0-step K-means for clustering Wikipedia search results

Julian Szymański, Kamil Węgrzynowicz
Department of Electronic, Telecommunication and Informatics
Gdańsk University of Technology, Gdańsk, Poland
Email: julian.szymanski@eti.pg.gda.

Abstract

This article describes an improvement for K-means algorithm and its application in the form of a system that clusters search results retrieved from Wikipedia. The proposed algorithm eliminates K-means disadvantages and allows one to create a cluster hierarchy. The main contributions of this paper include the following: (1) The concept of an improved K-means algorithm and its application for hierarchical clustering. (2) Description of the WikiClusterSearch system that employs the proposed algorithm to organize Wikipedia search results into clusters.

1. Introduction

Since the beginning of the World Wide Web information retrieval has become a widely researched problem. The growing number of websites raises the need of development methods for finding resources effectively. Numerous approaches from Archie and Gopher to modern search engines have been developed [9]. In 1998 Google started with its revolutionary PageRank [1] algorithm used for ranking a website's content, and till now it is the dominant search engine. Beside its popularity, it has some drawbacks.

A typical web search engine offers only line-up presentation of retrieved data. The end user receives a long list of results that match the query and are sorted according to some relevance measures. Often we have to go through many links before we find those interesting to us.

One way of improving organization within search results is introduction of categorization based on similarity measures between returned results (that can be treated as documents). The similarity allows one to organize the results into thematic groups which can be achieved applying clustering techniques. Clustering makes the search faster as it allows us to see the results on a higher level of abstraction. It improves searching quality, because users don't have to waste their time on irrelevant websites and are able to find groups of the most relevant. There are several projects aiming to group search results (e.g.: clusty, carrot^2, PolyMeta and many others). The main problem with them is the fact the clusters they produce are not domain oriented – the labels they use weakly describe the content. The other drawback is the fixed similarity measure that compares documents according to the same, one criterion [10].

In our research we find Wikipedia very useful as the area of the experiments. Researching the limited repository of the documents allows us to tune the algorithms and after successful evaluation scale up them into the unlimited size of whole Internet. Wikipedia also offers the system of the categories that can be used as cluster labels which will make the categorization more user friendly.

2. Clustering algorithms

Clustering algorithms can be divided into two main groups:

1. *Hierarchical algorithms*. During one step of the algorithm the smallest clusters are made. In the next steps these groups are merged into bigger ones until one big cluster is made. As a result we get a hierarchical tree of clusters.
2. *Partitioning algorithms*. This type of algorithm results with the creation of flat groups of the most similar objects.

In this section we describe the K-means algorithm as the most popular partitioning algorithm and the Hierarchical Agglomerative Clustering algorithm as a demonstrator of a hierarchical, bottom-up approach.

K-means was described by James MacQueen [3] and it's widely used due to its simplicity. It requires to provide a priori number of desired clusters *K* and creates *centroids* which are centers of each cluster. In each iteration documents are assigned to the nearest centroid. K-means starts with randomly selected points as centroids. Next it iterates through all points and assigns each of them to the nearest centroid. As a result we receive first clusters. Next, new centroids are calculated and everything starts from the beginning, until they stop moving. K-means

minimize an objective function of squared error [11] described by the formula:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

where $||x^{(j)}_i - c_j||^2$ is a distance measure between point $x^{(j)}_i$ and centroid c_j.

As we consider application of a K-means algorithm for documents clustering, the following steps can be selected:

1. Mark as centroid random *K* documents,
2. Assign to centroid nearest documents,
3. When all documents are assigned, calculate new centroid positions. The centroid is an element laying exactly in the centre of a cluster. A new centroid is calculated as an arithmetic mean of every document represented as a point in feature space which is calculated as:

$$c_i = \left\{ \frac{x_{11} + \cdots + x_{1(n_i-1)}}{n_i}; \ldots; \frac{x_{j1} + \cdots + x_{j(n_i-1)}}{n_i} \right\}$$

4. Repeat steps 2 and 3 until centroids stop moving.

Despite the fact K-means always finds clusters, it does not guarantee that they will be optimal. It has a tendency to fall into local minima [4] which results in non- optimal clusters as output. There's no guarantee to get a global minimum. Each K-means execution on the same dataset can provide different results which is due to its random initialization - because initial centroids are selected randomly, the algorithm may return different results for the same input data. It is also a reason for falling into the local minima. This is the main disadvantage of K-means. In order to avoid that problem, K-means can be executed several times and return the best (averaged) result on output. However, this method may be time-consuming.

The next disadvantage is the necessity of providing *K* parameter on the algorithm's input. In document clustering it is not easy to calculate the optimal number of clusters and one general solution does not exist for this issue [11]. A possible approach is repeating execution of the algorithm for different values of *K* each time and choosing the best result (e.g. the one which has the smallest average distance between documents and centroids).

Hierarchical Agglomerative Clustering (HAC) is a representative of hierarchical (bottom up) algorithms [4]. It creates a similarity tree of all objects that are to be clustered called a dendogram. While clustering of the documents is considered HAC treats each document as a separate cluster and it uses a similarity (distance) measure to merge a pair of them into one, bigger cluster. Usually, during each iteration only 1 pair of clusters is merged. It goes on until one, great cluster is made. A tree of document dependencies is created by HAC, which results in a cluster hierarchy. Using typically Euclidean distance as similarity measure, one of the following strategies is taken to bind clusters together [4]:

- Single-link - the smallest distance between neighbors is taken into consideration,
- Complete-link - the biggest distance between neighbors is taken,
- Average-link (group-average) - at first, all possible distances between neighbors are calculated and finally the arithmetic mean is calculated and taken into consideration,
- Centroid clustering - only similarity between centroids of clusters is used.

In both cases objects belong only to one group, so clusters do not overlap. This is a big simplification of real life applications.

2.1 K-means as an alternative to HAC

HAC is a algorithm which may cause problems with defining a threshold at which the dendogram should be cut. Also it tends to be slow - only one merging operation during iteration is performed. This issue arises especially for documents clustering where high dimensional spaces are processed. Also there is no guide how to produce high quality clusters from a dendogram. On the other hand, it produces a hierarchy of the clustered objects that while applied for document categorization is a big advantage because it shows conceptual structure of the documents set.

K-means can be used to produce hierarchy as well as HAC, yet it is more complicated. K-means itself produces only flat clusters. To make a hierarchy K-means should be applied to each cluster again. It can be repeated until single-item clusters would be made and. This is a top-down method of clustering. The other method could be merging clusters into bigger ones, until we get one cluster in the end. It would be a similar approach as HAC, but hierarchy is obtained in bottom-up approach.

The main problem with K-means is to determine the number of expected clusters. One of the attempts to estimate *K* was made in *G-means* algorithm [2]. G-means (Gaussian-means) run statistical tests against each cluster to test data consistency with normal (Gaussian) distribution. During each iteration G-means finds clusters without normal distribution and divides them into smaller clusters. Iterations end when all clusters have normal distribution [2].

Another attempt was made in *Seed K-means* [5]. The Algorithm has two phases: seed extraction and cluster creation. First phase consists in determining initial *K* seeds by analyzing patterns. A measure of seed goodness helps to distribute them correctly in order to choose best initial centroids. In the second phase K-means is used with initial centroids as input.

2.2 0-step K-means

The 0-Step approach is an improvement for a K-means algorithm. The basic idea of step 0 is to initially group the data into pre-clusters. It is a preparatory stage before launching K-means.
Text document are similar to each other in some degree. The **V**ector **S**pace **M**odel (VSM) [8] text representation with **T**erm **F**requency **I**nverse **D**ocument **F**requency (TFIDF) [7] weighting has been used to calculate similarities between documents.
Each of the documents is compared against each other and if the similarity measure[1] is beyond a defined threshold then they fall into the same group. When a document doesn't belong to any group, then a new one is created. The listing below shows pseudo-code of 0-step algorithm.

```
List stepZero(collection) {
  List groups = new List();
  foreach(document in collection) {
    if(groups.length == 0) {
      create new group;
      add document to new group;
    }
    foreach(group in groups) {
      foreach(doc in group) {
        if(similarity(doc,document) >
           DOCUMENT_SIMILARITY_THRESHOLD)
        {
          add document to group;
          break;
        }
      }
    }
    if(document not added to any group) {
      create new group;
      add document to new group;
    }
  }
  return groups;
}
```

Pseudo-code of 0-step K-means

Resulting groups of documents may overlap. The number of groups becomes an estimation of *K* parameter for K-means as well. Parameter *Document Similarity Threshold* controls how many groups (initial clusters) we will get. Usually it is set empirically. Output of Step 0 goes on input of K-means. This way we provide *K* and initial centroids for K-means.
This approach stabilizes K-means, because each time for the same input data Step 0 returns the same groups. Thus the randomness of K-means is eliminated and it takes less iterations to build clusters.

2.2.1. Hierarchy through binding clusters

Binding clusters is an idea for creating a hierarchy of clusters with a K-means algorithm. K-means may produce many small clusters compared to the total amount of data. These clusters may be similar to some others, so they can be bound into one group.
For each cluster (or cluster representative, e.g. centroid) similarity to others is measured. Clusters with the biggest similarity that exceeds the given parameter *Cluster Similarity Threshold* are bound in one group. Iterations end when no more binding operation is made. The process of clusters binding results in creation of a cluster hierarchy that allows one to present documents on different conceptual levels.

3. Evaluation measures

Evaluation of search result clustering is not a simple task, because every person can group the same set of documents differently. A single person is always subjective. Because hand assessment is time consuming, costly, and uncomfortable to realize, methods for automatic evaluation are needed. There are a number of such methods, but they can be divided into two main groups [6]:
Internal metrics. Is evaluation without external knowledge and cohesion and distance of clusters are validated here. Often these kind of metrics are similar to an objective function of a clustering algorithm that can be calculated as:

$$\Phi(C) = \frac{1}{|C|}\sum_{c_i \in C} |c_i| \times sim(c_i, c_i)$$

where *sim()* is similarity function. Internal metrics are reported to be the best measures for comparing clustering results on the same data collection. However one has to keep in mind that the whole computational environment cannot change (e.g. similarity measures, clustering algorithm, etc.).
External metrics. These metrics allow us to evaluate the usefulness of received clusters according to human made judgments. One of the informal metrics is to collect feedback from users of a clustering system in the form of

[1] Different similarity measures can be taken. Most popular are Euclidean distance, Manhattan distance, cosine similarity.

questionnaires. Another type of external metrics are formal metrics based on a relevance set. The most popular are Precision (P), Recall (R), F-measure and Purity.

Precision is the percentage of retrieved documents that are relevant:

$$P = \frac{Number\ of\ relevant\ documents}{Total\ amount\ of\ documents}$$

Recall is defined as the percentage of relevant documents that were retrieved:

$$R = \frac{Number\ of\ relevant\ documents}{Total\ amount\ of\ relevant\ documents}$$

F-measure is a composition of Precision and Recall (weighted harmonic mean) and keeps a balance between them [4]:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

where β (1,∞) is a weight coefficient. For β = 1 F-measure balances P and R. By increasing β we put emphasis on Precision. Most common values for β are 1, 3 and 5.

Purity can measured by selecting the number of correctly assigned documents and dividing it by *N* (total amount of documents), when each cluster is assigned to the class which is most frequent in the cluster:

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

where Ω represents clusters set, C is a set of classes, ω_k is k-th cluster and c_j is j-th class.

4. System prototype

Based on 0-Step approach we have created a prototype system named WikiClusterSearch. It can automatically organize Wikipedia. For now only Polish Wikipedia is supported.

WikiClusterSearch was written in C# in .NET 3.5 environment. Client side uses the advantage of ASP.MVC 2.0 technology combined with jQuery JavaScript library. The system is built in modular architecture, each responsible for performing a different task. Main modules are presented in Figure 1.

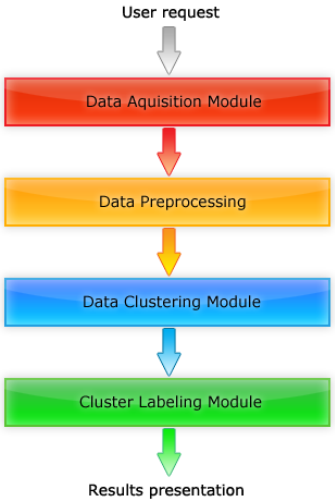

Figure 1 WikiClusterSearch main modules

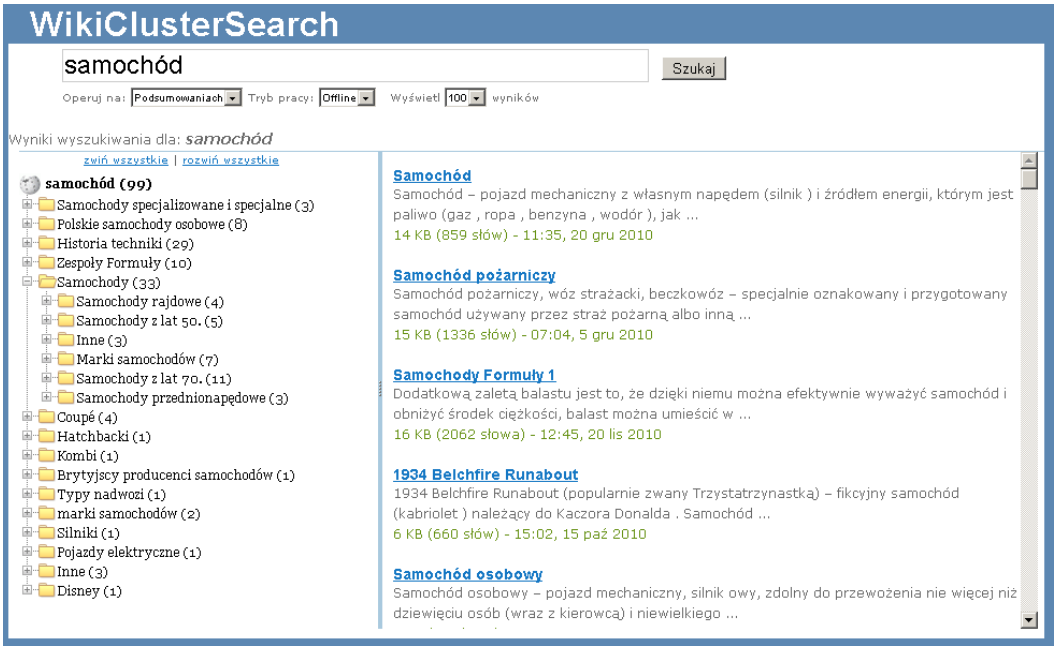

Figure 2 Snapshot of the system user interface available under http://swn.eti.pg.gda.pl/UniversalSearch

WikiClusterSearch (WCS) has demonstrated that Step 0 K-means can be used to obtain a good quality clusters hierarchy. The system is available on line under http://swn.eti.pg.gda.pl/UniversalSearch. For a given phrase it retrieves articles from Wikipedia and forms clusters in the fly.

Figure 2 shows example clusters formed by WCS system for articles retrieved from Polish Wikipedia for a query *samochód* (car).

5. Evaluation

Evaluation was performed using a relevance set, which was prepared manually for 7 test queries: kernel, nucleus, cat, equation car, networks, atomic nucleus, catholic church. The relevance set created manually was compared against the clusters returned by WikiClusterSearch system. Results of F-measure ($F_1$and F_5) and Purity are shown in Table 1.

Table 1 Results of clustering pages for given test queries

Query	F_1	F_5	Purity
kernel	0.82	0.85	0.87
cat	0.89	0.91	0.91
equation	0.73	0.74	0.78
car	0.74	0.76	0.76
networks	0.85	0.87	0.88
atomic nucleus	0.71	0.70	0.89
catholic church	0.61	0.62	0.62

What can be seen from the obtained results; Purity is kept on a high level for almost all queries. These facts come as a result of the number of clusters. WikiClusterSearch makes quite a large number of clusters, thus they have influence on the result. Purity takes into consideration the number of documents which belong to the most frequent class in that cluster. It doesn't matter if cluster label (class) is different. This is the disadvantage of a Purity metric.

An F-measure is a very popular metric used in evaluation of text documents clustering. It's well known to the Information Retrieval research society and its documentation is kept on a high standard level. It is also balancing (for $\beta = 1$) Precision and Recall replacing both of them with one metric. F_5 puts emphasis on clustering Precision. This situation is similar here as it was with the Purity measure. A high level of this metric indicates that clusters are of good quality according to human judgments. Only clusters for one query were scored below average.

6. Conclusions and future work

In this paper we presented our approach to constructing clusters organized in a hierarchy for text documents categorization. Step 0 as an improvement for K-means, eliminates the issues with the number of expected clusters, randomness, and initial centroids.

Evaluation shows the system WikiClusterSearch that employs the proposed method produces high quality clusters. Its modular architecture allows us to improve achieved results and test other approaches to text clustering.

System implementation can be improved. Code optimizations and cache would increase the speed of the system. Now WikiClusterSearch supports only Polish Wikipedia, and preprocessing of text is written only for Polish text. We plan to extend it to English and perhaps other languages.

Acknowledgment

This work was supported by the Polish Ministry of Higher Education under research grant no. N N519 432 338.

References

[1] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[2] Greg Hamerly and Charles Elkan. Learning the *k* in *k*-means. In *Advances in Neural Information Processing Systems*, volume 17, 2003.

[3] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[4] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, 2008.

[5] Eun Mi Kang Miyoung Shin, Seon Hee Park. Automatically finding good clusters with seed k-means. *Genome Informatics*, 14:326–327, 2003.

[6] Magnus Rosell. Introduction to text clustering, 2008.

[7] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[8] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[9] Bill Slawski. Just what was the first search engine? http://www.seobythesea.com/?p=106.

[10] J. Szymański and W. Duch. Dynamic Semantic Visual Information Management. *Proceedings of the 9th International Conference on Information and Management Sciences*, pages 107–117, 2010.

[11] R. Xu and D.C. Wunsch. *Clustering*. IEEE Press, 2009.