`tt-analyze` and `tt-generate`: Tools to Analyze and Generate Sequences with Trained Statistical Properties

Andre Dau and Johannes Krugel

Technische Universität München, Germany
{dau,krugel}@in.tum.de

Abstract. Algorithms working on sequences are influenced by the statistical properties of the sequences. Algorithms for fragment assembly for example usually produce a worse result if there are many repetitions. Also the space usage and running time of many data structures and algorithms depend on the statistical properties of the underlying text.
We implemented `tt-analyze`, a tool to analyze sequences for certain statistical properties, among others the entropy, the number and distribution of different substrings, and the repeat structure. Besides, we also designed and implemented `tt-generate`, a tool to generate synthetic sequences with certain predefined properties, using models such as a Markov process, a discrete autoregressive process, and a repeat model. In bioinformatics these models have primarily been used to analyze given sequences, whereas here, we use them to also generate synthetic ones. The respective parameters of the models can be defined manually or be learned from given training data.
The combination of both tools allows to generate sequences that are similar to real world sequences with respect to certain properties. This will allow to investigate the performance of algorithms under to some extent realistic, yet controlled conditions, and to determine the degree of dependence from parameters of the underlying sequence.
Both tools have an extensible design which allows the integration of new modules for other statistical properties or generating models with the same programming interface.

Keywords: models, genome analysis, efficient sequence analysis, efficient algorithms, machine learning, Markov process

1 Introduction

Analyzing the statistical properties of sequences plays an important role in bioinformatics as well as in other areas such as automatic translation, text compression, etc. Sequences of interest in these applications are for example DNA or protein sequences, natural language texts, or binary sequences. The wide range of properties that are analyzed include for instance the character distribution, different measures of entropy, the number of different substrings, and the number of repeats. This statistical analysis can then lead to biological or linguistic conclusions and can also help to understand the performance of algorithms and data structures.

Another general goal is to build theoretical models for the underlying generating process of the sequences. These models are often stochastic processes such as Markov chains, but can also be based on formal grammars or yet completely different approaches. Often these models are based on a set of parameters and these parameters can in some cases be learned from given real world data. With the resulting fully determined model it is then possible to generate new, artificial sequences which resemble certain modeled properties of real sequences.

Motivation Synthetically generated sequences can be used as test instances in bioinformatics, for example to evaluate the performance of genome sequencing methods. The benefit of artificial sequences is that complete knowledge about the correct solution is available. This is opposed to the sequencing of a real genome, where the correct solution is

unknown. Furthermore, a generator for artificial sequences allows to study in isolation the impact of one parameter on the quality of the solution.

Another motivation is to enable a systematic comparison of data structures and algorithms on strings. The space consumption and the running time of index structures and algorithms for pattern matching often depend on the statistical properties of the underlying text. The size of a q-gram index, a suffix tree, and a compressed index structure depends for example on the number of different q-grams, the number of repetitions and the entropy of the text, respectively (these dependencies will be explained in more detail below). If the statistical properties of given sequences can be determined and if it is even possible to generate artificial sequences with predefined properties, one can measure the impact of these properties on the performance of the algorithms and data structures. By learning the properties from real world data, it is possible to examine this dependency under controlled but realistic conditions. The goal is to make statements characterizing which algorithmic approach works well under which circumstances.

Contribution We built two tools: `tt-analyze` to analyze sequences for various statistical properties and `tt-generate` to generate artificial sequences based on different models. Both tools are implemented extensible and provide a uniform interface for the different statistical properties and generating models, respectively. The tools as well as the test data are available online.[1]

The generator uses methods that were previously used in bioinformatics mainly to analyze sequences, but not to generate them. This includes a Markov process, a discrete autoregressive process [JL83], and an approximate repeats model [AED98], where we extend the underlying generating model from a 0-order Markov process to a higher order Markov process. The parameters of the models can either be defined manually or learned from given training data. We studied the quality of this learning process by comparing the generated sequences to the original sequences and observed that some statistical properties are modeled well by the respective generating processes, whereas others are not.

Test Sequences The methods can be used with different types of biological sequences, such as DNA or protein sequences and we also evaluated the performance for natural language texts. For the presentation we focus here on DNA sequences. In the evaluation we used the following sequences:

- A fragment of coding DNA of the fruitfly: downloaded from the NCBI[2] in April 2010. (Total length: 6333 bp, FASTA header: `>gi|8203|emb|X54251.1| D. melanogaster neurogenic locus mastermind mRNA for a nuclear protein`)
- Human genome: the set of all human chromosomes, downloaded from the NCBI in August 2009. (Total length: ca. 3 Gbp)
- German: a concatenation of all German texts from the Project Gutenberg[3], downloaded in August 2009 (ca. 275 MB)
- French: see above (ca. 800 MB)

Outline The rest of this paper is organized as follows: Section 2 describes related work addressing the analysis and generation of sequences, in particular in the context of bioinformatics. Section 3 contains motivations and definitions for the statistical properties considered and shows how they are measured by `tt-analyze`. Section 4 explains the different models used by `tt-generate`, describes how to infer their parameters from training data, and shows experimental results. Section 6 concludes with a discussion and an outlook of other models to be integrated.

[1] http://www14.in.tum.de/papi/
[2] http://www.ncbi.nlm.nih.gov/
[3] http://www.gutenberg.org/

2 Related Work

At the time when the human genome had not yet been fully sequenced, Myers built a dataset generator for shotgun sequencing [Mye99]. This tool is called `celsim` and generates sequences that are similar to real genome DNA sequences regarding the repeat structures. This is achieved by using a stochastic formal grammar. Experiments comparing the synthetically generated sequences with real genomic sequences indicate that this model is a good approximation for the properties of the repeat structures. However, other properties such as the entropy or the distribution of the q-grams are not modeled. The tool further contains a module to produce other sequences of related individuals or species based on a given input sequence, by applying simple block deletions, translocations, or single point substitutions. Furthermore it is possible to simulate the shotgun sequencing process by extracting fragments from generated sequences. One important reason for building `celsim` was the lack of available real world data (only 6 % of the human genome had been sequenced at this time). Nevertheless, even today it is important to have a generator for synthetic, but realistic sequences to test the influence of the statistical parameters on the performance of algorithms.

Another model for DNA sequences tries to explicitly describe repeated regions, also for the case that the repeat contains some deviations [AED98]. This model is based on ideas from Lempel and Ziv and would allow to compress DNA sequences. But the actual goal of this model is to make conclusions about the statistical and thereby biological properties of the underlying sequence (see also [SACD01] and [DPA+07]). The base sequence is assumed to have been generated by a zero-order or first-order Markov model and the repeat model is used to generate new sequences. We will extend this by implementing a generator containing approximate repeats using a higher-order Markov model in Section 3.4.

In bioinformatics some related tools have been developed. On the one hand, there are tools to simulate the evolution of a given genome sequence, e. g. `sgEvolver` of the software package `Mauve` [DMBP04], `Mutagen` of the alignment evaluation suite `ThurGood` [SMM+04], and the recently developed program `Evolver` [EABS09]. On the other hand, there are tools to simulate the extraction of reads from a genome, e. g. `MetaSim` [ROA+08] and `Mason` [Hol10]. Both kinds of tools can be used very well together with a sequence simulator like `celsim` or the generator described in this paper.

In the context of index structures for pattern matching there is a text generator `gentext` available as part of the Pizza&Chili Corpus[4], a benchmark for compressed index structures [FGNV09]. It produces texts following a uniform distribution over an alphabet of adjustable size. Additionally there are also texts with artificially introduced repeats available for download, but their generator is not available online and presumably is not able to learn the parameters from given training data.

In bioinformatics and computer science there are countless other models trying to describe the generation of character sequences. One recent approach is the `Sequence Memoizer` [WGA+11] which tries to extend Markov models so that also long-range dependencies and power-law properties can be modeled, while staying computationally feasible. The applications are language modeling, a good prediction of the next character in the sequence, and text compression, but in this work the model is not used to generate sequences.

3 Sequence Analysis

In order to analyze statistical properties of sequences it is necessary to make an assumption about the underlying model that generates the sequences. Within this work a sequence over an alphabet Σ is regarded as an instance of a *stationary ergodic discrete stochastic process* $(X_t)_{t\in\mathbb{N}}$ with $X_t : \Omega \to \Sigma$. (In general the assumption of stationarity of DNA sequences is controversial, since for example chromosomes exhibit higher level structures. However, the properties analyzed here are not affected significantly by small deviations of stationarity [DHH03].) The simplifying

[4] http://pizzachili.dcc.uchile.cl/, http://pizzachili.di.unipi.it/

assumption of stationarity and ergodicity allows to use one finite sample of the generating process and to link the relative frequencies in the sequence with the probabilities of the process.

This section gives an overview of the statistical properties implemented in the tool `tt-analyze` so far. For each property we will motivate why it is interesting (especially in the context of bioinformatics) and describe how to measure it. In the implementation, the properties are realized as different modules with a uniform interface, allowing an integration of new modules.

3.1 *q*-gram Frequencies

A q-gram of a string is simply a substring of length q.[5] (The number of different 1-grams for example equals the size of the actually used alphabet.)

In the field of bioinformatics there is active research regarding the distribution of q-grams, for example in examining their frequency distribution in different species [CHG+09]. Another well known application of q-gram analysis is the difference of the CpG content in coding and non-coding regions respectively [FPJ+05,CHG+09]. In natural language texts, q-grams can for example be used to identify the language of a given text [Dun94].

Furthermore, the performance of some data structures for strings (like the q-gram index [NBY98]) depends on the number and the distribution of the q-grams. To evaluate the performance of q-gram based index structures one can therefore analyze texts for their q-gram distribution, perform experiments with the index structure, and then make statements relating the statistics of the text with the performance of the data structure.

`tt-analyze` calculates for a given sequence and a given value q the number of different q-grams and for each q-gram the number of occurrences. In the implementation, memory is saved by storing the q-grams not explicitly but solely the position within the sequence. Therefore it is possible to also handle sequences containing many different q-grams (e. g. like in natural language texts).

We used `tt-analyze` for example to analyze the 2-gram frequencies of the coding subsequence of Drosophila melanogaster (see Section 1) with human chromosome 22. The output of the tool facilitates to analyze the difference in the distributions of the 2-grams, for example the difference in the CpG content of both sequences.

3.2 Entropy

The *Shannon entropy* H is a well known measure for the degree of randomness (or equivalently the information) within a sequence [SPS48].[6] In information theory the entropy is a lower bound for the optimal compression rate. The entropy H of one random variable X_t is formally defined as $H(X_t) := -\sum_{w\in\Sigma} p(X_t = w) \log p(X_t = w)$ and represents the uncertainty about the value of the random variable X_t. It can canonically be extended to multiple random variables by using tuples of random variables. The conditional entropy of order n is denoted by $H(X_t \mid X_{t-n}, \ldots, X_{t-1})$ and represents the remaining uncertainty about the value of the random variable X_t given that the values of the random variables $X_{t-n}, \ldots, X_{t-1}$ are known.

Under the assumption that a stochastic process is stationary and ergodic, it is possible to estimate its entropy and conditional entropy of order n by analyzing one sequence generated by the process. The block entropy of order n of a stochastic process $(X_t)_{t\in\mathbb{N}}$ can then be defined as $H_n := H(X_1, \ldots, X_n)$.

In biological applications the entropy can be used to draw conclusions about biological sequences. In some species it is observed to be lower in non-coding regions, whereas in other species it lower in coding regions: An analysis of the genome of Escherichia coli revealed for example that the entropy in coding regions is slightly lower than in non-coding regions, and the entropy of triplets of nucleotides in the correct reading frame is significantly lower than in the wrong

[5] In other contexts q-grams are sometimes also called n-gram and k-mer.

[6] From here on we simply use the term *entropy* to denote the Shannon entropy.

reading frame [LIHL92]. Opposed to this, another study on the entropy of 37 eukaryotic sequences from GenBank (in the year 1994) observed a lower entropy in the non-coding regions [MBG+94]. In any case, depending on the species, the analysis of the entropy of a DNA sequence can possibly give an indication when searching for coding regions of previously unidentified genes.

The entropy also plays an important role in the analysis of data structures for strings, since many compressed index data structures (like the different versions of the compressed suffix arrays, the FM-index or Ziv-Lempel based index structures [NM07]) have the related measure *empirical entropy* [Man01] of the sequence as a factor in their space usage. The index structures therefore need less space if the text has a low entropy, which fortunately often is the case for real world instances.

`tt-analyze` measures the block entropy and the conditional entropy for a given order n by using relative $n+1$-gram and n-gram frequencies as estimates. However, this estimation is biased resulting in too small values for the entropy, since relative frequencies are used instead of actual probabilities of the (unknown) underlying stochastic process [SG96]. There are several possibilities to accommodate for this effect. `tt-analyze` uses a correction term which is based on a Taylor series of order 1 as described in [SG96].

Using techniques from [HES94,HD06] we tried to verify some of their results with `tt-analyze`. We calculated the conditional entropies for the first n orders and used the output from `tt-analyze` to create diagrams similar to the ones in [HES94,HD06]. It is possible to see the "increasing decline" (negative second derivative) of the conditional entropy in Figure 1 (right). When using `tt-analyze` to calculate the conditional entropy of natural language texts, we found that with higher orders it decreases fast (as expected) since of all possible q-grams only a small subsets occurs in natural language. This can be interpreted that the distribution of one character is mainly determined by very few characters preceding it.

3.3 Mutual Information

The mutual information I is an entropy-based measure for the correlation between two random variables: $I(X,Y) := H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$ [SPS48,Li90]. When used with a stochastic process it can measure the correlation a random variable at two points in time: $I(k) := I(X_t, X_{t+k})$. The mutual information function can model only simple kinds of dependencies but is relatively fast to compute even for larger ranges of the parameter k.

A biological application is the detection of short-range and long-range correlations in DNA sequences [LK92,HG95,HGB+03,GHBS00]. For example Grosse et al. claim that the mutual information function can be used to differentiate coding and non-coding regions without prior training or knowledge of the exact species [GHBS00]. In many cases the codon structure inside a coding region is visible through the mutual information function [HG95].

To measure the mutual information, `tt-analyze` estimates the pairwise conditional probabilities between two symbols of distance k by using relative frequencies. Since this can be done fast, it is possible to evaluate the function for many values of k allowing the analysis of its development with growing k.

Using `tt-analyze` we could expose the codon structure in the Drosophila melanogaster neurogenic locus mastermind mRNA as can be seen in Figure 1 (left). We could also observe in accordance with the results of [LK92] that the value of the function always stayed above the expected value for a random sequence also for higher orders k, indicating long-range correlations. Additionally we used `tt-analyze` to analyze natural language texts and observed that the mutual information function takes high values for small values of k (k in the order of a small word), but then decreases rapidly. (Our analysis of both German and French texts showed similar characteristics.)

3.4 Approximate Repeats

Repetitive structures are very common in DNA as well as in natural language texts. They influence the performance of algorithms and data structures, such as suffix trees where the internal nodes correspond to the substrings occurring

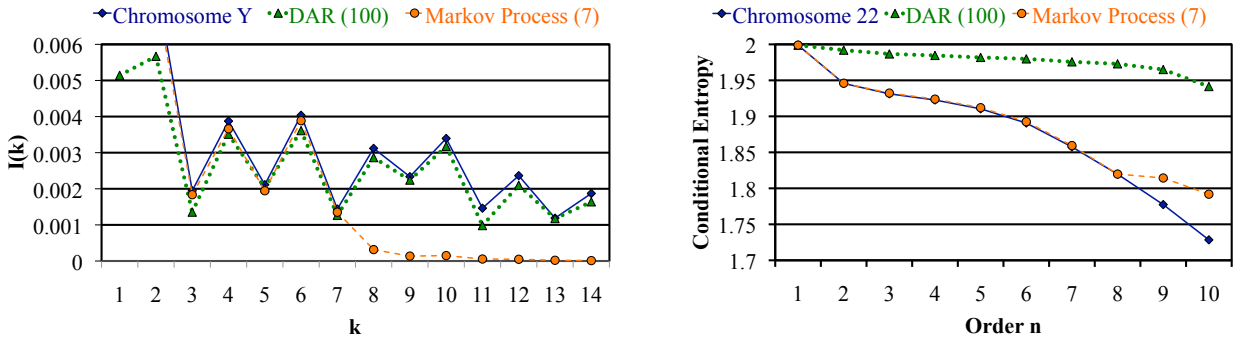

Fig. 1. *Mutual Information Function (left):* Comparison of the mutual information of the human chromosome Y, a sequence generated by a discrete autoregressive process (order 100), and a Markov process (order 7), both trained on the chromosome sequence. The copying mechanism of the discrete autoregressive process can reproduces the correlation structure well. The Markov process models the function well up to its order n and then drops rapidly, failing to model the mutual information for higher orders. (See also Section 3.3, Section 4.1, and Section 4.2.)
Conditional Entropy (right): Comparison of the conditional entropy of the human chromosome 22, a sequence generated by a discrete autoregressive process (order 100), and a Markov process (order 7), both trained on the chromosome sequence. It turns out that the rather simple discrete autoregressive process is not able to reproduce the correlations as measured by the entropy. The trained Markov process can model the properties of the original sequence up to order $n + 1$ after which the entropy remains almost constant. (See also Section 3.2, and Section 4.1.)

more than once [Gus97]. Repeats also have an influence on biological algorithms: fragment assembly in the shotgun sequencing process for example works best if there are few and short repeats. Detecting and modeling repeats in DNA is not trivial if one also wants to allow approximate repeats, i. e. repeats containing a certain number of deviations.

The model by Allison et al. [AED98] is based on a Markov process combined with ideas of Lempel and Ziv. Besides generating characters according to the Markov model, there is an additional possibility to start a repeat. Within a repeat there are four operations: copy, change, insert and delete.

Given a sequence, the goal is to extract the parameters of the underlying (unknown) model, that generated the sequence. The resulting parameters allow to make general statements about the frequency of the approximate repeats and the degree to which the repeats differ from each other within this sequence. This can for example help to determine if a given sequence contains a lot of small repeats or a few long repeats [AED98]. Other applications of the same model are [SACD01] and [DPA+07].

The estimation procedure works by defining a repeat graph (see [AED98] for an illustration) which represents possible explanations of how the training sequence might have been generated. The likelihood that the sequence was generated by this model is improved iteratively by an expectation-maximization (EM) algorithm. The algorithm has per iteration a running time quadratic in the length of the sequence and needs linear space. It is possible to speedup the process by only inspecting the most relevant parts of the graph around exact repeats of minimum length [AED98].

In `tt-analyze` we implemented this model and the parameter estimation procedure (including the speedup). In the original proposal a Markov chain of order 0 or 1 is used with two kinds of repeats, namely forward and reverse-complementary repeats. We extended this by using a higher order Markov chain and additionally reverse repeats.

We focused on evaluating the parameter estimation algorithm rather than analyzing biological sequences for their repeat structure (like it is done in [SACD01] and [DPA+07]). The conclusions regarding the parameter estimation are described together with the generating process in Section 4.3.

4 Sequence Generation

The tool `tt-generate` generates sequences based on a chosen model. The model's parameters can either be defined manually or trained from real sequences using `tt-analyze`. `tt-generate` has a modular framework facilitating the integration of new models. In this section we will give a brief overview of the models implemented so far.

For each model, we give a description together with a motivation and discuss the advantages and limitations. Each model was tested in the following way: A training sequence was analyzed and used for parameter estimation. Next, artificial sequences were generated with the trained model and analyzed again, using the methods described in the previous section. The results were then compared to the original sequence to evaluate the parameter estimation.

4.1 Markov Process

Markov processes have been studied in many areas and are also very popular in bioinformatics. Of special interest are time-homogeneous Markov processes where the transition probabilities do not depend on t. They can be defined by a starting distribution for the first n symbols and a single transition probability distribution for all X_t with $t > n$. The defining property of a Markov process of order n is that the outcome of a random variable X_t only depends on the outcome of the previous n variables. In the context of sequences this means that the distribution at a certain position only depends on the n preceding characters.

Markov processes allow to accurately model short-range correlations between positions within a distance of at most n. In the analysis of mammalian and non-mammalian DNA sequences it has for example also been observed that lower order Markov processes can also model complex q-gram distributions well [CHG+09].

In order to train a Markov process from a given sequence we assume that the underlying Markov process is time-homogeneous, irreducible, and stationary. Time-homogeneity allows us to estimate transition probabilities by relative transition frequencies. Stationarity ensures that the starting distribution is a stationary distribution. Irreducibility ensures that this stationary distribution can be estimated by n-gram frequencies. Both transition frequency and relative n-gram frequency can be calculated in a single run counting $n + 1$-grams. To ensure the resulting Markov process is irreducible as well, we decided to append the first n-gram to the end of the training sequence. If the sequence is long compared to n this modification will not change the distributions significantly.

We found that a trained Markov process of order n models almost exactly the q-gram distribution of the original sequence for $q \leq n + 1$. This is no coincidence since the trained Markov process is irreducible and therefore the relative frequencies in a generated text will be roughly equal to the stationary distribution. Since the stationary distribution of the trained process is approximately the relative frequency distribution in the original sequence, the q-gram distributions will be similar in original and generated sequences. As a consequence, both the entropy and the mutual information function of the generated and the original sequence are very similar up to order n or $n+1$ for block entropy respectively. Our experiments verify this theoretical result as can be seen in Figure 1 (left and right).

4.2 Discrete Autoregressive Process

Markov processes are very flexible but require an exponential amount of memory (since there are $|\Sigma|^n$ many distinct n-grams). A discrete autoregressive process of order n is a simplified Markov process [JL83]. It can be represented very compactly (with $n + |\Sigma| + 1$ parameters) and can therefore be used also with higher orders. The behavior is similar to a Markov chain of order zero but additionally has a certain probability to copy one of the n preceding characters according to a given distribution. A discrete autoregressive process models simple correlations up to order n.

In bioinformatics discrete autoregressive processes can model short-range correlations and can help to filter out long-range correlations [DHH03]. Discrete autoregressive processes are good in capturing the mutual information function of DNA sequences. Despite their simplicity they are accurate enough to allow the distinction between species [DPHH05]. An introduction to discrete autoregressive processes for DNA sequences can be found in [HD06].

`tt-analyze` estimates the parameters for the model using a method described in [JL83,DHH03] based on the autocorrelation coefficients of a discrete autoregressive process. (Note that these autocorrelation coefficients can also be used to analyze sequences as shown in [DPHH05,HD06]).

In our experiments we observed in accordance with [DHH03,HD06] that a trained discrete autoregressive process can model the mutual information of DNA to a certain extent because some of the correlation in DNA stems from duplications within the sequence (see Figure 1 left). The entropy of a generated sequence differs substantially from the training sequence, because the simple copying mechanism does not reflect more complex dependencies. The correlations in natural language texts are not based on the copying of a previous character, but on more complex structures. Therefore the discrete autoregressive model performs rather poorly on natural language texts.

4.3 Approximate Repeats Model

We implemented the approximate repeats model from section Section 3.4 in `tt-generate`. The parameters can be estimated by `tt-analyze`.

First, we analyzed the impact of the repeats on the entropy and mutual information function in sequences generated by this model. It turned out that the approximate repeats destroyed short-range correlations. Compared to a Markov process without repeats, the mutual information function was too low. For higher values, the function has small fluctuations preventing it from converging to zero.

Second, we tested the parameter estimation process. Similar to [AED98] we generated several sequences of length 500 and 1500 for several sets of predefined parameters. During the generation we used all three repeat types simultaneously together with a Markov process of order 4. We then tried to recover the known parameters from the generated sequences. Due to high fluctuations, we used the median of the parameter for the generated sequences in order to get acceptable results [AED98]. As expected, for sequences of length 1500 the algorithm performed better. If the initial values were too far away from the actual value, many iterations were needed. The time for one iteration is significantly higher for 1500 and at the moment it is impossible to analyze whole chromosomes due to the quadratic runtime. The heuristic speeds up the process, but quickly leads to a strong underestimation of repeat starts.

4.4 Other Models

Apart from the models mentioned above we also implemented two other more simple generators. The first generator outputs sequences according to a simple uniform distribution of characters over a given alphabet. The alphabet can be manually defined, but the user can also chose from predefined alphabets such as `dna` or `amino`. The other generator outputs Fibonacci words of a given length [Lot97]. Fibonacci words are especially interesting when analyzing algorithms or data structures working on texts (such as construction algorithms for suffix trees or suffix arrays [GKS03]) due to their repetitive structure.

5 Program Interface

Both `tt-analyze` and `tt-generate` are implemented in C++ as command line tools. The implementation is modular, allowing new modules for statistical properties and generators to be integrated into the framework.

The parameters for `tt-analyze` can be specified in a file or via command line. There are also a number of presets which reduce the number of parameters one has to specify. Options for `tt-analyze` include skipping of the first line (for FASTA files), unifying whitespace, ignoring certain characters (such as “N” in DNA sequences) and ignoring case. The output is given in the CSV (comma separated value) format.

`tt-generate` generates a sequence with specified length using the selected model and parameters. Parameter estimation results from `tt-analyze` can be piped directly to `tt-generate`.

6 Discussion

One motivation for the sequence generator was to be able to generate sequences similar to biological sequences. When the focus is on short-range correlations, this works fairly well by using the Markov process. However, long-range correlations, such as the complex repeat structure of a genome, can not be modeled due to the exponential growth of the number of parameters. The approximate repeat model has also some restrictions limiting its usability as simulator of genome-like sequences. Currently, the probability for a repeat start and end is constant, leading to uniformly distributed repeats of geometrically distributed lengths. This is not very realistic and can be extended to arbitrary distributions as formulated in [AED98]. Another suggested improvement is the incorporation of more advanced edit operations using gap costs. The main disadvantage of this model is its quadratic runtime which makes it infeasible for longer sequences such as whole chromosomes even when using the speedup heuristic. Unfortunately, the parameter estimation works the better the longer the training sequence is.

For using `tt-generate` also as a simulator for genome-like sequences we are currently integrating the idea of the `celsim` generator of Myers [Mye99] to use a stochastic grammar. This will allow to better simulate the repeat structure of genomes. Another possibility for generating genome-like sequences is to use for example a Markov process (which models short-range correlations fairly well) and to then apply one of the evolution simulators from Section 2 (which are able to model the complex repeat structure of a genome).

The other motivation for both tools was to be able to systematically examine the space-usage and running time of algorithms and data structures for strings, subject to statistical properties of the underlying sequence. With `tt-analyze` it is possible to measure, among others, the distribution of q-grams and the entropy of different orders for a given real world sequence. Furthermore, it is also possible to use `tt-generate` to output texts that have a predefined distribution of q-grams and entropy. Evaluating the performance of string algorithms and data structures with these sequences will make it possible, due to the very controlled conditions, to draw conclusions about the dependency on the statistical parameters.

Acknowledgments

This work was partially supported by DFG grant Ma 870/8-1 (SPP 1307: Algorithm Engineering). We thank Knut Reinert and Manuel Holtgrewe for pointing out some relevant related work, Ernst W. Mayr for discussions, and an anonymous referee for helpful suggestions.

References

[AED98] Lloyd Allison, Timothy Edgoose, and Trevor I. Dix. Compression of strings with approximate repeats. In Janice I. Glasgow, Timothy G. Littlejohn, François Major, Richard H. Lathrop, David Sankoff, and Christoph Sensen, editors, *6th International Conference on Intelligent Systems for Molecular Biology (ISMB'98)*, pages 8–16. AAAI Press, June 1998. https://www.aaai.org/Papers/ISMB/1998/ISMB98-002.pdf.

[CHG+09] Benny Chor, David Horn, Nick Goldman, Yaron Levy, and Tim Massingham. Genomic DNA k-mer spectra: models and modalities. *Genome Biology*, 10(10):R108, October 2009, http://dx.doi.org/10.1186/gb-2009-10-10-r108.

[DHH03] Manuel Dehnert, Werner E. Helm, and Marc-Thorsten Hütt. A discrete autoregressive process as a model for short-range correlations in DNA sequences. *Physica A: Statistical Mechanics and its Applications*, 327(3–4):535–553, September 2003, http://dx.doi.org/10.1016/S0378-4371(03)00399-6.

[DMBP04] Aaron C.E. Darling, Bob Mau, Frederick R. Blattner, and Nicole T. Perna. Mauve: Multiple alignment of conserved genomic sequence with rearrangements. *Genome Research*, 14(7):1394–1403, 2004, http://dx.doi.org/10.1101/gr.2289704.

[DPA+07] Trevor I. Dix, David R. Powell, Lloyd Allison, Julie Bernal, Samira Jaeger, and Linda Stern. Comparative analysis of long DNA sequences by per element information content using different contexts. *BMC Bioinformatics*, 8(Suppl 2):S10, May 2007, http://dx.doi.org/10.1186/1471-2105-8-S2-S10.

[DPHH05] Manuel Dehnert, Rainer Plaumann, Werner E. Helm, and Marc-Thorsten Hütt. Genome phylogeny based on short-range correlations in DNA sequences. *Journal of Computational Biology*, 12(5):545–553, 2005, http://dx.doi.org/10.1089/cmb.2005.12.545.

[Dun94] Ted Dunning. Statistical identification of language. Technical report, New Mexico State University, Las Cruces, NM, USA, March 1994.

[EABS09] Robert C. Edgar, George Asimenos, Serafim Batzoglou, and Arend Sidow. Evolver. Website http://www.drive5.com/evolver, 2009. http://www.drive5.com/evolver.

[FGNV09] Paolo Ferragina, Rodrigo González, Gonzalo Navarro, and Rossano Venturini. Compressed text indexes: from theory to practice. *Journal of Experimental Algorithmics*, 13:1.12–1.31, February 2009, http://dx.doi.org/10.1145/1412228.1455268.

[FPJ+05] Mehrnaz Fatemi, Martha M. Pao, Shinwu Jeong, Einav Nili Gal-Yam, Gerda Egger, Daniel J. Weisenberger, and Peter A. Jones. Footprinting of mammalian promoters: use of a CpG DNA methyltransferase revealing nucleosome positions at a single molecule level. *Nucleic Acids Research*, 33(20):e176, 2005, http://dx.doi.org/10.1093/nar/gni180.

[GHBS00] Ivo Grosse, Hanspeter Herzel, Sergey V. Buldyrev, and H. Eugene Stanley. Species independence of mutual information in coding and noncoding DNA. *Physical Review E*, 61(5):5624, 2000, http://dx.doi.org/10.1103/PhysRevE.61.5624.

[GKS03] Robert Giegerich, Stefan Kurtz, and Jens Stoye. Efficient implementation of lazy suffix trees. *Software – Practice and Experience*, 33(11):1035–1049, June 2003, http://dx.doi.org/10.1002/spe.535.

[Gus97] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, Cambridge, United Kingdom, 1997.

[HD06] Marc-Thorsten Hütt and Manuel Dehnert. *Methoden der Bioinformatik*. Springer, Berlin/Heidelberg, Germany, 2006, http://dx.doi.org/10.1007/3-540-32954-4.

[HES94] Hanspeter Herzel, Werner Ebeling, and Armin O. Schmitt. Entropies of biosequences: The role of repeats. *Physical Review E*, 50(6):5061–5071, December 1994, http://dx.doi.org/10.1103/PhysRevE.50.5061.

[HG95] Hanspeter Herzel and Ivo Große. Measuring correlations in symbol sequences. *Physica A: Statistical and Theoretical Physics*, 216(4):518–542, 1995, http://dx.doi.org/10.1016/0378-4371(95)00104-F.

[HGB+03] Dirk Holste, Ivo Grosse, Stephan Beirer, Patrick Schieg, and Hanspeter Herzel. Repeats and correlations in human DNA sequences. *Physical Review E*, 67(6):061913, 2003, http://dx.doi.org/10.1103/PhysRevE.67.061913.

[Hol10] Manuel Holtgrewe. Mason – A read simulator for second generation sequencing data. Technical Report B-10-06, Freie Universität Berlin, Berlin, Germany, October 2010. http://www.seqan.de/projects/mason.html.

[JL83] Patricia A. Jacobs and Peter A. W. Lewis. Stationary discrete autoregressive-moving average time series generated by mixtures. *Journal of Time Series Analysis*, 4(1):19–36, January 1983, http://dx.doi.org/10.1111/j.1467-9892.1983.tb00354.x.

[Li90] Wentian Li. Mutual information functions versus correlation functions. *Journal of Statistical Physics*, 60(5–6):823–837, 1990, http://dx.doi.org/10.1007/BF01025996.

[LIHL92] Gordan Lauc, Igor Ilic, and Marija Heffer-Lauc. Entropies of coding and noncoding sequences of DNA and proteins. *Biophysical Chemistry*, 42(1):7–11, 1992, http://dx.doi.org/10.1016/0301-4622(92)80002-M.

[LK92] Wentian Li and Kunihiko Kaneko. Long-range correlation and partial $1/f^{\alpha}$ spectrum in a noncoding DNA sequence. *EPL (Europhysics Letters)*, 17(7):655, 1992, http://dx.doi.org/10.1209/0295-5075/17/7/014.

[Lot97] M. Lothaire. *Combinatorics on Words*. Cambridge University Press, Cambridge, United Kingdom, 1997.

[Man01] Giovanni Manzini. An analysis of the Burrows-Wheeler transform. *Journal of the ACM*, 48(3):407–430, May 2001, http://dx.doi.org/10.1145/382780.382782.

[MBG+94] R. N. Mantegna, S. V. Buldyrev, A. L. Goldberger, S. Havlin, C. K. Peng, M. Simons, and H. E. Stanley. Linguistic features of noncoding DNA sequences. *Physical Review Letters*, 73(23):3169–3172, December 1994, http://dx.doi.org/10.1103/PhysRevLett.73.3169.

[Mye99] Gene Myers. A dataset generator for whole genome shotgun sequencing. In Thomas Lengauer, Reinhard Schneider, Peer Bork, Douglas L. Brutlag, Janice I. Glasgow, Hans-Werner Mewes, and Ralf Zimmer, editors, *7th International Conference on Intelligent Systems for Molecular Biology (ISMB'99)*, pages 202–210. AAAI, August 1999. http://www.aaai.org/Papers/ISMB/1999/ISMB99-024.pdf.

[NBY98] Gonzalo Navarro and Ricardo Baeza-Yates. A practical q-gram index for text retrieval allowing errors. *CLEI Electronic Journal*, 1(2):31–88, December 1998. http://www.clei.cl/cleiej/paper.php?id=32.

[NM07] Gonzalo Navarro and Veli Mäkinen. Compressed full-text indexes. *ACM Computing Surveys*, 39(1):2, April 2007, http://dx.doi.org/10.1145/1216370.1216372.

[ROA+08] Daniel C. Richter, Felix Ott, Alexander F. Auch, Ramona Schmid, and Daniel H. Huson. MetaSim – A sequencing simulator for genomics and metagenomics. *PLoS ONE*, 3:e3373, 10 2008, http://dx.doi.org/10.1371/journal.pone.0003373.

[SACD01] Linda Stern, Lloyd Allison, Ross L. Coppel, and Trevor I. Dix. Discovering patterns in plasmodium falciparum genomic DNA. *Molecular and Biochemical Parasitology*, 118(2):175–186, December 2001, http://dx.doi.org/10.1016/S0166-6851(01)00388-7.

[SG96] Thomas Schürmann and Peter Grassberger. Entropy estimation of symbol sequences. *CHAOS – An Interdisciplinary Journal of Nonlinear Science*, 6(3):414–427, June 1996, http://dx.doi.org/10.1063/1.166191.

[SMM+04] Hagit Shatkay, Jason Miller, Clark Mobarry, Michael Flanigan, Shibu Yooseph, and Granger Sutton. Thurgood: Evaluating assembly-to-assembly mapping. *Journal of Computational Biology*, 11(5):800–811, 2004, http://dx.doi.org/10.1089/cmb.2004.11.800.

[SPS48] Claude E. Shannon, N. Petigara, and S. Seshasai. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948. http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf.

[WGA+11] Frank Wood, Jan Gasthaus, Cédric Archambeau, Lancelot James, and Yee Whye Teh. The sequence memoizer. *Communications of the ACM*, 54:91–98, February 2011, http://dx.doi.org/10.1145/1897816.1897842.