An efficient generic approach for automatic taxonomy generation using HMMs

Sylvain Iloga, Olivier Romain & Maurice Tchuenté

Pattern Analysis and Applications

ISSN 1433-7541

Pattern Anal Applic
DOI 10.1007/s10044-020-00918-0

Springer

Pattern Analysis and Applications
https://doi.org/10.1007/s10044-020-00918-0

THEORETICAL ADVANCES

An efficient generic approach for automatic taxonomy generation using HMMs

Sylvain Iloga[1,2,4] · **Olivier Romain**[2] · **Maurice Tchuenté**[3,4]

Received: 5 March 2020 / Accepted: 9 September 2020

Abstract
Taxonomies are essential tools for fast information retrieval and classification of knowledge. Many existing techniques for automatic taxonomy generation strongly depend on the specific properties of a particular domain and are consequently hard to apply to other domains. Some attempts have been made to design taxonomies for multiple domains. Unfortunately, they induce high hierarchical classification error rates for some datasets. The automatic design of a taxonomy requires the capability of measuring the similarity between classes. More precisely, the fact that two classes are near intuitively implies that some elements of one class are scattered in the neighborhood of some elements of the other class. This observation is used in this paper to propose a new generic technique for automatic taxonomy generation. A topological analysis of the neighborhood of each instance is first performed. The results of this analysis are used to initialize and train a hidden Markov model for each class. The model of a given class c captures the frequencies of the classes found in the neighborhood of the instances of c, from the most dominant class to the least dominant. The similarities between these models are finally used to derive a taxonomy. Hierarchical classification experiments realized on 20 datasets from various domains showed an average accuracy of 97.22% and a standard deviation of 4.11%. Comparison results revealed that the proposed approach outperforms existing work with accuracy gains reaching 38.62% for one dataset.

Keywords Automatic taxonomy generation · Hidden Markov models · Hierarchical classification

1 Introduction

A taxonomy is an essential tool for fast information retrieval and classification of knowledge [1]. A taxonomy provides an efficient navigating and browsing mechanism by organizing huge volumes of data into a relatively small number of hierarchical clusters [2]. In this hierarchy, broad concepts are at the top and more specific concepts are further down [3]. Initially introduced for the classification of biological species, the concept of taxonomy is nowadays used in several other areas related to data (text, audio, image and video) mining and processing. One of the most important tasks in taxonomy generation is the evaluation of the quality of the resulting taxonomies. This can be achieved through hierarchical classification [4–10] or by using appropriate metrics [2, 3, 11–14]. It is also possible to entrust this evaluation to human experts [3] or to perform ontological tests using a specialized software program [15].

Several techniques for taxonomy generation have been designed, and detailed surveys on various approaches for taxonomy construction are available in [16, 17]. All these techniques can be organized into three main categories:

1. The *manual* approach [18–21].
2. The *semi-automated* approach [8, 22–24].
3. The *automated* approach [2–7, 9–15].

✉ Sylvain Iloga
sylvain.iloga@gmail.com

Olivier Romain
olivier.romain@gmail.com

Maurice Tchuenté
maurice.tchuente@gmail.com

[1] Department of Computer Science, Higher Teachers' Training College, University of Maroua, P.O.box 55, Maroua, Cameroon

[2] ENSEA, CNRS, ETIS UMR 8051, CY Cergy Paris University, 95000 Cergy, France

[3] Department of Computer Science, Faculty of Science, University of Yaoundé I, P.O.box 812, Yaoundé, Cameroon

[4] IRD, UMMISCO, University of Sorbonne, 93143 Bondy, France

Published online: 18 September 2020

Manually generated taxonomies are generally constructed using a top-down approach. This implies the selection according to the context, of a few number of higher categories and to further move down to more specific levels of lower subcategories. A bottom-up approach is generally applied for automatic taxonomy construction. This involves the selection of specific levels of categories to further move up to higher categories [1]. Manual taxonomy construction is a tedious process, and the resulting taxonomy is generally highly subjective [13]. Manually generated taxonomies generally use human semantics and are mainly suitable for human use, whereas automatically generated taxonomies are optimized for computational classifiers [19]. Furthermore, automatic approaches have the potential to enable humans or machines to easily understand a highly focused and potentially fast changing domain [13]. Details related to the advantages and limitations of these two approaches are presented in [1]. Semi-automated approaches generally share the advantages and drawbacks of manual and automatic approaches. In [5] and [25], several criteria that should guide the design of a '*good*' taxonomy are analyzed.

Some existing automatic techniques are applicable to multiple domains [4–8], but most of these techniques exhibit high hierarchical classification error rates. In some cases, the hierarchical classifiers derived from these techniques are even outperformed by the corresponding flat classifiers [4, 7, 8]. Other existing techniques strongly depend on the specific properties of a particular domain, and their application in other domains is not straightforward [2, 3, 9–15].

The automatic design of a taxonomy requires the capability of measuring the similarity between classes. When the elements of each class are observed as points scattered in the multidimensional affine space, the fact that two classes c_1 and c_2 are near intuitively induces that some elements of c_1 (resp. c_2) are found in the neighborhood of some elements of c_2 (resp. c_1). The probability to find elements of one class c_2 in the neighborhood $\aleph$ of another class c_1 depends on the zone covered by $\aleph$. Therefore, when measuring the similarity between two classes, the radius of the neighborhood that must be considered for each instance is a crucial parameter. Indeed, if $\aleph$ is very large (resp. very narrow), then the probability is almost always equal to 1 (resp. 0). Unfortunately, none of the existing techniques for automatic taxonomy generation exploits this fundamental observation.

To overcome the aforementioned limitations, an efficient generic approach for automatic taxonomy generation based on hidden Markov models (HMMs) is proposed in this paper. In the proposed approach, one HMM is first designed for each class c. This model captures the global frequencies of the classes found in the neighborhood of the instances of c, from the most dominant class to the least dominant one. The similarities between these models are finally used to derive a taxonomy. The performances of the proposed approach are then evaluated in a hierarchical classification process on 20 publicly available datasets from various domains.

The rest of this paper is organized as follows: The state of the art is presented in Sect. 2, followed by a summarized presentation of HMMs in Sect. 3. A detailed description of the approach proposed in this paper is given in Sect. 4. Experimental results are presented in Sect. 5, and the last section is devoted to the conclusion.

2 State of the art

2.1 Related work

An automatic taxonomy generation technique is generally based on algorithms designed to capture the hierarchical relationships existing between the categories found in a database. The performance of such a technique relies on:

1. The experts' opinion regarding the taxonomies.
2. The quality of the taxonomies.
3. The time complexity of the technique.

The problem of automatic taxonomy generation for speech archives was investigated in [11]. In that work, similar key terms manually extracted from the transcription of a speech archive are first grouped into clusters. Then, similar clusters are grouped into super clusters to form a taxonomy of the archive. The *Agglomerative Hierarchical Clustering* (*AHC*) [26] and the *Hierarchical Cluster Partitioning* (*HCP*) algorithms are used to achieve this goal. The principle of *HCP* is fully described in [11]. *AHC* produces a binary tree of clusters, and *HCP* generates a multi-way tree from the output of *AHC*. The *F-measure* was used as metric to evaluate the quality of the resulting taxonomies.

An automatic approach to extract information from the Web in order to build a taxonomy of terms and Web resources for a given domain was proposed in [15]. The most representative Web sites for each concept in the domain are first retrieved and categorized. Then, a *Sequential Agglomerative Hierarchical Non-Overlapping* (*SAHN*) clustering algorithm fully described in that paper is used to join the more similar terms, using the number of coincidences between their URLs sets as a similitude measure. Ontological tests have been performed in the software *PROTÉGÉ* to evaluate the correctness of the resulting taxonomies from a logical point of view. Experts estimate that the taxonomy obtained with this approach really represents the state of the art on the Web for a given concept.

The problem of generating *Attribute Value Taxonomies* (AVTs) was tackled in [4] and [6]. In [4], *AVT-Learner*, an algorithm for automated construction of AVTs from data, was introduced. This algorithm uses the *AHC* algorithm

to cluster attribute values based on the distribution of the classes that co-occur with the values. Hierarchical classification experiments on 37 datasets of the *UCI Machine Learning Repository*[1] showed that the AVTs generated by *AVT-Learner* were competitive with available human-generated AVTs. In [6], an adaptive genetic algorithm named *MCM-AVT-Learner* is proposed to generate AVTs. This algorithm imports the mutations and crossover matrices which makes effective use of the fitness ranking and loci statistics information. Experimented through hierarchical classification on eight datasets of the *UCI Machine Learning Repository* from diverse domains, *MCM-AVT-Learner* generated AVTs that were competitive with human-generated AVTs and also with AVTs generated by other algorithms including *AVT-Learner* [4].

The problem of learning *N*-ary taxonomies with no user-defined parameters is analyzed in [5]. As a solution, a framework that categorizes a '*good*' taxonomy is proposed and an algorithm named *constructTaxonomy* is developed to automatically find a good taxonomy. This algorithm is considerably less greedy than existing ones. The resulting taxonomies have been evaluated through hierarchical classification on several real-life datasets from diverse domains like text mining, hyper-spectral analysis and written character recognition. Linear support vector machines (SVMs) with Gaussian kernels and Bayesian classifiers were used in a fivefold cross-validation. The resulting taxonomies seemed to be more '*natural*' and produced better classification accuracies than binary taxonomies.

An automatic taxonomy generation method on relational datasets is proposed in [2]. That work is based on the relational clustering algorithm *DIVA* [27] which consists of two steps:

1. A *divisive step* where the dataset is divided into a user-defined number of clusters so that the variance of each cluster is lower than a fixed threshold.
2. Based on these clusters, a hierarchical dendrogram is generated using an *agglomerative algorithm*.

The *intra-node entropy* and the *inter-node entropy* were used to measure the quality of the generated taxonomies. Comparisons were made with the taxonomies generated by the *AHC* and the *bisecting K-means partitioning* algorithms on the synthetic Amazon dataset and a real movie dataset. The results showed that *DIVA* was slightly beaten by these two algorithms on the synthetic Amazon dataset with a quality loss reaching 4.2%. On the other hand, *DIVA* obtained a quality of 0.033 on the real movie dataset. This is considerably better than the two other approaches that produced solutions with qualities no higher than 0.026.

In [7], an algorithm named Propositionalized Attribute Learner (*PAT-Learner*) that automatically constructs taxonomies is introduced. This algorithm propositionalizes attributes and hierarchically clusters the propositionalized attributes based on the distribution of class labels that co-occur with them, to generate a taxonomy. *PAT-DTL*, an inductive algorithm that exploits the taxonomy generated by *PAT-Learner* as prior knowledge to generate compact decision trees, is also introduced in [7]. Hierarchical classification experiments realized on 37 datasets from the *UCI Machine Learning Repository* showed that *PAT-DTL* generates decision trees that are more compact than standard decision trees.

Some authors focus on the automatic generation of taxonomies from a text corpus [3, 12–14]. For instance in [12], it is proposed to use a framework which incrementally clusters terms from text documents based on ontology metric, in order to derive taxonomies. This approach combines the strengths of both lexico-syntactic patterns and clustering, through the use of heterogeneous features. More formally, a feature vector $h(c_x, c_y) \in \mathbb{R}^n$ is initially calculated for every input terms c_x and c_y. These features include contextual information, co-occurrence, syntactic dependency, lexico-syntactic patterns and miscellaneous. The resulting feature vectors are later used to incrementally insert the terms into the taxonomy. Experimental results reveal that this approach achieves higher F_1-*measure* than other related approaches, when the hypernym and the meronym taxonomies extracted from WorldNet and ODP are considered as gold standards.

The *Bayesian Rose Tree* (BRT) algorithm has been used to build a multi-branch taxonomy from a given set of keywords phrases [13]. This algorithm reduces the complexity of previous hierarchical clustering algorithms from $O(n^2 \log n)$ to $O(n \log n)$. The *Normalized Mutual Information* is then used as metric to access the quality of the generated taxonomies.

A graph-based solution for automatic taxonomy generation from a large text corpus is also proposed in [3]. This algorithm, *GraBTax*, uses statistical co-occurrences and lexical similarity to optimize the structure of the taxonomy. After extracting topical terms and their relationships from the text corpus, *GraBTax* constructs a weighted graph representing topics and their associations. Finally, the multilevel graph partitioning algorithm proposed in [28, 29] is used to generate the taxonomy. This algorithm finds the partitioning that minimizes the edge-cut while keeping the vertex strength balanced. The quality of the resulting taxonomies was manually evaluated by human experts. This evaluation was combined with the automatic calculation of the *semantic precision*. Experimented on 1.1 Million papers with titles and abstracts from the *CiteSeer*^{X} database gave

[1] https://archive.ics.uci.edu/ml/.

a semantic precision of 0.69. This is significantly higher than the semantic precision of 0.44 exhibited by the *AHC* algorithm.

A recent deep reinforcement learning approach for automatic taxonomy induction from a set of terms has been proposed in [14]. The aim of that approach is to unify hypernymy detection (i.e., extraction of term pairs from '*is-a*' relation) and hypernymy organization (i.e., organizing '*is-a*' term pairs into a tree structure hierarchy), in an end-to-end manner. To achieve this goal, the authors first design a policy neural network to incorporate semantic information of term pairs. They then use cumulative rewards to holistically measure the quality of the resulting taxonomies. Experimented on two public datasets [30, 31], this approach performed 19.6% better than existing techniques on the *ancestor* F_1 which is a metric that compares ancestors ('*is-a*' pairs) from the generated taxonomy with those from a gold taxonomy.

The authors of [8] use equal-depth discretization approach [32] and entropy-based Kononenko discretization [33] approach to generate taxonomies over data described by continuous attributes. For data described by nominal attributes, taxonomies are analyst-provided. Hierarchical classification experiments on 20 datasets from the *UCI Machine Learning Repository* show the effectiveness of this approach which improves the accuracy of state-of-art classifiers.

Other authors rather tackle the problem of automatic generation of music genre taxonomies [9, 10]. An overview of relevant existing techniques is available in [9] where a sequential pattern mining approach to automatically generate music genre taxonomies is proposed as follows. Consider a music database composed of M genres g_i ($1 \leq i \leq M$). Each song of g_i is initially transformed into a sequential pattern using the *k-means* [34] and the *K-NN* [35] algorithms. Thereafter, the characteristic sequential patterns of genre g_i are extracted using the *PrefixSpan* [36] and the *FeatureMine* [37] algorithms. These characteristic sequential patterns are then used to compute a descriptor vector $\overrightarrow{V_i}$ for genre g_i. The jth component of $\overrightarrow{V_i}$ is the percentage of characteristic sequential patterns of genre g_j ($1 \leq j \leq M$) that are contained in those of genre g_i. Finally, the *AHC* algorithm is applied on these vectors to derive the taxonomy. Hierarchical classification experiments applied to the music database *GTZAN* which belongs to the *MARSYAS repository* [2] exhibit the actual best accuracy.

An accurate HMM-based similarity measure between two sets of histograms is proposed in [10]. A model is first trained with the *Baum–Welch* algorithm [38] for each set of histograms. This HMM captures the bin values and the visual shapes of the histograms in the considered set, irrespective of their bin sizes. The similarity between the two input sets of histograms is then obtained by evaluating the similarity between their respective models. This similarity measure is then experimented in the automatic generation of music genres taxonomies as follows. Consider a music database composed of M genres g_i ($1 \leq i \leq M$). The 60-bin *Rhythm Histogram* (RH) [39] corresponding to each song of a genre g_i is first calculated, and a model λ_i is trained to capture the visual shape of these RHs. Then, a descriptor vector $\overrightarrow{W_i}$ is associated with g_i in such a way that the jth component of $\overrightarrow{W_i}$ is the similarity between the models λ_i and λ_j ($1 \leq j \leq M$). Similar to what was done in [9], the *AHC* algorithm is also applied on these vectors to derive the taxonomy. Experimental results show the actual best hierarchical classification accuracy for the music database *GTZAN+* [3] which is an extension of *GTZAN*. Table 1 summarizes the main information about the papers surveyed in this section.

2.2 Problem statement

As previously stated in this paper, the spatial proximity between two classes means that some elements of one class are scattered in the neighborhood of certain elements of the other class. None of the existing techniques for automatic taxonomy generation relies on this fundamental observation.

As it can be observed in Table 1 and in the surveys presented in [16, 17], a high number of existing techniques for automatic taxonomy generation strongly depend on the specific properties of a particular domain and are consequently hard to apply to other domains. For example, the techniques proposed in [9, 10] for music genre recognition strongly rely on the natural sequentiality of music excerpts. This property is not valid for the text documents manipulated in [3, 12–14]. Although a technique may be applicable to multiple domains [4–8], at present, no technique performs well in multiple domains. Indeed, most of these techniques exhibit high hierarchical classification error rates for some datasets. Table 2 presents the highest error rates of these existing works for the datasets involved in the experiments of the current work.

In some cases, existing work is even outperformed by the corresponding flat classifiers. This is the case, for example,

1. In [4] for the datasets *Iris*, *Primary-tumor*, *Vowel* and *Waveform-5000*.
2. In [7] for the datasets *Anneal*, *Audiology*, *Autos*, *Car*, *Glass*, *Hypothyroid*, *Letter*, *Lymph*, *Nursery*, *Segment*, *Vehicule* and *Vowel*.
3. In [8] for the dataset *Satimage*.

Table 3 summarizes the aforementioned specificities of the existing techniques. The content of this Table 3 leads us

[2] http://marsyas.info/downloads/datasets.html.

[3] https://perso-etis.ensea.fr/sylvain.iloga/GTZAN+/.

Table 1 Specificities of relevant related papers in automatic taxonomy generation

Authors [references]	Year	Domain	Algorithms	Interest of the approach	Evaluation of the taxonomy
Chien et al. [11]	2002	Speech processing	Manual extraction of similar key terms , *AHC*, *HCP*	Reduces the complexity by avoiding an automatic extraction of key terms	F-measure
Sánchez and Moreno [15]	2004	Web search	*SAHN*	The resulting taxonomies seemed to really represent the Web	Ontological tests in *PROTÉGÉ*
Li and Anand [2]	2008	Relational datasets	*DIVA*	More efficient than existing works	Intra-node and inter-node entropy
Hui Y. and Jamie C. [12]	2009		Incremental term clustering based on ontology metric	Achieves higher F_1 *-measure than existing techniques*	F_1-measure
Liu et al. [13]	2012	Text processing	*Bayesian Rose Tree* taking keywords as input	Reduces the complexity from $O(n^2 \log n)$ to $O(n \log n)$	Normalized Mutual Information
Treeratpituk et al. [3]	2013		*GraBTax*	Better at discovering meaningful relationship compared to *AHC*	Human experts and semantic precision
Mao et al. [14]	2018		Deep reinforcement learning	Performed 19.6% better than existing techniques on the *ancestor* F_1	Ancestor F_1
Iloga et al. [9]	2018	Music genre recognition	*PrefixSpan*, *k-means, K-NN, FeatureMine, AHC*	Actual best hierarchical classification accuracy in *GTZAN*	Hierarchical classification
Iloga et al. [10]	2018		*Baum–Welch*, HMMs similarity computation, *AHC*	Actual best hierarchical classification accuracy in *GTZAN+*	
Kang et al. [4] Kang et al. [4]	2004		*AVT-Learner*	Generates AVTs that were competitive with human-generated AVTs	
Punera et al. [5]	2006	Multiple domains	*constructTaxonomy* that produces *N*-ary taxonomies, $N \geq 2$	Exhibits better classification accuracies than binary taxonomies	
Jo et al. [6]	2008		*MCM-AVT-Learner*	Generates AVTs that were competitive with human-generated AVTs and with other algorithms including *AVT-Learner*	
Kang and Sohn [7]	2009		*PAT-Learner* and *PAT-DTL*	Generates more compact decision trees than standard decision trees	
Cagliero and Garza [8]	2013		*Equal-depth* and *entropy-based* discretization	Improved the accuracy of state-of-art classifiers	

Table 2 Highest error rates in hierarchical classification of relevant existing work for the datasets involved in the experiments of the current work

Authors [references]	Dataset	Error rate (%)
Jo et al. [6]	Anneal	26.33
Punera et al. [5]	Glass	27
Kang et al. [4]	Primary-tumor	52.22
Kang and Sohn [7]	Primary-tumor	55.46

to the following question: *Is it possible to develop an efficient generic approach for automatic taxonomy generation based on the topological analysis of the neighborhood of each instance?* This paper attempts to provide a positive answer to this question.

Given a database composed of many classes, we propose an approach that uses HMMs to model each class. HMMs generally deal with temporality in the data representation of the considered domain. Therefore, our biggest challenge

Table 3 Specificities of the techniques reviewed in this paper. 'S_1' to 'S_4,' respectively, refer to '*No topological analysis of the neighborhood*,' '*Strongly depend on the specific properties of a particular domain*,' '*Outperformed by flat classifiers*' and '*High error rates in hierarchical classification for some datasets*'

Authors [references]	Year	S_1	S_2	S_3	S_4
Chien et al. [11]	2002	✓	✓		
Kang et al. [4]	2004	✓		✓	✓
Sánchez & Moreno [15]	2004	✓	✓		
Punera et al. [5]	2006	✓			✓
Jo et al. [6]	2008	✓			✓
Li & Anand [2]	2008	✓	✓		
Hui Y. & Jamie C. [12]	2009	✓	✓		
Kang & Sohn [7]	2009	✓		✓	✓
Liu et al. [13]	2012	✓	✓		
Cagliero & Garza [8]	2013	✓		✓	
Treeratpituk et al. [3]	2013	✓	✓		
Iloga et al. [9]	2016	✓	✓		✓
Iloga et al. [10]	2018	✓	✓		✓
Mao et al. [14]	2018	✓	✓		

is to provide a uniform temporal representation of the data, irrespective of the considered domain. Once this is done, the robust existing algorithms associated with HMMs can be used to learn the data in order to derive the taxonomy.

3 Presentation of HMMs

HMMs are used in several domains such as the comparison of sets of histograms [10], handwriting recognition [40], texture retrieval and classification [41]. They are also thoroughly used in speech recognition. A detailed tutorial on HMMs and selected applications in speech recognition is available in [38].

3.1 Definition of HMM

An HMM $\lambda = \{A, B, \pi\}$ is composed of [38]:

1. The number N of states of the model. The set of states is $S = \{S_1, S_2, \ldots, S_N\}$, and the state of the model at time $t \geq 1$ is denoted $q_t \in S$.
2. The number M of symbols. The set of symbols is $V = \{v_1, v_2, \ldots, v_M\}$, and the symbol observed at time t is denoted $O_t \in V$.
3. The state transition probability distribution $A = \{a_{ij}\}$, where a_{ij} is the probability $P(q_{t+1} = S_j | q_t = S_i), 1 \leq i, j \leq N$.
4. The symbols probability distribution $B = \{b_i(k)\}$, where $b_i(k)$ is the conditional probability $P(v_k \text{ at time } t | q_t = S_i), 1 \leq i \leq N$ and $1 \leq k \leq M$.

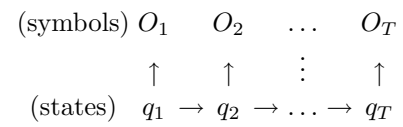


Fig. 1 A Markov chain

5. The initial states probability distribution $\pi = \{\pi_i\}$, where $\pi_i = P(q_1 = S_i), 1 \leq i \leq N$.

3.2 HMM as a sequence generator

An HMM $\lambda = \{A, B, \pi\}$ can be used to generate a sequence $O = O_1 O_2 \ldots O_T$ composed of T symbols observed by the sequence of states $Q = q_1 q_2 \ldots q_T$ as described in Fig. 1. In the rest of this paper, a diagram similar to that of Fig. 1 is called a *Markov chain*.

The Markov chain presented in Fig. 1 is obtained by executing the following algorithm:

1. Select the initial state $S_j \in S$ with respect to the distribution π, and set $t = 0$.
2. Set $t = t + 1$ and then change the current state to $q_t = S_j$
3. Select the symbol $O_t \in V$ to be observed at state q_t according to the distributions in B.
4. If $t < T$, **go to** step 5, or else **terminate**.
5. Perform the state transition from the current state q_t to the next state $S_j \in S$ according to the distribution A, and then, **go to** step 2.

3.3 HMM probability computing and training

Consider a sequence $O = O_1 O_2 \ldots O_T$ and an HMM $\lambda = \{A, B, \pi\}$. The probability $P(O|\lambda)$ to observe O given λ is efficiently calculated by the *Forward–Backward* algorithm which has a time complexity of $\theta(T.N^2)$ [38]. The parameters A, B and π of the HMM λ can be re-estimated in order to maximize the value of $P(O|\bar{\lambda})$, where $\bar{\lambda} = \{\bar{A}, \bar{B}, \bar{\pi}\}$ is the re-estimated model. This re-estimation is done by the *Baum–Welch* algorithm [38]. This algorithm improves the probability to observe O from the model, by iteratively using $\bar{\lambda}$ in place of λ and repeating this re-estimation until $\bar{\lambda} = \lambda$, or until a user-defined maximum number of iterations γ is reached. For one iteration, the time complexity of the *Baum–Welch* algorithm is dominated by computation of $P(O|\bar{\lambda})$ using the *Forward–Backward* algorithm. Therefore, the global algorithm runs in $\theta(\gamma.T.N^2)$. It is also possible to train λ to maximize the value of the probability $P(O|\bar{\lambda}) = \sum_{k=1}^{K} P(O^{(k)}|\bar{\lambda})$ where $O = \{O^{(1)}, \ldots, O^{(K)}\}$ is a set of K sequences and $O^{(k)} = O_1^{(k)} \ldots O_{T_k}^{(k)}$ is the kth sequence of O. In the case of multiple sequences, only the distributions

A and B are re-estimated by the *Baum–Welch* algorithm because $\bar{\pi}$ can be statistically determined from the initial states of the K observed sequences [38]. The time complexity of the *Baum–Welch* algorithm for multiple sequences can be approximated by $\theta\left(\gamma.(\sum_{k=1}^{K} T_k).N^2\right)$. In the particular case where all the K sequences have the same length T, this time cost is given by $\theta(\gamma.K.T.N^2)$

The *Baum–Welch* algorithm tends to converge to a local optimum that does not correspond to the global best parameters of the model. Indeed, the quality of the solution produced depends on how the parameters of the initial model fit the statistical content of the training sequences.

3.4 HMMs similarity measure

Numerous distance and similarity measures between two HMMs have been proposed [40–47]. For example, in order to compare two HMMs $\lambda = \{A, B, \pi\}$ and $\lambda' = \{A', B', \pi'\}$, one can compute the *Euclidean* distance between the rows of B and B' or calculate their *Kullback–Leibler* divergence [42]. Techniques for fast computation of an upper-bound of the *Kullback–Leibler* divergence were proposed in [45] and [46]. It is also possible to evaluate their *Monte Carlo* approximation [44]. However, all these measures are limited. For example, the temporal structure of the models is not captured in the *Euclidean* distance, and the *Kullback–Leibler* divergence of two HMMs is not symmetric (symmetric alternatives have later been proposed). The *Monte Carlo* approximation has a high computational cost, and the produced value may change from one computation to another. Other measures have been proposed to attenuate these limitations: the *co-emission probability* [43], the *stationary cumulative distribution* [47], the *Bayes probability error* [40] and the *generalized probability product Kernel* [41], among others.

In the current work, we opted for the accurate low-complexity similarity measure between two HMMs λ and λ' proposed in [44]. This similarity measure hereafter denoted $d_{sim}(\lambda, \lambda')$ is based on the probability that λ and λ' generate identical sequences following the principle of Fig. 1. Its theoretical time complexity is $\theta(N^3 + N^2.M^2)$, when both λ and λ' have N states and M symbols. A detailed computation scheme of d_{sim} can be found in [10].

4 The proposed approach

4.1 Main idea

Let $\Omega = \cup c_i, (1 \leq i \leq M)$ be a database composed of M classes to be organized into a taxonomy, each class c_i being represented by $|c_i|$ instances. The main idea of this work is based on the following hypothesis: '*the similarity between two classes can be derived from the topological analysis of a fixed user-defined neighborhood of their instances*.' More formally, if the results of such an analysis for the instances of two classes c_1 and c_2 can be, respectively, captured into two models λ_1 and λ_2, then the similarity between c_1 and c_2 can be obtained by calculating the similarity between λ_1 and λ_2.

In this paper, the topological analysis of the neighborhood of an instance X consists in detecting the classes appearing in this neighborhood, from the most dominant class to the least dominant class. In order to analyze the neighborhood of an instance, we assume that there exists a distance/dissimilarity measure between any two elements of Ω. Let us define $presence(c_i, X)$ as the percentage of instances of class c_i found in the neighborhood of X. In other words, $presence(c_i, X) = y\%$ means that $y.\frac{|c_i|}{100}$ instances of class c_i are located in the neighborhood of X. Let us, for example, set $M = 3$ and consider a particular instance X whose neighborhood has the following composition: $presence(c_1, X) = 25\%$, $presence(c_2, X) = 5\%$ and $presence(c_3, X) = 45\%$. When these values are sorted in decreasing order to arrange the classes according to their dominance in the neighborhood of X, this neighborhood can be transformed into the Markov chain shown in Fig. 2. This Markov chain contains classes (symbols) generated by the corresponding sequence of percentages (hidden states).

In Fig. 2, an up-arrow from $y\%$ to c_i indicates that $presence(c_i, X) = y\%$. When the left arrows are browsed from left to right in that figure, the kth left arrow materializes the transition from the kth dominant class to the $(k + 1)$th dominant class in the neighborhood of X.

The main idea of the current work relies on the combination of the frequencies and the dominance of the classes found in the neighborhood of each instance. The dominance does not basically matter when each instance is considered individually, but combining the frequencies and the dominance can enable to obtain finer results during the comparison of two instances. Indeed, two instances that are different (resp. identical) regarding the frequencies of the classes appearing in their respective neighborhoods may be similar (resp. different) regarding the dominance of these classes. A typical situation that illustrates the importance of combining the frequencies and the dominance is provided by the instances that are equidistant to a given instance. As an example, the instances B, C, D and E in Table 4 are equidistant to instance A considering the frequencies of the four classes appearing in their respective neighborhoods when the *Euclidean* or the *Manhattan* distance is applied. However, when the dominance is considered, B, C, D and E are not equidistant to A and one can obviously observe that the nearest neighbor of A is E.

The example of Table 4 illustrates the interest of combining the frequencies and the dominance during the comparison of two instances. Since the goal of the current work is to

Table 4 Importance of combining the frequencies and the dominance during the comparison of two instances

	Class 1	Class 2	Class 3	Class 4	Dominance
A	30%	20%	15%	35%	Class 4 → Class 1 → Class 2 → Class 3
B	35%	15%	20%	30%	Class 1 → Class 4 → Class 3 → Class 2
C	25%	15%	20%	40%	Class 4 → Class 1 → Class 3 → Class 2
D	35%	25%	10%	30%	Class 1 → Class 4 → Class 2 → Class 3
E	35%	15%	10%	40%	Class 4 → Class 1 → Class 2 → Class 3

$$\begin{array}{lccccc} \text{Classes } (symbols) & c_3 & & c_1 & & c_2 \\ & \uparrow & & \uparrow & & \uparrow \\ \text{Percentages } (states) & 45\% & \rightarrow & 25\% & \rightarrow & 5\% \end{array}$$

Fig. 2 Initial Markov chain associated with X

$$\begin{array}{ccccc} c_3 & & c_1 & & c_2 \\ \uparrow & & \uparrow & & \uparrow \\ 45\% & \rightarrow & 45\% & \rightarrow & 5\% \end{array} \qquad \begin{array}{ccccc} c_1 & & c_3 & & c_2 \\ \uparrow & & \uparrow & & \uparrow \\ 45\% & \rightarrow & 45\% & \rightarrow & 5\% \end{array}$$

(a) 1^{st} Markov chain (b) 2^{nd} Markov chain

Fig. 3 The two Markov chains associated with the instance X when classes c_1 and c_3 appear in identical proportions in its neighborhood

compare the available classes (and implicitly their instances) to derive a taxonomy, we thus needed to select a formal representation (and consequently a formal model) that can capture both: the frequencies and the dominance in order to obtain finer results. Markov chains (and consequently HMMs) appear to be an accurate choice as described in Fig. 2, although each class appears only once in this representation. Besides HMMs, probability distributions like multinomials may also be considered. However, probability distributions will only consider the frequencies while neglecting the dominance, unlike HMMs that can capture both.

When several classes appear in identical proportions in the neighborhood of an instance X, X is transformed into z Markov chains, where z is the number of possible results of the decreasing sorting of its neighborhood. For $M = 3$, if we have, for example, $presence(c_1, X) = 45\%$, $presence(c_2, X) = 5\%$ and $presence(c_3, X) = 45\%$, then X is transformed into the Markov chains presented in Fig. 3a and b.

When the proposed transformation is applied to all the instances of class c_i, a database δ_i of Markov chains is obtained. The content of δ_i is later used to initialize and train a model λ_i associated with class c_i. These models are finally used to derive the taxonomy. The main obstacle in this approach is that the states are represented by percentages that can take any value in the continuous interval [0, 100]. However, the number of states of an HMM is always finite. By analogy with what was done in [10] to overcome the same issue, the interval [0, 100] is split here into N slices as described in Equation 1 (Fig. 4):

$$S_1 = [0, \frac{100}{N}] \text{and} S_j =]\frac{100}{N} \times (j-1), \frac{100}{N} \times j], 2 \leq j \leq N \qquad (1)$$

In this equation, N is a user-defined number and each slice is now considered as a state after replacing each percentage appearing in the Markov chain by the identifier of the slice to which it belongs. For example, if $N = 5$, then the slices are: $S_1 = [0, 20]$, $S_2 =]20, 40]$, $S_3 =]40, 60]$, $S_4 =]60, 80]$ and $S_5 =]80, 100]$. Using these five slices, Fig. 2 is transformed into Fig. 5.

4.2 Methodology

The technique proposed in this paper requires an initial implicit step where each instance in the database must be

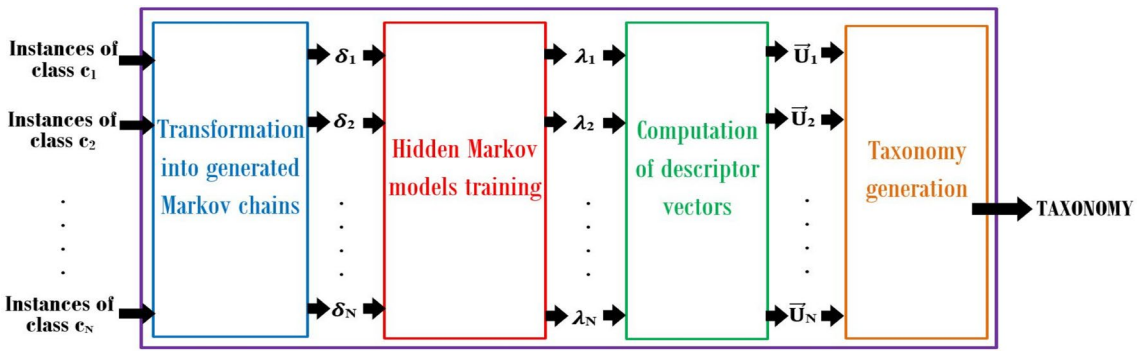


Fig. 4 Methodology of the proposed approach

$$\begin{array}{lccccc} \text{Classes } (symbols) & c_3 & & c_1 & & c_2 \\ & \uparrow & & \uparrow & & \uparrow \\ \text{Intervals } (states) & S_3 & \rightarrow & S_2 & \rightarrow & S_1 \end{array}$$

Fig. 5 The final Markov chain associated with the instance X when each percentage is replaced by the identifier of the slice to which it belongs

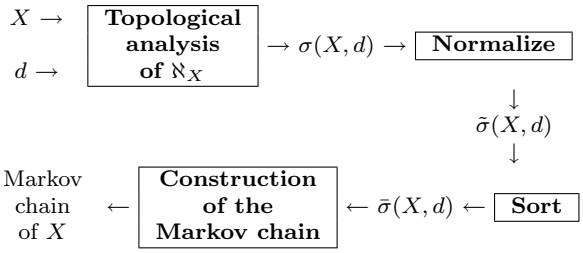


Fig. 6 Transformation of the instance X into a Markov chain considering its neighborhood $\aleph_X$ of radius d

stored as a vector $X \in \mathbb{R}^n$. This initial step is generally straightforward. This happens, for instance, in the classification of vertebrates as mammals, birds, fishes, reptiles or amphibians. The database then contains pairs (x, y), where y is the category and x represents the attributes of a vertebrate including its body temperature, its skin cover, its method of reproduction, its ability to fly and its ability to live in water [48]. However, there are applications where this initial step needs a sophisticated analysis. For instance, in music genre recognition, the representation of a song c requires three main steps [9, 10]: (1) Define an elementary time interval t (for instance one second) and extract appropriate timbre or rhythm music features from excerpts within windows of duration t. (2) Represent each c as a sequence $v_1 v_2 ... v_w$ of vectors of music features. (3) The last thing to do is to use the former sequences to derive a descriptor vector $X \in \mathbb{R}^n$ for c.

Let us assume now that each instance in the database is stored as a vector $X \in \mathbb{R}^n$. Figure 4 describes the following global process proposed in this paper to automatically generate a taxonomy:

1. Each instance $X \in c_i$ is transformed into a Markov chain according to the content of its neighborhood as described in Sect. 4.3. This produces a database δ_i of Markov chains for each class c_i.
2. A model λ_i associated with class c_i is then initialized and trained according to the content of δ_i as described in Sect. 4.4.
3. Class c_i is later associated with a descriptor vector $\overrightarrow{U_i}$ whose components are the similarities between λ_i and the various models $\lambda_j, 1 \leq j \leq M$. Section 4.5 presents the computation scheme of $\overrightarrow{U_i}$.
4. Finally, the descriptor vectors associated with all the M classes are used to generate the taxonomy as described in Sect. 4.6.

4.3 Transformation of an instance into a Markov chain

As it can be observed in Fig. 6, in order to transform an instance X into a Markov chain considering its neighborhood $\aleph_X$ of radius d, the composition of $\aleph_X$ is first analyzed topologically. The result of this analysis is a vector $\sigma(X, d)$ whose components are the frequencies of the classes found in $\aleph_X$. The components of $\sigma(X, d)$ are then normalized to obtain the vector $\tilde{\sigma}(X, d)$. Thereafter, the components of $\tilde{\sigma}(X, d)$ are sorted in decreasing order. The resulting sorted vector $\bar{\sigma}(X, d)$ is finally used to derive the Markov chain of X. The formal definitions of $\sigma(X, d)$, $\tilde{\sigma}(X, d)$ and $\bar{\sigma}(X, d)$ are given in the following subsections.

4.3.1 Definition of new concepts

Definition 1 Consider a user-defined positive real number d, and let *dist* be the distance/dissimilarity between two instances of Ω. For each $X \in \Omega$, we define in Equation 2 the set $\omega_j(X, d)$ containing the instances of class c_j found in the neighborhood $\aleph_X$ of X materialized by a ball of radius d centered at X:

$$\omega_j(X, d) = \{Y \in c_j | Y \neq X \text{ and } dist(X, Y) \leq d\} \tag{2}$$

A large value of d may lead to a situation where $\omega_j(X, d) \approx \Omega$. To avoid this situation, we suggest to select the value of d in such way that $d \leq \beta . d_{max}$, where $d_{max} = \max\{dist(X, Y) | X, Y \in \Omega\}$, and β is a user-defined real number lower than one. Note that d must not be too small; otherwise, we will have $\omega_j(X, d) \approx \emptyset$.

Definition 2 For each instance $X \in \Omega$, we define in Equation 3 the M-dimensional vector $\sigma(X, d)$ whose jth component is the percentage of instances of class c_j found in $\omega_j(X, d)$:

$$\sigma(X, d) = (\sigma_1(X, d), \sigma_2(X, d), \ldots, \sigma_M(X, d)), \text{ where}$$
$$\sigma_j(X, d) = \frac{100}{|c_j|} \times |\omega_j(X, d)|, \; 1 \leq j \leq M \tag{3}$$

Depending on the value of d, the components of $\sigma(X, d)$ can be very small real numbers. We thus propose to normalize $\sigma(X, d)$ in the next definition.

Definition 3 For each instance $X \in \Omega$, we define in Equation 4 the M-dimensional vector $\tilde{\sigma}(X, d)$ whose jth

$$\begin{array}{lccccccc} \text{Classes } (symbols): & c_{i_1} & \dots & c_{i_k} & \dots & c_{i_M} \\ & \uparrow & \vdots & \uparrow & \vdots & \uparrow \\ \text{Slices } (states): & S_{j_1} & \rightarrow \dots \rightarrow & S_{j_k} & \rightarrow \dots \rightarrow & S_{j_M} \end{array}$$

Fig. 7 Markov chain associated with an instance $X \in \Omega$

component is equal to $\sigma_j(X,d)$ normalized according to the value $\sigma_{max} = \max\{\sigma_j(X,d) | X \in \Omega \text{ and } 1 \leq j \leq M\}$:

$$\begin{aligned} &\tilde{\sigma}(X,d) = (\tilde{\sigma}_1(X,d), \tilde{\sigma}_2(X,d), \dots, \tilde{\sigma}_M(X,d)), \text{ where} \\ &\tilde{\sigma}_j(X,d) = \frac{\sigma_j(X,d)}{\sigma_{max}} \times 100,\ 1 \leq j \leq M \end{aligned} \tag{4}$$

4.3.2 Transformation of an instance

Given a radius d, the following steps are executed in order to transform the instances of Ω into Markov chains:

1. For every instance $X \in \Omega$ and each class $c_j (1 \leq j \leq M)$, compute $\sigma_j(X,d)$ using Equation 3 and save the highest value σ_{max}.
2. For every instance $X \in \Omega$, calculate the vector $\sigma(X,d)$ using Equation 3.
3. Use σ_{max} to normalize the components of every vector $\sigma(X,d)$ and save the result as the vector $\tilde{\sigma}(X,d)$ using Equation 4.
4. Sort the components of every vector $\tilde{\sigma}(X,d)$ in decreasing order and save the result in the vector $\bar{\sigma}(X,d)$.
5. For each vector $\bar{\sigma}(X,d)$:
   (a) Start browsing $\bar{\sigma}(X,d)$ by initializing k to 1.
   (b) Determine the interval S_{j_k} containing the kth component of $\bar{\sigma}(X,d)$ and determine the class c_{i_k} associated with this component.
   (c) Assign S_{j_k} to the current state and c_{i_k} to the current symbol.
   (d) If $k < M$, then set $k = k + 1$ and **go to** step 5-(b).
   (e) Stop the browsing of $\bar{\sigma}(X,d)$ and return the Markov chain illustrated in Fig. 7.

4.3.3 Example

Figure 8 depicts three classes c_1, c_2 and c_3 represented by circles, stars and squares in the affine space $\mathbb{R}^2$. Let us consider that the value of d is $\sqrt{74}$ which is the *Euclidean* distance between coordinates (13, 11) and (8, 4).

The algorithm presented in Sect. 4.3.2 can be applied as follows to construct the Markov chain associated with the instance X of class c_1 located at coordinates (13, 11):

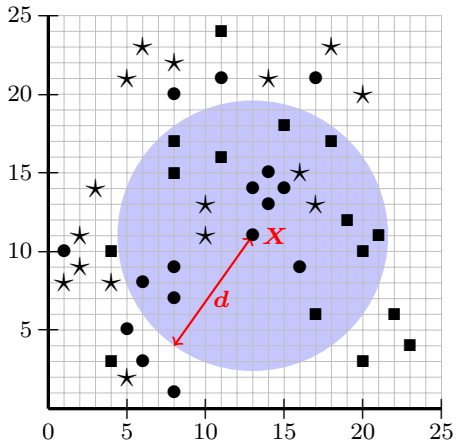


Fig. 8 Example of three classes c_1, c_2 and c_3 represented by circles, stars and squares in $\mathbb{R}^2$. The circled area represents the neighborhood $\aleph_X$ of the instance $X \in c_1$ located at coordinates (13, 11) when $d = \sqrt{74}$

$$\begin{array}{lccccc} \text{Classes } (symbols) & c_3 & & c_1 & & c_2 \\ & \uparrow & & \uparrow & & \uparrow \\ \text{Slices } (states) & S_8 & \rightarrow & S_7 & \rightarrow & S_4 \end{array}$$

Fig. 9 The Markov chain associated with the instance X of class c_1 located at coordinates (13, 11) in Fig. 8

- **Steps 1 to 3**: With Equation 3, we calculate $\sigma(X,d) = 100 \times (\frac{8}{16}, \frac{4}{16}, \frac{9}{15}) = (50, 25, 60)$. After calculations, we obtained $\sigma_{max} = 75\%$. This highest value is obtained for the instance of class c_1 located at coordinates (8, 9) whose neighborhood contains $\frac{12}{16}$ instances of class c_1. Equation 4 is then used to normalize $\sigma(X,d)$ to derive $\tilde{\sigma}(X,d) = 100 \times (\frac{50}{75}, \frac{25}{75}, \frac{60}{75}) = (66.6, 33.3, 80)$.
- **Step 4**: The components of $\tilde{\sigma}(X,d)$ are sorted into decreasing order and saved into $\bar{\sigma}(X,d) = (80, 66.6, 33.3)$. It is important to note that the classes of the three components of $\bar{\sigma}(X,d)$ are c_3, c_1 and c_2, respectively.
- **Step 5**: If we fix $N = 10$, then the interval [0, 100] is split into the ten following slices using Equation 1: $S_1 = [0, 10], S_2 =]10, 20]$, …, and $S_{10} =]90, 100]$. Therefore, the three components of $\bar{\sigma}(X,d)$, respectively, belong to the slices (S_8, S_7, S_4). Finally, the Markov chain of Fig. 9 is derived.

4.3.4 Time cost of the transformation

The goal of this section is to evaluate the time cost of the transformation of all the instances of Ω into Markov chains,

given a fixed radius d. Let us first evaluate the number of comparisons as follows:

1. For each $X \in \Omega$, the computation of $\sigma_j(X, d)$ involves $|c_j|$ comparisons between $dist(X, Y)$ and d.
2. The computation of $\sigma(X, d)$ therefore costs $(\sum_{j=1}^{M} |c_j|)$ which gives $|\Omega|$ comparisons for each instance. The global number of comparisons for all the instances of Ω is therefore $|\Omega|^2$.
3. There is no comparison during the calculation of $\tilde{\sigma}(X, d)$.
4. For each instance $X \in \Omega$, the sorting of $\tilde{\sigma}(X, d)$ to obtain $\bar{\sigma}(X, d)$ costs around M^2 comparisons (this can be reduced to $M.log(M)$ depending on the selected sorting algorithm). The global cost for all the instances of Ω is therefore $|\Omega|.M^2$.
5. For each instance $X \in \Omega$, the search of the interval associated with the kth component of $\bar{\sigma}(X, d)$ takes around N comparisons between that component and the lower bound of each interval, where N is the user-defined number used to split the interval [0,100]. This gives a cost of $M.N$ comparisons for all the components of $\bar{\sigma}(X, d)$ and, therefore, a cost of $|\Omega|.M.N$ comparisons for all the instances of Ω.

When we sum all the previously calculated values, we deduce that the algorithm takes around $(|\Omega|^2 + |\Omega|(M^2 + M.N))$ comparisons. If we now suppose that the computation of the distance/dissimilarity $dist(X, Y)$ between two instances of Ω needs around α arithmetic operations, then the number of arithmetic operations required by the algorithm is evaluated as follows:

1. For each $X \in \Omega$, the computation of $\sigma_j(X, d)$ involves two arithmetic operations augmented by $|c_j|$ computations of the distance/dissimilarity between two instances of Ω. Therefore, this step costs $(\alpha.|c_j| + 2)$ arithmetic operations.
2. The computation of $\sigma(X, d)$ needs $\sum_{j=1}^{M} (\alpha.|c_j| + 2)$ which gives $\alpha.|\Omega| + 2M$ arithmetic operations per instance. Thus, global number of arithmetic operations for all the instances of Ω is $\alpha.|\Omega|^2 + 2M.|\Omega|$.
3. For each instance $X \in \Omega$, the calculation of the vector $\tilde{\sigma}(X, d)$ involves $2M$ arithmetic operations. Therefore, the global cost for all the instances of Ω is therefore $2M.|\Omega|$.
4. For each instance $X \in \Omega$, the sorting of $\tilde{\sigma}(X, d)$ to obtain $\bar{\sigma}(X, d)$ costs around M^2 arithmetic operations. The global cost for all the instances of Ω is therefore $|\Omega|.M^2$.
5. No arithmetic operation is involved during this step.

When we sum all these values, we deduce that the algorithm takes around $(\alpha.|\Omega|^2 + 4M.|\Omega| + M^2)$ arithmetic operations. If we now consider both, comparisons and arithmetic operations, we obtain the following time complexity for the algorithm: $|\Omega|^2(\alpha + 1) + |\Omega|(2M^2 + 4M + M.N)$. Given that the value of $|\Omega|$ will generally be very high compared to the values of M and N, then we conclude that the algorithm runs in $\theta(\alpha.|\Omega|^2 + |\Omega|.M^2)$.

Fig. 10 Design of the HMM λ_i of class c_i

4.4 Design of the HMMs

In order to derive the HMM associated with a class c_i whose Markov chains are contained in the set δ_i, the parameters of an initial HMM λ_i^0 are first computed. These parameters must statistically capture the state transitions and the distributions of symbols from the content of δ_i. The parameters of λ_i^0 are later readjusted in order to avoid eventual divisions by zero and zero probabilities. The readjusted model λ_i^1 is finally trained, using the content of δ_i and the *Baum–Welch* algorithm, to derive the final model λ_i. Figure 10 summarizes this process, and the three main steps are formally presented in the following subsections.

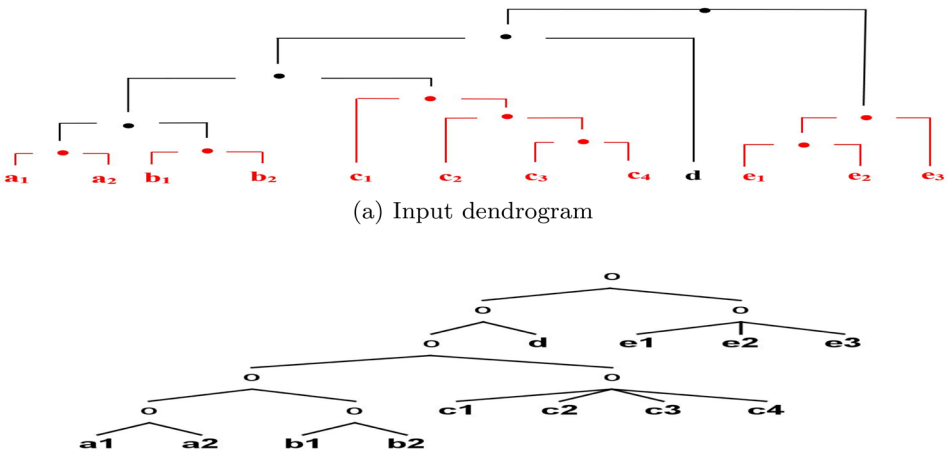
4.4.1 Initial HMMs

In order to guarantee that the initial models fit the raw data, the parameters of each initial HMM λ_i^0 of class c_i are set to statistically capture the state transitions and the distributions of symbols from the content of δ_i as follows:

1. The number of states is the user-defined positive integer N, and the set of states is $S = \{S_1, S_2, \ldots, S_N\}$, as defined in Equation 1.
2. The number of symbols is the number M of classes, and the set of symbols is $V = \{c_1, c_2, \ldots, c_M\}$.
3. Given a user-defined floating-point number $\varepsilon > 0$, the probability of transition from state S_j to state S_k is given by Equation 5, where $transit(j, k, \delta_i)$ is the number of transitions from state S_j to state S_k in δ_i and $transit(j, -, \delta_i)$ is the number of transitions from state S_j to any destination in δ_i:

$$A_i^0[j, k] = \frac{transit(j, k, \delta_i)}{transit(j, -, \delta_i) + \varepsilon} \tag{5}$$

4. The probability to observe symbol c_k at state S_j is given by Equation 6, where $observe(k, j, \delta_i)$ is the number of times symbol c_k is observed at state S_j in δ_i and $observe(-, j, \delta_i)$ is the number of occurrences of state S_j in δ_i:



(a) Input dendrogram

(b) Generated taxonomy

Fig. 11 Example of taxonomy derivation from a dendrogram

$$B_i^0[j,k] = \frac{observe(k,j,\delta_i)}{observe(-,j,\delta_i)+\varepsilon} \tag{6}$$

5. The probability that a sequence starts with state S_j is given by Equation 7, where $start(j,\delta_i)$ is the number of elements in δ_i starting with state S_j:

$$\pi_i^0[j] = \frac{start(j,\delta_i)}{|\delta_i|+\varepsilon} \tag{7}$$

4.4.2 Readjustment of the initial HMMs

The parameters of λ_i^0 are not probability distributions. This inconvenience was intentionally introduced by adding ε to the denominators of each component of (A_i^0, B_i^0, π_i^0) in order to avoid eventual divisions by zero and zero probabilities. In this paper, the value $\varepsilon = 1.0$ has been selected following [10, 49]. An equitable redistribution of the missing quantity is done to each element of each line in $\lambda_i^0 = (A_i^0, B_i^0, \pi_i^0)$ to obtain the readjusted model $\lambda_i^1 = (A_i^1, B_i^1, \pi_i^1)$ of class c_i whose parameters are:

1. $A_i^1[j,k] = A_i^0[j,k] + \frac{1}{N}\left(1 - \sum_{l=1}^{N} A_i^0[j,l]\right)$
2. $B_i^1[j,k] = B_i^0[j,k] + \frac{1}{M}\left(1 - \sum_{l=1}^{M} B_i^0[j,l]\right)$
3. $\pi_i^1[j] = \pi_i^0[j] + \frac{1}{N}\left(1 - \sum_{l=1}^{N} \pi_i^0[l]\right)$

4.4.3 Training of the HMMs

The readjusted initial HMM λ_i^1 of class c_i is trained to recognize the Markov chains in δ_i using the *Baum–Welch* algorithm. The resulting HMM λ_i is the final model associated with class c_i. During this training phase, the training sequences are exclusively composed of symbols (classes). If we consider, for example, the Markov chain presented in Fig. 9 obtained for the instance X of Fig. 8, then the training sequence is $c_3c_1c_2$.

4.5 Computation of descriptor vectors

Given that a model λ_i is now attached to each class c_i, one can compute a descriptor vector $\overrightarrow{U_i}$ for class c_i. The jth component u_{ij} of $\overrightarrow{U_i}$ is the similarity rate between the models λ_i and λ_j as specified in Equation 8, where u_{ij} is the probability that λ_i and λ_j generate identical Markov chains:

$$\begin{cases} \overrightarrow{U_i} = (u_{i1}, u_{i2}, \ldots, u_{iM}) \text{ where} \\ u_{ij} = 100 \times d_{sim}(\lambda_i, \lambda_j) \text{ with } 1 \leq j \leq M \end{cases} \tag{8}$$

4.6 Taxonomy generation

To generate a taxonomy, we use the algorithm applied in [10]. This algorithm consists of two main phases:

1. The *hierarchical clustering*: This phase is realized by the *AHC* algorithm [26] that outputs a binary dendrogram of the M classes. The principle of the *AHC* algorithm is as follows: Initially, each class c_i is considered as a cluster of size 1. In consecutive steps, the two nearest clusters are merged into a new cluster. This process is repeated until one single cluster is obtained.
2. The *taxonomy derivation*: The taxonomy is finally derived from the dendrogram as follows: *The taxonomy follows the general shape of the dendrogram, except for the sub-trees of the dendrogram in which each node that is not a leaf has exactly two children including at least one leaf. All the leaves of such sub-trees are merged into a k-ary cluster, where k is the number of leaves in the*

sub-tree. When this principle is applied to the dendrogram of Fig. 11a, the taxonomy of Fig. 11b is obtained. In these figures, the leaves are the classes and the inner nodes are the clusters.

5 Experimental results

5.1 Experimental datasets

In order to evaluate the performance of the proposed approach, we performed hierarchical classification experiments on 20 machine learning datasets from various application domains, including 18 datasets from the *UCI Machine Learning Repository*. The selected datasets vary in the number of classes, in the type of variables (nominal, numerical) and in the sample size. The two last datasets are *GTZAN* and *GTZAN+* used in [9, 10]. All the datasets have at least four classes, and the classes having no instances have been removed. Typically, class '*4*' was removed from *Anneal*, class '*-3*' was removed from *Autos* and class '*vehic non float*' was removed from *Glass*. The full description of the datasets from the *UCI Machine Learning Repository* will not be presented here because of space constraints. Nevertheless, this description is available online. Short descriptions of *GTZAN* and *GTZAN+* are presented below. Table 5 summarizes the main characteristics of the 20 experimental datasets.

1. **GTZAN**: This dataset is among the most used dataset in music genre recognition and is composed of ten music genres, each genre being represented by 100 instances (songs). In this paper, the same timbre (34) and rhythm (401) music descriptors extracted in [9, 10] for each song of *GTZAN* and *GTZAN+* are considered. The main difference between the characteristics of [9, 10] and those used in this paper is that in [9] the taxonomy was generated using only timbre music descriptors, while rhythm music descriptors were only used for the same purpose in [10]. But in this paper, timbre and rhythm music descriptors are both used to generate the taxonomy.
2. **GTZAN+**: This dataset is obtained from *GTZAN* by adding five new Afro genres. The music descriptors used in *GTZAN* are also used in *GTZAN+*.

Table 5 Main characteristics of the 20 experimental datasets. Columns 3, 4 and 5, respectively, contain the number of classes, the number of nominal attributes and the number of numeric attributes of each dataset

Dataset	Authors	#Classes	#Nom.	#Num.	Size
Anneal	Sterling & Buntine	5	23	9	798
Audiology	Quinlan	24	69	0	226
Autos	Schlimmer	6	9	16	205
Car	Bohanec & Zupan	4	6	0	1728
Dermatology	Ilter & Guvenir	6	33	1	366
Glass	Spiehler	6	0	9	214
GTZAN	Tzanetakis	10	0	435	1000
GTZAN+	Tzanetakis & Iloga	15	0	435	1500
Hypothyroid	Quinlan	4	22	7	3772
Letter	Slate	26	0	16	20000
Lymph	Zwitter & Soklic	4	15	3	148
Nursery	Bohanec & Zupan	5	8	0	12960
Pendigits	Alpaydin & Alimoglu	10	0	16	10992
Primary-tumor	Kononenko & Cestnik	21	17	0	339
Satimage	Forsyth	6	0	36	6430
Segment	Brodley	7	0	19	2310
Soybean	Michalski & Chilausky	19	0	35	683
Vehicule	Mowforth & Shepherd	4	0	18	846
Vowel	Turney	11	3	10	990
Zoo	Forsyth	7	16	1	101

5.2 Experimental parameters

We used the distance presented in Equation 9 to handle datasets with nominal attributes, numeric attributes or both. This distance computes the sum of the *discrete* distances between all the nominal attributes and adds it to the *Manhattan* distance between all the numeric attributes. This measure is a special case of the weighted version presented in 2017 by Yizhou Sun in his online course [4] (See page 25). The unweighted version is used here for two reasons. Firstly, if the weighted version was selected, the choice of each weight must objectively be justified. Additionally, the unweighted version has the nice property to coincide with the *Manhattan* distance when all the attributes are numeric. Missing values were considered as a nominal attributes with value '*?*'.

$$dist(x,y) = \sum_k f_k(x,y) \qquad \text{where}$$

$$f_k(x,y) = \begin{cases} |x(k) - y(k)| \text{ when the } k\text{th attribute is numeric} \\ 0 \text{ if } x(k) = y(k) \text{ and } 1 \text{ if } x(k) \neq y(k) \text{ otherwise} \end{cases} \quad (9)$$

[4] http://web.cs.ucla.edu/~yzsun/classes/2017Spring_CS249/Slides/09Evaluation_Clustering.pdf.

We experimented three different radii of the neighborhood for each dataset: $d_1 = 0.1 \times d_{max}$, $d_2 = 0.2 \times d_{max}$ and $d_3 = 0.3 \times d_{max}$. The value $N = 50$ has been used to split the interval [0, 100] into slices following [10] where the authors used HMMs to compare finite sets of histograms. To justify this choice, the authors studied for a specific example, the splitting of the interval [0, 100] in N slices, with N ranging from 10 to 100 with step of 5. They finally recommended to use the value $N = 50$. The *Baum–Welch* algorithm has been applied with a maximum number of iterations $\gamma = 100$ to avoid too high training time cost. During the experiments, the presence of many classes in identical proportions in the neighborhood of an instance was rarely observed. *AHC* involves the use of a pair of distances: one distance between two vectors and one distance between two clusters. The following distances also used in [9] have initially been selected in this work:

1. Between vectors: *Manhattan*, *Euclidean*, *Tchebychev* and *Cosine* distances.
2. Between clusters: *Minimum*, *Maximum*, *Average* and *Centroid* distances.

In unreported simulations, we observed that some of these distances always generate taxonomies with a high number of classes combined into one single cluster. Sometimes, a flat taxonomy (i.e., a taxonomy with a root that is the immediate parent of each class) was generated. We also observed that some distances often generate taxonomies containing a high number of isolated leaves, just like the isolated leaf *d* in Fig. 11b. Such taxonomies do not provide relevant information regarding the aim of this work. We therefore decided to exclude the considered distances. Hereafter, the *Centroid* distance is the unique selected distance between clusters. For vectors, the *Manhattan* and the *Euclidean* distances where initially preserved, but since there was no real gap between their final performances, only the results obtained by the *Manhattan* distance are presented here.

The descriptor vectors associated with the classes of the various experimental datasets are not presented in this paper for space constraints.

5.3 Hierarchical classification settings

Flat and hierarchical classification experiments were, respectively, performed in WEKA [50] and MEKA [5]. The two following basic classifiers have been selected for hierarchical classification (their corresponding names in WEKA/MEKA are in brackets): SVM with polynomial kernel (*SMO*) and multilayers perceptron (*MLP*). These classifiers have been used with their default settings, and a tenfold cross-validation (90% − 10%) has been applied.

5.4 Classification performances

For space constraints, only the best taxonomy derived for each dataset is graphically presented in Figs. 12a to o and 13a to e. In these figures, an identifier is assigned to each cluster and the names of the classes are attached to the leaves. Flat and hierarchical classification accuracies obtained for each dataset are presented in Table 6.

Table 6 reveals that for each dataset, the best hierarchical classification accuracy is always higher than the best flat classification accuracy, unlike what was observed in [4–8]. We experimentally observed that for few datasets, the same taxonomy was derived for different radii of the neighborhood. The impact of the radius on the classification accuracy was different from one dataset to another. The best experimental results were obtained 9/20 for the radius d_1, 10/20 for the radius d_2 and 11/20 for the radius d_3. Therefore, for any new dataset, we recommend to perform experiments with these three values of the radius, and to preserve the radius that exhibits the best performance.

5.5 Comparisons with related work

The approach proposed here has been compared with the techniques of Table 1 where hierarchical classification is performed [4–10]. The comparison with the remaining techniques found in Table 1 [2, 3, 11–13, 13–15] cannot be realized here because the terms (or key terms) involved in these works are not initially organized into separated classes. Indeed, the proposed approach relies on the topological analysis of the neighborhood of each instance. More precisely, the frequencies and the dominance of the various classes found in this neighborhood are captured. Consequently, this topological analysis cannot be performed for the terms appearing in [2, 3, 11–13, 13–15] due to the lack of classes. Table 7 presents the results of this comparison for the 20 experimental datasets including 18 datasets from the *UCI Machine Learning Repository*.

In Table 7, only the best accuracy of each work for each dataset is presented and the last column contains the value of accuracy gain (i.e., the best accuracy of the current work minus the best accuracy of the existing works on the considered dataset). Table 7 reveals that the approach proposed here outperforms most of the existing approaches with accuracy gains reaching +38.62% for the dataset *Primary-tumor*. Nevertheless, existing approaches slightly

[5] https://sourceforge.net/projects/meka/.

Table 6 Hierarchical classification performances using SMO and MLP as basic classifiers. Accuracies are in (%), and the best accuracies are bold

Dataset	Flat		d_1		d_2		d_3	
	SMO	MLP	SMO	MLP	SMO	MLP	SMO	MLP
Anneal	97.43	98.99	98.4	**99.2**	**98.4**	**99.2**	98.4	**99.2**
Audiology	81.85	83.18	**92.0**	90.7	90.3	91.2	90.7	91.6
Autos	71.22	80.0	88.8	88.3	81.5	87.3	**94.6**	94.1
Car	93.75	99.53	95.1	**99.99**	95.1	**99.99**	95.1	**99.99**
Dermatology	95.08	96.17	**100**	**100**	71.4	70.9	71.4	70.9
Glass	56.07	67.75	70.6	82.7	90.2	**95.3**	88.8	86
GTZAN	72.3	68.4	95.5	92.2	93.7	94.1	91.9	**96.0**
GTZAN+	75.5	41.9	96.0	96.1	91.1	91.9	96.3	**97.0**
Hypothyroid	93.61	94.16	94.8	95.5	98.7	**99.0**	98.7	**99.0**
Letter	82.34	82.08	98.3	**99.7**	97.6	**99.7**	97.1	99.6
Lymph	86.48	84.45	86.48	84.45	**87.2**	83.8	**87.2**	83.8
Nursery	93.07	99.72	**100**	**100**	98.9	**100**	**100**	**100**
Pendigits	97.96	94.61	99.7	**99.9**	99.7	96.7	97.7	**99.9**
Primary-tumor	46.9	38.34	85.5	85.8	**86.4**	83.3	83.8	80.8
Satimage	86.78	89.73	98.8	**99.5**	95.0	96.0	96.1	97.0
Segment	93.07	96.14	97.8	99.1	95.4	98.2	99.4	**99.7**
Soybean	93.85	93.41	94.17	93.9	**100**	**100**	99.4	99.7
Vehicule	74.34	81.67	93.9	**99.1**	79.4	84.4	79.4	84.4
Vowel	71.41	92.82	85.6	99.3	87.0	99.2	96.1	**99.99**
Zoo	96.03	96.03	98.0	97.0	**100**	**100**	98.0	98.0

performed better than the proposed approach, for the datasets *Hypothyroid* and *Pendigits* with low accuracy gaps of −0.36 and −0.01, respectively. But these two isolated cases do not have a major impact on the overall behavior of the proposed approach compared to existing techniques. Indeed, when the best hierarchical classification accuracy for each dataset is considered in the current work, an average best classification accuracy of 97.22% is obtained, with a standard deviation of 4.11%.

In Table 8, we reconsider the seven existing works presented in the columns of Table 7. For each of these existing works, we put :

- In column 1, the reference.
- In column 2, the number (#DS) of experimental datasets used in that paper.
- In column 3, the average accuracy ($\bar{x}$) obtained in that reference, for corresponding experimental datasets.
- In column 4, the average accuracy ($\bar{x}_{this}$) obtained with the method presented here, for the corresponding experimental datasets.
- In column 5, the average standard deviations (σ) obtained in that reference.
- In column 6, the average standard deviation (σ_{this}) obtained by the method presented here, for corresponding experimental datasets.

Clearly, for the experimental datasets used in the existing works, and taking as criteria the average and the standard deviation of the accuracy, the method presented here outperforms all the existing works.

6 Conclusion

The main objective of this work was to propose an efficient generic approach for automatic taxonomy generation. The design of these taxonomies relies on the hypothesis that the similarity between two classes can be derived from the topological analysis of the neighborhood of their instances. This topological analysis of the neighborhood of an instance consists in detecting the classes appearing in this neighborhood and to sort them from the most dominant class to the least dominant one. The result of this analysis is then saved as a Markov chain. Given a database Ω composed of many objects belonging to discrete classes, the collection of Markov chains resulting from the topological analysis of the content of each class $c_i \in \Omega$ $(1 \leq i \leq M)$ is used to train a model λ_i associated with c_i. The similarities between the resulting models are then calculated to associate a descriptor vector to each class. Finally, a hierarchical clustering algorithm

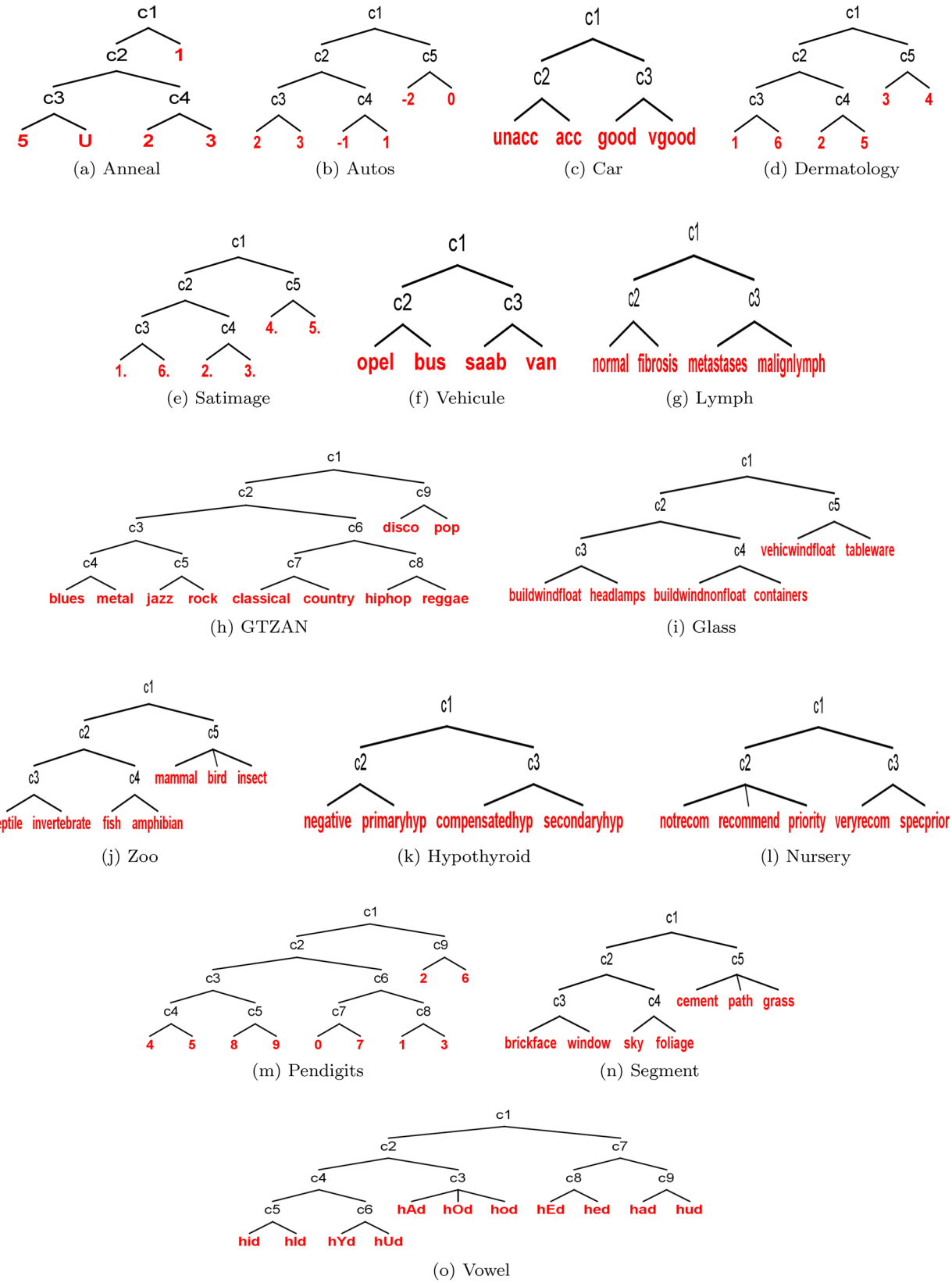
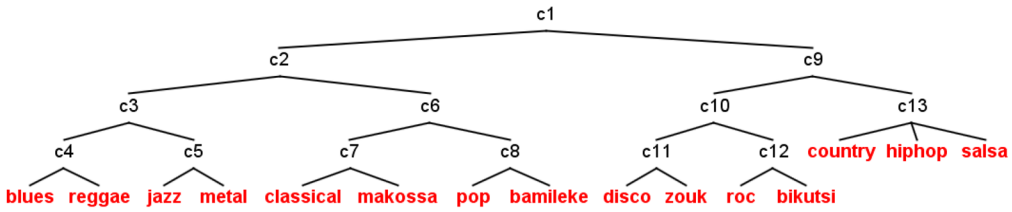
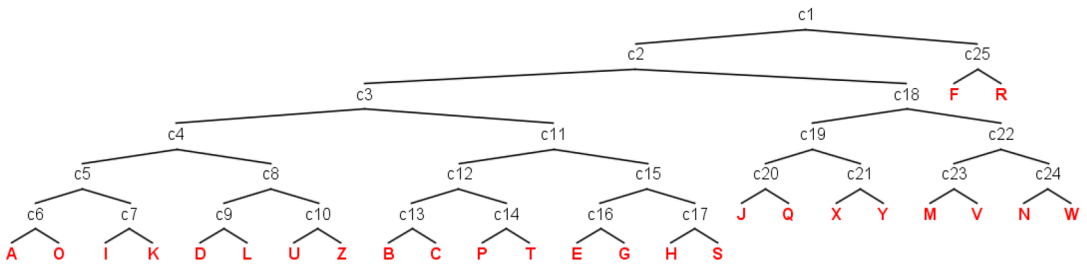


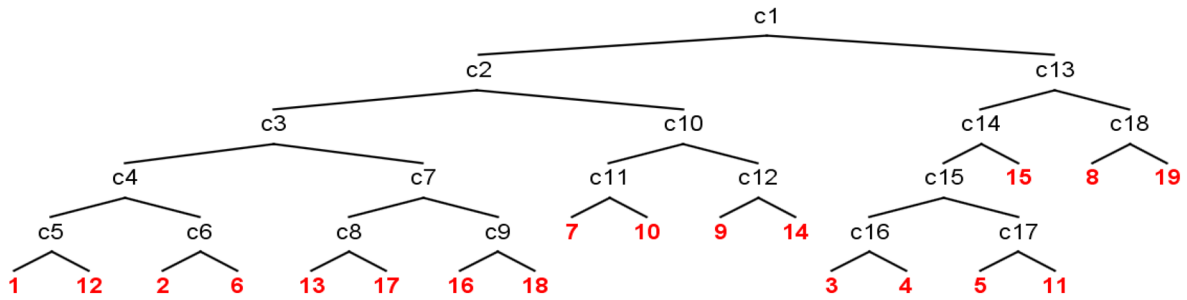
Fig. 12 Best taxonomies generated for the datasets *Anneal*, *Autos*, *Car*, *Dermatology*, *Glass*, *GTZAN*, *Hypothyroid*, *Lymph*, *Nursery*, *Pendigits*, *Satimage*, *Segment*, *Vehicule*, *Vowel* and *Zoo*



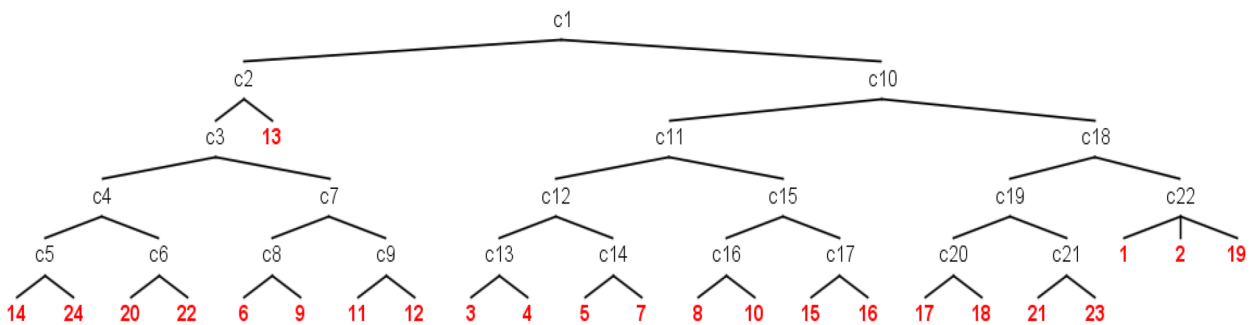
(a) GTZAN+



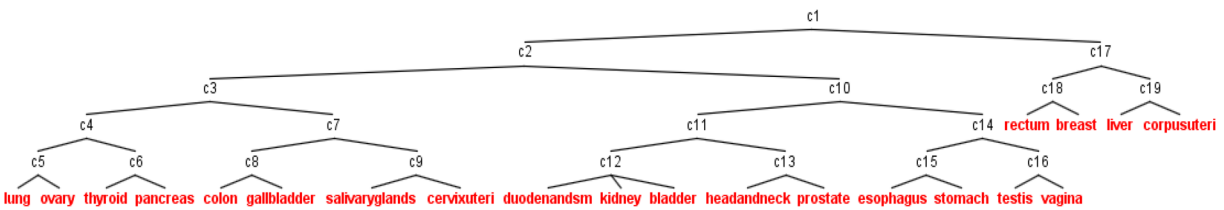
(b) Letter



(c) Soybean



(d) Audiology



(e) Primary-tumor

Fig. 13 Best taxonomies generated for the datasets *Audiology*, *GTZAN+*, *Letter*, *Primary-tumor* and *Soybean*

Table 7 Comparison with existing works for the 20 experimental datasets including 18 datasets from the *UCI Machine Learning Repository*. Accuracies are in (%), and the best accuracies are bold

Dataset	This work	Kang et al. [4]	Punera et al. [5]	Kang et al. [7]	Jo et al. [6]	Cagliero et al. [8]	Iloga et al. [9]	Iloga et al. [10]	Acc.gain
Anneal	**99.2**	98.99		99.11	73.67	96.6			+0.09
Audiology	**92.0**	76.99		74.34					+15.01
Autos	**94.6**	86.82		77.56					+7.78
Car	**99.9**	86.16		96.59	88.63				+3.31
Dermatology	**100**	98.08		96.99	95.49				+1.92
Glass	**95.3**	80.84	73.0	71.96					+14.46
GTZAN	**96.0**						84.9	91.6	+4.4
GTZAN+	**97.0**						92.0	85.2	+5.0
Hypothyroid	99.0	95.78		**99.36**		99.3			-0.36
Letter	**99.7**	70.53		81.72		93.6			+6.1
Lymph	**87.2**	84.45		77.03					+2.75
Nursery	**100**	90.32		99.06	98.86	98.7			+0.94
Pendigits	99.99		97.7			**100**			-0.01
Primary-tumor	**86.4**	47.78		44.54					+38.62
Satimage	**99.5**					88.7			+10.8
Segment	**99.7**	90.0		95.11		95.9			+3.8
Soybean	**100**	94.58		93.85	89.82	93.6			+5.42
Vehicule	**99.1**	67.84		70.21					+28.89
Vowel	**99.9**	42.42	76.87	90					+9.9
Zoo	**100**	96.03		93.07	91.41				+3.97

Table 8 Performances of existing works and the method presented in this paper. The values of $\bar{x}$, $\bar{x}_{this}$, σ and σ_{this} are in (%). The best values are in bold

Authors [references]	# DS	$\bar{x}$	$\bar{x}_{this}$	σ	σ_{this}
Iloga et al. [9]	2	88.45	**96.5**	3.55	**0.5**
Iloga et al. [10]	2	88.4	**96.5**	3.2	**0.5**
Punera et al. [5]	3	82.52	**98.39**	10.84	**2.18**
Jo et al. [6]	6	89.64	**99.85**	7.94	**0.29**
Cagliero et al. [8]	8	95.8	**99.62**	3.58	**0.34**
Kang et al. [4]	16	81.72	**97**	16.52	**4.49**
Kang et al. [7]	16	85.03	**97**	14.62	**4.49**

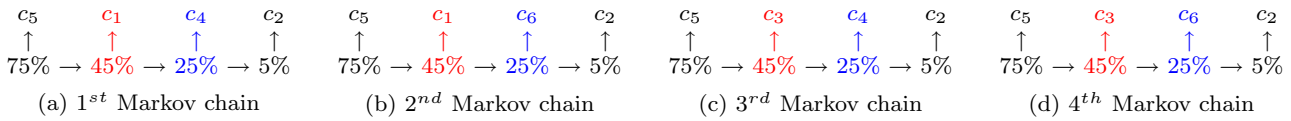
combined with a specific merging algorithm is applied on these vectors to derive the taxonomy.

The proposed approach has been experimented on 20 widespread datasets from various domains including 18 datasets belonging to the *UCI Machine Learning Repository*. Classification results showed an average accuracy of 97.22% with a standard deviation of 4.11%. The proposed approach outperforms existing techniques with accuracy gains reaching +38.62% for the dataset *Primary-tumor*.

The proposed approach is theoretically and experimentally better than existing techniques because:

1. It is based on the topological analysis of the neighborhood of each instance in order to capture the frequencies and the dominance of the various classes found in this neighborhood. This is a robust principle for detecting the proximity between classes.
2. It is highly flexible because the following generic parameters can be customized: the number N of slices used to split the interval [0, 100], the selected distance *dist* between two instances, the positive real number ϵ used during the design of the initial models and the radius d of the neighborhood of each instance. Future work may study the impact of the variation of these parameters on the performances of the proposed approach.
3. Unlike some existing techniques, it has never been beaten by any flat classifier.
4. It is efficient because it outperforms existing techniques in the hierarchical classification of the experimental datasets.

The fact that the proposed approach derives hierarchical classifiers that outperform the state of the art proves that the topological analysis of the neighborhood of each instance is a relevant idea. However, slightly significant error rates in hierarchical classification are observed for some datasets including *Audiology* (8%), *Autos* (5.4%), *Glass* (4.7%), *Lymph* (12.8%) and *Primary-tumor* (13.6%).

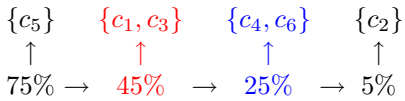


(a) 1^{st} Markov chain (b) 2^{nd} Markov chain (c) 3^{rd} Markov chain (d) 4^{th} Markov chain

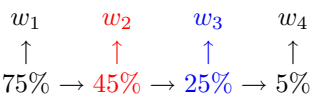
Fig. 14 The four Markov chains associated with X using the actual method

This justifies that a finer analysis of this neighborhood must be carried out in order to improve the hierarchical classification accuracy. This error rate may be due to the fact that the spatial distribution of the instances is ignored during the analysis of the neighborhood. Indeed, the fact that two instances are attached to identical Markov chains does not mean that the spatial distributions of the instances in their neighborhoods are the same. Such situations can introduce some bias during the analysis. Future work may propose a technique for neighborhood analysis that additionally takes into account the spatial distribution of the instances.

The technique proposed in Sect. 4.1 for the case where several classes are found in identical proportions in the neighborhood of an instance may lead to a combinatorial explosion of Markov chains. Such situations can considerably increase the HMMs training time. One way to attenuate this inconvenience will be studied in future work by considering the sets of classes having identical proportions in the neighborhood, as the symbols of the models. Let us, for example, assume that $M = 6$ and consider an instance X whose neighborhood has the following composition: $Presence(c_1, X) = 45\%$, $Presence(c_2, X) = 5\%$, $Presence(c_3, X) = 45\%$, $Presence(c_4, X) = 25\%$, $Presence(c_5, X) = 75\%$ and $Presence(c_6, X) = 25\%$. The method actually used will generate the four Markov chains presented in Fig. 14a to d for X. If the newly proposed method is used instead, the unique Markov chain presented in Fig. 15a is initially obtained. The next step consists of renaming the distinct sets of classes found in the database, i.e., $w_1 = \{c_5\}$, $w_2 = \{c_1, c_3\}$, $w_3 = \{c_4, c_6\}$, $w_4 = \{c_2\}$, ⋯



(a) Initial Markov chain



(b) Final Markov chain

Fig. 15 The initial and final unique Markov chain associated with X using the new method

Thereafter, the final unique Markov chain presented in Fig. 15b is obtained. In this way, only the number of symbols may increase, while the number of Markov chains remains unchanged.

References

1. Sujatha R, Bandaru R, Rao R (2011) Taxonomy construction techniques–issues and challenges. Indian J Comput Sci Eng IJCSE 2(5):661–671
2. Li T, Anand SS (2008) Automated taxonomy generation for summarizing multi-type relational datasets. In: International conference on data mining (DMIN 2008), Las Vegas, USA, pp 571–577
3. Treeratpituk P, Khabsa M, Giles CL (2013) Graph-based approach to automatic taxonomy generation (grabtax). arXiv preprint arXiv :1307.1718
4. Kang D-K, Silvescu A, Zhang J, Honavar V (2004) Generation of attribute value taxonomies from data for data-driven construction of accurate and compact classifiers. In: Fourth IEEE International Conference on Data Mining (ICDM'04), pp 130–137. IEEE
5. Punera K, Rajan S, Ghosh J (2006) Automatic construction of n-ary tree based taxonomies. In: null, pp 75–79. IEEE
6. Jo H, Na Y-C, Oh B, Yang J, Honavar V (2008) Attribute value taxonomy generation through matrix based adaptive genetic algorithm. In: 2008 20th IEEE International Conference on Tools with Artificial Intelligence, vol 1, pp 393–400. IEEE
7. Kang D-K, Sohn K (2009) Learning decision trees with taxonomy of propositionalized attributes. Pattern Recognit 42(1):84–92
8. Cagliero L, Garza P (2013) Improving classification models with taxonomy information. Data Knowl Eng 86:85–101
9. Iloga S, Romain O, Tchuenté M (2019) A sequential pattern mining approach to design taxonomies for hierarchical music genre recognition. Pattern Anal Appl 21(2):363–380
10. Iloga S, Romain O, Tchuenté M (2019) An accurate hmm-based similarity measure between finite sets of histograms. Pattern Anal Appl 22(3):1079–1104
11. Chien L-F, Huang C-C, Teng J-W, Chuang S-L (2002) Automatic taxonomy generation for speech archives. In: International Symposium on Chinese Spoken Language Processing
12. Yang H, Callan J (2009) A metric-based framework for automatic taxonomy induction. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pp 271–279
13. Liu X, Song Y, Liu S, Wang H (2012) Automatic taxonomy construction from keywords. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 1433–1441. ACM
14. Mao Y, Ren X, Shen J, Gu X, Han J (2018) End-to-end reinforcement learning for automatic taxonomy induction. arXiv preprint arXiv:1805.04044

15. Sánchez D, Moreno A (2004) Automatic generation of taxonomies from the www. In: International Conference on Practical Aspects of Knowledge Management, pp 208–219. Springer
16. Costa E, Lorena A, Carvalho ACPLF, Freitas A (2007) A review of performance evaluation measures for hierarchical classifiers. In: Evaluation methods for machine learning II: Papers from the AAAI-2007 workshop, pp 1–6
17. Sritha S, Mathumathi B (2016) A survey on various approaches for taxonomy construction. Indian J Innov Dev 5:6
18. Burred JJ, Lerch A (2003) A hierarchical approach to automatic musical genre classification. In: Proceedings of the 6th international conference on digital audio effects, pp 8–11. Citeseer
19. Li T, Ogihara M (2005) Music genre classification with taxonomy. In: Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005, vol 5, pp v–197. IEEE
20. Brecheisen S, Kriegel H-P, Kunath P, Pryakhin A (2006) Hierarchical genre classification for large music collections. In: 2006 IEEE international conference on multimedia and expo, pp 1385–1388. IEEE
21. Silla JCN, Freitas AA, et al (2009) Novel top-down approaches for hierarchical classification and their application to automatic music genre classification. In: SMC, pp 3499–3504
22. Zhang L , Liu S, Pan Y, Yang L (2004) Infoanalyzer: a computer-aided tool for building enterprise taxonomies. In: Proceedings of the thirteenth ACM international conference on Information and knowledge management, pp 477–483. ACM
23. Gates SC, Teiken W, Cheng K-SF (2005) Taxonomies by the numbers: building high-performance taxonomies. In: Proceedings of the 14th ACM international conference on Information and knowledge management, pp 568–577. ACM
24. Picca D, Popescu A (2007) Using wikipedia and supersense tagging for semi-automatic complex taxonomy construction. In: Computer aided language processing workshop 2007, Wolverhampton
25. Pachet F, Cazaly D (2000) A taxonomy of musical genres. In: Content-Based Multimedia Information Access-Volume 2, pp 1238–1245. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE
26. Sasirekha K, Baby P (2013) Agglomerative hierarchical clustering algorithm-a. Int J Sci Res Publ 83:83
27. Li T, Anand SS (2007) Diva: a variance-based clustering approach for multi-type relational data. In: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, pp 147–156. ACM
28. Karypis G, Kumar V (1998) Multilevel algorithms for multi-constraint graph partitioning. In: IEEE/ACM Conference on Supercomputing, 1998, SC98, pp 28–28. IEEE
29. Karypis G, Kumar V (1998) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput 20(1):359–392
30. Panchenko A, Faralli S, Ruppert E, Remus S, Naets H, Fairon C, Ponzetto SP, Biemann C (2016) Taxi at semeval-2016 task 13: a taxonomy induction method based on lexico-syntactic patterns, substrings and focused crawling. In: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pp 1320–1327, 2016
31. Bansal M, Burkett D, De MG, Klein D (2014) Structured learning for taxonomy induction with belief propagation. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 1041–1051
32. Tan P-N, Kumar V, Srivastava J (2002) Selecting the right interestingness measure for association patterns. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 32–41. ACM
33. Fayyad UM, Irani KB (1993) Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the 13th international joint conference on artificial intelligence (IJCAI-93), Chambèry, pp 1022–1027
34. Richard CD, Anil KJ (1988) Algorithms for clustering data. Prentice Hall, NJ
35. Thair NP (2009) Survey of classification techniques in data mining. Proc Int MultiConf Eng Comput Sci 1:18–20
36. Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu M-C (2001) Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In: ICCCN, pp 0215. IEEE
37. Lesh N, Zaki MJ, Oglhara M (2000) Scalable feature mining for sequential data. IEEE Intell Syst Appl 15(2):48–56
38. Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 77(2):257–286
39. Lidy TRA (2005) Evaluation of feature extractors and psychoacoustic transformations for music genre classification. In: ISMIR, pp 34–41
40. Bahlmann C, Burkhardt H (2001) Measuring hmm similarity with the bayes probability of error and its application to online handwriting recognition. In: ICDAR, p 0406. IEEE
41. Chen L, Man H (2005) Fast schemes for computing similarities between gaussian hmms and their applications in texture image classification. EURASIP J Adv Signal Process 2005(13):164742
42. Falkhausen M, Reininger H, Wolf D (1995) Calculation of distance measures between hidden Markov models. In: Fourth European Conference on Speech Communication and Technology
43. Lyngso RB, Pedersen CN, Nielsen H (1999) Metrics and similarity measures for hidden Markov models. In: Proc Int Conf Intell Syst Mol Biol, pp 178–186
44. Sahraeian SME, Yoon B-J (2011) A novel low-complexity hmm similarity measure. IEEE Signal Process Lett 18(2):87–90
45. Do MN (2003) Fast approximation of kullback-leibler distance for dependence trees and hidden Markov models. IEEE Signal Process Lett 10(4):115–118
46. Silva J, Narayanan S (2008) Upper bound kullback-leibler divergence for transient hidden Markov models. IEEE Trans Signal Process 56(9):4176–4188
47. Zeng J, Duan J, Chengrong W (2010) A new distance measure for hidden Markov models. Expert Syst Appl 37(2):1550–1555
48. Tan P-N, Steinbach M, Kumar V (2016) Introduction to data mining. Pearson Education India
49. Iloga S, Romain O, Bendaouia L, Tchuente M (2014) Musical genres classification using Markov models. In: 2014 international conference on audio, language and image processing (ICALIP), pp 701–705. IEEE
50. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. ACM SIGKDD Explor Newslett 11(1):10–18