# 聊天监控项目

[聊天监控项目 1](#_Toc8064639)

[一：业务理解 2](#_Toc8064640)

[二：数据预处理和打标签 2](#_Toc8064641)

[1,数据预处理 2](#_Toc8064642)

[(1)如何做分词 2](#_Toc8064643)

[(2)jieba所用到的主要算法. 3](#_Toc8064644)

[(3)正则匹配 4](#_Toc8064645)

[(4)为什么需要分词 4](#_Toc8064646)

[2，对数据做词向量处理 4](#_Toc8064647)

[(1)Word2Vec简介 4](#_Toc8064648)

[(2)word2vec模型 5](#_Toc8064649)

[3，one\_path聚类算法 6](#_Toc8064650)

[4，打标签 9](#_Toc8064651)

[(1)打标签规则： 9](#_Toc8064652)

[(2),如何判定文本是不是拉人敏感信息 9](#_Toc8064653)

[5，标签label 10](#_Toc8064654)

[三：Textcnn模型 11](#_Toc8064655)

[1，CNN模型 12](#_Toc8064656)

[(1).卷积神经网络的层级结构 12](#_Toc8064657)

[(2).数据输入层 12](#_Toc8064658)

[(3).卷积计算层 13](#_Toc8064659)

[(4).激励层 16](#_Toc8064660)

[(5).池化层 17](#_Toc8064661)

[(6).全连接层 18](#_Toc8064662)

[(7)总结 20](#_Toc8064663)

[2，CNN文本分类模型 20](#_Toc8064664)

[3，文本处理 22](#_Toc8064665)

[四：Textcnn模型更新与优化 23](#_Toc8064666)

[1， 模型本身优化 23](#_Toc8064667)

[2， 模型以外的因素 23](#_Toc8064668)

[五：接口和输出 24](#_Toc8064669)

[1， 输入数据 24](#_Toc8064670)

[2， 输出数据 24](#_Toc8064671)

# 一：业务理解

聊天监控针对游戏中拉人信息做出判断.现在使用神经网络模型Textcnn来做聊天监控.每个时刻都会产生很大的玩家聊天信息，借助大数据的一些方法把数据导入Textcnn模型当中，这个模型相当于一个过滤器它可以判断输入的信息是否是敏感信息.如下图所示：

聊天信息

输出判断信息

… …

聊天信息

# 二：数据预处理和打标签

## 1,数据预处理

首先对数据做分词和正则匹配，目的是给聊天信息去杂质.由于聊天信息一般比较短小，并没进行句子长短的筛选一致认为每个句子都包含有价值的信息.

### (1)如何做分词

分词定义：

指将一个汉字序列切分成一个一个单独的词，分词就是将连续的字序列按照一定的规范重新组合成词序列。

例如{我爱中国}使用jieba分词之后为{我/爱/中国}

Jieba分词简介：

他支持三种模式.

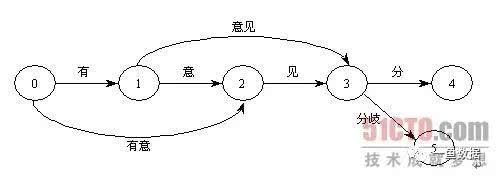
A,精确模式：试图将句子最精确的切分，适合文本分析.

B,全模式：把句子中所有的可以乘此的词语都扫描出来，速度非常快，但不能解决歧义.

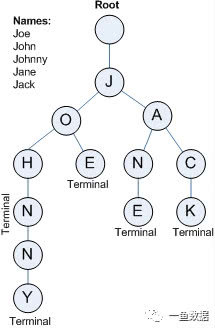
C,搜索引擎模式：在精确模式的基础上，对长词再次切分，提高召回率，合适用于搜素引擎分词.

### (2)jieba所用到的主要算法.

A,基于前缀词典实现高效的词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图(DAG)



B,使用数据结构里的前缀树或字典树.



如上图所示，这就是最简单的tree的原理在分词上的运用：我们有Johnny、Joe、Jane和Jack四个名字，假设我们要让计算机查找名字Jack是否存在，普通的方法来说，计算机可能会扫描所有的字符串，非常耗时低效，但tree的从上至下搜索、每一次只判定一个字母、如果某个特定的节点(node)的下一个节点(child node)不再符合搜索要求，那么搜索就会停止，这样效率就会大大的提高.

C,采用动态规划查找最大概率路径，找出基于词频的最大切分组合.

D,基于Viterbi算法做词性标注.(词:动词，名词，形容词等)

E,基于TF-IDF和TEXTRANK模型抽取关键词.

### (3)正则匹配

聊天信息中有表情符合，语音信息，特殊符号等，使用正则匹配去除或保留有用的信息.

例如句子：1，{谢了[39][39]}，2，{</e>2</e>你来啦}，

3，{ <voice filename=11219000001462\_1547399767.amr len=5 content=你看不到我发给你那个语音吗>}

1需要把[39]这种形式去掉，在python中使用re.compile(“\\[\\d\*?\\]”)之后1句变成{谢了}

2句使用re.compile(“<\/e>d\*?\<\/e>”)之后2变成{你来了}

3句使用re.compile(“voice.\*?content=(.\*?)>”)之后3变成{你看不到我发给你那个语音吗}

### (4)为什么需要分词

因为分完词之后一个是一个个词序列，词序列可以通过word2vec来处理成词向量，词向量就可以做运算了.

## 2，对数据做词向量处理

使用Word2Vec算法对所有的数据做词向量处理.并保存词向量.

### (1)Word2Vec简介

1中说到把中文转为词向量就可以做运算了，这里的工作就是把词转为向量的形式，例如一个句子the dog bark at the mailman.

假如用向量来表示每个单词，我们最先想到的是使用one-hot编码的方式来表达每个但是，如下：

the 表示为[1,0,0,0,0]

dog 表示为[0,1,0,0,0]

bark 表示为[0,0,1,0,0]

mailman 表示为[0,0,0,0,1]

我们可以看到每个单词其实就用一个向量来表示。我们发现几个问题：

第一，每个向量之间相互正交的（内积为0）。也就是说词与词之间是没有任何关系的。

第二，每个词的向量维度和不同词的个数有关。比方说，在上面的这个句子里，有5个不同的词，所以向量的维度是5。然而往往现实中，一篇文章的中不同的词的个数是很多的。这样，向量的维度会非常的高。

这种对词的向量表示方法存在着以上的问题，忽略了词与词之间的关系（比如说,dog和cat都是动物，并非完全无关系）。维度太高的向量作为模型输入会导致模型变的复杂（模型复杂需要更多的训练样本才能够收敛）

那么是否存在一种更加合理的对词的用向量的表达方式呢？有的，其实就是word embedding。

word embedding说的简单一点其实就是将高维稀疏的向量降维成稠密的低维向量。（一种高维到低维的映射）

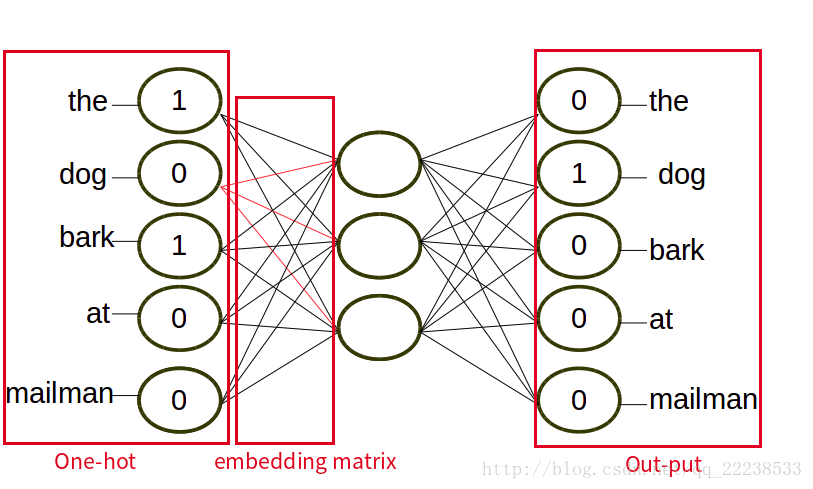
那么如和实现word embedding？

w2v其实就是一种高效实现word embedding的工具。

所以，如果用一句话来描述word2vector的话你会怎么描述？简单的说，我觉得可以这么说w2v其实是一种将单词转化向量的工具。这一种向量的其实是对单词更有效的表达。

### (2)word2vec模型

它可以看作将一个高维空间映射到低维空间的过程，用一个单层神经网络就可以实现这种共功能.如下图所示;



这里有几个细节：

(1)其中的SUM意思其实就是把各个上下文的词one-hot后的向量相加。

比如对于the的向量是[1,0,0,0,0]。bark向量是[0,0,1,0,0],SUM之后就是[1,0,1,0,0]这就是网络的输入。

输出就是[0,1,0,0,0]-dog

(2)我们所谓的embedding vetcor其实就第二个红框里的线，每一根线其实就是一个权值。

(3)第二个框里的红线其实就是dog这个单词的embedding结果（由5维变成3维）

(4)这个单层NN训练完毕之后有用的部分就是embedding martrix这部分，其大小为 输入个数（词汇表长度）×embedding后的维度。

## 3，one\_path聚类算法

由于我们的数据是无标签的，所有需要先对数据做聚类，看看数据的性能.

这个工作使用了one\_path聚类算法，也叫做一趟聚类.

One\_path算法步骤：

(1) 初始时从数据集读入一个新的对象

(2) 以这个对象构建一个新的簇

(3) 若达到数据集末尾，则转6，否则读入一个新的对象；计算它与每个已有簇之间的距离，并选择与它距离最小的簇。

(4) 若最小距离超过给定的阈值r，转2

(5) 否则将对象并入该簇，并更新簇心，转3

(6) 结束

在本算法中，采用的是句子与句子向量之间的余弦相似度.

公式如下：



one\_path算法过程如图所示：

原始数据

1

2

3

1

3

2

4

3

4

5

一趟聚类

聚类数据

1

1

2

2

3

3

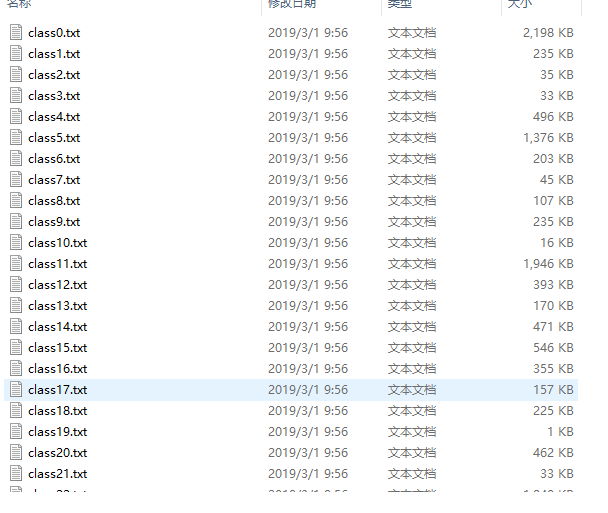
3

4

4

5

对于我们的数据我们可以得出很多类如下图所示



class5类的内容如下图;



## 4，打标签

2步骤聚出很多类，然后与客服那边合作打标签.我做了一个半自动的ui界面，客服小伙伴只需先把路径修改好在运行就可以直接打标签了.

### (1)打标签规则：

拉人敏感信息实例

拉人信息主要是：进q群x-x-x送金币。加微信x-x-x，送金币。加我微信x-x-x进联盟群不加自退。新手嘛，加微信x-x-x一起玩。等…

以上的信息可能是繁体字，拼音，空格，特殊符号等一起写在敏感信息里面，例如

峮 ： 六 - 七 - 四 - 七 - 四 - 九 - 二 - 零 - 八

我 才 玩 的 ， 嘻嘻 \* 迦 . 个 . 微 . 星 一块 玩 啊

这很有可能是拉人信息，所以就当作拉人信息

纯字母和数字的组合(长度大于4)

例：cyct130227, baby340, s19976297, fangwen836612

这种默认当作微信号，标记为拉人信息。

纯数字:qq号(至少是6个数字最多10位),手机号11位

18502843380 明显是手机号不作为敏感信息处理

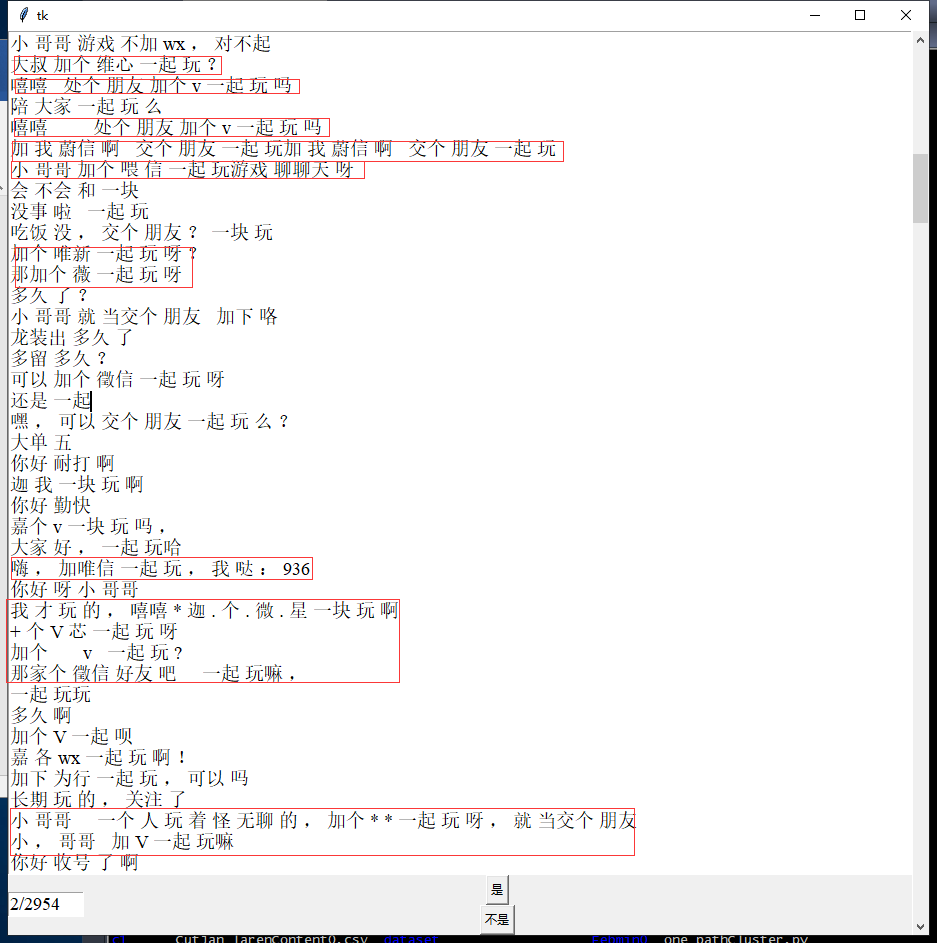
小于6位的纯数字，不是拉人敏感信息

1776185942 10位数字，当作qq号，是拉人敏感信息

网络流行语;222333,666666,这些都不是拉人信息

### (2),如何判定文本是不是拉人敏感信息

如图所示：



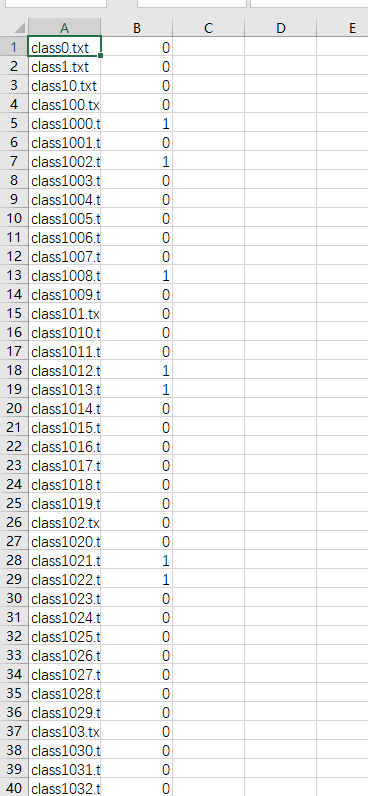
上图中红框内的信息是拉人敏感信息，

这篇文章中大约有40行，其中拉人敏感信息有15行占比为37.5%达到拉人信息占比，此文本标记为拉人信息。拉人敏感信息再一个文本中占比达到10%以上即可标记为拉人信息文本。

## 5，标签label

把打完标签数据中的敏感信息会生成一个label.csv文件记录敏感信息类，在3中是敏感信息点击’是’会生成一个标签为1的数字，不是敏感信息会生成一个标签为0的数字.并保存每个敏感类的聚类中心.再观察聚类中心，看是否可以合并以便提高算法的效率.

如图;



# 三：Textcnn模型

我们的模型主要是做二分类。一类是正常信息，一类是拉人敏感信息.我们的输入是使用w2v训练出的embedding向量输入到网络中进行分类.

## 1，CNN模型

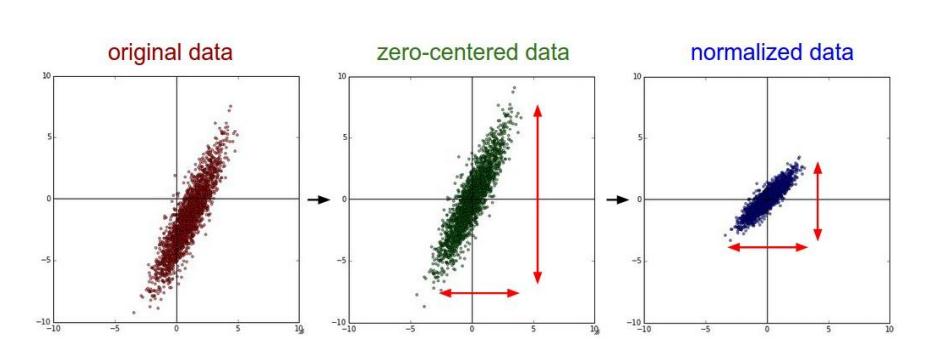
### **(1).卷积神经网络的层级结构**

数据输入层/ Input layer  
卷积计算层/ CONV layer  
ReLU激励层 / ReLU layer  
池化层 / Pooling layer  
全连接层 / FC layer

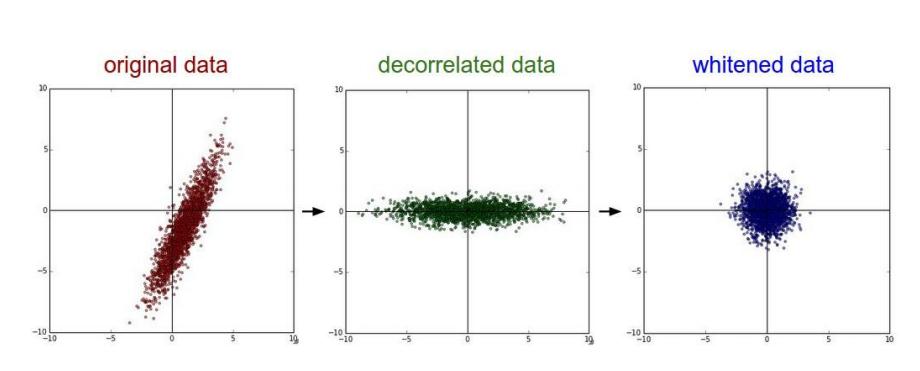
### **(2).数据输入层**

该层要做的处理主要是对原始图像数据进行预处理，其中包括：  
　　•去均值：把输入数据各个维度都中心化为0，如下图所示，其目的就是把样本的中心拉回到坐标系原点上。  
　　• 归一化：幅度归一化到同样的范围，如下所示，即减少各维度数据取值范围的差异而带来的干扰，比如，我们有两个维度的特征A和B，A范围是0到10，而B范围是0到10000，如果直接使用这两个特征是有问题的，好的做法就是归一化，即A和B的数据都变为0到1的范围。  
　　• PCA/白化：用PCA降维；白化是对数据各个特征轴上的幅度归一化

去均值与归一化效果图：



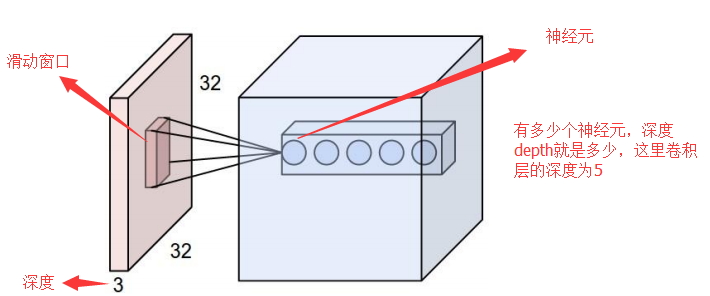
去相关与白化效果图



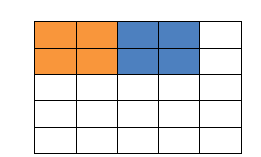
### **(3).卷积计算层**

这一层就是卷积神经网络最重要的一个层次，也是“卷积神经网络”的名字来源。  
在这个卷积层，有两个关键操作：  
　　• 局部关联。每个神经元看做一个滤波器(filter)  
　　• 窗口(receptive field)滑动， filter对局部数据计算

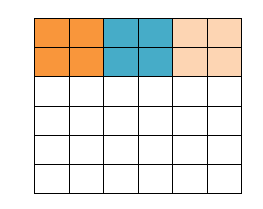
先介绍卷积层遇到的几个名词：  
　　• 深度/depth（解释见下图）  
　　• 步长/stride （窗口一次滑动的长度）  
　　• 填充值/zero-padding



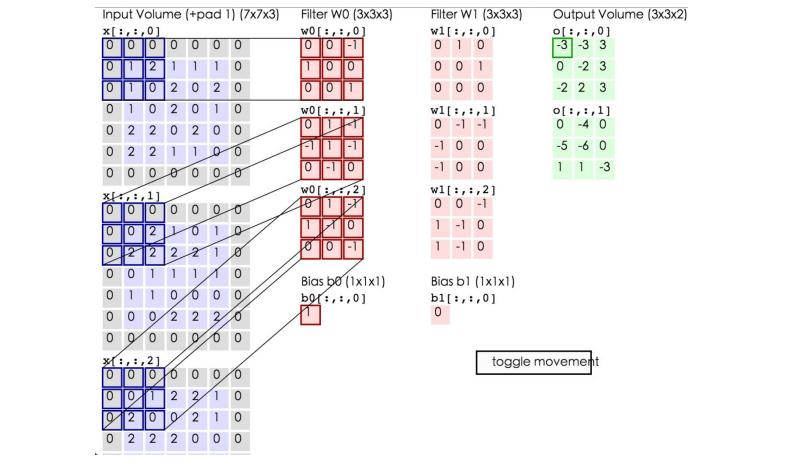
比如有这么一个5\*5的图片（一个格子一个像素），我们滑动窗口取2\*2，步长取2，那么我们发现还剩下1个像素没法滑完，那怎么办呢？



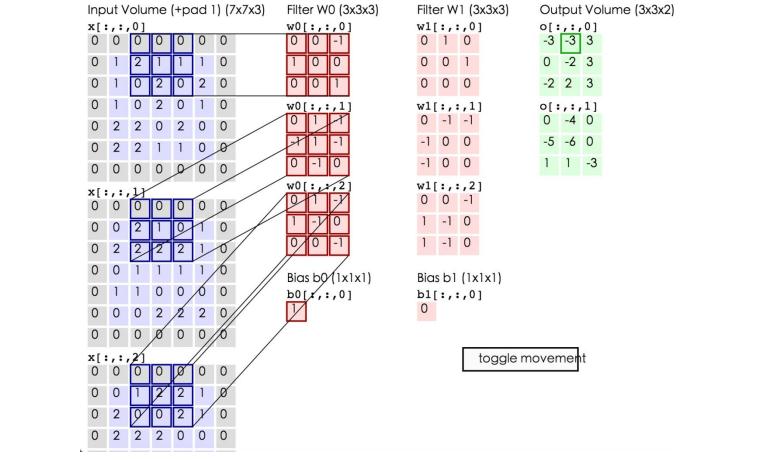
那我们在原先的矩阵加了一层填充值，使得变成6\*6的矩阵，那么窗口就可以刚好把所有像素遍历完。这就是填充值的作用



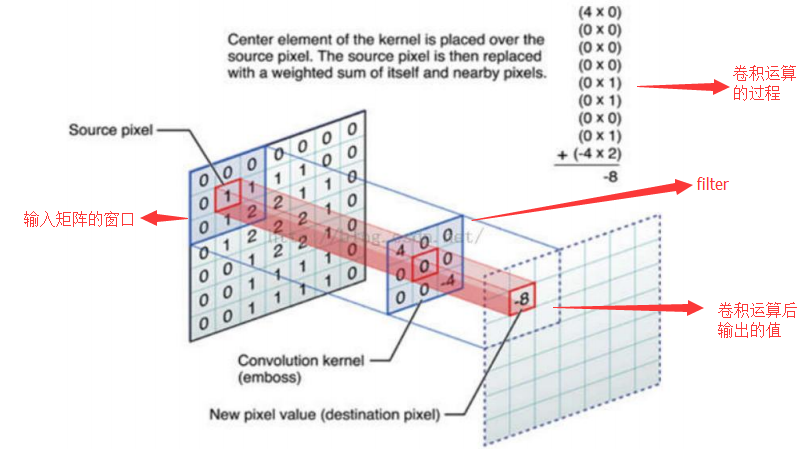
卷积的计算（注意，下面蓝色矩阵周围有一圈灰色的框，那些就是上面所说到的填充值）



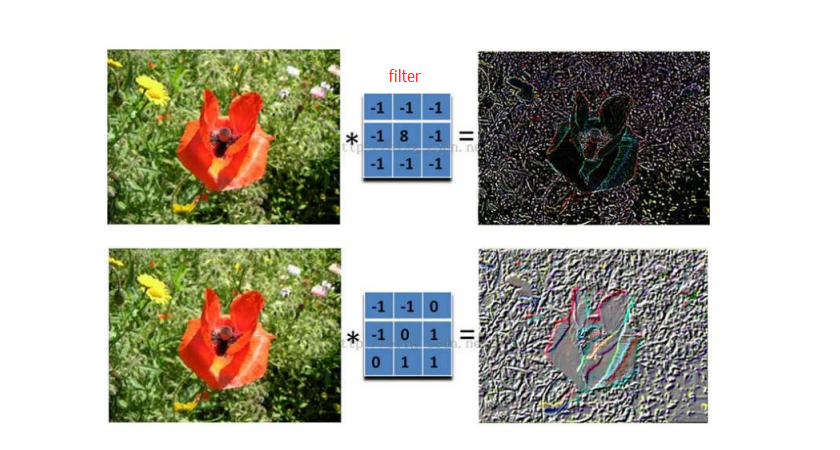
这里的蓝色矩阵就是输入的图像，粉色矩阵就是卷积层的神经元，这里表示了有两个神经元（w0,w1）。绿色矩阵就是经过卷积运算后的输出矩阵，这里的步长设置为2.



蓝色的矩阵(输入图像)对粉色的矩阵（filter）进行矩阵内积计算并将三个内积运算的结果与偏置值b相加（比如上面图的计算：2+（-2+1-2）+（1-2-2） + 1= 2 - 3 - 3 + 1 = -3），计算后的值就是绿框矩阵的一个元素。

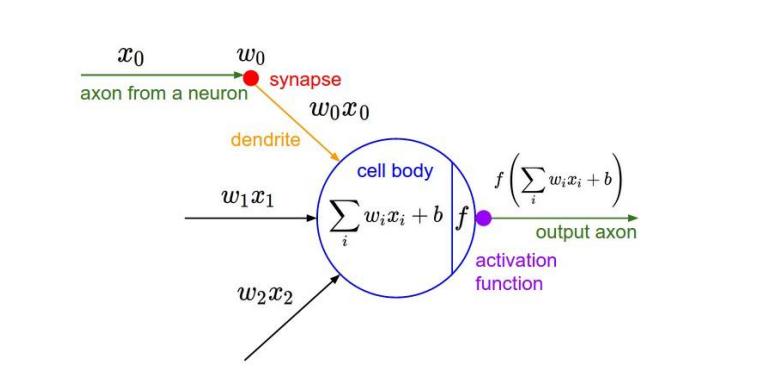


参数共享机制  
　　• 在卷积层中每个神经元连接数据窗的权重是固定的，每个神经元只关注一个特性。神经元就是图像处理中的滤波器，比如边缘检测专用的Sobel滤波器，即卷积层的每个滤波器都会有自己所关注一个图像特征，比如垂直边缘，水平边缘，颜色，纹理等等，这些所有神经元加起来就好比就是整张图像的特征提取器集合。  
　　• 需要估算的权重个数减少: AlexNet 1亿 => 3.5w  
　　• 一组固定的权重和不同窗口内数据做内积: 卷积

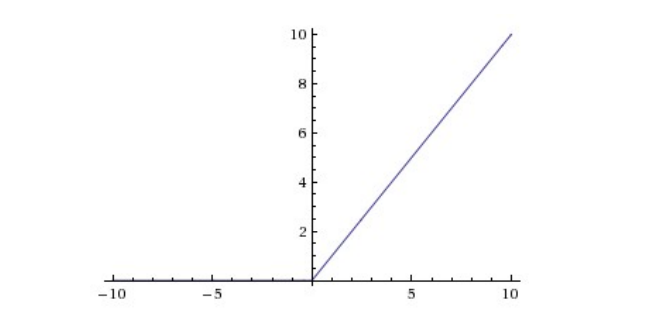


### **(4).激励层**

把卷积层输出结果做非线性映射。



CNN采用的激励函数一般为ReLU(The Rectified Linear Unit/修正线性单元)，它的特点是收敛快，求梯度简单，但较脆弱，图像如下。



激励层的实践经验：  
　　①不要用sigmoid！不要用sigmoid！不要用sigmoid！  
　　② 首先试RELU，因为快，但要小心点  
　　③ 如果2失效，请用Leaky ReLU或者Maxout  
　　④ 某些情况下tanh倒是有不错的结果，但是很少

### **(5).池化层**

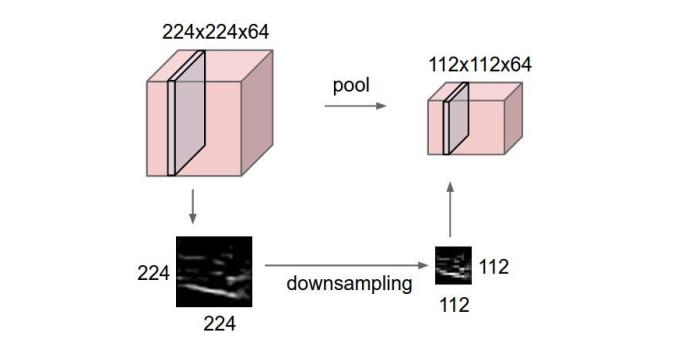
池化层夹在连续的卷积层中间， 用于压缩数据和参数的量，减小过拟合。  
简而言之，如果输入是图像的话，那么池化层的最主要作用就是压缩图像。

这里再展开叙述池化层的具体作用。

1. 特征不变性，也就是我们在图像处理中经常提到的特征的尺度不变性，池化操作就是图像的resize，平时一张狗的图像被缩小了一倍我们还能认出这是一张狗的照片，这说明这张图像中仍保留着狗最重要的特征，我们一看就能判断图像中画的是一只狗，图像压缩时去掉的信息只是一些无关紧要的信息，而留下的信息则是具有尺度不变性的特征，是最能表达图像的特征。

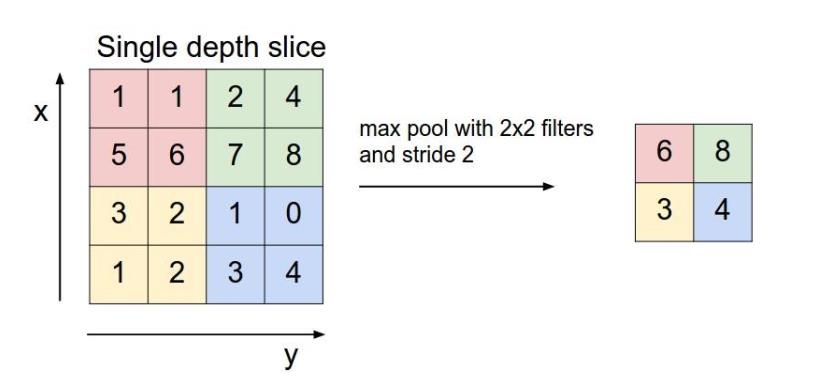
2. 特征降维，我们知道一幅图像含有的信息是很大的，特征也很多，但是有些信息对于我们做图像任务时没有太多用途或者有重复，我们可以把这类冗余信息去除，把最重要的特征抽取出来，这也是池化操作的一大作用。

3. 在一定程度上防止过拟合，更方便优化。



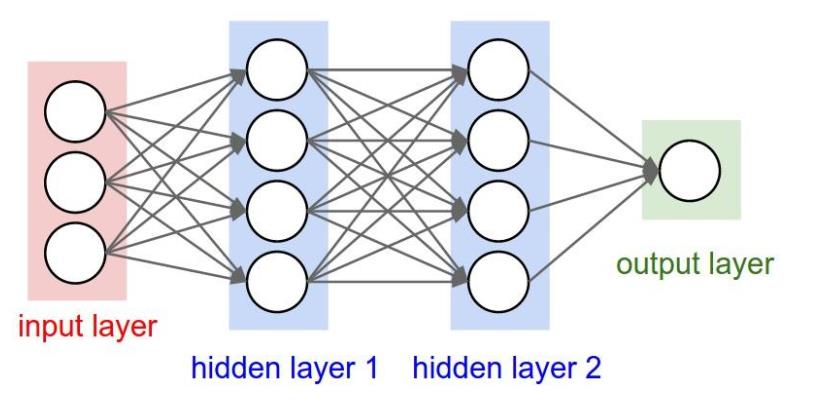
池化层用的方法有Max pooling 和 average pooling，而实际用的较多的是Max pooling。

这里就说一下Max pooling，其实思想非常简单。



对于每个2\*2的窗口选出最大的数作为输出矩阵的相应元素的值，比如输入矩阵第一个2\*2窗口中最大的数是6，那么输出矩阵的第一个元素就是6，如此类推。

(6).全连接层  
两层之间所有神经元都有权重连接，通常全连接层在卷积神经网络尾部。也就是跟传统的神经网络神经元的连接方式是一样的：



**一般CNN结构依次为**  
　　1. INPUT  
　　2. [[CONV -> RELU]\*N -> POOL?]\*M   
　　3. [FC -> RELU]\*K  
　　4. FC

**卷积神经网络之训练算法**  
　　1. 同一般机器学习算法，先定义Loss function，衡量和实际结果之间差距。  
　　2. 找到最小化损失函数的W和b， CNN中用的算法是SGD（随机梯度下降）。

**卷积神经网络之优缺点**  
优点  
　　• 共享卷积核，对高维数据处理无压力  
　　• 无需手动选取特征，训练好权重，即得特征分类效果好  
缺点  
　　• 需要调参，需要大样本量，训练最好要GPU  
　　• 物理含义不明确（也就说，我们并不知道没个卷积层到底提取到的是什么特征，而且神经网络本身就是一种难以解释的“黑箱模型”）

**卷积神经网络之典型CNN**  
　　• LeNet，这是最早用于数字识别的CNN  
　　• AlexNet， 2012 ILSVRC比赛远超第2名的CNN，比  
　　• LeNet更深，用多层小卷积层叠加替换单大卷积层。  
　　• ZF Net， 2013 ILSVRC比赛冠军  
　　• GoogLeNet， 2014 ILSVRC比赛冠军  
　　• VGGNet， 2014 ILSVRC比赛中的模型，图像识别略差于GoogLeNet，但是在很多图像转化学习问题(比如object detection)上效果奇好

**卷积神经网络之 fine-tuning**  
何谓fine-tuning？  
fine-tuning就是使用已用于其他目标、预训练好模型的权重或者部分权重，作为初始值开始训练。

那为什么我们不用随机选取选几个数作为权重初始值？原因很简单，第一，自己从头训练卷积神经网络容易出现问题；第二，fine-tuning能很快收敛到一个较理想的状态，省时又省心。

那fine-tuning的具体做法是？  
　　• 复用相同层的权重，新定义层取随机权重初始值  
　　• 调大新定义层的的学习率，调小复用层学习率

**卷积神经网络的常用框架**

**Caffe**  
　　• 源于Berkeley的主流CV工具包，支持C++,python,matlab  
　　• Model Zoo中有大量预训练好的模型供使用  
**Torch**  
　　• Facebook用的卷积神经网络工具包  
　　• 通过时域卷积的本地接口，使用非常直观  
　　• 定义新网络层简单  
**TensorFlow**  
　　• Google的深度学习框架  
　　• TensorBoard可视化很方便  
　　• 数据和模型并行化好，速度快

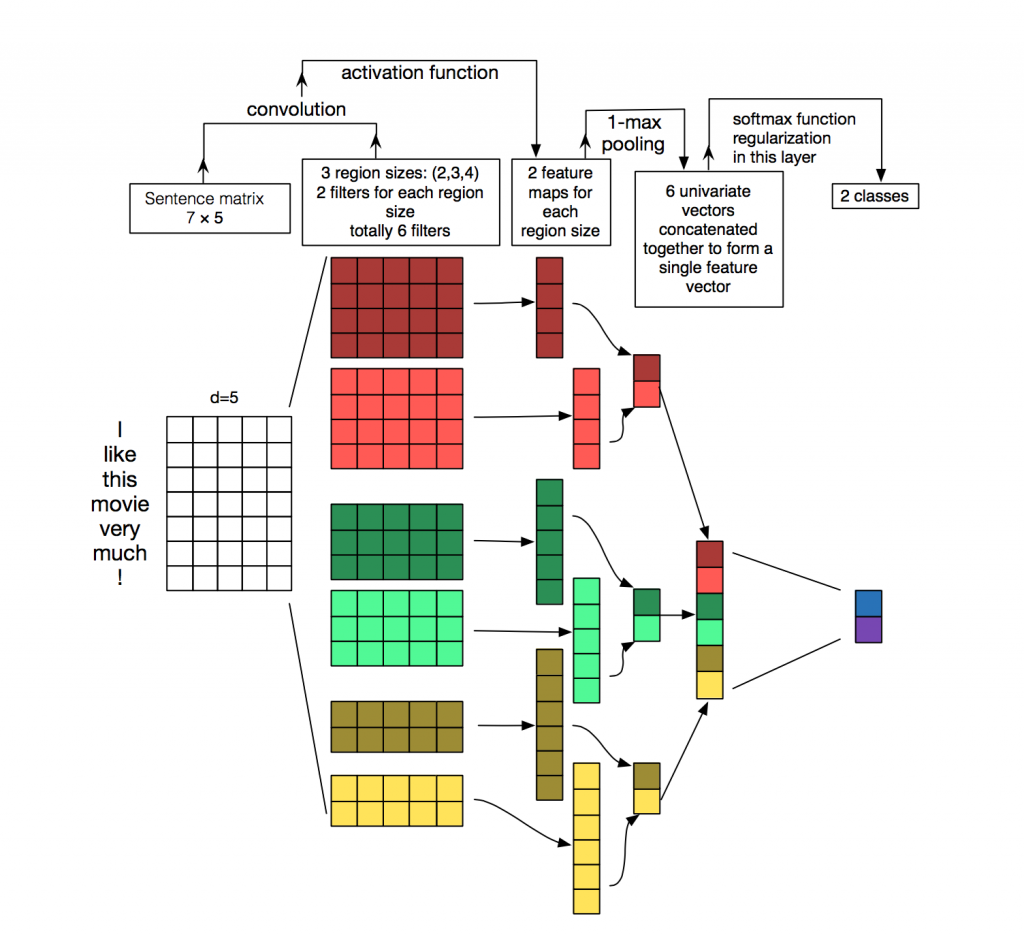
(7)总结  
卷积网络在本质上是一种输入到输出的映射，它能够学习大量的输入与输出之间的映射关系，而不需要任何输入和输出之间的精确的数学表达式，只要用已知的模式对卷积网络加以训练，网络就具有输入输出对之间的映射能力。

CNN一个非常重要的特点就是头重脚轻（越往输入权值越小，越往输出权值越多），呈现出一个倒三角的形态，这就很好地避免了BP神经网络中反向传播的时候梯度损失得太快。

卷积神经网络CNN主要用来识别位移、缩放及其他形式扭曲不变性的二维图形。由于CNN的特征检测层通过训练数据进行学习，所以在使用CNN时，避免了显式的特征抽取，而隐式地从训练数据中进行学习；再者由于同一特征映射面上的神经元权值相同，所以网络可以并行学习，这也是卷积网络相对于神经元彼此相连网络的一大优势。卷积神经网络以其局部权值共享的特殊结构在语音识别和图像处理方面有着独特的优越性，其布局更接近于实际的生物神经网络，权值共享降低了网络的复杂性，特别是多维输入向量的图像可以直接输入网络这一特点避免了特征提取和分类过程中数据重建的复杂度。

CNN基础知识参考Madcola的博客园

## 2，CNN文本分类模型



简单的分析一下：

（1）我们假定输入CNN的数据是二维的，其中每一行表示一个样本（即一个字词），如图中“I”、“like”等。每一个样本（字词）有d个维度，可以看成是词向量长度，即每个字词的维度，程序中用embedding\_dim表示。

（2）使用CNN的卷积对这个二维数据进行卷积：在图像的CNN卷积中，卷积核的大小一般是3\*3，5\*5等，但在NLP中就不就不能这么搞了，因为这里的输入数据每行是一个样本了！假设卷积核的大小为[filter\_height,filter\_width],那么卷积核的高度filter\_height可以为1，2，3等任意值，而宽度filter\_width只能是embedding\_dim的大小，这样才能把完整的样本框进去。

(3)Textcnn的主要参数

max\_sentence\_length = 300 # 最大句子长度，也就是说文本样本中字词的最大长度，不足补零，多余的截断

 embedding\_dim = 128 #词向量长度，即每个字词的维度

 filter\_sizes = [3, 4, 5, 6] #卷积核大小

 num\_filters = 200  # Number of filters per filter size 卷价个数

 base\_lr=0.001      # 学习率

 dropout\_keep\_prob = 0.5

 l2\_reg\_lambda = 0.0  # "L2 regularization lambda (default: 0.0)

## 3，文本处理

文本处理分为两个过程一个是训练时，一个是预测时.但是同一个代码可以两次使用完成此功能。

(1)我们的数据类型有几种比较特殊如下：

那不成蝴蝶了[8][8]

<voice filename=10761000002034\_1548950468.amr len=4 content=有人吗？有人吗？有没有活人>,

看，慢点都怕别人抢了<e>21</e><e>21</e>

以上的数据需要特殊处理使用正则匹配把没用的信息去掉保留有用的信息

代码如下：

#正则

partern=re.compile("\\[\\d\*?\\]")

partern2 = re.compile("\\<e>\\d\*?\\<\/e>")

partern3 = re.compile("voice.\*?content=(.\*?)>")

(2),使用jieba中文分词，gensim来训练word2vec模型

这一步的作用时把文字向量化变成计算机理解的数字形式存储，并代表着原本的文字信息。

(3)训练过程

处理及运算数据，输出模型和参数并保存.

TextCnn模型

输入参数

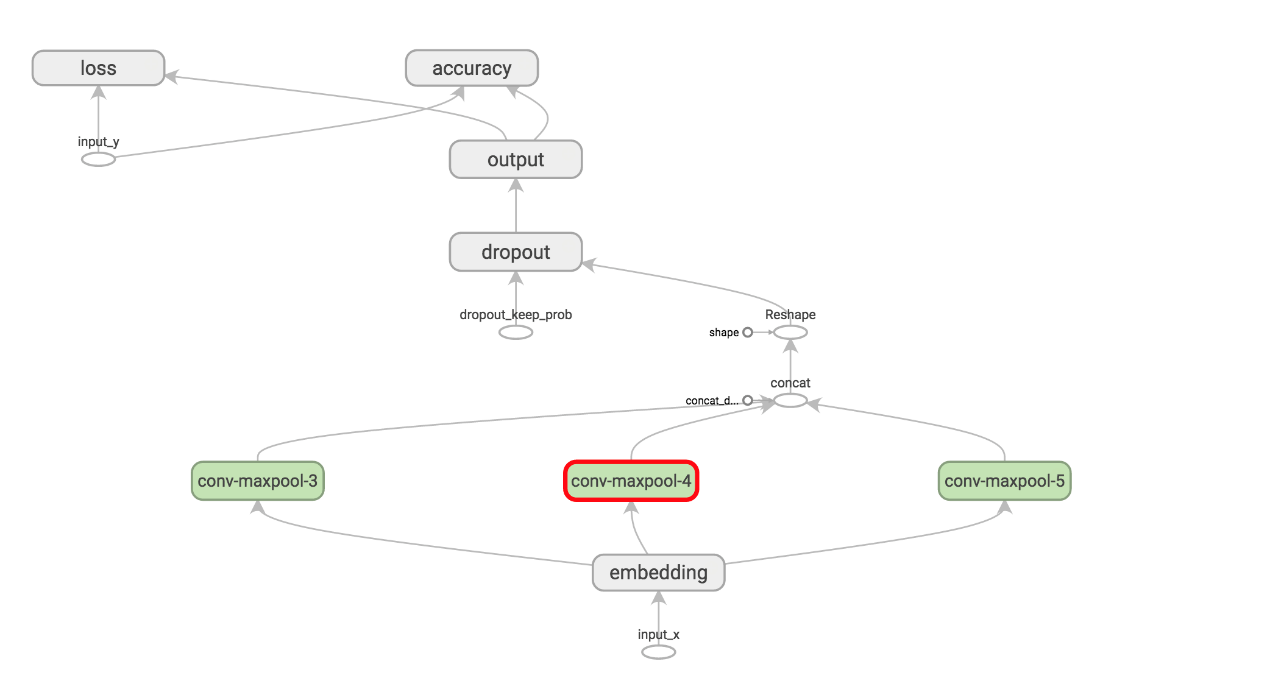
(4)做预测

输出测试结果,并判断是否为敏感信息

处理及运算数据

Load参数model和预测数据

(5)模型处理的模拟图



# 四：Textcnn模型更新与优化

模型优化主要从参数优化，训练集，w2v，dropout层，loss层等方面.

## 模型本身优化

模型参数调参，增加模型的准备度，dropout层的选择通过经验和大量实验做出合理的选择。Loss层同样通过大量实验选择合适的loss函数。

## 模型以外的因素

训练集和w2v的优化，机器硬件的支持等。

# 五：接口和输出

我的工作主要是构造Textcnn模型，构造完成之后需要上线运行。需要数据中心和其他部门的支持。

## 输入数据

从数据中心那里实时输入，使用kafka，flume等工具。

具体的内容需要再详细商讨。

## 输出数据

在textcnn模型处理之后输出处理后的数据，到数据库或者其他地方供客服人员调用。