УДК 519.713.1
doi:10.31799/1684-8853-2020-1-34-43

Научные статьи
Articles

Генерация автоматов на основе рекуррентных нейросетей и автоматического выбора кластеризации

П. Г. Грачев[a], *аспирант, orcid.org/0000-0002-1133-8432*
С. Б. Муравьев[a], *канд. техн. наук, программист, orcid.org/0000-0002-4251-1744*
А. А. Фильченков[a], *канд. физ.-мат. наук, доцент, orcid.org/0000-0002-1133-8432*
А. А. Шалыто[a], *доктор техн. наук, профессор, orcid.org/0000-0002-2723-2077, shalyto@mail.ifmo.ru*
[a]*Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, Кронверкский пр., 49, Санкт-Петербург, 197101, РФ*

Введение: *задача индуцирования регулярной грамматики состоит в построении детерминированных конечных автоматов по списку слов-примеров и слов-контрпримеров из неизвестного регулярного языка. Эта задача является одной из центральных в теории формальных языков и сопряженных с ней областях. Одним из наиболее успешных ее решений является обучение рекуррентной нейронной сети классификации слов и последующая кластеризация векторов в пространстве весов. Однако результат кластеризации не всегда позволяет построить непротиворечивый автомат. Более сложные модели для индуцирования грамматики требовательны к памяти, времени обучения и числу тренировочных примеров.* ***Цель:*** *построение модели для индуцирования грамматики, использующей новые методы машинного обучения.* ***Методы:*** *в качестве классификатора применена рекуррентная нейронная сеть, использующая предложенную авторами функцию ошибки. Для кластеризации реализован метод совместного выбора и настройки гиперпараметров алгоритма кластеризации для различных мер оценки.* ***Результаты:*** *проведено тестирование на наборах данных, соответствующих десяти различным регулярным грамматикам и, соответственно, десяти различным детерминированным автоматам. По результатам тестирования разработанная модель корректно синтезирует автоматы с числом состояний и с числом входных символов не более пяти. Для четырех из семи успешно индуцированных грамматик построенный автомат получился минимальным. Для трех наборов данных Tech. автомат построить не удалось — либо из-за недостаточности числа кластеров в предложенном разбиении, либо из-за невозможности построить из этого разбиения непротиворечивый автомат.* ***Обсуждение:*** *применение алгоритма поиска наибольшего правдоподобия между кластерами векторов и соответствующими состояниями автомата для принятия решений в случае нахождения противоречий могло бы расширить область применимости модели.*

Ключевые слова *— индуцирование грамматики, рекуррентные нейронные сети, кластеризация.*

Для цитирования: Грачев П. Г., Муравьев С. Б., Фильченков А. А., Шалыто А. А. Генерация автоматов на основе рекуррентных нейросетей и автоматического выбора кластеризации. *Информационно-управляющие системы*, 2020, № 1, с. 34–43. doi:10.31799/1684-8853-2020-1-34-43

For citation: Grachev P. G., Muravyov S. B., Filchenkov A. A., Shalyto A. A. Automata generation based on recurrent neural networks and automated cauterization selection. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2020, no. 1, pp. 34–43 (In Russian). doi:10.31799/1684-8853-2020-1-34-43

Введение

Формальный язык является одним из основных понятий математической логики и информатики. Он представляет собой множество конечных строк, состоящих из символов, которые принадлежат некоторому конечному алфавиту. Такой язык может быть определен не только перечислением всех его слов, но и списком некоторых формально описанных правил их порождения. Формальные языки широко используются в самых разнообразных областях, включая среды программирования [1], формальную логику и языки [2, 3] и представление естественных языков [4].

Согласно иерархии Хомского, формальные языки можно разбить на четыре вложенных класса: неограниченные, контекстно-зависимые, контекстно-свободные и регулярные [5]. Регулярные языки могут быть представлены с помощью детерминированных конечных автоматов (ДКА). Автомат — это абстрактная математическая модель, состоящая из состояний и переходов между ними. Детерминированные автоматы используются не только для представления регулярных языков, но и как самостоятельная математическая модель, например в различных управляющих системах, таких как светофоры, торговые автоматы, лифты, а также для сетевых протоколов [6]. Автоматы применяются в биологии [7], логистике [8], компьютерной безопасности [9] и т. д.

Построение ДКА, соответствующего некоторому формальному языку, по списку формальных правил или по текстовому описанию этого языка является одной из наиболее частых задач в теории формальных языков [10], а методы ее решения известны из литературы [11, 12]. Такая задача, как правило, возникает при использовании моделей и идей из теории формальных языков, например в системах, проверяющих коррект-

ность написания строк [13]. Однако когда правила неизвестны, требуется построить детерминированный автомат по подмножеству слов, принадлежащих некоторому неизвестному формальному языку. Такое описание не является полным и однозначным, поскольку регулярный язык может содержать бесконечно много слов. Проблема получения некоторого формального описания языка по конечному числу слов-примеров, а в некоторых случаях слов-контрпримеров, в англоязычной литературе называется *grammar inference* [14]. За этим понятием нет закрепившегося термина на русском языке. В рамках данной работы будет использоваться термин *индуцирование грамматики*.

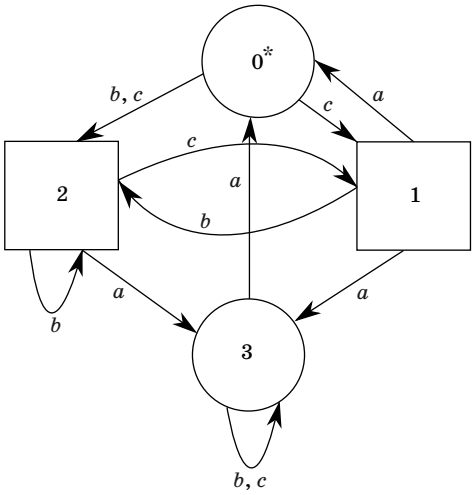
Базовым и исторически первым алгоритмом для индуцирования грамматики стал алгоритм смешения состояний [15]. Его идея состоит в построении префиксного дерева, эквивалентного примерам из тестовой выборки с последующим объединением состояний. Появление и развитие технологий машинного обучения позволило разработать новые модели для индуцирования грамматик. Одним из наиболее удачных решений является применение рекуррентной нейронной сети, классифицирующей слова по их принадлежности к неизвестному формальному языку [16–19]. Обученная нейронная сеть выделяет векторы (состояния весов), которые впоследствии подвергаются кластеризации. Кластеры состояний приводятся в соответствие с состояниями конструируемого автомата с помощью алгоритма, описанного далее. В результате строится детерминированный автомат, соответствующий неизвестному формальному языку.

В данной работе предлагается новый метод решения задачи индуцирования грамматики, использующий модификацию классифицирующей рекуррентной нейронной сети. В этой сети кластеризация весов выполняется с применением метода совместного выбора и настройки гиперпараметров алгоритма кластеризации на основе обучения с подкреплением.

Основные понятия

Детерминированный конечный автомат

Детерминированный конечный автомат — один из способов представления регулярных языков [20]. Автомат состоит из состояний, которые могут быть терминальными или нетерминальными, и переходов между ними. Каждый переход осуществляется по одному из символов алфавита. В графическом представлении одного из ДКА (рис. 1) квадратами показаны терминальные состояния, окружностями — нетерминальные. Стартовое состояние отмечено звездочкой. Все переходы между состояниями помечены символами, по которым переходы осуществляются.



■ ***Рис. 1.*** Пример детерминированного конечного автомата

■ ***Fig. 1.*** An example of deterministic finite-state automaton

Для того чтобы определить принадлежность слова к регулярному языку по ДКА, необходимо обработать данное слово с помощью автомата. Обработка начинается в нулевом состоянии и происходит посимвольно. Если после обработки автомат закончил свою работу в терминальном состоянии, то слово является принадлежащим искомому регулярному языку, и наоборот. В качестве иллюстрации рассмотрим автомат из рис. 1 и определим с его помощью принадлежность к описываемому им языку слова *'acab'*. Автомат начинает свою работу в нулевом состоянии и последовательно совершает переходы в другие состояния по символам *'a'*, *'c'*, *'a'* и *'b'*. Таким образом, цепочка переходов: $0 \to 1 \to 2 \to 3 \to 3$. Состояние 3 является нетерминальным, поэтому слово *'acab'* не принадлежит к регулярному языку.

Искусственные нейронные сети

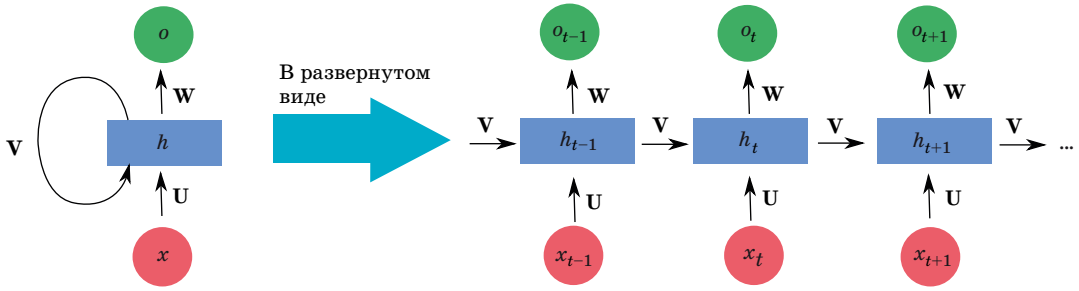
Искусственная нейронная сеть — это система соединенных между собой искусственных нейронов, каждый из которых принимает на вход числовые сигналы и формирует с использованием функции активации выходной числовой сигнал, который подается на вход других нейронов.

В данной работе применяются рекуррентные нейронные сети. Пример рекуррентной сети представлен на рис. 2, где **W, V, U** — матрицы весов.

Сеть состоит из входного X, скрытого H и выходного O слоев. Скрытые и выходные сигналы этой сети описываются соотношениями

$$h_t = f(\mathbf{V} \times h_{t-1} + \mathbf{U} \times x_t + c);$$

$$o_t = g(\mathbf{W} \times h_t + d),$$



■ ***Рис. 2.*** Пример рекуррентной нейронной сети
■ ***Fig. 2.*** An example of recurrent neural network

где f, g — функции активации; c, d — параметры сети.

Для того чтобы правильно настроить параметры сети (**W**, **V**, **U**, c, d), используется техника обучения с учителем. При этом часть данных помечается как тестовые, они последовательно подаются на вход сети, а ее ответ сравнивается с ожидаемым. Обычно используется некоторая функция ошибки $E(y, N(x))$, зависящая от реального значения y и ответа сети $N(x)$. Далее по значению функции ошибки пересчитываются значения параметров сети. Для этого, например, можно применить метод обратного распространения ошибки, который использует градиентный спуск.

Задача индуцирования грамматики и ее решение с помощью рекуррентных нейронных сетей

Формально задача индуцирования регулярной грамматики формулируется следующим образом. Задан алфавит и список строк над ним, каждая строка помечена как принадлежащая или не принадлежащая неизвестному регулярному языку. Необходимо построить детерминированный конечный автомат, соответствующий этому языку (он корректно обрабатывает все строки из входного списка). Этот автомат должен содержать как можно меньше состояний.

Классифицирующая рекуррентная нейронная сеть была впервые применена к задаче индуцирования регулярной грамматики в работе [16]. Предложенная авторами нейросеть обрабатывает слова посимвольно, а в качестве ответа возвращает число, которое можно интерпретировать как вероятность того, что обрабатываемое слово принадлежит искомому регулярному языку.

Обрабатываемое слово рассматривается как последовательность символов. Каждый символ алфавита кодируется в виде единичного вектора (вектора, состоящего из нулей на всех позициях, кроме одной, где стоит единица) размерности M, где M — мощность словаря. Если очередной символ слова является i-м символом алфавита, то соответствующий ему единичный вектор будет содержать единицу на i-й позиции и нули на всех остальных. Рекуррентный слой нейронной сети принимает на вход векторы символа и состояния и возвращает следующий вектор состояния. Каждый вектор состояния имеет одинаковую размерность N. Следующее состояние вычисляется по формуле: $\mathbf{s}_{i+1} = \mathbf{s}_i \times \mathbf{W} \times a_i$, где $\mathbf{s}_i$ — вектор состояний; $\mathbf{a}_i$ — вектор символов алфавита; **W** — параметрическая матрица размерности $N \times N \times M$. Ответ нейронной сети зависит от вектора-состояния, полученного после последовательной обработки всех символов слова. Нулевой элемент этого вектора рассматривается как ответ нейронной сети. Описанная нейронная сеть обучается на примерах до наступления условий останова.

После того как нейронная сеть обучена, формируется список векторов-состояний, получаемый при обработке слов-примеров. К векторам из этого списка применяется алгоритм кластеризации, описанный в следующем разделе, и каждый полученный кластер рассматривается как соответствующий определенному состоянию автомата.

Таким образом, каждому слову из списка слов-примеров ставится в соответствие некоторый кластер, который в свою очередь соответствует некоторому состоянию автомата. Если для определенной строки s известно соответствующее ей состояние f, а также известно состояние t, соответствующее строке $s*c$, где c — символ из словаря, а * — знак конкатенации, то это позволяет построить переход из состояния f в состояние t по символу c. С использованием данного правила строится ДКА, который и возвращается в качестве ответа для задачи индуцирования грамматики.

Описанная схема применения рекуррентной нейронной сети получила свое развитие в работах [17, 18] и остается актуальной для задачи грамматического индуцирования. Так, например, авторы [19] используют в качестве классификатора нейронную сеть архитектуры долгой крат-

косрочной памяти (*Long Short Term Memory — LSTM*), а в качестве алгоритма кластеризации — *L* Angluin*.

Предложенный алгоритм индуцирования грамматики

Используемая рекуррентная нейронная сеть

В основе применяемой модели для индуцирования грамматики лежит модификация рекуррентной нейронной сети, описанной в предыдущем разделе, которая решает задачу классификации. Размерность вектора, определяющего состояние сети, задается величиной параметра *n_neurons*.

В качестве функции ошибки предлагается специальная функция, названная *CardiLoss*. Значение функции рассчитывается по формуле

$$CardiLoss(x) = 4(N(x) - y)^2,$$

где x — слово-пример; $N(x)$ — ответ нейронной сети; y — правильный ответ. При тестировании модели авторами установлено, что сети, имеющие функцию ошибки, прямо пропорциональные $N(w)$, склонны сходиться к локальному минимуму, соответствующему возвращению ответа 0,5 на любые входные данные. Функция *CardiLoss* сформирована так, что имеет значение, близкое к 1,0 при ответе нейронной сети, равном 0,5, что позволяет избежать схождения к локальному минимуму.

Обучение нейронной сети осуществляется по эпохам. Перед началом очередной эпохи тренировочная выборка полностью перемешивается. В рамках одной эпохи обучение ведется последовательно по примерам. Если после обработки очередного примера значение функции *CardiLoss* оказалось выше значения гиперпараметра *expected_loss*, то считается, что нейронная сеть дала неправильный ответ. Обучение нейросети ведется до тех пор, пока в рамках одной эпохи нейросеть не даст правильные ответы на все слова из обучающей выборки.

Сбор векторов-состояний для последующей кластеризации

После того как нейронная сеть обучилась, начинается сбор векторов-состояний для последующей кластеризации. Для каждой строки w из списка слов-примеров рассмотрим список ее префиксов, состоящий из $|w| + 1$ строк. Каждый из них обрабатывается с помощью обученной рекуррентной нейронной сети и добавляется в список векторов-состояний.

Различные слова могут иметь совпадающие префиксы. Таким образом, в списке векторов-состояний возможны одинаковые элементы. Это делается для того, чтобы в итоговой кластеризации больший вес (в смысле числа соответствующих им векторов) имели наиболее часто встречающиеся состояния.

Применение метода выбора и настройки гиперпараметров алгоритма кластеризации векторов-состояний на основе обучения с подкреплением

Известно большое число алгоритмов кластеризации, каждый из которых обладает гиперпараметрами, влияющими на его работу. Выбор конкретного алгоритма и значений его гиперпараметров будет определять итоговую кластеризацию.

В настоящей работе предлагается подбирать лучший алгоритм кластеризации и настраивать его гиперпараметры за счет решения задачи оптимизации качества кластеризации. В силу отсутствия математически корректного определения задачи кластеризации [21] универсальную меру качества разбиения выделить невозможно. В данной работе использовалось четыре меры качества кластеризации: *Silhouette*, *Calinski-Harabasz*, *OS* и *gD*41. Обоснование и описание выбора мер приведено в диссертации [22].

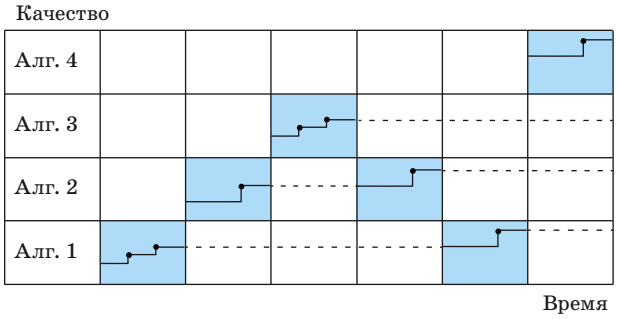
В рамках настоящей работы используется метод *MASSCAH* [23, 24], опишем его.

Рассмотрим некоторое множество алгоритмов кластеризации $\mathbf{A} = \{A^1, A^2, ..., A^m\}$, из которых будем осуществлять выбор. С каждым алгоритмом A^j связано пространство гиперпараметров Λ^j. Необходимо за фиксированное время T найти алгоритм $A^*_{\lambda^*}$ такой, что $A^*_{\lambda^*} \in \text{argmax}_{A^j \in \mathbf{A}, \lambda \in \Lambda^j} Q(A^j_\lambda, D)$. Критерий качества Q — значение одной из множества мер качества кластеризации; D — набор данных.

Рассмотрим алгоритм настройки гиперпараметров [25], который является итеративным (*sequential*) и позволяет выбирать новые значения гиперпараметров на основе истории их выбора настройки. Разобьем временной бюджет T на равные интервалы. В начале каждого из них метод будет определять, для какого алгоритма кластеризации из заданного множества $\mathbf{A}$ будет производиться настройка. В подобной постановке задача определения, какой алгоритм настраивать следующим, эквивалентна задаче о многоруком бандите [26]. Иллюстрация работы метода представлена на рис. 3.

Решение о том, какую из ручек выбирает алгоритм обучения с подкреплением, принимается с помощью алгоритма *Softmax* [26], которому на итерации k подается на вход следующий вещественный вектор $\mathbf{X}$:

$$\mathbf{X} = \text{Softmax}(\mathbf{R}^k) + \text{Softmax}(\mathbf{U}^k),$$



■ ***Рис. 3.*** Иллюстрация работы предложенного метода MASSCAH настройки гиперпараметров

■ ***Fig. 3.*** Schematic representation of operation of the presented method MASSCAH on hyperparameters tuning

где $\mathrm{Softmax}(x) = \left(\frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_i}} \right)_{i=1..n}$; $\mathbf{R}^k = (r_i^k)_{i=1..n}$ — вектор наград, получаемых каждой ручкой обучения с подкреплением; $\boldsymbol{U}^k = \left(\sqrt{\frac{2\ln\left(n + \sum_{j=1}^{n} \ln t_j^k\right)}{1 + \ln t_j^k}} \right)_{i=1..n}$ — вектор поправок, которые вносятся в соответствии с затраченным временем; t_i^k — время, которое было потрачено на вычисления целевой функции для ручки i до итерации k, с.

В данной работе использовалось семь моделей алгоритмов кластеризации, составляющих множество **A**: k-средних, *DBSCAN*, алгоритм распространения похожести, алгоритм агломеративной иерархической кластеризации, алгоритм среднего сдвига (*Mean Shift*), гауссова смесь и байесовская гауссова смесь. Реализации всех семи алгоритмов и соответствующие им множества гиперпараметров были заимствованы из библиотеки *scikit-learn* [27]. В качестве итеративного алгоритма выбора гиперпараметров использовался алгоритм *SMAC* [28].

Построение ДКА по построенной кластеризации

Предполагается, что каждый полученный кластер соответствует некоторому состоянию автомата. Для каждого вектора-состояния нейросети рассмотрим кластер c, в который он попал, и соответствующее ему слово w. Если это слово содержится в исходном наборе размеченных слов-примеров, то пометим кластер c в соответствии с этой разметкой. Если этот кластер уже помечен иначе, то обнаружено противоречие, и попытка построения считается проваленной. В противном случае рассмотрим слово w_0, являющееся вторым по длине префиксом строки w. При этом $w = w_0 + a_0$, где $a_0 = w[|w|]$ — последний символ слова w. Рассмотрим соответствующий w_0 кластер c_0. Построим ребро из состояния, соответствующего c_0, в состояние, соответствующее c, по символу a_0. Если же из состояния, соответствующего c_0, по символу a_0 уже имеется ребро, которое ведет не в c, то обнаружено противоречие, и попытка построения автомата считается проваленной.

Если при переборе векторов-состояний попытка построить автомат не будет провалена, то в результате работы приведенного алгоритма будет построен ДКА.

После этого проверим корректность построенного автомата. Переберем все слова из списка слов-примеров и применим к ним полученный автомат. Если для каждого слова он завершает свою работу в правильном состоянии — в состоянии, соответствующем разметке, то этим доказывается правильность построенного автомата.

Эксперименты

Тестовые данные

Значительная часть мировых исследований, посвященных проблеме индуцирования грамматик, использует для тестирования наборы данных, предложенные в работе [29]. Однако они соответствуют только грамматикам, содержащим не более двух символов в алфавитах, к тому же некоторые из этих грамматик не являются регулярными. Для полноценного тестирования модели необходимо использовать только регулярные грамматики, содержащие различное число символов в алфавите. Для тестирования описанной модели был разработан такой набор регулярных грамматик и, соответственно, набор размеченных списков слов-примеров.

Было разработано 10 регулярных грамматик, и для каждой из них был вручную построен минимальный ДКА. Построенные автоматы имеют от трех до десяти состояний и от двух до пяти символов в алфавитах. Далее был применен алгоритм генерации набора данных. По целевым автоматам и по сформированным наборам параметров для генерации были синтезированы 10 наборов слов-примеров, соответствующих регулярным грамматикам. Каждое слово в таком наборе помечено нулем или единицей в зависимости от принадлежности слова целевому регулярному языку. Описание разработанных автоматов и количество слов в соответствующих наборах данных представлены в табл. 1.

■ ***Таблица 1***. Тестовые регулярные грамматики и соответствующие наборы данных
■ ***Table 1***. The description of test regular grammars and corresponding datasets

ID	Описание регулярного языка	Число		
		символов в алфавите	состояний в минимальном автомате	слов в наборе данных
D1	Слова, которые не содержат одинаковых символов на соседних позициях	2	4	500
D2	Слова, которые не содержат одинаковых символов на соседних позициях	3	5	520
D3	Слова, длина которых кратна трем	3	3	586
D4	Слова, содержащие подстроку '*bac*'	3	4	580
D5	Слова, принадлежащие регулярному языку, соответствующему автомату на рис. 1	3	4	500
D6	Слова, содержащие подстроку '*abacaba*'	3	8	1200
D7	Слова, символы в которых отсортированы лексикографически	5	6	1200
D8	Слова, содержащие не более двух согласных или одной гласной подряд	5	5	1200
D9	Слова, длина которых кратна двум или пяти	3	10	1200
D10	Слова, не содержащие символа '*b*', длина которых по модулю 3 равна единице	5	4	1200

Программная реализация и тестирование

Классифицирующая рекуррентная нейронная сеть была реализована на языке *Python* 3 с использованием специализированной библиотеки машинного обучения *PyTorch*. При обучении сети на каждом из наборов данных использовались параметры $n\_neurons = 16$ и $expected\_loss = 0{,}02$.

Алгоритм одновременных выбора и настройки алгоритма кластеризации для получения разбиения векторов-состояний был реализован на языке *Python* версии 3.7.5. Для реализации данного алгоритма использовались библиотеки *SMAC* [30] и *ConfigSpace* [31]. Каждой паре «мера качества — набор данных» отводился временной бюджет 5000 с.

Вычисления проводились на двух серверах, которые при помощи специального планировщика распределяли задачи. Параметры серверов:

— 64-ядерный процессор *AMD Opteron* 6378 @ 2.4 GHz, 256 GB RAM;

— 64-ядерный процессор *AMD Opteron* 6380 @ 2.5 GHz, 496 GB RAM.

Алгоритм построения ДКА был реализован на языке *Python* версии 3.7.5. Если на основе полученной кластеризации удавалось построить непротиворечивый ДКА, корректно обрабатывающий каждое слово из списка слов-примеров, то попытка построения ДКА считалась успешной.

Результаты экспериментов

Из десяти автоматов, разработанных для поставленной задачи, семь были построены правильно, причем четыре из них получились минимально возможными. Приведем полную сводную таблицу результатов с указанием использованной меры настройки гиперпараметров для кластеризации (табл. 2).

■ ***Таблица 2***. Результаты тестирования предлагаемой модели
■ ***Table 2***. Results of the proposed model's testings

ID	Удалось ли построить корректный ДКА?	Число состояний		Примененная мера настройки гиперпараметров
		в минимальном автомате	в построенном моделью автомате	
D1	+	4	4	*gD*41
D2	+	5	7	*gD*41
D3	+	3	3	*gD*41
D4	+	4	8	*gD*41
D5	+	4	4	*gD*41
D6	–	8	–	–
D7	–	6	–	–
D8	+	5	5	*gD*41
D9	–	10	–	–
D10	+	4	5	*Calinski-Harabasz*

Для некоторых наборов данных, например для соответствующего языку D10, удалось построить корректный ДКА, верно обрабатывающий все строки-примеры, который, однако, не является минимальным по числу состояний. На рис. 4 изображен минимально возможный автомат, соответствующий языку D10, а на рис. 5 — автомат, построенный предложенным методом по словам-примерам из D10. Отметим, что эти автоматы эквивалентны — первый получается из второго слиянием состояний 3 и 0 и перенумерацией состояний, никак не влияющей на структуру автомата. Слияние состояний 3 и 0 допустимо, поскольку их терминальность совпадает, а переходы по одним и тем же символам ведут в одни и те же состояния.

Результаты экспериментов показали, что оптимальной для небольших по числу переходов автоматов (не превышающих 20–25 переходов) является мера *gD*41. Однако при ее использовании оказалось невозможным построить непротиворечивый автомат для D10. Соответствующую регулярную грамматику удалось индуцировать при использовании меры *Calinski-Harabasz*.

Автоматы, соответствующие наборам данных D6, D7 и D9, не удалось построить ни для каких мер. В каждом из случаев произошло одно из двух событий:

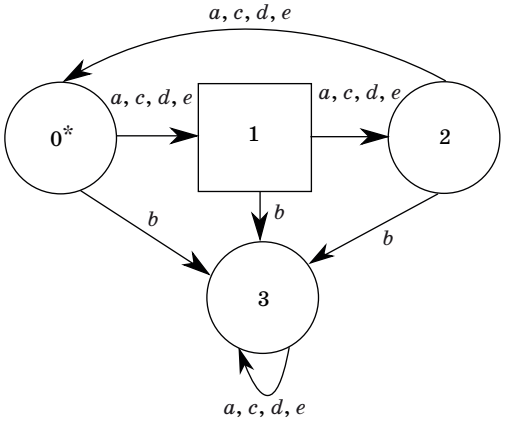
А) полученная кластеризация определяет число состояний, которое меньше числа состояний у минимально возможного автомата, соответствующего искомой регулярной грамматике;

Б) на основе полученной кластеризации невозможно построить непротиворечивый ДКА.

Детали тестирования модели на этих наборах данных отображены в табл. 3.

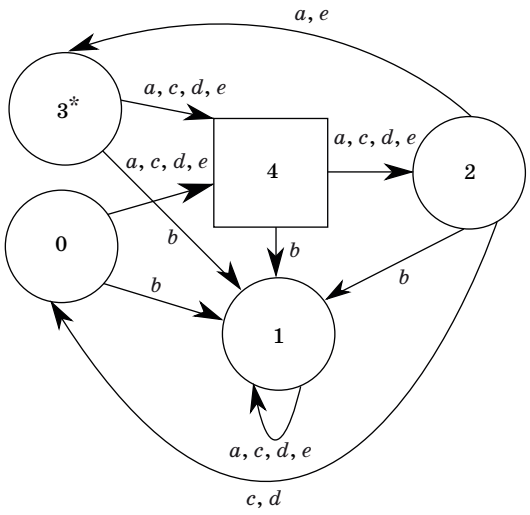
Поскольку для тестирования модели был сгенерирован специальный набор данных, то результаты экспериментов сложно сопоставимы с результатами аналогичных исследований других авторов. Однако предложенный метод показал лучшие результаты по сравнению с методом из статьи [30], в которой использовались те же наборы данных.

Сравнение полученных результатов с результатами из статьи [19], считающейся одной из наиболее актуальных работ последних лет по задаче индуцирования грамматик, показывает, что размеры получаемых автоматов сопоставимы. Однако нейронная сеть, которая используется в настоящей работе, содержит меньше настраиваемых параметров и быстрее обучается, чем используемая авторами статьи [19] сеть *LSTM*. Кроме того, авторы [19] считают положительным результатом построение автомата, чья точность на тренировочном наборе данных отличается от единицы (например, равна 0,99), в то время как в настоящей работе корректным считается только автомат, имеющий точность, равную единице.



■ ***Рис. 4.*** Целевой ДКА для языка D10

■ ***Fig. 4.*** Target DFA for the D10 language



■ ***Рис. 5.*** Автомат, корректно обрабатывающий все слова из набора данных для языка D10, построенный моделью

■ ***Fig. 5.*** Synthesized automaton which processes all the words from D10 dataset correctly

■ ***Таблица 3.*** Результаты тестирования на наборах данных D6, D7, D9

■ ***Table 3.*** Results of testings on D6, D7 & D9 datasets

Набор данных\ мера	*Calinski-Harabasz*	*gD*41	*OS*	*Silhouette*
D6	Б	А	А	Б
D7	Б	Б	А	Б
D9	Б	Б	А	А

Заключение

Полученные результаты показывают эффективность предлагаемого метода и применимость метода выбора и настройки гиперпараметров алгоритма кластеризации на основе обучения с подкреплением к задаче индуцирования грамматики.

Усложнение грамматик (свыше 25–30 переходов) приводит к тому, что автоматы не удается синтезировать. Для улучшения результатов модели можно предпринять следующие шаги:

1) использовать рекуррентные нейросети иных структур в качестве *RNN*-акцептора (к примеру, *LSTM* или модели с вниманием);

2) использовать иные меры настройки гиперпараметров для алгоритма кластеризации;

3) разработать и применить алгоритм, устраняющий противоречия в кластеризации в пользу наиболее правдоподобного варианта.

Финансовая поддержка

Данное исследование выполнено при поддержке проекта государственного задания №2.8866.2017/БЧ «Технология разработки программного обеспечения систем управления ответственными объектами на основе глубокого обучения и конечных автоматов», выполняемого в рамках базовой части государственного задания.

Литература

1. **Bahlke R., Snelting G.** The PSG system: from formal language definitions to interactive programming environments. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1986, vol. 8, no. 4, pp. 547–576.
2. **Van Fraassen B. C.** *Formal semantics and logic*. Macmillan, 1971. 343 p.
3. **Hoare C. A. R., Wirth N.** An axiomatic definition of the programming language PASCAL. *Acta Informatica*, 1973, vol. 2, no. 3, pp. 335–355.
4. **Kate R. J., Wong Y. W., Mooney R. J.** Learning to transform natural to formal languages. *AAAI*, 2005, pp. 1062–1068.
5. **Jäger G., Rogers J.** Formal language theory: refining the Chomsky hierarchy. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 2012, vol. 367, no. 1598, pp. 1956–1970.
6. **Gribkoff E.** *Applications of deterministic finite automata*. https://web.cs.ucdavis.edu/~rogaway/classes/120/spring13/eric-dfa.pdf (дата обращения: 15.12.2019).
7. **Sirakoulis G. C., Karafyllidis I., Mizas C., Mardiris V., Thanailakis A., Tsalides P.** A cellular automaton model for the study of DNA sequence evolution. *Computers in Biology and Medicine*, 2003, vol. 33, no. 5, pp. 439–453.
8. **Meng J. P., Dai S. Q., Dong L. Y., Zhang J. F.** Cellular automaton model for mixed traffic flow with motorcycles. *Physica A: Statistical Mechanics and its Applications*, 2007, no. 380, pp. 470–480.
9. **Filiol E.** *Computer viruses: from theory to applications*. Springer Science & Business Media, 2006. 417 p.
10. **Sakarovitch J.** *Elements of automata theory*. Cambridge University Press, 2009. 785 p.
11. **Angluin D.** Inference of reversible languages. *Journal of the ACM*, 1982, vol. 29, no. 3, pp. 741–765.
12. **Gold E. M.** Language identification in the limit. *Information and Control*, 1967, vol. 10, no. 5, pp. 447–474.
13. **Yu F., Bultan T., Ibarra O.** Relational string verification using multi-track automata. *Proceedings of the 15th International Conference "Implementation and Application of Automata"*, Manitoba, Canada, 2010, pp. 1909–1924.
14. **De la Higuera C.** *Grammatical inference: learning automata and grammars*. Cambridge University Press, 2010. 417 p.
15. **Barzdin I. M.** *Finite automata-behavior and synthesis*. North-Holland Publishing Company, 1973. 321 p.
16. **Giles C. L., Sun G. Z., Chen H. H., Lee Y. C., Chen D.** Higher order recurrent networks and grammatical inference. *Advances in Neural Information Processing Systems*, 1990, vol. 2, pp. 380–387.
17. **Gile C. L., Miller C. B., Chen D., Sun G. Z., Chen H. H., Lee Y. C.** Extracting and learning an unknown grammar with recurrent neural networks. *Proceedings of the 3rd Conference "Advances in Neural Information Processing Systems"*, 1992, pp. 317–324.
18. **Omlin C. W., Giles C. L.** Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the ACM*, 1996, vol. 43, no. 6, pp. 937–972.
19. **Weiss G., Goldberg Y., Yahav E.** *Extracting automata from recurrent neural networks using queries and counterexamples*. https://arxiv.org/pdf/1711.09576.pdf (дата обращения: 15.12.2019).
20. **Hopcroft J. E., Motwani R., Ullman J. D.** Introduction to automata theory, languages, and computation. *Acm Sigact News*, 2001, vol. 32, no. 1, pp. 60–65.
21. **Hennig C.** What are the true clusters? *Pattern Recognition Letters*, 2015, no. 64, pp. 53–62.
22. **Муравьев С. Б.** *Система автоматического выбора и оценки алгоритмов кластеризации и их параметров*: дис. ... канд. техн. наук: 05.13.17. Санкт-Петербург, Университет ИТМО, 2019. 308 с.
23. **Shalamov V., Efimova V., Muravyov S., Filchenkov A.** Reinforcement-based method for simultaneous clustering algorithm selection and its hyperparameters optimization. *Procedia Computer Science*, 2018, no. 136, pp. 144–153.

24. **Муравьев С. Б., Ефимова В. А., Шаламов В. В., Фильченков А. А., Сметанников И. Б.** Автоматическая настройка гиперпараметров алгоритмов кластеризации с помощью обучения с подкреплением. *Научно-технический вестник информационных технологий, механики и оптики*, 2019, т. 19. вып. 3, с. 508–515.
25. **Eggensperger K., Feurer M., Hutter F., Bergstra J., Snoek J., Hoos H., Leyton-Brown E.** Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. *NIPS Workshop on Bayesian Optimization in Theory and Practice*, 2013, vol. 10, pp. 1–5.
26. **Sutton R. S.** *Reinforcement learning.* MIT Press, 1998. 1054 p.
27. **Pedregosa F.** Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011, no. 12, pp. 2825–2830.
28. **Hutter F., Hoos H., Leyton-Brown K.** An evaluation of sequential model-based optimization for expensive blackbox functions. *Genetic and Evolutionary Computation Conference*, Amsterdam, July 2013, pp. 1209–1216.
29. **Tomita M.** Learning of construction of finite automata from examples using hill-climbing. RR: regular set recognizer. *Proceedings of Fourth International Cognitive Science Conference*, 1982, pp. 105–108.
30. **Hutter F., Hoos H. H., Leyton-Brown K.** Sequential model-based optimization for general algorithm configuration. *Learning and Intelligent Optimization. Springer*, 2011, pp. 507–523.
31. **Lindauer M., Eggensperger K., Feurer M., Biedenkapp A., Marben J., Müller P., Hutter F.** *BOAH: A tool suite for multi-fidelity bayesian optimization & analysis of hyperparameters.* https://arxiv.org/pdf/1908.06756v1.pdf (дата обращения: 15.12.2019).

UDC 519.713.1
doi:10.31799/1684-8853-2020-1-34-43

Automata generation based on recurrent neural networks and automated cauterization selection

P. G. Grachev[a], Post-Graduate Student, orcid.org/0000-0002-1133-8432
S. B. Muravyov[a], PhD, Tech., Programmer, orcid.org/0000-0002-4251-1744
A. A. Filchenkov[a], PhD, Phys.-Math., Associate Professor, orcid.org/0000-0002-1133-8432
A. A. Shalyto[a], Dr. Sc., Tech., Professor, orcid.org/0000-0002-2723-2077, shalyto@mail.ifmo.ru
[a]Saint-Petersburg National Research University of Information Technologies, Mechanics and Optics, 49, Kronverkskii Pr., 197101, Saint-Petersburg, Russian Federation

Intoduction: The regular inference problem is to synthesize deterministic finite-state automata by a list of words which are examples and counterexamples of some unknown regular language. This problem is one of the main in the theory of formal languages and related fields. One of the most successful solutions to this problem is training a recurrent neural network on word classification and clustering the vectors in the space of RNN inner weights. However, it is not guaranteed that a consistent automaton can be constructed based on the clustering results. More complex models require more memory, training time and training samples. **Purpose:** Creating a brand new grammar inference algorithm which would use modern machine learning methods. **Methods:** A recurrent neural network with an error function proposed by the authors was used for classification. For clustering, the method of joint selection and tuning of hyperparameter was used. **Results:** Ten different datasets were used for testing the models, corresponding to ten different regular grammars and ten automata. According to the test results, the developed model successfully synthesize automata with no more than five input characters and states. For four grammars, out of the seven successfully inferred ones, the constructed automaton was minimal. For three datasets, an automaton could not be built, either because of an insufficient number of clusters in the proposed partition, or because of the inability to build a consistent automaton for this partition. **Discussion:** Applying the algorithm of search for maximum likelihood between the clusters of vector and the corresponding states in order to resolve structural conflicts may expand the scope of the model.
Keywords — grammar inference, recurrent neural networks, clusterization.

For citation: Grachev P. G., Muravyov S. B., Filchenkov A. A., Shalyto A. A. Automata generation based on recurrent neural networks and automated cauterization selection. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2020, no. 1, pp. 34–43 (In Russian). doi:10.31799/1684-8853-2020-1-34-43

References

1. Bahlke R., Snelting G. The PSG system: from formal language definitions to interactive programming environments. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1986, vol. 8, no. 4, pp. 547–576.
2. Van Fraassen B. C. *Formal semantics and logic.* Macmillan, 1971. 343 p.
3. Hoare C. A. R., Wirth N. An axiomatic definition of the programming language PASCAL. *Acta Informatica*, 1973, vol. 2, no. 3, pp. 335–355.
4. Kate R. J., Wong Y. W., Mooney R. J. Learning to transform natural to formal languages. *AAAI*, 2005, pp. 1062–1068.
5. Jäger G., Rogers J. Formal language theory: refining the Chomsky hierarchy. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 2012, vol. 367, no. 1598, pp. 1956–1970.
6. Gribkoff E. Applications of deterministic finite automata. *UC Davis*, 2013. Available at: https://web.cs.ucdavis.edu/~rogaway/classes/120/spring13/eric-dfa.pdf (accessed 15 December 2019).
7. Sirakoulis G. C., Karafyllidis I., Mizas C., Mardiris V., Thanailakis A., Tsalides P. A cellular automaton model for the study of DNA sequence evolution. *Computers in Biology and Medicine*, 2003, vol. 33, no. 5, pp. 439–453.

8. Meng J. P., Dai S. Q., Dong L. Y., Zhang J. F. Cellular automaton model for mixed traffic flow with motorcycles. *Physica A: Statistical Mechanics and its Applications*, 2007, no. 380, pp. 470–480.
9. Filiol E. *Computer viruses: from theory to applications*. Springer Science & Business Media, 2006. 417 p.
10. Sakarovitch J. *Elements of automata theory*. Cambridge University Press, 2009. 785 p.
11. Angluin D. Inference of reversible languages. *Journal of the ACM*, 1982, vol. 29, no. 3, pp. 741–765.
12. Gold E. M. Language identification in the limit. *Information and Control*, 1967, vol. 10, no. 5, pp. 447–474.
13. Yu F., Bultan T., Ibarra O. Relational string verification using multi-track automata. *Proceedings of the 15th International Conference "Implementation and Application of Automata"*, Manitoba, Canada, 2010, pp. 1909–1924.
14. De la Higuera C. *Grammatical inference: learning automata and grammars*. Cambridge University Press, 2010. 417 p.
15. Barzdin I. M. *Finite automata-behavior and synthesis*. North-Holland Publishing Company, 1973. 321 p.
16. Giles C. L., Sun G. Z., Chen H. H., Lee Y. C., Chen D. Higher order recurrent networks and grammatical inference. *Advances in Neural Information Processing Systems*, 1990, pp. 380–387.
17. Gile C. L., Miller C. B., Chen D., Sun G. Z., Chen H. H., Lee Y. C. Extracting and learning an unknown grammar with recurrent neural networks. *Proceedings of the 3rd Conference "Advances in Neural Information Processing Systems"*, 1992, pp. 317–324.
18. Omlin C. W., Giles C. L. Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the ACM*, 1996, vol. 43, no. 6, pp. 937–972.
19. Weiss G., Goldberg Y., Yahav E. *Extracting automata from recurrent neural networks using queries and counterexamples*. Available at: https://arxiv.org/pdf/1711.09576.pdf (accessed 15 December 2019).
20. Hopcroft J. E., Motwani R., Ullman J. D. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 2001, vol. 32, no. 1, pp. 60–65.
21. Hennig C. What are the true clusters? *Pattern Recognition Letters*, 2015, no. 64, pp. 53–62.
22. Muravyov S. B. *Sistema avtomaticheskogo vybora i otsenki algoritmov klasterizatsii i ih parametrov*. Dis. kand. tehn. nauk: 05.13.17 [Automated system for selection and evaluation clusterization algorithms and their parametres, PhD tech. sci. diss.]. Saint-Petersburg, ITMO University, 2019. 308 p. (In Russian).
23. Shalamov V., Efimova V., Muravyov S., Filchenkov A. Reinforcement-based method for simultaneous clustering algorithm selection and its hyperparameters optimization. *Procedia Computer Science*, 2018, no. 136, pp. 144–153.
24. Muravyov S. B., Efimova V. A., Shalamov V. V., Filchenkov A. A., Smetannikov I. B. Automated hyperparameters tuning for clusterization algorithm using reinforcement learning. *Nauchno-tehnicheskiy vestnik informatsionnyh tehnologiy, mehaniki i optiki*, 2019, vol. 19, no. 3, pp. 508–515 (In Russian).
25. Eggensperger K., Feurer M., Hutter F., Bergstra J, Snoek J., Hoos H., Leyton-Brown E. Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. *NIPS Workshop on Bayesian Optimization in Theory and Practice*, 2013, vol. 10, pp. 1–5.
26. Sutton R. S. *Reinforcement learning*. MIT Press, 1998. 1054 p.
27. Pedregosa F. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011, no. 12, pp. 2825–2830.
28. Hutter F., Hoos H., Leyton-Brown K. An evaluation of sequential model-based optimization for expensive blackbox functions. *Genetic and Evolutionary Computation Conference*, Amsterdam, July 2013, pp. 1209–1216.
29. Tomita M. Learning of construction of finite automata from examples using hill-climbing: RR: regular set recognizer. *Proceedings of Fourth International Cognitive Science Conference*, 1982, pp. 105–108.
30. Hutter F., Hoos H. H., Leyton-Brown K. Sequential model-based optimization for general algorithm configuration. *Learning and Intelligent Optimization. Springer*, 2011, pp. 507–523.
31. Lindauer M., Eggensperger K., Feurer M., Biedenkapp A., Marben J., Müller P., Hutter F. *BOAH: A tool suite for multi-fidelity bayesian optimization & analysis of hyperparameters*. Available at: https://arxiv.org/pdf/1908.06756v1.pdf (accessed 15 December 2019).