Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование
информационных систем

Системное программирование

Лунина Полина Сергеевна

Комбинирование нейронных сетей и синтаксического анализа для обработки вторичной структуры последовательностей

Дипломная работа

Научный руководитель:
к. ф.-м. н., доцент Григорьев С. В.

Рецензент:
Специалист по анализу данных ООО ”Интеллоджик” Малыгина Т. С.

Санкт-Петербург
2019

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems

Software Engineering

Polina Lunina

The composition of neural networks and parsing for secondary structure processing

Graduation Thesis

Scientific supervisor:
Assistant Professor Semyon Grigorev

Reviewer:
Data scientist at Intellogic LLC Tatiana Malygina

Saint-Petersburg
2019

Оглавление

Введение

В совершенно разных предметных областях встречаются концептуально схожие задачи, связанные с анализом различных символьных цепочек. Например, распознавание и классификация геномных последовательностей в биоинформатике или поиск аномалий в цепочках системных вызовов в компьютерной безопасности. Часто оказывается, что исследуемые последовательности обладают достаточно специфической синтаксической структурой и, учитывая некоторым способом ее особенности при разработке алгоритмов для решения различных задач, можно значительно повысить их точность и эффективность. Применительно к биологии, синтаксическими структурами можно, в частности, описать вторичную структуру макромолекул. Далее мы будем называть такие синтаксические структуры вторичными структурами в том смысле, что они являются следующим уровнем организации после символьных цепочек. И это определение может применяться не только в биологической области, но и в иных сферах применения.

Одним из классических способов описания вторичной структуры являются формальные грамматики. Они обладают широкими выразительными возможностями и позволяют описать связь между символами, находящимися на большом расстоянии. Например, как показали исследования в области биоинформатики, с помощью контекстно-свободных стохастических грамматик можно смоделировать синтаксическую структуру всей цепочки [13, 3]. Тем не менее, в общем случае создание такой грамматики — достаточно сложная, а иногда и невозможная задача. Поэтому имеет смысл использовать более простую грамматику для описания только ключевых особенностей вторичной структуры, а для ее полноценного анализа применять другие методы.

Существенной проблемой при работе с реальными данными является возможное присутствие различного рода шумов, мутаций и случайных всплесков, что делает точные методы неприменимыми. Распространенный способ обработки таких данных — использование методов машинного обучения, в особенности, искусственных нейронных сетей.

Кроме того, нейронные сети предоставляют возможность эффективно находить сложные и не поддающиеся формализации структурные закономерности во входных данных.

В данной работе мы предлагаем новый подход для класса проблем, связанных с обработкой символьных данных, обладающих некоторой синтаксической структурой. Основная идея подхода — комбинация методов синтаксического анализа и машинного обучения. Мы используем грамматику для описания основных особенностей синтаксической структуры, извлекаем эти особенности с помощью алгоритмов синтаксического анализа и обрабатываем полученные данные с помощью нейронных сетей, сконструированных и обученных для решения конкретной задачи.

1. Постановка задачи

Целью данной работы является разработка подхода для анализа вторичной структуры последовательностей с использованием комбинации синтаксического анализа и нейронных сетей.

Для достижения данной цели в рамках работы были поставлены следующие задачи.

- Разработать архитектуру решения.
- Провести экспериментальные исследования.
- Создать документацию.

2. Обзор областей применения

2.1. Биоинформатика

Одной из областей, где необходим анализ большого количества символьных последовательностей, является биоинформатика. Точные и эффективные методы для решения таких задач, как распознавание и классификация организмов по их генетическим данным, предсказание функций и вторичной структуры белков, аннотация геномов и т.п. стали ключевыми направлениями в современной вычислительной геномике (протеомике).

Материалом для изучения являются нуклеотидные (или, в случае белков, аминокислотные) последовательности, некоторые участки которых соединяются между собой по определенным закономерностям, образуя сложную и стабильную вторичную структуру (рис. 1).

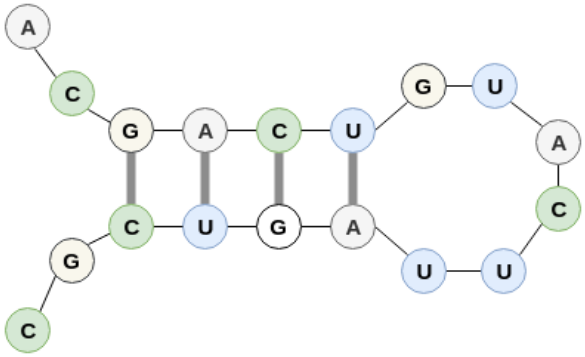

Рис 1: Образование вторичной структуры молекулы РНК

Идея о том, что именно особенности вторичной структуры генетических цепочек существенны для решения задач распознавания и классификации, описана в различных научных работах и широко используется на практике [14, 18]. Существующие подходы к описанию и моделированию вторичной структуры основаны на различных концепциях и алгоритмы, таких как скрытые марковские модели, ковариационные модели [2] и формальные грамматики [15, 13, 4]. Распространенные проблемы при реализации данных подходов заключаются в больших временных затратах и теоретической сложности создания точной грамма-

тики или модели. Тем не менее, многие их этих подходов успешно применяются на практике для разработки различных инструментов [7, 10].

Концептуально другим подходом к решению задач биоинформатики является использование методов машинного обучения [16, 8]. Они позволяют находить сложные закономерности в больших объемах данных и учитывать характерную для биологических данных зашумленность.

Предложенный в данной работе подход позволяет, во-первых, совместить преимущества подходов, основанных на задании вторичной структуры и на машинном обучении, а во-вторых, повысить производительность этапа синтаксического анализа, так использование грамматики только для описания характерных особенностей вторичной структуры вместо классического способа (моделирования структуры всей цепочки) позволит существенно сократить размер грамматики и, как следствие, время, затраченное на работу парсера.

2.2. Компьютерная безопасность

Еще одной потенциальной областью применения предложенного подхода является компьютерная безопасность. Одна из самых острых проблем в данной области — борьба с вредоносными программами. Для обнаружения их воздействия на систему применяют различные подходы, заключающиеся в поиске аномалий в последовательностях системных вызовов (трассах), совершенных другими программами [9, 19, 6]. Для этого нужно реализовать способ идентификации процессов, т.е. по некоторому набору особенностей научиться различать трассы различных программ и выявлять вирусы и отклонения. Трассы представляют из себя некоторые последовательности символов, в которых присутствуют закономерности, характерные для определенных видов программ, следовательно, формальное описание этих закономерностей может оказаться полезным при исследовании системных аномалий.

В работах [22, 23] описан алгоритм обнаружения процессов вредоносных программ, основанный на поиске в трассах некоторых характерных шаблонов, по которым для набора процессов строятся описываю-

щие их модели. Для каждого нового процесса проводится специальная оценка и подбирается наиболее близкая модель из существующих. Затем составляется вектор характеристик, оценивающих поведение процесса в рамках модели и подобные вектора используются для обучения нейронной сети, осуществляющей бинарную классификацию: процессы легитимных и вредоносных программ.

Описанная выше схема использует примерно те же идеи, что лежат в основе предложенного в данной работе подхода. В рамках нашего решения для создания модели понадобятся конъюнктивные грамматики, так как в трассах системных вызовов часто присутствуют повторяющиеся конструкции, выразимые в данном классе грамматик. (???)

3. Разработка архитектуры решения

В данном разделе сначала представлена общая схема предложенного подхода, а затем детально описаны все части архитектуры решения.

3.1. Описание предложенного подхода

Предложенный в данной работе подход может использоваться для решения различных задач во многих исследовательских областях. Ограничения, накладываемые на потенциальную область для апробации подхода следующие. Во-первых, исследуемые данные — некоторый набор символьных последовательностей с метаданными, для которых нужно решить задачу классификации по каким-либо признакам. Во-вторых, на основе анализа специфики области исследования и визуального изучения некоторого подмножества последовательностей можно выделить определенные характерные шаблоны и закономерности в их образовании, т.е. синтаксическую структуру.

Процесс проведения эксперимента состоит из нескольких основных шагов. Сначала создается грамматика, описывающая характерные особенности вторичной структуры рассматриваемых последовательностей. Затем эти особенности извлекаются путем применения некоторого алгоритма синтаксического анализа ко входным данным по заданной грамматике. Полученные в результате работы синтаксического анализатора матрицы разбора приводятся к удобному для дальнейшей обработки виду и подаются на вход нейронной сети, осуществляющей классификацию в соответствии с условиями поставленной задачи. Кроме того, в процессе работы могут понадобиться некоторые дополнительные действия по обработке данных, уникальные для конкретного эксперимента, например, составление выборки, фильтрация и т.п.

Для удобства использования предложенного подхода на практике было необходимо унифицировать и задокументировать все шаги от начальной обработки данных до фиксации результатов. Архитектура решения представлена на рис. 2 и состоит из следующих частей.

- Грамматика в формате yard или аналогичном
- База данных, хранящая сами входные последовательности, метаданные, результат парсинга, различные промежуточные данные и т.д.
- Parsing Tool — утилита для обработки данных синтаксическим анализатором с возможностью сохранения результата в различные форматы
- Neural Networks — модуль для обучения и тестирования нейронных сетей
- Data Processing — модуль для промежуточной обработки данных

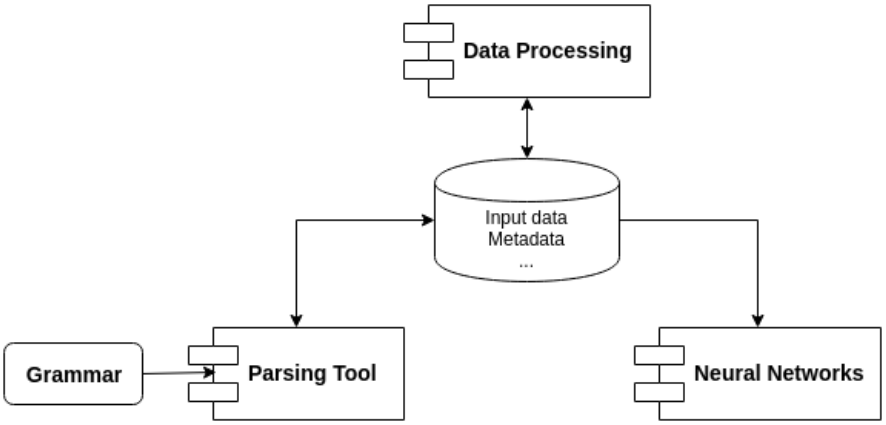

Рис 2: Архитектура решения

В следующих разделах будут детально описаны все шаги по использованию предложенного подхода.

3.2. Подготовка входных данных

В данном разделе описаны необходимые для использования предложенного подхода данные, а также детали их хранения и представления. В процессе апробации подхода были выведены некоторые общие требования, позволяющие наиболее эффективно и удобно проводить экспериментальные исследования.

Все необходимые данные помещаются в некоторое облачное хранилище, откуда могут извлекаться по мере необходимости. Входные данные — исследуемые цепочки — последовательно записаны в файлы формата fasta (для, например, биологических данных) или любого аналогичного текстового формата, причем каждой цепочке соответствует уникальный числовой идентификатор. Вся остальная информация о цепочках (метаданные, принадлежность к тестовой или обучающей выборке, класс и др.) хранится в специальной ссылочной таблице с доступом по идентификатору.

Кроме того, в процессе исследования могут понадобиться специфические скрипты для промежуточной обработки данных, например, выборка для обучения нейронной сети, выравнивание цепочек до фиксированной длины, фильтрация входных данных и др. Эти скрипты реализуются в модуле Data Processing и могут быть переиспользованы в похожих экспериментах.

3.3. Генерация данных с помощью синтаксического анализатора

В рамках предложенной архитектуры синтаксический анализатор — консольная утилита, принимающая на вход грамматику, файл с цепочками и список желаемых выходных форматов. Опишем основные принципы его работы.

Синтаксический анализ – процесс проверки выводимости некоторой подстроки в заданной грамматике. В контексте предложенного решения терминальными символами грамматики являются символы исследуемых последовательностей, правила грамматики описывают характерные особенности их вторичной структуры, а алгоритм синтаксического анализа используется для извлечения этих особенностей путем поиска всех выводимых подстрок для данной строки для всех нетерминалов. Наш подход не зависит от выбора конкретного алгоритма синтаксического анализа, однако в описанных ниже экспериментах мы предлагаем использовать разработанный в рамках проекта YaccConstructor [20] в

лаборатории JetBrains [11] алгоритм, основанный на матричных операциях [1], который демонстрирует высокую производительность на практике в связи с использованием параллельных вычислений.

Результатом работы матричного синтаксического анализатора для строки и фиксированного нетерминала является верхнетреугольная битовая матрица разбора. Мы предлагаем использовать такие матрицы как входные данные нейронной сети, поэтому необходимо привести их к удобному для обработки формату, учитывая специфику конкретной задачи. На данный момент в рамках расширения используемого алгоритма реализованы два формата преобразования матриц.

- Вектора характеристик, сохраняемые в виде строк csv-файла. Генерация векторов осуществляется следующим образом: отбрасывается пустая часть матрицы ниже главной диагонали, оставшиеся строки последовательно преобразуются в битовый вектор, а затем сжимаются в байтовый вектор.

- Черно-белые изображения в формате bmp, получаемые путем замены нулевых битов матрицы на белые пиксели, а единичных — на черные.

Кроме того, в исходный код инструмента можно легко добавить другие выходные форматы.

Классификация как векторов характеристик, так и изображений относится к классическим сценариям использования нейронных сетей, однако стоит отметить, что ввиду специфики конкретной задачи и особенностей входных цепочек, выбор формата данных может повлиять на эффективность и скорость обучения. Например, в процессе экспериментальных исследований было обнаружено, что несмотря на то, что векторные данные занимают меньше памяти и ускоряют процесс обучения, их использование предполагает выравнивание всех цепочек до одинаковой длины, что может оказаться не самым эффективным решением для задач, где входные последовательности имеют принципиально разные длины, так как большая часть вектора для коротких цепочек в данном случае будет заполнена незначащими нулями. Изображения

же можно сгенерировать для цепочек разных длин и затем привести к одному разрешению, что позволяет аккуратно сохранить особенности вторичной структуры даже для коротких цепочек, однако это приводит к ухудшению скорости обучения и вызывает необходимость хранения больших объемов данных.

3.4. Обучение нейронных сетей

Искусственные нейронные сети — широко применяемый метод решения задач классификации в областях, где входные данные обладают сложно формализуемыми закономерностями и могут содержать шумы и неточности. Мы предлагаем использовать нейронные сети для обработки сгенерированных синтаксическим анализатором данных, предполагая, что в них закодированы существенные для классификации особенности синтаксической структуры.

Архитектура нейронной сети уникальна для каждой конкретной задачи и предметной области, однако экспериментальные исследования выявили некоторые общие закономерности и интуиции. Для векторизованных данных высокую эффективность показало чередование полносвязных (dense) слоев ввиду утери информации о взаимном расположении элементов изначальной битовой матрицы и дропаут (dropout) слоев с нормализацией для разжатия данных. Для изображений мы предлагаем использовать небольшое количество сверточных слоев (так как они применяются в основном для извлечения каких-либо особенностей из входных данных, а в нашем случае это уже сделано на этапе синтаксического анализа), затем линеаризацию, а далее перейти к чередующимся dense и dropout слоям, аналогично архитектуре для ветоизованных данных. В качестве алгоритма оптимизации мы предлагаем использовать Adagrad (adaptive gradient), позволяющий выделить редко встречающиеся особенности входных данных, которые, тем не менее, могут оказаться достаточно информативными для решения поставленных задач.

Выше был описан стандартный в рамках нашего подхода способ ис-

пользования нейронных сетей, однако возможны различные его модификации, основанные на конструировании более сложных моделей с загрузкой весов уже обученных, что позволяет упростить задачу или повысить точность результата. Например, расширение нижней части нейронной сети слоями с большим количеством нейронов позволяет провести классификацию на большее количество классов, а расширение верхней части предоставляет возможность подачи на вход данных в другом формате, например, изначальной символьной последовательности вместо результата работы парсера.

Остановимся подробнее на последней модификации. Большинство алгоритмов синтаксического анализа работают за полином от длины входа, поэтому генерация большого количества данных на длинных цепочках потребует существенных временных затрат. Поэтому мы предлагаем следующую идею.

- Сгенерировать некоторый набор данных с помощью синтаксического анализатора и обучить на них нейронную сеть (NN1).
- Создать новую нейронную сеть (NN2), которая расширяет вход NN1 несколькими слоями, верхний из которых принимает символьные цепочки.
- Подгрузить веса NN1 на нижнюю часть NN2 и дообучить NN2.

Эта идея может быть применена как к векторизованным данным, так и к изображениям. В случае изображений необходимо использовать веса NN1, начиная с линеаризованного слоя. Таким образом можно, во-первых, уменьшить размер выборки для генерации парсером, а во-вторых, улучшить точность уже обученных нейронных сетей без временных затрат на дополнительную генерацию данных. Высокая точность и скорость обучения такой нейронной сети была подтверждена экспериментальным путем.

4. Эксперименты

В данном разделе приведены результаты апробации предложенного подхода применительно к некоторым задачам биоинформатики, связанных с классификацией транспортных РНК (тРНК), цепочки которых представляют из себя последовательности символов алфавита {A, C, G, T}, а синтаксической структурой являются закономерности образования вторичной (в биологическом смысле) структуры молекулы РНК.

Контекстно-свободная грамматика, использованная во всех приведенных далее экспериментах, приведена на рис. 3. Грамматика описывает основные структурные элементы вторичной структуры РНК — стемы и петли.

```
s1: stem<s0>
any_str : any_smb*[2..10]
s0: any_str | any_str stem<s0> s0
any_smb: 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> >
```

Рис 3: КС грамматика, описывающая особенности вторичной струкутры РНК

Для создания и обучения нейронных сетей в данной работе были использованы библиотека Keras [12] и фреймворк TensorFlow [17]. Для оценки качества работы описанных далее нейронных сетей были использованы классические метрики, используемые в задачах машинного обучения. Введем некоторые обозначения.

- P — общее количество образцов, по которым классификатор принял правильное решение, N – общее количество образцов, для которых класс был определен неверно.

- TP_c (True Positive) — количество образцов в пределах класса c, которые классификатор распознал как c.
- FP_c (False Positive) — количество образцов из выборки, которые были классифицированы как c, но при этом относятся к другому классу.
- FN_c (False Negative) — количество образцов из класса c, ошибочно отнесенных к другому классу.

На основе данных обозначений определим три метрики.

- $Accuracy = \frac{P}{N}$ ("правильность" — доля правильно определенных классификатором образцов по всей тестовой выборке).
- $Precision_c = \frac{TP_c}{TP_c+FP_c}$ ("точность" в пределах класса c — доля образцов, действительно принадлежащих c, относительно всех образцов, которые классификатор отнес к c).
- $Recall_c = \frac{TP_c}{TP_c+FN_c}$ ("полнота" в пределах класса c — доля найденных классификатором образцов, принадлежащих c, относительно всех образцов этого класса в тестовой выборке).

4.1. Классификация тРНК

В данном разделе описаны исследования возможности применения предложенного метода к задачам классификации последовательностей тРНК различных организмов. В описанных далее экспериментах поставлены следующие задачи: классификация тРНК на 2 класса (эукариоты и прокариоты) и на 4 класса (археи, бактерии, растения и грибы).

Как было упомянуто в разделе 3.3, в существующем инструменте для синтаксического анализа существует два формата выходных данных: изображения и числовые вектора. Кроме того, в разделе 3.4 была описана идея расширения обученных нейронных сетей верхними слоями, принимающими на вход исходную символьную последовательность,

а не результат работы парсера. На основе этих соображений, для поставленных выше задач были проведены два типа экспериментов на одних и тех же данных: обучение нейронной сети на векторизованных данных с последующим дообучением до строкового входа и аналогично для изображений.

4.1.1. Классификация тРНК: эукариоты и прокариоты

В данном эксперименте была поставлена задача бинарной классификации цепочек тРНК эукариотов и прокариотов (две группы, на которые разделяются все живые организмы в зависимости от наличия ядер в их клетках). Для обучения нейронных сетей было взято 35000 последовательностей из баз [5, 21] в соотношении 20000:5000:10000 образцов на обучение, валидацию и тестирование соответственно.

Подход на основе векторизованных данных

Перед генерацией векторов было необходимо выравнить цепочки до одной длины; мы выбрали верхнюю границу в 220 символов, так как, за исключением единичных случайных всплесков, большая часть последовательностей тРНК имеет меньшую длину. Для выравнивания цепочек был использован следующий алгоритм: если длина строки больше 220, то взять ее первые 220 символов, иначе добавить в конец необходимое количество специальных символов, не встречающихся в оригинальных цепочках, и учесть этот факт при создании грамматики.

На сгенерированных синтаксическим анализатором данных была обучена нейронная сеть (бинарный классификатор), состоящая из чередующихся dense и dropout слоев с нормализацией. Далее была сконструирована вторая нейронная сеть, состоящая из двух блоков: входной слой с 220 нейронами и затем несколько полносвязных слоев, постепенно расширяющихся до длины вектора (3028), а затем — слои вышеописанной нейронной сети, для которых при обучении были загружены уже посчитанные веса. Таким образом, на основе векторного классификатора был создан классификатор последовательностей тРНК, архитектура которого показана на рис. 4, здесь правая часть — модель нейронной

сети, принимающей на вход векторизованные данные, а левая часть — надстройка, преобразующая нуклеотидную последовательность в вектор.

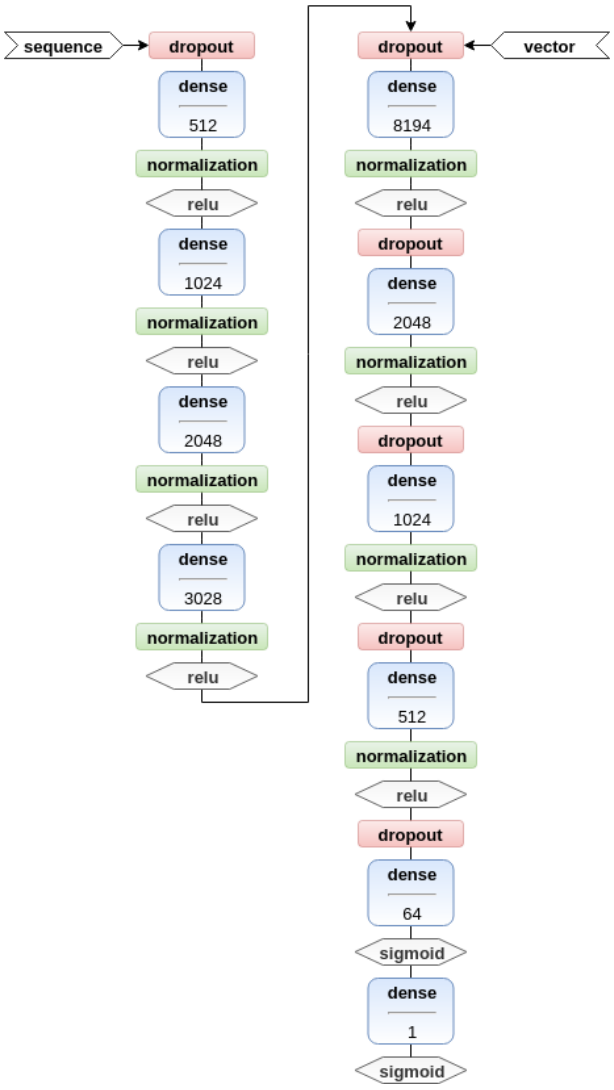

Рис 4: Архитектура нейронной сети для классификации тРНК

Подход на основе изображений

Результатом работы синтаксического анализатора на цепочке длины N является изображение размера $N \times N$, поэтому унификацию размеров сгенерированных данных можно провести путем последующего приведения изображений к единому разрешению, что позволяет обрабатывать последовательности разных длин без выравнивания.

В данном эксперименте на наборе сгенерированных и приведенных

к размеру 80×80 данных была обучена нейронная сеть со входным сверточным слоем (convolution layer), линеаризацией (flatten), а затем, аналогично архитектуре для векторных данных, чередованием dense и dropout слоев. Далее, по той же схеме, что было описано ранее, было реализовано расширение полученного классификатора до строкового входа. Единственное различие заключается в том, что из модели комбинированной нейронной сети был удален сверточный слой, таким образом, на стыке надстройки и обученной модели в данном случае находится линеаризованный слой.

Результаты

Результаты тестирования на тестовой выборке в 10000 образцов были оценены в соответствии с приведенными в начале раздела 4 метриками. В таблице 1 показана оценка классификаторов по метрике *Accuracy* до и после дообучения до строкового входа, а также суммарное время, затраченное на обучение. В таблице 2 продемонстрированы результаты вычисления метрик *Precision* и *Recall* для обоих классов на расширенных моделях.

	vector-based approach	image-based approach
base model accuracy	94.1%	96.2%
extended model accuracy	97.5%	97.8%
total training time	28300 s	6100 s

Таблица 1: Доля правильных ответов классификаторов и время обучения

class	vector-based approach		image-based approach	
	precision	recall	precision	recall
prokaryotic	95.8%	99.4%	96.2%	99.4%
eukaryotic	99.4%	95.6%	99.4%	99.5%
average	97.6%	97.5%	97.8%	99.5%

Таблица 2: Сравнение результатов тестирования подходов по метрикам precision и recall для каждого класса

4.1.2. Классификация тРНК: археи, бактерии, растения и грибы

В данном эксперименте была исследована возможность классификации цепочек тРНК на большее количество классов, менее крупных с точки зрения биологической систематики. Анализируемые последовательности в количестве 12000 образцов, поровну на каждый класс, были взяты из тех же самых баз [5, 21] и поделены для обучения, валидации и тестирования в соотношении 8000:3000:1000 образцов соответственно.

Аналогично эксперименту, описанному в предыдущем разделе, были обучены векторный классификатор и классификатор изображений с единственным различием в размере выходного слоя. Затем для обоих классификаторов было реализована надстройка, принимающая исходные цепочки. Время работы, а также различные оценки результатов, полученных при тестировании обоих мультиклассовых классификаторов по метрикам *Accuracy*, *Precision* и *Recall* приведены в таблицах 3 и 4.

	vector-based approach	image-based approach
base model accuracy	86.7%	93.3%
extended model accuracy	96.2%	95.7%
total training time	36000 s	4200 s

Таблица 3: Доля правильных ответов классификаторов и время обучения

class	vector-based approach		image-based approach	
	precision	recall	precision	recall
archaeal	91.1%	99.2%	91.6%	98.5%
bakterial	96.6%	95.1%	95.2%	95.5%
fungi	98.5%	94.9%	97.5%	94.3%
plant	99.4%	95.7%	99.2%	94.7%
average	96.4%	96.2%	95.9%	95.8%

Таблица 4: Сравнение результатов тестирования подходов по метрикам precision и recall для каждого класса

4.2. Анализ результатов

В результате экспериментов на генетических данных была доказана возможность применения комбинации синтаксического анализа и нейронных сетей к различным задачам биоинформатики.

Были исследованы два формата представления данных, хранящих информацию о вторичной структуре, и влияние выбора формата на процесс обучения нейронных сетей. На основе полученных результатов видно, что скорость обучения на изображениях в разы выше, чем на векторных данных при несущественных различиях в точности, так как в векторных данных нарушено взаимное расположение закодированных элементов вторичной структуры, а также сжатие информации при генерации вектора требует дополнительных действий по ее обработке при обучении. Тем не менее, в случае достаточно большой длины входной цепочки обучение нейронной сети на изображениях может привести к существенным затратам по времени и памяти вплоть до превышения ограничений на использование памяти GPU и CPU в фреймворках для машинного обучения. Кроме того, изображения требуют привлечения больших ресурсов для хранения.

Решение задач биоинформатики является частным и не единственным примером области исследования применимости предложенного подхода. В других областях могут потребоваться иные форматы представления данных и архитектуры нейронных сетей, более сложные грамматики и т.д.

Заключение

В ходе данной работы были получены следующие результаты.

- Разработана архитектура решения для использования предложенного подхода.
- Проведены экспериментальные исследования предложенного подхода на задачах классификации тРНК.
- Задокументировано описание схемы работы подхода и результаты проведенных экспериментов.
- Опубликована статья "The Composition of Dense Neural Networks and Formal Grammars for Secondary Structure Analysis" на конференции BIOINFORMATICS 2019 и представлен постер "16s rRNA Detection by Using Neural Networks" на конференции Biata 2018
- Исходный код и докуметация доступны по ссылке https://github.com/Lun

Существует несколько направлений дальнейшего развития полученных результатов.

- Эксперименты в области кибербезопасности — поиск аномалий в последовательностях системных вызовов.
- Предсказание функций белков.
- Распознавание химер.
- Моделирование вторичной структуры геномных последовательностей

Список литературы

[1] Azimov Rustam, Grigorev Semyon. Context-free Path Querying by Matrix Multiplication // Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA). — GRADES-NDA '18. — New York, NY, USA : ACM, 2018. — P. 5:1–5:10. — Access mode: http://doi.acm.org/10.1145/3210259.3210264.

[2] Biological sequence analysis: probabilistic models of proteins and nucleic acids / Richard Durbin, Sean R Eddy, Anders Krogh, Graeme Mitchison. — Cambridge university press, 1998.

[3] Dowell Robin D. RNA structural alignment using stochastic context-free grammars. — Washington University, 2004.

[4] Dowell Robin D, Eddy Sean R. Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction // BMC bioinformatics. — 2004. — Vol. 5, no. 1. — P. 71.

[5] Genomic tRNA Database [Электронный ресурс]. — Access mode: http://gtrnadb2009.ucsc.edu/ (online; accessed: 05.05.2019).

[6] Ghosh Anup K, Schwartzbard Aaron. A Study in Using Neural Networks for Anomaly and Misuse Detection. // USENIX security symposium. — Vol. 99. — 1999. — P. 12.

[7] HMMER [Электронный ресурс]. — Access mode: http://hmmer.org/ (online; accessed: 05.05.2019).

[8] Higashi Susan, Hungria Mariangela, Brunetto MADC. Bacteria classification based on 16S ribosomal gene using artificial neural networks // Proceedings of the 8th WSEAS International Conference on Computational intelligence, man-machine systems and cybernetics. — 2009. — P. 86–91.

[9] Hofmeyr Steven A, Forrest Stephanie, Somayaji Anil. Intrusion detection using sequences of system calls // Journal of computer security. — 1998. — Vol. 6, no. 3. — P. 151–180.

[10] Infernal [Электронный ресурс]. — Access mode: `http://eddylab.org/infernal/` (online; accessed: 05.05.2019).

[11] JetBrains Programming Languages and Tools Lab [Электронный ресурс]. — Access mode: `https://research.jetbrains.org/groups/plt_lab` (online; accessed: 05.05.2019).

[12] Keras [Электронный ресурс]. — Access mode: `https://keras.io/` (online; accessed: 08.05.2019).

[13] Knudsen Bjarne, Hein Jotun. RNA secondary structure prediction using stochastic context-free grammars and evolutionary history // Bioinformatics (Oxford, England). — 1999. — Vol. 15, no. 6. — P. 446–454.

[14] RNAscClust: clustering RNA sequences using structure conservation and graph based motifs / Milad Miladi, Alexander Junge, Fabrizio Costa et al. // Bioinformatics. — 2017. — Vol. 33, no. 14. — P. 2089–2096.

[15] Rivas Elena, Eddy Sean R. The language of RNA: a formal grammar that includes pseudoknots // Bioinformatics. — 2000. — Vol. 16, no. 4. — P. 334–340.

[16] Sherman Douglas. Humidor: Microbial Community Classification of the 16S Gene by Training CIGAR Strings with Convolutional Neural Networks. — 2017.

[17] TensorFlow [Электронный ресурс]. — Access mode: `https://www.tensorflow.org/` (online; accessed: 08.05.2019).

[18] Variation in secondary structure of the 16S rRNA molecule in cyanobacteria with implications for phylogenetic analysis /

Klára Řeháková, Jeffrey R Johansen, Mary B Bowen et al. // Fottea. — 2014. — Vol. 14. — P. 161–178.

[19] Wespi Andreas, Dacier Marc, Debar Hervé. Intrusion detection using variable-length audit trail patterns // International Workshop on Recent Advances in Intrusion Detection / Springer. — 2000. — P. 110–129.

[20] YaccConstructor [Электронный ресурс]. — Access mode: `https://github.com/YaccConstructor` (online; accessed: 05.05.2019).

[21] tRNADB-CE [Электронный ресурс]. — Access mode: `http://trna.ie.niigata-u.ac.jp/cgi-bin/trnadb/index.cgi` (online; accessed: 05.05.2019).

[22] Баклановский Максим Викторович, Ханов Артур Рафаэльевич. Поведенческая идентификация программ // Моделирование и анализ информационных систем. — 2015. — Vol. 21, no. 6. — P. 120–130.

[23] Оценка точности алгоритма распознавания вредоносных программ на основе поиска аномалий в работе процессов / МВ Баклановский, АР Ханов, КМ Комаров, ПА Лозов // Научно-технический вестник информационных технологий, механики и оптики. — 2016. — Vol. 16, no. 5.