Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

TIE algorithm: a layer over clustering-based taxonomy generation for handling evolving data#

Rabia IRFAN‡†[1], Sharifullah KHAN[1], Kashif RAJPOOT[1,2], Ali Mustafa QAMAR[1,3]
[1]School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad 44000, Pakistan
[2]School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK
[3]Department of Computer Science, College of Computer, Qassim University, Al Mulaida, Buraydah 52344, Saudi Arabia
†E-mail: 12phdrirfan@seecs.edu.pk

Received Aug. 4, 2017; Revision accepted Dec. 3, 2017; Crosschecked June 8, 2018

Abstract: Taxonomy is generated to effectively organize and access large volume of data. A taxonomy is a way of representing concepts that exist in data. It needs to continuously evolve to reflect changes in data. Existing automatic taxonomy generation techniques do not handle the evolution of data; therefore, the generated taxonomies do not truly represent the data. The evolution of data can be handled by either regenerating taxonomy from scratch, or allowing taxonomy to incrementally evolve whenever changes occur in the data. The former approach is not economical in terms of time and resources. A taxonomy incremental evolution (TIE) algorithm, as proposed, is a novel attempt to handle the data that evolve in time. It serves as a layer over an existing clustering-based taxonomy generation technique and allows an existing taxonomy to incrementally evolve. The algorithm was evaluated in research articles selected from the computing domain. It was found that the taxonomy using the algorithm that evolved with data needed considerably shorter time, and had better quality per unit time as compared to the taxonomy regenerated from scratch.

Key words: Taxonomy; Clustering algorithms; Information science; Knowledge management; Machine learning
https://doi.org/10.1631/FITEE.1700517 **CLC number:** TP312

1 Introduction

Data is now produced in large volume on a daily basis (Turner et al., 2014). According to the Computer Science Corporation (Koff and Gustafson, 2011) report on data revolution, experts are expecting 4300% increase in annual data generation by 2020. A major chunk of this data is unstructured text data, such as scholarly articles, technical reports, organizational policy documents (Blumberg and Atre, 2003). These dominate 80% of the data industry (Koff and Gustafson, 2011). This exponential growth of data is so overwhelming that it can actually lead to miss of new directions and emerging ideas, rather than discovering novel insights. To effectively use this data, we should process and transform it into valuable information. The organization of this information in a structured form, as with taxonomy, can be helpful in using it in a timely, effective, and accurate fashion. Taxonomy is a structured organization of hierarchical or parent–child relationships of concepts presented in data (Paukkeri et al., 2012). Muller et al. (1999) defined taxonomy as a thematic structure inherent in data. There are many applications of taxonomy. It is an effective way to categorize and organize data (Sujatha and Krishna Rao, 2011). It provides standardization, so that fewer interoperability issues may arise (Engel et al., 2010). Furthermore, it serves as a foundation structure for content and knowledge

‡ Corresponding author
# A preliminary version was presented at the 15th IEEE International Conference on Machine Learning and Applications, Anaheim, CA, USA, December 18–20, 2016
ORCID: Rabia IRFAN, http://orcid.org/0000-0002-7789-5338

management (Hedden, 2010), information search and navigation (Sánchez and Moreno, 2004), and analytics and text mining (Weng and Liu, 2004; Spangler et al., 2006; Li and Anand, 2009; Camiña, 2010; Dawelbait et al., 2010).

Currently, we find many automatic taxonomy generation techniques, which are effectively producing taxonomies for small to large data sets. However, the existing techniques ignore the fact that data is extremely growing at a rapid pace and the changes that occur in data should also be reflected in the taxonomy. The taxonomy that does not consider the changes occurring in data cannot truly represent the data. There can be two ways to handle such changes. One way is to regenerate taxonomy from scratch, which is not economical considering time and resources. Another way is to allow taxonomy to incrementally evolve. The taxonomy incremental evolution (TIE) algorithm proposed in this study is a novel attempt in this direction. The algorithm serves as a layer over an existing clustering-based taxonomy generation technique and incrementally evolves with the existing taxonomy. Clustering-based taxonomy generation techniques (Muller et al., 1999; Kashyap et al., 2005; Dietz et al., 2012) typically use hierarchical clustering and labeling techniques (Jain et al., 1999) to generate taxonomy.

The TIE algorithm takes an existing taxonomy, the respective hierarchical structure (i.e., a hierarchical organization of clusters) and newly arriving documents as its input. It identifies the closest cluster for each of the newly arriving documents based on a similarity score. The range of the similarity score determines the level of impact that a new document has on its closest cluster. Based on the level of impact, various reorganization operators are applied to adjust the newly arriving documents in the existing hierarchical structure. Finally, the existing taxonomy evolves to represent the changes occurring in the data. The algorithm is compared with the taxonomy regeneration approach based on complexity analysis and empirical evaluation. A text data set of scholarly articles selected from the computing domain was used for evaluation purpose. It is found that the incremental evolution is better than regeneration subject to time consumption. It produces better quality taxonomy per unit time than that of regeneration of taxonomy from scratch. The main contributions of this work are summarized as follows: The TIE algorithm evolves with the existing taxonomy whenever changes occur in data to represent an updated view of the data in a shorter time.

1. The TIE algorithm is independent of the clustering algorithm used for taxonomy generation, so existing clustering-based approaches to taxonomy generation can be effectively used with the TIE algorithm.

2. A new metric named the quality-time ratio is proposed to measure the effectiveness of taxonomy evolution in comparison to regeneration. It determines the ratio of taxonomy quality improvement per unit time.

3. Sensitivity analysis of the TIE algorithm is done to determine the impact of varying different factors influencing the evolution of taxonomy.

It is declared that this study is an extension of our earlier work (Irfan and Khan, 2016). Here the proposed algorithm is described with supplementary details. Additional evaluations in terms of the quality-time ratio and sensitivity analysis are included. The proposed algorithm is validated as a layer over both agglomerative and divisive approaches to hierarchical clustering.

2 Related work

Taxonomy can be generated through various approaches, such as clustering-based (Muller et al., 1999; Kashyap et al., 2005; Neshati et al., 2007; Dietz et al., 2012; Yang et al., 2015), rules and heuristics-based (Medelyan et al., 2013; Meijer et al., 2014; Lefever, 2015), and graph-based (Camiña, 2010; Qi et al., 2010; Fountain and Lapata, 2012; Velardi et al., 2013) approaches. The proposed TIE algorithm is a layer over an existing clustering-based taxonomy generation technique, so this review is limited to the clustering-based approaches. Clustering-based taxonomy generation techniques are reviewed here for two main reasons: first, they contribute as a foundation in this research; second, they are reviewed to identify their capabilities of handling evolving data. Other techniques that have addressed taxonomy evolution are also reviewed.

Clustering-based taxonomy generation techniques typically use hierarchical clustering and

labeling to generate taxonomy (Jain et al., 1999). The work of Muller et al. (1999) was one of the pioneer attempts of using a clustering-based approach for taxonomy generation. Their technique produced effective taxonomies with more focus on scalability so that the generated taxonomy should be less affected by the increasing size of data. Kashyap et al. (2005) developed an experimental framework to analyze the impact of varying different methods and parameters in the generation, such as the use of natural language processing versus non-natural language processing, the use of different similarity or distance measures in the clustering process, and document clustering versus term clustering. Dietz et al. (2012) focused on the generation of domain-specific taxonomies and found that general terms were not enough to construct a domain-specific taxonomy. Sclano and Velardi (2007) identified domain-specific concepts that were present in data by using domain-specific measures of domain pertinence and domain consensus. In addition, they compared different knowledge-rich methods (i.e., based on the involvement of external knowledge sources, such as Wikipedia and WordNet) and knowledge-poor methods (i.e., based on statistical and lexical properties extracted from the data) for the extraction of relevant concepts and hierarchical relationships to see their impact on the generated taxonomy. It was observed that clustering-based techniques involving knowledge-rich methods extracted relevant concepts and hierarchical relationships, and semantically produced a better taxonomy than those techniques involving knowledge-poor methods (Neshati et al., 2007). Some of the recent techniques (Paukkeri et al., 2012; Yang et al., 2015) use self-organizing maps to perform clustering-based taxonomy generation. The self-organizing map is a famous artificial neural network algorithm, which is effective in mapping a high dimensional input data to a low dimensional map. Each node in the map is called 'neuron' and it facilitates clustering by grouping similar data objects closer in the map or under a single neuron. In short, most of the existing automatic taxonomy generation techniques are effectively producing taxonomies for small to large data sets. However, they are not focusing on handling evolving data.

On the other hand, Marcacini and Rezende (2010) and Yao et al. (2012) explored taxonomy evolution along with its generation. Yao et al. (2012) performed the generation and evolution of taxonomy for tag space (i.e., semi-structured data) using an association rule graph (Irfan and Khan, 2016), whereas Marcacini and Rezende (2010) performed the generation and evolution of taxonomy for tag space (i.e., unstructured data) using an incremental hierarchical clustering based approach (Irfan and Khan, 2016). The nature of tag data is different from that of unstructured text data and the hierarchical relationships extracted from unstructured text data are more complex (Blumberg and Atre, 2003). Our research focuses on unstructured text data, and therefore the work of Yao et al. (2012) is not explored further. It was observed that Marcacini and Rezende (2010) performed clustering of terms instead of document clustering. The use of term clustering for hierarchy formation is helpful because it eliminates the need to associate labels with clusters. However, document clustering is preferred because it produces more distinct clusters than term clustering (Kashyap et al., 2005). We attempt to identify a solution that can incrementally evolve with an existing taxonomy independent of the clustering algorithm used in the generation of the taxonomy and serve as a layer over an existing clustering-based taxonomy generation technique.

3 Taxonomy generation process

Since the proposed solution to taxonomy evolution can serve as a layer over an existing clustering-based taxonomy generation technique, we describe the fundamentals of the clustering-based taxonomy generation process (TGP) in this section. The TGP typically comprises four steps: data pre-processing, data modeling, hierarchy formation, and node labeling (Irfan and Khan, 2016). We briefly describe the commonly used methods in these four steps for clustering-based taxonomy generation.

3.1 Data pre-processing

The data pre-processing step cleans unnecessary details from data and refines the terms that exist in the data. These refined terms reflect properties or characteristics of the data and form the vocabulary of the data (Kumar and Chandrasekhar, 2012). In TGP for text data, basic natural language processing

techniques, such as tokenization, stemming, part of speech tagging, and parsing (Nadkarni et al., 2011), have been applied to pre-process data (Muller et al., 1999; Kashyap et al., 2005; Spangler et al., 2006; Dietz et al., 2012; Paukkeri et al., 2012). The data, even after applying data pre-processing activities, is not in machine-readable format (i.e., a proper data model), so it is forwarded to the data modeling step for conversion into a machine-readable format.

3.2 Data modeling

The data modeling step involves finding a suitable model that expresses data in a machine-readable format for computation. The vector space model (VSM) is one of the most widely used data modeling techniques for text data (Baeza-Yates and Ribeiro-Neto, 2011) and was used by Muller et al. (1999), Kashyap et al. (2005), Spangler et al. (2006), and Paukkeri et al. (2012). For a collection of documents, VSM represents each document and its refined terms in the form of a vector that shows the occurrence of terms in the document. The occurrences of terms are represented with f_t and f_{id} (i.e., term frequency-inverse document frequency) scores in the VSM. f_t, which means the term frequency, is the number of times a particular term appears in a document. f_{id}, which means the inverse document frequency, represents how often the term appears in the collection of documents. In addition, various methods are applied in the data modeling step to determine the semantics of terms for their further refinement, such as involvement of external knowledge sources (e.g., WordNet, Wikipedia, Freebase, and DBbase) (Medelyan et al., 2013), use of advanced natural language processing techniques (e.g., word sense disambiguation (WSD) (Nadkarni et al., 2011)) (Dietz et al., 2012), and application of dimensionality reduction techniques (e.g., latent semantic indexing (Deerwester et al., 1990)) (Kashyap et al., 2005). After the modeling step, data is now ready in machine-readable format for extracting hierarchical structure that exists in the data, so it is passed to the hierarchy formation step.

3.3 Hierarchy formation

In the hierarchy formation step a hierarchical structure inherent in the data is identified and constructed. It comprises two sub-steps: relationship identification and hierarchy generation. By relationship identification the relationships that exist among different documents in data are determined. For relationship identification, similarity or distance measures, such as cosine similarity and Euclidean distance, are employed (Cha, 2007; Thada and Jaglan, 2013). The hierarchy generation arranges these relationships in the form of a hierarchical structure. Hierarchical clustering algorithms are used for hierarchy generation in clustering-based taxonomy generation techniques.

There are two types of hierarchical clustering algorithms: agglomerative (bottom up) and divisive (top down) (Jain et al., 1999). The agglomerative approach starts with every data object placed in a separate cluster and merges clusters at later stages. Hierarchical agglomerative clustering (HAC) was an example of the agglomerative approach and was used by Muller et al. (1999) and Dietz et al. (2012). Based on different merging styles, different flavors of HAC exist. The common ones are single link, complete link, average link, and centroid link (Manning et al., 2008). The divisive approach starts with all data objects in one cluster and divides them into more clusters at later stages. Bisect *K*-means clustering was an example of the divisive approach and was used by Kashyap et al. (2005). Agglomerative approaches have quadratic time complexity, whereas divisive approaches have linear time complexity. Nevertheless, the clustering quality is better in the case of agglomerative approaches than that in the case of divisive approaches (Steinbach et al., 2000).

Hierarchical structure, generated in this step, comprises a hierarchy of unlabeled clusters (i.e., nodes). The unlabeled hierarchical structure is forwarded to the node labeling step.

3.4 Node labeling

During the node labeling step labels are assigned to unlabeled nodes in a hierarchical structure. The assignment of meaningful and accurate labels to unlabeled nodes in a hierarchy is necessary to have a better understanding of the generated taxonomy. In clustering-based approaches, usually the centroid of a cluster (i.e., an average of all data objects in a cluster) is involved in finding labels for hierarchical nodes, as applied by Dietz et al. (2012). The labeling techniques are mostly combined with rules and heuristics in order to find appropriate labels for the taxonomy (Kashyap

et al., 2005). External knowledge sources (Carmel et al., 2009), such as WordNet, feature selection techniques (e.g., mutual information, chi-square χ^2) (Manning et al., 2008), and frequently occurring top k terms (Glover et al., 2002; Treeratpituk and Callan, 2006), have been used to identify suitable labels. The output of this step is a taxonomy, where hierarchical clusters are labeled to represent a hierarchical organization of the given data.

4 Taxonomy incremental evolution algorithm

We present a detailed and improved version of the TIE algorithm given in Irfan and Khan (2016). The TIE algorithm aims to find an incremental solution for evolving with an existing taxonomy whenever new documents are added to underlying data. The TIE algorithm is designed in a way that it serves as a layer over an existing clustering-based taxonomy generation technique. It comprises two steps:

1. identifying the closest cluster—This step determines the closest existing cluster for each of the newly arriving documents;

2. reorganizing the existing taxonomy—This step applies various reorganization operators to adjust a newly arriving document in the existing hierarchical structure and finally complete evolution of the existing taxonomy.

Note that the first step of the TIE algorithm, identifying the closest cluster, is fundamentally the same as given in our earlier work (Irfan and Khan, 2016), but it now defines the actions to be taken in case a newly arriving document has more than one closest cluster, or if it is not close to any of the existing clusters. The second step, i.e., reorganizing the existing taxonomy, now defines a new reorganization operator of insert_child for adjusting those newly arriving documents, which appear to be a part of the existing cluster but affect its quality. Next, we present the detailed explanation of these two steps of the TIE algorithm.

4.1 Identifying the closest cluster

When a new document arrives in existing data, the TIE algorithm first identifies an appropriate cluster for adjusting the new document. The algorithm takes an existing taxonomy T_X (which is the taxonomy obtained as a result of TGP), the respective hierarchical structure H (which is the output of the hierarchy formation step of TGP), and a set of newly arriving documents D' as its input. The hierarchy H maintains three types of information for each cluster: cluster centroid, cluster cohesion, and cluster deviation.

For a cluster $c \in H$ having m document vectors $\boldsymbol{d}_i$ (i=1, 2, …, m), mapped in a T-dimensional term space, cluster centroid $\boldsymbol{c}_{\text{cen}}$ is the center or middle point of the cluster and the average representation of all documents presented in the cluster, given as (Irfan and Khan, 2016)

$$\boldsymbol{c}_{\text{cen}} = \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{d}_i. \tag{1}$$

Let $\text{sim}(\boldsymbol{d}_i, \boldsymbol{c}_{\text{cen}})$ be the similarity between vectors of the i^{th} document $\boldsymbol{d}_i$ in cluster c and its centroid $\boldsymbol{c}_{\text{cen}}$. Then cluster cohesion c_μ is the average similarity of all documents presented in the cluster with the cluster centroid and a measure of tightness of the cluster, given as (Irfan and Khan, 2016)

$$c_\mu = \frac{1}{m}\sum_{i=1}^{m}\text{sim}(\boldsymbol{d}_i, \boldsymbol{c}_{\text{cen}}). \tag{2}$$

Cluster deviation c_σ denotes the deviation of a document from a cohesive cluster. It measures the limit of closeness or farness of all documents present in the cluster from the cluster centroid, given as (Irfan and Khan, 2016)

$$c_\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left((\text{sim}(\boldsymbol{d}_i, \boldsymbol{c}_{\text{cen}}) - c_\mu\right)^2}. \tag{3}$$

The main idea is to identify the closest cluster for a newly arriving document. Let $\text{sim}(\boldsymbol{d}_i, \boldsymbol{c}_{\text{cen}})$ be the similarity between a new document $\boldsymbol{d}' \in D'$ and centroid $\boldsymbol{c}_{\text{cen}}$ of cluster c. Consider that c is identified as the closest cluster for $\boldsymbol{d}'$ because of maximum similarity. Then the range of the similarity score determines the level of impact that a new document has on its closest cluster and is defined through the following three conditions (Irfan and Khan, 2016):

1. Far from the cluster centroid: If $\text{sim}(\boldsymbol{d}_i, \boldsymbol{c}_{\text{cen}}) < c_\mu - c_\sigma$, then $\boldsymbol{d}'$ is far from $\boldsymbol{c}_{\text{cen}}$. In this condition, the

new document $\boldsymbol{d}'$, due to its farness from $\boldsymbol{c}_{\text{cen}}$, might affect the quality of cluster c, and thus restructuring of the cluster will be required.

2. Close to the cluster centroid: If $\text{sim}(\boldsymbol{d}_i, \boldsymbol{c}_{\text{cen}}) > c_\mu + c_\sigma$, then $\boldsymbol{d}'$ is close to $\boldsymbol{c}_{\text{cen}}$. In this condition, the new document $\boldsymbol{d}'$, due to its closeness from $\boldsymbol{c}_{\text{cen}}$, might affect the labels of cluster c (which are assumed to be dependent on the center of the cluster as in most cases for clustering-based taxonomy generation techniques); thus, relabeling of the cluster will be required.

3. Neither close to nor far from the cluster centroid: If $c_\mu - c_\sigma \le \text{sim}(\boldsymbol{d}_i, \boldsymbol{c}_{\text{cen}}) \le c_\mu + c_\sigma$, then $\boldsymbol{d}'$ is neither close to nor far from $\boldsymbol{c}_{\text{cen}}$. In this condition, the new document $\boldsymbol{d}'$ because of its presence within the range of $\boldsymbol{c}_{\text{cen}}$ might belong to cluster c, and thus it will be merged in the cluster.

This step begins by associating three types of list structures, corresponding to each of the aforementioned conditions for each cluster in H. These list structures hold newly arriving documents based on the range of the similarity scores that newly arriving documents possess with their closest cluster. The list structures associated with each cluster are initialized to null. After initialization, the similarity between each of the new documents in D' and each of the existing clusters in H is calculated using the respective in cluster centroid. The cluster having maximum similarity (i.e., the closest cluster) is selected as a suitable candidate for a new document to be adjusted, as shown in steps 2–10 of Algorithm 1. For cluster $c \in H$, let α be the list structure associated with the condition 'far from the cluster centroid', β the list structure associated with the condition 'close to the cluster centroid', and γ the list structure associated with the condition 'neither close to nor far from the cluster centroid'. For a new document $\boldsymbol{d}'$ and its closest cluster c having centroid $\boldsymbol{c}_{\text{cen}}$, cohesion c_μ, deviation c_σ, and the range of the similarity score of $\boldsymbol{d}'$ from $\boldsymbol{c}_{\text{cen}}$ are then checked under the aforementioned conditions, and the following actions are taken:

1. If the condition 'far from the cluster centroid' is true, then add $\boldsymbol{d}'$ in the list α (see steps 11 and 12 in Algorithm 1);

2. If the condition 'close to the cluster centroid' is true, then add $\boldsymbol{d}'$ in the list β (see steps 13 and 14 of Algorithm 1);

3. If the condition 'neither close to nor far from the cluster centroid' is true, then add $\boldsymbol{d}'$ in the list γ (see steps 15 and 16 of Algorithm 1).

At the end of this step, the closest cluster is identified for each of the newly arriving documents. The list structures associated with each cluster contain the newly arriving documents, based on the range of the similarity that new documents possess with their closest cluster. The list structures for each cluster are the outputs of Algorithm 1.

Algorithm 1 Identifying the closest cluster

Input: document vectors for newly arriving documents in D', existing taxonomy T_X, and existing hierarchy H with centroid, cohesion, and deviation values for each cluster in H.
Output: list structures for each cluster in H.

```
1   initialize list structures for each cluster in H to null
2   for each d′∈D′ do
3     initialize maximum_similarity←0
      closest_cluster←null
4     for each c∈H do
5       calculate sim(d′, c_cen)
6       if sim(d′, c_cen)>maximum_similarity then
7         maximum_similarity←sim(d′, c_cen)
8         closest_cluster←c
9       end if
10    end for
11    if maximum_similarity<c_μ−c_σ for the closest_
      cluster then
      // Steps 11 and 12 in case d′ is far from c_cen
12    α←α+d′
13    else if maximum_similarity>c_μ+c_σ for the closest_
      cluster then
      // Steps 13 and 14 in case d′ is close to c_cen
14    β←β+d′
15    else if c_μ−c_σ≤maximum_similarity≤c_μ+c_σ for the
      closest_cluster then // Steps 15 and 16 in case d′ is
      // neither close to nor far from c_cen
16    γ←γ+d′
17    end if
18  end for
```

In the process of determining the closest cluster for a new document, there can be two exceptions. There is a possibility that a new document possesses equal similarity with more than one existing cluster. In that case, any of the existing clusters is selected randomly as a suitable candidate for its adjustment and appropriate actions are taken based on the range of the similarity score of that document with the centroid of the selected cluster. Another possibility is that a new document has no similarity with all of the existing clusters. In that case, any of the children of the root node of H is selected randomly, and the new

document is dealt with according to the condition 'far from the cluster centroid'.

4.2 Reorganizing the existing taxonomy

Taxonomy is a thematic representation of data. It is not necessary that the addition of a new document in the data brings drastic changes to the taxonomy. Therefore, in this step, before reflecting changes in the existing taxonomy, the quality of each of the existing clusters in H is checked by recalculating the cohesion score (using existing and new documents). A higher value of cohesion represents a better cluster. Based on whether or not a cluster quality has been deteriorated and the presence of new documents in its respective list structures, reorganization operators are applied to reorganize it. Consider the cluster $c \in H$, now comprising m existing and m' new documents and let c_{sib}, c_{par}, and c_{chl} denote its sibling, parent, and child clusters, respectively. The operators applied for reorganizing the cluster are given in Table 1.

This step begins by recalculating the cohesion of existing clusters in H, using existing and new documents. Thus, for the cluster $c \in H$ having m existing and m' new documents, let the new cohesion score be c_μ'. Then there can be two possible scenarios: deteriorated cluster quality and non-deteriorated cluster quality.

4.2.1 Scenario 1: deteriorated cluster quality

If $c_\mu' < c_\mu$, this shows that the cluster quality has been deteriorated. Then based on the presence of new documents in α, β, and γ lists of c, reorganization of the cluster c is performed through the following cases.

1. Super-level restructuring

Super-level restructuring occurs when a cluster quality is deteriorated and its list structure associated with 'far from the cluster centroid' condition is not empty. For the cluster c, if $\alpha \neq$null, this indicates that documents in α which are far from the cluster centroid have resulted in its quality deterioration and should be inserted as a separate node. So, the insert_sibling operator is applied to create a separate node, i.e., c_{sib}, which is then assigned labels by applying the label_cluster operator. This action leaves labels of parent cluster c_{par} of c as inaccurate, so it should be assigned new labels. However, before doing that the presence of documents in β and γ lists for c is also checked:

(1) If $\beta \neq$null for c, then check the presence of documents in its γ list:

1) If $\gamma \neq$null, this indicates that both the lists β and γ contain some documents. So, in this case, documents in β and γ first become part of c by applying the merge_doc operator resulting in an updated cluster c'. The documents in γ are those that are neither close to

Table 1 List of reorganization operators used by the TIE algorithm to reorganize the existing taxonomy

Input	Operator	Output	Function	Figure
Cluster c and list α	Insert_sibling	Sibling cluster c_{sib}	Taking documents that are far from a cluster and inserting them as its sibling cluster (Irfan and Khan, 2016)	Fig. 1a
Cluster c and list γ	Insert_child	Child cluster c_{chl}	Taking documents that are neither close to nor far from a cluster and inserting them as its child cluster	Fig. 1b
Cluster c	Label_cluster	Set of labels l_c	Assigning a set of unique labels to a cluster (Irfan and Khan, 2016)	Fig. 1c
Cluster c and its set of labels l_c	Alter_label	Set of new labels l_c'	Taking a set of existing labels for a cluster and assigning a set of new and unique labels to that cluster, where the set of new and existing labels may or may not be null (Irfan and Khan, 2016)	Fig. 1d
Cluster c and list θ, which contains n newly arriving documents to be merged in c, where $n \leq m'$	Merge_doc	Updated cluster c'	Merging documents, contained in a given list, in a cluster (Irfan and Khan, 2016)	Fig. 1e
Updated cluster c'	Alter_centroid	New centroid $\boldsymbol{c}_{\text{cen}}$	Recalculating centroid for an updated cluster (Irfan and Khan, 2016)	Fig. 1f

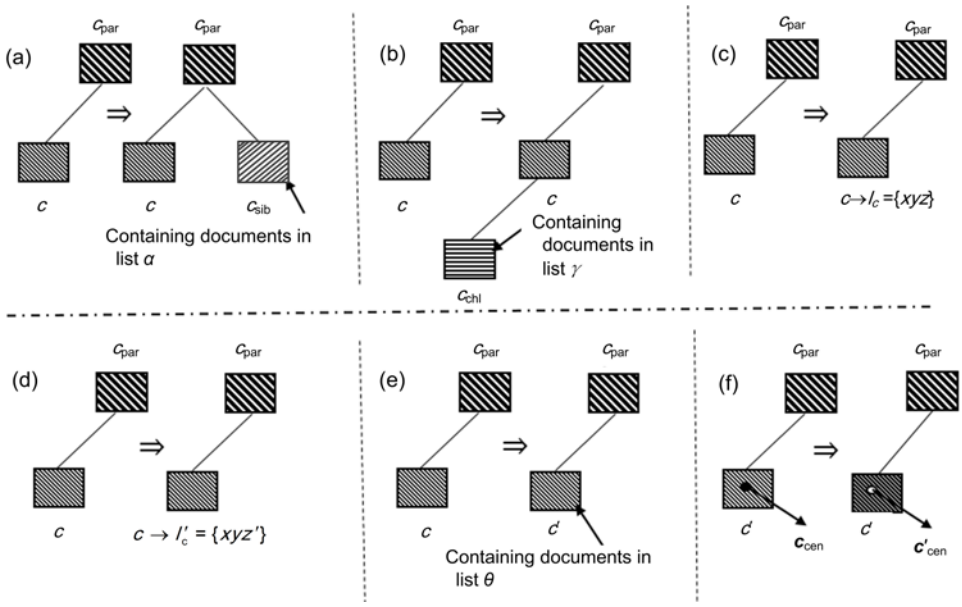

Fig. 1 Reorganization operators c_{sib}←insert_sibling(c, α) (a), c_{chl}←insert_child(c, γ) (b), l_c←label_cluster(c) (c), l'_c←alter_label(c, l_c) (d), c'←merge_doc(c, θ) (e), and c'_{cen}←alter_centroid(c') (f)

nor far from the cluster centroid, so they can be simply merged in the cluster (assuming to have minimal effect on its quality). However, the documents in β are close to the cluster centroid. The centroid is no longer an accurate representation of that cluster and this requires recalculation of the cluster centroid and assignment of new labels. So, the alter_centroid operator is applied to recalculate the c' centroid using existing and new documents. Finally, new labels are assigned to c' and then c_{par} by applying the alter_label operator.

2) If γ=null, this indicates that the list β contains some documents, but the list γ is empty. So, in this case, only documents in β first become part of c by applying the merge_doc operator resulting in an updated cluster c'. The documents in β are close to the cluster centroid. The centroid is no longer an accurate representation of that cluster and this requires recalculation of the cluster centroid and assignment of new labels. So, the alter_centroid operator is applied to recalculate the c' centroid using existing and new documents. Finally, new labels are assigned to c' and then c_{par} by applying the alter_label operator.

(2) If β=null for c, then check the presence of documents in its γ list:

1) If γ≠null, this indicates that though the list β is empty, the list γ contains some documents. So, in this case, only documents in γ first become part of c by applying the merge_doc operator resulting in an updated cluster c'. The documents in γ are neither close to nor far from the cluster centroid, so they can be simply merged in the cluster (assuming to have minimal effect on its quality); therefore, there is no need to recalculate the centroid and assign new labels to the updated cluster. New labels are assigned only to c_{par} by applying the alter_label operator.

2) If γ=null, this indicates that both the lists β and γ are empty and no new document is there in these lists to add in cluster c. So, in this case, without taking any further action new labels are assigned to c_{par} by applying the alter_label operator.

The actions taken for super-level restructuring are shown in steps 4–26 of Algorithm 2.

2. Relabeling

Relabeling occurs when cluster quality is deteriorated and its list structure associated with the 'far from the cluster centroid' condition is empty, but the one associated with the 'close to the cluster centroid' condition contains some documents. For cluster c, if α=null and β≠null, these indicate that documents in β that are close to the cluster centroid have affected its centroid representation. So, it should be recalculated, followed by the assignment of new labels to the cluster. However, before doing that the presence of documents in the γ list for c is also checked:

(1) If γ≠null, this indicates that the list γ also

contains some documents along with the list β. This case is similar to the super-level restructuring case, when the β and γ lists are not empty along with the α list, so the same set of reorganization operators is applied to perform the relabeling. The documents in both the lists β and γ become part of c by applying the merge_doc operator, resulting in an updated cluster c'. After that, the alter_centroid operator is applied to recalculate the c' centroid using existing and new documents. New labels are then assigned to c' by applying the alter_label operator. This action leaves labels of parent cluster c_{par} of c' as inaccurate, so new labels are assigned to c_{par} by applying the alter_label operator.

(2) If γ=null, this indicates that the list γ is empty. This case is similar to the super-level restructuring case when the β list is not empty along with the α list but the γ list is empty, so the same set of reorganization operators is applied to perform the relabeling. First, the documents in the β list become part of c by applying the merge_doc operator, resulting in an updated cluster c'. Afterwards the alter_centroid operator is applied to recalculate the c' centroid using existing and new documents. New labels are then assigned to c' by applying the alter_label operator. This action leaves labels of parent cluster c_{par} of c' as inaccurate, so new labels are assigned to c_{par} by applying the alter_label operator.

The actions taken for the relabeling are shown in steps 27–38 of Algorithm 2.

3. Sub-level restructuring

Sub-level restructuring occurs when cluster quality is deteriorated and its list structures associated with 'far from the cluster centroid' and 'close to the cluster centroid' conditions are empty, but the one associated with 'neither close to nor far from the cluster centroid' condition contains some documents. For cluster c, if α=null, β=null, and $\gamma\neq$null, these indicate that documents in γ that are neither close to nor far from the cluster centroid can be simply merged in the cluster (assuming to have minimal effect on its quality). However, in this case, both α and β lists are empty and only the documents in γ are affecting cluster quality, so instead of simply merging them in the cluster, we create a child node c_{chl} by applying the insert_child operator and insert the documents in the list γ in it. The operator label_cluster is then applied to assign labels to c_{chl}. This action leaves labels of c as inaccurate, so new labels are assigned to c by applying the alter_label operator.

The actions taken for sub-level restructuring are shown in steps 39–43 of Algorithm 2.

Example scenario 1 (Deteriorated cluster quality) For a cluster cx$\in H$, assume that α_{cx}, β_{cx}, and γ_{cx} are associated with the conditions 'far from the cluster centroid', 'close to the cluster centroid', and 'neither close to nor far from the cluster centroid', respectively, and that they contain document vectors for newly arriving documents (d_1', d_2'), (d_3'), and (d_4'), respectively. For cluster cx, further assume that its quality is found to be deteriorated when the quality is calculated with new and existing documents, i.e., $\text{cx}_\mu'<\text{cx}_u$. As per the steps shown in Algorithm 2, first the α_{cx} list is checked for the presence of documents. As $\alpha_{\text{cx}}\neq$null, super-level restructuring (i.e., steps 4–26) is applied here. The documents d_1' and $d_2'$are inserted as a sibling cluster of cx by applying the insert_sibling operator, and labels are assigned to the sibling cluster by applying the label_cluster operator. After that the β_{cx} list is checked for the presence of documents. Here $\beta_{\text{cx}}\neq$null, but before applying reorganization operators for $\beta_{\text{cx}}\neq$null, the γ_{cx} list is also checked. As in this case $\gamma_{\text{cx}}\neq$null, the merge_doc operator is applied to merge documents present in both β_{cx} and γ_{cx} lists. The documents d_3' and d_4' present in β_{cx} and γ_{cx}, respectively, become part of cx. Since the cluster cx is updated, we denote the updated cluster as cx′. Note that we check the γ_{cx} list before applying operators associated with the β_{cx} list due to the fact that if centroid recalculation and relabeling would be done before merging all the possible documents, then they would not accurately represent the cluster. As the γ_{cx} list is checked and appropriate actions have already been taken for the cx′, the centroid of the cluster is recalculated using the alter_centroid operator. Soon after altering the centroid, the updated cluster is assigned new labels by applying the alter_centroid operator. The addition of the sibling cluster and the updating of cx affect the label of their parent cluster, so the alter_centroid operator is applied to assign new labels to the parent cluster.

4.2.2 Scenario 2: non-deteriorated cluster quality

If $c_\mu'>c_\mu$, then updates are applied to the cluster through the following simple case: simple merging. Simple merging occurs when cluster quality has not

been deteriorated due to the addition of new documents. For cluster c, in this case documents in its α, β, and γ lists are simply merged in the cluster by applying the merge_doc operator, resulting in an updated cluster c', as shown in steps 44–46 of Algorithm 2.

Example scenario 2 (Non-deteriorated cluster quality) For a cluster cy$\in H$, assume that α_{cy}, β_{cy}, and γ_{cy} are associated with the conditions 'far from the cluster centroid', 'close to the cluster centroid', and 'neither close to nor far from the cluster centroid', respectively, and that they contain document vectors for newly arriving documents d_5', Φ, and d_6', respectively. For cluster cy, further assume that its quality is found not to be deteriorated when it is calculated with new and existing documents, i.e., $cy_\mu' > cy_\mu$. This is a simple merging (i.e., steps 44–46 of Algorithm 2) case and all the newly arriving documents are simply merged in the cluster. This is because the possible existence of newly arriving documents is not causing deterioration in the cluster quality. It would be time and cost efficient to simply merge the new documents in the cluster without making any adjustment, such as centroid recalculation or relabeling the cluster. So, as per the steps shown in Algorithm 2, the newly arriving documents, i.e., d_5' and d_6', are simply merged in the cluster cy by applying the merge_doc operator.

The output of this step of the TIE algorithm is an evolving version of the existing taxonomy.

Algorithm 2 Reorganizing the existing taxonomy

Input: list structures and cohesion for each cluster in H
Output: evolving taxonomy T'

```
1  for each c∈H do
2    calculate c'_μ for c using m existing and m' new documents
3    if c'_μ < c_μ then
4      if α≠null then // Steps 4–26: super-level // restructuring
5      apply c_sib←insert_sibling(c, α)
6      apply lc_sib←label_cluster(c_sib)
7       if β≠null then
8         if γ≠null then
9           apply c'←merge_doc(c, β+γ)
10          apply c'_cen←alter_centroid(c')
11          apply l'_c'←alter_label(c', l_c')
12          apply l'_c_par ←alter_label(c_par, l_c_par) //c_par: parent node
13        else
14          apply c'←merge_doc(c, β)
15          apply c'_cen←alter_centroid(c')
16          apply l'_c'←alter_label(c', l_c')
17          apply l'_c_par ←alter_label(c_par, l_c_par)
18        end if
19      else
20        if γ≠null then
21          apply c'←merge_doc(c, γ)
22          apply l'_c_par ←alter_label(c_par, l_c_par)
23        else
24          apply l'_c_par ←alter_label(c_par, l_c_par)
25        end if
26      end if
27    else if β≠null then // Steps 27–38: relabeling
28     if γ≠null then
29       apply c'←merge_doc(c, β+γ)
30       apply c'_cen ←alter_centroid(c')
31       apply l'_c'←alter_label(c', l_c')
32       apply l'_c_par ←alter_label(c_par, l_c_par)
33     else
34       apply c'←merge_doc(c, β)
35       apply c'_cen ←alter_centroid(c')
36       apply l'_c'←alter_label(c', l_c')
37       apply l'_c_par ←alter_label(c_par, l_c_par)
38     end if
39    else if γ≠null then // Steps 39–43: sub-level // restructuring
40     apply c_chl←insert_sibling(c, γ) //c_chl: child node
41     apply lc_chl←label_cluster(c_chl)
42     apply l'_c←alter_label(c, l_c)
43    end if
44   else // Steps 44–46: simple merging
45    apply c'←merge_doc(c, α+β+γ)
46   end if
47 end for
```

4.3 Complexity analysis

This subsection presents complexity analysis to check the independence of the TIE algorithm on the clustering algorithm used for the generation of the taxonomy. The time efficiency of taxonomy evolution in comparison to taxonomy regeneration is also checked through complexity analysis. Let n denote the number of documents in a data set. The time complexity of the HAC algorithm (agglomerative approach) is given as a function of $n \times n$ distance/similarity matrix, i.e., n^2 (Steinbach et al., 2000). The time complexity of the bisect K-means algorithm (divisive approach) is given as a function of the

partition parameter and the number of documents to partition. Let the partition parameter be denoted as K and the number of documents to partition as n. We can say that the time complexity of bisect K-means is Kn (Steinbach et al., 2000). Let x denote the number of hierarchical clusters formed as a result of applying hierarchical clustering (HAC or bisect K-means), where we logically assume $x \ll n$. Let n' denote the number of newly arriving documents, where for shorter time intervals we assume $n' \ll n$.

Time complexity of TGP: As hierarchy formation is a major step involved in the generation of a taxonomy, we ignore the cost associated with other steps of taxonomy generation. Let the time complexities of TGP in the case of HAC and bisect K-means be denoted as $\tau_{\mathrm{TGP}}(h)$ and $\tau_{\mathrm{TGP}}(b)$, respectively. They are approximated as

$$\tau_{\mathrm{TGP}}(h) \approx n^2, \tag{4}$$

$$\tau_{\mathrm{TGP}}(b) \approx Kn. \tag{5}$$

Time complexity of TIE: For the TIE algorithm, we can say that for each newly arriving document, x comparisons are made to identify the closest cluster. If we ignore the time taken in applying the reorganization operators because of the linear trend, then let the time complexities of TIE in the case of HAC and bisect K-means be denoted as $\tau_{\mathrm{TIE}}(h)$ and $\tau_{\mathrm{TIE}}(b)$, respectively. They are approximated as

$$\tau_{\mathrm{TIE}}(h) \approx xn', \tag{6}$$

$$\tau_{\mathrm{TIE}}(b) \approx xn'. \tag{7}$$

TIE is independent of the clustering algorithm used in TGP: From Eqs. (6) and (7), we can see that the time complexities of the TIE algorithm in both clustering algorithms (i.e., HAC and bisect K-means) are the same. This shows that the TIE algorithm is independent of the clustering algorithm used for the generation of the taxonomy and serves as a layer over any of the existing clustering-based taxonomy generation techniques.

Evolution versus regeneration: We can approximate the time of regeneration using Eqs. (4) and (5) in the case of HAC and bisect K-means, respectively. Let the time complexities of regeneration in the case of HAC and bisect K-means be denoted as $\tau_{\mathrm{r}}(h)$ and $\tau_{\mathrm{r}}(b)$, respectively. They are given as

$$\tau_{\mathrm{r}}(h) \approx n^2 + n^2, \tag{8}$$

$$\tau_{\mathrm{r}}(b) \approx Kn + Kn. \tag{9}$$

On the other hand, we can approximate the time of evolution by combining Eqs. (4) and (6) in the case of HAC, and by combining Eqs. (5) and (7) in the case of bisect K-means. Let the time complexities of evolution in the case of HAC and bisect K-means be denoted as $\tau_{\mathrm{e}}(h)$ and $\tau_{\mathrm{e}}(b)$, respectively. They are given as

$$\tau_{\mathrm{e}}(h) \approx n^2 + xn', \tag{10}$$

$$\tau_{\mathrm{e}}(b) \approx Kn + xn'. \tag{11}$$

In the case of HAC using Eqs. (8) and (10), we can see that $n^2+xn' \ll n^2+n^2$ (as $x \ll n$ and $n' \ll n$), so the taxonomy evolution time is much less than that of taxonomy regeneration. In the case of bisect K-means using Eqs. (9) and (11), we can see that $Kn+xn' \ll Kn+Kn$ (as $x \ll n$ and $n' \ll n$), so the taxonomy evolution time is much less than that of taxonomy regeneration.

5 Evaluation

The taxonomy incremental evolution was evaluated in comparison to taxonomy regeneration from scratch on a text data set of scholarly articles selected from the computing domain. The details of the data set, evaluation metrics, experiments, and their results are given next.

5.1 Data set specification

From the computing domain, 400 scholarly articles in pdf format were randomly downloaded from the ACM Digital Library (http://dl.acm.org/). These documents were then converted into text format using the Apache Tika (https://tika.apache.org/). The ACM Computing Classification System (CCS) (http://www.acm.org/about/class/2012), which was a standard topic hierarchy for the computing domain, was adopted as a reference taxonomy.

5.2 Metrics

The taxonomies obtained through evolution and regeneration approaches underwent three types of evaluation: time-based, quality-based, and quality- and time-based. The first two methods are the same as adopted in our earlier work (Irfan and Khan, 2016), but the last is the newly proposed method.

5.2.1 Time-based evaluation

The time-based evaluation was done by observing the runtime of applications associated with both approaches (Irfan and Khan, 2016). Let it be denoted as t_{run}.

5.2.2 Quality-based evaluation

The quality-based evaluation was based on the taxonomy's lexical and hierarchical quality. The lexical quality assesses the quality of taxonomy labels, whereas the hierarchical quality assesses the quality of hierarchical associations among the taxonomic nodes. For lexical quality assessment, we adopted lexical precision (LP), lexical recall (LR), and lexical *F*-measure (LF). These have been used in many existing taxonomy generation techniques (Cimiano et al., 2005; Dietz et al., 2012). The lexical quality matches a label in both computed and reference taxonomies. No matter what kind of relationship a label possesses with other labels in both taxonomies, it is considered a match. We assessed the lexical quality of both the evolving and regenerated taxonomies, in comparison to the reference taxonomy, using these metrics. For hierarchical quality assessment, we adopted hierarchical precision (HP), hierarchical (HR), and hierarchical *F*-measure (HF) proposed in Irfan and Khan (2016). For a label that appears in both computed and reference taxonomies, the hierarchical quality compares its parent's and ancestors' labels in both taxonomies to identify a match. The formulas and definitions for the lexical and hierarchical quality metrics are given in Tables 2 and 3, respectively.

5.2.3 Quality- and time-based evaluation

We propose a new metric for assessing the efficiency of taxonomy evolution in comparison to taxonomy regeneration. The quality-time ratio combines the quality- and time-based metrics and is the rate of improvement in taxonomy quality per unit time. Let the quality-time ratio be denoted as r_{QT}, defined as

Table 2 Lexical quality metrics

Measure	Formula	Description
Lexical precision (LP)	$\text{LP}=\lvert L_{\text{C}}\cap L_{\text{R}}\rvert/\lvert L_{\text{C}}\rvert$	Calculating the percentage of labels in computed (regenerated/evolved) taxonomy that are also in reference taxonomy (Irfan and Khan, 2016)
Lexical recall (LR)	$\text{LR}=\lvert L_{\text{C}}\cap L_{\text{R}}\rvert/\lvert L_{\text{R}}\rvert$	Calculating the percentage of labels in reference taxonomy that are also in computed (regenerated/evolved) taxonomy (Irfan and Khan, 2016)
Lexical *F*-measure (LF)	$\text{HF}=\dfrac{2(\text{LP}\times\text{LR})}{\text{LP}+\text{LR}}$	The harmonic mean of lexical precision and recall (Irfan and Khan, 2016)

L_{C} is a set of all labels in computed (regenerated/evolved) taxonomy, and L_{R} is a set of all labels in reference taxonomy

Table 3 Hierarchical quality metrics

Measure	Formula	Description
Hierarchical precision (HP)	$\text{HP}=\dfrac{\sum_{i=1}^{\lvert L_{\text{C}}\cap L_{\text{R}}\rvert}\lvert \text{par}_{\text{ai}}[T_{\text{C}}]\cap \text{anc}_{\text{ai}}[T_{\text{R}}]\rvert}{\sum_{i=1}^{\lvert L_{\text{C}}\cap L_{\text{R}}\rvert}\lvert \text{par}_{\text{ai}}[T_{\text{C}}]\rvert}$	Calculating the percentage of all parent associations in computed (regenerated/evolved) taxonomy that also appear in reference taxonomy as parent or ancestor associations (Irfan and Khan, 2016)
Hierarchical recall (HR)	$\text{HR}=\dfrac{\sum_{i=1}^{\lvert L_{\text{C}}\cap L_{\text{R}}\rvert}\lvert \text{par}_{\text{ai}}[T_{\text{R}}]\cap \text{anc}_{\text{ai}}[T_{\text{C}}]\rvert}{\sum_{i=1}^{\lvert L_{\text{C}}\cap L_{\text{R}}\rvert}\lvert \text{par}_{\text{ai}}[T_{\text{R}}]\rvert}$	Calculating the percentage of all parent associations in reference taxonomy that also appear in computed (regenerated/evolved) taxonomy as parent or ancestor associations (Irfan and Khan, 2016)
Hierarchical *F*-measure (HF)	$\text{HF}=\dfrac{2(\text{HP}\times\text{HR})}{\text{HP}+\text{HR}}$	The harmonic mean of hierarchical precision and recall (Irfan and Khan, 2016)

$L_{\text{C}}\cap L_{\text{R}}$ is a set of common labels in computed (regenerated/evolved) taxonomy and reference taxonomy; $\text{par}_{\text{ai}}[T']$ is a set of all parent associations and $\text{anc}_{\text{ai}}[T']$ is a set of all ancestor associations for the i^{th} term a_i in T', where T' can be computed (regenerated/evolved) taxonomy, i.e., T_{C}, or reference taxonomy, i.e., T_{R}, and note that $\text{par}_{\text{ai}}[T']\subseteq\text{anc}_{\text{ai}}[T']$

$$r_{\mathrm{QT}} = \frac{1}{t_{\mathrm{run}}} \sum \text{Quality-based metrics.} \quad (12)$$

The r_{QT} value is calculated by summing the values of both lexical and hierarchical quality metrics to determine the overall improvement in taxonomy quality per unit time (i.e., the runtime), in the case of evolution and regeneration. Although there is no limit on the range of values for this measure, in our case as the quality-based metrics range between 0 and 1 and time (in min) is greater than 1 in all cases, we are obtaining values in the range of 0–1.

5.3 Experiments

The experiments were divided into three parts: (1) An initial taxonomy was generated using a base data set to be adopted as a foundation for performing regeneration and evolution. (2) With the addition of new documents, taxonomy regeneration was performed for data sets of different sizes to measure the quality and time metrics. (3) With the addition of new documents, a taxonomy evolution was performed for data sets of different sizes to measure the quality and time metrics. The details of the experiments are given next.

5.3.1 Initial taxonomy generation

Since the main purpose was to check the improvement of evolution over regeneration rather than the improvement in the taxonomy generation process, we adopted a simple approach towards the generation of taxonomy. The type of data set is textual, and therefore we adopted methods suitable for the taxonomy generation process (TGP) of text data. Note that the adopted TGP is very similar to the one adopted in our earlier work (Irfan and Khan, 2016), with the exception of adopting two different types of hierarchical clustering approaches (i.e., agglomerative and divisive), so the independence of the TIE algorithm on the clustering algorithm used in the taxonomy generation process can be checked empirically. TGP was applied on a set of 200 randomly selected documents to generate an initial taxonomy. The steps involved in generating an initial taxonomy are as follows:

1. TGP performed natural language processing of tokenization, stemming, stop word removal, and parsing to extract nounphrases. These nounphrases formed the vocabulary of distinct terms.

2. Vector space modeling was then applied to express the vocabulary and the data set in the form of a vector model.

3. To evaluate the independence of the TIE algorithm from the clustering algorithm used in TGP, we performed experiments with two types of hierarchical clustering algorithms: HAC (agglomerative approach) and bisect *K*-means (divisive approach). For HAC, we implemented three different variants, i.e., single link, complete link, and average link, to select the best one. It was found through various test runs that the average link produced the best results for generating the taxonomy. The experimental results for different variants of HAC presented in Table 4 show the average precision, recall, and *F*-measure (lexical) values along with the standard deviations σ for the generated taxonomy on the data set of 200 documents. After this step, we had two hierarchical structures (i.e., hierarchical organization of unlabeled clusters) obtained from HAC (average link) and bisect *K*-means, respectively, and the evaluation was performed for the taxonomies obtained from each of these hierarchical structures. For bisect *K*-means, we approximated the value of $K \approx \sqrt{n/2}$ (as $K \ll n$) (Steinbach et al., 2000) where n is the number of documents to partition.

Table 4 Comparison of different variants of HAC based on lexical quality metrics for a data set of 200 documents

HAC variant	LP (average±σ)	LR (average±σ)	LF
Average link	0.580±0.010	0.457±0.004	0.511
Complete link	0.437±0.019	0.347±0.072	0.387
Single link	0.319±0.006	0.234±0.010	0.270

4. For labeling the hierarchical structures produced through both hierarchical clustering algorithms, the top 25 terms (based on f_{t} and f_{id} scores) of the cluster centroid were adopted as labels and they were further pruned to keep general terms as labels for parent clusters and specific terms as labels for child clusters. The reason for choosing 25 terms to label a cluster was that too few/many terms made the cluster (i.e., taxonomic node) hard to interpret. This step produced two taxonomies, each corresponding to one of the two hierarchical structures.

5.3.2 Taxonomy regeneration

To evaluate the regeneration of the taxonomy from scratch when new documents were added to the data set, the set of 200 randomly selected documents used to generate the initial taxonomy was adopted as a base data set. Two sets of experiments were performed: one for evaluating the regeneration of taxonomy in the case of HAC, and the other for evaluating the regeneration of taxonomy in the case of bisect *K*-means. For HAC, the first 20 documents were added to the data set of 200 documents and the TGP using HAC was performed from scratch for 220 documents. As a result, the initial taxonomy of 200 documents was replaced with the new taxonomy for 220 documents. After that, more than 30 documents were added to the data set of 220 documents and the TGP using HAC was performed from scratch for 250 documents. As a result, the existing taxonomy of 220 documents was replaced with the new taxonomy for 250 documents. Experiments like this were repeated by adding more than 40, 50, and 60 documents in the data sets of 250, 290, and 340 documents, respectively. Each of the regenerated taxonomies was evaluated in comparison to ACM CSS to obtain the results for the lexical and hierarchical quality metrics, as well as the runtime. The same set of experiments was repeated to evaluate the regeneration of taxonomy in the case of bisect *K*-means.

5.3.3 Taxonomy evolution

To evaluate the evolution of taxonomy, a similar set of experiments was performed as in the case of taxonomy regeneration (Section 5.3.2). Starting off with the initial taxonomy of 200 documents obtained using HAC, new documents (in sets of 20, 30, 40, 50, and 60) were added to the system and the TIE algorithm was run for data sets of different sizes. The lexical and hierarchical quality metrics, as well as the runtime of the taxonomy, evolving as a result of each run of the TIE algorithm, were noted. The same set of experiments was repeated to evaluate the evolution of taxonomy in the case of bisect *K*-means.

5.4 Results

Tables 5–7 show the runtime t_{run}, lexical quality metrics (i.e., LP, LR, and LF), and hierarchical quality metrics (i.e., HP, HR, and HF) obtained as results of taxonomy regeneration (Section 5.3.2) and taxonomy evolution (Section 5.3.3) in the case of HAC, whereas Tables 8–10 show the runtime, i.e., t_{run}, lexical quality metrics (i.e., LP, LR, and LF), and hierarchical quality metrics (i.e., HP, HR, and HF) obtained as results of taxonomy regeneration (Section 5.3.2) and taxonomy evolution (Section 5.3.3) in the case of bisect *K*-means. We calculated the average values for each measure by performing several runs of the experiments and randomly selecting the document sets for each run. All results mentioned in Tables 5–10 state the average values along with the standard deviation σ. The time and quality metrics listed in Tables 5–10 were used to calculate the quality-time ratio (i.e., r_{QT}), whose graphs were shown in Figs. 2a–2c in the case of HAC and in Figs. 3a–3c in the case of bisect *K*-means.

Table 5 Results of time-based evaluation in case of HAC

With HAC	T_{run} ($\pm\sigma$) (min)	
	TGP	TIE
200	22.842 (±0.308)	–
200+20=220	24.725 (±0.080)	4.905 (±0.232)
220+30=250	28.030 (±0.170)	7.182 (±0.245)
250+40=290	46.760 (±0.009)	9.810 (±0.180)
290+50=340	59.390 (±0.140)	15.601 (±0.238)
340+60=400	70.088 (±0.437)	20.675 (±0.337)

It was observed that the number of documents used to form the existing taxonomy and the number of newly added documents in the data set were the two factors that could mainly influence the evolution of the taxonomy. Let them be denoted as f_1 and f_2, respectively. We performed sensitivity analysis to check the impact of varying these factors on the evolution of the taxonomy. Two sets of additional experiments were performed for this purpose: first, taxonomy evolution was performed for data sets of different sizes by varying f_1 and keeping f_2 fixed; secondly, taxonomy evolution was performed for data sets of different sizes by keeping f_1 fixed and varying f_2. t_{run}, lexical quality metrics (i.e., LP, LR, and LF), and hierarchical quality metrics (i.e., HP, HR, and HF) obtained as results of these experiments are shown in Tables 11–14. Like Tables 5–10, the values in Tables 11–14 are based on several runs of the experiments and state the average values along with the standard deviation σ.

Table 6 Results of lexical quality based evaluation in case of HAC

Data set	LP (average±σ)		LR (average±σ)		LF	
	TGP	TIE	TGP	TIE	TGP	TIE
200	0.580±0.010	–	0.457±0.004	–	0.511	–
200+20=220	0.581±0.020	0.522±0.008	0.440±0.001	0.425±0.013	0.501	0.469
220+30=250	0.539±0.017	0.516±0.010	0.434±0.004	0.415±0.035	0.481	0.460
250+40=290	0.532±0.009	0.501±0.065	0.424±0.002	0.402±0.056	0.472	0.446
290+50=340	0.512±0.042	0.499±0.061	0.409±0.002	0.392±0.014	0.455	0.439
340+60=400	0.507±0.004	0.490±0.061	0.411±0.003	0.401±0.045	0.454	0.441

Table 7 Results of hierarchical quality based evaluation in case of HAC

Data set	HP (average±σ)		HR (average±σ)		HF	
	TGP	TIE	TGP	TIE	TGP	TIE
200	0.351±0.099	–	0.330±0.010	–	0.340	–
200+20=220	0.341±0.007	0.338±0.006	0.323±0.010	0.283±0.001	0.331	0.308
220+30=250	0.346±0.068	0.307±0.004	0.319±0.002	0.275±0.013	0.332	0.290
250+40=290	0.326±0.004	0.304±0.020	0.298±0.033	0.277±0.003	0.311	0.292
290+50=340	0.312±0.003	0.295±0.003	0.295±0.003	0.276±0.004	0.304	0.285
340+60=400	0.305±0.009	0.291±0.086	0.286±0.006	0.264±0.003	0.295	0.277

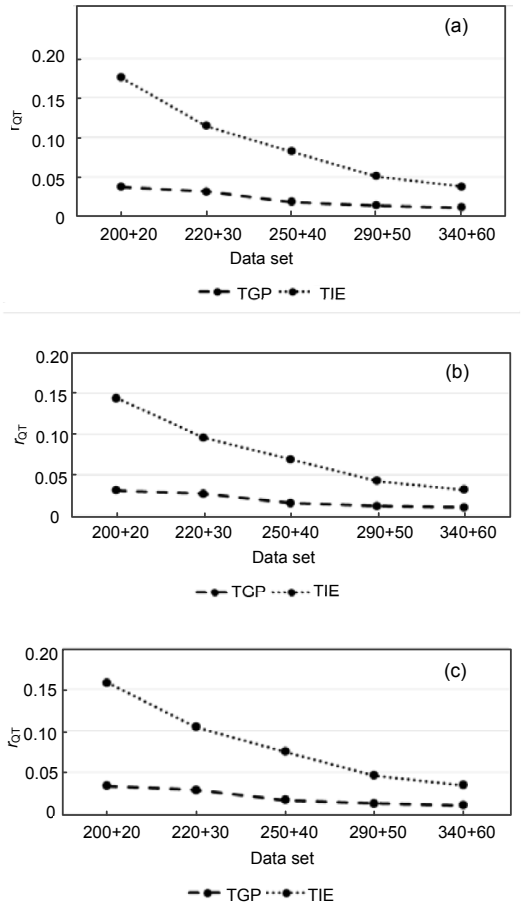

Fig. 2 r_{QT} in case of hierarchical agglomerative clustering: (a) LP+HP; (b) LR+HR; (c) LF+HF

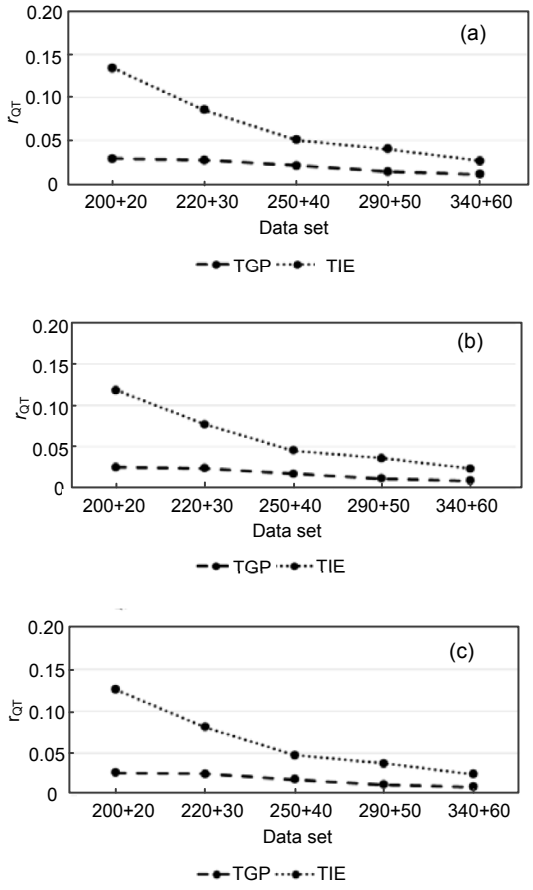

Fig. 3 r_{QT} in case of bisect *K*-means: (a) LP+HP; (b) LR+HR; (c) LF+HF

5.5 Discussion

Observations on the comparison of taxonomy evolution with taxonomy regeneration presented in Section 5.4 are given below:

1. We can see that the time-based evaluation, given in Table 5 in the case of HAC and Table 8 in the case of bisect *K*-means, clearly reflects the essence of the incremental evolution of taxonomy in comparison to regeneration. Note that, as expected, the amount of time it takes to evolve is less than that of regeneration, which is also demonstrated through the complexity analysis presented in Section 4.3. This difference in time is more pronounced, particularly with the increase in the data set size.

2. In the case of quality-based evaluation, precision, recall, and *F*-measure values (both lexical and hierarchical) for regeneration are slightly better than those obtained as a result of evolution, as shown in Tables 6 and 7 in the case of HAC and in Tables 9 and 10 in the case of bisect *K*-means. Moreover, from the graphs shown in Figs. 4a and 4b in the case of HAC and in Figs. 5a and 5b in the case of bisect *K*-means, we can see that the lexical measures give better results than the hierarchical measures in all cases for both regeneration and evolution.

3. In the case of quality- and time-based evaluation, we can see from all the graphs shown in Figs. 2 and 3 that r_{QT} is higher in the case of evolution for data sets of different sizes. However, the ratio shows a steady behavior for regeneration, which seems not much affected by the increase in the data set size. For the evolution case, r_{QT} is particularly higher when the data set is small and the number of newly added documents is smaller. As more documents are added to the system, r_{QT} for evolution shows a decline while maintaining an edge over regeneration.

4. Additionally, we can see that the values for the quality measures in the case of HAC (Tables 6 and 7) appear to be better than those of bisect *K*-means (Tables 9 and 10) in all the cases for both regeneration and evolution. However, bisect *K*-means has a slight edge over the runtime compared to that of HAC in all cases for both regeneration and evolution, as can be seen from Tables 5 and 8.

Observations related to the sensitivity analysis of taxonomy evolution presented in Section 5.4 are given below:

Table 8 Results of time-based evaluation in case of bisect *K*-means

Data set	t_{run} ($\pm\sigma$) (min)	
	TGP	TIE
200	21.157 (±0.012)	–
200+20=220	22.569 (±0.310)	4.438 (±0.010)
220+30=250	23.981 (±0.048)	6.479 (±0.011)
250+40=290	30.642 (±0.211)	10.911 (±0.020)
290+50=340	45.093 (±0.224)	13.714 (±0.014)
340+60=400	57.503 (±0.126)	19.704 (±0.050)

Table 9 Results of lexical quality-based evaluation in case of bisect *K*-means

Data set	LP (average±σ)		LR (average±σ)		LF	
	TGP	TIE	TGP	TIE	TGP	TIE
200	0.445±0.007	–	0.323±0.002	–	0.374	–
200+20=220	0.405±0.009	0.380±0.015	0.338±0.086	0.327±0.020	0.369	0.352
220+30=250	0.402±0.010	0.354±0.012	0.351±0.005	0.323±0.002	0.375	0.338
250+40=290	0.408±0.011	0.358±0.001	0.317±0.003	0.292±0.016	0.356	0.322
290+50=340	0.395±0.012	0.354±0.012	0.311±0.094	0.310±0.002	0.347	0.331
340+60=400	0.390±0.037	0.338±0.004	0.304±0.083	0.296±0.111	0.342	0.316

Table 10 Results of hierarchical quality-based evaluation in case of bisect *K*-means

Data set	HP (average±σ)		HR (average±σ)		LF	
	TGP	TIE	TGP	TIE	TGP	TIE
200	0.244±0.089	–	0.217±0.054	–	0.229	–
200+20=220	0.241±0.011	0.217±0.013	0.228±0.032	0.198±0.011	0.234	0.207
220+30=250	0.242±0.008	0.202±0.008	0.220±0.051	0.177±0.052	0.231	0.189
250+40=290	0.233±0.002	0.197±0.004	0.204±0.076	0.203±0.095	0.218	0.199
290+50=340	0.218±0.031	0.196±0.002	0.193±0.007	0.185±0.065	0.205	0.190
340+60=400	0.208±0.053	0.179±0.014	0.188±0.006	0.162±0.034	0.197	0.170

Table 11 Sensitivity analysis of the TIE algorithm by varying f_1 and keeping f_2 fixed in case of HAC

f_1	f_2	Data set	t_{run} (±σ) (min)	LP (average ±σ)	LR (average±σ)	LF	HP (average±σ)	HR (average±σ)	HF
200	30	230	5.024 (±0.093)	0.528±0.017	0.438±0.312	0.479	0.355±0.101	0.326±0.088	0.340
230	30	260	7.298 (±0.028)	0.526±0.003	0.430±0.077	0.473	0.348±0.002	0.322±0.011	0.334
260	30	290	10.016 (±0.005)	0.519±0.015	0.424±0.008	0.467	0.346±0.014	0.319±0.004	0.332
290	30	320	12.103 (±0.016)	0.514±0.163	0.417±0.009	0.460	0.333±0.631	0.314±0.014	0.323
320	30	350	14.876 (±0.096)	0.511±0.520	0.409±0.018	0.454	0.329±1.430	0.305±0.654	0.317
350	30	380	18.955 (±0.154)	0.502±0.612	0.405±0.122	0.448	0.321±0.996	0.300±0.216	0.310

Table 12 Sensitivity analysis of the TIE algorithm by keeping f_1 fixed and varying f_2 in case of HAC

f_1	f_2	Data set	t_{run} (±σ) (min)	LP (average±σ)	LR (average±σ)	LF	HP (average±σ)	HR (average±σ)	HF
200	30	230	5.024 (±0.093)	0.528±0.017	0.438±0.312	0.479	0.355±0.101	0.326±0.088	0.34
200	60	260	9.986 (±0.021)	0.503±0.188	0.410±0.512	0.452	0.349±0.220	0.317±0.096	0.33
200	90	290	12.645 (±0.003)	0.499±0.019	0.400±0.013	0.444	0.328±0.060	0.314±0.054	0.32
200	120	320	19.832 (±0.005)	0.435±0.016	0.375±0.305	0.403	0.294±0.089	0.285±1.001	0.28
200	150	350	22.608 (±0.014)	0.421±0.550	0.331±0.007	0.371	0.254±0.005	0.263±0.043	0.25
200	180	380	28.111 (±0.119)	0.408±0.715	0.311±0.013	0.353	0.248±0.021	0.247±0.167	0.24

Table 13 Sensitivity analysis of the TIE algorithm by varying f_1 and keeping f_2 fixed in case of bisect *K*-means

f_1	f_2	Data set	t_{run} (±σ) (min)	LP (average±σ)	LR (average±σ)	LF	HP (average±σ)	HR (average±σ)	HF
200	30	230	4.501 (±0.185)	0.377±0.003	0.330±0.057	0.352	0.257±0.002	0.219±0.076	0.236
230	30	260	7.001 (±0.009)	0.365±0.018	0.325±0.041	0.344	0.250±0.013	0.213±0.041	0.230
260	30	290	10.965 (±0.014)	0.360±0.154	0.311±0.190	0.334	0.236±0.009	0.202±0.199	0.218
290	30	320	11.909 (±0.006)	0.359±0.201	0.304±0.301	0.329	0.232±0.198	0.200±0.031	0.215
320	30	350	14.132 (±0.014)	0.347±0.109	0.301±0.188	0.322	0.219±0.005	0.194±0.005	0.206
350	30	380	16.502 (±0.176)	0.342±0.005	0.299±0.004	0.319	0.206±0.203	0.191±0.042	0.198

Table 14 Sensitivity analysis of the TIE algorithm by keeping f_1 fixed and varying f_2 in case of bisect *K*-means

f_1	f_2	Data set	t_{run} (±σ) (min)	LP (average±σ)	LR (average±σ)	LF	HP (average±σ)	HR (average±σ)	HF
200	30	230	4.501 (±0.185)	0.377±0.003	0.330±0.057	0.352	0.257±0.002	0.219±0.076	0.236
200	60	260	8.614 (±0.064)	0.362±0.011	0.316±0.0543	0.337	0.234±0.109	0.214±0.003	0.224
200	90	290	13.113 (±0.056)	0.359±0.009	0.301±0.087	0.327	0.200±0.122	0.185±0.001	0.192
200	120	320	17.340 (±0.088)	0.328±0.055	0.276±0.121	0.300	0.174±0.034	0.166±0.021	0.170
200	150	350	21.877 (±0.176)	0.315±0.062	0.263±0.139	0.287	0.155±0.142	0.154±0.046	0.154
200	180	380	27.190 (±0.316)	0.303±0.006	0.244±0.046	0.270	0.121±0.078	0.138±0.008	0.129

1. It can be seen through the results presented in Table 11 in the case of HAC and Table 13 in the case of bisect *K*-means that when f_1 is varied and f_2 is kept fixed, increase in time and decrease in taxonomy quality (both lexical and hierarchical) are showing a steadily deteriorating behavior with increasing values of f_1.

2. On the other hand, it can be seen through the results presented in Table 12 in the case of HAC and Table 14 in the case of bisect *K*-means that when f_1 is kept fixed and f_2 is varied, the increase in time and decrease in taxonomy quality (both lexical and hierarchical) are showing a sharply deteriorating behavior with increasing values of f_2.

3. Based on this analysis, we can say that f_2 is the factor that has influenced the time and quality metrics of the TIE algorithm more than f_1. It seems that the evolution is advantageous when new documents

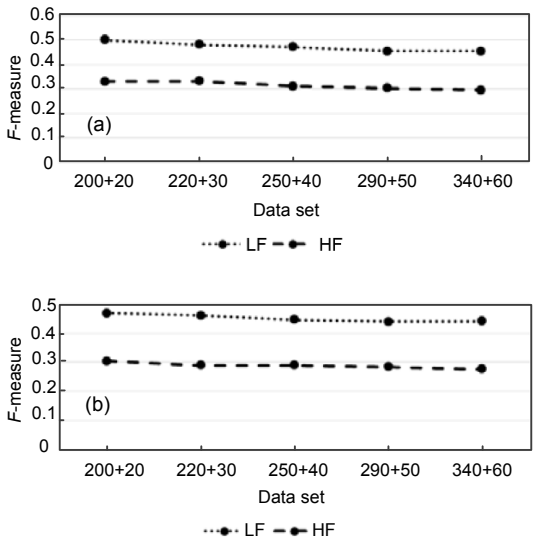

Fig. 4 Lexical versus hierarchical *F*-measure in case of hierarchical agglomerative clustering: (a) TGP; (b) TIE

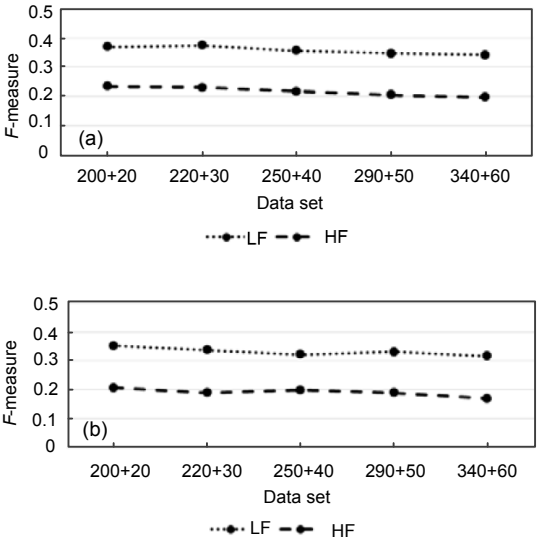

Fig. 5 Lexical versus hierarchical *F*-measure in case of bisect *K*-means: (a) TGP; (b) TIE

arrive in small chunk. If data arrives in large chunk, then there is a possibility that the evolution performance drops to a level at which it can lose its advantage over taxonomy regeneration.

The general observations related to the results presented in Section 5.4 are given below:

1. The application of the TIE algorithm on existing taxonomies generated through agglomerative and divisive approaches to hierarchical clustering without making any adjustment shows that the proposed algorithm easily serves as a layer over any of the existing clustering-based taxonomy generation techniques. This is also demonstrated through the complexity analysis presented in Section 4.3.

2. As far as the scalability of the TIE algorithm is concerned, overall, we can see a deteriorating trend in time and quality metrics with the increasing size of the data set used to form the taxonomy in all the tests performed, i.e., the test results showing the comparison of taxonomy evolution with regeneration (Tables 5–10) and the test results showing sensitivity analysis of taxonomy evolution (Tables 11–14). This is even the case for taxonomy regeneration from scratch in all the test results shown in Tables 5–10.

6 Conclusions and future work

In this paper, a TIE algorithm is designed to incrementally evolve with an existing taxonomy to adjust changes that occur in underlying data. The algorithm takes an existing taxonomy, the respective hierarchical structure, and newly arriving documents as its input. It identifies the closest cluster in the hierarchy for each of the newly arriving documents based on the similarity among them. It then applies various reorganization operators to adjust the newly arriving documents in the hierarchy and finally complete evolution of the existing taxonomy. The algorithm is compared with a taxonomy regeneration approach based on complexity analysis and empirical evaluation. The complexity analysis of the algorithm demonstrates that it is better than regeneration in terms of time, and that it is independent of the underlying clustering approach used for taxonomy generation. The empirical evaluation is performed using a data set of scholarly articles selected from the computing domain, based on three parameters. The time-based evaluation clearly shows that the TIE algorithm takes less time to adjust new documents in an existing taxonomy. Although taxonomy regeneration shows better results in terms of quality, the quality-time ratio of the TIE algorithm indicates that the rate of improvement in taxonomy quality per unit time is better than that of regeneration. Also, the results of sensitivity analysis of the TIE algorithm show that it performs better when the new data arrives in small chunks.

The proposed TIE algorithm produces an evolving taxonomy in a shorter time to represent changes occurring in underlying data for effective use of the taxonomy. The quality of the evolving taxonomy can be further enhanced by incorporating semantics in identifying the closest cluster for newly arriving documents. Furthermore, the proposed algorithm requires the setting of a time interval value between two consecutive runs of the evolution process. Currently, for the test data set, the running of the TIE algorithm is self-controlled based on the addition of new documents, but for a real data set it should be set depending upon the update characteristics (i.e., the number of newly arriving documents per unit time) of the data set. In general, for the data set for which new documents are frequently arriving, the time interval for the evolution should be set to be less than the time interval for the data set for which the arrival of new documents is not very frequent. Moreover, the deteriorating trend in time and quality metrics with the increasing size of the data set used to form the taxonomy, shows that the scalability aspect of the proposed solution should be improved further. In the future, the proposed algorithm can be applied to discover emerging trends and patterns in social media where data is rapidly evolving.

References

Baeza-Yates R, Ribeiro-Neto B, 2011. Modern Information Retrieval: the Concepts and Technology Behind (2nd Ed.). Pearson Education Limited, New York, USA.

Blumberg R, Atre S, 2003. The problem with unstructured data. *DM Rev*, 13(2):42-46.

Camiña SL, 2010. A Comparison of Taxonomy Generation Techniques Using Bibliometric Methods: Applied to Research Strategy Formulation. MS Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.

Carmel D, Roitman H, Zwerdling N, 2009. Enhancing cluster labeling using Wikipedia. Proc 32nd Int ACM SIGIR Conf on Research and Development in Information Retrieval, p.139-146. https://doi.org/10.1145/1571941.1571967

Cha SH, 2007. Comprehensive survey on distance/similarity measures between probability density functions. *Int J Math Models Methods Appl Sci*, 1(4):300-307.

Cimiano P, Hotho A, Staab S, 2005. Learning concept hierarchies from text corpora using formal concept analysis. *J Artif Intell Res*, 24(1):305-339.

Dawelbait G, Mezher T, Woon WL, et al., 2010. Taxonomy based trend discovery of renewable energy technologies in desalination and power generation. Proc Technology Management for Global Economic Growth, p.1-8.

Deerwester S, Dumais ST, Furnas GW, et al., 1990. Indexing by latent semantic analysis. *J Am Soc Inform Sci Technol*, 41(6):391-407. https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

Dietz EA, Vandic D, Frasincar F, 2012. TaxoLearn: a semantic approach to domain taxonomy learning. Proc IEEE/WIC/ACM Int Conf on Web Intelligence and Intelligent Agent Technology, p.58-65. https://doi.org/10.1109/WI-IAT.2012.129

Enhanced Taxonomy Generation. USA Patent 20 100 274 733.

Fountain T, Lapata M, 2012. Taxonomy induction using hierarchical random graphs. Proc Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, p.466-476.

Glover E, Pennock DM, Lawrence S, et al., 2002. Inferring hierarchical descriptions. Proc 11th Int Conf on Information and Knowledge Management, p.507-514. https://doi.org/10.1145/584792.584876

Hedden H, 2010. The Accidental Taxonomist. Information Today Inc., Medford, New Jersey, USA, p.18-28.

Irfan R, Khan S, 2016. TIE: an algorithm for incrementally evolving taxonomy for text data. Proc 15th IEEE Int Conf on Machine Learning and Applications, p.687-692. https://doi.org/10.1109/ICMLA.2016.0121

Jain AK, Murty MN, Flynn PJ, 1999. Data clustering: a review. *ACM Comput Surv*, 31(3):264-323. https://doi.org/10.1145/331499.331504

Kashyap V, Ramakrishnan C, Thomas C, et al., 2005. TaxaMiner: an experimentation framework for automated taxonomy bootstrapping. *Int J Web Grid Serv*, 1(2): 240-266. https://doi.org/10.1504/IJWGS.2005.008322

Koff W, Gustafson P, 2011. Data Revolution. Technical Report, Computer Sciences Corporation Leading Edge Forum.

Kumar AA, Chandrasekhar S, 2012. Text data pre-processing and dimensionality reduction techniques for document clustering. *Int J Eng Res Technol*, 1(5):1-6.

Lefever E, 2015. LT3: a multi-modular approach to automatic taxonomy construction. Proc 9th Int Workshop on Semantic Evaluation, p.944-948.

Li T, Anand SS, 2009. Exploiting domain knowledge by automated taxonomy generation in recommender systems. Proc 10th Int Conf on E-commerce and Web Technologies, p.120-131.

Manning CD, Raghavan P, Schütze H, 2008. Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA.

Marcacini RM, Rezende SO, 2010. Incremental construction of topic hierarchies using hierarchical term clustering. Proc 22nd Int Conf on Software Engineering and Knowledge Engineering, p.553-558.

Medelyan O, Manion S, Broekstra J, et al., 2013. Constructing a focused taxonomy from a document collection. Proc 10th Int Conf on the Semantic Web: Semantics and Big Data, p.367-381.

https://doi.org/10.1007/978-3-642-38288-8_25

Meijer K, Frasincar F, Hogenboom F, 2014. A semantic approach for extracting domain taxonomies from text. *Dec Support Syst*, 62:78-93. https://doi.org/10.1016/j.dss.2014.03.006

Muller A, Dorre J, Gerstl P, et al., 1999. The TaxGen framework: automating the generation of a taxonomy for a large document collection. Proc 32nd Annual Hawaii Int Conf on Systems Sciences, Article 2034.

Nadkarni PM, Ohno-Machado L, Chapman WW, 2011. Natural language processing: an introduction. *J Am Med Inform Assoc*, 18(5):544-551. https://doi.org/10.1136/amiajnl-2011-000464

Neshati M, Alijamaat A, Abolhassani H, et al., 2007. Taxonomy learning using compound similarity measure. Proc IEEE/WIC/ACM Int Conf on Web Intelligence, p.487-490. https://doi.org/10.1109/WI.2007.135

Paukkeri MS, García-Plaza AP, Fresno V, et al., 2012. Learning a taxonomy from a set of text documents. *Appl Soft Comput*, 12(3):1138-1148. https://doi.org/10.1016/j.asoc.2011.11.009

Qi XG, Yin DW, Xue ZZ, et al., 2010. Choosing your own adventure: automatic taxonomy generation to permit many paths. Proc 19th ACM Int Conf on Information and Knowledge Management, p.1853-1856. https://doi.org/10.1145/1871437.1871746

Sánchez D, Moreno A, 2004. Automatic generation of taxonomies from the WWW. Proc 5th Int Conf on Practical Aspects of Knowledge Management, p.208-219. https://doi.org/10.1007/978-3-540-30545-3_20

Sclano F, Velardi P, 2007. TermExtractor: a web application to learn the common terminology of interest groups and research communities. Proc 3rd Int Conf on Interoperability for Enterprise Software and Applications p.85-94.

Spangler WS, Kreulen JT, Newswanger JF, 2006. Machines in the conversation: detecting themes and trends in informal communication streams. *IBM Syst J*, 45(4):785-799. https://doi.org/10.1147/sj.454.0785

Steinbach M, Karypis G, Kumar V, 2000. A comparison of document clustering techniques. World Text Mining Conf, p.1-2.

Sujatha R, Krishna Rao BR, 2011. Taxonomy construction techniques—issues and challenges. *Ind J Comput Sci Eng*, 2(5):661-671.

Thada V, Jaglan DV, 2013. Comparison of jaccard, dice, cosine similarity coefficient to find best fitness value for Web retrieved documents using genetic algorithm. *Int J Innov Eng Technol*, 2(4):202-205.

Treeratpituk P, Callan J, 2006. Automatically labeling hierarchical clusters. Proc Int Conf on Digital Government Research, p.167-176. https://doi.org/10.1145/1146598.1146650

Turner V, Gantz J, Reinsel D, 2014. The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things. IDC White Paper, p.1-5. https://doi.org/10.7790/ajtde.v2n3.47

Velardi P, Faralli S, Navigli R, 2013. OntoLearn reloaded: a graph-based algorithm for taxonomy induction. *Comput Ling*, 39(3):665-707. https://doi.org/10.1162/COLI_a_00146

Weng SS, Liu CK, 2004. Using text classification and multiple concepts to answer e-mails. *Expert Syst Appl*, 26(4): 529-543. https://doi.org/10.1016/j.eswa.2003.10.011

Yang HC, Lee CH, Hsiao HW, 2015. Incorporating self-organizing map with text mining techniques for text hierarchy generation. *Appl Soft Comput*, 34:251-259. https://doi.org/10.1016/j.asoc.2015.05.005

Yao JJ, Cui B, Cong G, et al., 2012. Evolutionary taxonomy construction from dynamic tag space. *World Wide Web*, 15(5-6):581-602. https://doi.org/10.1007/s11280-011-0150-4