设计文档

一、编写目的

算法设计，首先需要把确定输入和输出，然后建立算法模型，建立算法模型需要进一步将算法分解成简单的模块，遵循模块划分独立性原则，使划分的模块的功能对大多数程序员都是易懂的。

算法模型设计过程中需要处理庞大的数据量，需要考虑算法模型在性能方面的优势和劣势，针对劣势，作出相应的处理。

二、背景

在电子商务网站中，用户在购买服饰时，经常有这样的需求，例如用户准备购买一件喜欢的上衣的同时往往希望选择一条与之搭配的裤子。穿衣搭配是电商场景下服饰鞋包导购中非常重要的课题。

为了解决这个需求，本课设拟通过足够多的数据集，同时利用商品图像信息以及文字信息等综合来设计穿衣搭配系统。基于天池淘宝穿衣搭配算法竞赛，我们在该比赛公开的数据集中获取数据，包括文本和图像数据。

三、概要设计

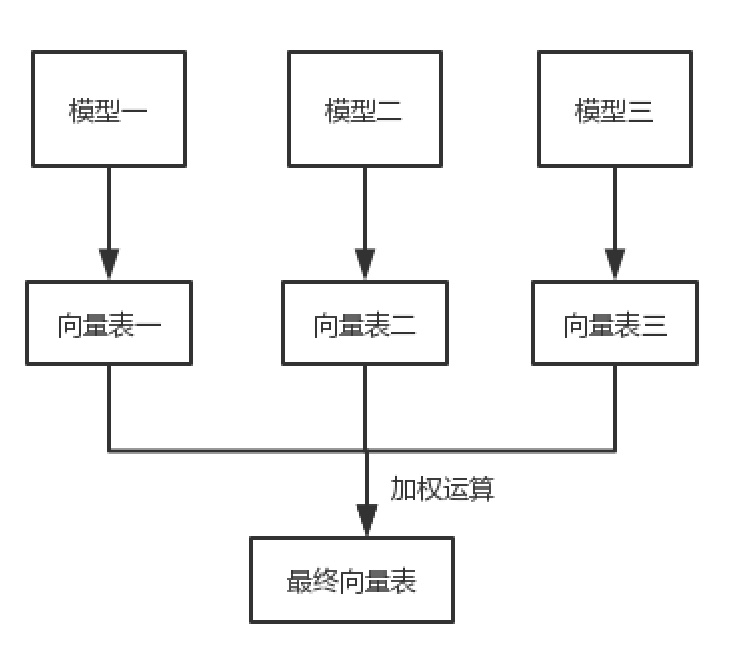
1、系统组织设计

本课设拟使用三个模型来获取商品的搭配集。三个模型分别是基于专家搭配集获取商品的搭配集合、基于商品标签分析法、基于用户历史行为计算搭配度。每个模型通过数据分析获得一个搭配集字典，每个字典的格式如下：

|  |  |
| --- | --- |
| 商品id | 搭配商品的id：搭配度 |
| 123 | 234：0.66,1123：0.54 |

对获取的三个字典进行权重打分，根据分数排名获得一个新的搭配集。该搭配集就是最后的结果。

系统设计如下：



2、设计全局数据库

提供了三个数据表：

商品基本信息数据(文本、图像)



用户历史行为数据



搭配套餐数据



**向量表格式：**

|  |  |
| --- | --- |
| 商品id | 相似度 |
| 例如：123 | 例如：50% |
| ... | ... |

3、数据结构

主要数据结构如下：

**TF-IDF文本处理：**

|  |  |  |
| --- | --- | --- |
| 类名 | 函数方法 | 作用 |
| tfIdf | \_\_init\_\_() | 初始化商品信息 |
| build\_dict() | 建立词表，生成一个以商品标签为索引的字典，并进行数据处理 |
| cal\_tf() | 计算TF词频(Term Frequency) |
| cal\_idf() | 计算IDF逆向文件频率（Inverse Document Frequence） |
| cal\_tf\_idf() | 计算一个商品的所有特征值的TF\_IDF |
| cal\_dict\_times() | 计算字典频次 |
| sim() | 单个计算计算TF\_IDF |

**参数处理：**

|  |  |  |
| --- | --- | --- |
| 类名 | 函数方法 | 作用 |
| params | \_\_init\_\_() | 初始化全局变量 |
| total\_params | \_\_init\_\_() | 初始化与文件有关的全局变量 |
| mini\_params | \_\_init\_\_() | 初始化处理后文件对应的全局变量 |

**计算类间搭配表：**

|  |  |  |
| --- | --- | --- |
| 类名 | 函数方法 | 作用 |
| fashion\_cat | \_\_init\_\_() | 初始化类目结构 |
| set\_item\_cat\_file() | 处理类目文件 |
| item\_cat() | 读取类目文件 |
| item\_to\_catogory() | 获取类间搭配表 |

四、详细设计

利用三个模型获得某商品对应的搭配集（向量表），最后加权运算融合成最终的搭配集。

1、模型一

该模型基于专家搭配集获取商品的搭配集合计算搭配度。

**第一步：数据处理**

除去缺省数据

**第二步：获取类间搭配表**

该模型将已提供的专家搭配集，即“搭配套餐数据”，与另一个表，即商品信息表联合起来得到一个类间表，这个类间搭配表最后的格式是：

|  |  |
| --- | --- |
| 商品所属类目id（cat\_id） | 搭配商品所属类目id(match\_cat\_id) |
| 12 | 16 |

联合类间搭配表的主要代码设计：



程序主要是进行两个表之间的联合。

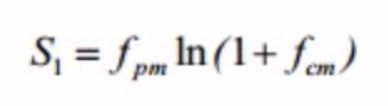
**第三步：计算频次**

输入一个商品A，之后输入一个商品A，在用户购买记录表中搜索出所有购买过该商品A的用户与购买时间，再用用户ID和购买时间在用户购买记录表中找该用户在该天购买的其他商品B,C,D…，查找过程中，过滤与A相同或者与A同类的商品。同时记录商品对（比如A与B）的同天购买次数ｆpm ，再查找商品对的类间搭配频数，该频数首先要查找这两个商品所属类别（比如Ca和Cb），用所属类别遍历类间搭配表， 查找Ca和Cb类的搭配频数ｆcm 。

**第四步：计算搭配度S1**

这里的类间搭配频数ｆcm 代表的不同类的搭配程度，频数越大，我们认为这两个类Ca和Cb的搭配程度就越高，反推A与B商品的搭配可能性就越大。这里给出我们的对于这个模型设计的评价函数： S1 越大，我们认为这两个商品越搭配。

通过以下公式计算搭配度S1：



我们找到了与A商品同天都买的商品B,C,D,E….. 计算每一对的S1 将这些数据排序输出。若S1算出来为0，说明ｆcm为0，则该两个商品类Ca和Cb不搭配，推出这两个商品A与B不搭配。

程序设计代码如下：



其中，参数self表示商品变量。

程序主要分为三个部分：

* 找到该商品对应的历史记录
* 获取该历史记录中商品对应的搭配商品
* 计算搭配商品出现的频次
* 利用频次计算搭配度

2、模型二

该模型利用商品标签计算商品搭配度，使用的是TFIDF文本相似度全局搜索的算法。

该算法中，用TF-IDF文本处理与余弦相似度匹配方法，查找与待测商品A相似但不同类的商品B,C,D….我们认为这些商品是可能与商品A搭配的商品。

该算法用以评估一个商品中的一个特征值对于该商品的重要程度。 TFIDF的主要思想是：如果某个词或短语在一篇文章中出现的频率TF高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。

TFIDF即TF \* IDF：

* TF词频(Term Frequency)，表示词条在文档d中出现的频率
* IDF逆向文件频率(Inverse Document Frequency)。表示如果包含词条t的文档越少，也就是n越小，IDF越大，则说明词条t具有很好的类别区分能力。

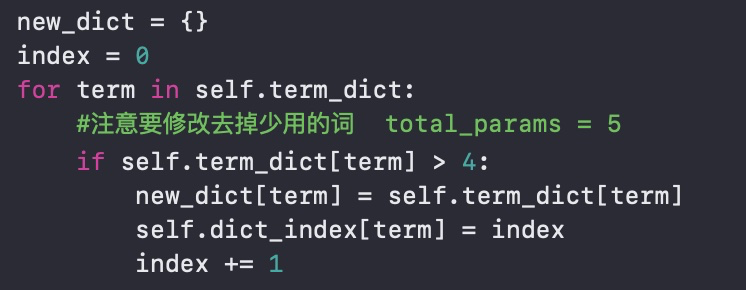
**第一步：数据处理**

词表的数据格式如下：

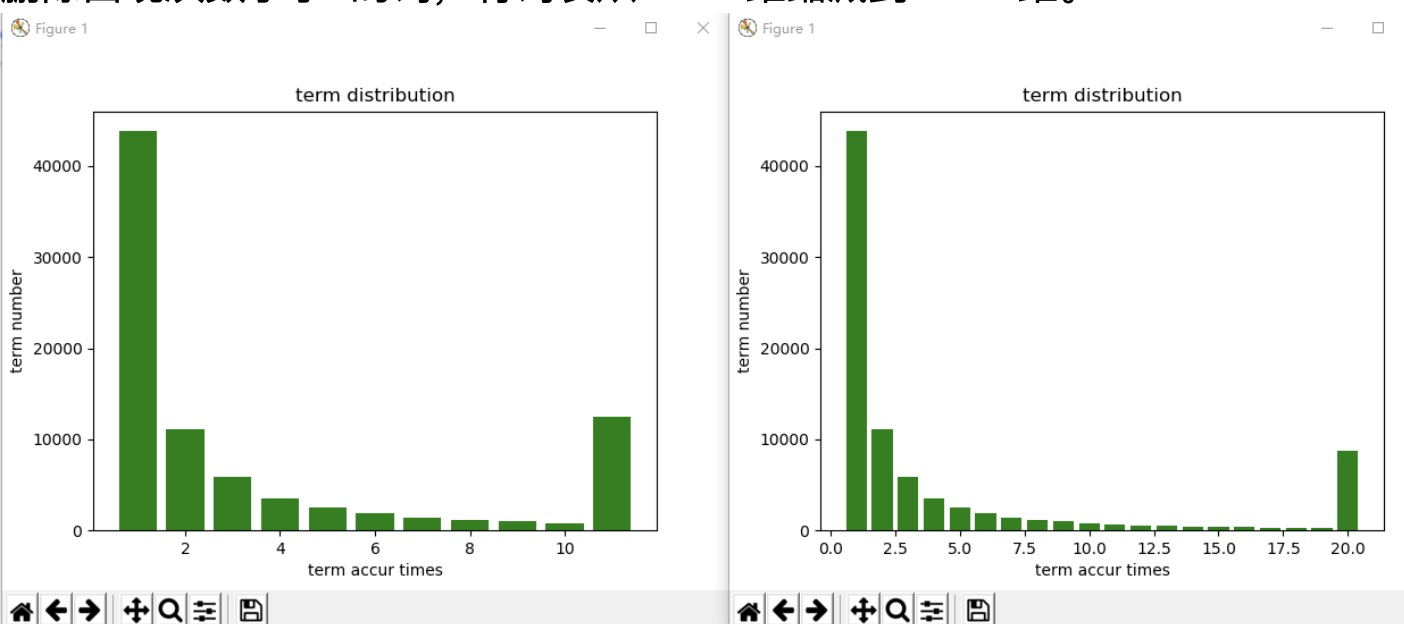
|  |  |
| --- | --- |
| 标签id | 次数 |
| 例如：“黑白”标签的id | number |

接着，过滤商品标签数量频次。筛选次数小于某个数值的项，降低部分异样样本对算法结果的影响。

程序设计代码如下：

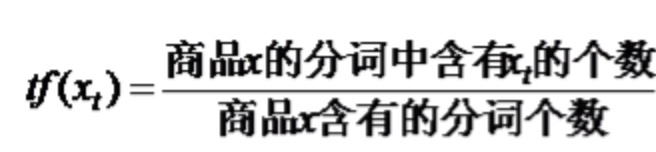


我们通过计算比较发现，在频次小于等于5次的数据量占比较大，能够提供较好的算法结果。



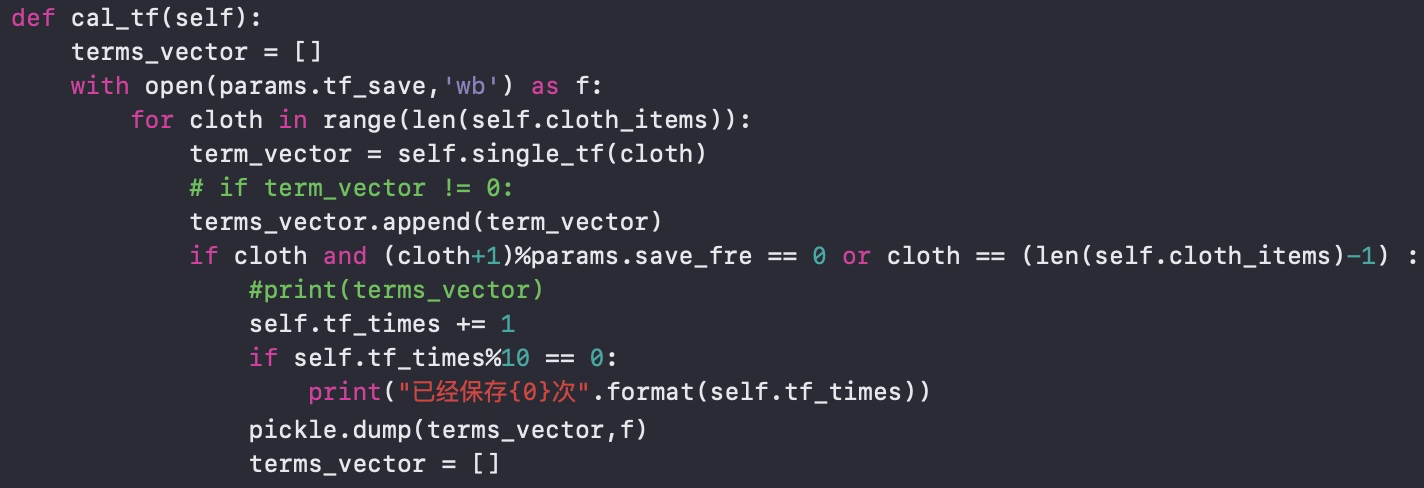
**第二步：计算TF值**

TF的计算公式为：



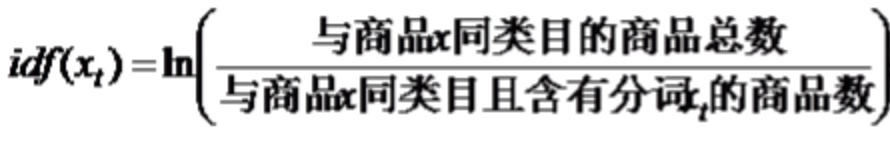
分子是该词在文件中的出现次数，而分母则是在文件中所有字词的出现次数之和。

程序设计代码如下：

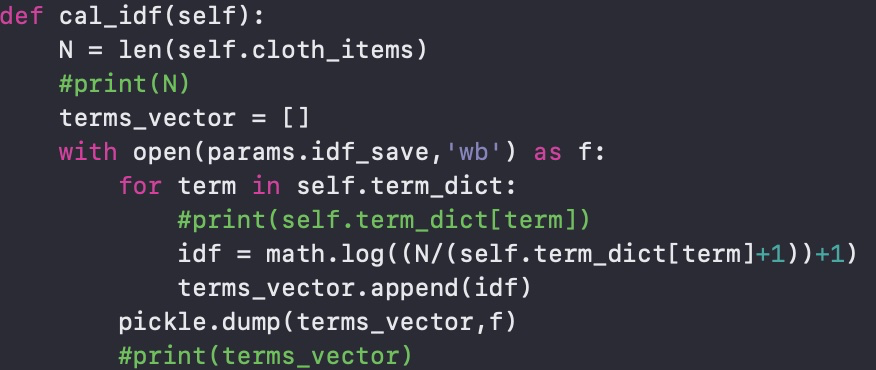


**第三步：计算IDF值**

IDF的计算公式为：



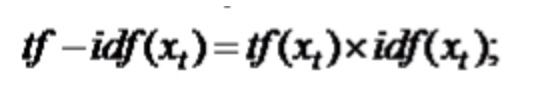
程序设计代码如下：



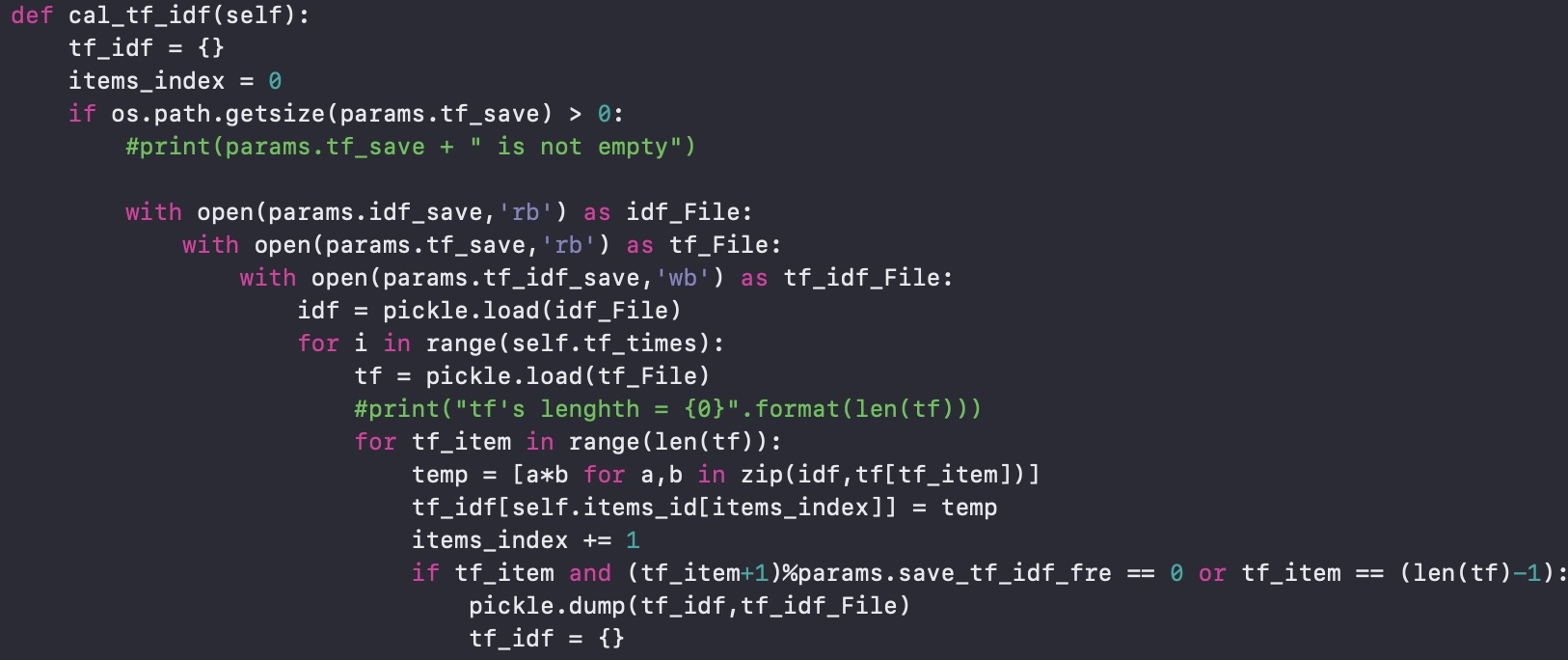
**第四步：计算TF\_IDF**

TF\_IDF在这里表示一个商品特征值的重要程度。值越大，该特征对于该商品的重要程度越大。(其中所有的分母都+1，保证分母不为0)。

TF\_IDF的计算公式如下：



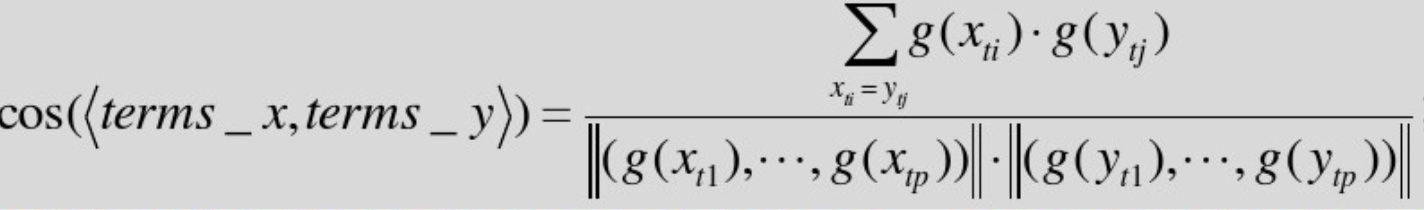
程序设计代码如下：



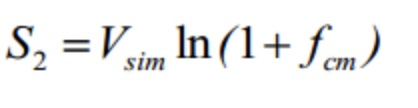
**第五步：TF\_IDF文本相似度**

利用余弦相似度算法计算搭配集。余弦相似度算法，对于两个向量来说，之间的夹角越小，这两个向量靠的越近，针对我们的额题目，就是商品的特征向量之间，如果计算出来的夹角的余弦值越接近于1，那么这两个商品之间的相似度就越高。如果这两个商品不同类，那么存在以下可能，①两个商品是同一家店铺的不同类商品，属于同一个系列或者有着同一种描述，那就有可能是搭配的。②有可能这两种商品不在同一家店中，但可能具有同种用途下互补的功能，比如沙滩裤，沙滩鞋，比基尼等商品都会有提到海边，沙滩，休闲这样的字眼，那这些商品之间就可能有搭配的潜在可能。

我们找到余弦相似度在0.4·0.6之间的商品，且S2较大的商品对，认为是搭配的。纳入搭配商品集合中。余弦相似度的计算公式：

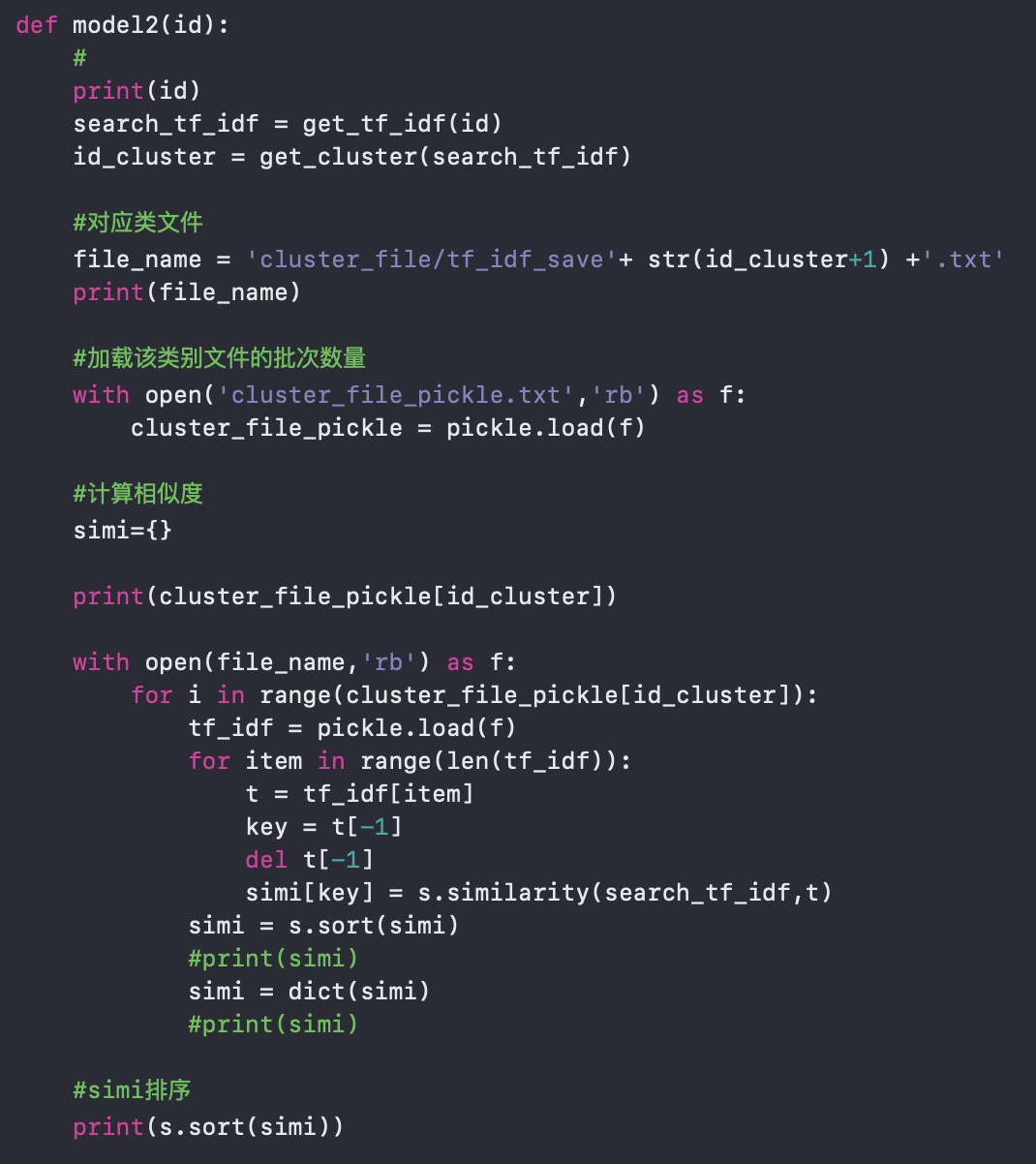


接着用找到的商品进行评价分析，获取最终的搭配集。评价函数如下：



其中Vsim是候选商品的待预测商品之间的TF\_IDF文本相似度。

程序设计代码如下：



3、模型三

该模型主要通过基于专家搭配集获取商品的搭配集合来得到商品的搭配子集，也是利用商品之间的相似度计算商品的搭配集。

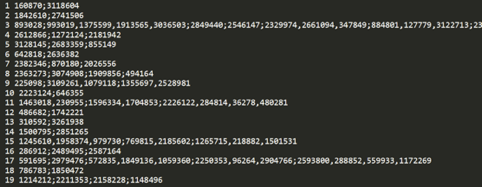
该模型中我们利用的主要思想是

（前提，待测商品为a，计算与b的搭配度，专家搭配集中b与c、d、e搭配）

* 假如a,c相似，那么a,b可能搭配，搭配度与a,c的相似度有关
* 假如a,d相似，那么a,b可能搭配，搭配度与a,d的相似度有关
* 假如a,e相似，那么a,b可能搭配，搭配度与a,e的相似度有关

最后，利用多个相似度计算搭配度。

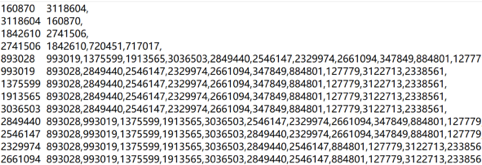
第一步：数据处理

题目提供的专家搭配集：

每一行数据表示商品id和它的搭配集商品id。

接着，对搭配集进行重新整合，这是为了后续算法实现便利。

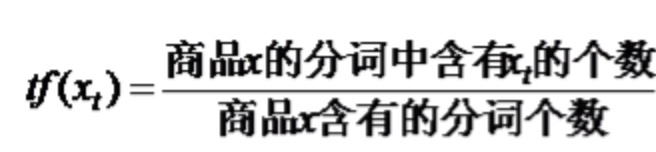
最终将数据处理为下面这种格式：

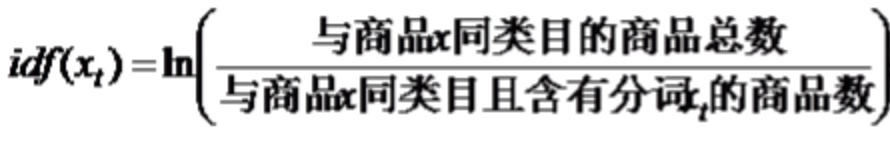


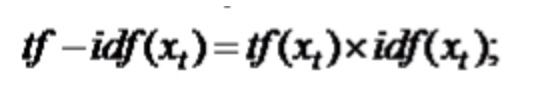
即原来160870的商品对应了搭配的商品3118604，那么商品3118604也可以搭配商品160870。

第二步：计算两个商品的相似度

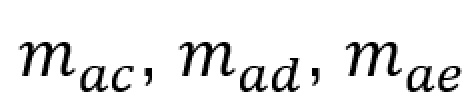
先计算TF\_IDF值:







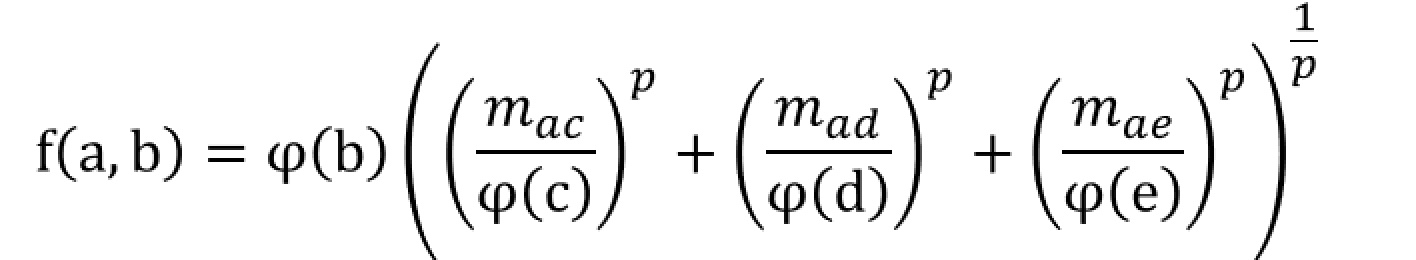
通过以上公式得到多个相似度，这里假设有3个相似度，计为：

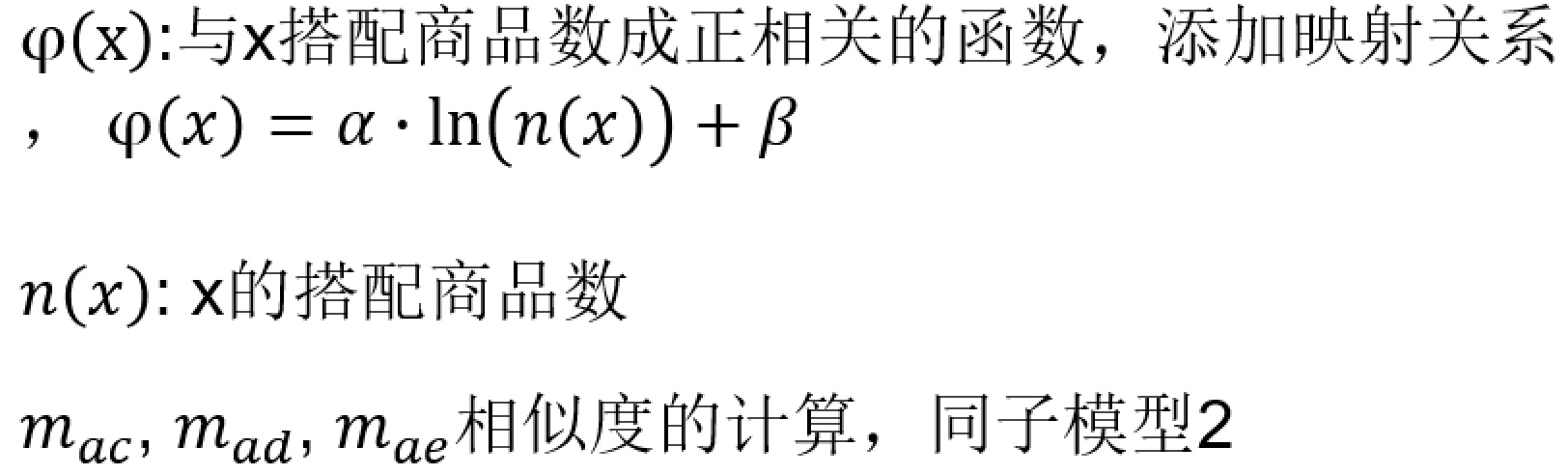


第三步：通过相似度计算搭配度

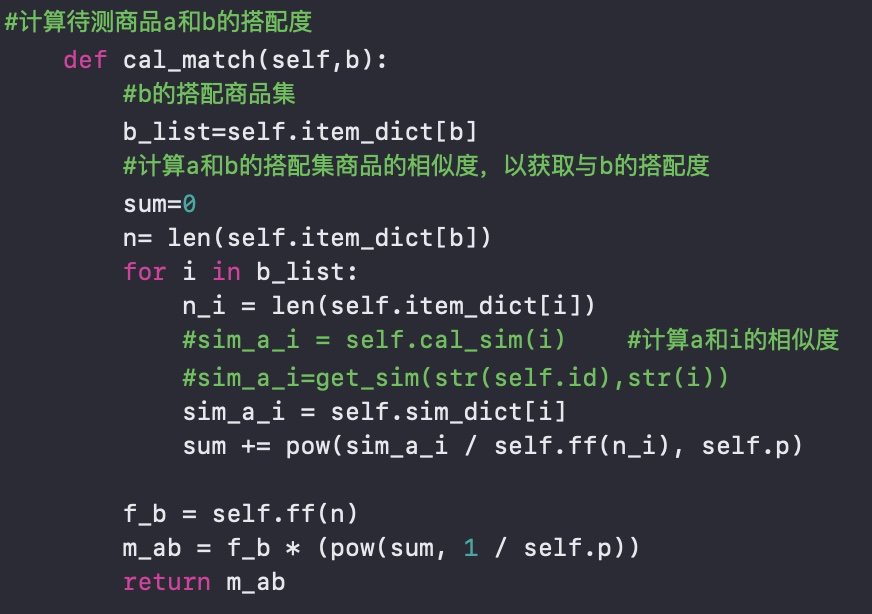
计算思路：

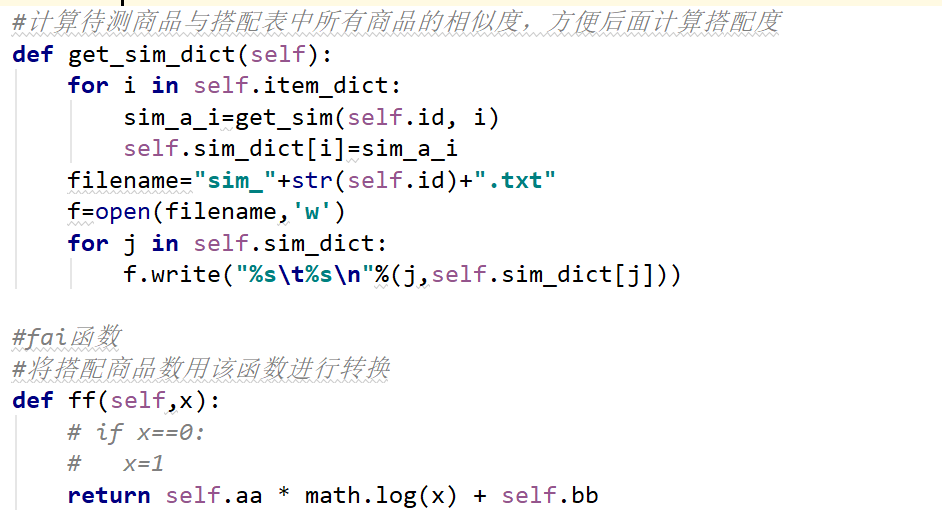
* 假如 c,d,e 是百搭的商品 ， 那么通过它们去计算 a,b 搭配度可能不太靠谱 （ 与它们的搭配商品数成负相关 ） 。
* 假如 b 是百搭的商品 ， 那么 a,b 相搭配的可能性变大 （ 与它的搭配商品数成正相关 ） 。
* 因为 c,d,e 与 a 的相似度是相互独立的 ，所以融合后的相似度应该不小于只考虑单独某一个时的相似度 ，简单的考虑就是累加 ，但是直接累加可能过大或过小 ，修正为取 p范数 。



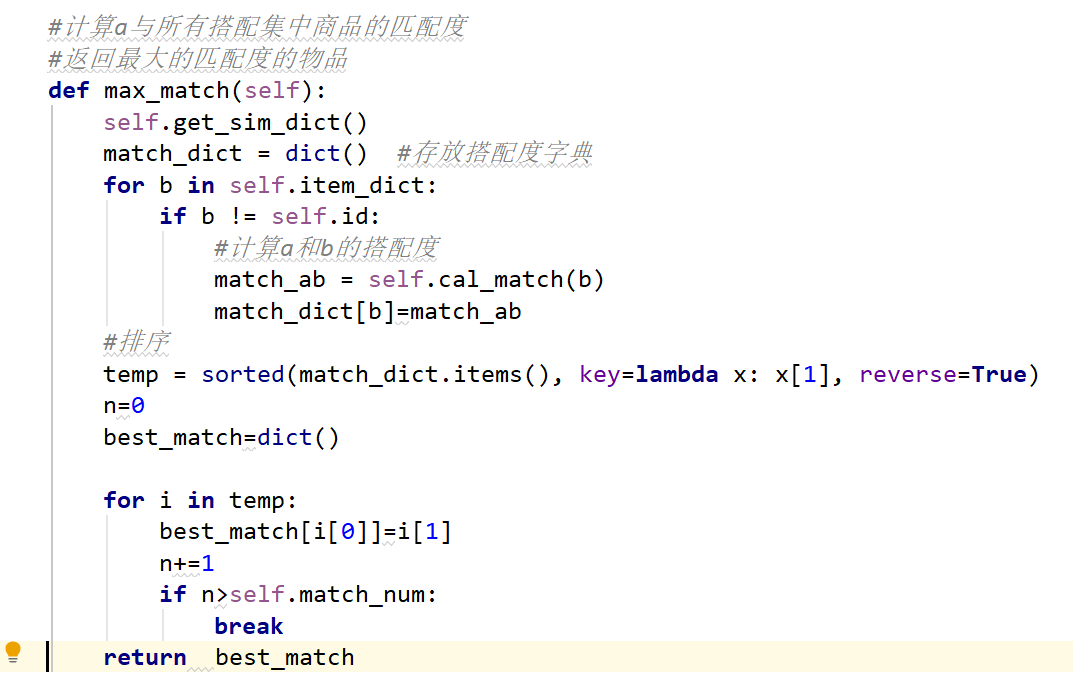


程序设计代码如下：

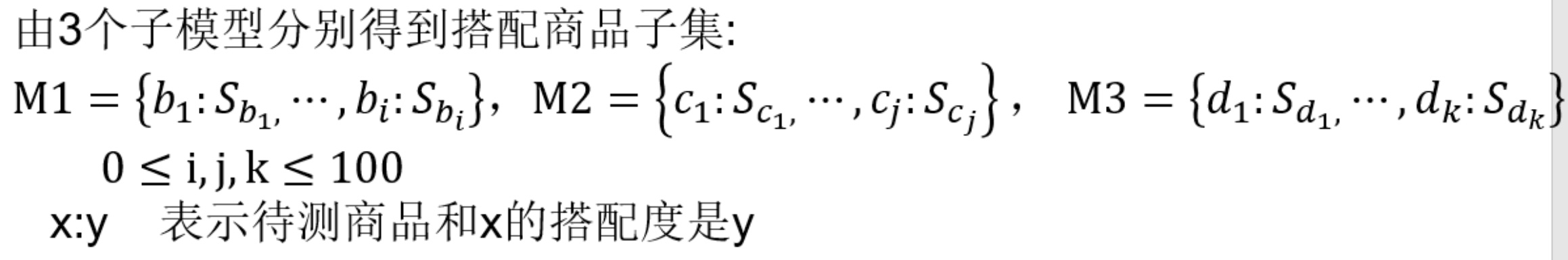


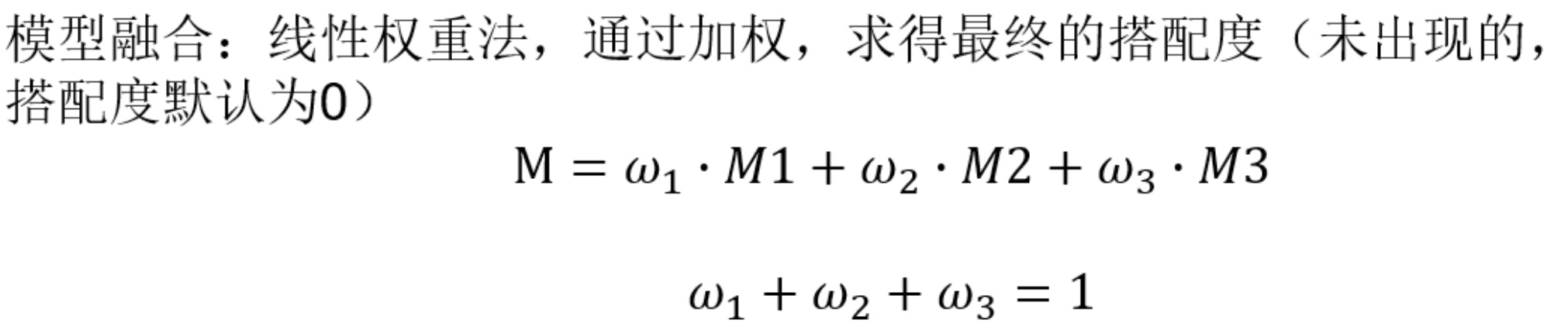


获取搭配子集：

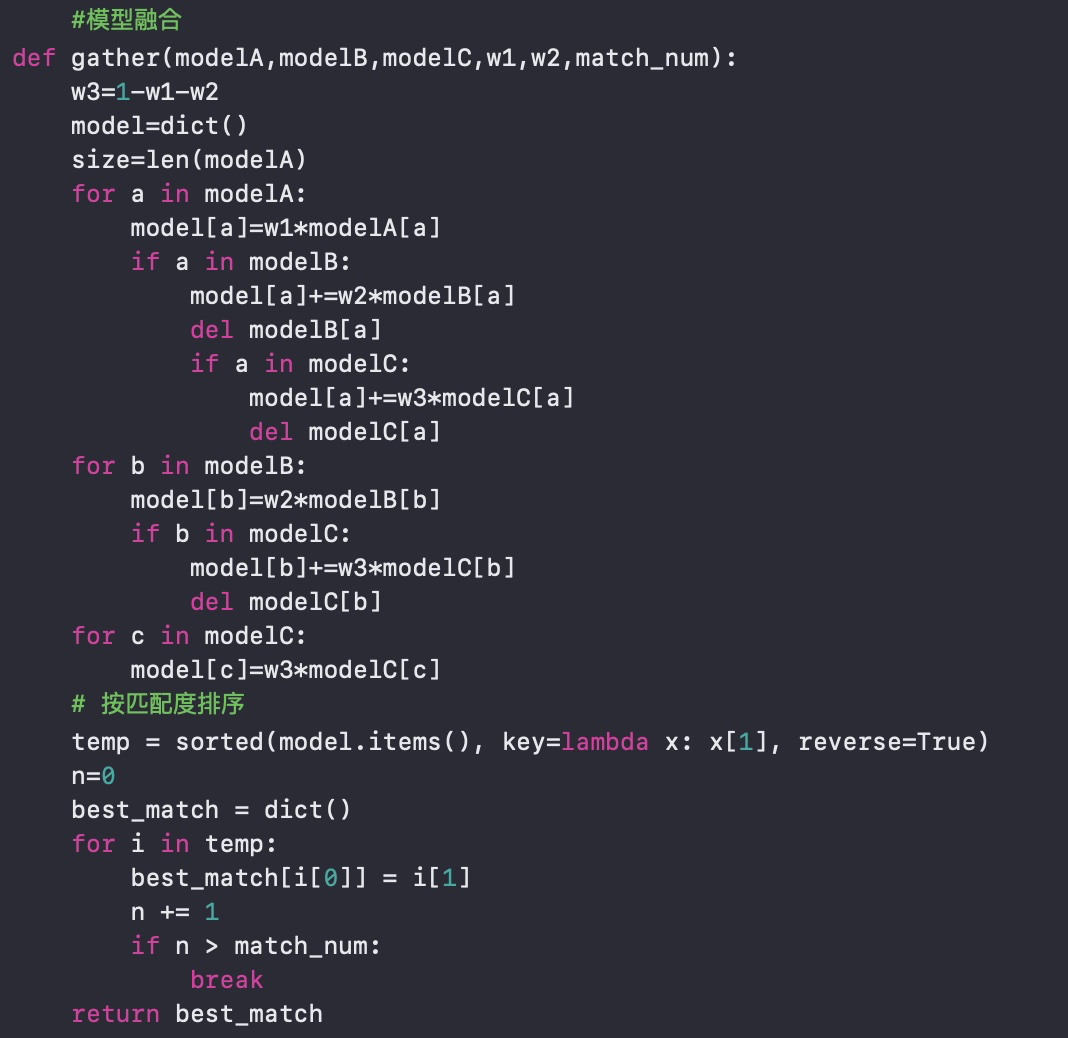


4、融合模型



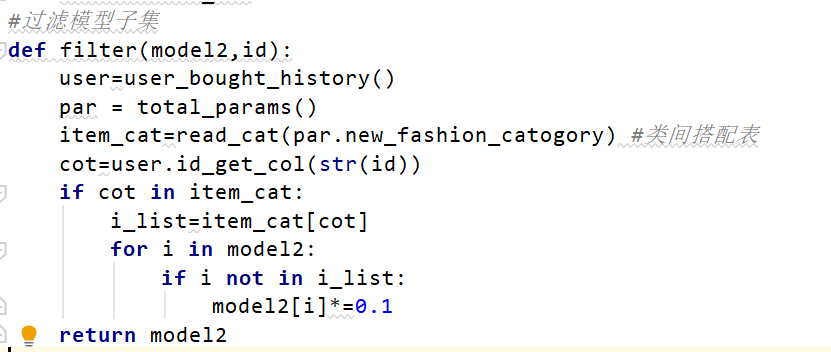


程序主要代码如下：



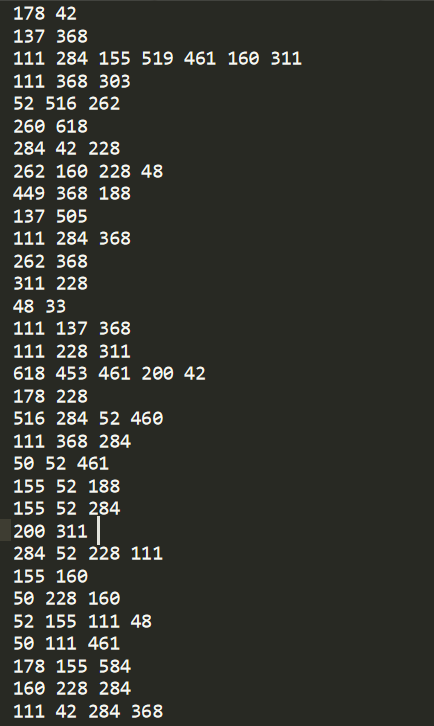
在模型融合前，需对模型二、三的搭配子集进行过滤，去除同类商品的搭配。以专家搭配表提取出的类间表为标准，如果不属于可以搭配的类，则认为是搭配度较低的类。将搭配度\*0.1，减小搭配度。

程序代码：

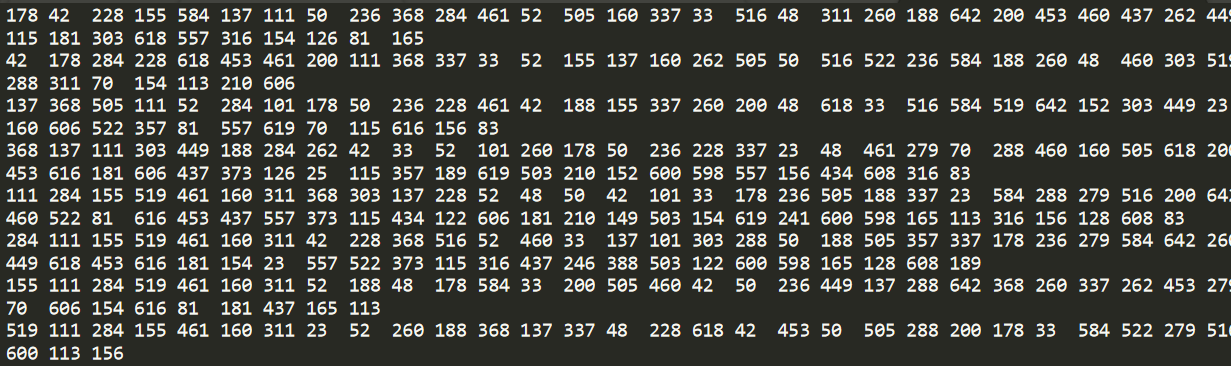


对原类间搭配表进行一些处理，将同一类可搭配的类进行整合，方便类间过滤。

原数据：

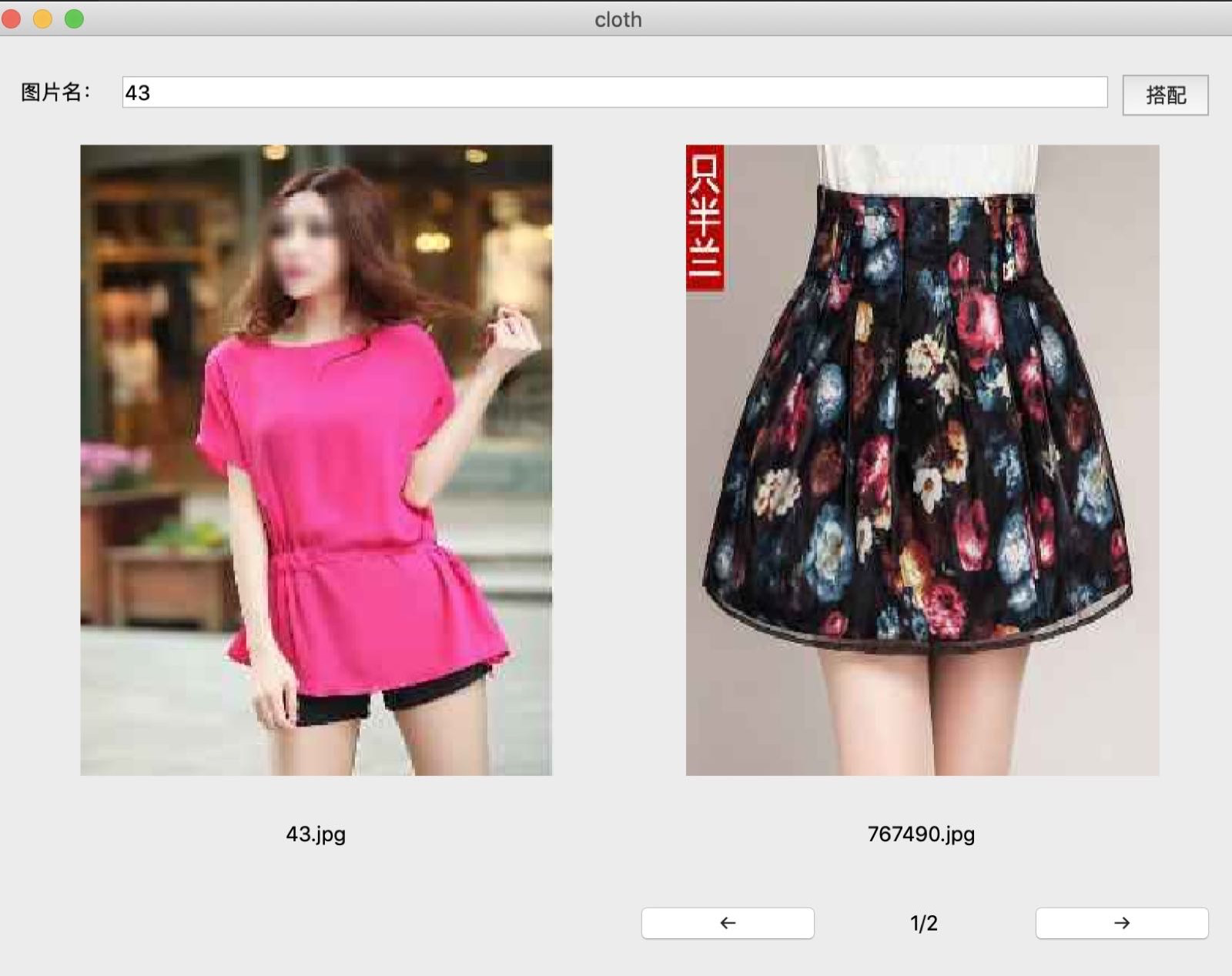


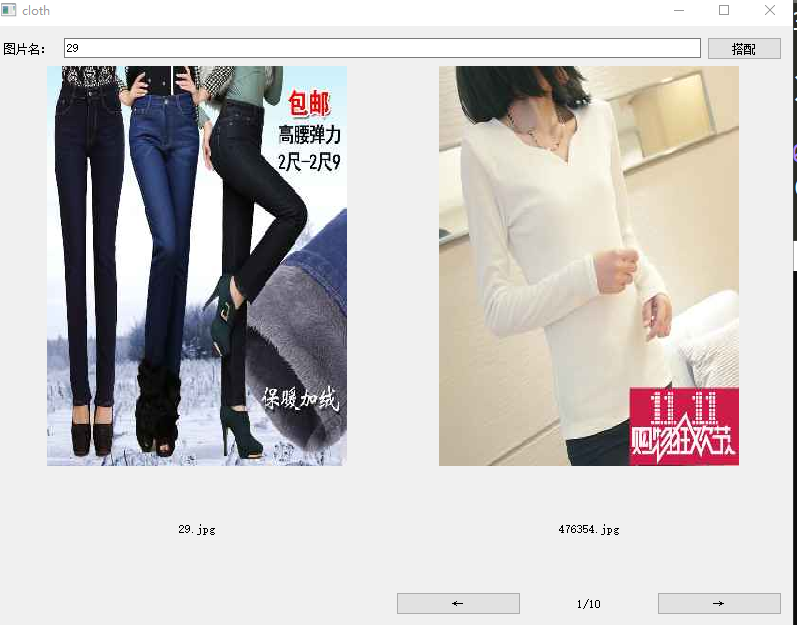
处理后数据：



5、模型界面设计

提供图片，输出该图片对应的搭配集合：





五、性能设计

1、在处理数据的过程中，通过Kmeans聚类缩小数据样本的查询效率

2、程序中使用公共变量减小非必要内存的开销

3、分离模块

模型2和模型3都使用到了TFIDF文本相似度全局搜索的算法，我们抽取相同的函数作为公共的模块，减小编译文件大小，同时提高编码效率。