Interactive HMM construction based on interesting sequences

Szymon Jaroszewicz

National Institute of Telecommunications
Warsaw, Poland
s.jaroszewicz@itl.waw.pl

Abstract. The paper presents a method of interactive construction of global Hidden Markov Models based on local patterns discovered in sequence data. The method works by finding *interesting* sequences whose probability in data differs from that predicted by the model. The patterns are then presented to the user who updates the model using their understanding of the modelled domain. It is argued that such an approach leads to more understandable models than automated approaches. An application to modelling webpage visitors behavior is presented, showing the strengths of the proposed approach.

1 Introduction

The presented method is based on previous work by Jaroszewicz, Scheffer and Simovici [6, 4, 5] on interactive construction of Bayesian networks. The overall approach is briefly described below.

The user provides a (possibly empty) description of their current knowledge about the domain and a dataset from which new knowledge is to be discovered. Knowledge description has the form of a *global* probabilistic model from which precise probabilistic inference is possible. In [6, 4, 5] a Bayesian network was used for this purpose due to its flexibility, understandability, as well as the fact that it represents a full joint probability distribution making inference possible.

Based on the model and data interesting patterns are discovered. A pattern is defined to be interesting if its probability in data differs significantly from that predicted by the global model. The patterns are then presented to the user whose task is to interpret them and update the model. The new model is then used again together with the data to find a new set of interesting patterns. In [5] it has been demonstrated that interactive model construction, using human intelligence in the process, gives models which represent the domain much better than models built automatically.

In this paper the approach is extended to models involving time, namely to Hidden Markov Models (HMMs) [9, 12]. The approach works by finding sequences whose frequency in the database differs significantly from what the HMM predicts. Such patterns are considered interesting and are shown to the user who updates the HMM usually by adding more hidden states, representing

new underlying behavior. The HMM parameters are then retrained using the Expectation Maximization (EM) algorithm and the process is repeated.

The advantage of such an approach is that the new states have clear, user given interpretation. The resulting model is thus understandable, and all hidden states have a clear meaning. This is not possible with automatic methods.

The approach has been tested on the web log of the National Institute of Telecommunications in Warsaw. The application proved that the proposed approach is highly practical and produces accurate models which are understandable and easy to interpret.

There has been a significant amount of work on mining patterns in sequence data, full discussion is beyond the scope of this work, see for example an overview in [8] or in [3].

Section 5 of [8] describes approaches to testing significance of sequence patterns based on comparing with a background model. The purpose however is to test statistical significance and the models are thus simple (e.g. based on independence assumption) and fixed throughout the discovery process, user's background knowledge does not come into the picture as it does in our approach. In [7] a separate small HMM with special structure is built for each temporal pattern (episode). Such small HMMs are later used for significance testing. The use of a global model being a mixture of small episode models is alluded to, but not developed further. Zaiane et al. [13, 10] work on discovering contrast sequences. This is similar to a single stage of the interactive model building process described here, except that one of the datasets is replaced by a HMM. The overall methodology and the discovery algorithms are thus very different.

2 Definitions and Notation

Vectors are denoted by lowercase boldface letters, and matrices with boldface uppercase letters. All vectors are assumed to be row vectors, explicit transposition ^{T} is used for column vectors.

Superscripts will be used to denote time and subscripts to denote element numbers, e.g. $\boldsymbol{\pi}_i^t$ denotes i-th element of the vector $\boldsymbol{\pi}$ at time t. Since matrices will not change with time in our applications, superscripts on matrices will denote matrix power, e.g. $(\mathbf{P})^t$ is the matrix $\mathbf{P}$ raised to the power t, parentheses around the matrix are added to avoid confusion.

2.1 Hidden Markov Models

Let us now define Hidden Markov Models which will be used to describe systems with discrete time. Only most important facts will be given, full details can be found in literature [9, 12].

A Hidden Markov Model is a modification of a Markov Chain, such that the state the model is in is not directly visible. However every state emits output symbols according to its own probability distribution. For example while modelling webpage visitors' behavior, the internal states could correspond to visitor's

intentions (wants to find some specific content) and output symbols to the pages he/she actually visits.

More formally, an HMM is a quintuple $\langle S, O, \boldsymbol{\pi}^0, \mathbf{P}, \mathbf{E}\rangle$ where $S = \{s_1, \ldots, s_n\}$ is a set of *states*, $O = \{o_1, \ldots, o_m\}$ the set of output symbols, $\boldsymbol{\pi}^0 = (\boldsymbol{\pi}^0_1, \ldots, \boldsymbol{\pi}^0_n)$ the vector of initial probabilities for each state, and $\mathbf{P}$ is the *transition matrix*, where $\mathbf{P}_{ij}$ is the probability of transition from state s_i to state s_j. Finally $\mathbf{E}$ is the *emission probability matrix* such that $\mathbf{E}_{ij}$ gives the probability of observing symbol o_j provided that the HMM is in state s_i.

Notice that the vector $\boldsymbol{\pi}^t$ of state probabilities at time t is $\boldsymbol{\pi}^t = \boldsymbol{\pi}^0(\mathbf{P})^t$. Similarly $\boldsymbol{\pi}\mathbf{E}$ is the vector of probabilities of observing each symbol provided that $\boldsymbol{\pi}$ is the vector of state probabilities.

We will now discuss how to determine the probability that a given sequence of output symbols is produced by an HMM. For that aim a key concept of forward probabilities will be introduced, full details can be found in [9, 12].

Let $\mathcal{O}^t = o^0, o^1, \ldots, o^t$ be a sequence of symbols, and let q_t denote the state of the HMM at time t. The *forward probability* of $\mathcal{O}^t$ is defined as

$$\alpha(\mathcal{O}^t, i) = \Pr\{o^0, \ldots, o^t, q_t = s_i\},$$

that is the probability that we have observed the sequence $o^0, \ldots, o^t$ and the HMM ended up in state s_i at time t. Grouping the probabilities for all states we obtain a vector

$$\boldsymbol{\alpha}(\mathcal{O}^t) = (\alpha(\mathcal{O}^t, 1), \alpha(\mathcal{O}^t, 2), \ldots, \alpha(\mathcal{O}^t, n)).$$

The probability of observing a sequence $\mathcal{O}^t$ can be computed by summing the elements of $\boldsymbol{\alpha}(\mathcal{O}^t)$.

An important property of the α probabilities is that they can be efficiently computed using dynamic programming by extending the sequence, symbol by symbol, using the following formula:

$$\alpha(o^0, \ldots, o^t, o^{t+1}, i) = \boldsymbol{\alpha}(o^0, \ldots, o^t)\mathbf{P}_{\cdot,i}\mathbf{E}_{i,o^{t+1}}. \tag{1}$$

Another important problem related to HMMs is estimating transition probabilities based on given sequence data. This is usually done using the Baum-Welch algorithm which is an Expectation Maximization algorithm. The details can be found in [9, 12] and will be omitted here.

The algorithm only guarantees convergence to a local minimum and has been reported to be slow. In practice we have noticed the algorithm is very dependent on the emission probabilities $\mathbf{E}$. If output symbols convey a reasonable amount of information about internal states, the algorithm converged quickly and reliably. We consider this to be the practically most useful case; unless the output symbols provide enough information on the internal state, one cannot expect to properly identify the underlying behavior.

2.2 Discovery of interesting sequences

The key component of the interactive global HMM model construction using the proposed framework is finding interesting sequences of output symbols. There

are several ways to formulate the problem. Here we will present one of them, assuming that provided data contains many sequences each starting at time $t = 0$. This approach is suitable for web log analysis which will be presented in the experimental section.

Let the provided dataset D contain N symbol sequences, each starting at $t = 0$,

$$D = \{\mathcal{O}_1, \ldots, \mathcal{O}_N\},$$

where $\mathcal{O}_j = o^0, o^1, \ldots, o^{t_j}$.

Let $\mathcal{O} = o^0, o^1, \ldots, o^t$ be a sequence of symbols starting at time $t = 0$. Denote by $\Pr^{HMM}\{\mathcal{O}\}$ the probability of observing the sequence $\mathcal{O}$ computed from the HMM. Analogously the probability of observing that sequence in data is denoted by

$$\Pr^{D}\{\mathcal{O}\} = \frac{|\{\mathcal{O}' \in D : \mathcal{O} \text{ is a prefix of } \mathcal{O}'\}|}{|D|}.$$

A sequence whose probability of occurrence is greater than or equal to ε is called *ε-frequent*.

The *interestingness* of $\mathcal{O}$ is defined (analogously to [6]) as

$$\mathcal{I}(\mathcal{O}) = \left|\Pr^{D}\{\mathcal{O}\} - \Pr^{HMM}\{\mathcal{O}\}\right|, \tag{2}$$

that is, as the absolute difference between the probabilities predicted by the HMM and observed in data. A sequence is called ε-interesting if its interestingness is not lower than ε.

An algorithm for finding all ε-interesting symbol sequences for a user specified minimum interestingness threshold ε is given in Figure 1. Key steps of the algorithm are described in detail below.

2.3 Finding frequent sequences in data

There are several algorithms for finding frequent sequences [3]. The situation presented here is much simpler however, since all sequences are assumed to start at $t = 0$, so a simpler approach is used.

First, all sequences in D are sorted in lexicographical order. Then, as the records are scanned sequentially, each record is compared to the previous one and their longest common prefix p is found. Counts of all prefixes of p are incremented by 1. The prefixes of the previous record which are longer than p will never appear again, so those whose support is lower than ε are removed.

2.4 Finding frequently occurring sequences in the Hidden Markov Model

This part has been implemented in the style of a depth first version of the well known Apriori algorithm [1]. We use the fact that appending additional symbols to a sequence cannot increase its probability of occurrence, so if a sequence is found to be infrequent, all sequences derived from it can be skipped. The

Input: A Hidden Markov Model HMM, dataset of symbol sequences D, the minimum interestingness threshold ε
Output: A set of interesting sequences: $\{\mathcal{O} : \mathcal{I}(\mathcal{O}) \geq \varepsilon\}$

1. Estimate the parameters $\boldsymbol{\pi}^0$, $\mathbf{P}$, $\mathbf{E}$ of HMM based on D using the Baum-Welch algorithm
2. Find the set $\mathcal{C}_D$ of sequences which are ε-frequent in D

$$\mathcal{C}_D = \{\mathcal{O} : \Pr^D\{\mathcal{O}\} \geq \varepsilon\}$$

3. Find the set $\mathcal{C}_{HMM}$ of sequences which are ε-frequent in the HMM

$$\mathcal{C}_{HMM} = \{\mathcal{O} : \Pr^{HMM}\{\mathcal{O}\} \geq \varepsilon\}$$

4. Compute $\Pr^D\{\mathcal{O}\}$ for each $\mathcal{O} \in \mathcal{C}_{HMM} \setminus \mathcal{C}_D$
5. Compute $\Pr^{HMM}\{\mathcal{O}\}$ for each $\mathcal{O} \in \mathcal{C}_D \setminus \mathcal{C}_{HMM}$
6. Compute $\mathcal{I}(\mathcal{O})$ for all sequences $\mathcal{O} \in \mathcal{C}_D \cup \mathcal{C}_{HMM}$
7. Return the set $\{\mathcal{O} \in \mathcal{C}_D \cup \mathcal{C}_{HMM} : \mathcal{I}(\mathcal{O}) \geq \varepsilon\}$

Fig. 1. The algorithm for finding all ε-interesting sequences with respect to a given Hidden Markov Model.

sequences are built symbol by symbol and their probability in the HMM is simultaneously updated using Equation 1. The algorithm is shown in Figure 2. The + symbol denotes sequence concatenation. In fact the presented algorithm is very efficient since computing probabilities in HMMs can be done much more efficiently than computing supports in large datasets.

3 Experimental Evaluation, Mining Webserver Logs

The presented approach will now be evaluated experimentally on a web server log data. Web log of the server of the National Institute of Telecommunications in Warsaw has been used. The full data covered the period of about one year, the first 100000 events have been used for experiments. The presented method has been used to create a model of behavior of visitors to the Institute's webpage.

After starting with a simple initial model, new internal states were added with the assumption that the new states represent underlying patterns of user behavior such as 'access to e-mail account through web interface' or 'access a paper in Institute's journal'. The identification of the underlying behavior and model updating was done entirely by the user.

3.1 Data preprocessing

Each record in a weblog contains a description of a single event, such as a file being retrieved. Several items such as date and time of the event, the referenced file, an error code, etc. are recorded. The expected data format is a database

Input: Hidden Markov Model HMM, minimum support threshold ε
Output: The set of ε-frequent sequences $\{\mathcal{O} : \Pr^{HMM}\{\mathcal{O}\} \geq \varepsilon\}$

1. **Call** FreqHMM($\emptyset$, $(1, 1, \ldots, 1)$)
2. Return the result set.

3. **Function** FreqHMM($\mathcal{O}$, $\boldsymbol{\alpha}(\mathcal{O})$):
4. **If** $\Pr^{HMM}\{\mathcal{O}\} = \sum_i \alpha(\mathcal{O}, i) \geq \varepsilon$:
5. Add $\mathcal{O}$ to the result set
6. For every output symbol o:
7. Compute $\boldsymbol{\alpha}(\mathcal{O} + o)$ using Equation 1
8. **Call** FreqHMM($\mathcal{O} + o$, $\boldsymbol{\alpha}(\mathcal{O} + o)$)

Fig. 2. An algorithm for finding all ε-frequent sequences in a Hidden Markov Model.

of sequences (each sequence corresponding to a single user session) of pages visited by users. Unfortunately the log does not contain explicit information about sessions. The data had to be *sessionized* in order to form training sequences. There are several sessionizing methods [2, 11]. Here, the simplest method was used, namely events which were less than 30 minutes apart were considered to belong to a single session. Despite its simplicity the method worked very well. Other methods, e.g. involving a limit on total session time were problematic, e.g. could not handle automated web crawlers' sessions which were very long.

At the end of each session an artificial `QUIT` symbol has been added such that an end of a session could be explicitly modelled.

Another choice that had to be made was the set of output symbols. Since the server contains a very large number of available files assigning a separate symbol to every file would have introduced a huge number of output symbols, adversely affecting the understandability of the model, as the user is often more interested in the behavior of users at a higher level. To solve this problem only top-level directory of each accessed file was used as output symbol. As the Institutes webpage is logically divided into subdirectories such as *journal*, *people* etc. such an approach gives a better overview of users' behavior than specific files. If finer level analysis is required it is probably better to build a separate model which covers only a subset of available pages in greater detail.

3.2 An initial model

As the author had no idea on how an initial model should look like, an empty model given in Figure 3 was used.

The `_all_` state can initially emit any of the symbols present in the log. The `quit` state can emit only the artificial `QUIT` symbol. The model corresponds to a user randomly navigating from page to page.

As more states are added, emitted symbols will be removed from the `_all_` state. This will allow for better identification of the internal state of the model

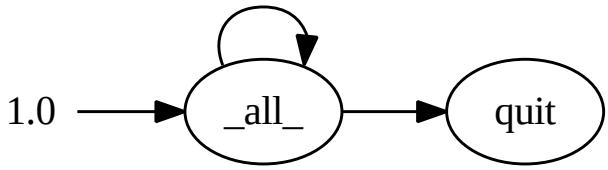

Fig. 3. An initial HMM describing the behavior of webpage visitors.

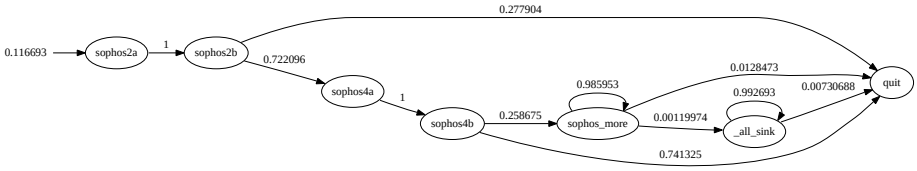

Fig. 4. The fragment of the HMM model describing connections to the Sophos antivirus updates.

based on the output symbol, leading to better understandability as well as better and more efficient parameter estimation using the Baum-Welch algorithm.

3.3 Model construction

We will now describe the most interesting stages of model construction.

The first interesting sequences were related to the Sophos antivirus program whose updates are available on the intranet. The most interesting sequence was `sophos`,`sophos`; it's probability in data was 11.48% while the initial model predicted it to be only 1.17%. The second most interesting sequence was one in which the `sophos` directory has been accessed four times. It is interesting that every session contained either two or four or more accesses to this directory; for over a year there has not been a single session where the directory would have been accessed only once or only three times.

In order to correctly predict the probabilities of those sequences the model has been updated by adding new states shown in Figure 4. Each of the new states emits only the `sophos` symbol. In addition that symbol has been removed from the list of symbols emitted by the `_all_` state.

The `_all_sink` state is used to model sessions in which, after some initial sequence, arbitrary pages can be visited. Here it is used to model sessions where after downloading antivirus updates the visitor moves to another part of the website.

After the new states have been added, the probabilities of related sequences were predicted accurately, and new sequences became interesting. The most interesting sequences for the updated model were about the directory `journal`

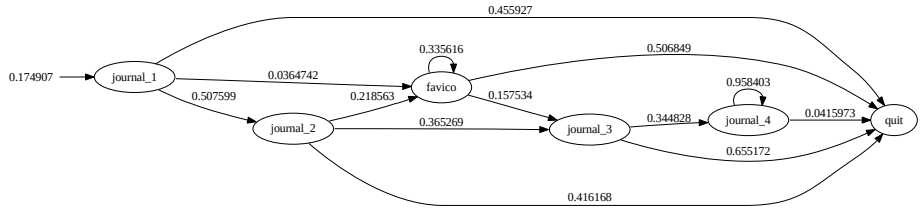

Fig. 5. The fragment of the HMM model describing visitors accessing journal articles.

containing the articles published in the Institute's journal in PDF format. Almost 2% of all sessions contained the sequence `journal`,`journal`,`/favicon.ico`, which according to the model should appear extremely infrequently. In addition the `/favicon.ico` file was absent and generated an error.

It turned out that the file `/favicon.ico` is the default location for a small icon appearing in the web-browser next to webpage address. On the Institutes website this file was however located in `img/favicon.ico` which was marked in the header of all HTML files. PDF files however did not contain this information which caused the browser to try the default location, and since the file was not there trigger an HTTP error.

To model sessions accessing journal pages several states have been added as shown in Figure 5.

It is interesting that very often the command to access PDF files has been issued twice in a row. After inspecting the log, it turned out that the transmission of the same file has often been restarted. The author was not able to explain this fact.

A large proportion (about 10%) of the sessions was initiated by automated web crawlers. Such sessions can be easily recognized, as they first access the `/robots.txt` file describing the site's policy towards such visitors. A further 5% of all sessions were initiated by RSS news readers, primarily Google reader.

Several other states have been added to the model in the above fashion, e.g. states relating to web interface to e-mail or pages related to conferences organized at the Institute. We omit the details. The final model is shown in Figure 6. Despite a fairly large number of states the model is perfectly understandable.

The final model predicts the probability of *all possible* input sequences with accuracy better than 0.01. We can thus say that either a sequence is modelled well, or it appears very infrequently and is thus of little importance to the overall picture of visitors' behavior.

Let us now compare the final model with models built automatically. An arbitrary number of twenty hidden states of an HMM has been picked and the Baum-Welch algorithm has been used to compute the transition and emission probabilities. Figure 7 shows the resulting HMM with only edges corresponding to transition probabilities greater than 0.01 shown, and Figure 8 the same automaton with edges corresponding to transition probabilities greater than 0.001.

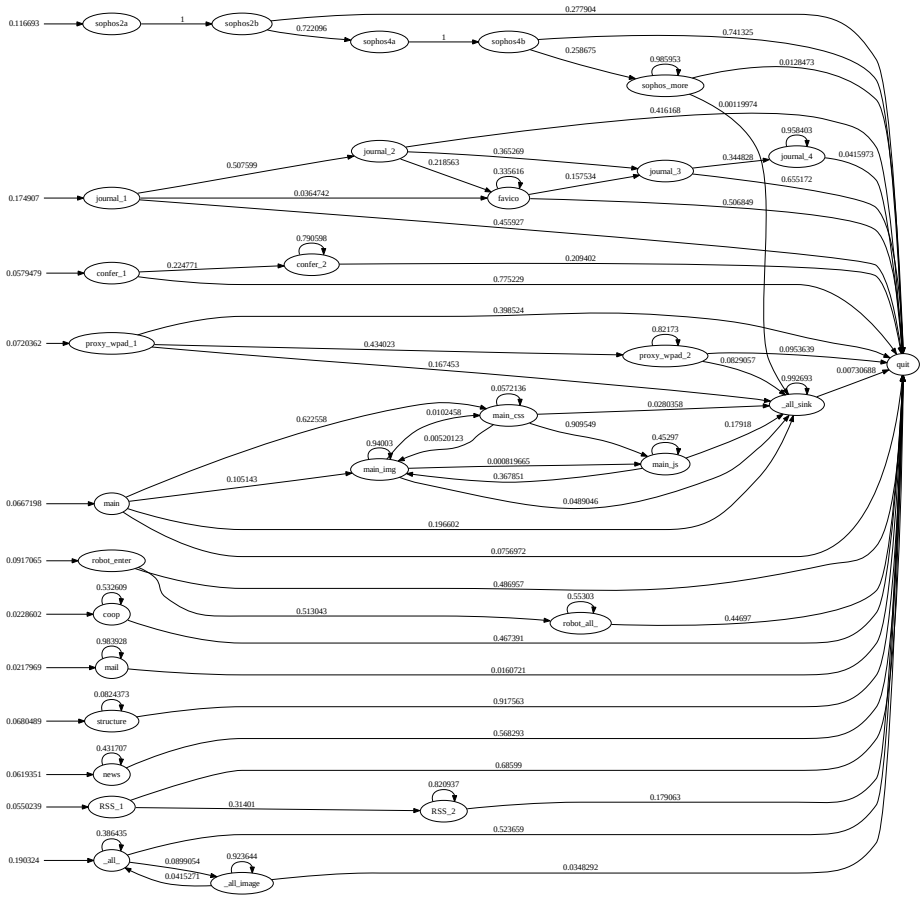

Fig. 6. The final HMM model describing website visitor's behavior

Each node contains the symbols emitted by the corresponding state (only symbols with emission probabilities greater than 0.01 are shown for clarity). Looking at Figure 7 it can be seen that the model's structure is not a good description of users' behavior. In the upper part of the picture is a connected group of nodes related to the web e-mail interface, but they are not connected to the rest of the graph. Moreover, nodes related to e-mail are also present in the large cluster of nodes below. The Sophos antivirus nodes are present and the discovered patterns could be inferred from their transition probabilities, albeit with some effort. Other discovered patterns are not clearly visible and it would be very hard to infer them from transition probabilities.

Adding more edges to the picture, as seen in Figure 8, does not help. On the contrary it makes the automaton practically incomprehensible.

Let us briefly comment on the efficiency of the proposed approach. Since the special case of sequences starting at $t = 0$ is considered, frequent sequence mining

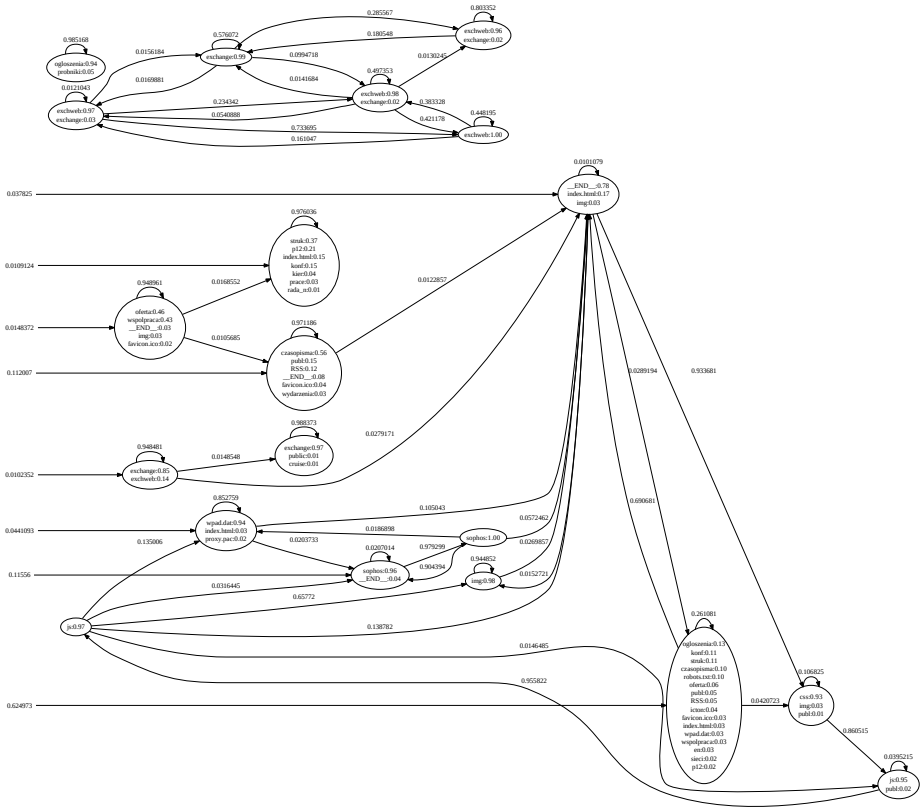

Fig. 7. Automatically build HMM for the weblog example. Only edges corresponding to probabilites > 0.01 are shown.

in data and in HMM is very fast. The computation time is almost completely dominated by the Baum-Welch algorithm. The algorithm can be quite slow, but we discovered that when the emission probability matrix **E** provides enough information about the internal state of the model given an output symbol, the algorithm requires few iterations. This is the case in the presented application, where many states may emit only a single symbol. The model shown in Figure 6 converged in just five iterations. The computation took about 6 minutes on a dataset of 100000 log events using a simple Python implementation.

It should also be noted that the Baum-Welch algorithm on the hand-built model converged much faster than in the case of an automatically built model. The automatic case usually required about a hundred iterations. The reason for that is that in the hand-built model the emission probability matrix conveyed a significant amount of information about the internal state of the model, which was not possible in case of the automatic approach.

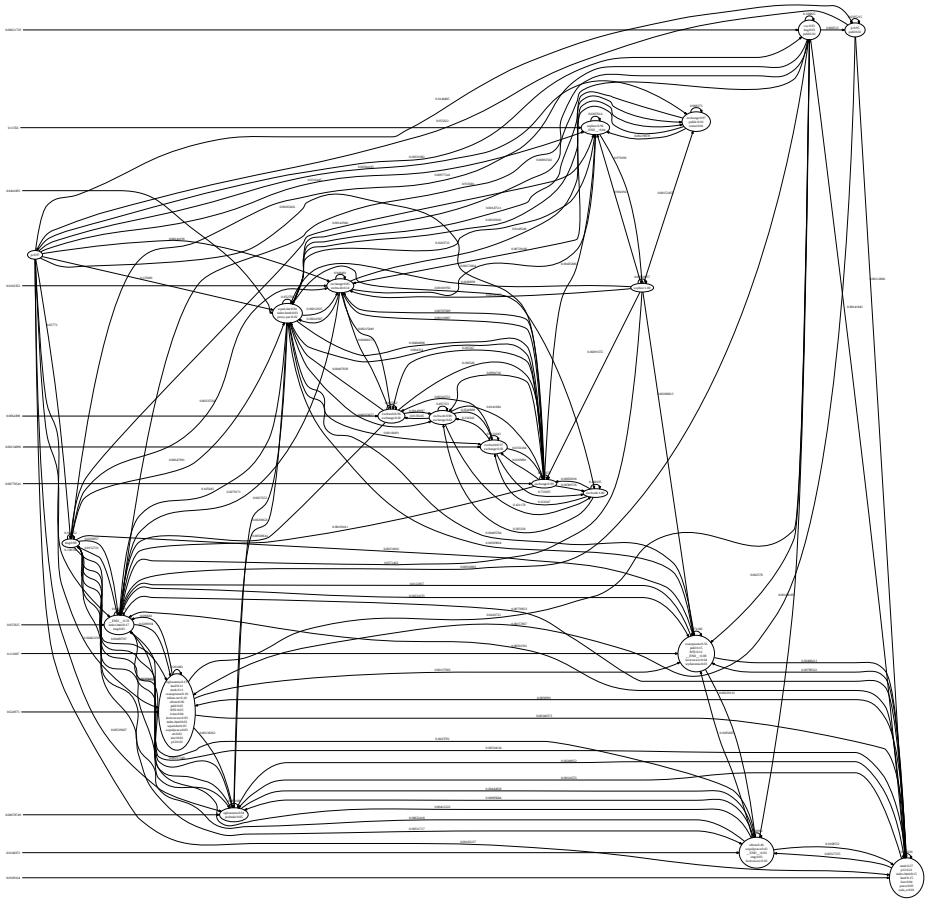

Fig. 8. Automatically build HMM for the weblog example. Edges corresponding to probabilites > 0.001 are shown.

4 Conclusions and future research

An approach to interactive construction of global HMM models based on local patterns has been presented. The approach is shown experimentally to produce accurate but easy to understand models. Future work will focus on generalizing the approach to other types of patterns such as episodes or motifs, and other types of models such as dynamic Bayesian networks.

Currently all updates to the model have to be done manually by the user. Although the author believes that the interactive approach has important advantages, an approach were the HMM is automatically updated based on the most interesting pattern may also be considered. The model could for example be a mixture of Episode Generating HMMs as suggested in [7].

The use of other interestingness measures such as Kullback-Leibler divergence will also be investigated.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Conf. on Management of Data*, pages 207–216, 1993.
2. M. H. Dunham. Data mining, introductory and advanced topics, part III. `http://engr.smu.edu/~mhd/dmbook/part3.ppt`.
3. J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Mining Knowledge Discovery*, 15(1):55–86, 2007.
4. S. Jaroszewicz and T. Scheffer. Fast discovery of unexpected patterns in data, relative to a Bayesian network. In *11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2005)*, pages 118–127, Chicago, IL, August 2005.
5. S. Jaroszewicz, T. Scheffer, and D.A. Simovici. Scalable pattern mining with Bayesian networks as background knowledge. *to appear in Data Mining and Knowledge Discovery*.
6. S. Jaroszewicz and D.A. Simovici. Interestingness of frequent itemsets using Bayesian networks as background knowledge. In *10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004)*, pages 178–186, Seattle, WA, August 2004.
7. S. Laxman and P.S. Sastry. Discovering frequent episodes and learning Hidden Markov Models: A formal connection. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1505–1517, 2005.
8. S. Laxman and P.S. Sastry. A survey of temporal data mining. *Sadhana*, 31(2):173–198, 2006.
9. L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.
10. A. Satsangi and O.R. Zaiane. Contrasting the contrast sets: An alternative approach. In *Eleventh International Database Engineering and Applications Symposium (IDEAS 2007)*, Banff, Canada, September 2007.
11. D.J. Smith and J.E. Pricer. Sessionizing clickstream data. Teradata Magazine Online, `http://www.teradata.com/t/page/116280/index.html`.
12. L.R. Welch. Hidden Markov Models and the Baum-Welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4):1,10–13, December 2003.
13. O.R. Zaiane, K. Yacef, and J. Kay. Finding top-n emerging sequences to contrast sequence sets. Technical Report TR07-03, Department of Computing Science, University of Alberta, Edmonton, AB, Canada, 2007.