**文本特征提取**

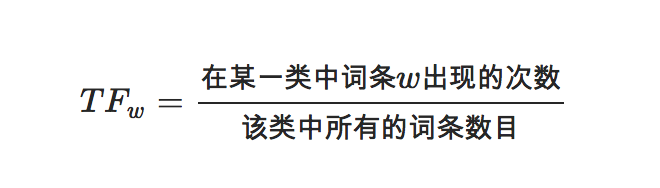
**三种特征提取方式：**

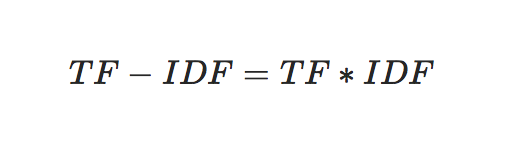
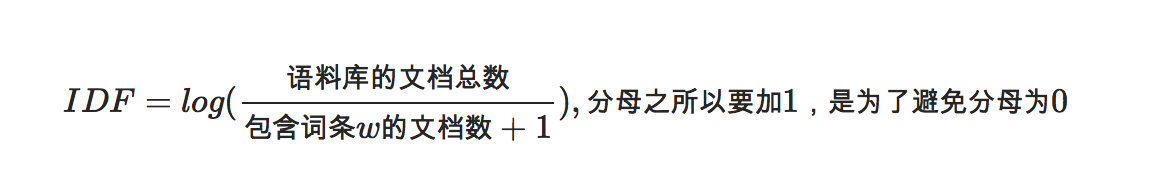
**TF-IDF**

**Word embedding**

**Word2vec**

**TF-IDF，词频-逆文件频率**



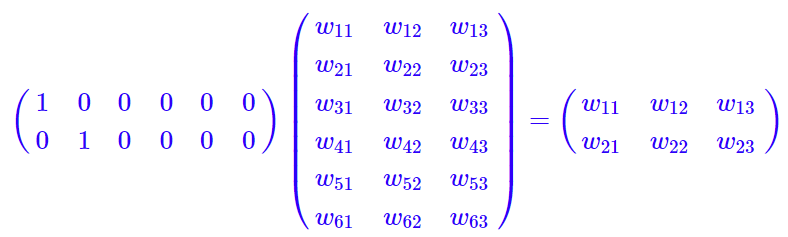


TF-IDF与一个词在文档中的出现次数成正比，与该词在整个语言中的出现次数成反比。所以，自动提取关键词的算法就很清楚了，就是计算出文档的每个词的TF-IDF值，然后按降序排列，取排在最前面的几个词。

文本特征提取函数CountVectorizer()和TfidfVectorizer()

<https://blog.csdn.net/lyn5284767/article/details/85316931>

**Word embedding**



Embedding层就是以one hot为输入、中间层节点为字向量维数的全连接层！而这个全连接层的参数，就是一个“字向量表”！字向量就是one hot的全连接层的参数！

**Word2vec**

“lecture5-word2vec原理.docx”

**深度学习文本分类方法**

卷积神经网络(TextCNN)

循环神经网络(TextRNN)

TextRNN+Attention

TextRCNN(TextRNN+CNN)

文本分类方法综述

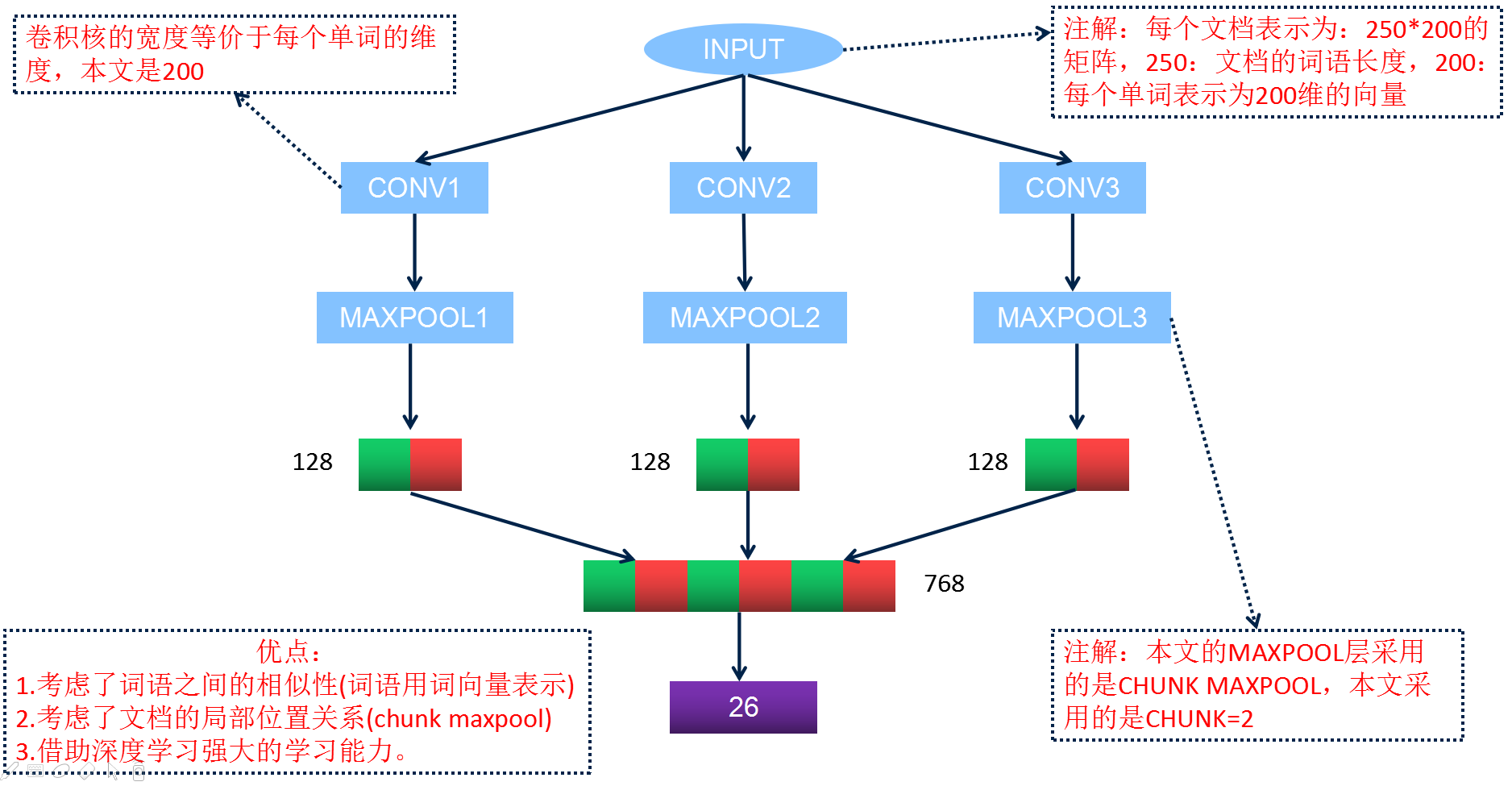
<http://blog.nsfocus.net/text-categorization-practice-based-keras/>

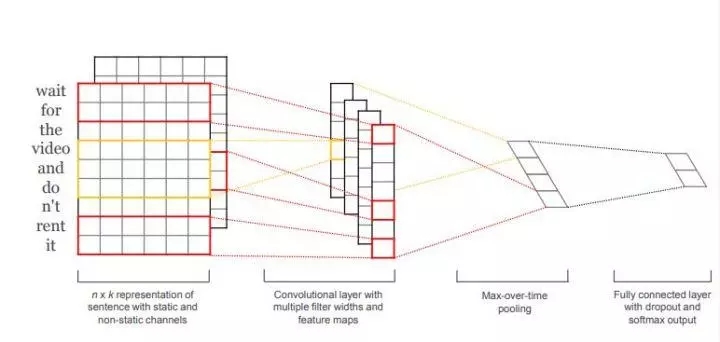
**传统文本分类方法**

基于逻辑回归模型

<https://github.com/Edward1Chou/Textclassification>

**针对文本分类的CNN结构**





“Convolutional Neural Networks for Sentence Classification”

模型的主要结构：

输入层+第一层卷积层+池化层+全连接+softmax层

# 创建tensor

print("正在创建模型...")

inputs=Input(shape=(sequence\_length,),dtype='int32')

embedding=Embedding(input\_dim=vocabulary\_size,output\_dim=embedding\_dim,input\_length=sequence\_length)(inputs)

reshape=Reshape((sequence\_length,embedding\_dim,1))(embedding)

# cnn

conv\_0=Conv2D(num\_filters,kernel\_size=(filter\_sizes[0],embedding\_dim),padding='valid',kernel\_initializer='normal',activation='relu')(reshape)

conv\_1=Conv2D(num\_filters,kernel\_size=(filter\_sizes[1],embedding\_dim),padding='valid',kernel\_initializer='normal',activation='relu')(reshape)

conv\_2=Conv2D(num\_filters,kernel\_size=(filter\_sizes[2],embedding\_dim),padding='valid',kernel\_initializer='normal',activation='relu')(reshape)

maxpool\_0=MaxPool2D(pool\_size=(sequence\_length-filter\_sizes[0]+1,1),strides=(1,1),padding='valid')(conv\_0)

maxpool\_1=MaxPool2D(pool\_size=(sequence\_length-filter\_sizes[1]+1,1),strides=(1,1),padding='valid')(conv\_1)

maxpool\_2=MaxPool2D(pool\_size=(sequence\_length-filter\_sizes[2]+1,1),strides=(1,1),padding='valid')(conv\_2)

concatenated\_tensor = Concatenate(axis=1)([maxpool\_0, maxpool\_1, maxpool\_2])

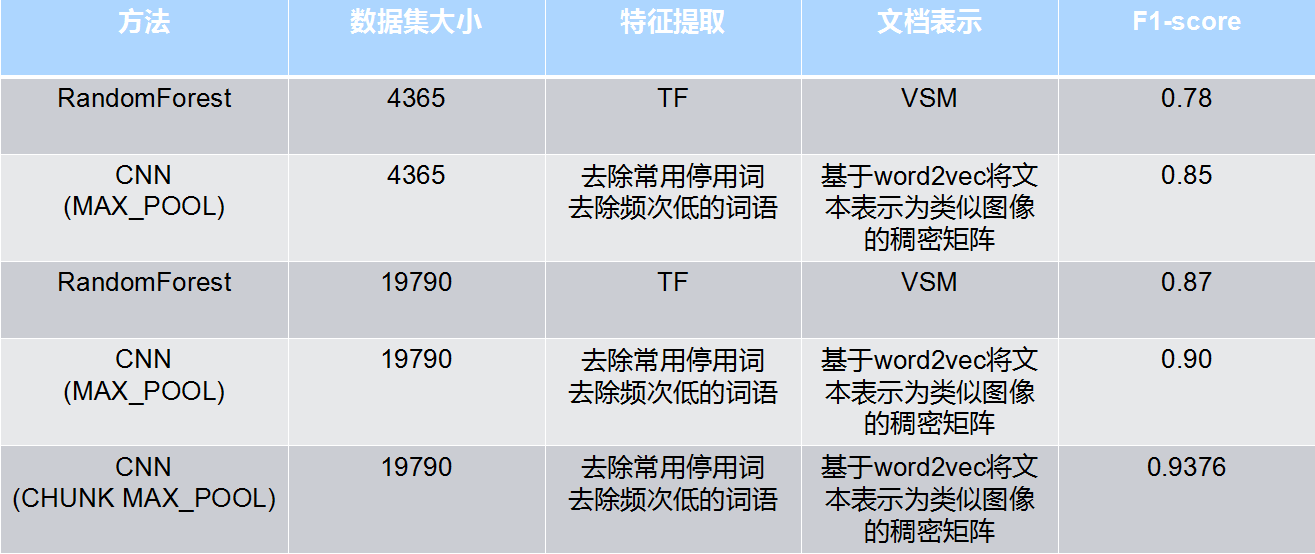
flatten = Flatten()(concatenated\_tensor)

dropout = Dropout(drop)(flatten)

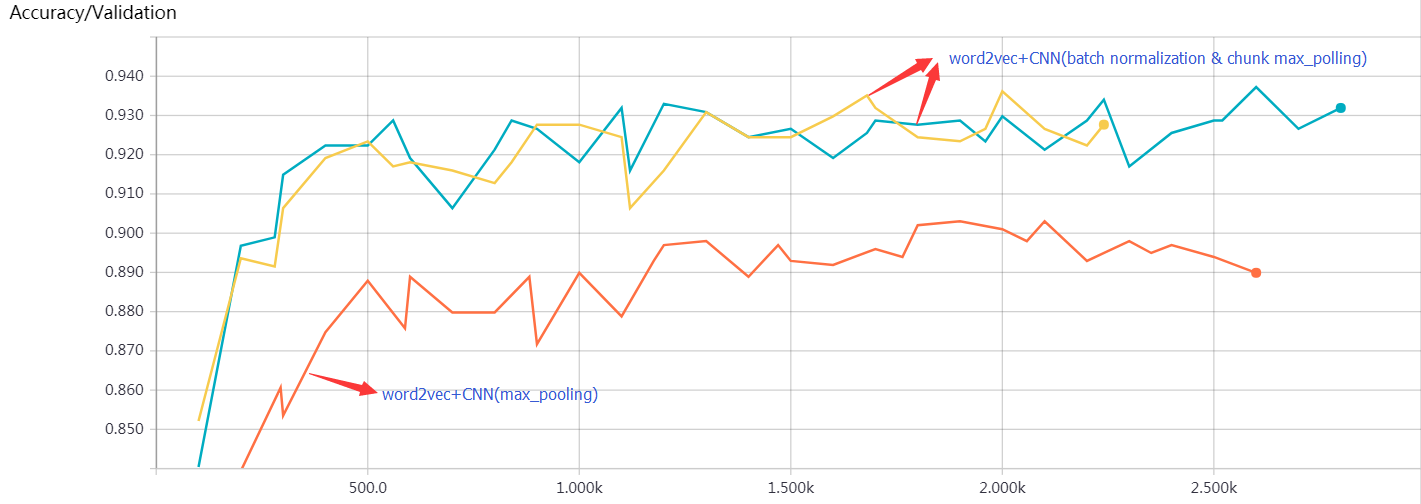
output = Dense(units=2, activation='softmax')(dropout)

model=Model(inputs=inputs,outputs=output)

**实验结果**

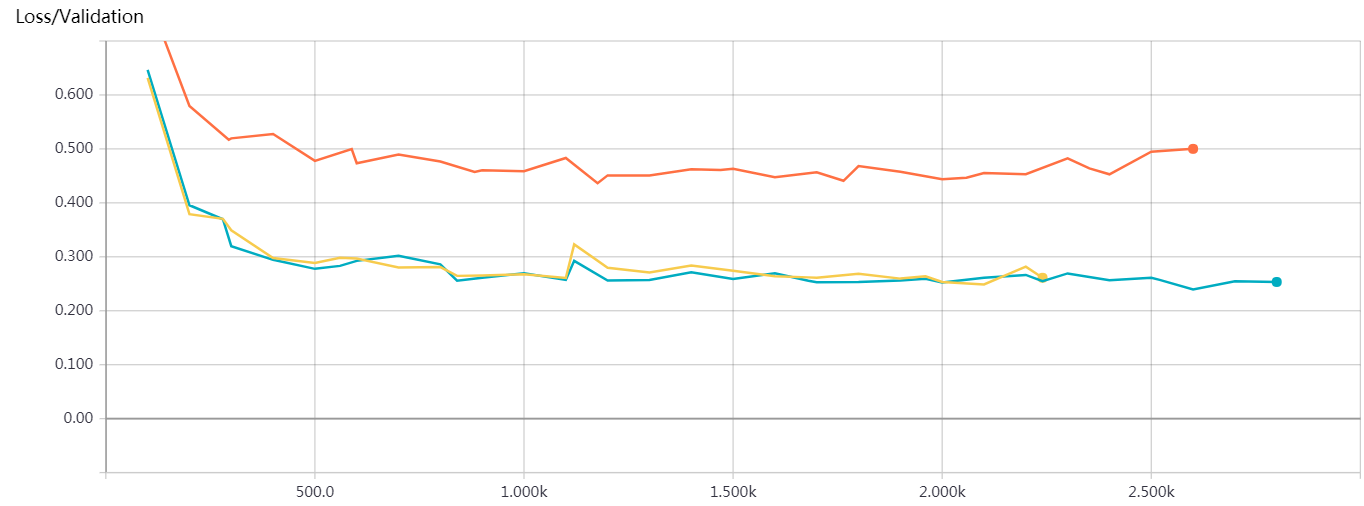


为了检验模型在真实数据上的分类准确率，我们又额外人工审核了1000条深圳地区的案情数据，相较于原来分类准确率的68%，提升到了现在的90%，说明我们的模型确实有效，相对于原来的模型有较大的提升。



红色：word2vec+CNN(max\_pooling)在验证集上的准确率走势图

黄色和蓝色：word2vec+CNN(batch normalization & chunk max\_pooling:2 chunk)在验证集上的准确率走势图



实例

Logistic regression 与 CNN text

<https://github.com/Edward1Chou/Textclassification>

CNN文本分类流程

<https://blog.csdn.net/vivian_ll/article/details/80829474>