数据内容：

<http://www.appzapp.us/Index.html>网站上16000余个app的名字，点击量，点赞量，链接，类型。

图表的贮存路径：

\img\

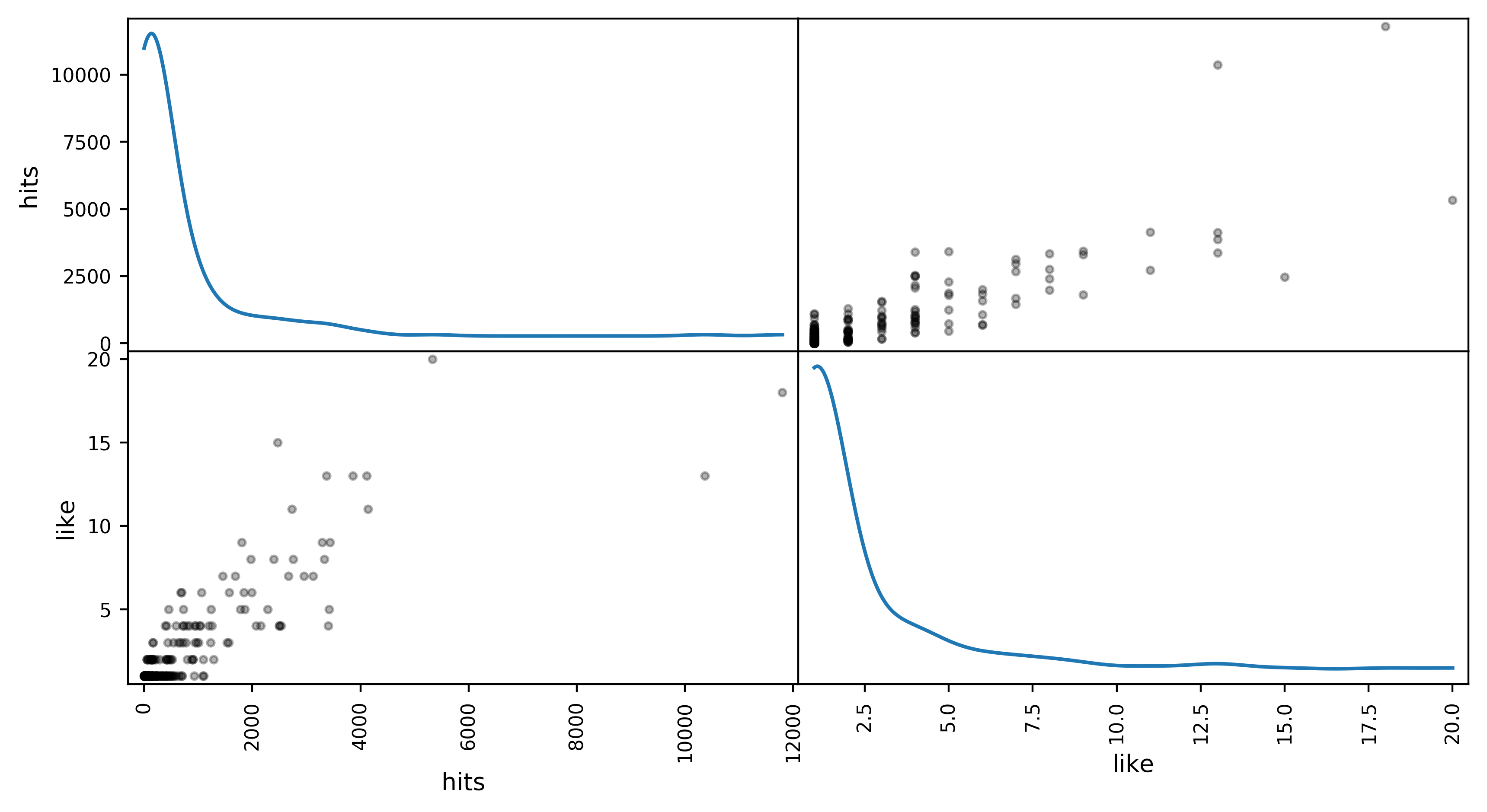
数据存贮路径：

\File\

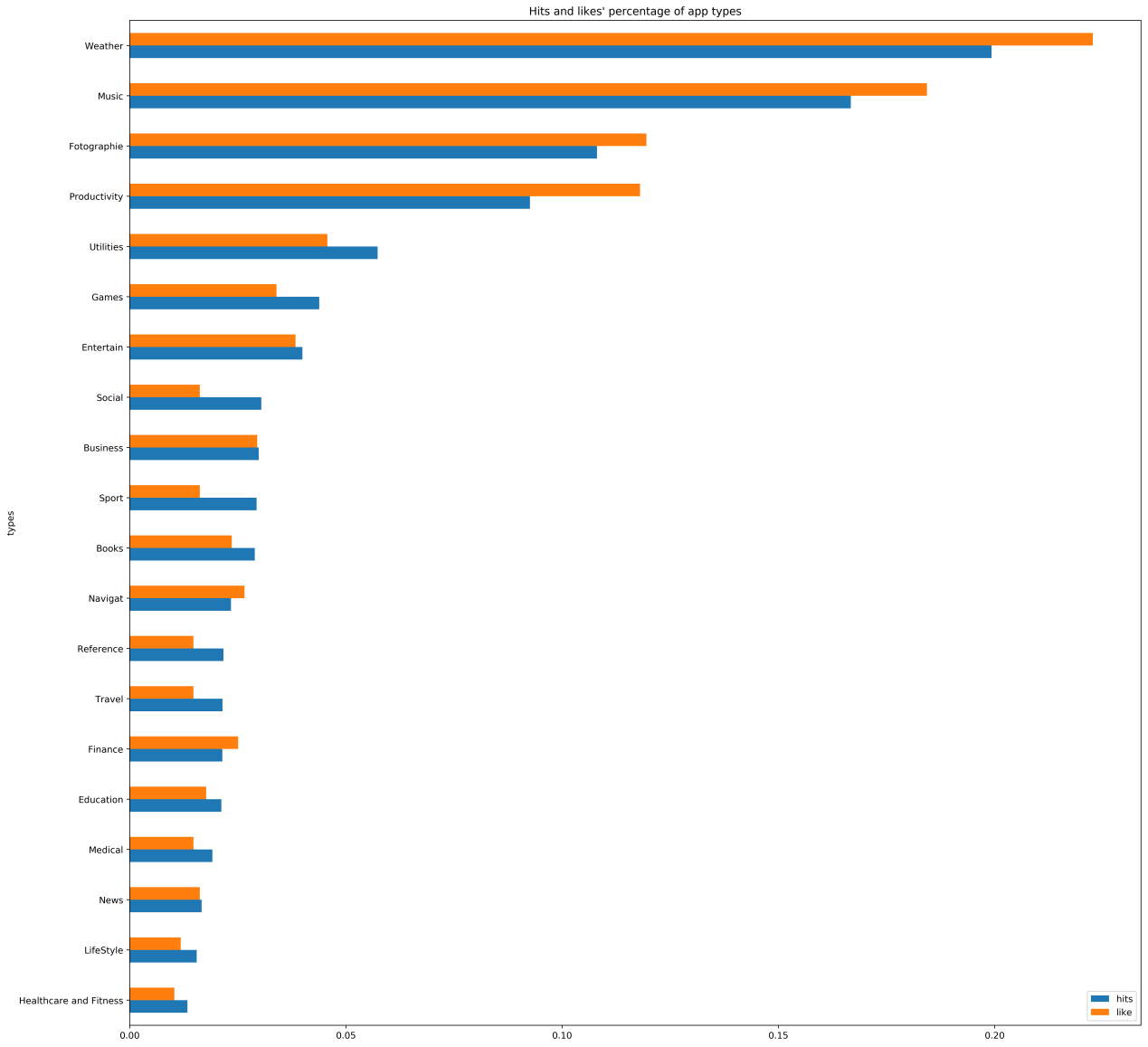
（一开始是不打算存储整个app表的数据的，但是发现实际上数据的大小并不大，于是就一起存在了File文件夹中，但是代码实现的是从数据库中读写原始数据，并且在File中存储的数据是经过处理后的数据，所以这些数据仅作为参考）

分析报告：

首先，我由app的点击量与点赞量的数据变化得出协方差为0.85，并获得二者之间的散布图，得到二者之间的关系确实很大，由此进行下一步分析



然后我对各个类型的app的点击量与点赞量占总的点击量和点赞量的百分比作出统计，并得到柱形图



由图表可以发现，点击量最多的是天气类的app，并且，尽管是在不同类型的app中，app的点赞量也依然与点击量呈近乎正相关的关系。

**关于app名字的取名与点击量和点赞量的数据分析**

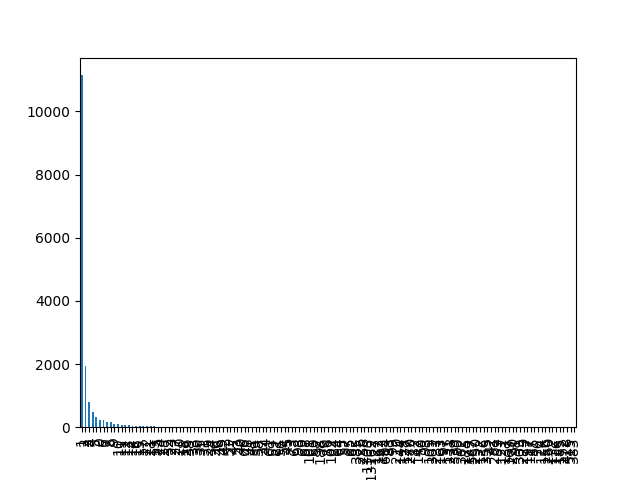
当然，这只是纯粹理论上的分析，因为实际上用户在点击链接时，还包括用户对于页面上app简介，app价格，app图片等一系列信息的了解，以及用户本身对自己要找的某类型app的意向，这些都是影响因素，但由于时间关系，修改爬虫在短时间内再去爬取这些数据会不太容易，主要是爬取的时间太长，所以建议留待以后改进。

首先，我将app的名字分割成多个词，洗掉一些奇怪的词语，然后将所有的词转换成小写，并将这些词分别保存起来，这些词我们简称其为词集。

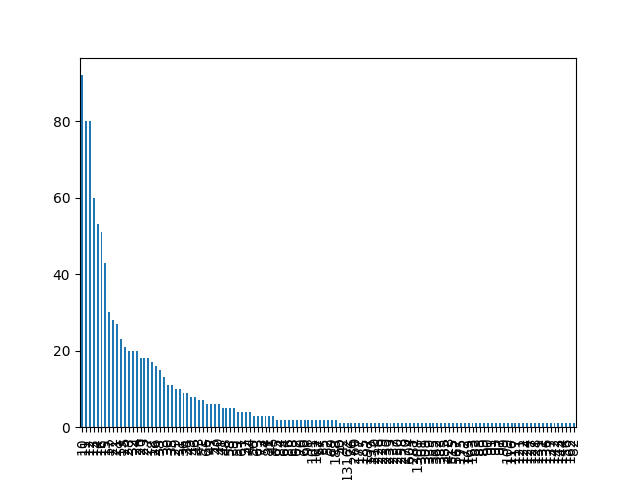
然后，统计出词集中的每一个词在app名字集中出现的次数，从这一点我们可以看到一些app制作者取名的风格。这是一张用词频作出来的词云，用于表现各个词的出现频率（作出词云的代码我放在了Util包里面，这里我用了wordcloud的第三方库）。



统计完之后我还需要得到每个频率在计数表中出现的次数，以便于我区分出一些生活中并不常用的词语，这些词语会对我之后的数据造成很大的影响，因为它们并不是常用词语，但是却会因为某个app点击量特别高而显得它们对点赞量的影响特别大，但事实上，它们并不是什么关键字，只不过是一些奇形怪状的词被恰好用了进去。



这张图表的x轴比较密集，但不难看出，前面的图形相当陡峭，在大概x=10到x=12的时候陡峭程度变缓，我调整代码，将那部分的图片放大



可以看出，在经过12这个词频之后的词的词频变化虽然依然陡峭，但有渐趋平缓的形势。不过，为了更加确定，我还是调用了一下pandas的统计函数

weight frequency weight\_pct frequency\_pct

count 1024.000000 1024.000000 1024.000000 1024.000000

mean 22.716367 48.274414 0.000977 0.000977

std 49.650971 415.609701 0.002134 0.008408

min 0.217391 10.000000 0.000009 0.000202

25% 4.797917 13.000000 0.000206 0.000263

50% 8.254274 19.000000 0.000355 0.000384

75% 18.967391 33.000000 0.000815 0.000668

max 806.461538 13162.000000 0.034669 0.266259

std表示一项数据的标准差，我试了几个范围，确定区分低频词汇的界限确实在词频10左右（上面那个是在没有调整之前的统计数据，将低区间词频调整到了10之后std在66左右），在不包含去除高频词汇的影响的状况下，将词频低于10的词删掉能让std尽可能的降到最低（当然，把低频词区间调的很高也会用同样的效果，但那显然不是因为低频词的关系）。这是目前我能找出的一个稍微比较好一点的解决办法

事实上，在图片靠右部分的词是高频词汇，这个也会影响我的数据，但是在这里不需要对它作出舍弃，因为后面我们可以在计算中消除它的影响。

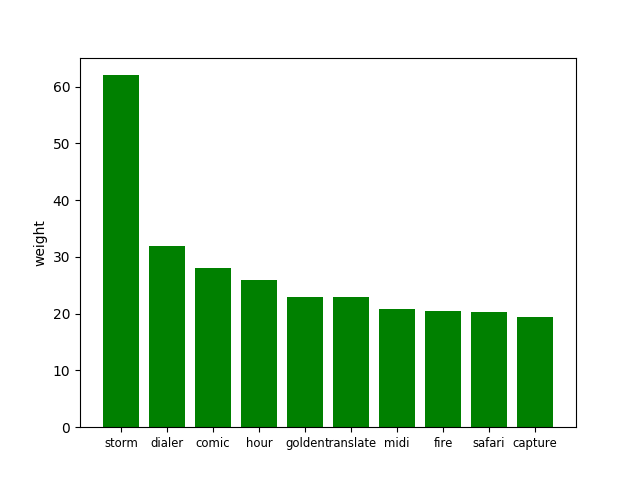
我的计算方式是这样的（当然这也是我自己想出来的，正确的算法应该会复杂的多）：

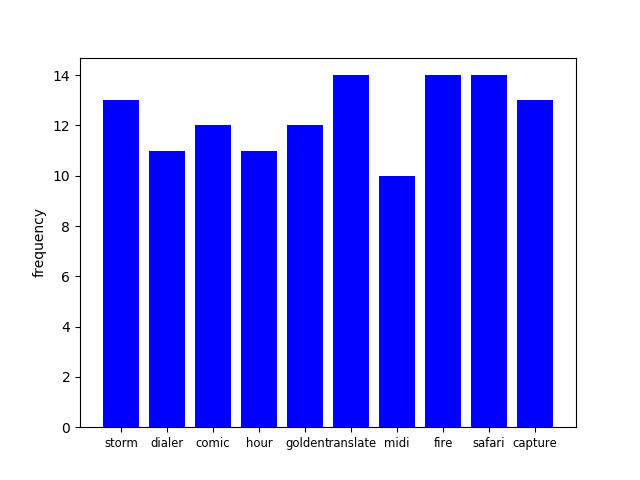
先统计出词集中每个词在app里面时他获得的点击量的数量的总和，这里我简称为词点击量，举个例子，假设有一个app叫‘Angry Bird’，它的点击数有300，还有一个app叫‘Angry grandma’，它的点击数有400，那么‘Angry’这个词的词点击量就是300+400=700，而Bird只有300，不过当然，大小写的影响是要消除掉的。

得到一个词的词点击量之后还要拿它除以它的词频，这就是我前面说的消除高频词的影响的方法，得到的结果便是该词的权数。

简单来说：权数=词点击量/词频

这是我对于这个简单算法的想法，执行之后，我对于权数排名前十的词作了两个柱形表





绘图代码，因为源文件中不便保存绘图代码而这次采用的代码稍微复杂了一点，于是在这里做个记录

‘’’

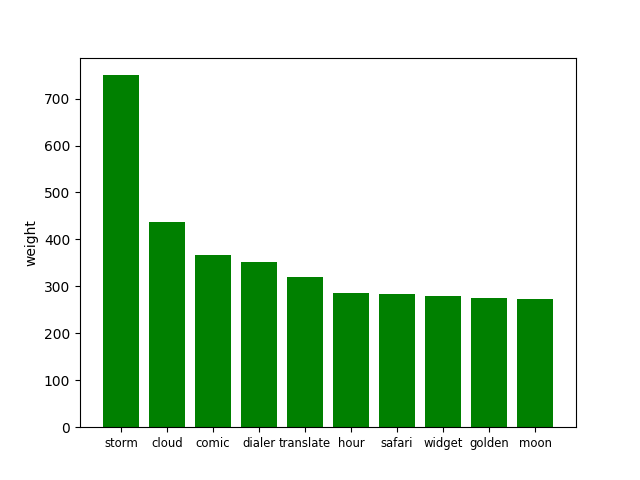
fig,axes=plt.subplots(1,1)  
axes.set\_xticks(range(0,10))  
axes.set\_xticklabels(word\_frame.index[:10],fontsize='small')  
axes.set\_ylabel('frequency')  
axes.bar(range(0,10),word\_frame['frequency'][:10],color='b')  
plt.show()

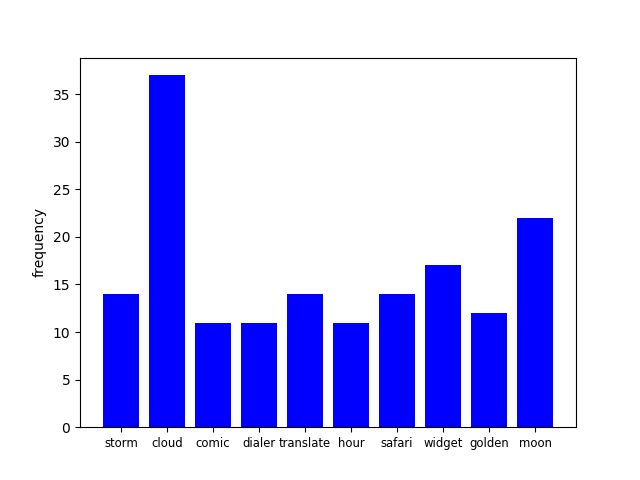
’’’

根据数据来看，实际上的情况貌似并不如我所愿，因为能吸引点击量的词汇也往往可能会因为被用作高频词汇，从而被一些点击量少的app拉低这个词的权数，异常数据依然存在影响，得出这个结论，是因为被判作高权词汇的词，词频太接近10，我又去观察了其它的权数排名靠前的词，发现词频都相当低，这意味这我对于高频词汇的权数限制在某些地方出了问题。

于是我做了两个改动，在生成数据表的最开始，先将点击数为0的app去除掉，因为它们的零点击量会对词频产生影响，会过分减少词权，然后我将词的词点击量也加入到后面的计数表中，看看能不能通过观察这一列数据来找到更好的计算方式。

下面是修改之后的权数前十的词语的权数以及词频





修改了之后的分析结果显然有一部分变化，权数的图表似乎还是与之前相似,除了x轴上的词有所变化了,而词频的图表就与之前区别较为明显了,不再是词频越低权对应数越高,至少,我们能看出来,cloud和moon这两个词,哪怕被较高频的使用,平均到每一个app上也依然引起了很多人的点击.

到这一步，再需要加大深度就不是那么容易了，而且这个数据分析结果本身，也还是具有很大的不确定性的，像是上面说的cloud，可能是因为本身带有cloud字样的app就是一些与云计算有关的app，像是百度云，它们本身就具有很强的功能性能吸引大量用户，与名字取得如何关系不大。

不过，这项分析结果，也从一面体现出哪些app的功能会比较受欢迎，像是comic，显然是指动漫类app，translate，显然是指翻译类app，比起苹果官网自己提供的十九个大分类，我能更清楚地感受到在功能分类细化之后，究竟是哪类app更吸引用户点击，这算是个意外之喜，起码我的这次分析结果也不是那么不科学。