A greedy set-covering approach for the recomputation of the cluster centres in the FPAC algorithm

Ivan Feliciano[1] and Edgar Hernandez-Gonzalez[2]

[1] National Institute of Astrophysics, Optics and Electronics
ivan.felavel@gmail.com
[2] National Institute of Astrophysics, Optics and Electronics
edgarmoy.28@gmail.com

Abstract. En este trabajo presentamos una modificación al algoritmo K-means usando una heurística que permite hacer el recalculo de centroides de una manera diferente al k-means tradicional. El procedimiento se basa en… primero, después, por ultimo.

Keywords: Clustering · Information Retrieval · FPAC

1 Introduction

The amount of information, specifically text, that is generated everyday is increasing rapidly and dramatically. This necessitates application of effective and efficient content management techniques to fulfill the task of finding groups of similar documents in a collection of documents. This task is also known as clustering. These document clusters can then be useful for a variety of applications, such as document alignment, information retrieval (IR), text classification, etc [1].

We can define the goal in clustering as follows. Given a set of documents, $D = \{d_1, \dots, d_N\}$, a desired number of clusters, K, and an objective function that evaluates the quality of a clustering. we want to compute an assignment $\gamma : D \to \{1, \dots, K\}$ that minimizes (or, in other cases, maximizes) the objective function. In most cases, we also demand that γ is surjective, that is, that none of the K clusters is empty. The objective function is often defined in terms of similarity or distance between documents.

K-means is perhaps the most widely used clustering algorithm because of its simplicity and efficiency. The objective in K-means clustering is to minimize the average distance between documents and their centroids or, equivalently, to maximize the similarity between documents and their centroids.

K-means is an EM-based algorithm, in which starting with a set of randomly chosen initial centres, each input point is repeatedly assigned to its nearest cluster centre, the E-step. The cluster centres are then recomputed by making use of the current assignments, the M-step.

The K-means clustering algorithm does not scale well to datasets that are considerably large in size and large in dimensionality, e.g. large document collections where each document is a sparse vector of large dimension (vocabulary size of the collection). In the case of clustering vectors of large dimensionality, the computational overhead of the K-means algorithm arises from both the E and the M steps, that is when: (a) assigning vectors to one of the centres (interchangeably referred to as 'cluster centres'); and (b) recomputing the centres. Concretely speaking, the computation required to assign each vector in the collection to one of the centroids is expensive because it involves computing the similarity of this data point with every cluster cen- tre, which is an expensive operation if the collection is very large.

Since the introduction to the K-means algorithm, research has progressed towards making it more efficient for cases where the number of data points or the dimensionality of the data is large, e.g. the case when the dataset is comprised of a collection of documents.

The Fast Partitional Clustering Algorithm (FPAC) [1] involves the nearest neighbour based heuristic to scale up K-means clustering for large document collections. The main contribution of the FPAC on the K-means algorithm is the use of an operation, namely $TOP(x)$ which gives a list of the top mos similar vectors with respect to the current vector. This operation is used in the three fundamental steps of the K-mean, the selection of the initial cluster centres, the assignment of a cluster to a non-centroid vector d, and the recomputation of the centre clusters.

Our work is based on the FPAC algorithm and we have focused in the recomputation of the centroids step. In the FPAC algorithm they select as the cluster centroid the vector with the highest number of distinct terms. Despite the efficiency of just take the vector with the maximum number of unique terms, what happen if the set of terms in the vector is disjoint from the other set of terms in the cluster vocabulary. That case inspired us to develop a solution based on the set cover problem.

Our contribution is the extension of the method for the recomputation of the cluster centroids. Instead of using just one centroid that tries to cover most of the vocabulary within a cluster, we proposed the selection of M centroids such that they cover the maximum number of terms in a cluster vocabulary.

2 Related Work

The Fast PArtitional Clustering (FPAC) algorithm makes use of the nearest neighbour based heuristic. This heuristic consists of using a set of the most similar vector from a each cluster vector x. The operation of getting the set of the top nearest vectors is called $TOP(x)$ and is efficiently computated by using a inverted index data structure.

In a very general way we can hightlight three fundamental steps in the FPAC algorithm: the selection of the initial centroids, the assinment of a cluster to a non-centroid vector and the recomputation of the cluster centres.

To ensure that the initially selected cluster centres are dissimilar to each other, they select the first cluster centre, C_1, randomly from the whole collection. The next cluster centroid, C_2, is then selected such that it has a low similarity with the already chosen centroid. More precisely, C_2 it is a randomly chosen vector from the collection that is composed by the vectors that do not occur in the retreived list of C_1, $TOP(C_1)$. For selecting C_3, a vector is chosen at random that does not occur in the union of the retrieved lists for C_1 and C_2, and the process continues till K cluster centres are obtained.

To assign a cluster to a non-centroid vector d, they use the K centroids as queries. If the cluster centres themselves are dissimilar to each other, it is expected that the ranked lists retrieved for each centre will have a small intersection. If a vector d it is included it is retrieved in the top set of only a single ranked list, $TOP(C_k)$, it is included in the set of vector assigned to the cluster corresponding to C_k. If a vector d is retrieved in multiple ranked lists, it is assigned to the cluster for which the normalized similarity score is maximum. The vectors which are not retrieved in the union set of the ranked lists for each centre are assigned randomly to any one of the K cluster centroids.

For computing the new centroid for a cluster they treat the vector which has the maximum number of unique terms within a cluster (i.e. the one with maximum length) as the updated cluster centre.

The results they obtained from the experiments of the FPAC algorithm compared to the K-means and SK-means algorithms, show an increase in the execution speed attributed to the IR-based approach and the two additional heuristics related to initial centroid selection and centroid recomputation.

The FPAC similarity-based initial centroids selection produces the best results in terms of clustering effectiveness. The maximum term coverage based heuristics yields better in comparasion to SK-means. However, the results with the maximum term coverage heuristics is worse than a version of the true centroids calculation of FPAC.

We can think of an example that shows why the coverage of terms with the centroid of maximum length is not a good enough solution for the recomputation of the cluster centres. We can see that the vector with the maximum number of terms could be one such its intersection with the others vectors would be the empty set or even worse, what if the terms composing the cluster centre are not useful even if it has a lot of terms. We concentrate our efforts for solving the first case by using a set cover approach where we select M centroids, for each cluster, that cover most of the terms in a cluster vocabulary.

3 Proposed Solution

Because we are using M centroids for each cluster instead of one, we adapt the cluster centres initialization and the computation of the most similar cluster to a non-centroid vector. In the next subsections we first show how we use the similarity-based initial centroid selection for the $M \times K$ cluster centres, then we explain how do we get the closest cluster of a non-centroid vector by iterate over each of the M cluster centres of each cluster. Finally we explain our principal contribution to the FPAC algorithm trying to generalize the heuristc of selecting the document with the highest number of distinc terms.

3.1 Selecting initial cluster centres

We use the same idea of selecting a random document, grow a region around it and choose as the next candidate centroid a document that does not belong to this region. We took advantage of the computation of the region related to a documet for the assigment of the M cluster centres in a cluster.

First we select a random vector C_{11}, as the centroid of the first cluster, then we get its $M-1$ most similar vectors and assign each of this vectors as a centroid of the first cluster, so we get the centres $C_{11}, C_{12}, \ldots, C_{1M}$. For the second cluster centroids we get one unrelated vector from the first cluster centres and repeat the computation of the $M-1$ most similars vectors and they are defined as the cluster centres for this cluster, so we get $C_{21}, \ldots, C_{2M}$. For the next cluster, a vector is chosen such that does not related with the centroids of the first two clusters, and define the $M-1$ remaing cluster centres of this cluster from the top list of that vector. This process continues until we have defined the M cluster centres for K clusters.

3.2 Clustering non-centroid vectors

Compared to the FPAC algorithm, instead of only iterate through K clusters, we should also go through M vectors for each cluster. We assign a non-centroid vector to the cluster for which the sum of normalized similarity score of the M centroids of each cluster is maximum. For doing that we obtain the top list of vectors of each centroid to check if it a vector is similar to one of them. For each cluster we compute a local normalized similarity score and assign the vector to the cluster with the highest value.

If a non-centroid vector does not appear in any one of the top lists we randomly assign a cluster to it.

3.3 Recomputation of the cluster centres

For computing the new set of M centroids for each cluster we try a solution similar to the greedy approximation algorithm for the set covering problem. An instance $(X, \mathcal{F})$ of the set-covering problem consists of a finite set X and a family $\mathcal{F}$ of subsets of X, such that every element of X belongs to at least one subset in $\mathcal{F}$

$$X = \bigcup_{S \in \mathcal{F}} S.$$

We say that a subset $S \in \mathcal{F}$ covers its elements. The problem is to find a minimum-size subset $\mathcal{C} \subseteq \mathcal{F}$ whose members cover all of X

$$X = \bigcup_{S \in \mathcal{C}} S.$$

Using this approach we can define as V_k the set of terms in the cluster k. This is analogous to the set X. The set of vectors within a cluster k are the family $\mathcal{F}_k$. We want to find a subset $\mathcal{C}_k \subseteq \mathcal{F}$, such that $|\mathcal{C}_k| = M$ and whose members cover all or at least most of V_k.

We use a greedy method for solving this problem. It works iterating through each cluster and over each of the vectors associated with it. We pick as one of the new centroids of the cluster, the vector $C_{ki} \in \mathcal{F}_k$ that covers the greatest number of remaining elements that are uncovered in the set V_k. This process is repeated until the set V_k is completely covered or have been selected M cluster centres. This method is explained in detail in the Algorithm 1.

Algorithm 1 Greedy approximation algorithm for the recomputation of cluster centres.

```
for $k = 1$ to $K$ do
  $i \leftarrow 1$
  while $V_k \neq \emptyset$ and $i \leq M$ do
    $C_{ki} \leftarrow d \in \mathcal{F}_k$ that maximizes $|d \cap V_k|$
    $i \leftarrow i + 1$
    $V_k \leftarrow V_k - C_{ki}$
    $\mathcal{C}_k \leftarrow \mathcal{C}_k \cup \{C_{ki}\}$
  end while
end for
```

4 Experiments

4.1 Dataset

We could not replicate the experiments on the same collection of documents proposed by Ganguly[1]. However we choose two datasets: the 20 Newsgroups and the sentiment140 dataset. The 20 Newsgroup consists of 20000 messages taken from 20 newsgroups. The sentiment140 collection it is composed by 1.6 million tweets and contains a list of words and their associations with positive and negative sentiment.

Table 1. DataSets (ACOMODAR).

Name DataSet	Data	Labels
Sentiment140	100,000	2
20 Newsgroups	11,293	20

4.2 Clustering evaluation

This section introduces about external criteria of clustering quality. Purity is a simple and transparent evaluation measure. Normalized mutual information can be information-theoretically interpreted. The Rand index penalizes both false-positive and false-negative decisions during clustering. To compute purity, each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by N. Formally:

$$purity(\Omega, C) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

Bad clusterings have purity values close to 0, a perfect clustering has a purity of 1. A measure that allows us to make this tradeoff is normalized mutual information or NMI:

$$NMI(\Omega, C) = \frac{I(\Omega; C)}{[H(\Omega) + H(C)]/2}.$$

I is mutual information

$$\begin{aligned} I(\Omega; C) &= \sum_k \sum_j P(\omega_k \cap c_j) \log \frac{P(\omega_k \cap c_j)}{P(\omega_k) P(c_j)} \\ &= \sum_k \sum_j P(\omega_k \cap c_j) \log \frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|} \end{aligned}$$

H is entropy

$$\begin{aligned} H(\Omega) &= -\sum_k P(\omega_k) \log P(\omega_k) \\ &= -\sum_k \frac{|\omega_k|}{N} \log \frac{|\omega_k|}{N} \end{aligned}$$

We want to assign two documents to the same cluster if and only if they are similar. A true-positive (TP) decision assigns two similar documents to the same cluster, a true-negative (TN) decision assigns two dissimilar documents to different clusters. There are two types of errors we can commit. A false-positive (FP) decision assigns two dissimilar documents to the same cluster. A false-negative (FN) decision assigns two similar documents to different clusters. The Rand index (RI) measures the RI percentage of decisions that are correct. That is, it is simply accuracy

$$RI = \frac{TP + TN}{TP + TN + FP + FN}$$

4.3 Compared approaches

We use four baselines clustering methods. We employ the next algorithms: K-means, SK-means and two versions of the FPAC, a version with the recomputation of the centre clusters using the classic approach from the K-means and the other using the heuristics-based approach of maximum term coverage. We compare the previous procedures with our FPAC version that uses M centroids.

4.4 Parameters

5 Results

- Por cada dataset una gráfica del NMI, Purity y RI
-

6 Conclusions and future work

References

1. Ganguly, D.: A fast partitional clustering algorithm based on nearest neighbours heuristics. Pattern Recognition Letters **112**, 198–204 (2018). https://doi.org/10.1016/j.patrec.2018.07.017
2. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to information retrieval. Cambridge University Press (2009)