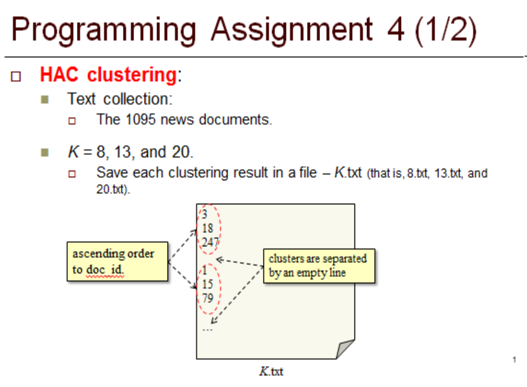
===

Name： 李桂欽 ID：R04725050 Department：資訊管理 碩一 Homework： 4

===

Programming Assignment 4:

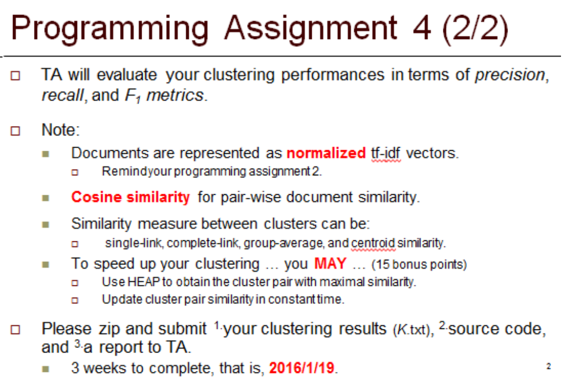


Programming Assignment 4 (1/2)

- **HAC clustering**:
  - Text collection:
    - The 1095 news documents.
  - *K* = 8, 13, and 20.
    - Save each clustering result in a file – *K*.txt (that is, 8.txt, 13.txt, and 20.txt).

ascending order to doc_id
3
18
247
1
15
79
…
clusters are separated by an empty line
K.txt

1



Programming Assignment 4 (2/2)

- TA will evaluate your clustering performances in terms of *precision*, *recall*, and F_1 *metrics*.
- Note:
  - Documents are represented as **normalized** tf-idf vectors.
    - Remind your programming assignment 2.
  - **Cosine similarity** for pair-wise document similarity.
  - Similarity measure between clusters can be:
    - single-link, complete-link, group-average, and centroid similarity.
  - To speed up your clustering … you **MAY** … (15 bonus points)
    - Use HEAP to obtain the cluster pair with maximal similarity.
    - Update cluster pair similarity in constant time.
- Please zip and submit [1] your clustering results (*K*.txt), [2] source code, and [3] a report to TA.
  - 3 weeks to complete, that is, **2016/1/19**.

2

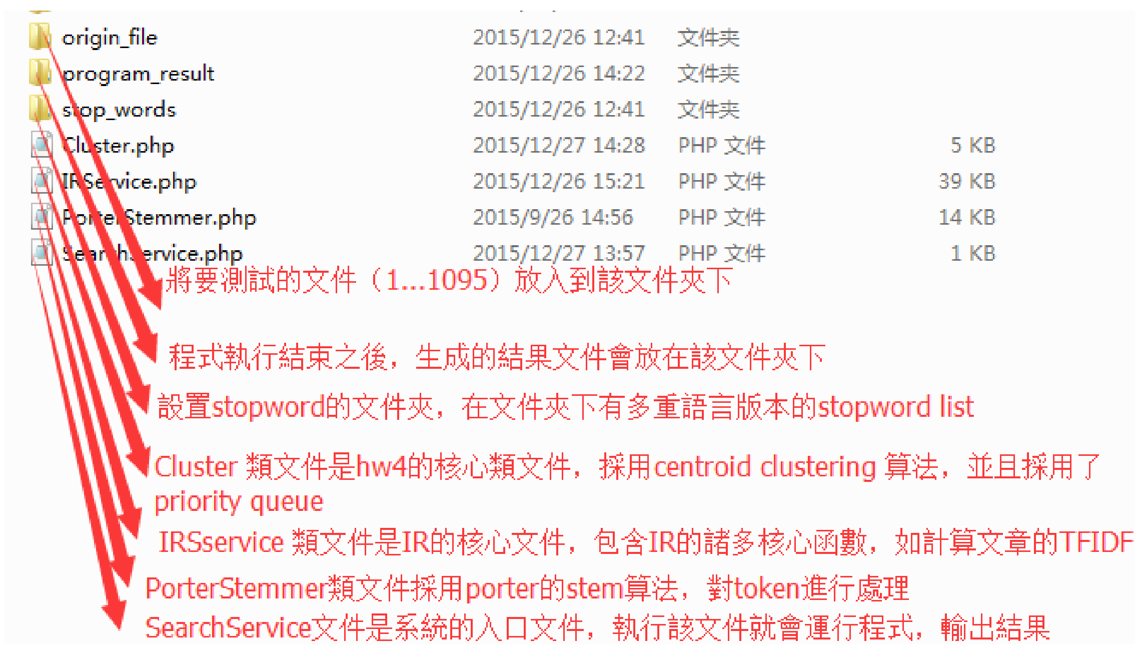
My program result:

Step1：部署 Hw4(將程式文檔放在 PHP 運行環境下)

Step2：在流覽器輸入 http://localhost/IR/SearchService.php

生成的 Result 文檔，詳見 program_result 檔夾

My program architecture:



origin_file	2015/12/26 12:41	文件夹	
program_result	2015/12/26 14:22	文件夹	
stop_words	2015/12/26 12:41	文件夹	
Cluster.php	2015/12/27 14:28	PHP 文件	5 KB
IRService.php	2015/12/26 15:21	PHP 文件	39 KB
PorterStemmer.php	2015/9/26 14:56	PHP 文件	14 KB
SearchService.php	2015/12/27 13:57	PHP 文件	1 KB

將要測試的文件（1...1095）放入到該文件夾下

程式執行結束之後，生成的結果文件會放在該文件夾下

設置stopword的文件夾，在文件夾下有多重語言版本的stopword list

Cluster 類文件是hw4的核心類文件，採用centroid clustering 算法，並且採用了priority queue

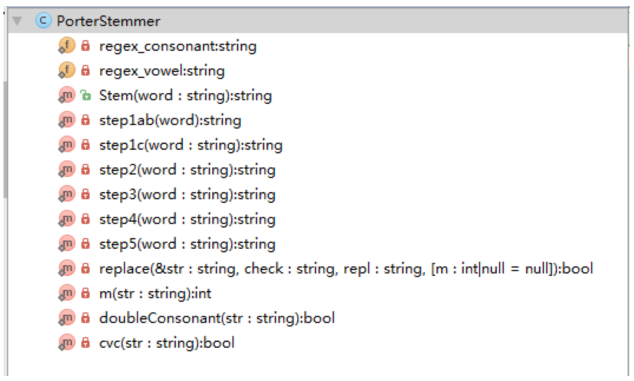
IRSservice 類文件是IR的核心文件，包含IR的諸多核心函數，如計算文章的TFIDF

PorterStemmer類文件採用porter的stem算法，對token進行處理

SearchService文件是系統的入口文件，執行該文件就會運行程式，輸出結果

My program main class:

PorterStemmer Class Structure：



```
PorterStemmer
    regex_consonant:string
    regex_vowel:string
    Stem(word : string):string
    step1ab(word):string
    step1c(word : string):string
    step2(word : string):string
    step3(word : string):string
    step4(word : string):string
    step5(word : string):string
    replace(&str : string, check : string, repl : string, [m : int|null = null]):bool
    m(str : string):int
    doubleConsonant(str : string):bool
    cvc(str : string):bool
```

演算法實現過程：

第一步，處理複數，以及 ed 和 ing 結束的單詞。

第二步，如果單詞中包含母音，並且以 y 結尾，將 y 改為 i。

第三步，將雙尾碼的單詞映射為單尾碼。

第四步，處理-ic-，-full，-ness 等等尾碼。

第五步，在<c>vcvc<v>情形下，去除-ant，-ence 等尾碼。

第六步，也就是最後一步，在 m()>1 的情況下，移除末尾的“e”。

演算法使用說明：

傳入的單詞必須是小寫

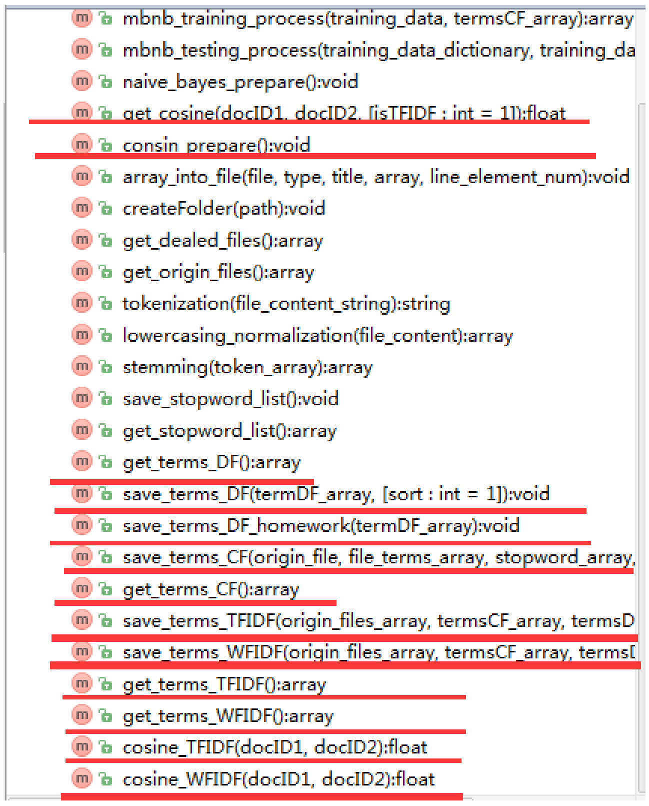
參考學習網站：

http://tartarus.org/~martin/PorterStemmer/

http://snowball.tartarus.org/algorithms/english/stemmer.html

http://blog.csdn.net/noobzc1/article/details/8902881

IRService Class Structure：

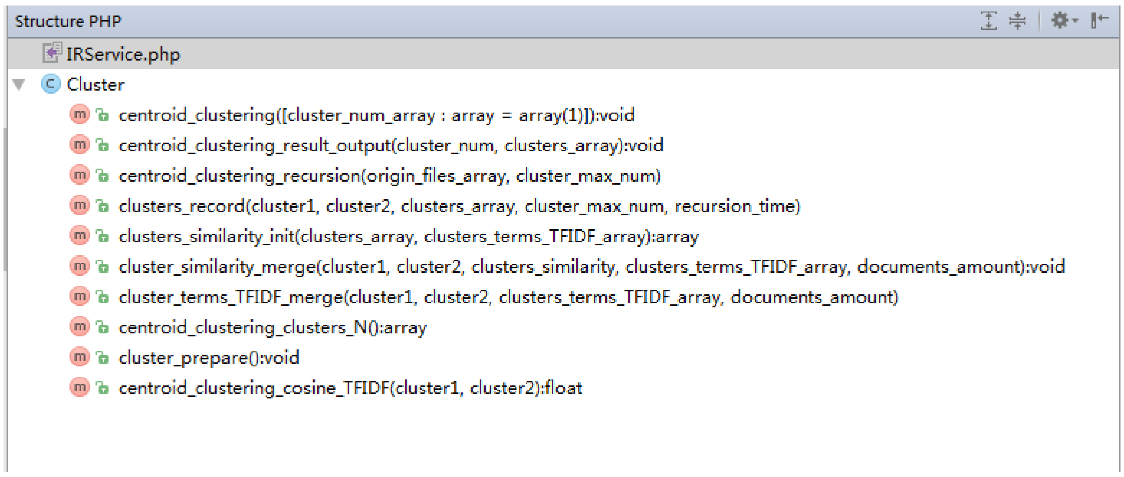


```
mbnb_training_process(training_data, termsCF_array):array
mbnb_testing_process(training_data_dictionary, training_da
naive_bayes_prepare():void
get_cosine(docID1, docID2, [isTFIDF : int = 1]):float
consin_prepare():void
array_into_file(file, type, title, array, line_element_num):void
createFolder(path):void
get_dealed_files():array
get_origin_files():array
tokenization(file_content_string):string
lowercasing_normalization(file_content):array
stemming(token_array):array
save_stopword_list():void
get_stopword_list():array
get_terms_DF():array
save_terms_DF(termDF_array, [sort : int = 1]):void
save_terms_DF_homework(termDF_array):void
save_terms_CF(origin_file, file_terms_array, stopword_array,
get_terms_CF():array
save_terms_TFIDF(origin_files_array, termsCF_array, termsD
save_terms_WFIDF(origin_files_array, termsCF_array, termsI
get_terms_TFIDF():array
get_terms_WFIDF():array
cosine_TFIDF(docID1, docID2):float
cosine_WFIDF(docID1, docID2):float
```

類的主要函數：

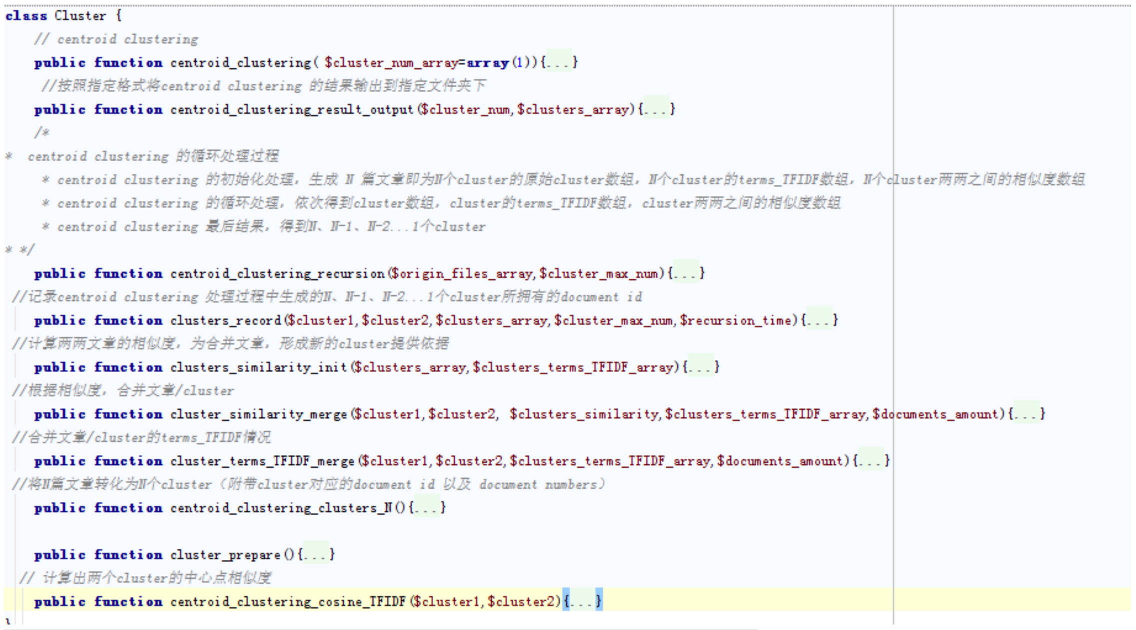
```
/*
 *  计算指定的两篇文章的相似度
 * */
public function get_cosine()
/*
 *  计算文章相似度的前期工作
 * */
public function consin_prepare()
/*
 *  计算出每个文件每个特异单词的 TF-IDF，并且保存在 TXT 文件中
 * */
 public function save_terms_TFIDF()
```

Cluster Class Structure：



```
Structure PHP
  IRService.php
▼ Cluster
    centroid_clustering([cluster_num_array : array = array(1)]):void
    centroid_clustering_result_output(cluster_num, clusters_array):void
    centroid_clustering_recursion(origin_files_array, cluster_max_num)
    clusters_record(cluster1, cluster2, clusters_array, cluster_max_num, recursion_time)
    clusters_similarity_init(clusters_array, clusters_terms_TFIDF_array):array
    cluster_similarity_merge(cluster1, cluster2, clusters_similarity, clusters_terms_TFIDF_array, documents_amount):void
    cluster_terms_TFIDF_merge(cluster1, cluster2, clusters_terms_TFIDF_array, documents_amount)
    centroid_clustering_clusters_N():array
    cluster_prepare():void
    centroid_clustering_cosine_TFIDF(cluster1, cluster2):float
```

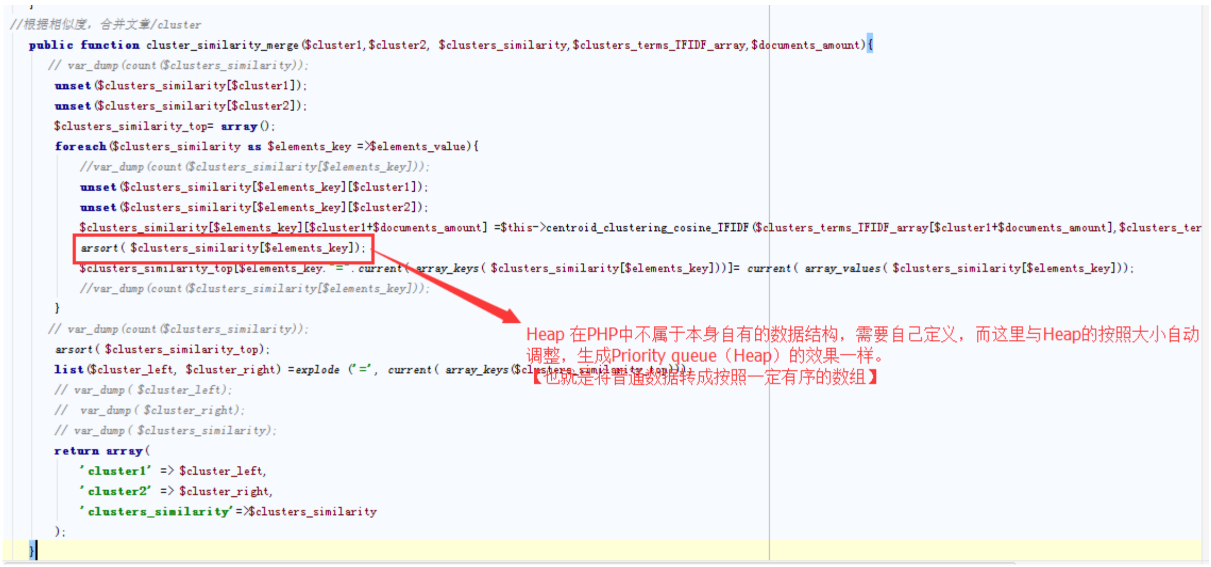
類的主要函數：



```
class Cluster {
    // centroid clustering
    public function centroid_clustering( $cluster_num_array=array(1)){...}
     //按照指定格式将centroid clustering 的结果输出到指定文件夹下
    public function centroid_clustering_result_output($cluster_num,$clusters_array){...}
    /*
*  centroid clustering 的循环处理过程
     * centroid clustering 的初始化处理，生成 N 篇文章即为N个cluster的原始cluster数组，N个cluster的terms_TFIDF数组，N个cluster两两之间的相似度数组
     * centroid clustering 的循环处理，依次得到cluster数组，cluster的terms_TFIDF数组，cluster两两之间的相似度数组
     * centroid clustering 最后结果，得到N、N-1、N-2...1个cluster
* */
    public function centroid_clustering_recursion($origin_files_array,$cluster_max_num){...}
 //记录centroid clustering 处理过程中生成的N、N-1、N-2...1个cluster所拥有的document id
    public function clusters_record($cluster1,$cluster2,$clusters_array,$cluster_max_num,$recursion_time){...}
 //计算两两文章的相似度，为合并文章，形成新的cluster提供依据
    public function clusters_similarity_init($clusters_array,$clusters_terms_TFIDF_array){...}
 //根据相似度，合并文章/cluster
    public function cluster_similarity_merge($cluster1,$cluster2, $clusters_similarity,$clusters_terms_TFIDF_array,$documents_amount){...}
 //合并文章/cluster的terms_TFIDF情况
    public function cluster_terms_TFIDF_merge($cluster1,$cluster2,$clusters_terms_TFIDF_array,$documents_amount){...}
 //将N篇文章转化为N个cluster（附带cluster对应的document id 以及 document numbers）
    public function centroid_clustering_clusters_N(){...}

    public function cluster_prepare(){...}
  // 计算出两个cluster的中心点相似度
    public function centroid_clustering_cosine_TFIDF($cluster1,$cluster2){...}
}
```

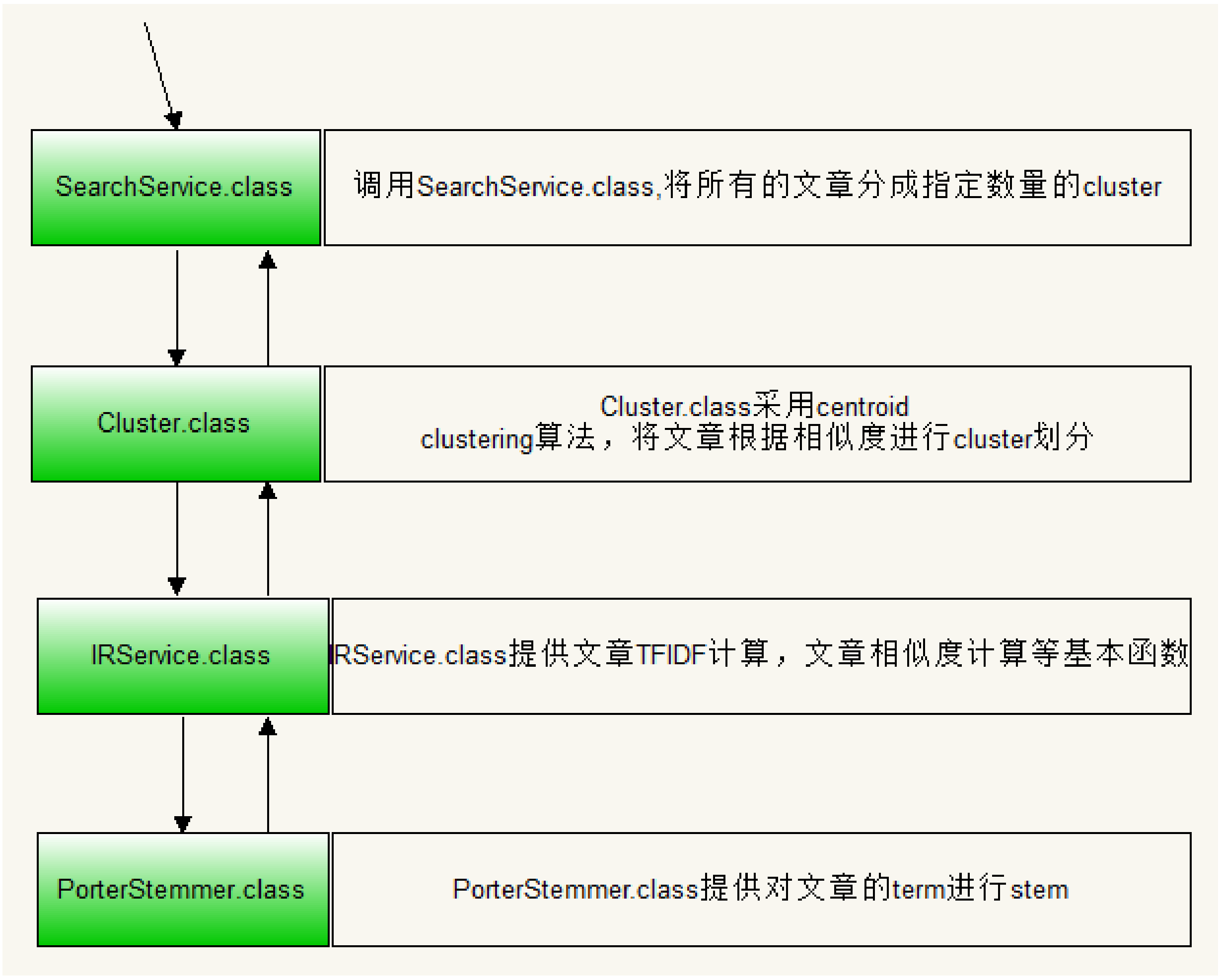
Use HEAP to obtain the cluster pair with maximal similarity. 說明



```
}
//根据相似度，合并文章/cluster
 public function cluster_similarity_merge($cluster1,$cluster2, $clusters_similarity,$clusters_terms_IFIDF_array,$documents_amount){
   // var_dump(count($clusters_similarity));
    unset($clusters_similarity[$cluster1]);
    unset($clusters_similarity[$cluster2]);
    $clusters_similarity_top= array();
    foreach($clusters_similarity as $elements_key =>$elements_value){
        //var_dump(count($clusters_similarity[$elements_key]));
        unset($clusters_similarity[$elements_key][$cluster1]);
        unset($clusters_similarity[$elements_key][$cluster2]);
        $clusters_similarity[$elements_key][$cluster1+$documents_amount] =$this->centroid_clustering_cosine_IFIDF($clusters_terms_IFIDF_array[$cluster1+$documents_amount],$clusters_ter
        arsort( $clusters_similarity[$elements_key]);
        $clusters_similarity_top[$elements_key.'='.current( array_keys( $clusters_similarity[$elements_key]))]= current( array_values( $clusters_similarity[$elements_key]));
        //var_dump(count($clusters_similarity[$elements_key]));
    }
   // var_dump(count($clusters_similarity));
    arsort( $clusters_similarity_top);
    list($cluster_left, $cluster_right) =explode ('=', current( array_keys($clusters_similarity_top)));
    // var_dump( $cluster_left);
    //  var_dump( $cluster_right);
    // var_dump( $clusters_similarity);
    return array(
        'cluster1' => $cluster_left,
        'cluster2' => $cluster_right,
        'clusters_similarity'=>$clusters_similarity
    );
 }
```

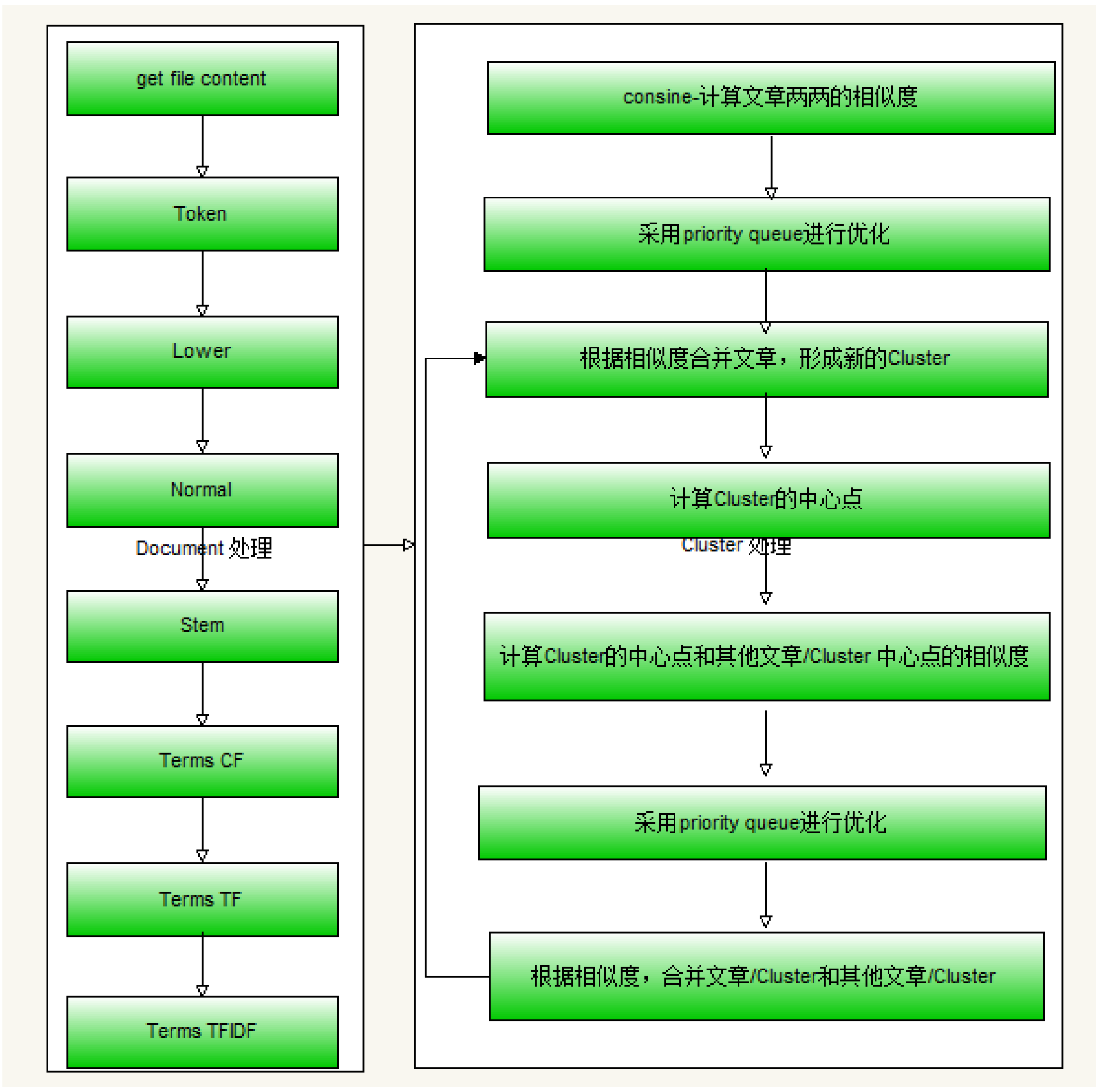
Heap 在PHP中不属于本身自有的数据结构，需要自己定义，而这里与Heap的按照大小自动调整，生成Priority queue（Heap）的效果一样。
【也就是将普通数据转成按照一定有序的数组】

多個 Class 的工作協調過程:



SearchService.class
调用SearchService.class,将所有的文章分成指定数量的cluster
Cluster.class
Cluster.class采用centroid clustering算法，将文章根据相似度进行cluster划分
IRService.class
IRService.class提供文章TFIDF计算，文章相似度计算等基本函数
PorterStemmer.class
PorterStemmer.class提供对文章的term进行stem

Clustering work flow


get file content
Token
Lower
Normal
Document 处理
Stem
Terms CF
Terms TF
Terms TFIDF
consine-计算文章两两的相似度
采用priority queue进行优化
根据相似度合并文章，形成新的Cluster
计算Cluster的中心点
Cluster 处理
计算Cluster的中心点和其他文章/Cluster 中心点的相似度
采用priority queue进行优化
根据相似度，合并文章/Cluster和其他文章/Cluster