



YaccConstructor

Задачи на весенний семестр 2019

Семён Григорьев

Лаборатория языковых инструментов JetBrains
Санкт-Петербургский государственный университет
Математико-механический факультет

8 июня 2019г.

- Исследовательская группа на Математико-Механическом факультете СПбГУ
- Исследовательская группа в лаборатории языковых инструментов JetBrains Research
- Руководитель группы: Семён Григорьев
 - ▶ rsdpisuy@gmail.com
 - ▶ semyon.grigorev@jetbrains.com
 - ▶ <https://research.jetbrains.org/researchers/gsv>

- Исследовательская группа на Математико-Механическом факультете СПбГУ
- Исследовательская группа в лаборатории языковых инструментов JetBrains Research
- Руководитель группы: Семён Григорьев
 - ▶ rsdpisuy@gmail.com
 - ▶ semyon.grigorev@jetbrains.com
 - ▶ <https://research.jetbrains.org/researchers/gsv>
- Сферы интересов
 - ▶ Теория формальных языков
 - ▶ **Применение теории формальных языков для решения прикладных задач**

Анализ вторичной структуры: синтаксический анализ + искусственные нейронные сети

- Формальная грамматика — способ описать особенности вторичной структуры
 - ▶ А не смоделировать структуру всей цепочки
 - ▶ Используем обыкновенные грамматики, а не вероятностные
- Синтаксический анализ — способ извлечь особенности вторичной структуры
- Искусственная нейронная сеть — вероятностная модель для обработки извлечённых особенностей

```
s1: stem<s0>
any_str: any_smb*[2..10]
any_smb: A | T | C | G
stem1<s>:          \\ stem of height exactly 1
             A s T | T s A | C s G | G s C
stem3<s>:          \\ stem of height exactly 3
             stem1< stem1< stem1<s> > >
stem<s>:           \\ stem of height 3 or more
             A stem<s> T
             | T stem<s> A
             | C stem<s> G
             | G stem<s> C
             | stem3<s>
s0: any_str | any_str stem<s0> s0
```

```
s1: stem<s0>
```

```
any_str: any_smb*[2..10]
```

```
any_smb: A | T | C | G
```

```
stem1<s>:                \\ stem of height exactly 1  
    A s T | T s A | C s G | G s C
```

```
stem3<s>:                \\ stem of height exactly 3  
    stem1< stem1< stem1<s> > >
```

```
stem<s>:                \\ stem of height 3 or more  
    A stem<s> T  
    | T stem<s> A  
    | C stem<s> G  
    | G stem<s> C  
    | stem3<s>
```

```
s0: any_str | any_str stem<s0> s0
```

```
s1: stem<s0>
```

```
any_str: any_smb*[2..10]
```

```
any_smb: A | T | C | G
```

```
stem1<s>:          \\ stem of height exactly 1
```

```
    A s T | T s A | C s G | G s C
```

```
stem3<s>:          \\ stem of height exactly 3
```

```
    stem1< stem1< stem1<s> > >
```

```
stem<s>:           \\ stem of height 3 or more
```

```
    A stem<s> T
```

```
    | T stem<s> A
```

```
    | C stem<s> G
```

```
    | G stem<s> C
```

```
    | stem3<s>
```

```
s0: any_str | any_str stem<s0> s0
```

s1: stem<s0>

any_str: any_smb*[2..10]

any_smb: A | T | C | G

stem1<s>: \\ stem of height exactly 1
 A s T | T s A | C s G | G s C

stem3<s>: \\ stem of height exactly 3
 stem1< stem1< stem1<s> > >

stem<s>: \\ stem of height 3 or more
 A stem<s> T
 | T stem<s> A
 | C stem<s> G
 | G stem<s> C
 | stem3<s>

s0: any_str | any_str stem<s0> s0

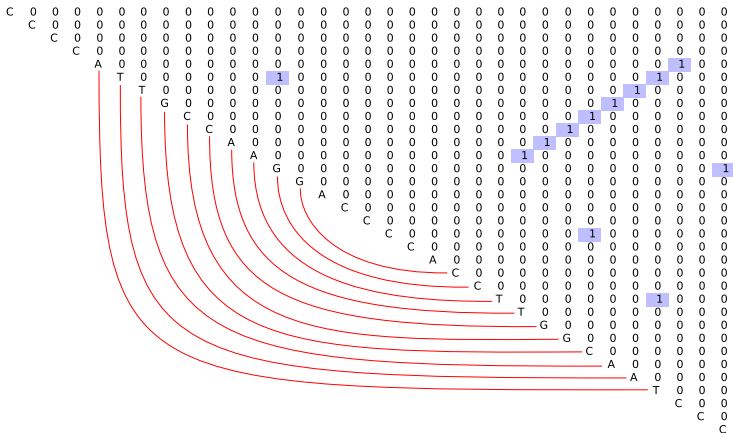

```
s1: stem<s0>
any_str: any_smb*[2..10]
any_smb: A | T | C | G
stem1<s>:          \\ stem of height exactly 1
             A s T | T s A | C s G | G s C
stem3<s>:          \\ stem of height exactly 3
             stem1< stem1< stem1<s> > >
stem<s>:          \\ stem of height 3 or more
             A stem<s> T
             | T stem<s> A
             | C stem<s> G
             | G stem<s> C
             | stem3<s>
s0: any_str | any_str stem<s0> s0
```

```
s1: stem<s0>
any_str: any_smb*[2..10]
any_smb: A | T | C | G
stem1<s>:          \\ stem of height exactly 1
             A s T | T s A | C s G | G s C
stem3<s>:          \\ stem of height exactly 3
             stem1< stem1< stem1<s> > >
stem<s>:          \\ stem of height 3 or more
             A stem<s> T
             | T stem<s> A
             | C stem<s> G
             | G stem<s> C
             | stem3<s>
s0: any_str | any_str stem<s0> s0
```

```
s1: stem<s0>
any_str: any_smb*[2..10]
any_smb: A | T | C | G
stem1<s>:          \\ stem of height exactly 1
               A s T | T s A | C s G | G s C
stem3<s>:          \\ stem of height exactly 3
               stem1< stem1< stem1<s> > >
stem<s>:          \\ stem of height 3 or more
               A stem<s> T
               | T stem<s> A
               | C stem<s> G
               | G stem<s> C
               | stem3<s>
s0: any_str | any_str stem<s0> s0
```

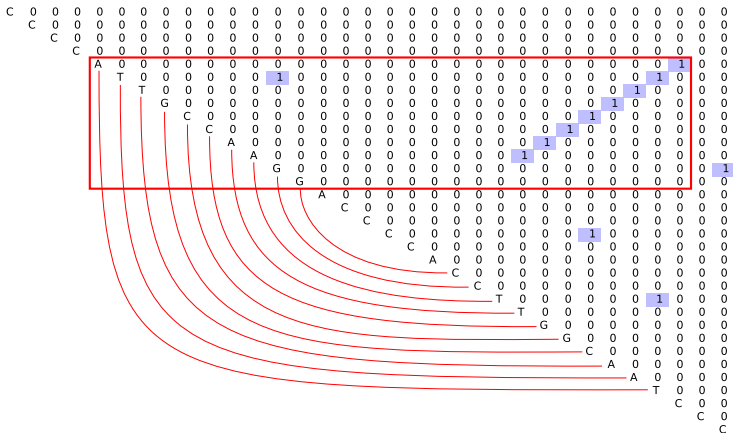
Пример 1: Stem

CCCCATTGCCAAGGACCCCACTTGGCAATCCC



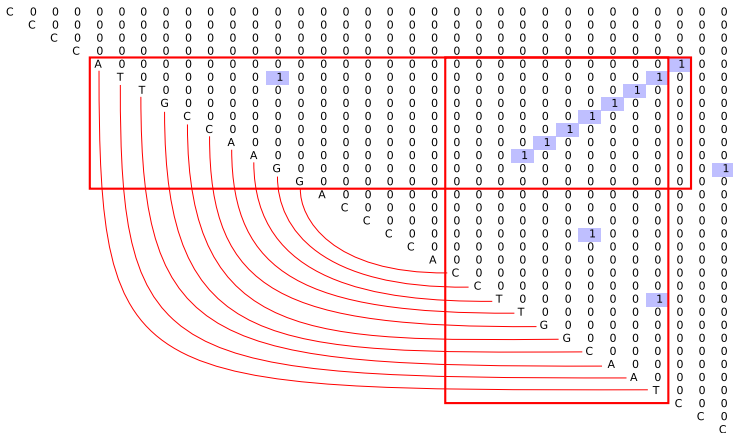
Пример 1: Stem

CCCCATTGCCAAGGACCCACCTTGGCAATCCC



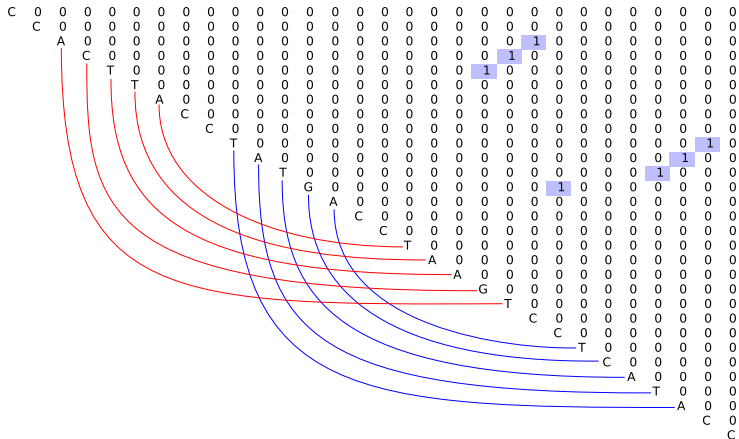
Пример 1: Stem

CCCCATTGCCAAGGACCCACCTTGGCAATCCC



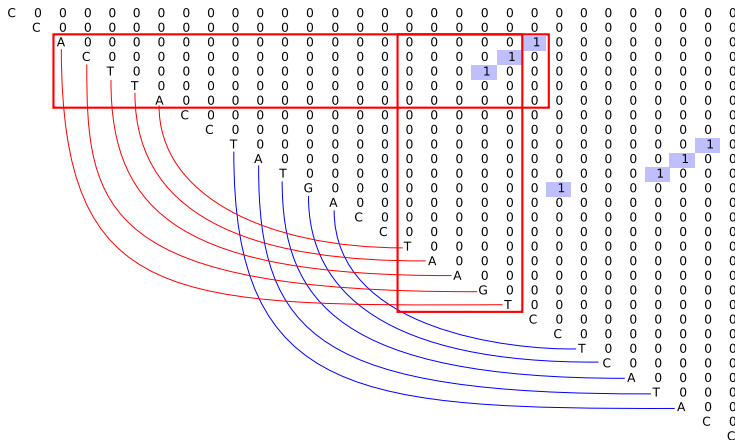
Пример 2: псевдоузел

CCACTTACCTATGACCTAAGTCCTCATACC



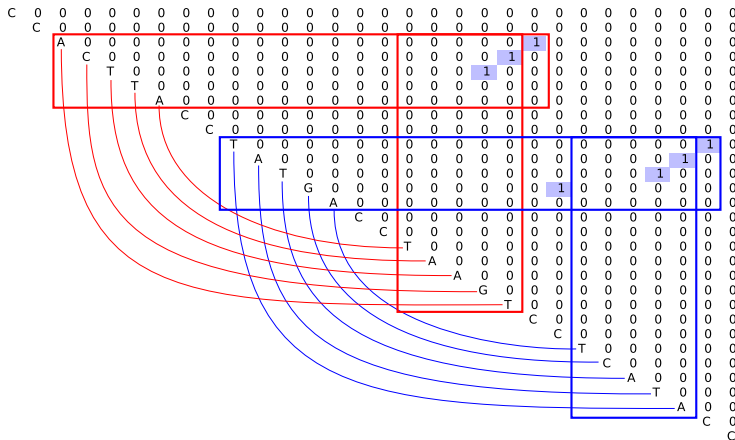
Пример 2: псевдоузел

CCACTTACCTATGACC TAAGT CCTCATACC



Пример 2: псевдоузел

CCACTTACCTATGACC TAAGT CCTCATACC



- Анализ вторичной структуры РНК-последовательностей
 - ▶ Фильтрация химер 16s рРНК
 - ▶ Предсказание вторичной структуры цепочек
 - ▶ Классификация 16s рРНК

- Анализ вторичной структуры РНК-последовательностей
 - ▶ Фильтрация химер 16s рРНК
 - ▶ Предсказание вторичной структуры цепочек
 - ▶ Классификация 16s рРНК
- Анализ вторичной структуры белковых последовательностей
 - ▶ Предсказание функций белков

Требования к кандидатам

- Знания в геномике/протеомике: первичная и вторичная структуры, задачи поиска и классификации последовательностей, умение работать с базами соответствующих последовательностей
- Базовые знания теории формальных языков: формальная грамматика, вывод в грамматике, синтаксический анализ
- Хорошие знания в машинном обучении: подготовка и проведение экспериментов, подготовка данных, оценка качества моделей
- Хорошие знания в искусственных нейронных сетях: прикладные пакеты для создания и обучения, принципы построения сетей, смысл параметров и метапараметров

- 1 месяц, предпочтительно июль
- Можно удалённо, но с регулярным общением с руководителем
- Стипендия

Evaluation: 16s rRNA detection

- Training data
 - ▶ All sequences are 512 symbols in length
 - ▶ Totally up to 310000 sequences
 - ▶ Positive: random subsequences of 16s rRNA sequences from the Green Genes database
 - ▶ Negative: random subsequences of full genes from the NCBI database
- Validation set: up to 81000 sequences
- Accuracy is 90% after training

Evaluation: tRNA classification

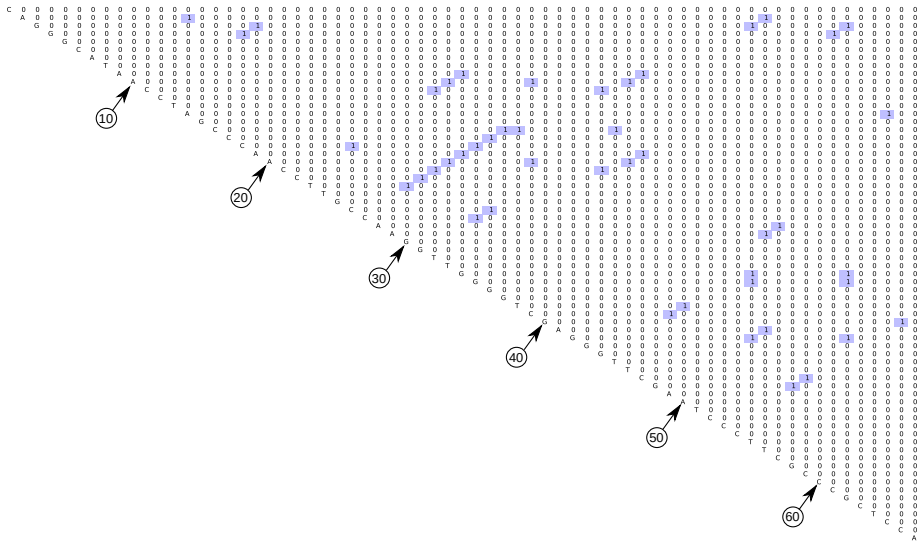
- Training data: 50000 sequences from GtRNADB
- Input data normalization
 - ▶ Set the upper bound of sequence length to 220
 - ▶ First k symbols of the input are tRNA and the rest $220 - k$ symbols are filled by the special symbol
- Validation set: 217984 sequences for prokaryotes and 62656 sequences for eukaryotes from tRNADB-CE 3
- Accuracy is 97% after training
 - ▶ 3276 of eukaryotes (5.23% of all eukaryotes in the validation set) are classified as prokaryotes
 - ▶ 4373 of prokaryotes (2.01% of all prokaryotes in the validation set) are classified as eukaryotes

Example 3: real tRNA

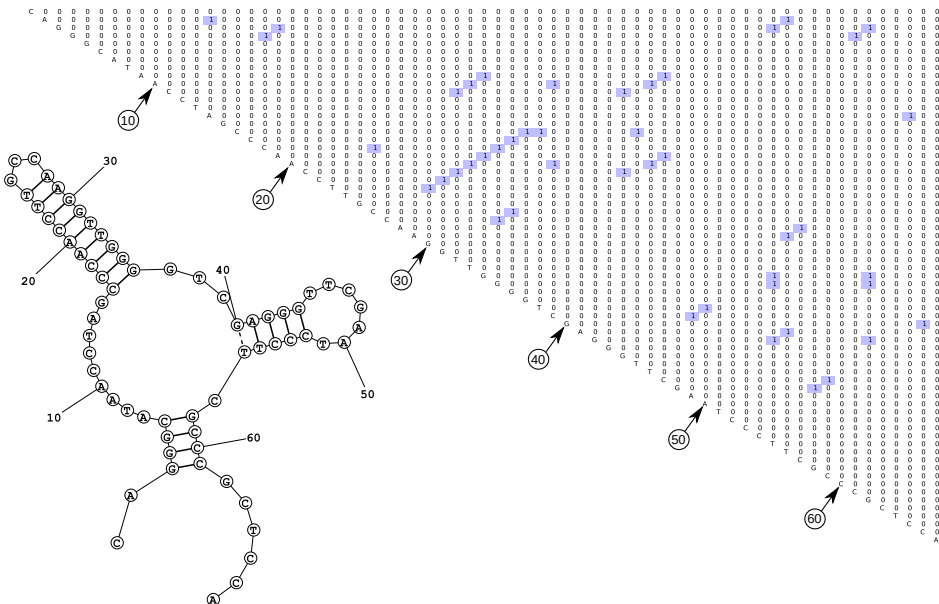
```
CAGGGCATAACCTAGCCCAACCTTGCCAAGG  
TTGGGGTCGAGGGTTCGAATCCCTTCGCCCGCTCCA
```

- *Novosphingobium aromaticivorans* DSM 12444
chr.trna57-GlyGCC(268150-268084) Gly (GCC) 67 bp Sc: 22.9, from GtRNAb
- Predicted secondary structures are given by using the Fold Web Server with default settings

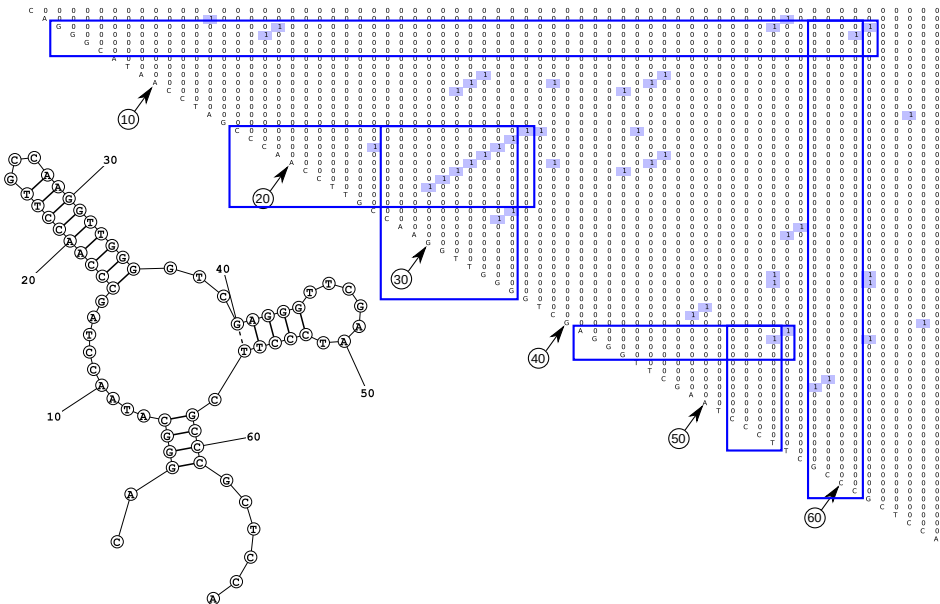
Example 3: real tRNA



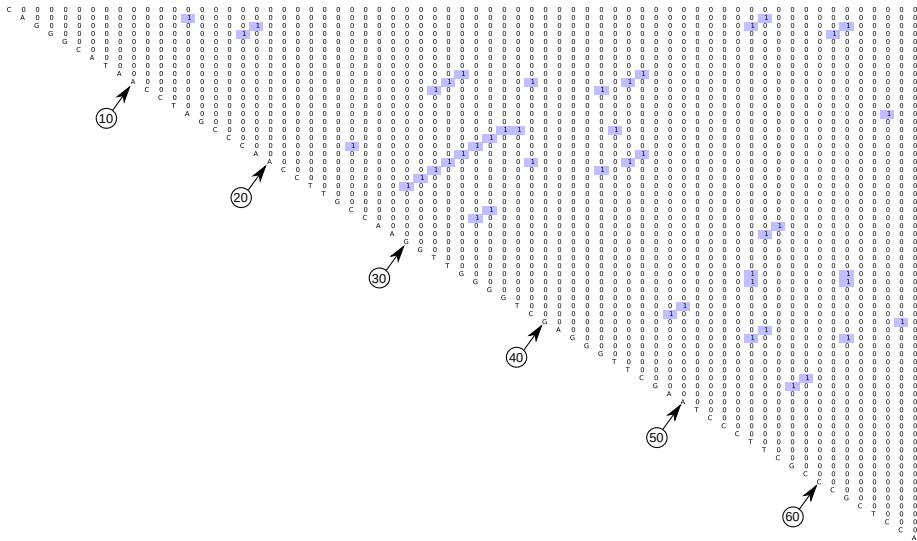
Example 3: real tRNA



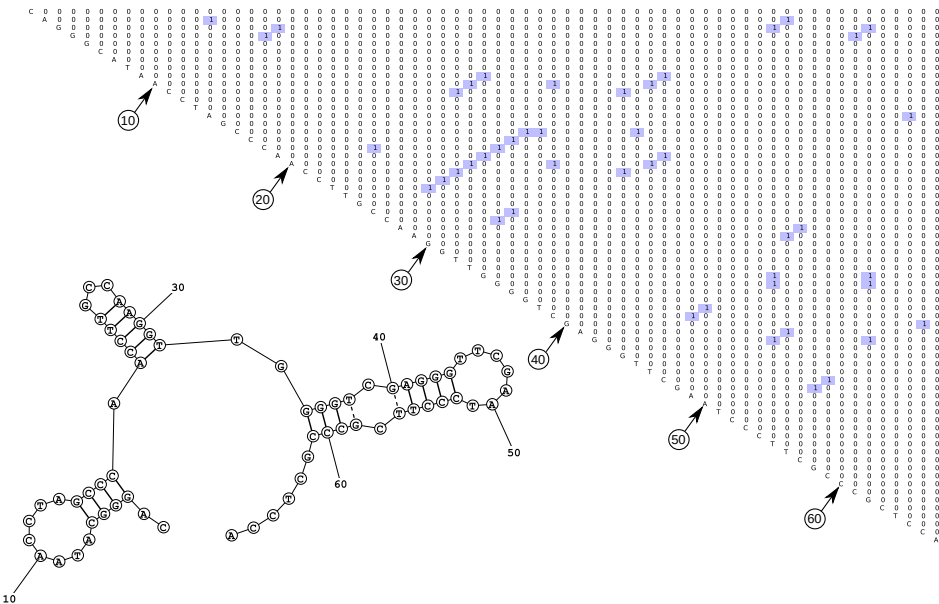
Example 3: real tRNA



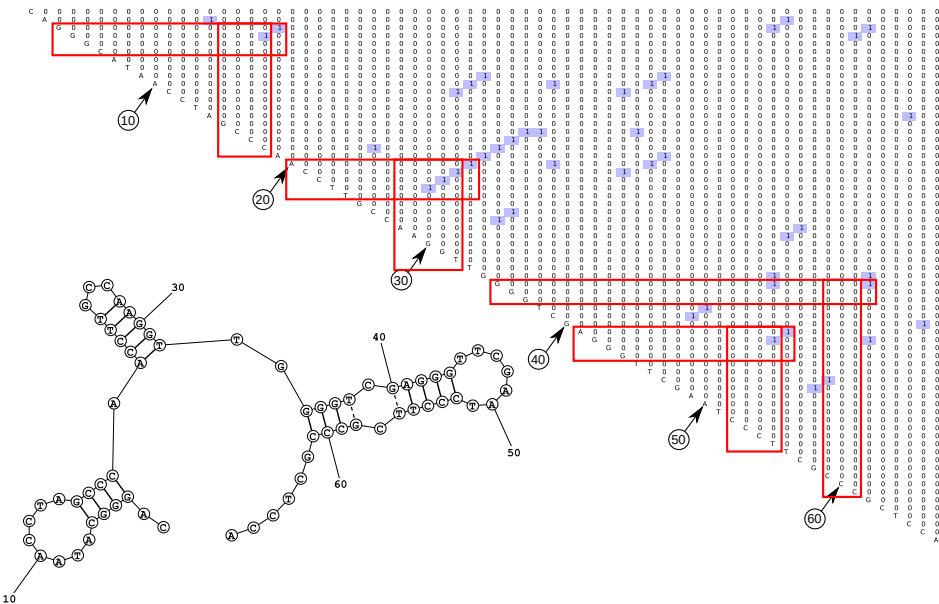
Example 3: real tRNA



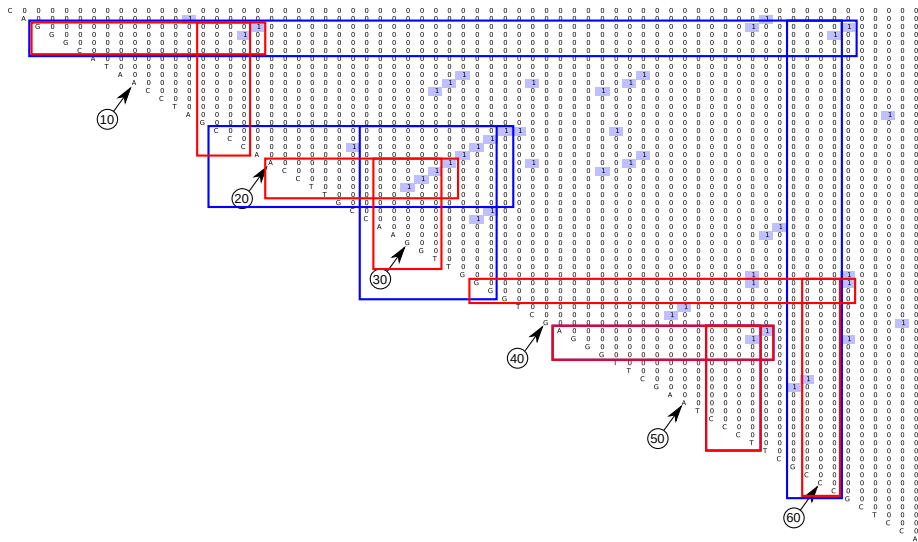
Example 3: real tRNA



Example 3: real tRNA



Example 3: real tRNA



Solution Structure

Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure and can be tuned to increase the quality of result.

Sequences

Each sequence is treated as a text in $\{A, C, G, T\}$ alphabet.

Result of classification

Parser

Parser extracts features of the given sequence secondary structure. Implementation of parsing algorithm is based on matrix multiplications (Valiant, Okhotin) and utilizes GPGPU.

Neural Network

Dense neural network with more than 10 dense layers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:

Dropout (75%)	input: 1024
	output: 1024

Dense	input: 1024
	output: 1024

BatchNormalization	input: 1024
	output: 1024

Activation (relu)	input: 1024
	output: 1024

Matrices

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Parsing result is (0-1) matrix M which represents secondary structure features for sequence ω :

$$M[i, j] = 1 \iff s1 \xrightarrow{*} \omega[i, j], \text{ and } 0 \text{ otherwise.}$$

Vectors

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\downarrow$$
$$[0, 1, 0, 1, 0, 1, 0, 0, 1, 0]$$

$$\downarrow$$
$$[84, 128]$$

Line-by-line compressed matrix representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so can be ignored.

Solution Structure

Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure and can be tuned to increase the quality of result.

Sequences

Each sequence is treated as a text in $\{A, C, G, T\}$ alphabet.

Result of classification

Parser

Parser extracts features of the given sequence secondary structure. Implementation of parsing algorithm is based on matrix multiplications (Valiant, Okhotin) and utilizes GPGPU.

Neural Network

Dense neural network with more than 10 dense layers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:

Dropout (75%)	input: 1024
	output: 1024

Dense	input: 1024
	output: 1024

BatchNormalization	input: 1024
	output: 1024

Activation (relu)	input: 1024
	output: 1024

Matrices

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Parsing result is (0-1) matrix M which represents secondary structure features for sequence ω :

$$M[i, j] = 1 \iff s1 \xrightarrow{*} \omega[i, j], \text{ and } 0 \text{ otherwise.}$$

Vectors

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\downarrow$$
$$[0, 1, 0, 1, 0, 1, 0, 0, 1, 0]$$

$$\downarrow$$
$$[84, 128]$$

Line-by-line compressed matrix representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so can be ignored.

Solution Structure

Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure and can be tuned to increase the quality of result.

Sequences

Each sequence is treated as a text in $\{A, C, G, T\}$ alphabet.

Result of classification

Parser

Parser extracts features of the given sequence secondary structure. Implementation of parsing algorithm is based on matrix multiplications (Valiant, Okhotin) and utilizes GPGPU.

Matrices

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Parsing result is (0-1) matrix M which represents secondary structure features for sequence ω :

$$M[i, j] = 1 \iff s1 \xrightarrow{*} \omega[i, j], \text{ and } 0 \text{ otherwise.}$$

Vectors

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\downarrow$$
$$[0, 1, 0, 1, 0, 1, 0, 0, 1, 0]$$

$$\downarrow$$
$$[84, 128]$$

Line-by-line compressed matrix representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so can be ignored.

Neural Network

Dense neural network with more than 10 dense layers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:

Dropout (75%)	input: 1024
	output: 1024

Dense	input: 1024
	output: 1024

BatchNormalization	input: 1024
	output: 1024

Activation (relu)	input: 1024
	output: 1024

Solution Structure

Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure and can be tuned to increase the quality of result.

Sequences

Each sequence is treated as a text in $\{A, C, G, T\}$ alphabet.

Result of classification

Parser

Parser extracts features of the given sequence secondary structure. Implementation of parsing algorithm is based on matrix multiplications (Valiant, Okhotin) and utilizes GPGPU.

Neural Network

Dense neural network with more than 10 dense layers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:

Dropout (75%)	input: 1024
	output: 1024

Dense	input: 1024
	output: 1024

BatchNormalization	input: 1024
	output: 1024

Activation (relu)	input: 1024
	output: 1024

Matrices

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Parsing result is (0-1) matrix M which represents secondary structure features for sequence ω :

$$M[i, j] = 1 \iff s1 \xrightarrow{*} \omega[i, j], \text{ and } 0 \text{ otherwise.}$$

Vectors

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$[0, 1, 0, 1, 0, 1, 0, 0, 1, 0]$$

$$[84, 128]$$

Line-by-line compressed matrix representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so can be ignored.

Solution Structure

Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure and can be tuned to increase the quality of result.

Sequences

Each sequence is treated as a text in $\{A, C, G, T\}$ alphabet.

Result of classification

Parser

Parser extracts features of the given sequence secondary structure. Implementation of parsing algorithm is based on matrix multiplications (Valiant, Okhotin) and utilizes GPGPU.

Neural Network

Dense neural network with more than 10 dense layers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:

Dropout (75%)	input: 1024 output: 1024
------------------	-----------------------------

Dense	input: 1024 output: 1024
-------	-----------------------------

BatchNormalization	input: 1024 output: 1024
--------------------	-----------------------------

Activation (relu)	input: 1024 output: 1024
----------------------	-----------------------------

Matrices

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Parsing result is (0-1) matrix M which represents secondary structure features for sequence ω :

$$M[i, j] = 1 \iff s1 \xrightarrow{*} \omega[i, j], \text{ and } 0 \text{ otherwise.}$$

Vectors

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$



$[0, 1, 0, 1, 0, 1, 0, 0, 1, 0]$



$[84, 128]$

Line-by-line compressed matrix representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so can be ignored.

Solution Structure

Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure and can be tuned to increase the quality of result.

Sequences

Each sequence is treated as a text in $\{A, C, G, T\}$ alphabet.

Result of classification

Parser

Parser extracts features of the given sequence secondary structure. Implementation of parsing algorithm is based on matrix multiplications (Valiant, Okhotin) and utilizes GPGPU.

Neural Network

Dense neural network with more than 10 dense layers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:

Dropout (75%)	input: 1024
	output: 1024

Dense	input: 1024
	output: 1024

BatchNormalization	input: 1024
	output: 1024

Activation (relu)	input: 1024
	output: 1024

Matrices

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Parsing result is (0-1) matrix M which represents secondary structure features for sequence ω :

$$M[i, j] = 1 \iff s1 \xrightarrow{*} \omega[i, j], \text{ and } 0 \text{ otherwise.}$$

Vectors

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\downarrow$$
$$[0, 1, 0, 1, 0, 1, 0, 0, 1, 0]$$

$$\downarrow$$
$$[84, 128]$$

Line-by-line compressed matrix representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so can be ignored.

Solution Structure

Grammar

Fixed formal grammar (not necessarily context-free) describes features of secondary structure and can be tuned to increase the quality of result.

Sequences

Each sequence is treated as a text in $\{A, C, G, T\}$ alphabet.

Result of classification

Parser

Parser extracts features of the given sequence secondary structure. Implementation of parsing algorithm is based on matrix multiplications (Valiant, Okhotin) and utilizes GPGPU.

Neural Network

Dense neural network with more than 10 dense layers. Aggressive dropout and batch normalization for learning process stabilization. Typical building block:

Dropout (75%)	input: 1024
	output: 1024

Dense	input: 1024
	output: 1024

BatchNormalization	input: 1024
	output: 1024

Activation (relu)	input: 1024
	output: 1024

Matrices

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Parsing result is (0-1) matrix M which represents secondary structure features for sequence ω :

$$M[i, j] = 1 \iff s1 \xrightarrow{*} \omega[i, j], \text{ and } 0 \text{ otherwise.}$$

Vectors

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$[0, 1, 0, 1, 0, 1, 0, 0, 1, 0]$$

$$[84, 128]$$

Line-by-line compressed matrix representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so can be ignored.