Формальные грамммтики и искусственные нейронные сети для анализа вторичной структуры

Семестровый проект на осень 2019

Семён Григорьев

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

14 сентября 2019г.

Кто мы

- 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`

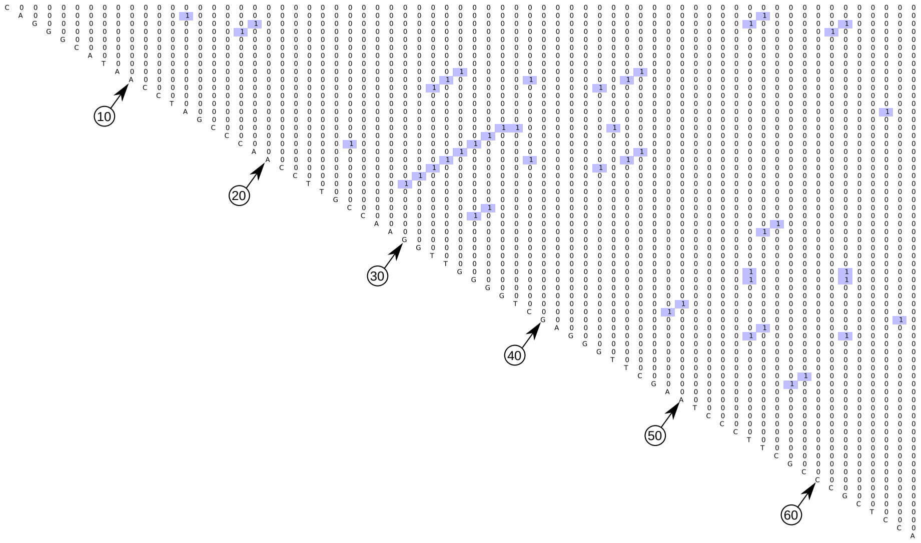
Кто мы

- 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 интереснов
  - Теория формальных языков
  - **Применение теории формальных языков для решения прикладных задач**

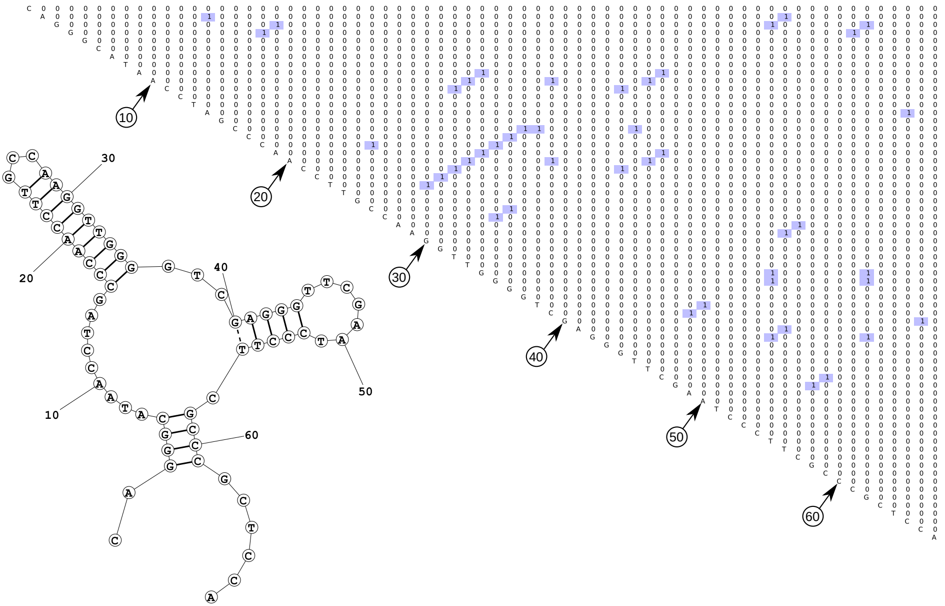
Анализ вторичной структуры: синтаксический анализ + искусственные нейронные сети

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

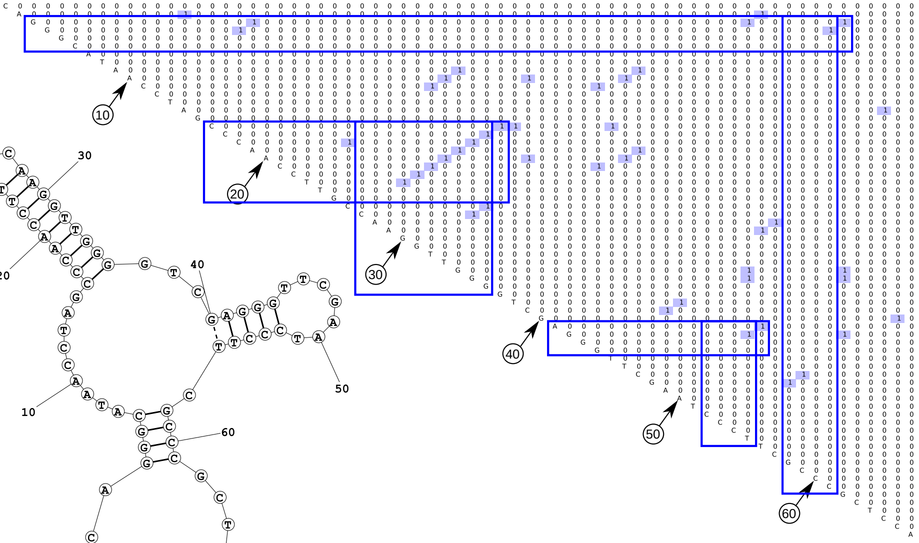
Example 3: real tRNA

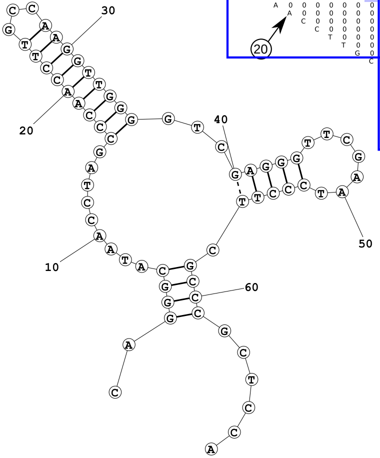


Example 3: real tRNA



Example 3: real tRNA





Задачи

- Подготовка данных для обучения нейронных сетей
  - Поиск и анализ баз РНК-цепочек
  - Подготовка набора данных для обучения: фильтрация, сбор метаданных, приведение к общему формату

Задачи

- Подготовка данных для обучения нейронных сетей
  - Поиск и анализ баз РНК-цепочек
  - Подготовка набора данных для обучения: фильтрация, сбор метаданных, приведение к общему формату
- Подготовка инструментария для анализа вторичной структуры
  - Анализ и сравнение существующих инструментов предсказания вторичной структуры РНК последовательностей
  - Выбор лучшего и его интеграция в процесс обучения нейронных сетей

Требования к кандидатам

- Знание Python (потребуется для автоматизации процесса)
- Знание C/C++ и сопутсвующего инструментария (потребуется при работе с интсрументами)

Перспективы развития

- Создание и обучение моделей для различных задач: предсказание вторичной структуры, классификация, фильтрация химер
- Применить аналогичный подход к белковым цепочкам
- Курсовая/диплом/публикация

Грамматика

```
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
```

Грамматика

```
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
```

Грамматика

```
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
```

Грамматика

```
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
```

Грамматика

```
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
```

Грамматика

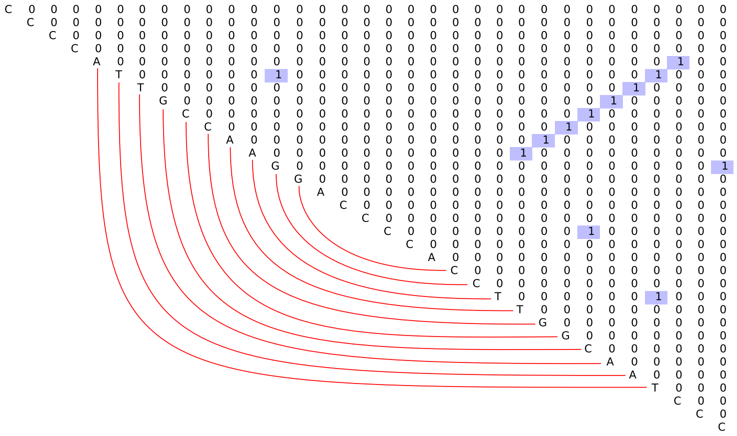
```
s1: stem<s0>
any_str: any_smb*[2..10]
any_smb: A | T | C | G
stem1<s>:                 \\ stem of height exactly 1
      A s T | T s A | C s G | G s C
stem3<s>:                 \\ stem of height exactly 3
      stem1< stem1< stem1<s> > >
stem<s>:                  \\ stem of height 3 or more
    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

Грамматика

```
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
```

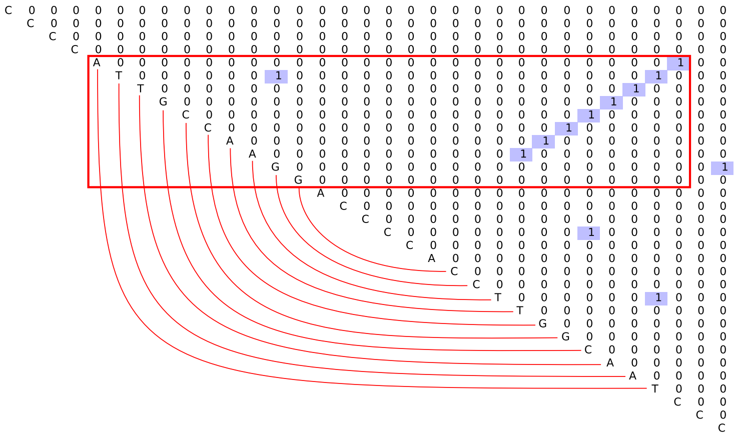
Пример 1: Stem

CCCCATTGCCAAGGACCCCACCTTGGCAATCCC



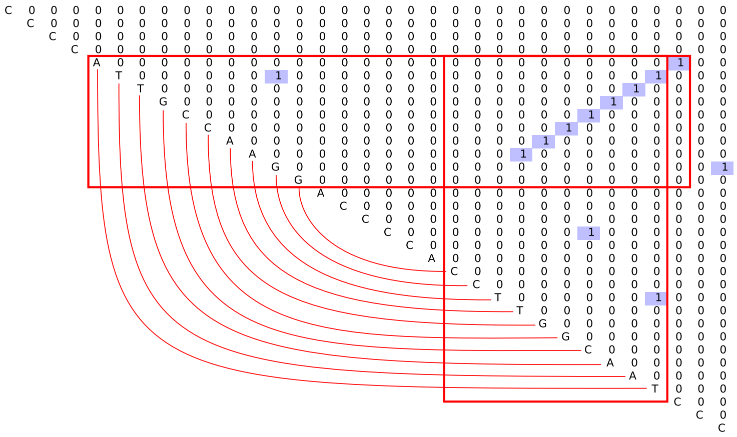
Пример 1: Stem

CCCCATTGCCAAGGACCCCACCTTGGCAATCCC



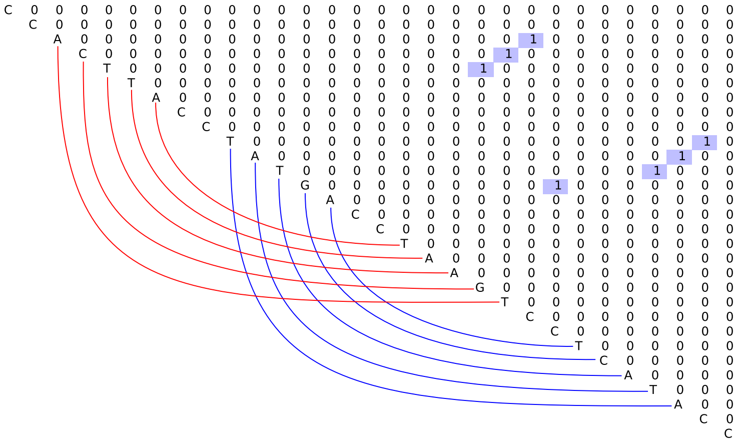
Пример 1: Stem

CCCCATTGCCAAGGACCCCACCTTGGCAATCCC



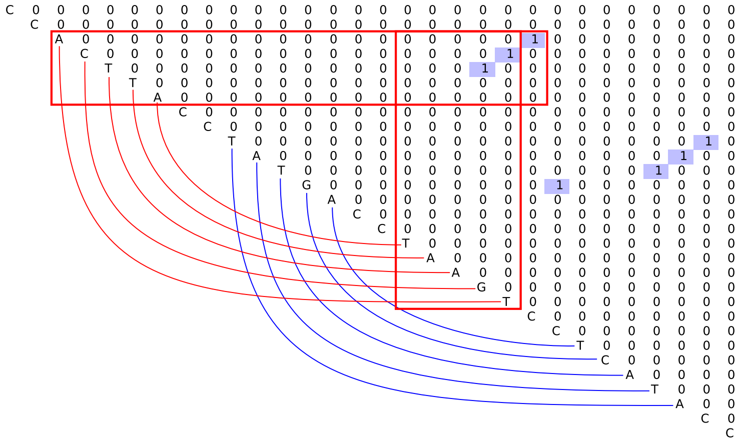
Пример 2: псевдоузел

CCACTTACCTATGACCTAAGTCCTCATACC



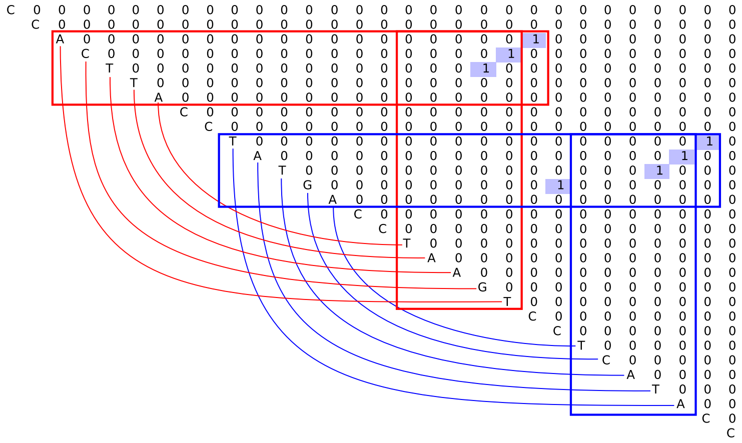
Пример 2: псевдоузел

CCACTTACCTATGACCTAAGTCCTCATACC



Пример 2: псевдоузел

CCACTTACCTATGACCTAAGTCCTCATACC

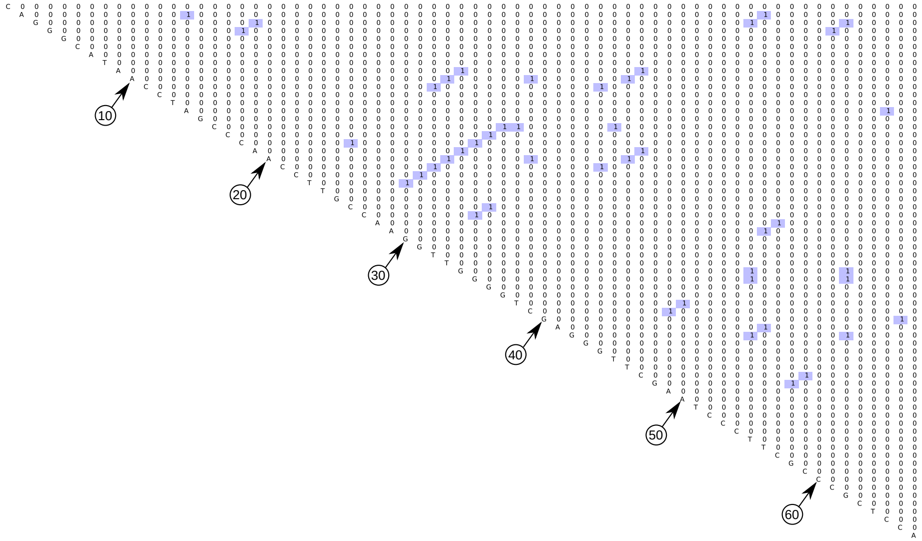


Example 3: real tRNA

```
    CAGGGCATAACCTAGCCCAACCTTGCCAAGG
TTGGGGTCGAGGGTTCGAATCCCTTCGCCCGCTCCA
```

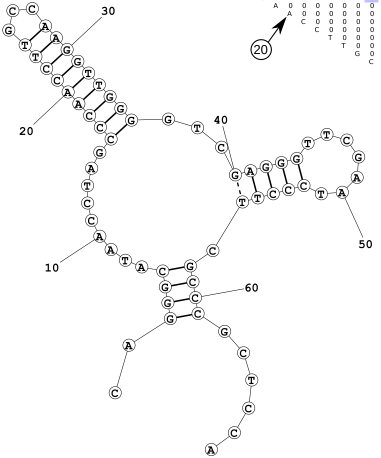
- Novosphingobium aromaticivorans DSM 12444 chr.trna57-GlyGCC(268150-268084) Gly (GCC) 67 bp Sc: 22.9, from GtRNAdb
- Predicted secondary structures are given by using the Fold Web Server with default settings

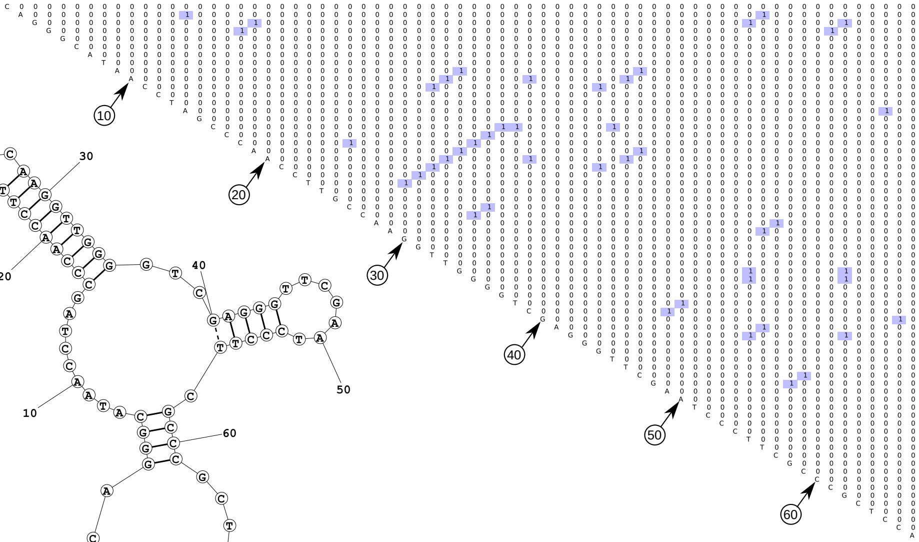
Example 3: real tRNA



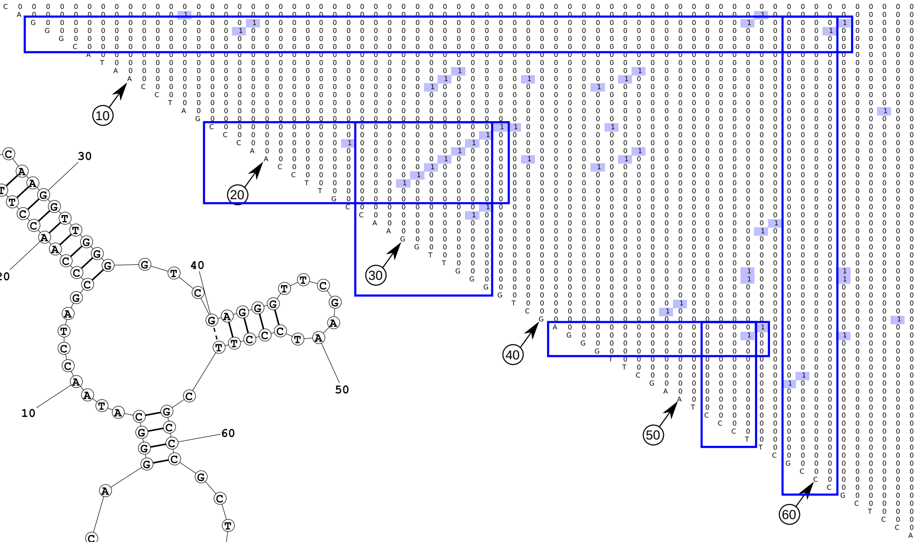
CAGGGCATAACCTAGCCCAACCTTGCCAAGGTTGGGGTCGAGGGTTCGAATCCCTTCGCCCGCTCCA
10
20
30
40
50
60

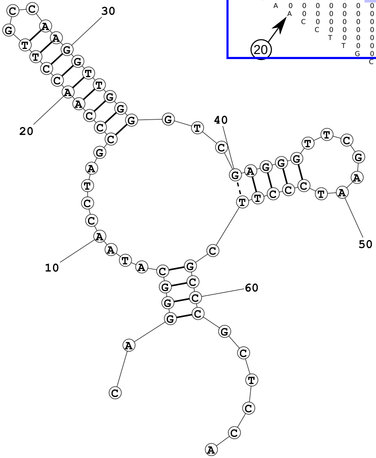
Example 3: real tRNA



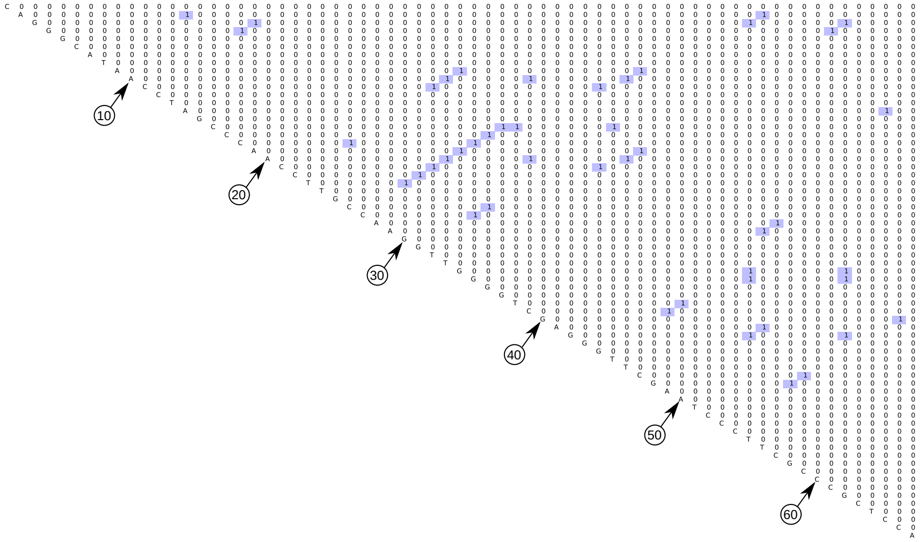


Example 3: real tRNA

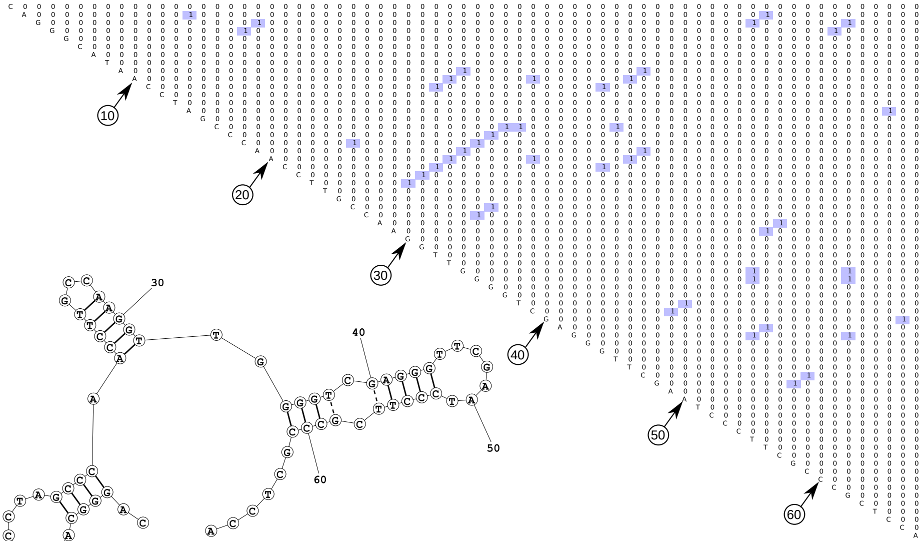


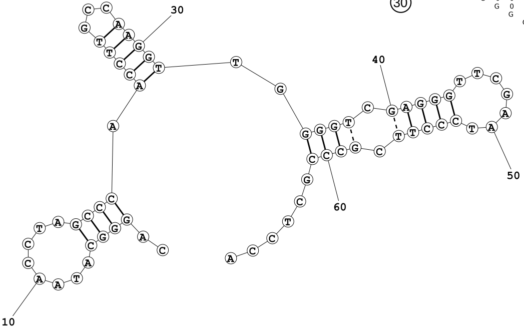


Example 3: real tRNA

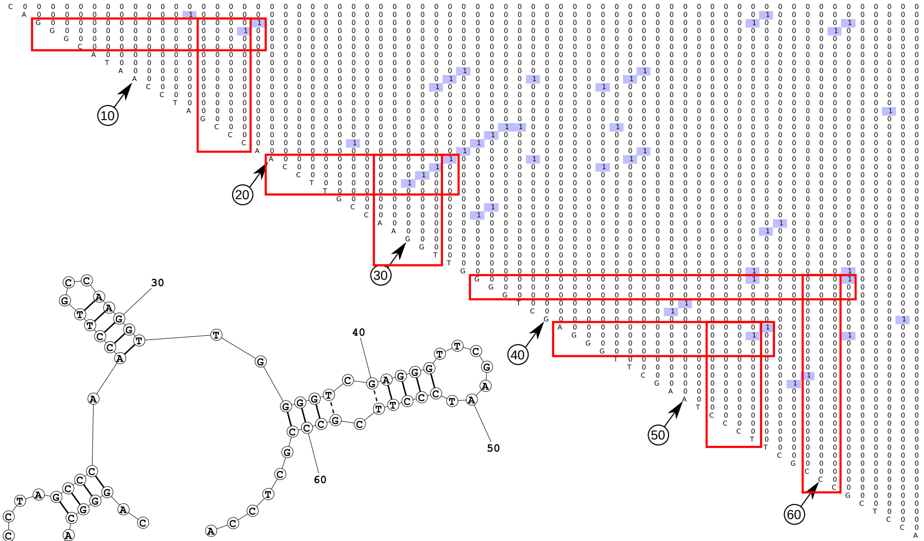


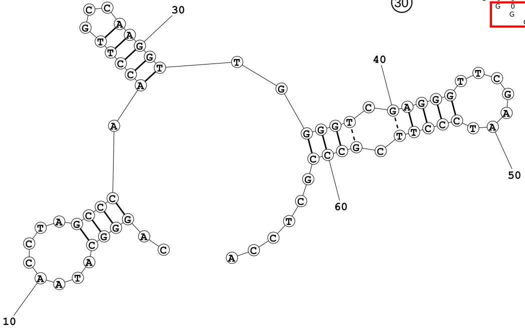
Example 3: real tRNA



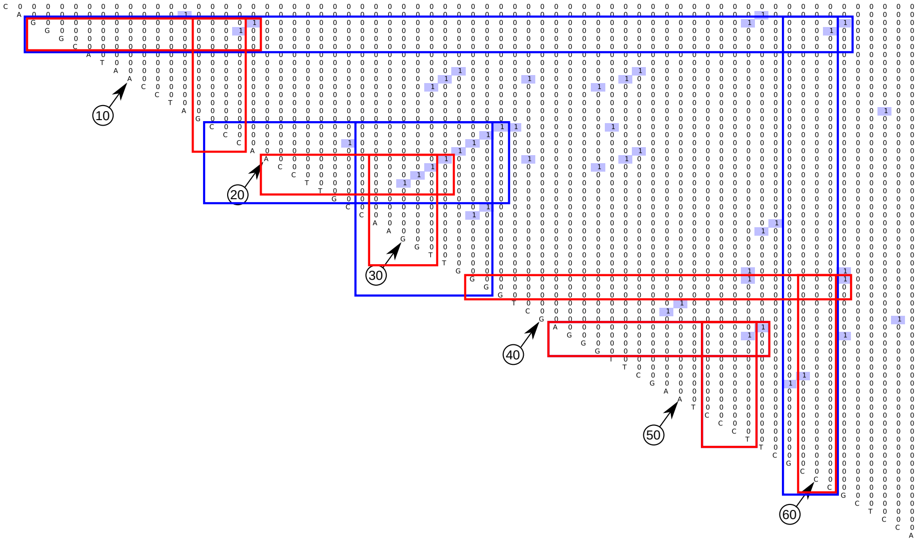


Example 3: real tRNA

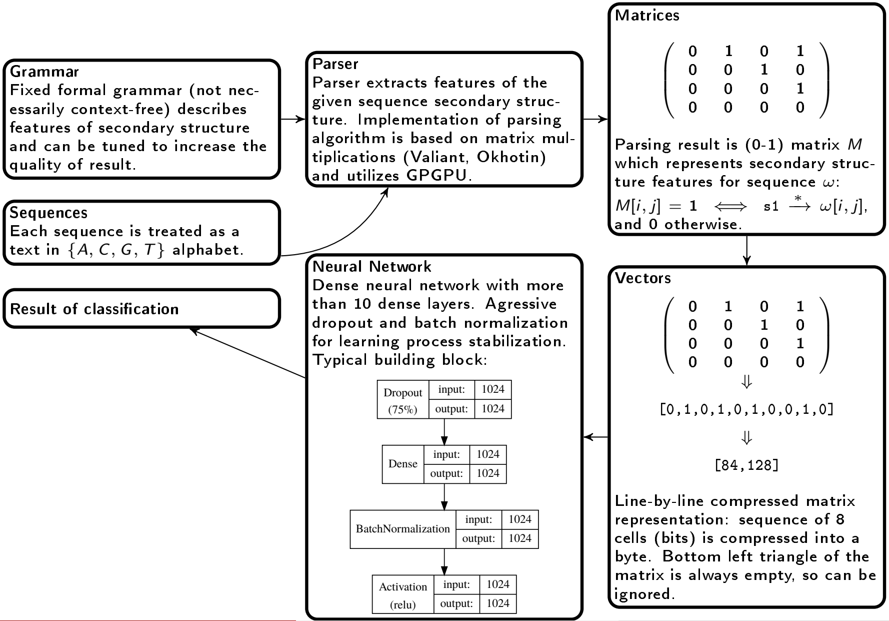




Example 3: real tRNA



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.

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

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

Vectors

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

$\Downarrow$

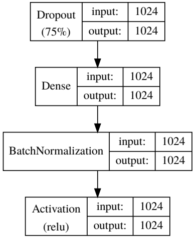
`[0,1,0,1,0,1,0,0,1,0]`

$\Downarrow$

`[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. Agressive 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

Result of classification

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.

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

Vectors

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

$$\Downarrow$$

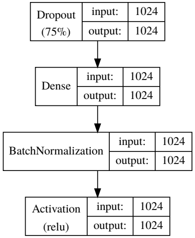
`[0,1,0,1,0,1,0,0,1,0]`

$$\Downarrow$$

`[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. Agressive dropout and batch normalization for learning process stabilization. Typical building block:



Result of classification

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.

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

Vectors

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

$$\Downarrow$$

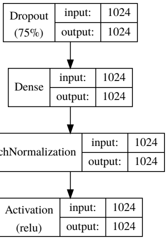
`[0,1,0,1,0,1,0,0,1,0]`

$$\Downarrow$$

`[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. Agressive dropout and batch normalization for learning process stabilization. Typical building block:



Result of classification

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.

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

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

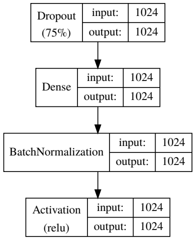
Parsing result is (0-1) matrix M which represents secondary structure features for sequence ω:
$M[i,j] = 1 \iff \mathtt{s1} \xrightarrow{*} \omega[i,j]$,
and 0 otherwise.

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. Agressive dropout and batch normalization for learning process stabilization. Typical building block:



Result of classification

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.

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

Vectors

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

$\Downarrow$

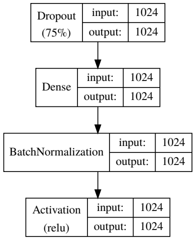
`[0,1,0,1,0,1,0,0,1,0]`

$\Downarrow$

`[84,128]`

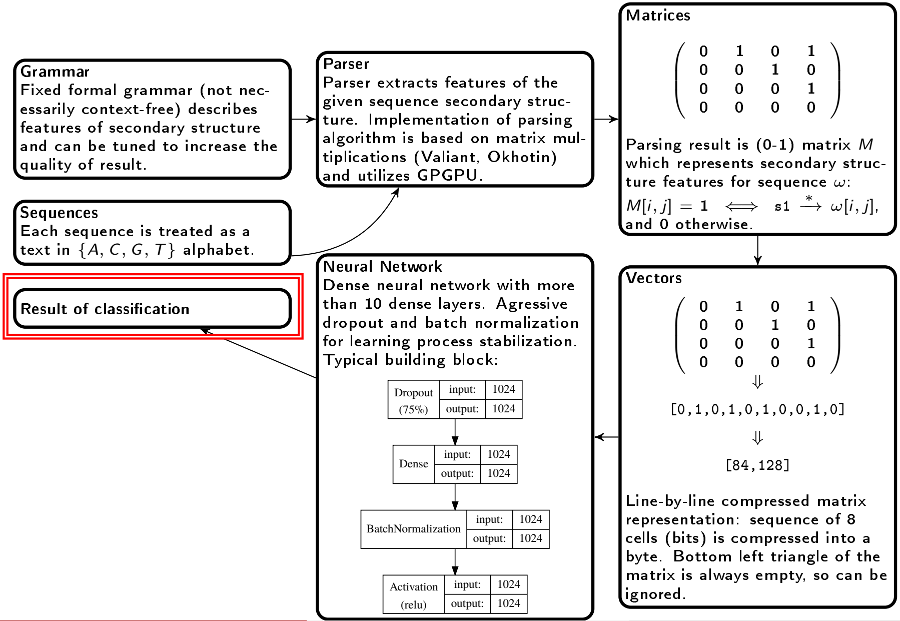
Line-by-line compressed matrix representation: sequence of 8 cells (bits) is compressed into a byte. Bottom left triangle of the matrix is always empty, so can be ignored.

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



Result of classification

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.

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

Result of classification