Dense Neural Networks for Secondary Structure Analysis of Biological Sequences

Semyon Grigorev[1,2], Polina Lunina[1,2]
[1]*St. Petersburg State University, 7/9 Universitetskaya nab., St.Petersburg, Russia*
[2]*JetBrains Research, Universitetskaya emb., 7-9-11/5A, St.Petersburg, Russia*
s.v.grigoriev@spbu.ru, Semen.Grigorev@jetbrains.com, lunina_polina@mail.ru

Keywords: Dense Neural Network, DNN, Machine Learning, Secondary Structure, Formal Grammar, Parsing

Abstract: Abstract is very abstract.

1 INTRODUCTION

Accurate sequences classification and subsequences detection are an open problems in different areas of bioinformatics, such as genomics and protenomics. Challenge here is a high variability of sequences which one want to mark as a same class. For some type of sequences its secondary structure is a !!! and this fact may be used for !!!.

For example, algorithms that can efficiently and accurately identify and classify bacterial taxonomic hierarchy have become a focus in computational genetics. The idea that secondary structure of genomic sequences is sufficient for solving the detection and classification problems lies at the heart of many tools (Rivas and Eddy, 2000; Knudsen and Hein, 1999; Yuan et al., 2015; Dowell and Eddy, 2004). The secondary structure can be specified in terms of formal grammars. The sequences obtained from the real bacteria usually contain a huge number of mutations and "noise" which renders precise methods impractical. Probabilistic grammars and covariance models (CMs) are a way to take the noise into account (Durbin et al., 1998). For example, CMs are successfully used in the Infernal tool (Nawrocki and Eddy, 2013). Neural networks is another way to deal with "noisy" data. The works (Sherman, 2017; Higashi et al., 2009) utilize neural networks for 16s rRNA processing and demonstrate promising results.

In this work we propose the way to combine formal grammars and neural networks for secondary structure features processing. The key idea is does not try to model full (sub)sequence of interest by grammar, but create grammar which describes features of secondary structure and use neural network for these features processing.

2 PROPOSED SOLUTION

We propose to combine neural networks and ordinary context-free grammars (not probabilistic which are usually used in this area) in order to handle information of sequences' secondary structure. Namely, we propose to extract secondary structure features by using the ordinary context-free grammar and use the dense neural network for features processing. Features can be extracted by any parsing algorithm and then presented as a boolean matrix but we choose parsing algorithm based on matrix multiplication.

In this section we describe all components of our recipe and provide some examples end explanations on it.

2.1 Context-Free Grammars

The first component is a context-free grammar. It is a well-known fact that secondary structure of sequence may be approximated by using formal gram-

```
s1: stem<s0> any

any_str : any*[2..10]

s0: any_str | any_str stem<s0> s0

any: A | T | C | G

stem1<s>: A s T | G s C | T s A | C s G

stem2<s>: stem1< stem1<s> >

stem<s>:
      A stem<s> T
    | T stem<s> A
    | C stem<s> G
    | G stem<s> C
    | stem1< stem2<s> >
 }
```

Figure 1: Context-free grammar G_0 for RNA secondary structure features extraction

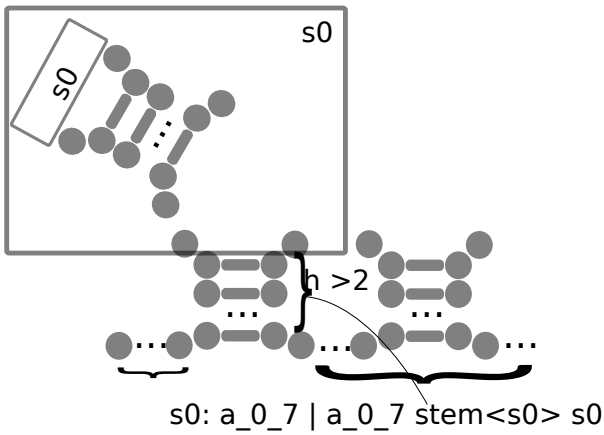

Figure 2: Graphical explanation of pattern which is described by grammar in the figure 1

mars. There is number of works that utilize this fact for different purposes (**?**).

Usually probabilistic context-free grammars are used for secondary structure modeling because it allows to dealt with variations (mutations or some kinds of noise). In the opposite of it, we use ordinary (not probabilistic) grammars. Our goal is not to model secondary structure of whole sequence (which required probabilistic grammars), but describe features of secondary structure, such as stems, loops, pseudoknots and it's composition. The set of feature types is limited by class of the grammar which we use. For example, pseudoknots can not be expressed by context-free grammars, but can be expresed by using cinjunctive (Devi and Arumugam, 2017; Zier-Vogel and Domaratzki, 2013; Okhotin, 2001) or multiple context-free (Seki et al., 1991; Riechert et al., 2016).

The context-free grammar G_0 which we use in our experiments is presented in figure 1. It is a context-free grammar over four-letters in the alphabet $\Sigma = \{A,C,G,T\}$ with start nonterminal `s1`. This grammar which describes composition of stems with bounded minimal height.

First of all, we provide a brief description of grammar specification language. Left hand side and right hand side of rule are separated by the : sign. In the right hand side one can use extended regular expressions over union alphabet of terminals and nonterminals. Such constructions as bounded repetition and alternative are available. For example, `any*[2..10]` is a bounded repetition and it is stands that nonterminal `any` may be repeated any number of times from 2 up to 10. Example of rule which uses alternatives is `any: A | T | C | G` which stands that `any` is one of four terminals.

Another important feature of the language is parametric rules or metarules which allow one to create a reusable grammar templates. More details on metarules one can find in (**?**). The example of metarule in our grammar is `stem1<s>: A s T | G s C | T s A | C s G`. This rule has one parameter `s` which stands for something that should be embedded into stem. Application of this rule to `any_str` allow one to define stem with loop of length from 2 up to 10. In our grammar we use metarules in order to describe stems with bounded minimal height: `stem1<s>` is a stem with height exactly 1, `stem2<s>` is a stem with height exactly 2, and `stem<s>` is a stem with height not lower than 3.

Now we explain what does this grammar meens. This grammar describe a recursive composition of stems. To see it one can look at the rule for `s0` which is recursive and shows that composition of stems may be embedded into stem (`|stem<s0>|` in the right side of this rule). Every stem should has height not lower then 3 and builds only from classical base pairs. Stems may be connected by arbitrary sequence of length from 2 up to 10, and loops have the same length. Graphical explanation of this description one can find in figure 2.

Note that grammar is a variable parameter and may be tuned for specific cases. For example, one can vary length of unfoldable sequence by changing rule for `any_str`: `any_str : any*[0..10]`, `any_str : any*[1..8]`, or something else. Also one can increase (or decrease for some reason) the minimal height of stem, or add some new features, such as pseudoknots, in the grammar (in case of usage of conjunctive grammars instead of context-free

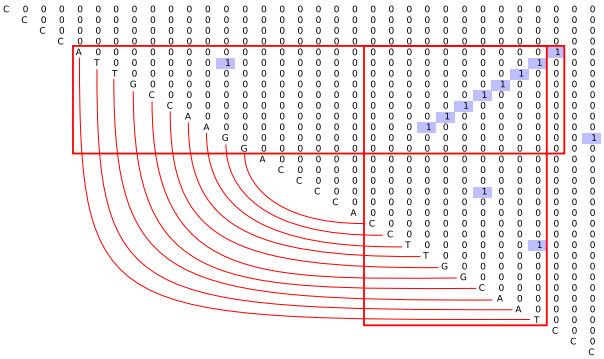

Figure 3: Parsing result for sequence which should folds to stem

one).

2.2 Parsing Algorithm

Parsing is a feature extraction, so undirected parsing: we want to find all derivable substrings of given string for all nonterminals, not to check derivability of given string or find the most probably derivation.

CYK — as a classical well-known algorithm.

Matrices.

Valiant (Valiant, 1975) — subcubic algorithm based on matrix multiplication.

Rustam (Azimov and Grigorev, 2018) — generalization for graph. Theoretical time complexity is !!! than complexity of the Valiant's algorithm, but in practice these algorithm avoid machinery on submatrices manipulation and demonstrate better performance with simple implementation.

Sparse matrices, boolean, GPGPU, etc.

Matrix-based approach can be generalized to conjunctive and even boolean grammars (Okhotin, 2014), as far as to multiple context-free grammars (Cohen and Gildea, 2016), which can provide a base for more expressive features descriptions handling.

2.3 Matrices

The result of parsing is a set of square boolean matrices. Each matrix M_N contains information of all substrings which can be derived from nontermnal N. In the other word, $M_N[i,j] = 1$ iff $N \Rightarrow^*_G w[i,j-1]$ where w is the input sequence and G is context-free grammar, and N is a nonterminal. Thus, result of parsing is a set of matrices: one matrix for each nonterminal form grammar. For futher processing we can select nonterminals of interest. For our case, for grammar G_0 we select matrix for nonterminal `s1`.

The example of such matrix is provided in figure 3. This matrix is a result of parsing of the sequence

$w =$`CCCCATTGCCAAGGACCCCACCTTGGCAATCCC`

w.r.t the grammar G_0. In the figure one can see upper right triangle of parsing matrix (bottom left is always empty, so omitted) with input string on te diagonal. Note that string is added only for example readability and real matrix does not contains input string, only results of its parsing. Each filled cell $[i,j]$ which contains 1 is denote that subsequence $w[i,j-1]$ is derivable form `s1` in G_0 (so, this subsequence folds to stem with heigh 3 or more). In order to find stems with heght more than 3 one should detect diagonal chains of 1-s: in our example stem has height equals 10 and one can find chain of 1-s of the length $8 = 10-2$ (first 1 is a root of the stem of heght 3 and each next 1 is a new basepair upon this stem — root of the stem with height increased by one). Red boxes and contact map are added for navigation sinplification.

Our goal is to extract all features of secondary structure, so our parser finds all substrings which can be derived from `s1`. As a result there are some 1-s out of the chain. These are correct results: corresponded subsequences can be derived from `s1`. In the current exmaple these 1-s may be treated as noise in some sense, but as we show late such behaviour may be useful in some cases.

We use these matrices as an input for artifactal newural network which should detect sufficient features (long chain in our example) and utilize these for applied problem solution (sequence detection or classification, for example). We drop out bottom left triangle and vectorize matrices row-by-row in order to get bit vector which then converts to byte vector and uses as an input. Transition from bit vector to byte vector is done in order to decrease sie of the input which is critical for long sequences. On the other hand, such operation may significantly complicate network architecture and training, and it is a reason to try to use bitwise networks (Kim and Smaragdis, 2016) in the future.

2.4 Neural Networks

One of possible choice for classification. Classical scenario is to provide features vectors and try to classify them which meens that network can select important features for each class. In our case ht fact that "$w[i,j-1]$ is derivable from nonterminal N" which is encoded in the matrix is a feature. So, matrix is a set (or vector) of features.

We use dense neural network because locality is broken during vectorization and any convolutions is inapplicable.

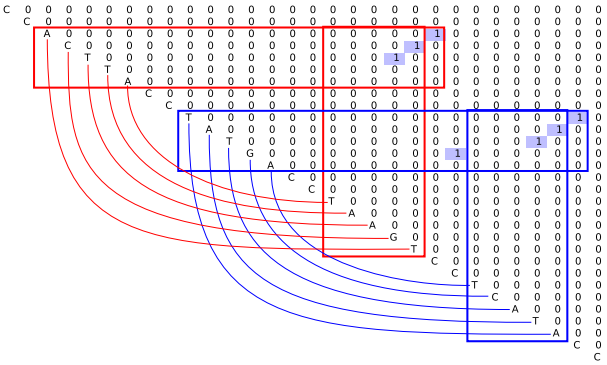

Figure 4: Parsing result for sequence which should folds to pseudoknot

Current architecture and its motivation and explanation. Huge dropout and batch normalization.

2.5 Examples

Here we provide more examples of matrices and point out some observations about it.

First is observation about pseudoknots. Sequence: CCACTTACCTATGACCTAAGTCCTCATACC As mentioned above, pseudoknots can not be expressed in terms of context-free grammars. But Blue and red If newural network is powerful enough to !!! that these two fwetures should occurce simulatenously, then thay can detect pseudocnots.

The second is an example of matrix for real tRNA. Sequence: CAGGGCATAACCTAGCCCAACCTTGCCAAGGTTGGGGTCGAGGGTTCGAATCCCTTCGCCCGCTCCA Spec: Novosphingobium_aromaticivorans_DSM_12444_chr.trna57-GlyGCC (268150-268084) Gly (GCC) 67 bp Sc: 22.97

Result of folding by tool!!!

Matrix contains all these variants. Blue and red boxes. And some additional.

Nontrivial compositions may be detected. May we use NNs for secondary structure prediction?

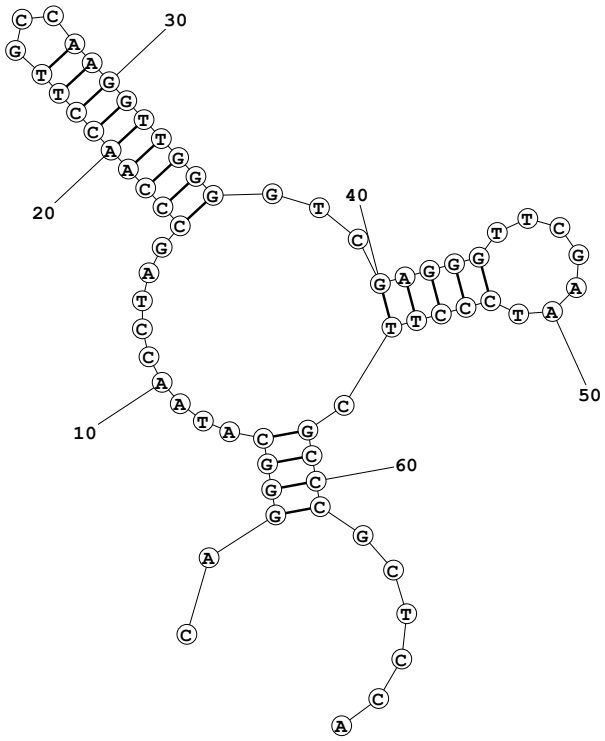

Figure 5: !!!!!

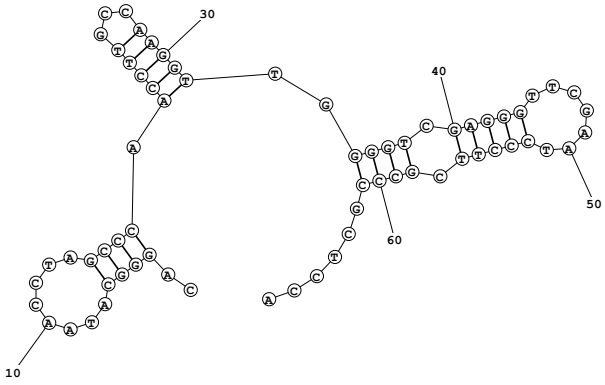

Figure 6: !!!!!

3 EVALUATION

Two cases. 16s rRNA and tRNA.

We evaluate the proposed approach for 16s rRNA detection. We specify context-free grammars which detect stems with the hight of more than two pairs and their arbitrary compositions. For network training we use dataset consisting of two parts: random subsequences of 16s rRNA sequences from the Green Genes database (DeSantis et al., 2006) form positive examples, while the negative examples are random subsequences of full genes from the NCBI database (Geer et al., 2010). All sequences have the length of 512 symbols, totally up to 310000 sequences. After training, current accuracy is 90% for validation set (up to 81000 sequences), thus we conclude that our approach is applicable.

Description of evaluation on tRNA classification. Procariot — eucarion. 500000 sequences. Upper bound of length. Results!!!

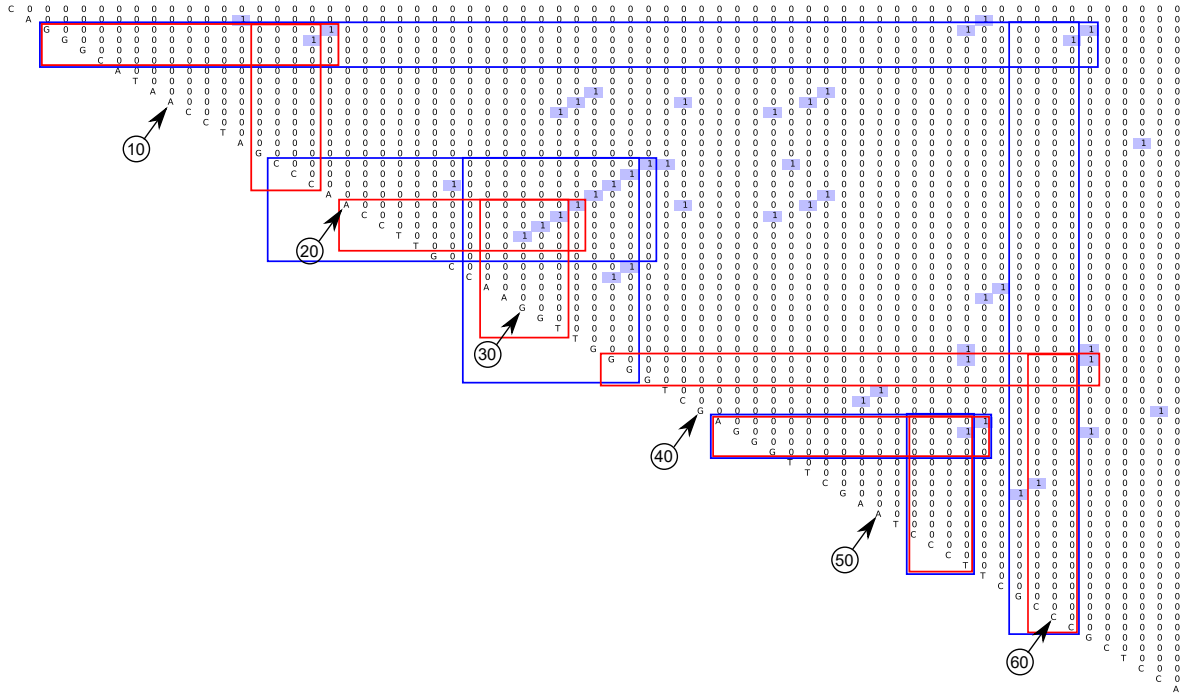

Figure 7: !!!!!

4 FUTURE WORK

The presented is a work in progress. The ongoing experiment is finding all instances of 16s rRNA in full genomes. Also we plan to use the proposed approach for the filtration of chimeric sequences and the classification. Composition of our approach with other methods and tools as well as grammar tuning and detailed performance evaluation may improve the applicability for the real data processing.

5 DISCUSSION

Protenomics (Witold Dyrka) (Dyrka et al., 2018) More complex grammar: more symbols in alphabet, more complex featres. More powerful languages required. One of the possible crutial problem is functionally equivalence sequences with different length in protenomics.

Different lengths. Is a problem. How can we normalize input?

Construct network which can handle sequences, not parsing data. It may help to create an embedding. It may be done by the next way.

1. Build and train the network which handle vectorized matrices.
2. Extend this network with head which should convert sequence to !!!
3. Train. Weights of first network is fixed.
4. For concrete problem we can tune weights for full network after second trained to appropriate quality.

Other types of NNs. Binary, convolutional (try to process matrix as a picture). Pictures: problem with size, typical matrix size is big.

Prblems with data: how to create balanced set for training. Datasets (like GreenGenes) contains huge nomder of samples for some well-studied organisms and very small number of samples for other.

Huge amount of experiments in different directions. Plans should be discussed with community.

ACKNOWLEDGEMENTS

The research was supported by the Russian Science Foundation grant 18-11-00100 and a grant from JetBrains Research.

REFERENCES

Azimov, R. and Grigorev, S. (2018). Context-free path querying by matrix multiplication. In *Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, GRADES-NDA '18, pages 5:1–5:10, New York, NY, USA. ACM.

Cohen, S. B. and Gildea, D. (2016). Parsing linear context-free rewriting systems with fast matrix multiplication. *Computational Linguistics*, 42(3):421–455.

DeSantis, T. Z., Hugenholtz, P., Larsen, N., Rojas, M., Brodie, E. L., Keller, K., Huber, T., Dalevi, D., Hu, P., and Andersen, G. L. (2006). Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. *Appl. Environ. Microbiol.*, 72(7):5069–5072.

Devi, K. K. and Arumugam, S. (2017). Probabilistic conjunctive grammar. In *Theoretical Computer Science and Discrete Mathematics*, pages 119–127. Springer International Publishing.

Dowell, R. D. and Eddy, S. R. (2004). Evaluation of several lightweight stochastic context-free grammars for rna secondary structure prediction. *BMC bioinformatics*, 5(1):71.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press.

Dyrka, W., Coste, F., and Talibart, J. (2018). Estimating probabilistic context-free grammars for proteins using contact map constraints. *CoRR*, abs/1805.08630.

Geer, L. Y., Marchler-Bauer, A., Geer, R. C., Han, L., He, J., He, S., Liu, C., Shi, W., and Bryant, S. H. (2010). The NCBI BioSystems database. *Nucleic Acids Res.*, 38(Database issue):D492–496.

Higashi, S., Hungria, M., and Brunetto, M. (2009). Bacteria classification based on 16s ribosomal gene using artificial neural networks. In *Proceedings of the 8th WSEAS International Conference on Computational intelligence, man-machine systems and cybernetics*, pages 86–91.

Kim, M. and Smaragdis, P. (2016). Bitwise neural networks. *CoRR*, abs/1601.06071.

Knudsen, B. and Hein, J. (1999). Rna secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics (Oxford, England)*, 15(6):446–454.

Nawrocki, E. P. and Eddy, S. R. (2013). Infernal 1.1: 100-fold faster RNA homology searches. *Bioinformatics*, 29(22):2933–2935.

Okhotin, A. (2001). Conjunctive grammars. *J. Autom. Lang. Comb.*, 6(4):519–535.

Okhotin, A. (2014). Parsing by matrix multiplication generalized to boolean grammars. *Theoretical Computer Science*, 516:101 – 120.

Riechert, M., Höner zu Siederdissen, C., and Stadler, P. F. (2016). Algebraic dynamic programming for multiple context-free grammars. *Theor. Comput. Sci.*, 639(C):91–109.

Rivas, E. and Eddy, S. R. (2000). The language of rna: a formal grammar that includes pseudoknots. *Bioinformatics*, 16(4):334–340.

Seki, H., Matsumura, T., Fujii, M., and Kasami, T. (1991). On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191 – 229.

Sherman, D. (2017). Humidor: Microbial community classification of the 16s gene by training cigar strings with convolutional neural networks.

Valiant, L. G. (1975). General context-free recognition in less than cubic time. *J. Comput. Syst. Sci.*, 10(2):308–315.

Yuan, C., Lei, J., Cole, J., and Sun, Y. (2015). Reconstructing 16s rrna genes in metagenomic data. *Bioinformatics*, 31(12):i35–i43.

Zier-Vogel, R. and Domaratzki, M. (2013). Rna pseudoknot prediction through stochastic conjunctive grammars. *Computability in Europe 2013. Informal Proceedings*, pages 80–89.