Skeletal Algorithms: Sequential Pattern Mining

Michal R. Przybylek

***Abstract*—The basic idea behind the skeletal algorithm is to express a problem in terms of congruences on a structure, build an initial set of congruences, and improve it by taking limited unions/intersections, until a suitable condition is reached. Skeletal algorithms naturally arise in the context of data/process mining, where the skeleton is the "free" structure on initial data and congruence corresponds to similarities in data. In this paper we study skeletal algorithms applied to sequential pattern mining and compare their performance with real models, Markov chains and models based on Shannon entropy.**

***Index Terms*—Evolutionary algorithms, pattern mining, process mining, language recognition, skeletal algorithms.**

I. Introduction

"Nowadays, there is no longer any question that the quality of a company's business processes has a crucial impact on its sales and profits. The degree of innovation built into these business processes, as well as their flexibility and efficiency, are critically important for the success of the company. The importance of business processes is further revealed when their are considered as the link between business and IT; business applications only become business solutions when the processes are supported efficiently. The essential task of any standard business software is and always will be to provide efficient support of internal and external company processes."— Torsten Scholz

In order to survive in today's global economy more and more enterprises have to redesign their business processes. The competitive market creates the demand for high quality services at lower costs and with shorter cycle times. In such an environment business processes must be identified, described, understood and analysed to find inefficiencies which cause financial losses.

One way to achieve this is by modelling. Business modelling is the first step towards defining a software system. It enables the company to look afresh at how to improve organization and to discover the processes that can be solved automatically by software that will support the business. However, as it often happens, such a developed model corresponds more to how people think of the processes and how they wish the processes would look like, then to the real processes as they take place.

Another way is by extracting information from a set of events gathered during executions of a process. Process mining [1]-[14] is a growing technology in the context of business process analysis. It aims at extracting this information and using it to build a model. Process mining is also useful to check if the "a priori model" reflects the actual situation of executions of the processes. In either case, the extracted knowledge about business processes may be used to reorganize the processes to reduce they time and cost for the enterprise.

Manuscript received February 25, 2014; revised April 26, 2014. This work has been partially supported by Polish National Science Center, project~DEC-2011/01/N/ST6/02752.

Michal R. Przybylek is with the University of Warsaw, Poland (e-mail: mrp@mimuw.edu.pl).

TABLE I: An Event Log

Case	Observable Action	Actor	Timestamp	Data
127	START	Dr. Moor	11:30:52 07.02.2011	
127	Listen to patient's complaints	Dr. Moor	11:34:27 07.02.2011	headache
127	Listen to patient's complaints	Dr. Moor	11:35:59 07.02.2011	fever
107	START	Dr. No	11:36:50 07.02.2011	
127	Listen to patient's complaints	Dr. Moor	11:39:33 07.02.2011	catarrh
107	Listen to patient's complaints	Dr. No	11:39:37 07.02.2011	pain in the left foot
127	Select a candidate disease	Dr. Moor	11:58:30 07.02.2011	angina
127	Query patient about symptoms	Dr. Moor	12:01:11 07.02.2011	sore throat? — yes
127	Query patient about symptoms	Dr. Moor	12:08:21 07.02.2011	white patches on the tonsils? — yes
107	Select a candidate disease	Dr. No	12:10:31 07.02.2011	broken leg
107	Query patient about symptoms	Dr. No	12:11:01 07.02.2011	swollen leg? — No
107	Select a candidate disease	Dr. No	12:11:33 07.02.2011	joint dislocation
107	Query patient about symptoms	Dr. No	12:14:00 07.02.2011	blood inflammation? — Yes
107	Make a diagnosis	Dr. No	12:16:02 07.02.2011	joint dislocation
107	END	Dr. No	12:16:50 07.02.2011	
127	Make a diagnosis	Dr. Moor	12:34:01 07.02.2011	angina
127	END	Dr. Moor	12:34:55 07.02.2011	
...	...	...	...	...

Table I shows a typical event-log gathered during executions of the process to determine and identify a possible disease or disorder of a patient. In this paper, we assume that with every such an event-log there are associated:

1) An identifier referring to the execution (the case) of the process that generated the event.
2) A unique timestamp indicating the particular moment when the event occurred.
3) An observable action of the event; we shall assume, that we are given only some rough information about the real actions.

and we shall forget about any additional information and attributes associated with an execution of a process. The first property says that we may divide a list of events on collections of events corresponding to executions of the process, and the second property let us linearly order each of

DOI: 10.7763/IJCTE.2015.V7.944

the collections. If we use only information about the relative occurrences of two events (that is: which of the events was first, and which was second), then the log may be equally described as a finite list of finite sequences over the set *Observable Action* of all possible observable actions. Therefore we may think of the log as a finite sample of a probabilistic language over alphabet *Observable Action*, or more accurately, as the image of a finite sample of a probabilistic language over *Action*, under a morphis $h : Action \to Observable\ Action$ Morphism *h* describes our imperfect information about the real actions. In the example from Table I (here we use the first letter of the name of an action as abbreviate for the action)

$$Observable\ Action = \{l, s, q, m\}$$

and the sample contains sequences

$$S = \{<l, l, l, s, q, q, m>, <l, s, q, s, q, m>\}$$

Fig. 1 shows a model recognized from this sample. Here *Action* = *Observable Action* and *h* is the identity morphism (there are no duplicated events).

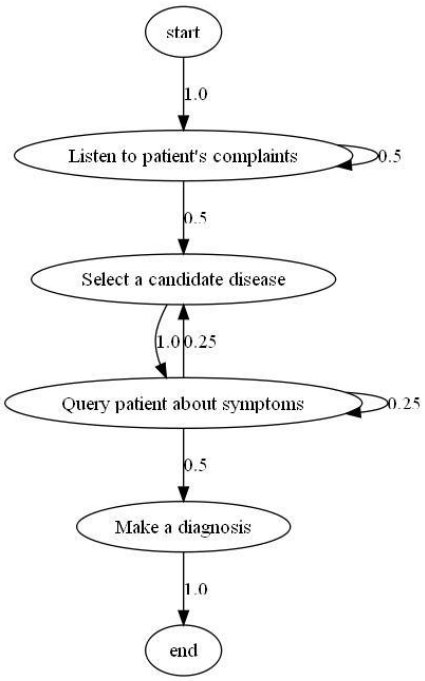

Fig. 1. Model mined from Table I.

Organization of the Paper

In this paper we compare performance of three algorithms for sequential pattern mining: the skeletal algorithm as described in [15], Bayesian inference method to build a Markov chain from a sample, and arithmetic coding based on Shannon entropy.

We assume that the reader is familiar with basic mathematical concepts. The paper is structured as follows. In Section II we shall briefly recall the idea of skeletal algorithms [15], Markov chain models and Shannon entropy-based models. Section III describes experimental results on our benchmark instances and discusses the outcomes. We conclude the paper in Section IV.

II. Models

If we forget about additional information and attributes associated with an execution of a process, then the task of identifying a process reduces to the task of language recognition. The theory of language recognition that gives most negative results is "identification of a language in the limit" developed by Mark Gold [16]. The fundamental theorem published by Dan Angluin [17] says that a class of recursively indexed languages is (recursively) identifiable in the limit iff for every language *L* from the class there exists an effectively computable finite "tell-tale" - that is: a subset *T* of *L* such that: if *T* is a subset of any other language *K* from the class, then *K* is not a subset of *L*. An easy consequence of this theorem is that the set of regular languages is not identifiable in the limit. Another source of results in this context is the theory of PAC-learning developed by Leslie Valiant [2].

Although these results are fairly interesting, in the context of sequential pattern recognition, we are mostly given a very small set of sample data, and our task is to find the most likely hypothesis --- the question: "if we were given sufficiently many data, would it have been possible to find the best hypothesis?" is not really practical.

In this section we briefly present three approaches to mine sequential patterns: skeletal algorithms, Markov chains and models based on Shannon entropy.

Skeletal algorithm [1], [2] is a new branch of evolutionary metaheuristics [3]-[6] concerned on data and process mining. The crucial observation that leads to skeletal algorithms bases on Minimum Description Length Principle [7], which among other things, says that the task of finding "the best model" describing given data is all about discovering similarities in the data. Thus, when we start from a model that fully describes the data (i.e. the skeletal model of the data), but does not see any similarities, we may obtain a "better model" by unifying some parts of the model. Unifying parts of a model means just taking a quotient of that model, or equally - finding a congruence relation.

Thus, skeletal algorithms search for a solution of a problem in the set of quotients of a given structure called the skeleton of the problem. More formally, let *S* be a set, and denote by *Eq*(*S*) the set of equivalence relations on *S*. If *i* is any element of *S* and *A* is an element of *Eq*(*S*) then by $[i]_A$ we shall denote the abstraction class of *i* in *A* - i.e. the set $\{j \in S : j\ A\ i\}$. We shall consider the following skeletal operations on *Eq*(*S*):

1) Splitting

The operation $split : \{0,1\}^S \times S \times Eq(S) \to Eq(S)$ takes a predicate $P : S \to \{0,1\}$, an element *i* from *S*, an equivalence relation *A* from *Eq*(*S*) and gives the largest equivalence relation *R* that is contained in *A* and satisfies: $\forall_{j \in [i]_A}\ i\ R\ j \Rightarrow P(i) = p(j)$. That is, it splits the equivalence class $[i]_A$ on two classes: one for the elements that satisfy *P* and the other of the elements that do not.

2) Summing

The operation $sum : S \times S \times Eq(S) \to Eq(S)$ takes two elements *i, j* from *S*, an equivalence relation *A* from *Eq(S)* and gives the smallest equivalence relation *R* satisfying *i R j* and

containing *A*. That is, it merges the equivalence class $[i]_A$ with $[j]_A$.

3) Union

The operation $union: S \times Eq(S) \times Eq(S) \to Eq(S) \times Eq(S)$ takes one element *i* from *S*, two equivalence relations *A*, *B* from *Eq(S)* and gives a pair $\langle R, Q \rangle$, where *R* is the smallest equivalence relation satisfying $\forall_{j \in [i]_B} i \, R \, j$ and containing *A*, and dually *Q* is the smallest equivalence relation satisfying $\forall_{j \in [i]_A} i \, Q \, j$ and containing *B*. That is, it merges the equivalence class corresponding to an element in one relation, with all elements taken from the equivalence class corresponding to the same element in the other relation.

4) Intersection

The operation intersection: $S \times Eq(S) \times Eq(S) \to Eq(S) \times Eq(S)$ takes one element *i* from *S*, two equivalence relations *A*, *B* from *Eq*(*S*) and gives a pair $\langle R, Q \rangle$, where *R* is the largest equivalence relation satisfying $\forall_{x,y \in [i]_A} x \, R \, y \Rightarrow x, y \in [i]_B \vee x, y \notin [i]_B$ and contained in *A*, and dually *Q* is the largest equivalence relation satisfying $\forall_{x,y \in [i]_B} x \, Q \, y \Rightarrow x, y \in [i]_A \vee x, y \notin [i]_A$ and contained in B. That is, it intersects the equivalence class corresponding to an element in one relation, with the equivalence class corresponding to the same element in the other relation.

Furthermore, we shall assume that there is also a fitness function $\Delta : Eq(S) \to \Re$. The general template of skeletal algorithm is shown on Fig. 2.

Given a finite list *K* of sample terms over a common alphabet $\sum$, we shall construct the skeletal automaton $skeleton(K) = \langle Q, q_s, \Delta \rangle$ of *K*, where:

- $S = \{\langle i,k \rangle : i \in n, k \in \{1,\ldots,|K(i)|\}\} \cup \{-\infty, \infty\}$

  $l(-\infty) = start, l(\infty) = end, l(\langle i,k \rangle) = K(i)_k$, where the subscript *k* indicates the *k*-th element of the sequence
- $\delta(-\infty, \langle i,1 \rangle) = 1, \delta(\infty, \infty) = 1,$

  $\delta(\langle i, |K(i)| \rangle, \infty) = 1, \delta(\langle i,k \rangle, \langle i,k+1 \rangle) = 1$

So the skeleton of a list of data is just an automaton corresponding to this list enriched with two states --- initial and final. This automaton describes the situation, where all actions are different. Given a list of sample data *K*, our search space *Eq(S)* consists of all congruences on *S*. Skeletal algorithm will try to glue some actions that give the same output (shall search for the best fitting automaton in the set of quotients of the skeletal automaton).

The Markov chain over *K* is the probabilistic transition system $markov(K) : \Sigma' \times \Sigma' \to [0,1]$, where:

- $\sum' = \sum \cup \{start, end\}$ is alphabet $\sum$ enriched with two additional symbols: the starting symbol *start* and the terminal symbol *end*
- if we denote by *K'* the list of sequences: $K'(i) = \langle start, K(i), end \rangle$

then for all $x, y \in \Sigma'$:

$$markov(K)(x,y) = \frac{the\ number\ of\ substrings \langle x,y \rangle\ in\ K'}{the\ number\ of\ occurrences\ of\ x\ in\ K'}$$

The Shannon model over *K* is the probability distribution $shannon(K) : \sum \cup \{end\} \to [0,1]$ given by:

$$shannon(K)(end) = \frac{n}{the\ total\ number\ of\ symbols\ in\ K + n}$$

and for all *x* in $\sum$:

$$shannon(K)(x) = \frac{the\ number\ of\ occurrences\ of\ x\ in\ K}{the\ total\ number\ of\ symbols\ in\ K + n}$$

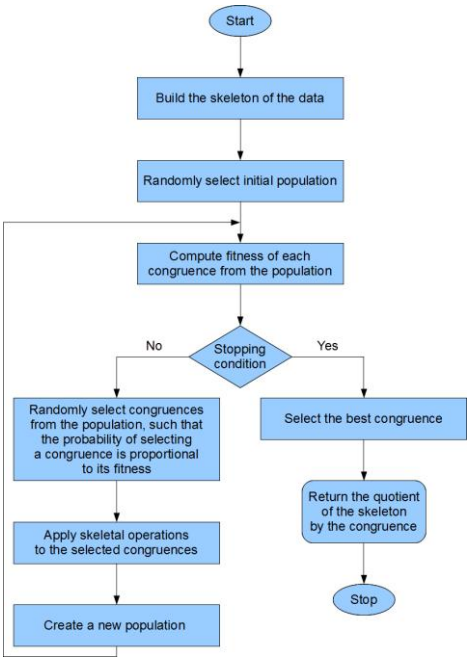

Fig. 2. Skeletal algorithm.

III. Experimental Results

We examine three classes of instances mentioned in [15]: finite state automaton, non-rational probability source, and some "hardcoded" sequences.

A. Non-Deterministic Automata

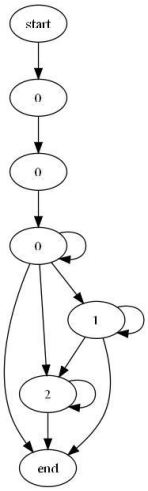

Fig. 3. Non-deterministic automaton.

Given a non-deterministic automata like on Fig. 3, we generate training sample of n=4, 16 words and a testing sample of 128 words by moving through each arrow with equal probability. Then we build three models based on the training sample: the model discovered by skeletal algorithms, the Markov chain model and the model based on Shannon entropy. For each of 128 sample words we compute the probability of generating the word by the true (optimal) model and each of the described models.

Fig. 4 shows the complexities in natural bits corresponding to the probabilities for n=4. Only 11 out of 128 words have finite complexities (i.e. non-zero probabilities) according to the model discovered by skeletal algorithms, and 80 out of 128 words have finite complexities according to the Markov chain model. Only Shannon model classifies correctly every word from our testing sample. Fig. 5 shows relative distances to the optimal complexity generated by the true model. Observe that all 11 cases of finite complexities discovered by skeletal algorithms, are lower than the optimal. Moreover, the total complexity of the training sample shown on Fig. 13: according to the skeletal models, was about four times lower than the optimal complexity. Clearly, the training sample was too short to learn reasonable models, and models overfit the data.

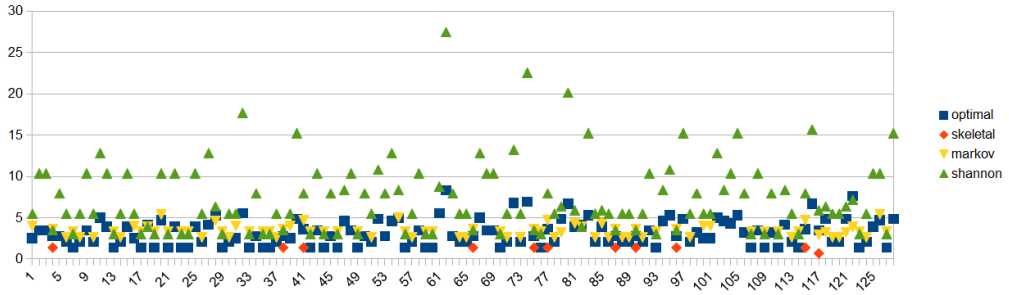

Fig. 4. Complexities of testing words for n=4. Average: optimal=3.17, skeletal=∞, markov=∞, shannon=7.33.

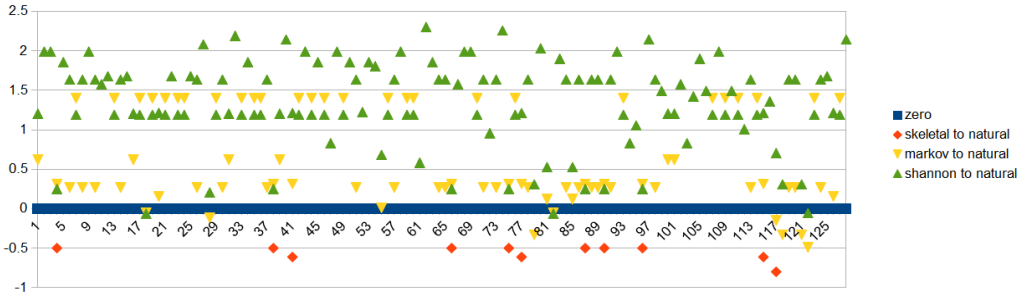

Fig. 5. Relative distances to the optimal complexity for n=4.

Fig. 6 shows the complexities in natural bits for n=16. All testing words have finite complexities in every model (Fig. 6). The model discovered by skeletal algorithms is close to the optimal (average 3.17 natural bits vs. 3.22 natural bits). As shown on Fig. 7, 76 words from our testing sample have higher complexity, and 52 words have lower complexity according to the skeletal model. Markov model cannot take advantage of the knowledge that every word starts from three consecutive zeros and, thus, uses about one natural bit more information to describe samples. The total complexity of the training sample (see Fig. 13) in the skeletal model is still a bit lower than the optimal one, which means that the model is still slightly overfitted.

B. Prime Numbers

In this example we show how one can learn from a probabilistic source p that does not correspond to any finite-state model. We define p to be non-zero only on prime numbers, and such that the probability for the k-th prime number is proportional to the number of bits in binary representation of k.

We generate training sample of n=64, 1024 prime numbers and a testing sample of 128 prime numbers. Then we build three models based on the training sample: the model discovered by skeletal algorithms, the Markov chain model and the model based on Shannon entropy. For each of 128 sample prime numbers we compute the probability of generating the number by p (times the probability of generating a number at all; we call the probability distribution obtained in such a way "the natural model") and the probability in each of the described models. For n=64 the average natural complexity is 1.33 natural bits, 1.37 according to the skeletal model, 2.22 according to the Markov chain model and 3.51 natural bits according to the Shannon model (see Fig. 8 and Fig. 9). Detailed data for n=1024 are shown on Fig. 10 and Fig. 11. Although there is no model corresponding to p that can be discovered by skeletal algorithms (i.e. the language of prime numbers is not regular), the skeletal complexity is almost indistinguishable form the natural complexity induced by probability distribution p.

C. Sequences

In this example we use samples from [15], [26]:

- L1 = A, B, C, A, B, C, B, A, C, B, A, C, A, B, C, B, A, C,

B, A, C, A, B, C, B, A, C, A, B, C, A, B, C, B, A, C, B, A

- L2 = A, B, C, D, C, E, F, G, H, G, I, J, G, I, K, L, M, N, O, P, R, F, G, I, K, L, M, N, O, P, Q, S

Fig. 12 shows the total complexities of sequences L1 and L2 according to the models discovered by skeletal algorithms, the Markov model and the model based on Shannon entropy. The exact values are shown on Fig. 13. The complexities of L2 are pretty low in both skeletal and Markov models - almost the whole information about the sequence is encoded in the models themselves.

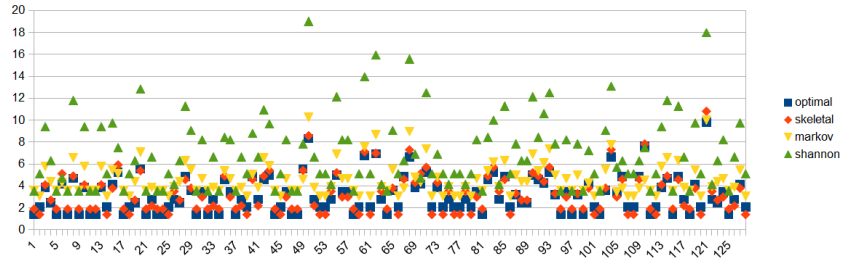

Fig. 6. Complexities of testing words for n=16. Average: optimal=3.17, skeletal=3.22, markov=4.53, shannon=6.88.

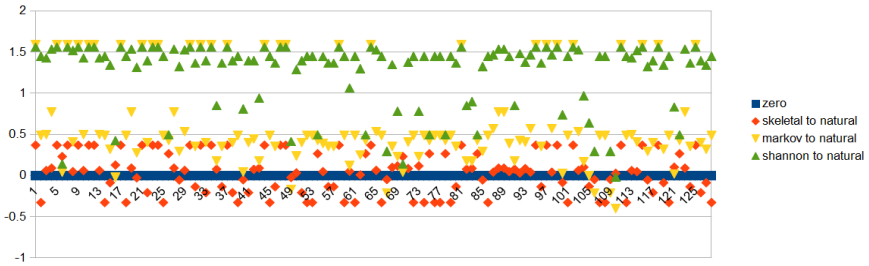

Fig. 7. Relative distances to the optimal complexity for n=16.

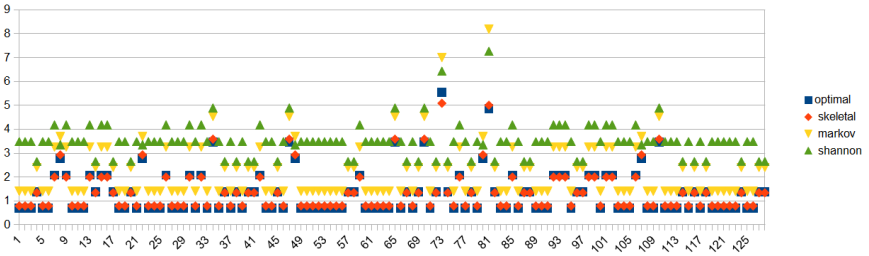

Fig. 8. Complexities of primary numbers, n = 64. Average: natural=1.33, skeletal=1.37, markov=2.22, shannon=3.51.

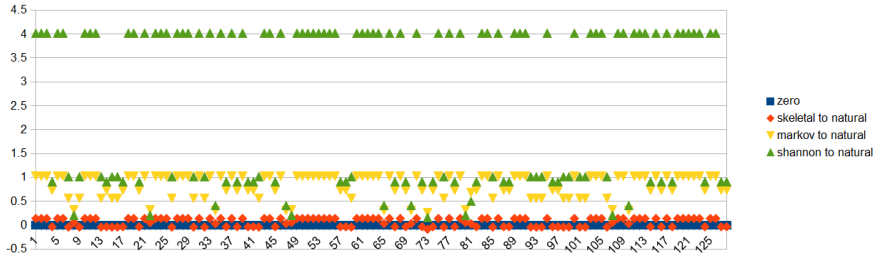

Fig. 9. Relative distances to the natural complexity for primary numbers, n = 64.

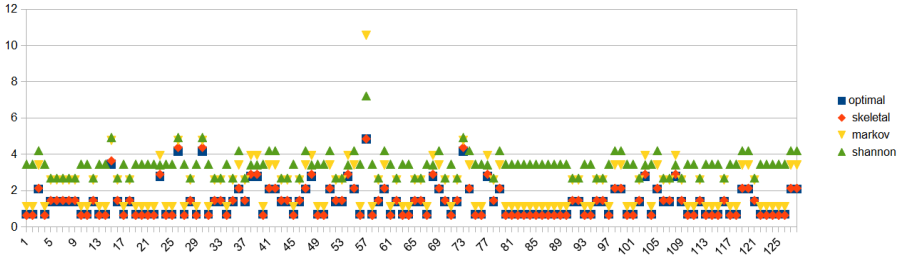

Fig. 10. Complexities of primary numbers, n = 64. Average: natural=1.33, skeletal=1.35, markov=2.37, shannon=3.4.

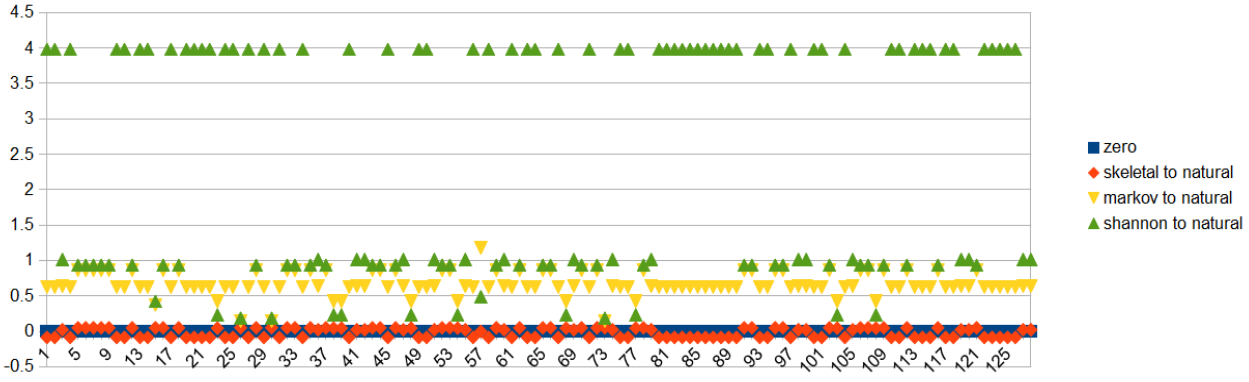

Fig. 11. Relative distances to the natural complexity for primary numbers, n = 1024.

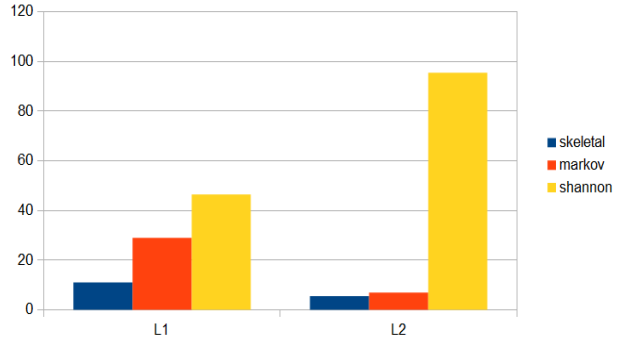

Fig. 12. Total complexities of L1 and L2 according to the models.

	optimal	skeletal	markov	shannon
Model 4	13.2875794663	4.1588830834	14.24390502	23.17784778
Model 16	48.1805045656	46.1078825264	68.19640608	106.5784143
Primes 64	97.733752459	99.4620165859	158.2788421	227.4110817
Primes 1024	1400.157304731	1400.321399947	2254.668189	3567.847518
L1	nd	11.0211334199	28.965961	46.37134111
L2	nd	5.5451774445	6.931471806	95.45338033

Fig. 13. Complexities of training samples according to the models.

IV. Conclusion

This paper compares performance of three algorithms for sequential pattern mining: the skeletal algorithm as described, Bayesian inference method to build a Markov chain, and arithmetic coding based on Shannon entropy. Our benchmark instances include: finite state automaton, non-rational probability source (prime numbers) and some hardcoded sequences. We generated random samples from the instances and divided them on two classes: the training samples and the testing samples. We used training samples to rediscover models, and then tried to find complexities of testing samples (or tried to "compress" testing samples) according to the models (i.e. the shorter description is, more the model knows about the sample). In each case skeletal algorithms outperformed Markov chains followed by Shannon models - on condition that the training samples were of a reasonable size. Moreover, when "a priori" distribution was given, skeletal algorithms built the model close to the real one.

In future work we will be interested in performance of skeletal algorithms applied to non-sequential pattern recognition.

References

[1] S. W. M. P. van der Aalst, *Process Mining: Discovery, Conformance And Enhancement of Business Processes*, Springer Verlag, 2011.
[2] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, 1984.
[3] A. J. M. M. Weijters and W. M. P. van der Aalst, "Process mining: Discovering workflow models from event-based data," in *Proc. the 13th Belgium-Netherlands Conference on Artificial Intelligence*, Maastricht, 2001.
[4] A. K. A de Medeiros, B. F. van Dongen, W. M. P. van der Aalst, and A. J. M. M. Weijters, "Process mining: Extending the alpha-algorithm to mine short loops," BETA Working Paper Series, Eindhoven, Eindhoven University of Technology, 2004.
[5] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," BPM Center Report, 2000.
[6] N. R. Mabroukeh and C. I. Ezeife, "A taxonomy of sequential pattern mining algorithms," *ACM Computing Surveys*, vol. 43, issue 1, 2010.
[7] C. H. Mooney and J. F. Roddick, "Sequential pattern mining - Approaches and algorithms," *Computing Surveys*, vol. 45, issue 2, 2013.
[8] M. T. Wynn, D. Edmond, W. M. P. van der Aalst, and A. H. M. ter Hofstede, "Achieving a general, formal and decidable approach to the or-join in workflow using reset nets," BPM Center Report BPM-04-05, 2004.
[9] W. M. P. van der Aalst, A. K. Alves de Medeiros, and A. J. M. M. Weijters, "Process equivalence in the context of genetic mining," BPM Center Report BPM-06-15, 2006.
[10] W. M. P. van der Aalst and M. S. M. Pesic, "Beyond process mining: From the past to present and future," BPM Center Report BPM-09-18, 2009.
[11] A. George and D. Binu, "An approach to products placement in supermarkets using prefixspan algorithm," *Journal of King Saud University-Computer and Information Sciences*, vol. 25, issue 1, 2013.
[12] W. M. P. van der Aalst and B. van Dongen, "Discovering workflow performance models from timed logs," *Engineering and Deployment of Cooperative Information Systems*, pp. 45-63, 2002.
[13] L. Wen, J. Wang, and J. Sun, "Detecting implicit dependencies between tasks from event logs," *Lecture Notes in Computer Science*, vol. 3841, 2006.
[14] C. R. Ren, L. J. Wen, J. Dong, H. W. Ding, W. Wang, and M. M. Qiu, "A novel approach for process mining based on event types," in *Proc. IEEE International Conference on Service Computing*, 2007, pp. 721-722.
[15] M. R. Przybylek, "Skeletal algorithms in process mining," *Studies in Computational Intelligence*, vol. 465, 2013.
[16] E. M. Gold, "Language identification in the limit," *Information and Control*, vol. 10, 1967.
[17] D. Angluin, "Inductive inference of formal languages from positive data," *Information and Control*, vol. 42, 1980.
[18] M. R. Przybylek, "Tree automata mining," *Studies in Computational Intelligence*, Springer-Verlag, 2014.
[19] M. R. Przybylek, "Dynamic data discovery," *Advances in Intelligent Systems and Computing*, Springer-Verlag, 2014.
[20] M. R. Przybylek, "Algebraic pattern recognition," in *Proc. 5th International Conference on Graphic and Image Processing*, Hong Kong, 2013.

[21] H. J. Bremermann, "Optimization through evolution and recombination," *Self-Organizing Systems*, Washington, Spartan Books, 1962.
[22] R. M. Friedberg, "A learning machines part I," *IBM Journal of Research and Development*, vol. 2, 1956.
[23] R. M. Friedberg, B. Dunham, and J. H. North, "A learning machines part II," *IBM Journal of Research and Development*, vol. 3, 1959.
[24] J. H. Holland, *Adaption in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, 1975.
[25] P. D. Grunwald and J. Rissanen, *The Minimum Description Length Principle, Adaptive Computation and Machine Learning Series*, The MIT Press, 2007.
[26] J. E. Cook and A. L. Woolf, "Discovering models of software processes from event-based data," *ACM Transactions on Software Engineering and Methodology*, vol.7, issue 3, 1998.

Michal R. Przybylek was born on 22 April, 1983 in Bydgoszcz, Poland. He is a PhD student in computer science at University of Warsaw. He graduated from University of Warsaw in 2008 with a master's degree in computer science. He also holds a diploma of electronics from Technical School of Electronics in Bydgoszcz. His major field of study is multicriteria optimization, but is also interested in categorical logic and type theory.

From 2008 to 2012 he worked for a military resort - designed and co-developed systems to allow for communication on a battlefield; the systems are currently used by Polish soldiers in missions abroad. Since 2013 he is a researcher in Polish-Japanese Institute of Information Technology and the leader of a research group aimed at multicriteria optimisation.