Kotouza *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2020) 9:2
https://doi.org/10.1186/s13677-019-0150-y

Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

# A dockerized framework for hierarchical frequency-based document clustering on cloud computing infrastructures

Maria Th. Kotouza[1*], Fotis E. Psomopoulos[2,3] and Pericles A. Mitkas[1]

**Abstract**

Scalable big data analysis frameworks are of paramount importance in the modern web society, which is characterized by a huge number of resources, including electronic text documents. Document clustering is an important field in text mining and is commonly used for document organization, browsing, summarization and classification. Hierarchical clustering methods construct a hierarchy structure that, combined with the produced clusters, can be useful in managing documents, thus making the browsing and navigation process easier and quicker, and providing only relevant information to the users' queries by leveraging the structure relationships. Nevertheless, the high computational cost and memory usage of baseline hierarchical clustering algorithms render them inappropriate for the vast number of documents that must be handled daily. In this paper, we propose a new scalable hierarchical clustering framework, which uses the frequency of the topics in the documents to overcome these limitations. Our work consists of a binary tree construction algorithm that creates a hierarchy of the documents using three metrics (Identity, Entropy, Bin Similarity), and a branch breaking algorithm which composes the final clusters by applying thresholds to each branch of the tree. The clustering algorithm is followed by a meta-clustering module which makes use of graph theory to obtain insights in the leaf clusters' connections. The feature vectors representing each document derive from topic modeling. At the implementation level, the clustering method has been dockerized in order to facilitate its deployment on cloud computing infrastructures. Finally, the proposed framework is evaluated on several datasets of varying size and content, achieving significant reduction in both memory consumption and computational time over existing hierarchical clustering algorithms. The experiments also include performance testing on cloud resources using different setups and the results are promising.

**Keywords:** Hierarchical document clustering, Topic modeling, Docker, Performance testing

## Introduction

Hierarchical clustering has been proven to be a useful technique in the field of document organization, as it constructs a hierarchy structure of document collections and sub-collections. Such a structure can make the browsing and navigation process easier and quicker [1] by hiding irrelevant information from the users. Since each cluster and the corresponding sub-clusters represent a set of topic and sub-topics relationships [2], the hierarchy can help automated systems to return only relevant information to the user, by exploiting the relationships stored in the structure. Moreover, the hierarchy can be used to visualize and interactively explore large amounts of documents [3]. Finally, the hierarchy may be used as a decision tree for the categorization of new documents. However, existing solutions for hierarchical document clustering are faced with serious challenges.

Some of the current problems with document clustering [4] include the selection of appropriate document features and similarity measures, the quality assessment of the clusters, the implementation of an efficient clustering algorithm which can make optimal use of the available memory and CPU resources, the association of meaningful labels to the final clusters, and the consideration of the

*Correspondence: maria.kotouza@issel.ee.auth.gr
[1]Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece
Full list of author information is available at the end of the article

semantic relationships between words. Hierarchical document clustering methods have to deal with additional challenges, including the handling of the very high dimensionality of the data. A medium to large set of documents can contain over 10,000 documents; this means that there can be millions of term-document relations, thus leading to an extremely high computational complexity and memory usage. This issue arises from the way most classical hierarchical clustering methods are implemented: they are based on the formulation of high dimensional distance matrices, used for pairwise comparisons between all the available data points.

The high volume of documents that have to be handled daily on the web presents a challenge to a cloud environment as well. In order to provide efficient solutions, researchers are increasingly turning towards scalable approaches, such as the utilization of cloud resources in addition to local computational infrastructures. The combination of running big data analytics algorithms using cloud computing infrastructures seems to be the solution. Cloud computing [5] provides shared computing resources on-demand over the Internet, including large numbers of compute servers and other resources, that have the ability to be scaled up and down according to the computational requirements. The topology of the computers in the cloud is usually hidden from the end user.

Taking all these issues into account, this work focuses on implementing a scalable hierarchical clustering algorithm for document clustering. It attempts to overcome limitations regarding the number of documents that can be handled by existing algorithms due to memory limitations, and to reduce the overall computational time. The innovation of our proposed algorithm lies in the fact that, instead of constructing an NxN similarity matrix by computing the pairwise similarities between all data points of the dataset in order to construct the hierarchical tree, we build a low-dimensionality frequency matrix as a representation of the root cluster. This cluster is then split recursively while moving down in the hierarchy, which significantly reduces the memory requirements. Additionally, the implementation of this work is based on a distributed computing architecture and therefore can handle an increasing number of documents based on the available resources. The input of our algorithm consists of documents represented as a bag of topics derived from topic modeling. The documents are transformed into the appropriate for the algorithm format during the preprocessing and transformation phases in our proposed framework. The whole framework has been dockerized in order to facilitate easy deployment on cloud computing infrastructures.

This work is an extended version of our previous work [6] that presented a multi-metric hierarchical clustering framework for item clustering. Here, we extend the previous work by re-designing our framework in order to be applicable to the more general field of document clustering, and we add a meta-clustering module to the framework. We explore the effectiveness and the performance of our method regarding memory usage and computational time through a more detailed evaluation and many more experiments, utilizing several datasets of varying sizes and content. We compare the results with more baseline hierarchical clustering methods, and we make use of the external evaluation metric FScore. Furthermore, we extend the previous work by parallelizing our clustering algorithm to achieve scalability, we make it suitable for cloud execution using a virtualization solution, and we measure the performance of the method using different hardware resources.

The rest of the paper is organized as follows; the "Literature review" section discusses related work, while the proposed integrated framework for document clustering is analysed in "A new document clustering framework" section. In "The hierarchical clustering algorithm" section our innovative hierarchical clustering algorithm is detailed, whereas "Experiments and evaluation" section contains the experimental results and the clustering evaluation. Finally, conclusions and future work are highlighted in the "Conclusion" section.

## Literature review

### Topic modeling in document clustering

Getting from an initial collection of documents to a clustering of the collection is an elaborate procedure, which usually involves several stages. The basic operations are feature extraction and selection, document representation and clustering [4]. Feature extraction is usually the first step of the process and filters out non-appropriate words from the documents' descriptions. Feature selection is a preprocessing method that removes noisy features and reduces the dimensions of the feature space, in order to yield a better understanding of the data and overall better performance of the clustering method that takes as input those data. In the feature selection stage, various probabilistic models have been used in the literature, like Latent Dirichlet Allocation (LDA) [7] and Probabilistic Latent Semantic Analysis (PLSA) [8]. Today, a lot of research works around topic modeling focus on distributed implementations of LDA, such as AD-LDA [9], PLDA [10] and PLDA+ [11]. BigARTM [12] is another distributed implementation for topic modeling which includes all popular models such as LDA, PLSA, and many others. Other approaches make use of deep learning techniques for topic extraction (e.g. lda2vec [13]).

Ahmadi et al. [14] proved that topic model based clustering methods generally achieve better results than only applying traditional clustering algorithms like the K-

means. LDA has been used in many papers for representation and dimensionality reduction of text documents, as well as for uncovering semantic relations in the text [15]. Ma et al. [16] used LDA for document representation and identification of the most significant topics, the K-means++ algorithm was used to define the initial centers of the clusters and the K-means algorithm was used to form the final clusters. Qiu and Xu [17] presented a clustering method, where the LDA was used to extract topics from the texts and the centroids of the K-means algorithm were selected among the nouns with the highest probability values. More recently, Onan et al. [18] proposed an improved ant clustering algorithm, where two novel heuristic methods are proposed to enhance the clustering quality of ant-based clustering. The latent Dirichlet allocation (LDA) was used to represent textual documents. Except from the classical LDA method, many variants were examined in the literature [19, 20], including hierarchical LDA, correlated topic models and hierarchical Dirichlet process.

#### Document clustering

According to [21], document clustering can be divided into hard clustering, where each document is assigned to exactly one cluster, and soft clustering, where each document is allowed to appear in more than one clusters. Hard clustering methods can be further categorized in the following sub-categories: **1)** Partitioning methods, which allocate documents into a fixed number of clusters with K-means algorithm and its variants being the most popular one, **2)** Hierarchical methods [3], which build a dendrogram of clusters and **3)** Frequent itemset-based methods [22], which use association rule mining techniques to form the clusters. In [7] some representative papers applying those three categories are reviewed.

Hierarchical clustering algorithms [23] are categorized in two major categories: **a)** agglomerative (or top-down) algorithms and **b)** divisive (or bottom-up) algorithms. Agglomerative algorithms can be further categorized according to the similarity measures they employ into single-link, complete link, group-average, and centroid similarity. Top-down algorithms typically are more complex, as they hold information about the global distribution of the dataset, in contrast to bottom-up methods that make clustering decisions based on local patterns. The advantages of Hierarchical clustering algorithms are that they compose a tree of clusters that comprises a richer data structure with more information than those provided by flat algorithms' output, and the fact that they do not require users to define the number of clusters.

Nevertheless, the complexity of the naive hierarchical clustering algorithm is $O(N^3)$ as for every decision that needs to be taken, an exhaustive scan of the $NxN$ similarity matrix is necessary. Other more efficient algorithms can reduce the complexity to $O(N^2 logN)$ (with a heap in the general case) or even $O(N^2)$ (with SLINK [24] for single-linkage, CLINK [25] for complete-linkage clustering in the general case, and ROCK [26], Chameleon [27] for categorical data). BIRCH [28] and its extensions [29] comprise hierarchical clustering procedures that are especially suitable for very large databases, and comprise state of the art incremental hierarchical methods. However, the creation of the $NxN$ similarity matrix is necessary for the majority of the algorithms, hence memory requirement demands become extremely high.

There have been many recent studies on Hierarchical Clustering algorithms. In [30], an alternative approach of a single-linkage clustering algorithm was proposed, which was based on minimum spanning trees and had the same complexity as the single-linkage algorithm. In [31], a new non-greedy incremental algorithm for hierarchical clustering was suggested, which efficiently routes new data points to the leaves of an incrementally-built tree. Another recent work [32] proposed a hierarchical clustering algorithm based on the hypothesis that two reciprocal nearest data points should be put in one cluster. In another line of work, many researchers treated similarity-based hierarchical clustering as an optimization problem, making use of suitable objective functions [33, 34]. In [35] for example, the author introduces a cost function that, given pairwise similarities between data points, assigns a score to any possible tree on those points.

In this paper, we introduce a method for clustering documents represented by a number of topics, using an approach that does not demand pairwise comparisons between the documents, but it is instead based on the use of low dimensional frequency matrices. Since the main algorithm makes use of the frequency of occurrence of the main terms in the documents, we call it Frequency-based Hierarchical Clustering (FBHC). A relevant clustering method that we presented in one of our previous works [36] makes use of frequency matrices to construct an hierarchy of biological sequences.

## A new document clustering framework

In this section, we present an efficient framework for hierarchical document clustering which makes use of topic modeling to extract feature vectors that represent the processed documents. The proposed framework is shown schematically in Fig. 1 and it is formally described in the following steps:

**Input** Documents in bag-of-words representation

**Step 1: Word preprocessing**

*Stemming, Removing stop words, Making orthographic transformations, Stripping punctuation and substitution, Excluding words that are not included in the WordNet database*

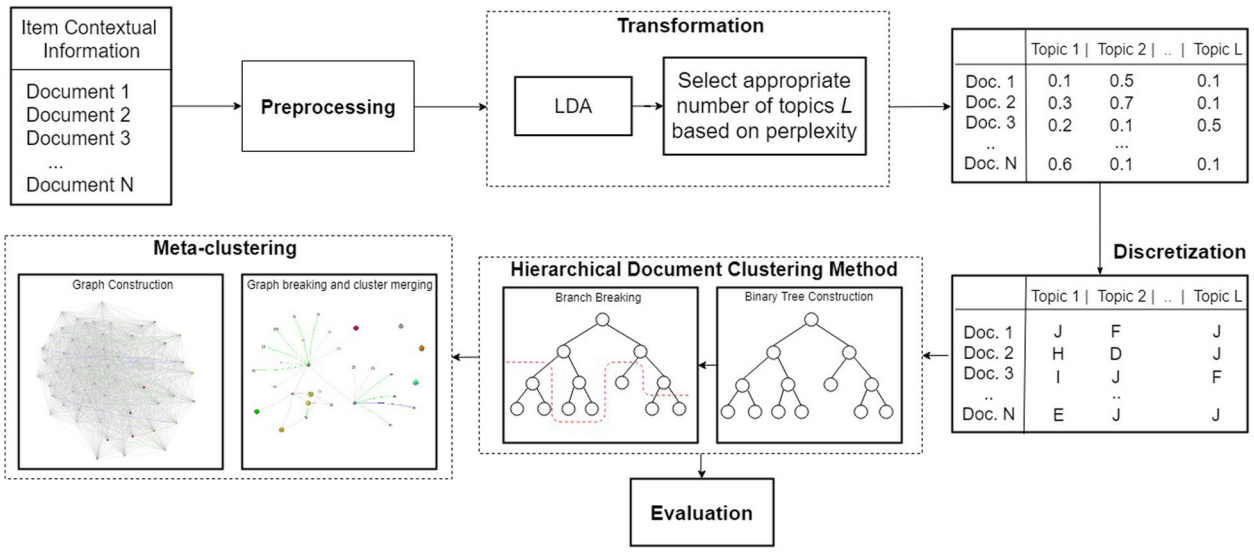


**Fig. 1** The proposed framework for hierarchical document clustering

**Step 2: Data transformation**
***Step 2.1:*** Select a method to perform topic modeling
***Step 2.2:*** Select the number of topics $N_\theta$ and the number of words per topic $N_w$
***Step 2.3:*** Transform documents to topic vectors
**Step 3: Data discretization**
***Step 3.1:*** Select the number of bins $B$
***Step 3.2:*** Discretize the topic vectors
**Step 4: Hierarchical Clustering**
***Step 4.1:*** Apply the Binary Tree Construction Algorithm (Algorithm 1)
***Step 4.2:*** Apply the Branch Breaking Algorithm (Algorithm 2)
**Step 5: Meta-Clustering**
***Step 5.1:*** Graph Construction
***Step 5.2:*** Graph Clustering
***Step 5.2.1:*** Select the threshold *thrG*
***Step 5.2.2:*** Exclude all those edges with *weights* $<$ *thrG*
***Step 5.2.3:*** Apply the Graph Merging Algorithm (Algorithm 3)
**Step 6: Evaluation**
***Step 6.1:*** Select a technique to compute semantic similarity between the derived topics
***Step 6.2:*** Compute Topic Similarity *TS* for each cluster
***Step 6.3:*** If the actual class labels are known, compute *Fscore*

### Data preprocessing module

The initial data that are taken as input to the framework are documents composed of words. Each word has a corresponding frequency of appearance. Before importing the data to the data transformation module, the words are preprocessed using various methods, including stemming (using the Porter Stemming Algorithm [1]), removing stop words, making orthographic transformations (using the spelling corrector [2]), stripping punctuation and substitution. The words that are not included at the WordNet [37], a lexical database of English words, are excluded from the dataset at this module.

### Data transformation module

The data transformation module employs topic modeling to the desired input document in order to transform it into a compressed representation in terms of its topics. In this way we can deal with the high dimensionality and the sparsity of the features of the documents. Topic modeling is based on the assumption that each document $d$ is described as a random mixture of topics $P(\theta|d)$ and each topic $\theta$ as a focused multinomial distribution over terms $P(w|\theta)$. The number of topics $N_\theta$ and the number of terms per topic $N_W$ are specified by the user and express the degree of specialization of the latent topics. As the data transformation module is not part of the proposed clustering method, any topic modeling method can be used as part of this module, as a plugin. Literature on topic modeling offers hundreds of models adapted to different situations.

**LDA**: Latent Dirichlet Allocation [38, 39] is a commonly used method to extract semantic information from the documents and create a feature vector for each document. LDA builds a set of $N_\theta$ thematic topics, each expressed with a set of $N_W$ terms, utilizing terms that tend to co-occur in a given set of documents. The topic-term distribution $P(\theta|d)$ and the document-term distribution $P(w|\theta)$

[1] https://tartarus.org/martin/PorterStemmer/index-old.html
[2] https://github.com/amarjeetanandsingh/spell_correct

are estimated from an unlabeled corpus of documents $D$ using Dirichlet priors.

**BigARTM**: BigARTM [12] is an open-source library for regularized multimodal topic modeling of large collections, which is based on a non-Bayesian multicriteria approach — Additive Regularization of Topic Models, ARTM [40]. It is a distributed implementation which is proven to be very fast and ideal for big collections of documents.

**Lda2vec**: lda2vec [13] is a deep learning-based model which creates topics by mixing Dirichlet topic models and word embedding. It constructs a context vector by adding the composition of a document vector and the word vector, which are simultaneously learned during the training process.

#### Data discretization module

The numeric vectors created by the data transformation module, i.e. the mixture of topics $P(\theta|d)$ calculated by the topic modeling process, are discretized into $B$ partitions by assigning each value into a bin based on the closed interval where it belongs to. By making use of alphabetic letters to represent the bins, the numeric vectors are converted into character vectors, which constitute the input data to the clustering procedure. Practically, it is a lossy compression where the number of bins $B$ is selected based on the amount of information we want to be considered by the model.

#### Design for cloud

Since the proposed clustering algorithm is oriented towards analyzing big data that may not fit in a single machine, provision for cloud execution becomes a necessity. Cloud computing [41] is moving from large-scale centralized data centres to more distributed multi-cloud settings, which may contain networks of larger and smaller virtualized infrastructure runtime nodes. The use of containers constitutes a lightweight virtualization solution characterized by low resource and time consumption.

Docker [42] is a containerization platform that allows Linux applications, their dependencies, and their settings to be composed into Docker images. These images run as Docker containers on any machine running the Docker daemon, which utilizes kernel namespaces and control groups to isolate running containers and control their set of resources. This makes the deployment of cloud-oriented applications easy, as the image of an application has to be built only once and then it can be deployed on every system running the Docker deamon. Docker is also appropriate for software benchmarking experiments, since multiple Docker images can be created based on the same root image but containing different benchmarked configurations.

An image of the proposed FBHC algorithm was built using the Docker technology in order to run performance experiences using different hardware resources in the cloud. The resources that were used and the corresponding experimental results are described in "Performance testing in the cloud" section.

### The hierarchical clustering algorithm

In this section, we propose a novel hierarchical clustering algorithm, consisting of two phases: **1)** the construction of a top down binary tree by consecutively dividing the frequency matrix [6] into two sub-matrices until only unique sequences remain at the leaf-level, and **2)** the branch breaking algorithm, where each branch of the tree is pruned at an appropriate level using thresholds for the metrics. The metrics that are used to form the clusters are: **a)** Identity ($I$), **b)** Entropy ($H$) and **c)** Bin Similarity ($BS$), and are described in [6]. In the final, meta-clustering phase, a graph of the leaf clusters generated by the clustering algorithm is constructed.

#### Binary tree construction

The first phase of the clustering method consists of a top down hierarchical clustering algorithm (Algorithm 1). At the beginning of the process, it is assumed that all $N$ sequences belong to a single cluster ($C_0$), the root cluster, which is consequently split recursively while moving down the different levels of the tree. Ultimately, the constructed output of the clustering process is presented as a binary tree. The tree is constructed per level by following a procedure for each cluster ($C_i$) of the specific level, that can run in parallel. This can be formally described in the following steps:

**Step 1** Construct frequency and frequency-similarity based matrices ($FM_i$, $FSM_i$).

**Step 2** Compute Identity, Entropy and Bin Similarity metrics of the matrices ($I_i$, $IS_i$, $H_i$, $HS_i$, $BS_i$) applying the equations described in [6], on the $FM_i$ and the $FSM_i$ respectively. From now on, the identity metric computed on the $FSM$ will be called Similarity ($IS$).

**Step 3** Split the frequency matrix into two sub matrices according to the following criteria:

- ***Criterion 1:*** Select the element of the $FM_i$ with the highest percentage.
- ***Criterion 2:*** If the highest percentage value exists in more than one elements of $FM_i$, the column with the lowest entropy value is selected.
- ***Criterion 3:*** In the case where more than one columns exhibit the exact same entropy value, *Criterion 1* is applied to the $FSM_i$.
- ***Criterion 4:*** In the case of non-unique columns, *Criterion 2* is applied to the $FSM_i$.

***Criterion 5:*** If the number of columns is still more than one, one column from the above sub group of columns is randomly selected.

**Step 4** Update the Level matrix ($Y$) and the Metric matrix ($M$) that contains the metrics for each cluster ($I$, $IS$, $H$, $HS$, $BS$).

**Step 5** Check for leaf-cluster.

At the beginning of the process, the user can select the type of the algorithm, i.e. whether the split of the matrices is performed on the *FM* ($identity_{algo}$ option), or on the *FSM* ($similarity_{algo}$ option). In the $similarity_{algo}$ option, *Criteria 1 & 2* are skipped at Step 3.

#### Branch breaking

The second phase of the clustering method consists of the branch breaking process (Algorithm 2). This algorithm is applied to the binary tree derived from the first phase. In the field of document clustering, creating a hierarchy of the documents can be very useful for document organization. In many cases, except from the formulation of a hierarchy structure of the clusters, extracting meaningful groups can also be useful. In a partitioning clustering algorithm, the exact number of clusters to be created is chosen by the user. In the FBHC algorithm, a solution to extract useful groups from the binary tree would be to cut the tree at a specific level $T_C$, obtaining all those clusters that belong to level $T_C$ and all the leaf clusters that belong to higher levels than $T_C$.

Since the tree is asymmetric and the number of documents in each cluster varies, the tree cannot be cut by selecting a unique level $T_C$ for the overall tree. A more accurate procedure to address this problem is to prune the tree using branch-specific thresholds: For each branch, the parent cluster is compared to its two children clusters recursively as one goes down through the path of the tree branch. The comparison is applied using the metrics that have been computed for each cluster $C_i$ ($I_i$, $IS_i$, $H_i$, $HS_i$, $BS_i$) and user selected thresholds for each metric (*thrI*, *thrH*, *thrBS*). An additional limitation set for the identity metric is that the leaf clusters must have an Identity value higher than 20%. This lower threshold is set to avoid pruning at a very high level of the tree in the case that Identity is too small and the improvement in the metrics is not big enough.

#### Graph construction

The hierarchy structure created during the clustering phase is ideal for the graph theory application. A graph can be useful to uncover connections between the clusters and obtain an insight of how similar the leaf clusters are. This information can be used to merge similar clusters together as a next step. To this end, an undirected, weighted and fully connected graph is constructed using the binary tree.

**Algorithm 1:** Binary Tree Construction

N: #Sequences, TL: Tree Depth, NC: #Clusters
X: Input matrix ($N \times 2$) {sequence id, sequence}
Y: Level-Cluster matrix ($N \times TL$)
M: Metric Matrix ($NC \times 5$) {I, IS, H, HS, BS}
type: The algorithm type ($identity_{algo}$ or $similarity_{algo}$)

**Data**: X, type
**Result**: Y, M

**Initialization**
Create a root node (cluster $C_0$) for the tree;
**Iteration foreach** *new level-l* **do**
  **beginParallelism()**;
  **forall the** *cluster-i in l* **do**
    *leaf* ← False;
1  Compute $FM_i$ and $FSM_i$ matrices;
2  Compute metrics ($I_i$, $IS_i$, $H_i$, $HS$, $BS$);
3  **if** $C_i$ *is leaf* **then**
      *leaf* ← True;
      **Return**(*leaf*)
    **end**
4  Update M matrix;
    **Select** $cell_i$ **of** $FM_i$ **or** $FSM_i$ **according to the criteria**
5  **if** *type* = $similarity_{algo}$ **then**
      Elm ← the elements of $FSM_i$ with the max value;
      Go to step 9;
    **end**
6  Elm ← the elements of $FM_i$ with the max value;
7  **if** *Elm.length* <2 **then**
      $cell_i$ ← Elm;
      Go to step 14;
    **end**
8  Elm ← the elements of Elm with the $min(H_i)$;
9  **if** *Elm.length* <2 **then**
      $cell_i$ ← Elm;
      Go to step 14;
    **end**
10  Elm ← the elements of Elm with the $max(FSM_i)$;
11  **if** *Elm.length* <2 **then**
      $cell_i$ ← Elm;
      Go to step 14;
    **end**
12  Elm ← the elements of Elm with the $min(HS_i)$;
13  **if** *Elm.length* <2 **then**
      $cell_i$ ← random element of Elm;
    **end**
    **Division**
14  j ← index of the left child of node i ($C_i$);
15  Add the sequences that belong to $cell_i$ to cluster $C_j$;
16  Add all the other sequences to cluster $C_{j+1}$ ;
    **Return**(*leaf*, *j*, $C_j$, $C_{j+1}$);
  **end**
  **endParallelism()**;
17  Collect the results from the parallel processes and fill in column $Y[, level + 1]$ with the corresponding cluster ids;
**end**
**Return** Y, M;

The graph is built by computing the graph similarity matrix which is a square matrix with order equal to the number of leaf clusters ($C$). The graph matrix is computed

**Algorithm 2:** Branch Breaking

Y: Level-Cluster matrix ($N \times TL$)
M: Metric Matrix ($NC \times 5$) {I, IS, H, HS, BS}
thrA: the threshold set for metric A
type: The algorithm type ($identity_{algo}$ or $similarity_{algo}$)

**Data**: Y, M, thrI, thrH, thrBS, algo
**Result**: Y, M

**Initialization**
Find all unique paths of tree from the root till the leaves;
**Iteration**
**foreach** *path-i* **do**
1 $PN_i \leftarrow$ the cluster ids that constitute the path
**Compare each cluster of the path with its children using the metrics**;
2 **if** $type = identity_{algo}$ **then**
condI $\leftarrow$ ($|I_j\text{-}I_{j+1}|+|IS_j\text{-}IS_{j+1}|$)*0.5 < thrI;
condH $\leftarrow$ ($|H_j\text{-}H_{j+1}|+|HS_j\text{-}HS_{j+1}|$)*50 < thrH;
condBS $\leftarrow$ $|BS_j\text{-}BS_{j+1}|$ < thrBS;
cut.condition $\leftarrow$ $I_j$>20 & (*condI*||*condH*||*condBS*);
**else**
condI $\leftarrow$ $|IS_j\text{-}IS_{j+1}|$<thrI;
condH $\leftarrow$ $|HS_j\text{-}HS_{j+1}|$*100 < thrH;
condBS $\leftarrow$ $|BS_j\text{-}BS_{j+1}|$ < thrBS;
cut.condition $\leftarrow$ $IS_j$>20 & (*condI*||*condH*||*condBS*);
**end**
3 **if** *cut.condition* **then**
Convert $C_j$ to leaf;
Update Y, M matrices;
break;
**end**
**end**
**Return** Y, M;

on the patterns that represent the clusters. The graph similarity $G_{i,j}$ between two clusters $C_i$ and $C_j$ is calculated as the combination of three aspects: **a)** the number of bins that these clusters have in common through the whole pattern that is computed using the identical bins ($pI$), **b)** the number of bins that these clusters have in common through the whole pattern that is computed using the groups of bins ($pS$), and **c)** the distance between the nodes $C_i$, $C_j$ of the tree. $TL$ is defined as the maximum distance presented in the binary, i.e. the distance between the nodes that are far apart from each other.

$$G_{i,j} = \frac{2}{N_\theta}|pI_i \cap pI_j| + \frac{2}{N_\theta}|pS_i \cap pS_j| + \frac{2}{TL}|C_i, C_j| \quad (1)$$

The representative pattern of a cluster is a string of length equal to the number of topics and it is extracted using the cluster's frequency matrix, as follows: The positions of the strings with an exact alignment are represented by the corresponding bin, whereas the rest of them are represented by the symbol "_". Suppose that we are interested in the distance between clusters 53 and 18. The clusters' patterns computed with the identical and grouped bins of cluster 53 and the corresponding patterns for cluster 18 are as follows:

$pI_{53}$ = _ R G _ _ _ _ _ R Y Y Y Y G _ _ V,
$pS_{53}$ = _ R G _ _ _ _ B R Y Y Y Y G _ A V,
$pI_{18}$ = _ R G _ _ _ _ _ _ _ _ _ _ _ D _,
$pS_{18}$ = _ R G _ _ _ _ _ G _ _ _ _ _ _ D _,
Then, $|pI_{53} \cap pI_{18}| = 2$ and $|pS_{53} \cap pS_{18}| = 2$.

After the graph construction, the graph is clustered into sub groups. Graph clustering is the task of grouping the graph nodes into clusters taking into consideration the weights of the edges, in such a way that there should be many high weighted edges within each node-cluster and relatively low between the node-clusters. The graph can be clustered using a user-selected threshold *thrG*, excluding from the graph all those edges that are characterized by a weight smaller than *thrG*. This threshold is expressed as a percentage and can be selected by observing the distribution of the weights' values. If the user wants to export a specific number of clusters, then a graph merging procedure can be applied. As described in Algorithm 3, the clustered graph is composed of sub-graphs *SGs*. By sorting the weights in descending order, the most highly similar and strongly connected *SGs* can be merged by assigning each node to the corresponding central node of the *SGs* where it belongs to and forming merged clusters, until the desired number of clusters is reached.

## Experiments and evaluation

In this section the datasets, the external evaluation measures and the four sets of experiments performed to evaluate and validate the proposed framework are presented.

### Experimental setup

#### Datasets

In order to evaluate the effectiveness of the proposed clustering method on text documents, we used various datasets from several domains such as sentiment analysis, news articles, medical documents, web pages and abstracts provided by [43, 44]. More specifically, we used 23 benchmark datasets from [43] in order to test the accuracy of our framework, with the smallest and the largest ones consisting of 204 and 18,808 documents, accordingly. The number of actual classes

of these documents vary from 4 to 51. The table also shows the number of terms of the original documents, i.e. the number of different words, and the final number of terms after the preprocessing. We also used two big datasets from [44], the NYTimes news articles and the PubMed abstracts, in order to evaluate the performance of the method in terms of computational time and memory usage. All these datasets are summarized in Table 1.

In order to apply the proposed clustering procedure, the datasets were preprocessed in the preprocessing module and then were transformed into numeric vectors using topic modeling. As use case in this paper we utilized the LDA method in the data transformation module. The number of topics $N_\theta$ that will represent the documents was chosen to be equal to 20, after experimenting on the values of $N_\theta$ from 5 to 500 and evaluating the results using the perplexity metric, as described in our previous work [6]. Thus, for each document of the datasets, we created topic vectors of length 20.

The document vectors were then discretized in 10 bins represented by alphabetic letters from A to J, making each document represented by a sequence of characters. The bin with the highest percentage is represented by *A*, whereas the one with the lowest percentage is described with *J*. In order to create the *FSM*, the groups of similar bins that were used are non-overlapping and are given by pairing bins in descending order i.e. $< A, B >, < C, D >, < E, F >, < G, H >$.

### Evaluation measures

Two external evaluation metrics were used to evaluate the effectiveness of the clustering procedure: *FScore* and *Topic Similarity*.

#### *FScore*

When we have knowledge about the true class where each document belongs to, then we can use *FScore* to measure the accuracy of the clustering results. A commonly used technique to measure *Fscore* in hierarchical clustering is

**Table 1** Summary of datasets used to evaluate the hierarchical clustering algorithms

| Dataset | Domain | # of docs | # of terms | # of terms after prep. | # of classes | Source |
|---|---|---|---|---|---|---|
| Classic4 | Abstracts | 7095 | 7749 | 6576 | 4 | [43] |
| Reviews | News articles | 4069 | 22,927 | 12,431 | 5 | [43] |
| Tr23 | TREC documents | 204 | 5833 | 4384 | 6 | [43] |
| LATimes | News articles | 6279 | 10,020 | 6389 | 6 | [43] |
| Tr31 | TREC documents | 927 | 10,129 | 6946 | 7 | [43] |
| La2s | News articles | 3075 | 12,433 | 8517 | 7 | [43] |
| WebKb | Web pages | 8282 | 22,892 | 11,009 | 7 | [43] |
| Tr12 | TREC documents | 313 | 5805 | 4283 | 8 | [43] |
| Re8 | News articles | 7674 | 8901 | 5379 | 8 | [43] |
| Tr11 | TREC documents | 414 | 6430 | 4632 | 9 | [43] |
| Tr45 | TREC documents | 690 | 8262 | 6016 | 10 | [43] |
| Tr41 | TREC documents | 878 | 7455 | 5406 | 10 | [43] |
| Oh10 | Medical documents | 1050 | 3239 | 2425 | 10 | [43] |
| Dmoz-Science | Web pages | 6000 | 5011 | 3719 | 12 | [43] |
| Dmoz-Health | Web pages | 3500 | 4217 | 3172 | 13 | [43] |
| Re0 | Articles | 1504 | 2886 | 2209 | 13 | [43] |
| Dmoz-Computers | Web pages | 9500 | 5011 | 3527 | 19 | [43] |
| Wap | Web pages | 1560 | 8460 | 5988 | 20 | [43] |
| 20 Newsgroups | E-mails | 18,808 | 45,434 | 16,499 | 20 | [43] |
| Re1 | Articles | 1657 | 3758 | 2863 | 25 | [43] |
| ACM | Digital library | 3493 | 60,768 | 16,315 | 40 | [43] |
| New3 | News articles | 9558 | 26,833 | 14,483 | 44 | [43] |
| Opinosis | Reviews | 6457 | 2693 | 2201 | 51 | [43] |
| NYTimes | News articles | 300,000 | 102,660 | 18,001 | - | [44] |
| PubMed | Abstracts | 8,200,000 | 141,043 | 21,451 | - | [44] |

**Algorithm 3:** Graph Merging

G: The graph similarity matrix containing the weights of the edges of the connected nodes
thrG: the threshold set for edge exclusion
A: the number of actual classes – if the number of actual classes is unknown, this is set equal to *NULL*
M: Metric Matrix ($NC \times 5$) {I, IS, H, HS, BS}

**Data**: G, thrG, A
**Result**: M

**Initialization**
Exclude all those edges of *G* with weights < *thrG*;
**if** *A != NULL* **then**
  Find the nearest neighbor of each node ;
  Find the most highly connected sub-graphs *SGs* ;
  Sort *SGs* by weight values in descending order ;
  Find the central nodes of *SGs* i.e. the nodes which have the bigger number of connections ;
  Starting from the strongest connections, assign the nodes of *SGs* to the corresponding central node and form merged cluster until the desired number of clusters is reached
**end**

**Return** M;

to take into account the overall set of clusters that are represented in the hierarchical tree. In this paper, we use the *FScore* introduced by [45]. Given a particular class $L_r$ of size $n_r$ and a particular cluster $C_i$ of size $n_i$ and assuming that $n_{ri}$ documents of cluster $C_i$ belong to the real class $L_r$, then the *Fscore* of this class and cluster is given by (4). To compute *FScore*, (2) and (3) must be used as follows.

$$F(L_r, C_i) = \frac{2 \times R(L_r, C_i) \times P(L_r, C_i)}{R(L_r, C_i) + P(L_r, C_i)} \quad (2)$$

where $R(L_r, C_i)$ is the recall value defined as $n_{ri}/n_r$, and $P(L_r, C_i)$ is the precision value defined as $n_{ri}/n_i$ for the class $L_r$ and the cluster $C_i$. The *FScore* of the class $L_r$, is the maximum *FScore* value attained at any node in the hierarchical clustering tree *T*. That is,

$$F(L_r) = max(F(L_r, C_i)), C_i \in T \quad (3)$$

The *FScore* of the entire clustering solution is then defined to be the sum of the individual class *FScore* weighted according to the class size.

$$FScore = \sum_{r=1}^{c} \frac{n_r}{n} F(L_r) \quad (4)$$

where *c* is the total number of classes. The higher the *FScore* values, the better the clustering solution is.

#### *Topic similarity*

Due to the sparsity of the frequency matrix of each cluster and the fact that each cluster is characterized by only a few topics, we evaluated the clustering results by calculating the semantic similarity between major topics of each cluster. The topic similarity is extracted using semantic analysis of the topics that were derived from topic modeling.

**Table 2** The FScores and the TS values for the different datasets using various hierarchical clustering methods

| Dataset | FScore | | | | | | TS-TS Actual | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Average | Single | Complete | Ward | Diana | FBHC | Average | Single | Complete | Ward | Diana | FBHC |
| Classic4 | 0.505 | 0.453 | 0.595 | **0.859** | *0.670* | 0.588 | 0.410 | NA | 0.410 | **0.00** | 0.410 | 0.410 |
| Reviews | 0.417 | 0.414 | 0.451 | 0.501 | **0.542** | *0.514* | 0.000 | NA | 0.000 | 0.000 | 0.000 | 0.000 |
| Tr23 | 0.428 | *0.429* | 0.433 | 0.436 | 0.435 | **0.447** | **0.001** | NA | 0.358 | 0.239 | *0.218* | NA |
| LATimes | 0.419 | 0.329 | 0.449 | **0.470** | 0.353 | *0.468* | 0.464 | NA | 0.464 | 0.464 | 0.464 | **0.016** |
| Tr31 | 0.387 | 0.387 | 0.387 | *0.388* | 0.387 | **0.391** | - | - | - | - | - | - |
| La2s | 0.273 | 0.272 | 0.272 | *0.274* | *0.274* | **0.274** | - | - | - | - | - | - |
| WebKb | 0.417 | 0.413 | 0.424 | **0.522** | 0.456 | *0.483* | **0.068** | NA | 0.276 | 0.276 | *0.166* | 0.191 |
| Tr12 | 0.284 | 0.283 | 0.286 | *0.290* | *0.290* | **0.305** | - | - | - | - | - | - |
| Re8 | 0.669 | 0.497 | 0.643 | 0.665 | *0.685* | **0.708** | 0.198 | NA | 0.001 | 0.198 | **0.005** | *0.006* |
| Tr11 | 0.310 | 0.308 | 0.315 | *0.318* | 0.311 | **0.329** | - | - | - | - | - | - |
| Tr45 | 0.245 | 0.242 | **0.256** | 0.254 | 0.248 | *0.254* | - | - | - | - | - | - |
| Tr41 | 0.294 | 0.294 | 0.293 | 0.294 | *0.295* | **0.302** | - | - | - | - | - | - |
| Oh10 | 0.264 | 0.207 | 0.238 | **0.277** | 0.239 | *0.271* | - | - | - | - | - | - |
| Dmoz-Science | 0.327 | 0.154 | 0.289 | *0.394* | 0.322 | **0.397** | 0.131 | NA | 0.131 | 0.069 | 0.112 | **0.050** |
| Dmoz-Health | 0.353 | 0.143 | 0.341 | *0.402* | 0.393 | **0.408** | 0.201 | NA | 0.083 | 0.201 | 0.136 | **0.019** |
| Re0 | 0.371 | 0.359 | *0.376* | 0.367 | 0.370 | **0.381** | - | - | - | - | - | - |
| Dmoz-Computers | 0.351 | 0.100 | 0.280 | **0.371** | 0.343 | *0.365* | 0.134 | NA | 0.178 | 0.178 | *0.102* | **0.025** |
| Wap | 0.188 | 0.181 | 0.186 | 0.188 | *0.190* | **0.193** | - | - | - | - | - | - |
| 20 Newsgroups | 0.556 | 0.096 | 0.449 | *0.566* | 0.546 | **0.569** | **0.064** | NA | 0.102 | *0.064* | 0.061 | 0.080 |
| Re1 | 0.210 | 0.198 | 0.210 | *0.212* | 0.211 | **0.221** | - | - | - | - | - | - |
| ACM | 0.422 | 0.050 | 0.408 | **0.440** | 0.414 | *0.425* | **0.003** | NA | 0.207 | 0.131 | 0.070 | *0.034* |
| New3s | 0.377 | 0.058 | 0.369 | 0.398 | 0.371 | **0.408** | 0.058 | NA | 0.229 | 0.251 | *0.091* | **0.019** |
| Opinosis | 0.339 | 0.058 | 0.328 | *0.347* | 0.340 | **0.353** | *0.020* | NA | **0.001** | 0.016 | 0.070 | 0.181 |

The best results achieved by an algorithm for each one of the datasets are highlighted as boldface, whereas the second highest results are presented in italics

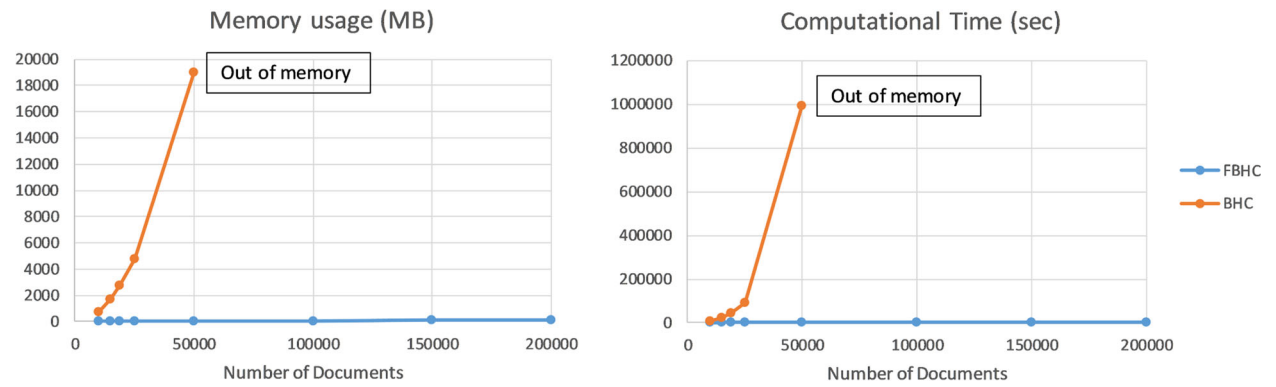


**Fig. 2** Comparison of the FBHC and BHC algorithms using subsets of the NYTimes dataset, in terms of performance

Semantic similarity, in contrast to string-based matching can identify semantically relevant concepts that consist of different strings. More specifically, semantic similarity is a metric that is used to measure the distances between a set of terms contained in documents based on their meaning or semantic concept. Many techniques to compute semantic similarities of words are reported in the literature. Using Word Embeddings such as Google's Word2Vec, or a semantic net such as WordNet are common techniques to compute semantic similarity.

**Word2vec**: Word2vec [46] is a group of models that are used to produce word embeddings. These models are neural networks that are trained to learn high-quality word vectors from huge data sets with billions of words, and with millions of words in the vocabulary. Word2vec takes as its input a large corpus of text and produces a vector space, with each unique word in the corpus being assigned a corresponding vector in the space. Words that share common contexts in the corpus are located in close distance to one another in the space. Similarity between two vectors is defined as a cosine. To compute topic similarity, we use an R implementation of Word2vec to train a model for each dataset by making use of the documents' description. The similarity between two documents of a dataset is computed using the cosine similarity between the topic vectors that have been extracted after topic modeling.

**WordNet**: Making use of a lexical taxonomy (i.e. WordNet) to define distances between concepts is another commonly used technique. WordNet structure [37, 47, 48] is a large lexical database of English with words grouped into sets of synonyms (synsets). Nouns, verbs, adjectives and adverbs are grouped into synsets, each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. There are many different distance metrics that make use of the WordNet taxonomy to obtain semantic similarities. In this work, in order to calculate the similarity between two words, we use the Resnik distance [49], where the information content of a word is denoted as the logarithm of the probability of finding the word in a given corpus. This metric only considers the information content of the lowest common level in the hierarchy, i.e. the concept in the taxonomy which has the shortest distance from the concepts compared.

$$TS_{ij} = \frac{2 \times Match(\theta_i, \theta_j)}{|\theta_i| + |\theta_j|} \tag{5}$$

Given that each topic-$i$ is represented by a set of words $\theta_i$, in order to compute the topic similarity between two topics-$i, j$, at first we obtained the pairwise similarities between all the words contained in $\theta_i$, $\theta_j$ using the Resnik distance. To compute the overall matching score between the two topics, i.e. the pairwise Topic Similarity ($TS_{i,j}$), we used the matching average method (5) [50], which calculates the similarity between two topics $\theta_i$ and $\theta_j$ by dividing the sum of similarity values of all match candidates of both sets by the total number of set tokens. More specifically, the $Match(\theta_i, \theta_j)$ function of the equations counts

**Table 3** Statistical evaluation of the memory usage of the FBHC method compared to the BHC

| N | Mean Memory BHC (MB) | Mean Memory FBHC (MB) | DF | 95% Confidence interval of the differences | | Mean of the differences | t-value | *p*-value |
|---|---|---|---|---|---|---|---|---|
| 10000 | 786.941 | 27.908 | 4 | -759.220 | -758.845 | -759.033 | -11240.60 | 3.76e-16 |
| 15000 | 1743.466 | 33.561 | 4 | -1710.150 | -1709.660 | -1709.900 | -19389.60 | 4.24e-17 |
| 19000 | 2783.334 | 38.137 | 4 | -2745.480 | -2744.920 | -2745.200 | -27174.30 | 1.10e-17 |
| 25000 | 4800.901 | 44.289 | 4 | -4756.9721 | -4756.2516 | -4756.612 | -36659.18 | 3.32e-18 |
| 50000 | >19000.000 | 56.549 | - | - | - | - | - | - |

**Table 4** Statistical evaluation of the computational time of the FBHC method compared to the BHC

| N | Mean Time BHC (sec) | Mean Time FBHC (sec) | DF | 95% Confidence interval of the differences | | Mean of the differences | t-value | *p*-value |
|---|---|---|---|---|---|---|---|---|
| 10000 | 5807.156 | 92.382 | 4 | -7870.3 | -3559.24 | -5714.77 | -7.361 | 0.0018150 |
| 15000 | 22776.650 | 110.438 | 4 | -29521.5 | -15810.90 | -22666.20 | -9.178 | 0.0007820 |
| 19000 | 43940.550 | 121.944 | 4 | -59602.3 | -28034.90 | -43818.60 | -7.708 | 0.0015250 |
| 25000 | 89810.912 | 135.176 | 4 | -102093.2 | -77258.24 | -89675.73 | -20.050 | 0.0000365 |
| 50000 | >993600.000 | 183.000 | - | - | - | - | - | - |

the number of highly similar words of the two topics, i.e. the number of words that have Resnik similarity higher than the threshold 1. By employing (5), a $N_\theta \times N_\theta$ similarity matrix with the pairwise *TS* between all the $N_\theta$ topics was created.

### Results and discussion

We have performed a number of experiments to evaluate the effectiveness and the performance of our framework. Therefore, this subsection is divided into five parts: **a)** the comparison against baseline hierarchical clustering algorithms in terms of effectiveness is further discussed in "Effectiveness evaluation" section, **b)** the comparison against a baseline division hierarchical clustering algorithm in terms of memory usage and computational time is further discussed in "Performance statistical evaluation" section, **c)** the performance experiments of the proposed method running in the cloud is further discussed in "Performance testing in the cloud" section, **d)** the complexity analysis is presented in "Complexity analysis" section, and **e)** the overall proposed framework presented in "A new document clustering framework" section applied on the NYTimes dataset is further discussed in "Experimental results on the NYTimes dataset" section.

#### *Effectiveness evaluation*

The first set of experiments was focused on evaluating the quality of the proposed Frequency based hierarchical clustering (FBHC) method, by experimenting on the 10 first datasets described in Table 1. The effectiveness of the FBHC was examined using the external metrics *TS* and *FScore*, and comparing the results with baseline hierarchical clustering algorithms implemented in R language. Both division (*Diana*)[3] and agglomerative (*Average*, *Single*, *Complete* and *Ward*)[4] hierarchical clustering algorithms were used as baselines.

In Table 2, average *FScore* and *TS* values on the proposed algorithm and the baseline algorithms are presented. The best results achieved by an algorithm for each one of the datasets are highlighted as boldface, whereas the second highest results are presented in italics. The *FScore* was calculated taking into account the whole hierarchy structures that were created by the compared algorithms. For most of the datasets used in the experimental analysis, the highest FScore values are obtained by the proposed FBHC algorithm. For the LATimes, Oh10, Dmoz-Computers, Oh10 datasets, *ward* has a higher *FScore* and the *FBHC* algorithm comes second with a small difference, whereas for the Reviews dataset *Diana* comes first and *FBHC* comes second.

The average *TS* values were calculated on the final clusters that were set equal to the actual classes for each dataset. To obtain the final clusters of the dendrogram trees that were constructed using the baseline algorithms, the *cutree*[5] R function was used, whereas for the *FBHC* method, the branch breaking algorithm followed by the meta-clustering module were applied experimenting on different thresholds until the desired number of clusters were obtained.

To compute the average *TS* value for each dataset presented in Table 2, we extracted the major topics *i*, *j* of each cluster, we computed the $TS_{i,j}$ values using WordNet and (5) for all the clusters and we computed the average value. Instead of WordNet, we could also use Word2vec to calculate $TS_{i,j}$. However, the results in Table 2 would remain the same, as we present the difference $TS - TSActual$. Topic Similarity was calculated only for those clusters that contain more than 5 elements and include at least one major topic. Furthermore, *TS* was calculated for those datasets with actual classes characterized by major topics. The Tr31, Las2s, Tr12, Tr11, Tr45, Tr41, Oh10, Re0 and Re1 datasets do not follow the rule described in "Topic similarity" section, hence most of their clusters have *NA* values for the *TS* metric. The maximum value that *TS* may assume is 1, which indicates that each one of the clusters is characterized by a unique major topic. The *Single* method failed to create clusters with major topics, because it assigned most of the elements in one cluster with the rest of the clusters containing only one element each. Table 2 shows that the *FBHC* method usually produces *TS* values closer to the actual ones, compared to the other methods.

[3] https://stat.ethz.ch/R-manual/R-devel/library/cluster/html/diana.html
[4] https://stat.ethz.ch/R-manual/R-devel/library/stats/html/hclust.html
[5] https://www.rdocumentation.org/packages/dendextend/versions/1.9.0/topics/cutree

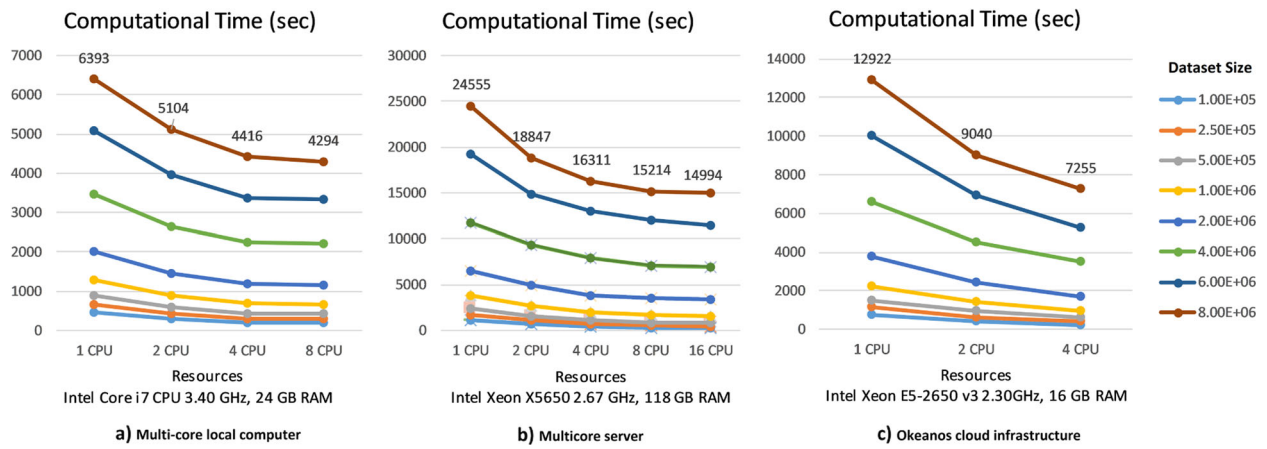


**Fig. 3** Computational time of the FBHC algorithm using subsets of the PubMed dataset running as a docker image on a local server and in a cloud infrastructure

### *Performance statistical evaluation*

The second set of experiments focused on evaluating the performance of the proposed clustering method, in terms of memory usage and computational time. The experiments were run with R on a computer with Intel Core i7 CPU 3.40 GHz with 8 cores and 24 GB RAM, using one core. The Frequency based Hierarchical Clustering (FBHC) algorithm was compared to the Baseline division Hierarchical Clustering algorithm (BHC) *Diana*. Figure 2 makes clear that using subsets of the NYTimes dataset of different sizes, the BHC algorithm has much higher memory demands. For the experiment with $N$ equal to 50,000 documents, the BHC algorithm was running for 11 days before it aborted with an "out of memory" error.

Additional results can be found in Tables 3 and 4, where the average memory usage and computational time for both FBHC and BHC algorithms, and the corresponding results of the statistical evaluation of the aforementioned values, for each subset size are analyzed. Statistical evaluation is performed to ensure the significant difference of the performance of our proposed algorithm and the baseline one. This was necessary because the memory usage and computational time of the baseline algorithms varied in each execution. By the use of the statistical tests the results can be generalized.

In the statistical test, we hypothesize that using the BHC algorithm instead of the FBHC one we can achieve better performance in terms of memory usage and computational time. To determine whether this hypothesis must be rejected, a statistical hypothesis test in name t-test is used (more details about the statistical method can be found in [51]). Tables 3 and 4 report the Degree of Freedom (DF) i.e. the amount of information in the data, the 95% Confidence interval of the differences, the average values of the differences, the t-test value and finally the probability value ($p$-value) which is used to make a decision about the statistical significance of the terms and model. According to the reported results, the $p$-values for all subsets never exceed $\alpha = 0.05$, which means the null hypothesis must be rejected and that the second hypothesis is supported.

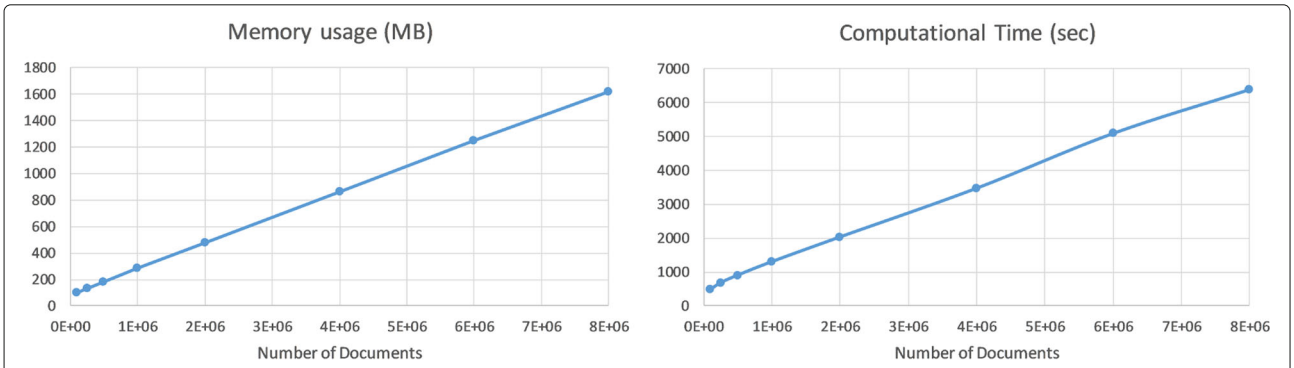


**Fig. 4** Memory usage and computational time of the FBHC algorithm using subsets of the PubMed dataset running on a local server

**Table 5** Computational complexity of the FBHC method

| N | FBHC (1 core) | FBHC (8 core) | Diana | HAC | SLINK | Birch |
|---|---|---|---|---|---|---|
| $10^5$ | 14.939 min | 03.224 min | 025.786 days | 004.502 days | 04.324 min | 01.175 min |
| $10^6$ | 21.591 min | 11.229 min | 298.746 days | 051.553 days | 07.637 h | 13.137 min |
| $10^7$ | 02.215 h | 01.489 h | 008.296 years | 001.450 years | 04.604 days | 02.212 h |
| $10^8$ | 20.912 h | 14.521 h | 083.080 years | 014.519 years | 46.473 days | 22.149 h |
| $10^9$ | 08.661 days | 06.031 days | 830.915 years | 145.217 years | 01.270 years | 09.229 days |

Tables 3 and 4 make clear that as the number of documents increases, the absolute value of the t-value of the statistical t-test for both memory usage and computational time increases, except for the first run (the subset with the smallest size) where the t-value of the memory usage was extremely high. This means that the difference between the performance of the two methods becomes more and more statistically important with the increment of the number of documents. For 25,000 documents, our method achieved over 99% reduction in both memory usage and computational time.

#### *Performance testing in the cloud*

The third set of experiments focused on evaluating the performance of the proposed clustering method in the cloud. In this round of experiments, we used the biggest dataset of our collection, the PubMed dataset, which contains 8 million documents. In order to test different cloud resource configurations, we built a docker image of the proposed clustering algorithm. The image is publicly available in the Docker hub (mariakotouza/fbc:pubmed) and includes all the subset datasets that were used in these experiments.

The docker image was run as a container on three different configurations: **a)** the local computer used in the second round of experiences, **b)** a server that had the following specifications: Ubuntu 18.04.3 LTS (kernel 4.15.0.58-generic), 2 x Intel Xeon X5650 @ 2.67 GHz with 16 cores and 118 GB RAM, and **c)** a configuration provided by the Okeanos national cloud infrastructure, with the following specifications: VMs with Ubuntu server 18.04, Intel(R) Xeon(R) CPU E5-2650 v3 2.30GHz with 4 cores and 16 GB RAM.

The scalability of our algorithm can be observed in Fig. 3, where different numbers of CPUs of the local computer, the cloud resources and the server were used for each subset. The X axis represents the number of CPUs, and the Y axis represents the execution time in seconds. The different lines in the figures correspond to a different subset size $N$. Comparing the three plots in the figure we observe that the computational time is highly affected by the available hardware. As for the memory usage, the demands for each core that is used are the same as those presented in Fig. 4 in the following sub-section.

#### *Complexity analysis*

Figure 4 shows the results for memory usage and computational time for different subsets of the dataset running on the local computer using one core. The figure makes clear that both metrics follow a linear model with the complexity being equal to $O(N)$, which means that the running time and the memory usage increase at most linearly with the size of the input $N$.

The same result can be obtained using a theoretical analysis to estimate the computational cost of analyzing datasets of different sizes. Table 5 shows the expected computational cost of various clustering algorithms applied to the corresponding datasets and executed on the local computer, using 1 core (the results of FBHC running on 8 cores are also shown). The hypothetical values were predicted after training a regression model using as $X$ and $Y$ variables the Number of documents ($N$) and the corresponding computational time ($T$) that were calculated using the PubMed dataset and are depicted on Fig. 4.

The computational complexity of our proposed algorithm was compared to the following state-of-the-art hierarchical clustering procedures:

(a) Diana [6] - The division hierarchical clustering algorithm which was used as a baseline in a previous set of experiments.
(b) HAC [7] - The agglomerative hierarchical clustering algorithm using different linkage criterion, that were utilized as baselines in the previous sets of experiments.
(c) SLINK [8] - An optimized implementation for hierarchical clustering using the single-linkage criterion with $O(N^2)$ time complexity.
(d) Birch [9] - A top-down incremental hierarchical clustering method where points are inserted greedily using the node statistics, which is ideal for large datasets.

[6] https://www.rdocumentation.org/packages/cluster/versions/2.1.0/topics/diana
[7] https://www.rdocumentation.org/packages/cluster/versions/2.1.0/topics/agnes
[8] https://github.com/battuzz/slink
[9] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html

**Table 6** Average values of the internal clustering evaluation metrics per tree level after the application of the binary tree construction algorithm on the NYTimes dataset

| Level | #C | I | IS | HS | BS | TS |
|---|---|---|---|---|---|---|
| 0 | 1 | 0.000 | 0.000 | 0.320 | 58.072 | NA |
| 1 | 2 | 0.000 | 2.500 | 0.238 | 69.462 | 1.000 |
| 2 | 4 | 25.000 | 30.000 | 0.152 | 79.915 | 1.000 |
| 3 | 7 | 36.429 | 47.857 | 0.121 | 84.922 | 0.814 |
| 4 | 13 | 48.462 | 62.692 | 0.084 | 90.207 | 0.539 |
| 5 | 21 | 55.238 | 71.667 | 0.068 | 92.481 | 0.524 |
| 6 | 32 | 58.125 | 75.938 | 0.060 | 93.833 | 0.468 |
| 7 | 49 | 61.837 | 80.204 | 0.048 | 95.010 | 0.422 |
| 8 | 72 | 66.181 | 83.819 | 0.041 | 95.926 | 0.393 |
| 9 | 101 | 68.911 | 86.584 | 0.034 | 96.594 | 0.367 |
| 10 | 138 | 71.377 | 88.659 | 0.029 | 97.071 | 0.350 |
| 11 | 184 | 72.962 | 90.272 | 0.025 | 97.417 | 0.335 |
| 12 | 235 | 73.617 | 91.447 | 0.022 | 97.626 | 0.314 |
| 13 | 298 | 74.295 | 92.416 | 0.020 | 97.814 | 0.298 |
| 14 | 374 | 75.013 | 93.583 | 0.018 | 97.993 | 0.286 |
| 15 | 460 | 75.304 | 94.543 | 0.016 | 98.127 | 0.278 |
| 16 | 561 | 75.383 | 95.463 | 0.015 | 98.240 | 0.275 |
| 17 | 681 | 75.206 | 96.300 | 0.013 | 98.335 | 0.263 |
| 18 | 827 | 75.224 | 97.056 | 0.011 | 98.432 | 0.256 |
| 19 | 997 | 74.493 | 97.723 | 0.010 | 98.484 | 0.249 |
| 20 | 1206 | 73.669 | 98.184 | 0.008 | 98.521 | 0.254 |
| 21 | 1462 | 72.302 | 98.683 | 0.005 | 98.519 | 0.269 |
| 22 | 1753 | 71.714 | 99.395 | 0.001 | 98.549 | 0.279 |
| 23 | 1965 | 74.221 | 100.000 | 0.000 | 98.711 | 0.287 |

**Table 7** Average values of the internal clustering evaluation metrics per tree level after the application of the branch breaking algorithm on the NYTimes dataset

| Level | #C | I | IS | HS | BS | TS |
|---|---|---|---|---|---|---|
| 0 | 1 | 0.000 | 0.000 | 0.32 | 58.072 | NA |
| 1 | 2 | 0.000 | 2.500 | 0.238 | 69.462 | NA |
| 2 | 4 | 25.000 | 30.000 | 0.152 | 79.915 | NA |
| 3 | 7 | 36.429 | 47.857 | 0.121 | 84.922 | NA |
| 4 | 13 | 48.462 | 62.692 | 0.084 | 90.207 | 0.184 |
| 5 | 21 | 55.238 | 71.667 | 0.068 | 92.481 | 0.196 |
| 6 | 30 | 55.167 | 74.000 | 0.065 | 93.397 | 0.196 |
| 7 | 40 | 57.875 | 77.250 | 0.057 | 94.202 | 0.204 |
| 8 | 46 | 61.957 | 79.565 | 0.051 | 94.636 | 0.204 |
| 9 | 47 | 60.638 | 78.830 | 0.052 | 94.432 | 0.204 |
| 10 | 48 | 59.375 | 78.229 | 0.055 | 94.223 | 0.204 |
| 11 | 49 | 58.163 | 77.755 | 0.056 | 94.071 | 0.204 |
| 12 | 50 | 57.000 | 77.400 | 0.058 | 93.913 | 0.204 |
| 13 | 51 | 55.882 | 77.157 | 0.059 | 93.794 | 0.204 |
| 14 | 52 | 54.808 | 77.019 | 0.061 | 93.692 | 0.204 |
| 15 | 53 | 53.774 | 76.981 | 0.062 | 93.619 | 0.204 |
| 16 | 54 | 52.778 | 77.037 | 0.062 | 93.575 | 0.204 |
| 17 | 55 | 51.818 | 77.182 | 0.063 | 93.554 | 0.211 |
| 18 | 56 | 50.893 | 77.411 | 0.063 | 93.554 | 0.211 |
| 19 | 57 | 50.000 | 77.719 | 0.062 | 93.575 | 0.270 |
| 20 | 58 | 49.138 | 78.103 | 0.061 | 93.596 | 0.326 |

The values presented on Table 5 are hypothetical, as the ones for the FBHC algorithm. The results prove once again that the FBHC algorithm outperforms the rest of the methods in terms of computational time. The second-best algorithm that scales for large number of documents is the Birch algorithm. However, the algorithm is not scalable in terms of memory usage, as we were not able to run it on the local computer for datasets consisting of more than 80,000 documents due to memory limitation problems.

#### *Experimental results on the NYTimes dataset*

The last round of experiments include the application of our proposed hierarchical clustering framework on the NYTimes dataset. Using the binary tree construction algorithm of type $similarity_{algo}$, a binary tree with 23 levels and 1965 leaf clusters was constructed. Table 6 shows that the Identity and Similarity metrics began with 0 values at the root of the tree, whereas the Entropy metric began with 0.32. These values improved when descending down the different tree levels, until at the leaf level the Similarity value was equal to 100% while the Entropy was equal to 0.

During the second phase, the tree was pruned by applying the branch breaking algorithm using the percentage of 0.5% as threshold for all the comparisons of the metrics. The final tree consists of 20 levels and 58 leaf clusters. The average values of each level's metrics using the FM and the FSM matrices are summarized in Table 7. The table shows that the identity value increased towards the leaves of the tree. Notably, when groups of similar bins are used instead of the bins themselves, the similarity value (IS) was a little higher as expected. The values of the Topic Similarity (TS) metric, which is discussed in the following sub-section, are also included in the table.

During the meta-clustering phase of the procedure, a graph is constructed using all the leaf clusters that have been formed after the branch breaking algorithm. The hierarchy structure of the clusters are presented in Fig. 5, where similar clusters are depicted using characteristic colors. The graph is clustered using a threshold equal to 10%, removing all the edges that were connecting the most dissimilar clusters. Figure 6 presents the fully connected

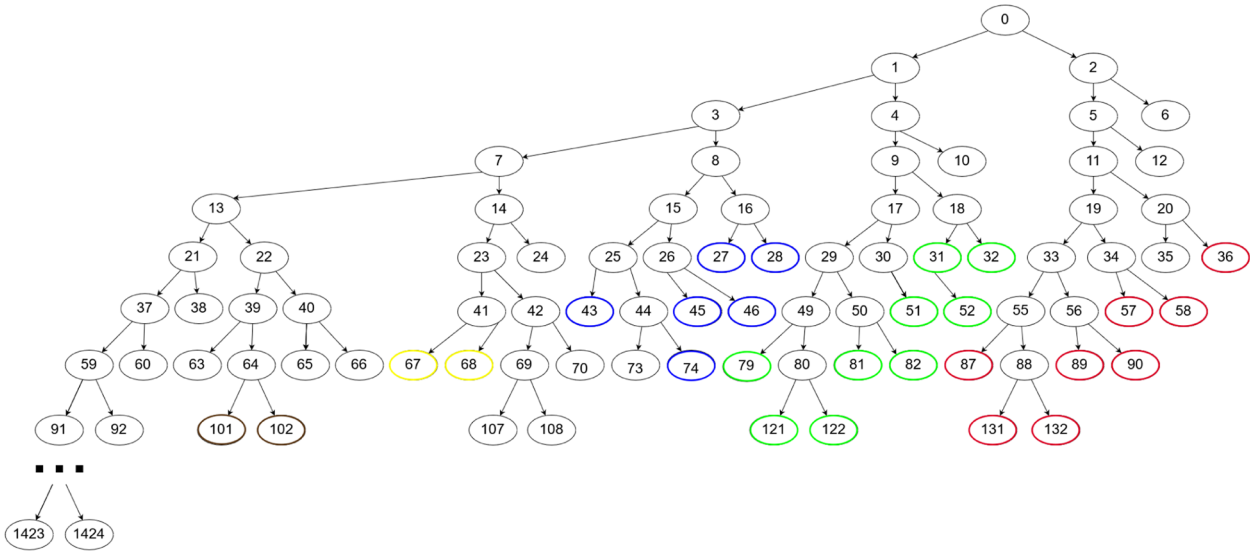


**Fig. 5** The binary tree of the NYTimes dataset. The most similar leaf clusters that was discovered during the graph construction module are presented using the same colors

and the clustered graphs. Evidently, most of the big clusters do not have similarities with other clusters, but some smaller clusters like 12, 35, 36, 57, 58, 89, 90, 131, 132 could be merged with the cluster 87 due to the high connectivity that is observed. The aforementioned clusters that were given as an example for merging are presented on Fig. 5 using red color.

## Conclusion

In this paper, we presented a new scalable multi-metric hierarchical clustering framework for document clustering. The input documents are preprocessed and transformed into feature vectors using topic modeling, and afterword they are discretized forming sequences of characters. The clustering method is composed of three distinct phases: the binary tree construction algorithm, the branch breaking algorithm, and a meta-clustering module for generating graphical representations of the output. The metrics that are used to form the clusters include Identity, Similarity, Entropy and Bin Similarity. The clustering method exhibits a high degree of parallelism and several sub-processes can be distributed in multiple CPUs

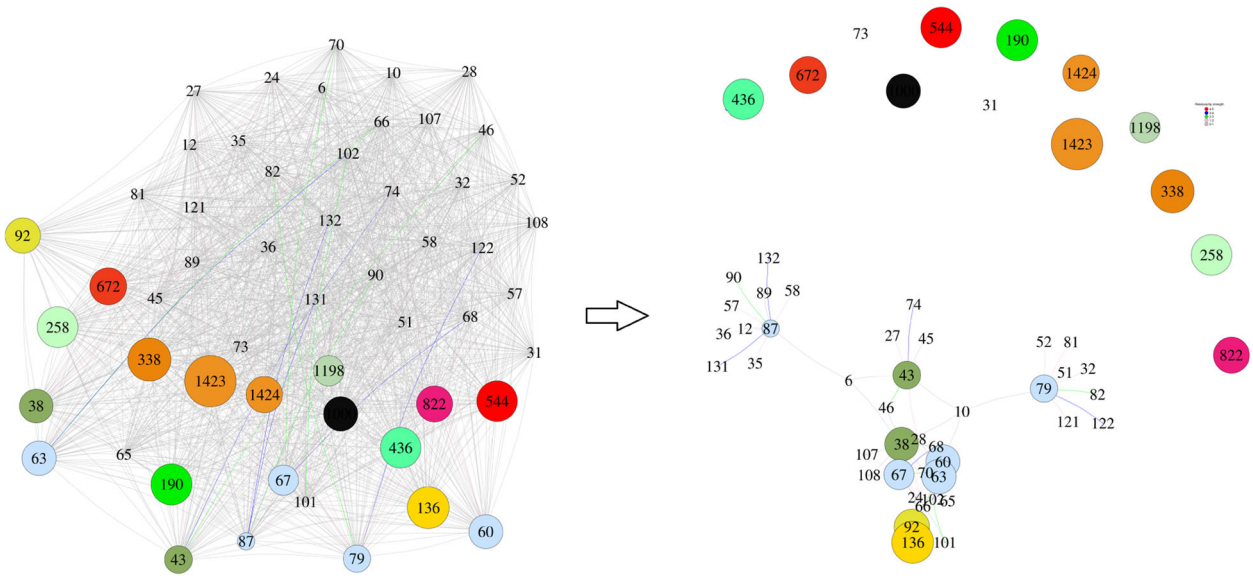


**Fig. 6** The initial graph and the clustered graph of the NYTimes dataset

to speedup the whole process. It is also dockerized, to enable execution in almost any configuration in the cloud.

Using this frequency-based approach to perform hierarchical document clustering, many limitations on computational time and memory usage, as the number of documents increases, can be overcome. Our algorithm has increased scalability compared to existing hierarchical clustering algorithms, because it uses frequency tables to form the clusters instead of making pairwise comparisons between all the elements of the dataset. A series of efficiency and performance evaluation experiments have shown considerable reduction in both execution times and memory requirements over a wide variety of publicly available document sets and of cloud infrastructure.

A limitation of our proposed method may be the information loss that comes from the data discretization module, but it is up to the users to select the number of bins B in such a way that the amount of information that is considered by the model is sufficient, depending on the problem. Considering the effectiveness of the proposed method in the cloud, Future work involves further parallelization of the clustering algorithm in order to optimize the use of allocated resources in the cloud, including GPU usage. Moreover, the proposed framework could be extended to handle real time applications running in the cloud that demand new document categorization. This could be done by implementing a decision-making algorithm that exploits the hierarchy of the clusters to perform new document categorization into the existing clusters.

#### Abbreviations

LDA: Latent Dirichlet allocation; FBHC: Frequency based hierarchical clustering; BHC: Baseline hierarchical clustering

#### Acknowledgements

We would like to thank Thanasis Vergoulis, Konstantinos Zagganas, and Theodore Dalamagas for providing us access to SCHeMa (https://schema.imsi.athenarc.gr/), the Cloud platform on which some of our experiments presented in "Performance testing in the cloud" subsection were conducted.

#### Authors' contributions

All authors read and approved the final manuscript.

#### Funding

This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE – INNOVATE (project code:T1EDK-03464)

#### Availability of data and materials

The datasets supporting the conclusions of this article are available in the LABIC repository, http://sites.labic.icmc.usp.br/text_collections/, and the UIC Machine Learning repository, https://archive.ics.uci.edu/ml/datasets/Bag+of+Words. The source code of the proposed hierarchical clustering method is publicly available on github, https://github.com/mariakotouza/FBHC, whereas the docker image of the method, which includes all the tested subsets of the Pubmed dataset, is publicly available on Docker hub with the name mariakotouza/FBHC:fbc, https://cloud.docker.com/u/mariakotouza/repository/docker/mariakotouza/fbc. Finally, the docker image of the whole clustering framework is available on Docker hub with the name mariakotouza/fbc:framework.

#### Competing interests

The authors declare that they have no competing interests.

#### Author details

[1]Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece. [2]Institute of Applied Biosciences, Centre for Research and Technology Hellas, 57001 Thessaloniki, Greece. [3]Dept of Molecular Medicine and Surgery, Karolinska Institutet, Stockholm, Sweden.

Received: 15 March 2019 Accepted: 30 December 2019
Published online: 17 January 2020

#### References

1. Jaiswal A, Janwe N (2011) Hierarchical document clustering: a review. In: 2nd National Conference on Information and Communication Technology (NCICT) 2011 Proceedings published in International Journal of Computer Applications®(IJCA). pp 37–41
2. Roul RK, Asthana SR, Sahay SK (2015) Automated document indexing via intelligent hierarchical clustering: A novel approach. In: 2014 International Conference on High Performance Computing and Applications, ICHPCA 2014. https://doi.org/10.1109/ICHPCA.2014.7045347
3. Zhao Y, Karypis G (2002) Evaluation of hierarchical clustering algorithms for document datasets. In: Proceedings of the eleventh international conference on Information and knowledge management - CIKM '02. p 515. https://doi.org/10.1145/584792.584877
4. Shah N, Mahajan S (2012) Document Clustering: A Detailed Review. Int J Appl Inf Syst (IJAIS) 4(5):30–38. URL https://doi.org/10.5120/8202-1598
5. Bhardwaj S, Jain L, Jain S (2010) Cloud computing: A study of infrastructure as a service (iaas). Int J Eng Inf Technol 2(1):60–63
6. Kotouza M, Vavliakis K, Psomopoulos F, Mitkas P (2018) A hierarchical multi-metric framework for item clustering. In: 2018 IEEE/ACM 5th International Conference on Big Data Computing Applications and Technologies (BDCAT). IEEE. https://doi.org/10.1109/bdcat.2018.00031
7. Chen CL, Tseng FSC, Liang T (2010) An integration of WordNet and fuzzy association rule mining for multi-label document clustering. Data Knowl Eng 69(11):1208–1226. https://doi.org/10.1016/j.datak.2010.08.003
8. Brants T, Chen F, Tsochantaridis I (2002) Topic-based document segmentation with probabilistic latent semantic analysis. In: Proceedings of the eleventh international conference on Information and knowledge management - CIKM '02. p 211. https://doi.org/10.1145/584792.584829
9. Newman D, Asuncion A, Smyth P, Welling M (2009) Distributed algorithms for topic models. J Mach Learn Res 10(Aug):1801–1828
10. Wang Y, Bai H, Stanton M, Chen W-Y, Chang EY (2009) Plda: Parallel latent dirichlet allocation for large-scale applications. In: International Conference on Algorithmic Applications in Management. Springer. pp 301–314. https://doi.org/10.1007/978-3-642-02158-9_26
11. Liu Z, Zhang Y, Chang EY, Sun M (2011) Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing. ACM Trans Intell Syst Technol (TIST) 2(3):26
12. Vorontsov K, Frei O, Apishev M, Romov P, Dudarenko M (2015) Bigartm: Open source library for regularized multimodal topic modeling of large collections. In: Communications in Computer and Information Science. Springer International Publishing. pp 370–381. https://doi.org/10.1007/978-3-319-26123-2_36
13. Moody CE (2016) Mixing dirichlet topic models and word embeddings to make lda2v ec. arXiv preprint arXiv:1605.02019. https://arxiv.org/abs/1605.02019
14. Ahmadi P, Gholampour I, Tabandeh M (2018) Cluster-based sparse topical coding for topic mining and document clustering. Adv Data Anal Classif 12(3):537–558. https://doi.org/10.1007/s11634-017-0280-3
15. Rafi M, Shaikh MS, Farooq A (2010) Document Clustering based on Topic Maps. Int J Comput Appl 12(1):32–36
16. Ma Y, Wang Y, Jin B (2014) A three-phase approach to document clustering based on topic significance degree. Expert Syst Appl 41(18):8203–8210. https://doi.org/10.1016/j.eswa.2014.07.014
17. Lin Q, Jungang X (2013) A Chinese Word Clustering Method Using Latent Dirichlet Allocation and K-means. In: Proceedings of the 2nd International Conference on Advances in Computer Science and Engineering. Atlantis Press. URL https://doi.org/10.2991/cse.2013.60
18. Onan A, Bulut H, Korukoglu S (2017) An improved ant algorithm with LDA-based representation for text document clustering. J Inf Sci 43(2):275–292. https://doi.org/10.1177/0165551516638784

19. Yau CK, Porter A, Newman N, Suominen A (2014) Clustering scientific documents with topic modeling. Scientometrics 100(3):767–786. https://doi.org/10.1007/s11192-014-1321-8
20. Xu J, Zhou S, Qiu L, Liu S, Li P (2014) A Document Clustering Algorithm Based on Semi-constrained Hierarchical Latent Dirichlet Allocation. Springer International Publishing. https://doi.org/10.1007/978-3-319-12096-6_5
21. Steinbach M, Karypis G, Kumar V, et al. (2000) A comparison of document clustering techniques. In: KDD Workshop on Text Mining, vol 400. arXiv, Boston. pp 525–526. https://dl.acm.org/doi/10.1145/233269.233324
22. Chen CL, Tseng FSC, Liang T (2010) Mining fuzzy frequent itemsets for hierarchical document clustering. Inf Process Manag 46(2):193–211. https://doi.org/10.1016/j.ipm.2009.09.009
23. Manning CD (2009) Intro to Information Retrieval. Inf Retrieval:1–18. https://doi.org/10.1109/LPT.2009.2020494
24. Sibson R (1973) Slink: an optimally efficient algorithm for the single-link cluster method. Comput J 16(1):30–34
25. Defays D (1977) An efficient algorithm for a complete link method. Comput J 20(4):364–366
26. Guha S, Rastogi R, Shim K (2000) Rock: A robust clustering algorithm for categorical attributes. Inf Syst 25(5):364–366
27. Karypis G, Han E-HS, Kumar V (1999) Chameleon: Hierarchical clustering using dynamic modeling. Computer 32(8):68–75. https://doi.org/10.1109/2.781637
28. Zhang T, Ramakrishnan R, Livny M (1996) Birch: an efficient data clustering method for very large databases. In: ACM Sigmod Record, vol 25. ACM. pp 103–114. https://dl.acm.org/doi/10.1145/233269.233324
29. Fichtenberger H, Gillé M, Schmidt M, Schwiegelshohn C, Sohler C (2013) Bico: Birch meets coresets for k-means clustering. In: Lecture Notes in Computer Science. Springer Berlin Heidelberg. pp 481–492. https://doi.org/10.1007/978-3-642-40450-4_41
30. Agarwal A, Roul RK (2018) A novel hierarchical clustering algorithm for online resources. In: Recent Findings in Intelligent Computing Techniques. Springer. pp 467–476. https://link.springer.com/chapter/10.1007/978-981-10-8636-6_49
31. Kobren A, Monath N, Krishnamurthy A, McCallum A (2017) A hierarchical algorithm for extreme clustering. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM. pp 255–264. https://dl.acm.org/doi/10.1145/3097983.3098079
32. Xie W-B, Lee Y-L, Wang C, Chen D-B, Zhou T (2019) Hierarchical clustering supported by reciprocal nearest neighbors. arXiv preprint arXiv:1907.04915. https://arxiv.org/abs/1907.04915
33. Charikar M, Chatziafratis V, Niazadeh R (2019) Hierarchical clustering better than average-linkage. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics. pp 2291–2304. https://doi.org/10.1137/1.9781611975482.139
34. Cohen-Addad V, Kanade V, Mallmann-Trenn F, Mathieu C (2019) Hierarchical clustering: Objective functions and algorithms. J ACM (JACM) 66(4):26
35. Dasgupta S (2015) A cost function for similarity-based hierarchical clustering. arXiv preprint arXiv:1510.05043. https://arxiv.org/abs/1510.05043
36. Tsarouchis S-F, Kotouza MT, Psomopoulos FE, Mitkas PA (2018) A multi-metric algorithm for hierarchical clustering of same-length protein sequences. In: IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer International Publishing. pp 189–199. https://doi.org/10.1007/978-3-319-92016-0_18
37. Oram P (2001) WordNet: An Electronic Lexical Database(Fellbaum C, ed.). MA: MIT Press, 1998, Cambridge. 423. Applied Psycholinguistics, 22(1), 131-134
38. Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3(Jan):993–1022
39. Newman D, Baldwin T, Cavedon L, Huang E, Karimi S, Martinez D, Scholer F, Zobel J (2010) Visualizing search results and document collections using topic maps. Web Semant Sci Serv Agents World Wide Web 8(2-3):169–175
40. Vorontsov K, Potapenko A (2015) Additive regularization of topic models. Mach Learn 101(1-3):303–323
41. Pahl C, Lee B (2015) Containers and clusters for edge cloud architectures–a technology review. In: 2015 3rd International Conference on Future Internet of Things and Cloud. IEEE. https://doi.org/10.1109/ficloud.2015.35
42. Merkel D (2014) Docker: lightweight linux containers for consistent development and deployment. Linux J 2014(239):2
43. Rossi RG, Marcacini RM, Rezende SO (2013) Benchmarking text collections for classification and clustering tasks. Institute of Mathematics and Computer Sciences, University of Sao Paulo. http://sites.labic.icmc.usp.br/ragero/docs/TR_395.pdf
44. Dheeru D, Karra Taniskidou E (2017) UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. Accessed Jan 2019
45. Larsen B, Aone C (1999) Fast and effective text mining using linear-time document clustering. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining – KDD '99. ACM Press. https://doi.org/10.1145/312129.312186
46. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. https://arxiv.org/abs/1301.3781
47. Feinerer I, Hornik K (2017) Wordnet: WordNet Interface. R package version 0.1-14. https://CRAN.R-project.org/package=wordnet. Accessed Jan 2019
48. Wallace M (2007) Jawbone Java WordNet API. http://mfwallace.googlepages.com/jawbone. Accessed Jan 2019
49. Resnik P (1995) Using Information Content to Evaluate Semantic Similarity in a Taxonomy 1. https://doi.org/10.1.1.55.5277
50. Lin D (1998) Automatic retrieval and clustering of similar words. In: COLING 1998 Volume 2: The 17th International Conference on Computational Linguistics. Association for Computational Linguistics. https://doi.org/10.3115/980432.980696
51. Kyun KT (2015) T test as a parametric statistic. Korean J Anesthesiol 68(6):540–546. https://doi.org/10.4097/kjae.2015.68.6.540


## Publisher's Note