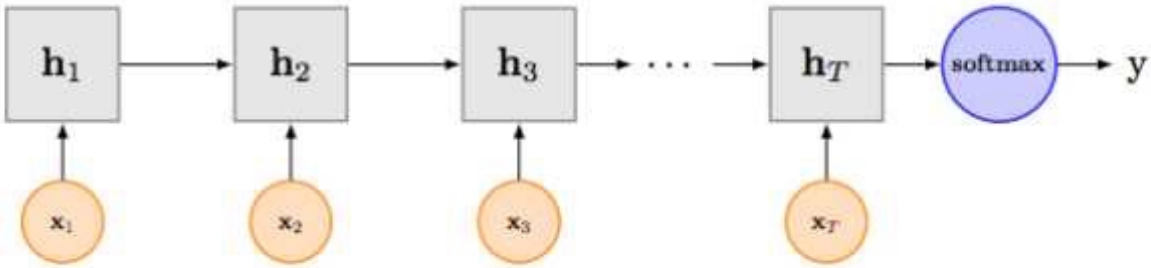


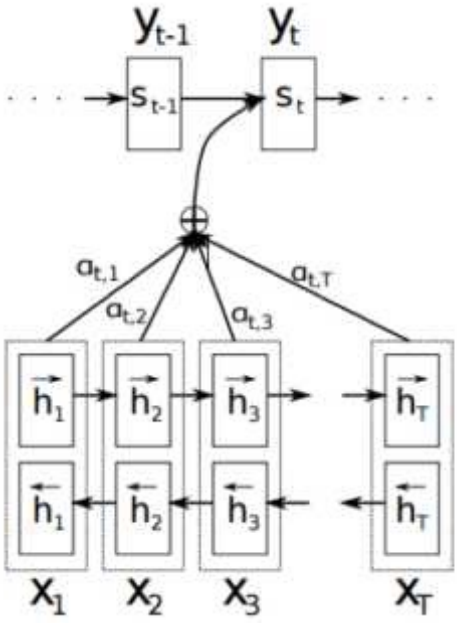
Figure 1: Recurrent Neural Network for Classification

2.2.4 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，研究了下学术界果然有类似做法。

Attention机制介绍：详细介绍Attention恐怕需要一小篇文章的篇幅，感兴趣的可参考14年这篇paper NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE。

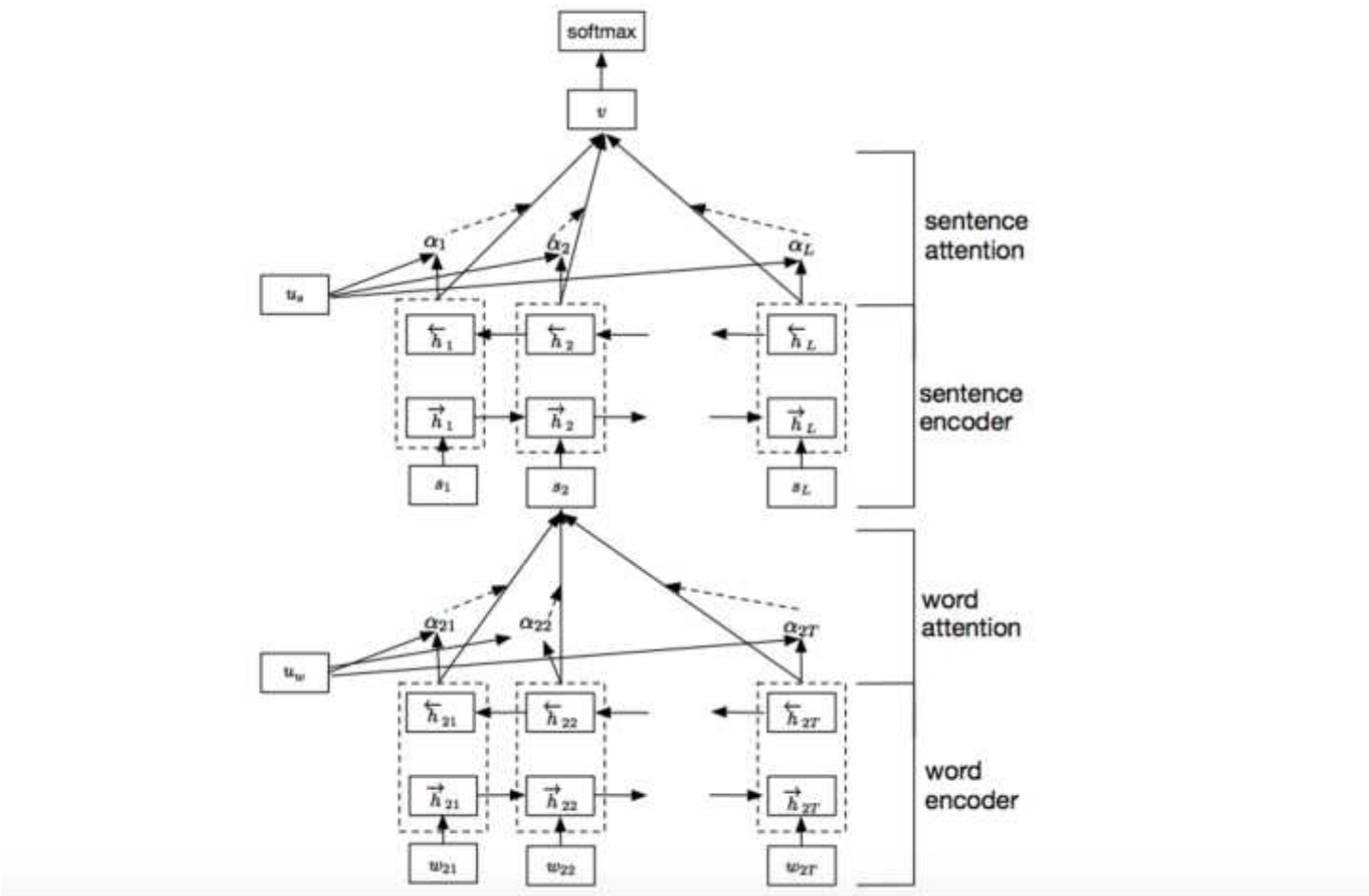
以机器翻译为例简单介绍下，下图中 x_t 是源语言的一个词，y_t 是目标语言的一个词，机器翻译的任务就是给定源序列得到目标序列。翻译 y_t 的过程产生取决于上一个词 y_{t-1} 和源语言的词的表示 h_j（x_j 的 bi-RNN 模型的表示），而每个词所占的权重是不一样的。比如源语言是中文“我 / 是 / 中国人” 目标语言 “i / am / Chinese”，翻译出“Chinese”时候显然取决于“中国人”，而与“我 / 是”基本无关。下图公式，α_{ij}则是翻译英文第 i 个词时，中文第 j 个词的贡献，也就是注意力。显然在翻译“Chinese”时，“中国人”的注意力值非常大。



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。

2.2.5 TextRNN + Attention

我们参考了这篇文章 Hierarchical Attention Networks for Document Classification，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。淘宝标题场景只需要 word-level 这一层的 Attention 即可。



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。

2.2.6 TextRCNN（TextRNN + CNN）

我们参考的是中科院15年发表在AAAI上的这篇文章 Recurrent Convolutional Neural Networks for Text Classification 的结构：

$$\boldsymbol{c}_l(w_i) = f(W^{(l)}\boldsymbol{c}_l(w_{i-1}) + W^{(sl)}\boldsymbol{e}(w_{i-1})) \quad (1)$$

$$\boldsymbol{c}_r(w_i) = f(W^{(r)}\boldsymbol{c}_r(w_{i+1}) + W^{(sr)}\boldsymbol{e}(w_{i+1})) \quad (2)$$

利用前向和后向RNN得到每个词的前向和后向上下文的表示：

$$c_l(w_i) = f(W^{(l)}c_l(w_{i-1}) + W^{(sl)}e(w_{i-1})) \quad (1)$$
$$c_r(w_i) = f(W^{(r)}c_r(w_{i+1}) + W^{(sr)}e(w_{i+1})) \quad (2)$$

这样词的表示就变成词向量和前向后向上下文向量concat起来的形式了，即：

$$x_i = [c_l(w_i); e(w_i); c_r(w_i)] \quad (3)$$

最后再接跟TextCNN相同卷积层，pooling层即可，唯一不同的是卷积层 filter_size = 1就可以了，不再需要更大 filter_size 获得更大视野，这里词的表示也可以只用双向RNN输出。

三、一点经验

理论和实践之间的Gap往往差异巨大，学术paper更关注的是模型架构设计的新颖性等，更重要的是新的思路；而实践最重要的是在落地场景的效果，关注的点和方法都不一样。这部分简单梳理实际做项目过程中的一点经验教训。

模型显然并不是最重要的：不能否认，好的模型设计对拿到好结果的至关重要，也更是学术关注热点。但实际使用中，模型的工作量占的时间其实相对比较少。虽然再第二部分介绍了5种CNN/RNN及其变体的模型，实际中文本分类任务单纯用CNN已经足以取得很不错的结果了，我们的实验测试RCNN对准确率提升大约1%，并不是十分的显著。最佳实践是先用TextCNN模型把整体任务效果调试到最好，再尝试改进模型。

理解你的数据：虽然应用深度学习有一个很大的优势是不再需要繁琐低效的人工特征工程，然而如果你只是把他当做一个黑盒，难免会经常怀疑人生。一定要理解你的数据，记住无论传统方法还是深度学习方法，数据 sense 始终非常重要。要重视 badcase 分析，明白你的数据是否适合，为什么对为什么错。

关注迭代质量 - 记录和分析你的每次实验：迭代速度是决定算法项目成败的关键，学过概率的同学都很容易认同。而算法项目重要的不只是迭代速度，一定要关注迭代质量。如果你没有搭建一个快速实验分析的套路，迭代速度再快也只会替你公司心疼宝贵的计算资源。建议记录每次实验，实验分析至少回答这三个问题：为什么要实验？结论是什么？下一步怎么实验？

超参调节：超参调节是各位调参工程师的日常了，推荐一篇文本分类实践的论文 A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification，里面贴了一些超参的对比实验，如果你刚开始启动文本分析任务，不妨按文章的结果设置超参，怎么最快的得到超参调节其实是一个非常重要的问题，可以读读 萧瑟的这篇文章 深度学习网络调参技巧 - 知乎专栏。

一定要用 dropout：有两种情况可以不用：数据量特别小，或者你用了更好的正则方法，比如bn。实际中我们尝试了不同参数的dropout，最好的还是0.5，所以如果你的计算资源很有限，默

认0.5是一个很好的选择。

fine-tuning 是必选的：上文聊到了，如果只是使用word2vec训练的词向量作为特征表示，我赌你一定会损失很大的效果。

未必一定要 softmax loss： 这取决与你的数据，如果你的任务是多个类别间非互斥，可以试试着训练多个二分类器，也就是把问题定义为multi lable 而非 multi class，我们调整后准确率还是增加了>1%。

类目不均衡问题：基本是一个在很多场景都验证过的结论：如果你的loss被一部分类别 dominate，对总体而言大多是负向的。建议可以尝试类似 booststrap 方法调整 loss 中样本权重方式解决。

避免训练震荡：默认一定要增加随机采样因素尽可能使得数据分布iid，默认shuffle机制能使得训练结果更稳定。如果训练模型仍然很震荡，可以考虑调整学习率或 mini_batch_size。

没有收敛前不要过早的下结论：玩到最后的才是玩的最好的，特别是一些新的角度的测试，不要轻易否定，至少要等到收敛吧。

四、写在最后

几年前校招面阿里时，一面二面聊的都是一个文本分类的项目（一个新浪微博主题分类的学校课题项目），用的还是文中介绍的传统的做法。面试时对特征项处理和各个分类器可谓如数家珍，被要求在白板上写了好几个特征选择公式，短短几年传统做法已经被远远超越，不得不感慨深度学习的发展。

值得感慨的一方面是今天技术的发展非常快，故步自封自然是万万万万不可取，深知还有很多理论尚且不懂还要继续深读paper；另一方面，理解理论原理和做好项目间实际非常有巨大的gap，特别是身处工业界的同仁们，学术圈值得钻但要把握分寸，如果仅仅追逐技术深度，不免容易陷入空中阁楼。