Data Min Knowl Disc (2009) 19:227–244
DOI 10.1007/s10618-009-0141-6

Taxonomy-driven lumping for sequence mining

Francesco Bonchi · Carlos Castillo · Debora Donato · Aristides Gionis

Received: 12 June 2009 / Accepted: 24 June 2009 / Published online: 21 July 2009
Springer Science+Business Media, LLC 2009

Abstract Given a taxonomy of events and a dataset of sequences of these events, we study the problem of finding efficient and effective ways to produce a compact representation of the sequences. We model sequences with Markov models whose states correspond to nodes in the provided taxonomy, and each state represents the events in the subtree under the corresponding node. By lumping observed events to states that correspond to internal nodes in the taxonomy, we allow more compact models that are easier to understand and visualize, at the expense of a decrease in the data likelihood. We formally define and characterize our problem, and we propose a scalable search method for finding a good trade-off between two conflicting goals: maximizing the data likelihood, and minimizing the model complexity. We implement these ideas in TAXOMO, a taxonomy-driven modeler, which we apply in two different domains, query-log mining and mining of moving-object trajectories. The empirical evaluation confirms the feasibility and usefulness of our approach.

Keywords Data mining · Sequence analysis · Markov models · Query-log analysis · Spatial-data analysis

---

Responsible editors: Aleksander Kołcz, Wray Buntine, Marko Grobelnik, Dunja Mladenic, and John Shawe, Taylor.

F. Bonchi · C. Castillo · D. Donato · A. Gionis (✉)
Yahoo! Research, Diagonal 177, Barcelona 080018, Spain
e-mail: gionis@yahoo-inc.com

F. Bonchi
e-mail: bonchi@yahoo-inc.com

C. Castillo
e-mail: chato@yahoo-inc.com

D. Donato
e-mail: debora@yahoo-inc.com

1 Introduction

Many real-world application domains are naturally equipped with hierarchically organized ontologies or taxonomies, and such knowledge can be useful when modeling the data because, first, we can produce more compact, meaningful and understandable abstractions of the data by presenting it in terms of more general nodes in the taxonomy, and second, taxonomies can constraint the search space of data-mining algorithms, allowing to devise more efficient and scalable techniques.

Taxonomy-driven data mining has been mainly considered in the context of frequent pattern extraction: originally taxonomies were used for mining association rules (Srikant and Agrawal 1995) and sequential patterns of itemsets (Srikant and Agrawal 1996) in market-basket data, where each item is a member of a hierarchy of product categories. More recently taxonomy-based methods were used for mining frequent-subgraph patterns in biological pathways, where graphs of interacting proteins annotated with functionality concepts form a very large taxonomy (Cakmak and Özsoyoglu 2008).

The problem we address in this paper is of a different nature. Given a taxonomy tree on a set of events (or symbols, or items), and a dataset of event sequences, we study the problem of finding efficient and effective ways of producing a compact representation of the sequences. We then use this representation to cluster the sequences.

Example scenarios. Many different application domains fit within our framework: in general any problem regarding user profiling, where the set of possible actions is hierarchically structured. As an example consider a large web site containing different pages and providing different services. The site owner may be interested in profiling users with respect to their activities (Manavoglu et al. 2003), and understanding which pages and services appear to be sequentially connected.

As another example application consider the interaction between a software system and its users. The system may record user activities, that can later be analyzed in order to understand how users interact with the system and how to improve it. User actions are commands which are naturally organized in a hierarchy, e.g. in the different toolbars and menus of the software system.

Other contexts in which our framework is useful are the two applications that we present in Sect. 5: (1) query-log mining, where user queries are classified in a topical taxonomy, and (2) mining trajectories of moving objects, where the hierarchy is given by the natural spatial proximity. In all of the above-mentioned applications it is interesting and challenging to automatically define the most appropriate level of granularity to represent the information.

Sequence model. The representation we adopt is a Markov model. The states of the model are nodes in the taxonomy, where the last level (leaves) contains the observable symbols. Upon visiting each state, the Markov model emits one symbol, which can be any leaf in the sub-tree under the node corresponding to that state. Since we speak about transitions and emissions, one notices immediately the connection to hidden Markov models (HMMs). However, our model is not an HMM: due to the one-to-many mapping from states to symbols, we can always recover exactly the emitting state.

By using internal nodes in the taxonomy tree to represent the Markov states, we decrease the likelihood of the data given the model with respect to a model created at the leaf level. The higher we go in the taxonomy the more we decrease the likelihood, but the more we gain in the simplicity of the obtained models, which help making them more general, more meaningful for the domain experts, and easier to visualize. As usual, we are faced with the model selection problem, and the decision between alternative models needs to be taken based on the following criteria: (1) the data likelihood should be maximal, and (2) the model complexity should be minimal. The approach of using (non-hidden) Markov states to represent disjoint subsets of symbols is inspired by the fundamental ideas of *lumpability* and *approximate lumpability* in Markov chains (see Sect. 2). In our case, in addition to lumping, we also consider the emission probabilities of the symbols at the lumped states to express the total likelihood of the data. We note that since the model we consider does not have hidden states we can bypass the computational challenges of the HMMs, such as the estimation of the model parameters using the Baum-Welch algorithm. In the model we consider the parameters that yield the maximum likelihood, and those are given by simple frequency counts. The computation of the likelihood of a sequence given a model is also done by counting, and not by using a sequence decoder such as the Viterbi algorithm. This allows our method to scale to large data sets.

Another nice feature of our model is that it provides a direct and effective visualization: e.g., in a trajectory clustering application, we can lay the lumped nodes of the Markov model directly over the geographic map, to represent large or small areas with the associated transition probabilities among them.

Contributions. In contexts where the data is organized in sequences events, and events are naturally structured in taxonomy, we provide the following contributions:

- We introduce the new problem of learning a Markov model in which the observed states can be grouped along the nodes of the hierarchy. We prove that the maximum likelihood model is the one with the maximum number of states. In order to reduce model complexity we suggest to optimize a function that balances the data likelihood and the number of states.
- We present a family of search algorithms based on state merging, for finding the best model. The commonly used agglomerative algorithm is a specific instance in the family of search algorithms we present, and our experimental results show that other strategies outperform it.
- We describe a direct and efficient evaluation method for computing the likelihood of a dataset after merging states in a model, enabling a fast exploration of the model space.
- We demonstrate the relevance of our problem on real-world applications, and we perform experiments on synthetic and real data sets that validate our algorithmic solution.

Roadmap. This paper is organized as follows. Section 2 surveys previous work related to ours. Section 3 formally states the problems we study, for which algorithms are presented in Sect. 4. We implement those algorithms and describe experimental results in Sect. 5. The last section presents our conclusions and outlines future work.

2 Related work

We survey previous related works related to ours, ranging from Hidden Markov Models to sequence clustering. There is a wealth of literature on these topics, and our coverage of previous work is by no means complete.

Hidden Markov Models. Markov chains are fundamental mathematical structures widely used for describing and predicting processes from natural and physical sciences, computer science and engineering systems. For a detailed discussion we refer the reader to e.g. (Tijms 1986). A Hidden Markov Model (HMM) is an extension of a Markov chain in which observable symbols are emitted in each of the states, but it is not possible to know exactly the current state from the symbol observed. The HMM parameters of a model can be adjusted by the well-known Baum-Welch method (Welch 2003) to increase the likelihood of observed sequences of symbols. Another class of techniques for learning HMM parameters from data are based on model merging (Stolcke and Omohundro 1994). These approaches start with a maximum likelihood HMM that directly encode all the observable samples. At each step more general models are produced by merging previous simpler submodels. The submodel space is explored using a greedy search strategy and the states to be merged are chosen so to maximize the data likelihood.

In our algorithm, the initial model is built starting from the relative frequencies of each sample and it is a maximum likelihood model for the data. Our model differs from the previous ones in two aspects. First, the constraint on how symbols are emitted allow us to infer the states directly, with the advantage of not having hidden states. Moreover, since the greedy search is “guided” by such a hierarchical structure, two states can be merged only if they have the same parent in the taxonomy. In this way, the hierarchy-based merging drastically reduces the search space.

State aggregation in Markov models. Many of the processes that can be represented by HMMs suffer from the state space explosion problem. Since minimizing memory and time is a crucial requirements for most of the applications, aggregation techniques (Meyer 1989; Simon and Ando 1961) for reducing the number of states have been extensively studied. Many approaches are based on the notion of *lumpability* (Kemeny and Snell 1959; White et al. 2000) that is a property of Markov chains for which there exists a partition of the original state space into aggregated states such that the aggregated Markov chain maintains the characteristics of the original one. Bicego et al. (2001) present a different approach that reduces the structure of HMM by partitioning the states using the bi-simulation equivalence, so that equivalent states can be aggregated in order to obtain a minimal set that do not significantly affect model performance. A simple heuristic for HMMs is to merge states that have the most similar emission probabilities, this is applied to the domain of gesture recognition in Lee and Kim (1999). It is worth noting the different perspective that we have: in our context states aggregation, or abstraction, is done for the sake of useful and actionable knowledge modeling, and not for reducing computational requirements.

Sequence clustering and applications. Sequence clustering is one of the most common tasks in sequence mining and bioinformatics. Such a task has been handled by using frequent subsequences or n-grams statistics as features (Guralnik and Karypis

2001; Manning et al. 1997) or considering the edit distances among all the candidate sequence (Wang et al. 2007). Traditional methods often require sequence alignment and do not handle efficiently variable-length sequences. One of the first works using HMM for sequence clustering was presented by Smyth (1997) who computed the pairwise-distance matrix for all the observed sequences by training an HMM for each sequence. The log-likelihood of each model given the sequence is used to cluster the sequences in K clusters using EM algorithm. In Law and Kwok (2000), the HMMs are used as cluster prototypes. The clustering is computed by a combined approach of the HMMs and a rival-penalized competitive learning procedures. Bicego et al. (2003) extend the paradigm introduced by Smyth (1997): they use HMMs to build a new representative space, where the features are the log-likelihoods of each sequence to be clustered with respect to a predefined number of HMMs trained over a set of reference sequences.

In Sect. 5 we experiment our method in the context of web usage mining. Markov models have been applied for modelling user web navigation sessions (Borges and Levene 2004), describing user behavior (Manavoglu et al. 2003), mining web access logs (Girolami and Kaban 2003; Felzenszwalb et al. 2004) and for query recommendation (Cao et al. 2008). The other application that we present in Sect. 5 is in the domain of mobility data analysis, a research area rapidly gaining a great deal of attention as witnessed by the amount of spatio-temporal data mining techniques that have been developed in the last years, see e.g. (Nanni and Pedreschi 2006; Lee et al. 2007, 2008; Li et al. 2007).

Markov models with or without hidden states, first-order or higher-order, with or without lumping of states, have been extensively applied to sequence mining in the past. To the best of our knowledge, the problem stated in the next section, that introduces a natural constraint and leads to a scalable algorithmic solution, has not been described before in the literature.

3 Problem statement

We first introduce our notation, and review some well-known background concepts, in order to facilitate the presentation of the method in the next section.

3.1 Preliminaries

We are given a set of r sequences $\mathcal{D} = \{\sigma_1, \ldots, \sigma_r\}$, over a set of $m > 2$ symbols $\Sigma = \{\alpha_1, \ldots, \alpha_m\}$, and a taxonomy $\mathcal{T}$, which is simply a tree whose leaves are the symbols in Σ. We also consider two special symbols: a *starting* symbol $\alpha_1 = \triangleright$, with which all sequences start, and a terminal symbol $\alpha_m = \square$, with which all sequences end. For any symbol $\alpha \in \Sigma$ we denote by c_α the total number of times that α appears in all sequences. We extend this notation by using $c_{\alpha\beta}$ to denote the total number of times that symbol β follows symbol α in all sequences.

Markov models. Consider a set of s states $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_s\}$. A *Markov model* on X is defined by *transitional probabilities* $p_{\mathbf{x},\mathbf{y}}$, for each pair of states $\mathbf{x}, \mathbf{y} \in X$, which determine the probability that the next state will be $\mathbf{y}$ given that the current state is $\mathbf{x}$. For all $\mathbf{x} \in X$, we have $\sum_{\mathbf{y}\in X} p_{\mathbf{x},\mathbf{y}} = 1$.

A *hidden Markov model* (HMM) is a Markov model, which at each state outputs a symbol from the set Σ. It is considered that the output symbols are *observed* while the states are *hidden*. The output of symbols is governed by emission probabilities $q_{\mathbf{x},\alpha}$, defined for each $\mathbf{x} \in X$ and all $\alpha \in \Sigma$. Again we have $\sum_{\alpha \in \Sigma} q_{\mathbf{x},\alpha} = 1$, for all $\mathbf{x} \in X$.

In our setting we consider a Markov model $\mathcal{M}$ with $s > 2$ states $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_s\}$, where each state $\mathbf{x} \in X$ corresponds to a set of symbols $A(\mathbf{x}) \subseteq \Sigma$, and we assume that $\{A(\mathbf{x})\}$ forms a *partition* of Σ. In other words, we have $\bigcup_{\mathbf{x} \in X} A(\mathbf{x}) = \Sigma$ and $A(\mathbf{x}) \cap A(\mathbf{y}) = \emptyset$ for all $\mathbf{x}, \mathbf{y} \in X$. Conversely, for a symbol α we denote by $\mathbf{x}(\alpha)$ the unique state to which α is assigned, i.e., $\alpha \in A(\mathbf{x}(\alpha))$. We assume that $\mathbf{x}_1$ and $\mathbf{x}_s$ are special states, used for denoting the starting and terminal symbols, and no other symbols are assigned to those states.

Overall a Markov model in our setting is denoted by $\mathcal{M} = (X, A, \mathbf{p}, \mathbf{q})$, where X is the set of states, A is the function mapping states to sets of symbols, and $\mathbf{p}$ and $\mathbf{q}$ are vectors containing the transition and emission probabilities.

Since our model is described in terms of transition and emission probabilities, it appears to be a hidden Markov model. However, as we mentioned in our introduction, due to the fact each symbol corresponds to a unique state, there are no truly hidden states. We give a more formal statement of the fact that there are no hidden states, because it is needed in the proof of Lemma 4. Given a model $\mathcal{M} = (X, A, \mathbf{p}, \mathbf{q})$ as described above, we can define a Markov model $\mathcal{M}' = (\Sigma, \mathbf{r})$, whose set of states is the set of symbols Σ. We can show that $\mathcal{M}$ and $\mathcal{M}'$ are equivalent, and that $\mathcal{M}'$ is a proper first-order Markov model with no hidden states.

Lemma 1 *Given a model $\mathcal{M} = (X, A, \mathbf{p}, \mathbf{q})$ with transition and emission probabilities there is an equivalent Markov model $\mathcal{M}' = (\Sigma, \mathbf{r})$ with no hidden states, where*

$$r_{\alpha,\beta} = p_{\mathbf{x}(\alpha),\mathbf{x}(\beta)} q_{\mathbf{x}(\beta),\beta}.$$

Likelihood of a dataset. Given a dataset of sequences $\mathcal{D} = \{\sigma_1, \ldots, \sigma_r\}$, and a Markov model $\mathcal{M} = (X, A, \mathbf{p}, \mathbf{q})$, we compute the likelihood of the data given the model as

$$L(\mathcal{D} \mid \mathcal{M}) = \prod_{\alpha,\beta \in \Sigma} \left(p_{\mathbf{x}(\alpha),\mathbf{x}(\beta)}\right)^{c_{\alpha\beta}} \prod_{\alpha \in \Sigma} \left(q_{\mathbf{x}(\alpha),\alpha}\right)^{c_\alpha}.$$

The first product is due to transitions among states, and the second product is due to emissions of symbols. As usual, to avoid numerical underflow it is convenient to work with the minus log-likelihood, which is

$$S_L(\mathcal{D} \mid \mathcal{M}) = -\sum_{\alpha,\beta \in \Sigma} c_{\alpha\beta} \log p_{\mathbf{x}(\alpha),\mathbf{x}(\beta)} - \sum_{\alpha \in \Sigma} c_\alpha \log q_{\mathbf{x}(\alpha),\alpha}. \tag{1}$$

Maximum-likelihood estimation. We now assume that we are given the input sequences $\mathcal{D} = \{\sigma_1, \ldots, \sigma_r\}$, and a Markov model $\mathcal{M}$ in which only the states X and the state-to-symbol mapping A have been specified. The task is to compute the transition and emission probabilities, $\mathbf{p}$ and $\mathbf{q}$, so that the $S_L(\mathcal{D} \mid \mathcal{M})$ function is minimized. It is well known that the maximum-likelihood probabilities are estimated as the observed frequencies

$$\overline{p}_{\mathbf{x},\mathbf{y}} = \frac{c_{\mathbf{xy}}}{c_{\mathbf{x}}}, \tag{2}$$

and

$$\overline{q}_{\mathbf{x},\alpha} = \begin{cases} c_\alpha / c_{\mathbf{x}} & \text{if } \alpha \in A(\mathbf{x}) \text{ and} \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $c_{\mathbf{xy}} = \sum_{\alpha \in A(\mathbf{x}); \beta \in A(\mathbf{y})} c_{\alpha\beta}$ is the total number of times that a symbol of state $\mathbf{x}$ is followed by a symbol of state $\mathbf{y}$, and $c_{\mathbf{x}} = \sum_{\alpha \in A(\mathbf{x})} c_\alpha$ is the total number of times that a symbol of state $\mathbf{x}$ appears in the data. We have.

Lemma 2 *Given a set of sequences $\mathcal{D}$, and a Markov model $\mathcal{M} = (X, A, \cdot, \cdot)$ for which only the states are prespecified, the optimal score of the minus log-likelihood function is given by*

$$S_L^*(\mathcal{D} \mid \mathcal{M}) = -\sum_{\mathbf{x},\mathbf{y} \in X} c_{\mathbf{xy}} \log \frac{c_{\mathbf{xy}}}{c_{\mathbf{x}}} - \sum_{\alpha \in \Sigma} c_\alpha \log \frac{c_\alpha}{c_{\mathbf{x}(\alpha)}}. \tag{4}$$

3.2 Problem definition: optimal model

The next task is to the model that yields the highest likelihood for a dataset. That is, among all possible cuts in the taxonomy tree, the cut that gives a model $\mathcal{M}^*$ that minimizes the score in Eq. 4. We have the following.

Lemma 3 *The Markov model that minimizes Eq. 4 is the "leaf-level" model with $|\Sigma| = m$ states $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, where $\mathbf{x}_i = \{\alpha_i\}$.*

Lemma 3 is a direct consequence of the following more general fact.

Lemma 4 *Consider Markov models $\mathcal{M}_1$ and $\mathcal{M}_2$ for which for every state $\mathbf{x}$ of $\mathcal{M}_1$ and every state $\mathbf{y}$ of $\mathcal{M}_2$ it is either $A(\mathbf{x}) \subseteq A(\mathbf{y})$ or $A(\mathbf{x}) \cap A(\mathbf{y}) = \emptyset$. In other words, the states of $\mathcal{M}_1$ is a sub-partition of the states of $\mathcal{M}_2$. Then*

$$S_L^*(\mathcal{D} \mid \mathcal{M}_1) \leq S_L^*(\mathcal{D} \mid \mathcal{M}_2).$$

Intuitively, we want models that have small number of states, since such models are more useful to understand the data, and they avoid overfitting. However, as the previous Lemma shows, there is a trade-off between likelihood and simplicity of the model. A natural problem to consider is finding the best model for a given number of states.

Problem 1 **(k-state-optimal model)** Given a set of sequences $\mathcal{D} = \{\sigma_1, \ldots, \sigma_r\}$ and a number k, find a Markov model $\mathcal{M}$ that has at most k states and minimizes the score $S_L^*(\mathcal{D} \mid \mathcal{M})$.

However, the constraint of using k states might be too stringent and in many cases we may not know which is the correct number of states. We would like to have an objective function that balances the likelihood score and the number of states. The problem we are facing is a typical model-selection problem, and many different approaches have been proposed, including minimum-description length (MDL) criteria, Bayesian information criterion (BIC), cross-validation methods, etc. We experimented with BIC (Schwarz 1978), and found that it does not perform well for the size of the data we were working with. Essentially, for large data, the logarithmic factor of the BIC formula is orders of magnitude smaller than the minus log-likelihood score, and thus there is no sufficient penalization for model complexity.

What we found to work well in practice is a model-selection objective in which the minus log-likelihood and the model complexity are considered together, and the task is to find the model that is as close as possible to a model with an ideal score (but probably not feasible). In particular, let $\mathcal{M}_0$ be the model with the minimum possible number of states $s_{\min}$ that achieves the maximum possible score $S^*_{L,\max}$, and let $\mathcal{M}_m$ be the model with the maximum possible number of states $s_{\max} = m$ that achieves the minimum possible score $S^*_{L,\min}$. Then, for a model $\mathcal{M}$ with s states that achieves score $S^*_L = S^*_L(\mathcal{D} \mid \mathcal{M})$, we define the objective function

$$\mathrm{Dist}^2(\mathcal{M}) = \left(\frac{s - s_{\min}}{s_{\max} - s_{\min}}\right)^2 + w \cdot \left(\frac{S^*_L - S^*_{L,\min}}{S^*_{L,\max} - S^*_{L,\min}}\right)^2.$$

The parameter w is a scale factor controlling the importance of the two terms. We then consider the corresponding model-selection problem.

Problem 2 Given a set of sequences $\mathcal{D} = \{\sigma_1, \ldots, \sigma_r\}$, find a Markov model $\mathcal{M}$ that minimizes the objective $\mathrm{Dist}^2(\mathcal{M})$.

We also find it useful to consider a hybrid objective function, in which we have an upper bound on a desirable number of states, and still want to minimize the objective $\mathrm{Dist}^2(\mathcal{M})$.

Problem 3 Given a set of sequences $\mathcal{D} = \{\sigma_1, \ldots, \sigma_r\}$ and a number k, find a Markov model $\mathcal{M}$ that has at most k states and minimizes the score $\mathrm{Dist}^2(\mathcal{M})$.

4 Algorithms

4.1 State-merging algorithm

Our main algorithm is a generic "bottom-up" search algorithm with respect to the taxonomy tree. The algorithm has the following components: (1) an objective function g; (2) a search policy p; (3) a priority queue $\mathcal{Q}$ of candidate models to evaluate; (4) a set $\mathcal{E}$ of models already evaluated.

We consider a family of objective functions $\{g\}$ defined as $g : \mathbb{R} \times \mathbb{N} \to \mathbb{R}$, where the first argument represents a minus log likelihood score and the second argument represents the number of states of a model. For Problems 1 and 3, if a model has more than k states we assume that g returns the value ∞.

The search policy p is a total ordering on pairs (v, s) where again v represents a log likelihood score and s represents the number of states of a model. The algorithm also takes as input a hierarchy $\mathcal{T}$ on the symbol set Σ, so that each state-merging step done by the algorithm is by merging the children of a single node $\mathbf{x}$ in $\mathcal{T}$ to $\mathbf{x}$.

The algorithm starts by putting in the queue $\mathcal{Q}$ the "leaf-level" model, i.e., the model with states $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, where $\mathbf{x}_i = \{\alpha_i\}$. In the main iteration, as long as the queue $\mathcal{Q}$ is not empty and a maximum number of iterations has not been reached, the model $\mathcal{M}$ with the best score according to the policy p is removed from the queue. The log-likelihood $v = S^*_L(\mathcal{D} \mid \mathcal{M})$ is evaluated for $\mathcal{M}$, and the number of states of $\mathcal{M}$ is assigned to s. The model $\mathcal{M}$ is inserted in the set $\mathcal{E}$ of evaluated models, along with the score $g(v, s)$. In addition, all models $\mathcal{M}_i$ that result from $\mathcal{M}$ with one merging according to the hierarchy $\mathcal{T}$ are generated. If $\mathcal{M}_i$ has not already been evaluated (check in $\mathcal{E}$), then $\mathcal{M}_i$ is put in the queue of candidates $\mathcal{Q}$, according to the ordering $p(v, s)$. When the algorithm terminates, it returns the model with the best score g, among all models that have been evaluated and put in the set $\mathcal{E}$.

Notice that each candidate model is inserted in $\mathcal{Q}$ with the minus log likelihood score of its parent, which is a lower bound on its own score.

Let $\mathcal{M}_i$ and $\mathcal{M}_j$ be two models to be compared, respectively with scores v_i and v_j and states s_i and s_j. We experiment with the three following policies.

(1) ProbabilityFirst: $p(v_i, s_i) > p(v_j, s_j)$ if $v_i < v_j$ or ($v_i = v_j$ and $s_i < s_j$);
(2) StatesFirst: $p(v_i, s_i) > p(v_j, s_j)$ if $s_i < s_j$ or ($s_i = s_j$ and $v_i < v_j$);
(3) DistSqFirst: $p(v_i, s_i) > p(v_j, s_j)$ if $g(v_i, s_i) < g(v_j, s_j)$.

In the first case, we favor the likelihood minimization with respect to the complexity of the model. The converse is done in the second case. Notice that the first iterations of the strategy StatesFirst correspond to a typical agglomerative algorithm that makes consequtive merges along the hierarchy. In the last case we balance the likelihood score and the number of states, selecting the model that minimizes the objective function.

Even though we do not know the complexity of Problem 3 and thus we cannot exclude the case that a polynomial solution exists, the search algorithm we suggest is an exhaustive search over the search space. In practice, the algorithm stops after a certain number of iterations and the best solution found at that point is returned. Thus the effect of the different search strategies is to explore different parts of the search space, and thus, to return different solutions.

4.2 Model-evaluation algorithm

Without the constraint imposed by the taxonomy, the search space of the algorithm is the set of all possible partitions of the symbol set Σ. Allowing to perform merges only along the hierarchy reduces dramatically the search space.

Even with the use of a hierarchy, the search space has exponential size, so in practice it is impossible to explore it completely for all but toy-size datasets. Finding a good solution depends mostly on using a search policy that allows to reach such a good solution fast. But assuming a given search policy, it is also very important to be able to evaluate fast candidate models.

The costly part of evaluating the objective function of a model is computing the score S^*_L. We note that for computing S^*_L we only need the counts c_α, and $c_{\alpha\beta}$. So we have to read the input dataset $\mathcal{D}$ only once, summarize all the information in the counts c_α and $c_{\alpha\beta}$, and then perform all evaluation using those counts. In fact, assuming a sparse representation of the matrix $c_{\alpha\beta}$, the number of its non-zero entries can be at most n, where n is the cummulative length of all sequences Considering also that $c_{\alpha\beta}$ is a matrix of dimension $m \times m$, the amount of space (and time) needed for the evaluation of each model is $O(\min\{n, m^2\})$.

A faster model evaluation can be done incrementally, by computing the score of a model with respect to the S_L^* score of its "parent" model, which has already been evaluated as a by-product of our bottom-up algorithm. In particular, consider a model $\mathcal{M}_1$ with state space X, and a child model $\mathcal{M}_2$ which is built from $\mathcal{M}_1$ by merging a subset of d states of $\mathcal{M}_1$ into a single state in $\mathcal{M}_2$. Lemma 2 allows us to express the difference in the S_L^* score of the two models in terms only of the counters $c_{\mathbf{xy}}$ that involve the states $\mathbf{x}$, $\mathbf{y}$ participating in the merging, without computing over the whole frequency matrix c. The evaluation of each child model can be done in time $O(\min\{n, md\})$ given that the score of the parent node is known. The update formula for the incremental evaluation is included in Appendix 2.

4.3 Clustering algorithm

Given the methods described above, we can devise a simple clustering solution based on them. In particular, we adopt standard Expectation-Maximization (EM) method in which cluster representatives are the models described so far.

Let k be the number of clusters we want to obtain; the clustering will be a partition of the sequences $\mathcal{D}$ into k sets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$. To initialize the method, an initial partition is done, in our particular case we do a random initial assignment but other initialization procedures can be used. The algorithm then proceeds iteratively. On each iteration, using the algorithms previously described, we find the model $\mathcal{M}_i^*$ that maximizes $S_L(\mathcal{D}_i \mid \mathcal{M})$ for each subset $\mathcal{D}_i$. Next, we scan the whole set of sequences $\mathcal{D} = \{\sigma_1, \ldots, \sigma_r\}$ and find for each sequence σ_j the index k_j^* that maximizes the likelihood of that particular sequence: $k_j^* = \text{argmax}_{k_j \in \{1,2,\ldots,k\}} S_L(\{\sigma_j\} \mid \mathcal{M}_{k_j}^*)$. Finally, we partition again the sequences such that sequence $\sigma_j \in \mathcal{D}_{k_j^*}$, that is, we re-assign each sequence to the partition whose model gives the maximum likelihood for that sequence. We stop when the fraction of elements re-assigned becomes negligible.

The model inference phase in each iteration is faster than the one using HMMs given the absence of hidden states. Also, in the evaluation phase for each sequence σ, one Viterbi evaluation having cost $m^2|\sigma|$ where m is the number of states, is turned into a simple summation of $2|\sigma|$ terms

5 Experiments

We built an efficient sequence mining tool implementing the ideas described in Sect. 4. Our reference implementation, named TAXOMO("TAXOnomy-driven markovian MOdeler") provides the following features:

– Model inference given a taxonomy, a set of sequences, and a search strategy.

– EM-based clustering given a taxonomy, a set of sequences, a number of clusters, and a search strategy for modeling each partition.
– Computation of log-likelihood of a set of sequences for a given model.
– Generation of a set of sequences given a model.

TAXOMO is written in Java, uses a sparse-matrix representation of the sequence database by the counters c_α and $c_{\alpha\beta}$, and uses the fast candidate model evaluation method presented in Sect. 4.2.

5.1 Datasets

We use TAXOMOto compare search strategies in three datasets having alphabets of 64, 100 and 1,024 symbols, respectively.

Known generative model dataset. We first experimented on a toy problem in which the generative model is known. We started with a binary tree with 64 leaves as a synthetic “taxonomy”. In this tree, we generated 100 tree-cuts at random, and used the leaves of each tree-cut as states. For each of the 100 sets of states, we generated a random Markov model, and a database of 10,000 sequences using that model.

Query-log dataset. Our set of queries is a sample of 85,000 queries obtained in 2008 from Yahoo! search engine. Each query in each session is automatically assigned to a topical category using majority voting among the Yahoo! directory categories in the top 200 results for the query. For instance if a user enters the three queries “`ferrari photos`”, “`motorcycles`”, and “`car racing`”, then the input sequence for our mining method would be: “`recreation/automotive/autos`”, “`recreation/automotive/motorcycle`” and “`recreation/sports/auto-racing`”. The categories form an unbalanced tree of maximum height 6 with 100 categories at the leaf level.

Trajectories dataset. We generate a large database containing 500,000 spatio-temporal points for 80,000 trajectories using Brinkhoff’s network-based synthetic generator of moving objects (Brinkhoff 2003), over the Oldenburg city map. In order to obtain sequences from the trajectories, the map has been discretized using a 32×32 regular grid, obtaining an alphabet of 1,024 symbols. To create a hierarchy over the points, a tree is created by taking an area covering the whole city as the root, and then recursively dividing this area in quadrants until the 32×32 grid is reached. Figure 1a, b shows this dataset.

It is worth noting that the number of sequences is not the crucial factor for the running time of our modeling methods, as the sequences are read only once at the beginning, and all the information is recorded in the frequency tables c_α and $c_{\alpha,\beta}$. So, the number of states is the variable that dominates the running time of the computation.

5.2 Experiment with known generative model

The purpose of this experiment is to compare for each database, the log likelihood obtained by the models that result from our search procedure, with the log likelihood obtained by the actual model used to generate the sequences. To make this comparison

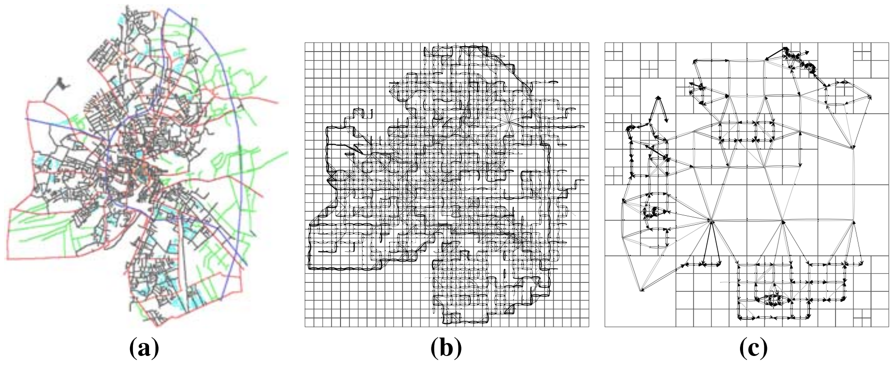

Fig. 1 **a** Road network used to generate the trajectories dataset. **b** Markov model of the whole dataset at the leaves level (i.e., the original grid with 1,024 states). **c** Markov model of the whole dataset with 175 states, found by TAXOMO. Arc thickness represents the probability of the transition

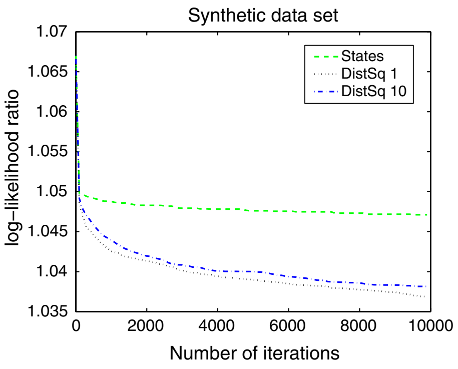

Fig. 2 Approximation error versus number of iterations

fair, we compared models having the same number of states, that is, if the generative model has m states, then we get the best m-states model found by our search procedure.

The results are shown in Fig. 2; we are presenting the average obtained after 10,000 iterations for each search strategy and each of the 100 databases. The strategy PROBABILITYFIRST is not included as it almost always outputs only models having a number of states larger than m. The other three strategies perform very well and very quickly they reach to less than 5% error, where the DISTSQFIRST and DISTSQFIRST- 10 strategies (for the values of the parameter w equal to 1 and 10, respectively) perform slightly better than STATESFIRST.

5.3 Search strategies comparison

We experiment with the three strategies described in Sect. 4: PROBABILITYFIRST, STATESFIRST, and DISTSQFIRST. For the last strategy we have two versions DISTSQFIRST-1 and DISTSQFIRST-10, for the values of the parameter w equal to 1 and 10, respectively.

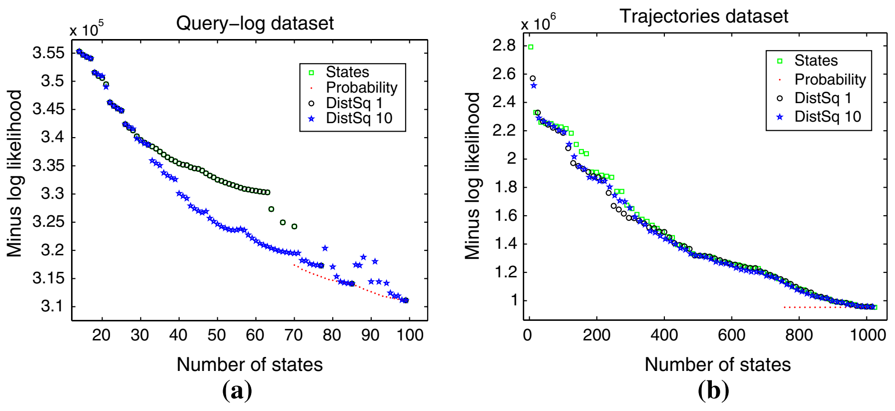

Fig. 3 Search space profile: probability of the best model found by the search algorithms, at each number of states, within 10,000 iterations

To compare the strategies, we assume a fixed budget of 10,000 model evaluations, and run the different search strategies. In a commodity 1.7GHz Intel-based PC with 1 GB of RAM under Linux, 10,000 models in the query dataset (100 states) are tested in 20 s, while in the trajectories dataset (1,024 states) the same number of models can be evaluated in 7–8 min.

After the search strategy is run, we select the model with the higher probability for each number of states. The result for both datasets is shown in Fig. 3a, b. For the query-log dataset, DISTSQFIRST-10 outperforms the other methods as it can generate models with the same number of states as the others, but having a higher data likelihood. For the trajectories dataset, all the methods are comparable with DISTSQFIRST-1 and DISTSQFIRST-10 winning by a small margin at different ranges of the number of states

The method PROBABILITYFIRST gives models with very good likelihood scores, especially for the trajectories dataset, but it reduces the number of states too slowly, so it does not even reach the appropriate exploration depth to be compared with the others within the same budget of model evaluations. This observation is evident by examining Fig. 4, in which we show where the search methods spend most of their exploration budget. Clearly, the methods DISTSQFIRST and STATESFIRST explore more models having less number of states. The running time depends on the number of states in the model being evaluated, so the method PROBABILITYFIRST also has the disadvantage of being slower, basically because of the candidate generation phase which generates one candidate for every node that can be merged from the current set of states. Thus, its applicability is limited to settings where a small but conservative reduction on the size of the model is desired.

To see the effect of increasing the budget of model evaluations we fix the search strategy to the winning DISTSQFIRST-10 and we plot the number of states vs. likelihood profile for three different budget values. This is shown in Fig. 5. For the query-log dataset we see that a larger budget allows the method to find a higher-probability model for the same number of states, particularly at intermediate values of the number

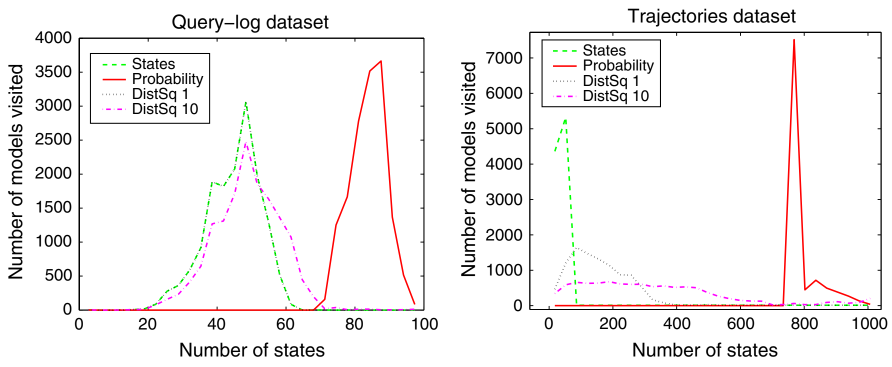

Fig. 4 Distribution of the number of states visited by the search algorithms

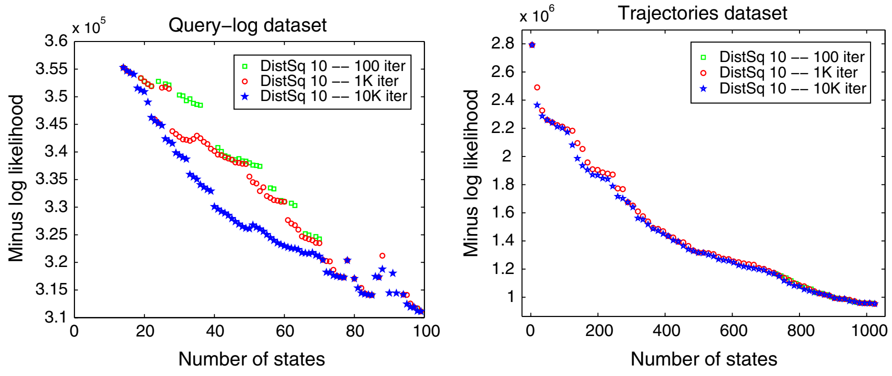

Fig. 5 Profile of search space explored by DistSqFirst-10 policy for different number of iterations

of states (neither too high nor too low), where the search space is correspondingly bigger. For the trajectories dataset, we see that the quality achieved stabilized very fast; the likelihood scores found for 100 model evaluations is not much worse than the likelihood scores found for 10,000 model evaluations.

Qualitatively, we observe that the pruning of the original taxonomy that is done by the model, is not uniform, but guided by the data. For instance, in the trajectories dataset, in the areas where traffic is more homogeneous the lumping of states is more aggressive than in other areas. In Fig. 1b, c we compare the Markov model obtained at the leaf level (1,024 states) with a model obtained after 10,000 iterations (175 states).

5.4 Clustering

Figure 6 shows the models of 4 clusters (from a total of 8) for the trajectories dataset. For the EM algorithm 50 iterations are performed, which is enough for convergence.

Each figure provides a direct visualization of the representative model of a cluster. An edge between two neighboring grid squares denotes that there is a high-probability transition between the squares, and edge thickness is proportional to the probability.

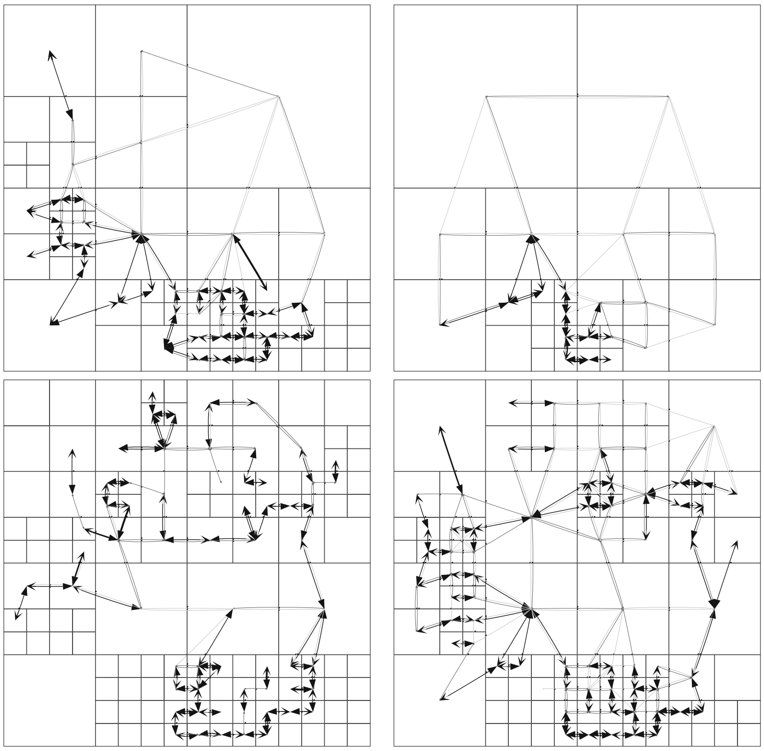

Fig. 6 Four example clusters from the trajectories dataset

Each cluster has its own model, and the flexibility of the modeling phase in terms of adaptively merging states is evident. This also highlights a nice visualization feature of our model: just laying the nodes of the Markov model directly over the geographic map, we can have an immediate and effective visual representation of a cluster.

6 Conclusions

We presented a scalable method for taxonomy-driven sequence mining based on Markov models. The method receives a database of sequences of symbols, and a taxonomy over those symbols. An initial Markov model is created for the sequences without considering the taxonomy, and then refine it iteratively by merging states driven by the taxonomy. The likelihood of the data given a new model generated by this merging procedure, can be computed directly from the likelihood of the data given the model before the merging. This yields a fast model evaluation method that can explore many configurations in a short time. We also implement efficient strategies that guide the search process.

There are many applications for this approach to sequence mining that we plan to explore in the future, in particular those related to modeling and clustering the actions of people, among others.

Appendix 1 Proofs

Proof (*of Lemma 2*) If we denote by S_1 the first sum in Eq. 1, using 2 we have

$$S_1 = -\sum_{\alpha,\beta\in\Sigma} c_{\alpha\beta} \log \frac{c_{\mathbf{x}(\alpha)\mathbf{x}(\beta)}}{c_{\mathbf{x}(\alpha)}} = -\sum_{\mathbf{x},\mathbf{y}\in X} \sum_{\substack{\alpha\in A(\mathbf{x})\\ \beta\in A(\mathbf{y})}} c_{\alpha\beta} \log \frac{c_{\mathbf{x}(\alpha)\mathbf{x}(\beta)}}{c_{\mathbf{x}(\alpha)}}$$
$$= -\sum_{\mathbf{x},\mathbf{y}\in X} \log \frac{c_{\mathbf{xy}}}{c_{\mathbf{x}}} \sum_{\substack{\alpha\in A(\mathbf{x})\\ \beta\in A(\mathbf{y})}} c_{\alpha\beta} = -\sum_{\mathbf{x},\mathbf{y}\in X} c_{\mathbf{xy}} \log \frac{c_{\mathbf{xy}}}{c_{\mathbf{x}}}$$

The second sum, due to the emission probabilities, is straightforward. □

Proof (*of Lemma 1*) First notice that $\mathbf{r}$ is a proper stochastic matrix defining the Markov model $\mathcal{M}'$. Indeed, for all $\alpha \in \Sigma$ we have

$$\sum_{\beta\in\Sigma} r_{\alpha,\beta} = \sum_{\beta\in\Sigma} p_{\mathbf{x}(\alpha),\mathbf{x}(\beta)} q_{\mathbf{x}(\beta),\beta} = \sum_{\mathbf{y}\in X} \sum_{\beta\in A(\mathbf{y})} p_{\mathbf{x}(\alpha),\mathbf{y}} q_{\mathbf{y},\beta} = \sum_{\mathbf{y}\in X} p_{\mathbf{x}(\alpha),\mathbf{y}} \sum_{\beta\in A(\mathbf{y})} q_{\mathbf{y},\beta}$$
$$= \sum_{\mathbf{y}\in X} p_{\mathbf{x}(\alpha),\mathbf{y}} \quad = 1.$$

Second, notice that any transition in $\mathcal{M}'$ can be simulated (and vice versa) by a transition and an emission in $\mathcal{M}$, with exactly the same probabilities. □

Proof (*of Lemma 4*) Let the two models $\mathcal{M}_1$ and $\mathcal{M}_2$ be defined on the set of states X_1 and X_2, respectively. We first assume that there is a set of states $Y \subseteq X_1$, so that $X_2 = X_1 \setminus Y \cup \{\mathbf{z}\}$. In other words, X_2 is obtained by a single merge of a subset Y of states of X_1 into a new state $\mathbf{z}$. The more general case of a relation between X_1 and X_2 can be handled by induction.

From Lemma 2, the models $\mathcal{M}_1 = (X_1, A_1, \mathbf{p}_1, \mathbf{q}_1)$ and $\mathcal{M}_2 = (X_2, A_2, \mathbf{p}_2, \mathbf{q}_2)$ that minimize the score functions $S_L^*(\mathcal{D} \mid \mathcal{M}_1)$ and $S_L^*(\mathcal{D} \mid \mathcal{M}_2)$ have transition and emission probabilities that are obtained by the observed frequencies, as in Eqs. 2 and 3.

Consider now the Markov models $\mathcal{M}_1' = (\Sigma, \bar{\mathbf{r}}_1)$ and $\mathcal{M}_2' = (\Sigma, \bar{\mathbf{r}}_2)$, as defined in Lemma 1, that is, $\mathcal{M}_1'$ and $\mathcal{M}_2'$ are Markov models on the set of symbols and they are equivalent to $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. The crucial observation is that $S_L^*(\mathcal{D} \mid \mathcal{M}_1)$ and $S_L^*(\mathcal{D} \mid \mathcal{M}_2)$ correspond to $H(\mathcal{M}_1')$ and $H(\mathcal{M}_2')$ (respectively), where $H(\mathcal{M})$ is the *entropy rate* of a Markov chain. For more details and the definition of the entropy rate of a Markov chain see (Cover and Thomas 1991, Chapter 4). Here we assume that the Markov chains are irreducible. The fact that our definition of a Markov chain contains starting and terminal states, can be handled by adding a transition of probability 1 from the terminal state to the starting state. Such a transition signifies the beginning of a new sequence in $\mathcal{D}$.

On the other hand, we can show that $H(\mathcal{M}_1') = H(\mathcal{M}_2' \mid \mathbf{Y})$, where $\mathbf{Y}$ is a random variable that denotes a transition in the set of states Y for $\mathcal{M}_1$ corresponding to a transition in $\mathbf{z}$ for $\mathcal{M}_2$. Intuitively, the additional information in $\mathcal{M}_1$ with respect to $\mathcal{M}_2$

can be recovered by conditioning on the outcome of the transitions in Y. Since *conditioning always decreases entropy*, we have $H(\mathcal{M}_2' \mid \mathbf{Y}) \leq H(\mathcal{M}_2')$, which proves our claim. □

Appendix 2 Incremental model evaluation

Let $\mathcal{M}_1$ be the optimal model having states X. Let $\mathcal{M}_2$ be a model built from by merging the states $Y \subseteq X$ into one new state $\mathbf{z}$. Let $|Y| = d$ and let $\overline{X} = X \setminus Y$. From Eq. 4, an optimal model $\mathcal{M}$ for a database of sequences $\mathcal{D}$ has:

$$S_L^*(\mathcal{D} \mid \mathcal{M}) = -\sum_{\mathbf{x},\mathbf{y} \in X} c_{\mathbf{xy}} \log \frac{c_{\mathbf{xy}}}{c_{\mathbf{x}}} - \sum_{\alpha \in \Sigma} c_\alpha \log \frac{c_\alpha}{c_{\mathbf{x}(\alpha)}},$$

the difference $\Delta = S_L^*(\mathcal{D} \mid \mathcal{M}_2) - S_L^*(\mathcal{D} \mid \mathcal{M}_1)$ is given by:

$$\begin{aligned}
\Delta = &\sum_{\mathbf{x} \in \overline{X}} \left(c_{\mathbf{xz}} \log \frac{c_{\mathbf{xz}}}{c_{\mathbf{x}}} + c_{\mathbf{zx}} \log \frac{c_{\mathbf{zx}}}{c_{\mathbf{z}}} \right) + c_{\mathbf{zz}} \log \frac{c_{\mathbf{zz}}}{c_{\mathbf{z}}} \\
&- \sum_{\mathbf{x} \in \overline{X}} \sum_{\mathbf{y} \in Y} \left(c_{\mathbf{xy}} \log \frac{c_{\mathbf{xy}}}{c_{\mathbf{x}}} + c_{\mathbf{yx}} \log \frac{c_{\mathbf{yx}}}{c_{\mathbf{y}}} \right) \\
&- \sum_{\mathbf{y}_1 \in Y} \sum_{\mathbf{y}_2 \in Y} \left(c_{\mathbf{y}_1\mathbf{y}_2} \log \frac{c_{\mathbf{y}_1\mathbf{y}_2}}{c_{\mathbf{y}_1}} \right) + \sum_{\alpha \in A(\mathbf{z})} c_\alpha \log \frac{c_{\mathbf{y}(\alpha)}}{c_{\mathbf{z}(\alpha)}},
\end{aligned}$$

The last term only involves the emission probabilities of the symbols under the subtree of the merged state $\mathbf{z}$. The value of Δ can be computed in time $O(\min\{n, md\})$. The number of states merged d is bounded by the maximum degree in the taxonomy tree.

References

Bicego M, Dovier A, Murino V (2001) Designing the minimal structure of hidden Markov model by bisimulation. Energy Minimization Methods Comput Vis Pattern Recognit 2001:75–90

Bicego M, Murino V, Figueiredo M (2003) Similarity-based clustering of sequences using hidden Markov models. Mach Learn Data Min Pattern Recognit 2003:95–104

Borges J, Levene M (2004) A dynamic clustering-based Markov model for web usage mining. arxiv:cs/0406032

Brinkhoff T (2003) Generating traffic data. IEEE Data Eng Bull 26(2):19–25

Cakmak A, Özsoyoglu G (2008) Taxonomy-superimposed graph mining. In: Proceedings of 11th international conference on Extending Database Technology (EDBT)

Cao H, Jiang D, Pei J, He Q, Liao Z, Chen E, Li H (2008) Context-aware query suggestion by mining click-through and session data. In: Proceeding of the 14th ACM SIGKDD international conference on Knowledge Discovery and Data Mining (KDD)

Cover TM, Thomas JA (1991) Elements of information theory. Wiley-Interscience

Felzenszwalb PF, Huttenlocher DP, Kleinberg JM (2004) Fast algorithms for large-state-space hmms with applications to web usage analysis. In: Advances in Neural Information Processing Systems (NIPS)

Girolami M, Kaban A (2003) Simplicial mixtures of Markov chains: distributed modelling of dynamic user profiles. In: Advances in Neural Information Processing Systems (NIPS)

Guralnik V, Karypis G (2001) A scalable algorithm for clustering protein sequences. In: BIOKDD
Kemeny J, Snell JL (1959) Finite Markov chains. Springer-Verlag
Law MH, Kwok JT (2000) Rival penalized competitive learning for model-based sequence clustering. Pattern Recognition, International Conference on 2
Lee HK, Kim JH (1999) An hmm-based threshold model approach for gesture recognition. IEEE Trans Pattern Anal Mach Intell 21(10):961–973
Lee JG, Han J, Li X, Gonzalez H (2008) *raClass*: trajectory classification using hierarchical region-based and trajectory-based clustering. In: Proceedings of the 34th international conference on Very Large Databases (VLDB)
Lee JG, Han J, Whang KY (2007) Trajectory clustering: a partition-and-group framework. In: Proceedings of the 2007 ACM SIGMOD international conference on management of data (SIGMOD)
Li X, Han J, Lee JG, Gonzalez H (2007) Traffic density-based discovery of hot routes in road networks. In: Proceedings of the 10th international Symposium on Advances in Spatial and Temporal Databases (SSTD)
Manavoglu E, Pavlov D, Giles CL (2003) Probabilistic user behavior models. In: Proceedings of 3rd IEEE International Conference on Data Mining (ICDM)
Manning AM, Brass A, Goble CA, Keane JA (1997) Clustering techniques in biological sequence analysis. In: PKDD
Meyer CD (1989) Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems. SIAM Rev 31(2):240–272
Nanni M, Pedreschi D (2006) Time-focused clustering of trajectories of moving objects. J Intell Inf Syst 27(3):267–289
Schwarz G (1978) Estimating the dimension of a model. Ann Stat 6(2)
Simon H, Ando J (1961) Aggregation of variables in dynamic systems. Econometrica 29:111–138
Srikant R, Agrawal R (1995) Mining generalized association rules. In Proceedings of 21th international conference on Very Large Data Bases (VLDB)
Srikant R, Agrawal R (1996) Mining sequential patterns: generalizations and performance improvements. In: Proceedings of 5th international conference on Extending Database Technology (EDBT)
Smyth P (1997) Clustering sequences with hidden Markov models. In: Advances in neural information processing systems, vol 9, pp 648–654
Stolcke A, Omohundro SM (1994) Best-first model merging for hidden Markov model induction
Tijms H (1986) Stochastic modelling and analysis: a computational approach. Wiley, New York
Wang J, Zhang Y, Zhou L, Karypis G, Aggarwal CC (2007) Discriminating subsequence discovery for sequence clustering. In: SDM
Welch LR (2003) Hidden Markov models and the baum-welch algorithm. IEEE Inf Theory Soc Newsl 53(4)
White LB, Mahony R, Brushe GD (2000) Lumpable hidden Markov models - model reduction and reduced complexity filtering. IEEE Trans Automat Contr 45(12)