2018 IEEE/ACM 5th International Conference on Big Data Computing Applications and Technologies (BDCAT)

A hierarchical multi-metric framework for item clustering

Maria Th. Kotouza
Electrical and Computer Engineering
Aristotle University of Thessaloniki
Thessaloniki 54124, Greece
maria.kotouza@issel.ee.auth.gr

Konstantinos N. Vavliakis
Electrical and Computer Engineering
Aristotle University of Thessaloniki
Thessaloniki 54124, Greece
kvavliak@issel.ee.auth.gr

Fotis E. Psomopoulos
Institute of Applied Biosciences,
Centre for Research and Technology Hellas,
Thessaloniki, 57001, Greece
fpsom@certh.gr

Pericles A. Mitkas
Electrical and Computer Engineering
Aristotle University of Thessaloniki
Thessaloniki 54124, Greece
mitkas@auth.gr

***Abstract*—Item clustering is commonly used for dimensionality reduction, uncovering item similarities and connections, gaining insights of the market structure and recommendations. Hierarchical clustering methods produce a hierarchy structure along with the clusters that can be useful for managing item categories and sub-categories, dealing with indirect competition and new item categorization as well. Nevertheless, baseline hierarchical clustering algorithms have high computational cost and memory usage. In this paper we propose an innovative scalable hierarchical clustering framework, which overcomes these limitations. Our work consists of a binary tree construction algorithm that creates a hierarchy of the items using three metrics, a) Identity, b) Similarity and c) Entropy, as well as a branch breaking algorithm which composes the final clusters by applying thresholds to each branch of the tree. The proposed framework is evaluated on the popular MovieLens 20M dataset achieving significant reduction in both memory consumption and computational time over a baseline hierarchical clustering algorithm.**

Keywords— Hierarchical item clustering, topic modeling, sequence similarity, sequence identity

I. INTRODUCTION

Modern web applications require an interdisciplinary set of techniques for efficient and accurate filtering, as well as ranking and personalizing vast amounts of information in real-time fashion. Nowadays users browse billions of movies, songs, videos, social media entities and e-commerce products on a daily basis. In order to ensure timely and proper data processing that will enable real time operation of smart applications, a commonly applied technique is data reduction in the form of grouping similar items into clusters.

Item clustering is frequently used in recommender systems that assist users discovering items of interest. The most popular approach used in the field of recommendation systems is collaborative filtering (CF) [1], a technique that looks for patterns in the overall user activity to produce recommendations. One of the main problems CF faces is sparsity, since the overall number of available items is usually enormous, but each user is interested in only a very small subset of them. In this case item and/or user clustering is implemented for reducing complexity, thus allowing real time predictions, and increasing their accuracy by excluding information not relevant to the question at hand.

Another area of interest for item clustering is the process of decision making in business. Analysing inter-product similarities is crucial for understanding product-market structures and competitive market relationships [2]. Especially in marketing, item categorization and clustering has multifarious applications; developing new products, shelf placement optimization for retail products, product replacement in case of out of stocks, and relationship analysis among products are only a few examples [3].

Hierarchy is of paramount importance for a diverse number of item clustering applications. Hierarchy provides insights to retailers for efficiently managing categories and sub-categories. Moreover, for long term strategic decisions, management must take into account not only current and direct competition from similar entities (e.g. similar brands), but indirect competition as well. Indirect competition comes from entity variants (i.e. entities from different levels of the hierarchy) and may be more substantial threats in the future. In that case hierarchical clustering has been found to be of considerable help [2]. Additionally, the hierarchy extracted by these methods can be further considered as a decision tree that can assist in the classification of new items. However, hierarchical clustering methods require high computational power and memory usage as they are based on the formulation of high dimensional distance matrices, used for pairwise comparisons between all the available data points.

Our research focuses on implementing a new scalable multi-metric algorithm for hierarchical item clustering. Our innovation lies in the fact that, instead of performing pairwise comparisons between all items of the dataset, we build a low dimensional frequency matrix for the root cluster which is split recursively as one goes down the hierarchy. Thus, our proposed framework requires both less computational power and memory. The input of our algorithm consists of items represented by a mixture of topics that derive from topic modeling item's contextual information.

The rest of the paper is organized as follows; Section II discusses related work, while the proposed integrated framework for item clustering is analysed in Section III. In Section IV our innovative hierarchical clustering algorithm is described, whereas Section V contains the experimental

978-1-5386-5502-3/18/$31.00 ©2018 IEEE
DOI 10.1109/BDCAT.2018.00031

results and the clustering evaluation. Finally, conclusions and future work are highlighted in Section VI.

II. Background and Related Work

Recently, item clustering has attracted increased interest mainly due to the WWW explosion, as item clustering is a core technology for recommendation and prediction engines used in online recommender systems. Fremal et al. [4], in an attempt to improve rating prediction results of a CF based recommender system, proposed a clustering approach that used weighting strategies on the items' genres. On the other hand, Das et al. [5] introduced a clustering based recommender system, where voting systems combined opinions of different users for improving recommendations to new users. Furthermore, Wang et al. [6] adopt item clustering for obtaining latent factors in order to apply matrix factorization as a next step.

Business analytics is another common field of application for item clustering methods. Srivastave et al. [2] proposed an iterative hierarchical clustering procedure that is suitable for numerous marketing applications. In the work of Yang et al. [7] item clustering was employed for industry categorization, where Doc2vec was employed for document embedding and Ward's hierarchical clustering algorithm was applied to group similar firms together. Finally, in the work of Hol et al. [3] a genetic algorithm for basket data analysis and item clustering was proposed.

Hierarchical clustering algorithms [8] compose a tree of clusters that comprises a richer data structure than flat algorithms' output. In addition, they do not require users to define the number of clusters. Hierarchical clustering algorithms 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 with bottom-up methods that make clustering decisions based on local patterns.

Nevertheless, the complexity of the naïve hierarchical clustering algorithm is $O(N^3)$ as for every decision needs to be taken, an exhaustive scan of the NxN similarity matrix is necessary. Other more efficient algorithms can reduce the complexity to $O(N^2 logN)$ or even $O(N^2)$ but the creation of the NxN similarity matrix is necessary, hence memory requirement demands become extremely high.

Thereupon, although there is a wide range of hierarchical item clustering algorithms, they have substantial computational and memory requirements. In this paper we introduce an innovative top-down hierarchical clustering algorithm which instead of computing pairwise comparisons between data points, it is based on the construction of low dimensional frequency matrices, thus is scalable and has low memory and computational requirements.

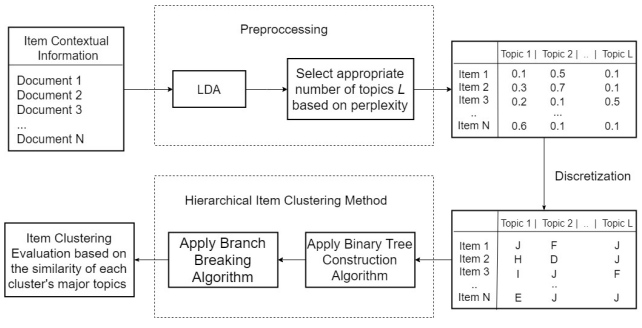

Fig. 1. A framework for hierarchical item clustering.

III. A New Item Clustering Framework

In this section we propose an efficient framework (Fig. 1) for hierarchical item clustering based on item topics modeling. The proposed framework is composed of 1) the data preprocessing module, which performs topic modeling using item's contextual data, 2) the data discretization module, 3) the hierarchical clustering algorithm, and 4) the clustering evaluation module that is based on semantic similarity between the major topics of each cluster.

A. Data Preprocessing Module

The preprocessing module employs topic modelling by the use of Latent Dirichlet Allocation (LDA) for extracting semantic information [9], [10]. 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 θ as a focused multinomial distribution over terms $P(w|\theta)$. LDA builds a set of N_Θ thematic topics, each expressed with a set of N_W terms, utilizing terms that tend to co-occur in a given set of documents. Parameters N_Θ and N_W are user specified and can be used to adjust the degree of specialization of the latent topics.

The topic-term distribution $P(\theta|d)$ and the document-term distribution $P(w|\theta)$ are estimated from an unlabeled corpus of documents D using Dirichlet priors. The Gibbs sampler [11] iterates multiple times over each term w_i in document d and samples a new topic j, according to (1), until the LDA model parameters converge. In (1), (2), (3), $C^{W\Theta}$ maintains a count of all topic–term assignments, $C^{D\Theta}$ counts the document-topic assignments, W is the set of all available terms, θ_{-i} are all topic-term and document-topic assignments except the current assignment θ_i for term w_i, and α and β are the hyperparameters of the Dirichlet priors. The posterior probabilities of (1) are estimated using (2) and (3).

$$P(\theta_i = j | w_i, d_i, \theta_{-i}) \propto P(w_i|\theta_i = j) \times P(\theta_i = j|d_i) \quad (1)$$

$$P(w_i|\theta_i = j) = \frac{C^{W\Theta}_{w_i j} + \beta}{\sum_w C^{W\Theta}_{Wj} + W\beta} \quad (2)$$

$$P(\theta_i = j|d_i) = \frac{C^{D\Theta}_{d_i j} + \alpha}{\sum_\theta C^{D\Theta}_{d_i \theta} + \Theta a} \quad (3)$$

The most common way to evaluate a probabilistic model is to measure the log-likelihood of a held-out test set, thus we use perplexity (4) which is utilized by convention in language modeling [9]. A lower perplexity score indicates better generalization performance.

$$perplexity = exp\left\{-\frac{\sum_{i=1}^{N}\log p(w_i)}{\sum_{i=1}^{N} d_i}\right\} \quad (4)$$

B. Data Discretization Module

The input vectors of the clustering algorithm, in our case the mixture of topics $P(\theta|d)$, calculated by the topic modeling process are discretized into partitions of B lengths by assigning each value into a bin based on the interval where it belongs to. B is selected based on the amount of information we want to be considered by the model.

C. Theoretical basis for the clustering algorithm

1) Frequency Matrix

The hierarchical clustering algorithm employs a frequency 2-dimensional matrix, where the number of rows equals to the number of bins (B) selected for the discretization of the topic mixture, and number of columns equals to the number of topics (N_θ) of the sequences provided as input. Each element (i,j) of the matrix corresponds to the number of times bin i is present in topic j for all sequences. The count matrix (CM) contains the absolute values, whereas the frequency matrix FM ($FM = CM / N$) contains the corresponding frequencies. In addition to FM, a frequency similarity matrix (FSM) is constructed using the same approach, but instead of the B bins, groups of similar bins are used under given schemes.

2) Bin Similarity

Bin similarity (8) is defined as the average column sum of the Bin Similarity Matrix (7), which is a weighted version of the FM. To compute column j of BSM, the non-zero elements of FM are multiplied by a weight factor that derives from the similarity of the bins that participate in topic j. $BSSM$ is a square matrix with order B, where each element (i,j) indicates how similar bins i, j are (5).

$$BSSM_{ij} = 1 - \frac{|i-j|}{B} \quad (5)$$

$$mask = \begin{cases} 1, & if FM_{ij} \neq 0 \\ 0, & if\ FM_{ij} = 0 \end{cases} \quad (6)$$

$$BSM = (FM \times BSSM) \cdot mask \quad (7)$$

where $\times$ denotes matrix multiplication and $\cdot$ denotes element-wise multiplication.

$$BS = \sum_{j=1}^{N_\theta} BSM_j / N_\theta \quad (8)$$

3) Identity

Identity is a similarity metric that is computed for each cluster based on the corresponding frequency matrix. This metric, indicates how compact the cluster is and it is expressed as the percentage of sequences contained in the cluster with an exact alignment. This means that the identity of column j (id_j) of the FM is equal to 100% when all sequences belong to same bin of topic j. Otherwise, the identity of column j is equal to zero. The overall identity (9) of the cluster is the average value of the columns' identity.

$$I = \sum_{j=1}^{N_\theta} id_j / N_\theta \quad (9)$$

$$id_j = \begin{cases} 100\%, & if\ max(FM_{ij}) = 100\% \\ 0\%, & if\ max(FM_{ij}) \neq 100\% \end{cases} \quad (10)$$

4) Entropy

The entropy quantifies the expected value of the information contained in a vector. It is computed for each column of the frequency matrix and represents the diversity of the column. The Shannon entropy equation [12] provides a way to encode a string of symbols, based on the frequency of the symbols. It is depicted in (11) where p_i is the probability of bin i showing up in the topic j. The entropy is ranging from 0 to Inf. A lower entropy value indicates a more homogeneous column.

$$H_j = -\sum p_i \log(p_i) \quad (11)$$

IV. A NEW ITEM CLUSTERING FRAMEWORK

In this section we propose a novel clustering algorithm that consists of two phases: 1) the first phase includes the construction of a top down binary tree by consecutively dividing the frequency matrix into two sub-matrices until only unique item vectors remain at the leaf-level, and 2) the second phase is a branch breaking algorithm, where each branch of the tree is cut at an appropriate level using thresholds for the metrics.

A. Binary Tree Construction

The first phase consists of a top down hierarchical clustering method (Algorithm 1). At the beginning of the process, it is assumed that all N items belong to a single cluster (C_0), which is consequently split recursively while moving along the different levels of the tree. Ultimately, the constructed output of the clustering process is presented as a binary tree. The process applied to each cluster (C_i) while constructing the tree can be formally described in the following steps:

Step 1: Create frequency and frequency-similarity based matrices (FM_i, FSM_i).

Step 2: Compute Identity, Entropy and Bin Similarity of the matrices (I_i, IS_i, H_i, HS_i, BS_i) applying (9), (11), (8) on the FM_i and the FSM_i respectively. The identity metric computed on the FSM is named as 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. if the split of the matrices wants to be performed on the *FM* (option *identity_algo*), or on the FSM (option *similarity_algo*). In the case that *similarity_algo* is selected, the split at step 3 is performed using only criteria 3 and 4.

Algorithm 1: Binary Tree Construction

N: Number of sequences, TL: Number of tree levels, NC: Number of clusters
X: Input matrix (Nx2) with the sequence id and the N_Θ-length sequences
Y: Level-Cluster matrix (NxTL)
M: Metric Matrix (NCx6) with the identity values (I, IS), the entropy values (H, HS) and the bin similarity (BS) of all clusters computed on FM and FMS
type: The algorithm type (identity_algo or similarity_algo)

Input: X, type
Output: Y, M
1. **Initialization**:
   Create a root node (cluster C_0) for the tree
2. **Iteration**:
   Repeat for every new cluster-i
   Compute the level that C_i belongs to
   Compute FM_i and FSM_i matrices
   Compute metrics (I_i, IS_i, H_i, HS, BS) and update M matrix
   Criteria for Division
   Select $cell_i$ of FM_i or FSM_i according to the criteria:
   If (type = similarity_algo) **then**
   Go to step *.4.11
   Elm ← the elements of FM_i with the maximum value
   if Elm.length < 2 **then**
   $cell_i$ ← Elm
   Go to step *.5
   end if
   Elm ← the elements of Elm with the minimum value of E_i
   if Elm.length < 2 **then**
   $cell_i$ ← Elm
   Go to step *.5
   end if
   Elm ← the elements of Elm with the maximum value of FSM_i
   if Elm.length < 2 **then**
   $cell_i$ ← Elm
   Go to step *.5
   end if
   Elm ← the elements of Elm with the minimum value of ES_i
   if Elm.length < 2 **then**
   $cell_i$ ← random element of Elm
   end if
   Division
   j ← index of the left child of node-i (C_i)
   Add the sequences that belong to $cell_i$ to cluster C_j
   Add all the other sequences to cluster C_{j+1}
   Fill in column Y[,level+1] with C_j or C_{j+1} accordingly
   Check if C_j or C_{j+1} is leaf
   End
3. return Y, M

Algorithm 2: Branch Breaking

Y: Level-Cluster matrix (NxTL)
M: Metric Matrix (NCx5) with the Identity values (I, IS), the Entropy values (H, HS), and the Bin Similarity (BS) of all clusters computed on FM and FSM
thrA: the threshold set for metric A
type: [identity_algo, similarity_algo]

Input: Y, M, thrI, thrH, thrBS, algo
Output: Y, M
1. **Initialization**:
   Find all unique paths of tree from the root till the leaves
2. **Iteration**:
   Repeat for each path-i
   PN_i <- the cluster ids that constitute the path
   Compare each cluster of the path with its child using the metrics
   Repeat for each cluster-j of PN_i
   if (type = identity_algo)
   condI ← $((I_j-I_{j+1})*0.5 + (IS_j-IS_{j+1})*0.5) < thrI$
   condH ← $(abs(H_j-H_{j+1})*50 + abs(HS_j-HS_{j+1})*50) < thrH$
   condBS ← $abs(BS_j-BS_{j+1}) < thrBS$
   cut_condition ← $I_j>20$ **&** (condI || condH || condBS)
   else
   condI ← $(IS_j-IS_{j+1}) < thrI$
   condH ← $abs(HS_j-HS_{j+1})*100 < thrH$
   condBS ← $abs(BS_j-BS_{j+1}) < thrBS$
   cut_condition ← $IS_j>20$ **&** (condI || condH || condBS)
   end if
   if (cut_condition) **then**
   Convert C_j to leaf
   Update Y, M matrices
   break
   end if
   End
   End
3. **return** Y, M

B. Branch Breaking

The second phase consists of the branch breaking process (Algorithm 2). For each branch of the constructed binary tree, the appropriate level to be cut is examined. Since the tree is asymmetric and the number of items that each cluster consists of varies, the tree is not cut by selecting a unique level for the overall tree, but branch-specific thresholds are used instead. 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 that is set for the identity metric is that the leaf clusters must have an Identity value higher than 20%. This 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.

V. Experimental Results

A. Experimental setup

Next, we evaluate the proposed hierarchical clustering framework. Analysis and evaluation are performed with benchmark data provided by the popular MovieLens 20M dataset [13], that consists of 465,000 tag applications applied to 27,000 movies by 138,000 users. We formed documents used as input to the preprocessing module using only user tags (after removing stop-words, special characters and acronyms) and not the movie titles, while the latter was found to be misleading in terms of semantically defining a

movie. Since the dataset also contains some movies without any user tags, we ended up with 19,370 items. Afterwards, we created 20-length item vectors after applying LDA on the aforementioned documents, experimenting on the number of topics N_Θ from 5 to 500 and evaluating the results using the perplexity metric.

The item vectors were then discretized in 10 bins represented by alphabetic letters from A to J. 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>.

B. Results and discussion

Through the application of the binary tree construction algorithm of *type similarity_algo*, a binary tree with 22 levels and 739 leaf clusters was constructed. The Identity and Similarity metrics began with 0 values at the root of the tree, whereas the Entropy metric began with 0.266. These values were improved as moving down the different levels of the tree and at the leaf level the Similarity value was close to 100 and the Entropy was close to 0.

TABLE I. AVERAGE VALUES OF THE EVALUATION METRICS OF THE CLUSTERS PER TREE LEVEL AFTER BRANCH BREAKING

L	*#C*	*I*	*IS*	*H*	*BS*	*TS*
0	1	0.000	0.000	0.266	59.967	NaN
1	2	0.000	2.500	0.202	70.165	NaN
2	4	21.250	27.500	0.154	80.505	NaN
3	8	46.875	55.000	0.103	88.143	NaN
4	13	56.538	66.538	0.084	91.148	NaN
5	19	61.579	73.421	0.067	92.874	NaN
6	25	60.800	76.000	0.068	93.581	NaN
7	33	62.576	79.697	0.062	94.437	39.400
8	40	66.750	82.625	0.053	95.070	39.400
9	43	66.977	82.791	0.052	95.124	50.300
10	44	65.455	82.045	0.054	94.907	50.300
11	45	64.000	81.444	0.057	94.709	50.300
12	46	62.609	80.978	0.059	94.595	49.600
13	47	61.277	80.638	0.061	94.457	49.600
14	48	60.000	80.417	0.063	94.356	49.600
15	49	58.776	80.306	0.064	94.306	49.600
16	50	57.600	80.300	0.065	94.251	49.600
17	51	56.471	80.392	0.066	94.221	49.600
18	52	55.385	80.577	0.066	94.218	46.300
19	53	54.340	80.849	0.066	94.237	57.000

At the second phase, the tree was cut by applying the branch break algorithm using the percentage of 0.5% as threshold for all the comparisons of the metrics. The final tree consists of 19 levels and 53 leaf clusters. The average values of each level's metrics using the *FM* and the *FSM* matrices are summarized in Table I. 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 it selves, 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.

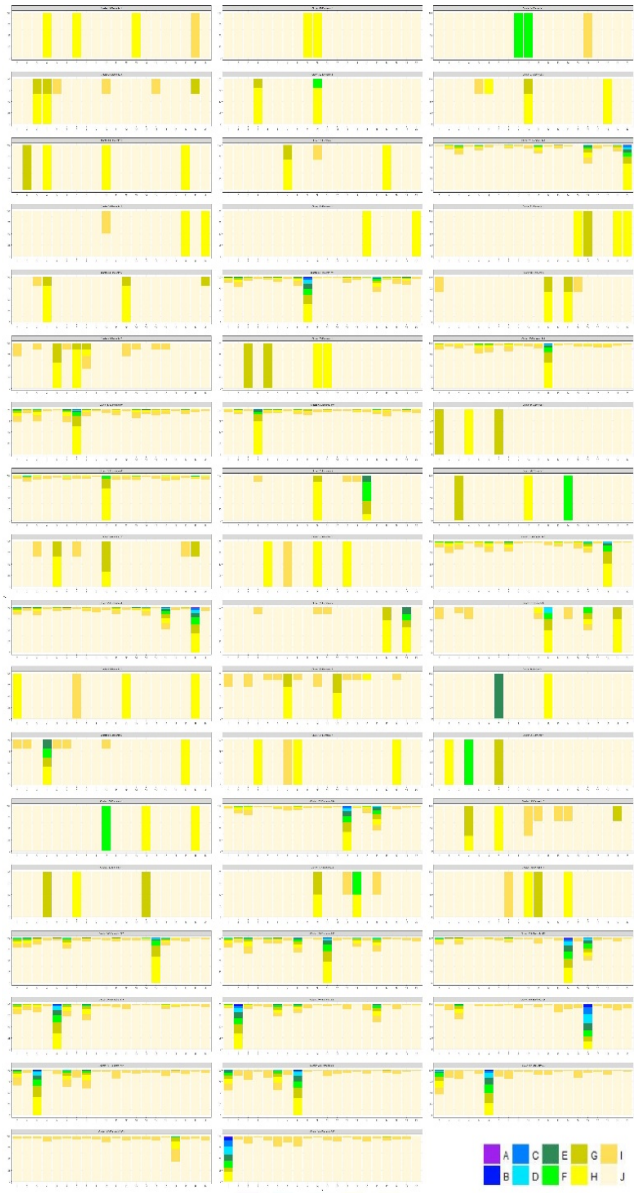

Fig. 2. Bar plots of the 53 clusters created after the branch breaking algorithm.

Fig. 2 shows the bar plots for each one of the leaf clusters after the application of the branch breaking algorithm. Bins with high percentage at a specific topic are visualized with blue colors, whereas bins with low percentage are presented with yellow colors. Due to the nature of the data, each cluster has a very low percentage at the majority of the topics and a middle or a high percentage at only a few topics. So, each cluster can be characterized by its major topics. For example, cluster 1, which is represented by the pattern JJJ**H**JJ**H**JJJJJ**H**JJJJJIJ, is characterized by the topics 4, 7 and 13, whereas the last cluster is characterized by topic 1.

C. Clustering evaluation

1) Semantic similarity for cluster evaluation

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.

Semantic similarity, in contrast to string-based matching can identify semantically relevant concepts that consist of different strings. A widely used metric for calculating semantic similarity is the Normalized Google Distance (*NGD*) [14], which is derived from the number of hits returned by the Google search engine for a given set of keywords. Keywords with similar meanings tend to be close in "units" of Google distance, while words with dissimilar meanings tend to be farther apart. The *NGD* between two search terms θ_i and θ_j is presented in (12), where *WP* is the total number of web pages searched by Google, $f(\theta_i)$ and $f(\theta_j)$ are the number of hits for terms θ_i and θ_j respectively, and $f(\theta_i,\theta_j)$ is the number of web pages that contain both θ_i and θ_j. In practice, NGD values belong to the range [0,1], with 1 referring to complete semantic match.

$$NGD = \frac{\max\{\log f(\theta_i), \log f(\theta_j)\} - \log f(\theta_i, \theta_j)}{\log(WP) - \min\{\log f(\theta_i), \log f(\theta_j)\}} \quad (12)$$

Although NGD is a reliable semantic metric, calculating the similarity of thousands of terms may turn out to be a time consuming process. As web search engines have limited throughput for computer generated queries, calculating the *NGD* value for a large number of documents within an acceptable time period is not feasible. Thus, in this work we use Wikipedia documents to employ an alternate of the NGD, the Wikipedia-based Semantic Similarity Metric.

To calculate the Wikipedia-based semantic similarity, we followed the approach proposed by Kolb [15]. First, we used a simple context window of three words for counting co-occurrences. By moving the window over a Wikipedia corpus consisting of 420,184 words (resulting into 1.9 billion tokens[1]), we generated a co-occurrence matrix. In order to find a word's distributionally similar words, one should compare every word vector with all other word vectors. For vector comparison, Lin's information theoretic measure [16] was employed. To compute the overall matching score between two topics, i.e. the pairwise Topic Similarity (*TS*) we used the matching average method (13), which calculates the similarity between two topics θ_i and θ_j by dividing the sum of similarity values of all match candidates of both sets by the total number of set tokens.

$$TS_{ij} = \frac{2 * Match(\theta_i, \theta_j)}{|\theta_i| + |\theta_j|} \quad (13)$$

By employing (13), a $N_\Theta \times N_\Theta$ similarity matrix with the pairwise *TS* between all the N_Θ topics was created. Its main diagonal entries were equal to 100%, whereas the rest of the entries were taking values between the interval [3, 63]%.

[1] http://www.linguatools.de/disco/disco-download_en.html.

[2] https://www.rdocumentation.org/packages/cluster/versions/2.0.7-1/topics/diana.

2) Comparison with a baseline hierarchical algorithm

In this section we present the experimental results on the MovieLens 20M dataset using the Frequency Based Hierarchical Clustering (FB HC) algorithm and a Baseline Divisive Hierarchical[2] (BHC) algorithm with four different evaluation metrics: *I*, *IS*, *H*, *BS*, *TS*. Although it is common for internal clustering evaluation to use the Silhouette coefficient, this metric measures how close each point in one cluster is to points in the neighboring clusters and is computed by making pairwise comparisons between the elements of the dataset. Computing pairwise distances is counter to the main point of our algorithm due to the high complexity it introduces as well as the exponential growth with respect to the size of the data. The pairwise comparisons are replaced by the usage of the frequency matrix in our case, so the Silhouette coefficient and other relevant metrics for cluster validation do not satisfy our algorithm's needs.

TABLE II. AVERAGE VALUES OF THE EVALUATION METRICS OF THE CLUSTERS CREATED USING THE FB AND BASELINE HC ALGORITHMS

#C	Algorithm	I	IS	H	BS	TS
23	BHC	0.652	13.696	0.167	85.769	56.400
	FB HC	54.565	74.783	0.081	93.264	NaN
33	BHC	4.697	24.091	0.153	87.837	49.600
	FB HC	68.939	83.939	0.050	95.439	NaN
53	BHC	11.415	35.189	0.139	89.847	44.400
	FB HC	54.340	80.849	0.066	94.237	57.000
125	BHC	24.800	53.080	0.120	92.886	42.300
	FB HC	69.120	90.600	0.038	96.981	40.300

Table II depicts the comparison between the average values of the evaluation metrics that have been computed for each cluster, using four different number of clusters: 23, 33, 53, 125. For the FB HC algorithm these clusters were created by applying the branch break algorithm using as threshold for all the metrics the values: 5, 1, 0.5, 0.25, accordingly, whereas for the BHC algorithm the desired number of clusters is an input parameter. Topic similarity is computed only for those clusters that have more than one major topics and contain more than 10 elements. High values of Identity, Bin Similarity and Topic Similarity, and low values for Entropy indicate high quality clustering results, so this table makes clear that FB HC algorithm outperforms the baseline algorithm.

Fig. 3 shows the distribution of the metric values that correspond to each cluster that has been created using the two algorithms for number of clusters equal to 53. The box plots and the single points that correspond to each cluster's metrics are presented. The lines extending vertically from the boxes indicate the variability outside the upper or the lower quartiles, whereas the horizontal lines indicate the median values of the data points. We observe that the distributions of the FB HC algorithm for all *IS*, *H* and *BS* metrics are much better than those created by the BHC algorithm. *TS* metric is also better for our method but the difference here is not so high. More specifically, in the FB HC algorithm, metric *IS* is 45.6% higher, *H* is 52.5% lower and *TS* is 4.4% higher than the corresponding values that the BHC algorithm achieved.

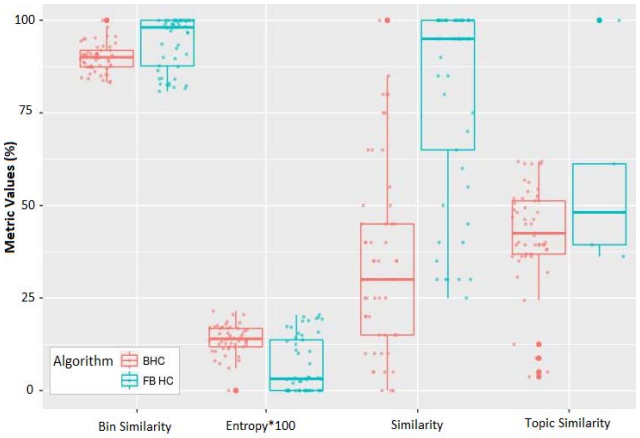

Fig. 3. Comparison between BHC and FB HC algorithms for 53 clusters.

Regarding the memory demands, the average amount of memory that was used during the application of the FB HC algorithm is only 63.64MB, whereas the corresponding value for the BHC algorithm equals to 3063.81MB. As for the computational time, the FB HC took 6.7min to run, whereas the BHC took 1260min. This means that our method achieved a 98% reduction in memory usage and a 99.4% reduction in computational time.

VI. Conclusion and Future Work

In this work we proposed a novel hierarchical clustering framework for clustering items that derive from topic modeling. Our framework consists of two main phases: 1) the application of a binary tree construction algorithm, which constructs a hierarchy of items using Identity, Similarity and Entropy measures to form the clusters, and 2) a branch breaking algorithm, which prunes each branch of the tree at an appropriate level using thresholds for the evaluation metrics. Our approach overcomes limitations regarding the number of items that can be handled by a hierarchical clustering algorithm due to memory limitations. Moreover, our algorithm has increased scalability compared to a baseline hierarchical clustering algorithm, as instead of making pairwise comparisons between all the elements of the dataset to form the clusters, frequency tables are used.

The clustering method was applied to the MovieLens 20M dataset and results of this analysis showed that the Identity and Similarity values increase as one transitions from the root towards the leaves of the tree. The final number of clusters can be selected by setting appropriate thresholds as input to the branch breaking algorithm.

Future work involves the application of graph theory in order 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. Moreover, a valuable extension of this work for real time applications would be the implementation of a decision-making algorithm that exploits the hierarchy of the clusters to perform new item categorization into the existing clusters. Finally, additional testing and optimization for performance is needed, focusing on much bigger datasets, as well as a robust statistical evaluation of the evident improvement in memory usage and computational time.

Acknowledgment

This work was partially funded by an IKY scholarship funded by the "Strengthening of Post-Academic Researchers" Act from the resources of the OP "Human Resources Development, Education and Lifelong Learning" with Priority Axes 6,8,9 and co-funded by the European Social Fund ECB and the Greek government.

References

[1] Ranjbar Kermany, N., & Alizadeh, S. H. (2017). A hybrid multi-criteria recommender system using ontology and neuro-fuzzy techniques. Electronic Commerce Research and Applications, 21, 50–64.

[2] Srivastava, R. K., Leone, R. P., & Shocker, A. D. (1981). Market structure analysis: hierarchical clustering of products based on substitution-in-use. The Journal of Marketing, 38-48.

[3] Hol, V., & Sokol, O. (2017). Clustering retail products based on customer behaviour. Applied Soft Computing, 60(C), 752-762.

[4] Frémal, S., & Lecron, F. (2017). Weighting strategies for a recommender system using item clustering based on genres. Expert Systems with Applications, 77, 105-113.

[5] Das, J., Mukherjee, P., Majumder, S., & Gupta, P. (2014, November). Clustering-based recommender system using principles of voting theory. In Contemporary computing and informatics (IC3I), 2014 international conference on (pp. 230-235). IEEE.

[6] Wang, X., Wang, X., Ding, Z., Nie, X., & Xiao, L. (2017). A New Algorithm Based on Item Clustering and Matrix Factorization. International Journal of Engineering and Technology, 9(2), 160.

[7] Yang, H., Lee, H. J., Cho, S., & Cho, E. (2016, December). Automatic classification of securities using hierarchical clustering of the 10-Ks. In Big Data (Big Data), 2016 IEEE International Conference on (pp. 3936-3943). IEEE.

[8] Larson, R. R. (2010). Introduction to information retrieval. Journal of the American Society for Information Science and Technology, 61(4), 852-853.

[9] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.

[10] D. Newman, T. Baldwin, L. Cavedon, E. Huang, S. Karimi, D. Martinez, F. Scholer, J. Zobel, Visualizing search results and document collections using topic maps, Web Semant. Sci. Serv. Agents World Wide Web 8 (2010) 169–175.

[11] G. Casella, E.I. George, Explaining the gibbs sampler, Am. Stat. 46 (1992) 167

[12] Lin, J. (1991). Divergence measures based on the Shannon entropy. IEEE Transactions on Information theory, 37(1), 145-151.

[13] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Trans. Interact. Intell. Syst. 5, 4, Article 19 (December 2015), 19 pages.

[14] R.L. Cilibrasi, P.M.B. Vitanyi, The google similarity distance, IEEE Trans. Knowl. Data Eng. 19 (2007) 370–383.

[15] P. Kolb, DISCO: a multilingual database of distributionally similar words, in: A. Storrer, A. Geyken, A. Siebert, K.-M. Würzner (Eds.), KONVENS 2008 — Ergänzungsband: Textressourcen und lexikalisches Wissen, 2008, pp. 37–44.

[16] D. Lin, Automatic retrieval and clustering of similar words, Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics — Volume 2, ACL '98, Association for Computational Linguistics, Stroudsburg, PA, USA, 1998, pp. 768–774.