Международный журнал информационных технологий и энергоэффективности

Том 6 Номер 3 (21)

2021

Международный журнал информационных технологий и энергоэффективности
Т.6 №3 (21) 2021 г.

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

СОДЕРЖАНИЕ / CONTENT

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Нагорных М. Э., Быков А.Н., Чернышёв С. А.. Программное обеспечение распознавания азбуки глухонемых // Международный журнал информационных технологий и энергоэффективности. – 2021. – Т. 6 № 3(21) с. 3–10

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.058:004.93

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РАСПОЗНАВАНИЯ АЗБУКИ ГЛУХОНЕМЫХ

[1] Нагорных М. Э., [2] Быков А.Н., [3,4] Чернышёв С. А.

[1,2]Санкт-Петербургский государственный университет аэрокосмического приборостроения, Россия (190000, г. Санкт-Петербург, ул. Большая Морская, 67, лит. А),
e-mail: [1] nagornykh_max@mail.ru, [2] alexey_bykovoff@mail.ru
[3] Санкт-Петербургский государственный университет промышленных технологий и дизайна, Россия (191186, г. Санкт-Петербург, ул. Большая Морская, 18)
[4] Санкт-Петербургский государственный экономический университет, Россия (191023, г. Санкт-Петербург, ул. Садовая, 21), e-mail: chernyshev.s.a@bk.ru

Последнее время наблюдается социальная направленность во внутренней политике различных государств. Особенно это заметно по требованиям, предъявляемым к образовательным организациям. Не должно существовать разделения между обычными студентами и студентами с ограничениями по здоровью. В связи с этим на первый план выходит инклюзивный подход к организации образовательного процесса. К сожалению, не все образовательные организации могут позволить себе специальное оборудование, необходимое для проведения занятий с участием студентов с ограничениями по здоровью. В статье подробно описывается процесс разработки программного обеспечения для распознавания азбуки глухонемых. Приводится краткий обзор и обоснование выбора модели обучения нейронной сети, а также продемонстрирован полученный результат.

Ключевые слова: глухонемые, нейросеть, Faster-RCNN, Python, Tensorflow.

DEAF-MUTE ALPHABET RECOGNITION SOFTWARE

[1]Nagornykh M.E., [2] Bykov A.N., [3,4] Chernyshev S.A.

[1,2] Saint Petersburg State University of Aerospace Instrumentation, Russia (190000, Saint Petersburg, Bolshaya Morskaya st., 67, letter A),
e-mail: [1] nagornykh_max@mail.ru, [2] alexey_bykovoff@mail.ru
[3] Saint Petersburg State University of Industrial Technology and Design, Russia (191186, Saint-Petersburg, Bolshaya Morskaya st., 18)
[4] Saint Petersburg State University of Economics, Russia (191023, Saint Petersburg, Sadovaya st., 21), e-mail: chernyshev.s.a@bk.ru

Recently, there has been a social orientation in the internal politics of various states. This is especially noticeable in the requirements for educational organizations. There should be no division between ordinary students and students with disabilities. In this regard, an inclusive approach to the organization of the educational process comes to the fore. Unfortunately, not all educational organizations can afford the special equipment required to conduct classes with students with disabilities. The article describes in detail the process of developing software for recognizing the alphabet of the deaf-mute. A brief overview and justification of the choice of a neural network training model are given, and the result obtained is demonstrated.

Keywords: deaf-mute, neural network, Faster-RCNN, Python, Tensorflow.

Введение

Большинство людей не понимают язык глухонемых, из-за чего при общении с глухонемыми может возникать проблема взаимопонимания в простых жизненных ситуациях. Примером таких ситуаций могут выступать: общение между преподавателем и учеником, обращение в медицинское учреждение или даже при покупках в магазинах и т.д. [1-3].

С развитием технологий появляется все больше возможностей для решения этих проблем. В статье представлен процесс разработки приложения, которое используя веб-камеру компьютера, производит конвертацию языка жестов в текст. Это достигается посредством использования нейронных сетей и машинного обучения. Данная технология активно развивается последние десятилетия и набирает популярность не только при решении промышленных, но и социально-значимых задач [4-9].

Создание такого программного обеспечения позволяет приблизиться к решению проблемы коммуникации, появляющейся при общении с глухонемыми людьми. Данная тема имеет остросоциальную необходимость, поскольку общество стремиться к тому, чтобы люди не чувствовали себя ограниченными в своих возможностях.

1. Принцип работы.

Алгоритм работы разработанного приложения состоит из следующих стадий:

- Выбор используемой веб-камеры из списка доступных на компьютере;
- Получение изображения из видео потока;
- Анализ его с помощью обученной нейросети, получение координат расположения жеста в видеопотоке и его классификация;
- Отображение распознанного символа в конце списка для составления слова.

2. Выбор нейронной сети.

На сегодняшний день существует множество программных решений для создания и обучения нейронных сетей. В данном проекте использовалась Tensorflow. Эта библиотека привлекает пользователей своим широким спектром возможностей для решения задач классификации и активно поддержи поддерживается как её разработчиками, так и сообществом программистов [10, 11].

TensorFlow предоставляет несколько моделей обнаружения объектов (предварительно обученные классификаторы с конкретными архитектурами нейронных сетей). Некоторые модели (например, модель SSD-MobileNet) имеют архитектуру, которая обеспечивает более быстрое обнаружение, но с меньшей точностью, в то время как некоторые модели (например, модель Faster-RCNN) обеспечивают более медленное обнаружение, но большую точность.

Система Faster-RCNN состоит из двух модулей: первый модуль — это сверточная нейронная сеть, которая находит потенциальные объекты на изображении и разбивает их на области, а второй модуль — это Fast-RCNN, который использует предложенные области и является детектором. Вся система представляет собой единую унифицированную сеть для обнаружения объектов.

Faster-RCNN быстрее и лучше выполняет задачи поиска и классификации, чем ее предшественники. На сегодняшний день она широко используется и остается одним из лучших решений доступных пользователю. Именно поэтому в качестве модели обучения использовалась Faster-RCNN [12].

3. Процесс разработки программного обеспечения

На самой первой стадии был сформирован набор обучающих данных, где каждый экземпляр представлял из себя фотографию с указанным расположением жеста и его значением. Для этих целей использовалось приложение LabelImg [13], работа в котором представлена на рисунке 1.

Рисунок 1 - Пример обработки обучающего фото в LabelImg

Фотографии были сделаны на различном фоне и при различных условиях освещения. Это позволило избежать нахождения нейросетью ненужных взаимосвязей и сконцентрировать ее внимание на анализе жестов кистей рук.

В результате было сделано 1650 фотографий по 50 на каждый жест, из них 30% было выделено для тестовых данных.

Из набора маркированных изображений были сгенерированы TFRecords файлы, которые служат входными данными для модели обучения TensorFlow. На следующем шаге создавалась карта меток, на которой указывалось соответствие id объекта с его наименованием [14]. Пример такой карты представлен в Таблице 1.

Таблица 1 – Карта меток.

item{
id: 1
name: ‘A’
}
 	
item{
id: 2
name: ‘B’
}
	item{
	id: 7
	name: ‘G’
	}
	
	item{
	id: 8
	name: ‘H’
	}
		item{
		id: 13
		name: ‘M’
		}
		
		item{
		id: 14
		name: ‘N’
		}

item{
id: 3
name: ‘C’
}
 	
item{
id: 4
name: ‘D’
}
 	
item{
id: 5
name: ‘E’
}
 	
item{
id: 6
name: ‘F’
}
	item{
	id: 9
	name: ‘I’
	}
	
	item{
	id: 10
	name: ‘G’
	}
	
	item{
	id: 11
	name: ‘K’
	}
	
	item{
	id: 12
	name: ‘L’
	}
		item{
		id: 15
		name: ‘O’
		}
		
		item{
		id: 16
		name: ‘P’
		}
		
		item{
		id: 17
		name: ‘R’
		}
		
		item{
		id: 18
		name: ‘S’
		}

После выбора модели обучения и подготовки обучающей выборки следовало создание конфигурационного файла, в котором указывались параметры обучения: количество итераций, количество классов распознавания и пути к файлам обучения. В результате обучения нейросети, которое заняло 38 часов, был сформирован файл, содержащий весовые коэффициенты.

С помощью использования библиотеки OpenCV, в приложении реализовано получение видеопотока с веб-камеры компьютера [15]. Каждый кадр отправляется на обработку нейронной сетью, в которой вычисляются координаты области нахождения жеста и происходит его классификация. Если жест в течение 1.5 секунды встречается на кадрах с частотой выше 90%, он заносится в поле для индикации показанных жестов.

Для реализации пользовательского интерфейса, который представлен на рисунке 2, использовался кроссплатформенный фреймворк Qt, а точнее его реализация для языка программирования Python - PyQt.

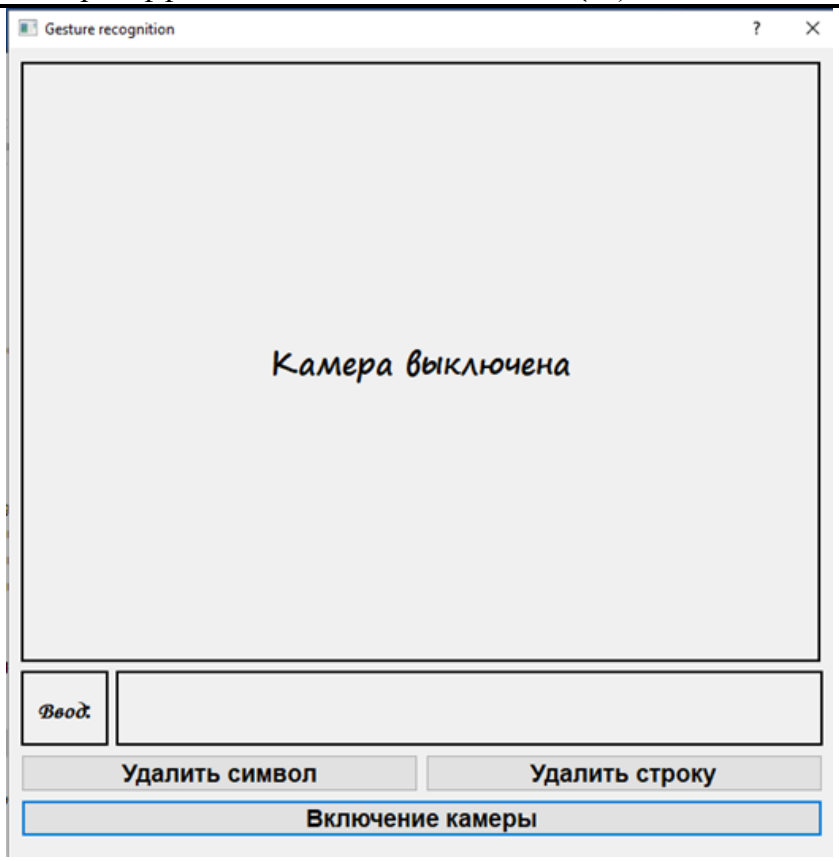

Рисунок 2 - Интерфейс приложения

Кнопка «Включить камеру» запускает видеопоток в приложении отдельно от потока обработки кадров и выводит изображение камеры на интерфейс. В случае возникновения ошибки со стороны пользователя или программы, нажатие на кнопку «Удалить символ» удаляет последний символ из слова. Аналогичную функцию выполняет кнопка «Удалить слово», после нажатия которой, происходит полное очищение строки поля вывода.

4. Демонстрация работы

Пример ввода слов продемонстрирован на рисунках 3 и 4.

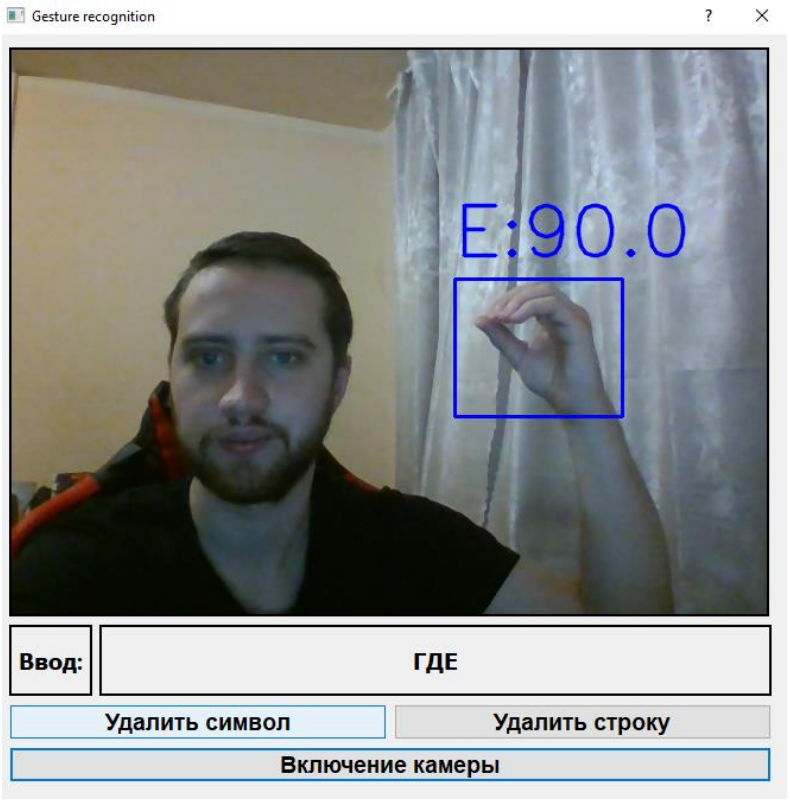

Рисунок 3 - Ввод слова «где» с помощью языка жестов

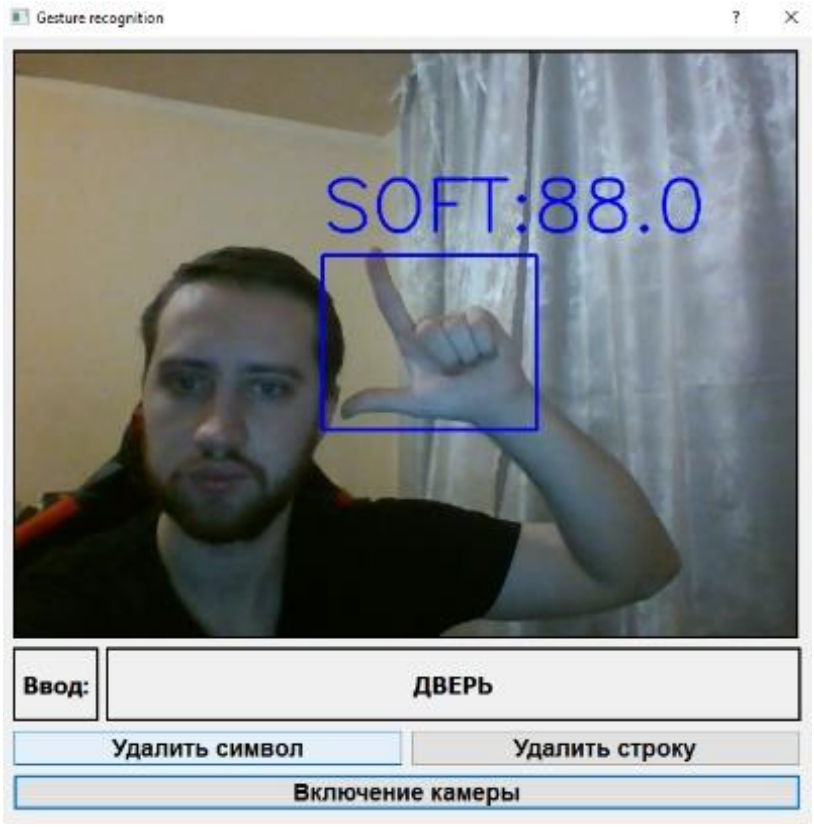

Рисунок 4 - Ввод слова «дверь» с помощью языка жестов

Заключение

Представленные на рисунках 3 и 4 примеры работы разработанного программного обеспечения демонстрируют, как оно находит и распознает жест кисти руки на изображении, преобразует его в текст и записывает в поле вывода на интерфейсе приложения.

Также напротив названия класса найденного объекта можно наблюдать процент вероятности, что нейросети смогла правильно классифицировать жест. На данных примерах минимальным пороговым значением является 98 процентов. Снижение этого значения, может вызвать ложные срабатывания и ошибки в классификации жеста.

В дальнейшим планируется использовать приложение для распознавания жестов, обозначающих отдельные слова или фразы, распознавания латинской азбуки жестов, а также преобразование написанного текста в звуковую запись с помощью голосового робота. Дополнительно, планируется использовать обученную нейронную сеть и отработанные методологии при написании мобильного приложения. Это позволит сделать серьезный шаг к решению проблемы коммуникации, появляющейся при общении с глухонемыми людьми.

Список литературы

1. М. С. Григорьев, А. В. Ляшков. Они нас не слышат (жизнь российских глухих и слабослышащих). Кучково поле, Москва, 2017 г.
2. Базоев В.З., Паленный В.А. Человек из мира тишины. – М.: Академкнига, 2002.
3. Базоев В. З., Гаврилова Е. Н., Егорова И. А., Ежова В. В., Давиденко Т. П., Чаушьян Н. А. Словарь русского жестового языка. М.: Флинта, 2009.
4. Галушкин, А.И. Нейронные сети: основы теории. М.: РиС, 2014.
5. Enis Berk Coban. Neural Networks and Their Applications. Istanbul Sehir University, 2016.
6. Nelson Piedra, Janneth Chicaiza, Jorge López, Jesús García. Study of the application of neural networks in internet traffic engineering. International Book Series "Information Science and Computing", 2008.
7. Adam Oken. An Introduction To and Applications of Neural Networks. 2017.
8. Zijie J. Wang, Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng (Polo) Chau. CNN EXPLAINER: Learning Convolutional Neural Networks with Interactive Visualization.
9. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. Погружение в мир нейронных сетей. Питер, 2018 г.
10. Peter Goldsborough. A Tour of TensorFlow. Fakultät für InformatikTechnische Universität München, 2016.
11. Tensorflow [Электронный ресурс] – Режим доступа: https://www.tensorflow.org
12. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.
13. LabelImg [Электронный ресурс] – Режим доступа: https://github.com/tzutalin/labelImg
14. Коэльо Л.П., Ричарт В. Построение систем машинного обучения на языке Python. ДМК Пресс, 2016.
15. Daniel Lélis Baggio, Shervin Emami, David Millán Escrivá, Khvedchenia Ievgen, Naureen Mahmood, Jason Saragih, Roy Shilkrot. Mastering OpenCV 3. - Второе издание, Packt Publishing, 2017 г.

References

1. M. S. Grigoriev, A. V. Lyashkov. They do not hear us (the life of Russian deaf and hard of hearing). Kuchkovo field, Moscow, 2017
2. Bazoev V.Z., Palenny V.A. A man from the world of silence. Akademkniga, 2002.
3. Bazoev V.Z., Gavrilova E.N., Egorova I.A., Ezhova V.V., Davidenko T.P., Chaushyan N.A. Dictionary of Russian Sign Language. Flinta, 2009.
4. Galushkin, A.I. Neural networks: foundations of the theory. Moscow: RiS, 2014.
5. Enis Berk Coban. Neural Networks and Their Applications. Istanbul Sehir University, 2016.
6. Nelson Piedra, Janneth Chicaiza, Jorge López, Jesús García. Study of the application of neural networks in internet traffic engineering. International Book Series "Information Science and Computing", 2008.
7. Adam Oken. An Introduction To and Applications of Neural Networks. 2017.
8. Zijie J. Wang, Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng (Polo) Chau. CNN EXPLAINER: Learning Convolutional Neural Networks with Interactive Visualization.
9. Nikolenko S., Kadurin A., Arkhangelskaya E. Deep learning. Dive into the world of neural networks. Peter, 2018
10. Peter Goldsborough. A Tour of TensorFlow. Fakultät für InformatikTechnische Universität München, 2016
11. Tensorflow Available at: https://www.tensorflow.org (accessed 7 December 2020).
12. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.
13. LabelImg Available at: https://github.com/tzutalin/labelImg (accessed 10 December 2020).
14. Coelho L.P., Richart V. Building machine learning systems in Python. DMK Press, 2016.
15. Daniel Lélis Baggio, Shervin Emami, David Millán Escrivá, Khvedchenia Ievgen, Naureen Mahmood, Jason Saragih, Roy Shilkrot. Mastering OpenCV 3. - Второе издание, Packt Publishing, 2017 г.

Челышев Э.А., Оцоков Ш.А., Раскатова М.В. Разработка информационной системы для автоматической рубрикации новостных текстов // Международный журнал информационных технологий и энергоэффективности. – 2021 – Т.6 №3(21) с.11-17

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.896

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ АВТОМАТИЧЕСКОЙ РУБРИКАЦИИ НОВОСТНЫХ ТЕКСТОВ

[1] Челышев Э.А., Оцоков Ш.А., Раскатова М.В.
Федеральное государственное бюджетное образовательное учреждение высшего образования «Национальный исследовательский университет «МЭИ», Россия (111250, г.Москва, ул. Красноказарменная, д.14); e-mail: [1] chel.ed@yandex.ru

В статье рассматривается проектирование и разработка информационной системы рубрикации русскоязычных текстов с использованием машинного обучения, состоящей из обученной модели классификации и веб-сайта. Выполнена подготовка текстовых данных. Проведен ряд экспериментов по обучению моделей классификации с использованием языка программирования Python. Обобщающая способность обученных моделей оценена по ряду метрик. Для реализации веб-сайта были использованы язык программирования Python и фреймворк Django, а также система управления базами данных MySQL.

Ключевые слова: информационная система, рубрикация, машинное обучение, классификация, метрика.

DEVELOPMENT OF INFORMATION SYSTEM FOR AUTOMATIC RUBRICATION OF NEWS TEXTS

Chelyshev E.A., Otsokov Sh. A., Raskatova M.V.
National Research University "Moscow Power Engineering Institute", Russia (111250, Moscow, Krasnokazarmennaya street, 14); e-mail: chel.ed@yandex.ru

In this paper, we consider the design and development of an information system for the rubrication of Russian-language texts using machine learning, consisting of a trained classification model and a website. The text data has been prepared. A number of experiments were conducted to train classification models using the Python programming language. The generalizing ability of the trained models is estimated by a number of metrics. To implement the website, the Python programming language and the Django framework were used, as well as the MySQL database management system.

Keywords: information system, rubrication, machine learning, classification, metric.

В последние десятилетия наблюдается быстрый рост объема производимых и накопляемых человечеством данных. Так, например, общемировой объем данных в 2018 году составил 33 зеттабайтов, а прогнозируемый общемировой объем данных к 2025 году составит уже 175 зеттабайтов [6], то есть вырастет более чем в пять раз.

Безусловно, с ростом накопленных данных человеку становится все сложнее ориентироваться в них, все более возрастает потребность в автоматизированной обработке информации. Весьма популярными становятся сейчас новостные агрегаторы, рубрикаторы научных статей и прочие решения по автоматизированной обработке информации, которые в своей работе используют автоматическую рубрикацию текстов на естественном языке. Она, в свою очередь, может быть быстро и качественно осуществлена в режиме реального времени с использованием алгоритмов машинного обучения. С точки зрения машинного обучения

рубрикация является задачей классификации на несколько непересекающихся классов, где под классом подразумевается отдельная рубрика [3]. В данной статье рассматриваются алгоритмы машинного обучения по прецедентам (с учителем).

Коллектив авторов поставил перед собой задачу разработки удобной для конечного пользователя информационной системы, которая бы в режиме реального времени могла осуществлять рубрикацию русскоязычных новостных текстов при помощи алгоритмов машинного обучения.

На рисунке 1 представлена наглядная схема взаимодействия компонентов разработанной системы. База данных осуществляет хранение новостных статей, система автоматической рубрикации, содержащая в себе обученную модель классификации, через некоторые фиксированные промежутки времени считывает из базы данных тексты новостных статей, подлежащих рубрикации, классифицирует их и изменяет содержимое базы данных, указывая для каждой статьи принадлежность к конкретной рубрике. Веб-сайт отображает результаты работы информационной системы, взаимодействуя с пользователем.

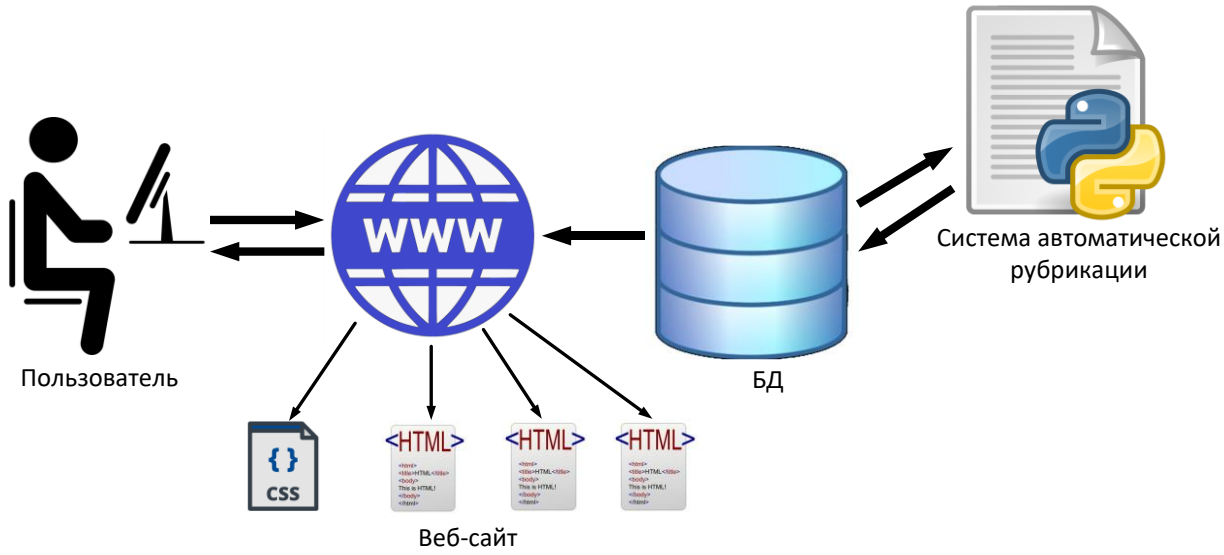

Рисунок 1 - Схема взаимосвязи компонентов разработанной системы

Для обучения моделей классификации был использован набор данных новостного интернет-портала LENTA.RU [5], из которого были выделены новостные статьи, соответствующие следующим девяти рубрикам: Дом, Интернет и СМИ, Культура, Наука и техника, Политика, Путешествия, Силовые структуры, Спорт, Экономика и бизнес. Этот набор данных в соответствии с принятой в машинной обработке естественного языка терминологией далее называется **корпусом** новостных статей.

Одним из важнейших этапов решения задач машинного обучения является подготовка данных. Подготовка данных в рассматриваемой задаче включает в себя ряд специфичных для машинной обработки естественного языка этапов, для реализации которых был разработан программный модуль на языке **Python**.

Вначале из текстов новостных статей с использованием регулярных выражений были удалены нерелевантные (т.е. небуквенные, исключая пробелы) символы, затем все заглавные буквы были заменены на свои строчные аналоги. После этого текст каждой новостной статьи был разбит на отдельные слова, называемые **токенами**. Затем каждый токен был нормализован, т.е. приведен к своей начальной (словарной) форме с использованием морфологического анализатора русского языка **pymorphy2** [4]. Из множества нормализованных токенов каждой статьи были удалены токены, соответствующие **стоп-**

словам, то есть таким словам языка, которые используются для связности предложений, однако при этом не несут в рассматриваемой задаче полезной нагрузки (местоимения, союзы, частицы и т.п.).

Последним этапом подготовки является **векторизация** новостных статей, в результате которого для каждой статьи генерируется вектор некоторого n-мерного векторного пространства $\mathbb{R}^n$. В рассматриваемой работе для этого используется предобученная модель векторизации **FastText**, которая доступна для свободного использования на интернет-ресурсе [7]. Достоинством данного метода векторизации является сохранение семантической, т.е. смысловой близости: близким по значению токенам ставятся в соответствие близкие по расстоянию вектора [1]. При помощи модели векторизации каждому токену ставится в соответствие свой собственный вектор пространства $\mathbb{R}^n$. Вектор для новостной статьи определяется как среднее арифметическое всех векторов отдельных токенов, входящих в данную новостную статью. В рассматриваемой задаче $n = 300$, т.е. токенам ставились в соответствие вектора 300-мерного векторного пространства.

При разработке системы автоматической рубрикации были испробованы четыре метода машинного обучения: наивный байесовский классификатор, логистическая регрессия, случайный лес решающих деревьев, а также искусственная нейронная сеть (ИНС). Первые три модели были построены и обучены с использованием библиотеки языка Python **Scikit-learn**. Искусственная нейронная сеть была реализована с использованием библиотеки языка Python **Keras**.

Для определения значений гиперпараметров, использование которых придает модели машинного обучения наибольшую обобщающую способность, был использован алгоритм решетчатого поиска, заключающийся в последовательном обучении некоторой модели машинного обучения на одних и тех же данных, но при различных значениях гиперпараметров [2]. Подготовленный ранее корпус был разделен на обучающую и тестовую выборки. Размер тестовой выборки составил 25% от общего числа статей корпуса.

Оптимизируемыми гиперпараметрами выступали: для модели классификации на основе логистической регрессии – параметр регуляризации, для случайного леса решающих деревьев – значения числа деревьев и максимального числа признаков.

Структура построенной ИНС представлена на рисунке 2.

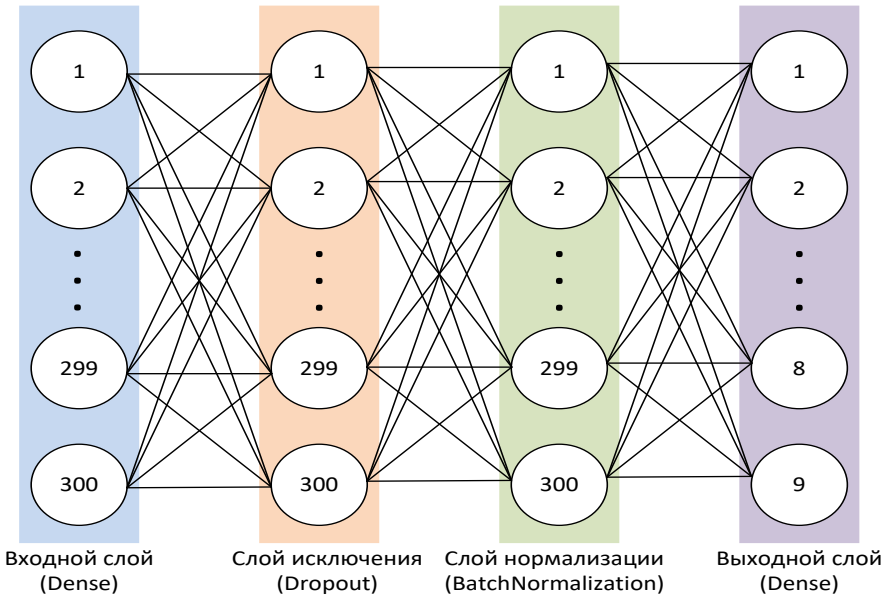

Рисунок 2 - Структура искусственной нейронной сети

Входной полносвязный слой содержит 300 нейронов, на каждый из которых подается соответствующая координата представляющего отдельную новостную статью 300-мерного вектора. Первый внутренний слой является слоем исключения и реализован с использованием встроенного класса **Dropout** библиотеки Keras. Принцип работы слоя исключения следующий: при работе с каждым из объектов обучающей выборки случайным образом отключаются отдельные нейроны этого слоя. Доля отключаемых нейронов определяется коэффициентом исключения, который указывается в конструкторе класса Dropout. Как правило, данное значение лежит в диапазоне от 0,2 до 0,5. Отключение отдельных нейронов позволяет снизить вероятность переобучения. Второй скрытый слой разработанной ИНС является слоем нормализации и реализован при помощи встроенного класса **BatchNormalization** библиотеки Keras и предназначен для статистической нормализации значений, получаемых на выходах предыдущих слоев. Выходной слой является полносвязным и содержит 9 нейронов, каждый из которых соответствует определенному классу, т.е. рубрике.

Для оценки обобщающей способности построенных моделей классификации были использованы метрики ***precision*** (точность) и ***recall*** (полнота**)**, а также ***F*-мера**, которые рассматриваются для каждого класса в отдельности [2]. Метрики точности $precision$ и полноты $recall$ определяются по формулам (1) и (2) соответственно.

$$precision = \frac{TP}{TP + FP} \tag{1}$$

$$recall = \frac{TP}{TP + FN} \tag{2}$$

где TP – количество объектов, которые были правильно классифицированы как относящие к данному классу;

TN – количество объектов, которые были правильно классифицированы как не относящиеся к данному классу;

FP – количество объектов, которые были ошибочно классифицированы как относящиеся к данному классу;

FN – количество элементов, которые были ошибочно классифицированы как не относящиеся к данному классу.

В качестве комбинированной метрики классификации используется *F*-мера, которая определяется в соответствии с формулой (3), где параметр β имеет смысл веса метрики точности $precision$. Частным случаем *F*-меры является F_1-мера, в которой $\beta=1$.

$$F_\beta = (1 + \beta^2)\frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{3}$$

В таблице 1 представлены значения рассмотренных выше метрик на тестовой выборке для каждой модели классификации в отдельности. По результатам проведенной оценки можно сделать вывод, что все модели классификации были успешно обучены и обладают обобщающей способностью. Однако нетрудно заметить, что наибольшие значения рассмотренных метрик имеет модель классификации на основе искусственной нейронной сети. Случайный лес решающих деревьев и логистическая регрессия показывает значения метрик классификации ниже, чем у искусственной нейронной сети, а наивный байесовский классификатор как самая простая из всех построенных моделей показывает наихудшие в данном исследовании результаты.

Таблица 1 - Сводная таблица значений метрик классификации на тестовой выборке для построенных моделей классификации

Модель классификации	**Среднее взвешенное по классам значение метрики точности**	**Среднее взвешенное по классам значение метрики полноты**	**Среднее взвешенное по классам значение F_1-меры**
Наивный байесовский классификатор	0,81459	0,79775	0,75367
Логистическая регрессия	0,90216	0,90236	0,90222
Случайный лес решающих деревьев	0,88318	0,88310	0,88221
Искусственная нейронная сеть	0,9253	0,9250	0,9251

Можно сделать вывод, что для дальнейшего использования в системе автоматической рубрикации наиболее пригодной является модель классификации на основе ИНС.

Для удобного взаимодействия пользователя с информационной системой при помощи фреймворка **Django** языка Python и системы управления базами данных **MySQL** был разработан веб-сайт, интерфейс которого представлен на рисунке 3. Боковая панель присутствует на всех веб-страницах и содержит пункты меню, которые используются для навигации по веб-сайту. Веб-страница, соответствующая отдельной рубрике, содержит набор блоков, каждый из которых посвящен отдельной новостной статье и содержит: дату публикации, заголовок статьи и начальную часть ее текста. При нажатии на блок происходит переход на страницу новостного ресурса с оригинальной новостной статьей.

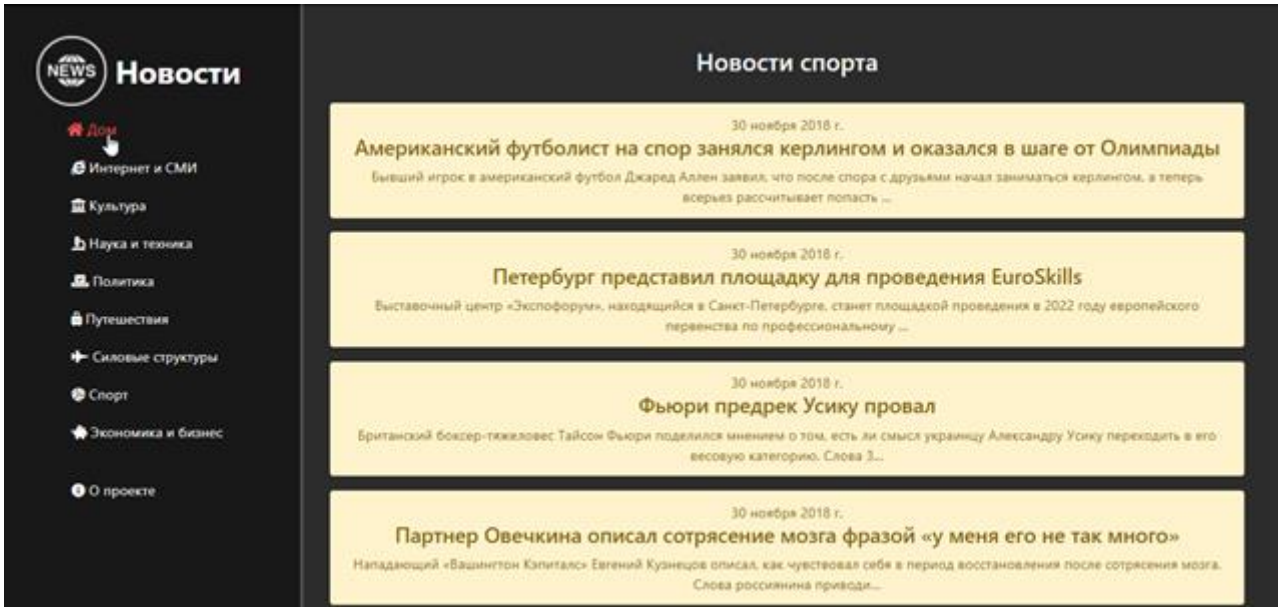

Рисунок 3 - Интерфейс веб-сайта информационной системы

Разработанная коллективом авторов информационная система может быть использована для автоматической рубрикации русскоязычных новостных статей в режиме реального времени с высокими показателями точности, а также обладает удобным пользовательским интерфейсом. Однако данная система может быть улучшена путем использования более точных и совершенных алгоритмов машинного обучения, добавлением возможности сбора

новостных статей, в том числе из нескольких источников, и расширением функционала веб-сайта.

Список литературы

1. Жеребцова, Ю.А., Чижик А.В. Сравнение моделей векторного представления текстов в задаче создания чат-бота. // Вестник НГУ. 2020. Т.18. URL: https://cyberleninka.ru/article/n/sravnenie-modeley-vektornogo-predstavleniya-tekstov-v-zadache-sozdaniya-chat-bota/viewer. (Дата обращения: 19.03.2021).
2. Мюллер, А. Введение в машинное обучение с помощью Python / А. Мюллер, С. Гвидо; пер. с англ. – М.: Альфа-книга, 2018. – 480 с.: ил. – ISBN: 978-1-449-36941-5.
3. Шаграев, А.Г. Модификация, разработка и реализация методов классификации новостных текстов: дисс. … канд. технических наук: 05.13.17. – НИУ «МЭИ», Москва, 2014. – 108 с.
4. Korobov, M.: Morphological Analyzer and Generator for Russian and Ukrainian Languages // Analysis of Images, Social Networks and Texts, pp. 320-332 (2015).
5. News dataset from Lenta.ru [Электронный ресурс] // Kaggle: Your Home for Data Science. URL: https://www.kaggle.com/yutkin/corpus-of-russian-news-articles-from-lenta. (Дата обращения 08.02.2021)
6. Reinsel, D. The Digitalization of the World / D. Reinsel, J. Gantz, J. Rydning – 2018. – 28 с.: [Электронный ресурс]. – URL: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf. (Дата обращения: 11.03.2021).
7. RusVectores: семантические модели для русского языка: [Электронный ресурс]. URL: https://rusvectores.org/ru/. (Дата обращения: 14.02.2021).

References

1. Zherebtsova, Yu. A., Chizhik A.V. Comparison of models of vector representation of texts in the task of creating a chatbot. // Bulletin of NSU. 2020. Vol. 18. URL: https://cyberleninka.ru/article/n/sravnenie-modeley-vektornogo-predstavleniya-tekstov-v-zadache-sozdaniya-chat-bota/viewer. (Accessed 19.03.2021).
2. Muller, A. Introduction to machine learning using Python / A. Muller, S. Guido; trans. from English. – M.: Alpha-book, 2018. – 480 p.: ill. – ISBN: 978-1-449-36941-5.
3. Shagraev, A. G. Modification, development and implementation of methods for classifying news texts: diss. ... Candidate of Technical Sciences: 05.13.17. - NRU "MEI", Moscow, 2014. - 108 p.
4. Korobov, M.: Morphological Analyzer and Generator for Russian and Ukrainian Languages // Analysis of Images, Social Networks and Texts, pp. 320-332 (2015).
5. News dataset from Lenta.ru [Electronic resource] // Kaggle: Your Home for Data Science. URL: https://www.kaggle.com/yutkin/corpus-of-russian-news-articles-from-lenta. (Accessed 08.02.2021).
6. Reinsel, D. The Digitalization of the World / D. Reinsel, J. Gantz, J. Rydning – 2018. – 28 с.: [Electronic resource]. – URL: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf. (Accessed 11.03.2021).

7. RusVectores: semantic models for the Russian language: [Electronic resource]. URL: https://rusvectores.org/ru/. (Accessed 14.02.2021).

Балашов О.В., Букачев Д.С. Методика оценки качества решений в системах организационного управления на основе нечёткой логики // Международный журнал информационных технологий и энергоэффективности. – 2021 –
Т. 6 № 3(21) с. 18–23

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 681.3.06

МЕТОДИКА ОЦЕНКИ КАЧЕСТВА РЕШЕНИЙ В СИСТЕМАХ ОРГАНИЗАЦИОННОГО УПРАВЛЕНИЯ НА ОСНОВЕ НЕЧЕТКОЙ ЛОГИКИ

[1]Балашов О.В., [2]Букачев Д.С.
[1]*Смоленский филиал АО «Радиозавод», Россия, (214027, г. Смоленск, улица Котовского,2), e-mail: smradio@mail.ru*
[2]*ФГБОУ ВО Смоленский государственный университет, Смоленск, Россия (214000, г. Смоленск, ул. Пржевальского, 4), e-mail: dsbuka@yandex.ru*

В данной статье рассмотрены основные принципы, лежащие в основе подхода к оценке качества управленческих решений, характерного для систем организационного управления. Представлено алгоритмическое описание методики оценки качества управленческих решений на основе аппарата нечеткой логики и принципа Беллмана-Заде. Даны рекомендации по модификации методики оценки качества в случае большого числа критериев.

Ключевые слова: управленческое решение, лингвистическое описание, оценка качества.

METHODOLOGY FOR ASSESSING THE QUALITY OF SOLUTIONS IN ORGANIZATIONAL MANAGEMENT SYSTEMS BASED ON FUZZY LOGIC

[1]Balashov O.V., [2]Bukachev D.S.
[1] *Smolensk branch of joint-stock company "Radio factory", Russia, (214027, Smolensk, street Kotovskogo, 2), e-mail: smradio@mail.ru*
[2] *Federal State Educational Institution of Higher Education Smolensk State University, Smolensk, Russia (214000, Smolensk, street Przewalski, 4), e-mail: dsbuka@yandex.ru*

This article discusses the basic principles underlying the approach to assessing the quality of management decisions, which is characteristic of organizational management systems. An algorithmic description of the methodology for assessing the quality of management decisions based on the apparatus of fuzzy logic and the Bellman-Zade principle is presented. Recommendations are given for modifying the quality assessment methodology in the case of a large number of criteria.

Keywords: management decision, linguistic description, quality assessment.

Важность проблем, связанных с принятием решений, привлекает к ним внимание широкого круга ученых и практических работников. Принятие решений – процесс систематизированный. Ответственность за принятие организационных решений велика, ведь от решения руководителя может зависеть судьба самой организации и/или отдельных её членов. С экономической и управленческой точек зрения принятие решения следует рассматривать как фактор повышения эффективности хозяйственной деятельности. Кризис качества управленческих решений на всех уровнях власти и хозяйствования тесным образом

связан со сложившимися в стране подходами к построению систем управления, уровнем подготовки управленцев и специалистов в высших учебных заведениях.

Качество управленческих решений определяется следующими основными факторами (см., например, [1])

- качеством принятой методики вычисления тех или иных величин для принятия решений;
- качеством принятой методики вычисления тех или иных величин для принятия решений;
- качеством исходных данных (информация состояния), на которых принимается решение;
- безошибочностью выполнения расчётов ЛПР.

В данной статье речь пойдёт о методике вычисления тех или иных величин для принятия решений.

Предлагаемая методика оценки качества управленческих решений в своей основе имеет следующие, как представляется, понятные на интуитивном уровне принципы, характерные для систем организационного управления.

1. *Уникальность операции*. Данный принцип, по сути, констатирует тот факт, что совпадающих операций не бывает. Это означает, в частности, неприменимость для выработки и оценки качества управленческих решений вероятностного подхода (не выполняется условие повторяемости опытов, т. е. вероятностной устойчивости).
2. *Неопределённость* условий задачи описывается и представляется с помощью средств нечёткой логики [2, 3].
3. *Множественность показателей качества (критериев)*. Этот принцип отражает необходимость комплексной, всесторонней оценки последствий любого решения, в противном случае задача становится практически тривиальной.
4. *Сочетание качественных (вербальных) и количественных характеристик*. Этот принцип допускает (и рекомендует) комбинировать различные виды моделей описания качества решения, поскольку далеко не всегда чисто количественный подход (из-за разномасштабности показателей, имеющейся неопределённости и т. п.) позволяет сравнивать различные альтернативы.
5. *Рациональность*. Принцип рациональности отражает введенное выше понятие критерия пригодности.

Кроме того, аппарат нечеткой логики с успехом применяется для решения задач, в которых данные об объектах управления являются ненадежными и сложноформализуемыми, не все цели выбора управленческих решений и условия, влияющие на их выбор, могут быть выражены в виде количественных соотношений, большая часть информации представлена в виде мнений экспертов. Сильной стороной такого подхода также является то, что описание условий и метода решения задачи выполняется на языке, близком к естественному. Кроме того, согласно теореме FAT (Fuzzy Approximation Theorem), доказанной Б. Коско (B. Kosko), любая математическая система может быть аппроксимирована системой, основанной на нечёткой логике [3].

С учетом приведенных принципов предлагаемая методика может быть описана последовательностью следующих шагов.

Шаг 1. ЛПР более высокого уровня иерархии, чем лицо, непосредственно принимающее решение, формирует перечень показателей Y_j (j = 1, 2, …, n), отражающих качество принимаемого решения. При формировании этого перечня необходимо принимать во внимание результативность, ресурсоемкость и оперативность операции, ради реализации которой принимается решение [4].

Шаг 2. По каждому количественному показателю определяются (оцениваются экспертным путем) минимальные $Y_{i\,min}$ и максимальные $Y_{i\,max}$ значения, после чего по формуле (1) осуществляется их нирмализация:

$$y_i = \frac{Y_i - Y_{i,\min}}{Y_{i,\max} - Y_{i,\min}}, \qquad (1)$$

$y_i \in [0,1]$, $i = 1, 2,\ldots, m$.

Шаг 3. Каждый нормированный показатель y_i рассматривается как лингвистическая (нечёткая) переменная [1–3], имеющая 5–9 уровней (термов, значений). Использование более 9 уровней нецелесообразно, поскольку из психологии известно, что в памяти человека удерживается одновременно не более 7±2 понятий [5]. Такие уровни могут иметь, например, следующую содержательную интерпретацию: 1 – незначительный; 3 – малый; 5 – средний; 7 – большой; 9 – значительный или очень большой; 2, 4, 6, 8 – промежуточные оценки.

Например, показатель "убытки" имеет качественные значения "незначительные убытки", "малые убытки", средние убытки" и т. д. Нетрудно видеть, что таким образом можно учитывать не только количественные, но и качественные (номинальные) показатели, характеризующие решение. Вообще говоря, данный шаг является необязательным, но зачастую он облегчает реализацию шага 6.

Шаг 4. Устанавливаются требования к рациональному решению, например: по первому критерию – уровень не меньше 5-го, по второму – не меньше 3-го и т.д.

Шаг 5. ЛПР формируется множество всех возможных вариантов принятия решения (альтернатив) A = {a_1, a_2,…, a_m}.

Шаг 6. Экспертным путем производится оценка каждой альтернативы по каждому из введенных показателей так, что для каждого j-го показателя y_j i-й альтернативы a_i определяется степень достижения поставленной цели (степень принадлежности) μ_{ij}, при этом $0 \le \mu_{ij} \le 1$.

Шаг 7. Составляется таблица (матрица) с элементами μ_{ij} (таблица 1).

Таблица 1 - Матрица для определения наилучшего решения.

a_i	y_j						min μ_{ij}
	y_1	y_2	. . .	y_j	. . .	y_n	
a_1	μ_{11}	μ_{12}	. . .	μ_{1j}	. . .	μ_{1n}	μ_1
a_2	μ_{21}	μ_{22}	. . .	μ_{2j}	. . .	μ_{2n}	μ_2
.	.	.		.		.	.

. .	. .	. .	. . .	. .	. . .	. .	. .
a_i	μ_{i1}	μ_{i2}		μ_{ij}		μ_{in}	μ_i
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
a_m	μ_{m1}	μ_{m2}	. . .	μ_{mj}	. . .	μ_{mn}	μ_m

Последний (правый) столбец данной матрицы содержит значения равные min μ_{ij}, т. е. значения, минимальные по строкам.

Шаг 8. Производится ранжирование вариантов решения (альтернатив) на основе пересечения нечётких множеств – критериев, которые отвечают известной в теории принятия решений схеме Беллмана–Заде [2].

Базируясь на принципе Беллмана–Заде, наилучшей системой будет считаться та, которая одновременно лучше по критериям y_1, y_2, …, y_n. Поэтому нечёткое множество, которое необходимо для рейтингового анализа, определяется в виде пересечения (интегральный критерий оценки качества решения). Учитывая, что в теории нечётких множеств операции пересечения соответствует минимум, получаем

$$\mu_i = \min \mu_{ij}. \quad (2)$$

В соответствии с логикой приведённых рассуждений наилучшей будет альтернатива a_g, для которой величина μ_g является наибольшей, т. е.

$$\mu_g = \max \mu_i. \quad (3)$$

Шаг 9. Задается некоторый минимальный уровень μ^* степени соответствия интегрального критерия оценки качества решения поставленной перед ЛПР цели ($0 \le \mu^* \le 1$), и проверяется неравенство

$$\mu^* \le \mu_g. \quad (4)$$

В случае его выполнения выявленная наилучшая альтернатива признается пригодной.

Качество решений в приведенной процедуре оценивается степенями соответствия μ_i альтернатив поставленной цели.

Замечания

1. Неравновесные критерии. Пусть заданы v_1, v_2, …, v_L – коэффициенты относительной важности (или ранги) критериев y_1, y_2, …, y_n такие, что $v_1+v_2+...+v_L=1$, при этом $v_1 - v_n$ считаются заданными. При наличии коэффициентов важности соотношение (2) принимает вид

$$\mu_i = \min\left(\mu_{ij}^{v_j}\right) \quad (5)$$

Окончательный выбор альтернативы осуществляется, как и выше, с помощью соотношения (3).

2. Применение принципа Беллмана – Заде, использующее в (2) операцию взятия минимума, во многих случаях, а именно, при большом числе критериев, может привести к тому, что итоговые степени принадлежности будут весьма близки к нулю, что затруднит их сравнение в соответствии с (3) и сделает общие выводы ненадежными. В этой связи представляется целесообразным операцию пересечения реализовывать не как операцию взятия минимума, а как операцию нахождения медианного значения из ряда имеющихся после их упорядочивания, в частности, в порядке возрастания.

Иначе говоря, имеет смысл отказаться от идеи выбирать в качестве интегральной оценки общую часть (пересечение) всех частных оценок, заменив её более гибким и продуктивным принципом выбора в качестве интегральной той частной оценки, которую дает некоторый специально сконструированный "наиболее представительный эксперт". Результаты проведённых исследований показывают, что такой "эксперт" должен в каждой точке области всех возможных альтернатив выбирать в качестве меры принадлежности этой точки интегральной оценке ту из мер её принадлежности частным оценкам, которая в общем случае удалена от крайних оценок и занимает некоторое "среднее" (медианное) положение.

Такой выбор означает, что объединение частных оценок в интегральную производится не по правилу пересечения нечётких множеств (где берется минимальная из оценок меры принадлежности) или по правилу их объединения (берется максимальная из оценок), или в соответствии с любой другой известной операцией над нечёткими множествами, а представляет собой некоторую новую операцию над такими множествами, а именно их упорядоченный набор с выбором затем одного из элементов такого упорядоченного набора.

Список литературы

1. Балашов О.В., Букачев Д.С. Подход к оценке качества управленческих решений на основе нечёткой логики // Международный журнал информационных технологий и энергоэффективности. – 2020. – Т.5, № 1 (15), с. 3-7.
2. Беллман Р., Заде Л. Принятие решений в расплывчатых условиях. В кн.: Вопросы анализа и процедуры принятия решений. - М.: Мир, 1976. - С. 172-215.
3. Круглов В. В., Дли М. И., Голунов Р. Ю. Нечеткая логика и искусственные нейронные сети. — М., «ФИЗМАТЛИТ», 2001.
4. Анфилатов В.С., Емельянов А.А., Кукушкин А.А. Системный анализ в управлении: Учебное пособие – М.: Финансы и статистика, 2002.
5. Miller G. A. The Magic Number Seven plus or Minus Two: Some Limits on Our Capacity for Processing Information // Psychological Review. 1956. № 63. P. 81–97.

References

1. Balashov O.V., Bukachev D.S. Podhod k ocenke kachestva upravlencheskih reshenij na osnove nechyotkoj logiki // Mezhdunarodnyj zhurnal informacionnyh tekhnologij i energoeffek-tivnosti. – 2020. – T.5, № 1 (15), s. 3-7.
2. Bellman P., Zade L. Prinyatie reshenij v rasplyvchatyh usloviyah. V kn.: Voprosy analiza i procedury prinyatiya reshenij. - M.: Mir, 1976. - S. 172-215.
3. Kruglov V. V., Dli M. I., Golunov R. YU. Nechetkaya logika i iskusstvennye nejronnye seti. — M., «FIZMATLIT», 2001.

4. Anfilatov V.S., Emel'yanov A.A., Kukushkin A.A. Sistemnyj analiz v upravlenii: Ucheb-noe posobie – M.: Finansy i statistika, 2002.
5. Miller G. A. The Magic Number Seven plus or Minus Two: Some Limits on Our Capacity for Processing Information // Psychological Review. 1956. № 63. P. 81–97.

Нагорных М. Э., Твардовский Г.В., Чернышёв С. А. Ускорение кода на Python // Международный журнал информационных технологий и энергоэффективности. – 2021 – Т. 6 № 3(21) с. 24-30

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.043:004.051

УСКОРЕНИЕ КОДА НА PYTHON

[1] Нагорных М. Э., [2] Твардовский Г.В., [3,4] Чернышёв С. А.

[1,2]Санкт-Петербургский государственный университет аэрокосмического приборостроения, Россия (190000, г. Санкт-Петербург, ул. Большая Морская, 67, лит. А),
e-mail: [1] nagornykh_max@mail.ru, [2] gtvardovsky@gmail.com
[3] Санкт-Петербургский государственный университет промышленных технологий и дизайна, Россия (191186, г. Санкт-Петербург, ул. Большая Морская, 18)
[4] Санкт-Петербургский государственный экономический университет, Россия (191023, г. Санкт-Петербург, ул. Садовая, 21), e-mail: chernyshev.s.a@bk.ru

В настоящее время Python является одним из наиболее популярных языков программирования. Несмотря на множество его преимуществ, у этого языка программирования есть существенный минус - его быстродействие. В данной статье анализируются и сравниваются различные способы решения обозначенной проблемы, такие как: использование дополнительных программных модулей, использование различных интерпретаторов и выполнение оптимизации кода. Приводится разбор каждого метода, а также продемонстрированы результаты, отражающие эффективность каждого из них.

Ключевые слова: Python, Cython, Psyco, интерпретатор, PyPy, Numba, профилирование.

ACCELERATING CODE ON PYTHON

[1]Nagornykh M.E., [2] Tvardovsky G.V., [3,4] Chernyshev S.A.

[1,2] Saint Petersburg State University of Aerospace Instrumentation, Russia (190000, Saint Petersburg, Bolshaya Morskaya st., 67, letter A),
e-mail: [1] nagornykh_max@mail.ru, [2] alexey_bykovoff@mail.ru
[3] Saint Petersburg State University of Industrial Technology and Design, Russia (191186, Saint-Petersburg, Bolshaya Morskaya st., 18)
[4] Saint Petersburg State University of Economics, Russia (191023, Saint Petersburg, Sadovaya st., 21), e-mail: chernyshev.s.a@bk.ru

Python is currently one of the most popular programming languages. Despite its many advantages, this programming language has a significant disadvantage - its performance. This article analyzes and compares various ways to solve the indicated problem, such as: using additional software modules, using different interpreters and performing code optimization. An analysis of each method is given, and the results are shown that reflect the effectiveness of each of them.

Keywords: Python, Cython, Psyco, interpreter, PyPy, Numba, profiling.

Введение

На сегодняшний день Python - самый популярный и быстроразвивающийся язык программирования (если опираться на исследование StackOverflow [1]). Разработчики его

выбирают за легкий порог вхождения, читаемость и простоту кода, а также за огромное количество постоянно обновляющихся библиотек. Код, выполняющий один и тот же функционал, написанный на языке Python, выглядит намного читаемее и компактнее, чем код на Си, Java или других языках программирования.

Например, рассмотрим простейшую программу, которая выводит строку на экран.

“Си” справится в 4 строки:

```
#include <stdio.h>
int main(int argc, char **argv) {
    printf(“Hello, world!”)
}
```

“Java” выполнит эту же программу в 5 строк, но код будет более сложный:

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println(“Hello, world!”);
    }
}
```

А вот листинг кода на Python:

```
print(‘Hello, world!’)
```

На основе приведенных примеров, можно сделать вывод, что код на языке Python более компактный и понятный, чем на других языках программирования и эта разница будет увеличиваться в разы с увеличением сложности, а также объема кода.

Однако, у языка программирования Python имеется один существенный недостаток по сравнению с другими языками высокого уровня - он имеет низкую скорость работы. Полностью избавиться от данного недостатка невозможно, но существуют различные способы ускорения работы кода [2]. В данной статье рассматриваются следующие из них: использование дополнительных программных модулей, интерпретаторов и нахождение оптимального алгоритма для решения той или иной задачи.

1. Использование дополнительных программных модулей

Существуют различные программные решения для ускорения кода на Python. Самые популярные из них: Psyco и Cython.

1.1. Cython

Cython является промежуточным слоем между Python и языком C/C++. Он позволяет писать Python-код с незначительными модификациями, который затем транслируется в C-код [3]. Для этого необходимо лишь добавить тип каждой переменной. Например, для того чтобы присвоить переменной “y” числовое значение “0,13”, необходимо явно указать ее тип: “cdef float y = 0.13”.

Это необходимо, потому что при использовании обычного Python-кода, типы переменных определяются динамически, а для языка С необходимо явное указание типа переменной.

При работе с функциями, Cython также дает возможность объявлять их по-другому для ускорения кода. Вместо классического “def” имеется возможность объявлять функции с помощью “cdef” и ”cpdef”. “cdef” - функция, которую можно вызывать только в пределах Cython-кода. cpdef-функция дает возможность вызова не только в пределах Python-кода, а также и из С-кода.

Пример листинга кода функций с использованием Cython и без:

```
def test(x):
    y = 1
    for i in range(1, x+1):
        y *= i
    return y
```

```
cpdef int test(int x):
    cdef int y = 1
    cdef int i
    for i in range(1, x+1):
        y *= i
    return y
```

Анализ эффективности Cython продемонстрирован в Таблице 1. Как из нее видно, Cython дает возможность значительно ускорить код на Python, не прикладывая особых усилий. Результат будет зависеть от количества циклов в коде программы, а также от количества обрабатываемых данных.

1.2. Psyco

Альтернативным вариантом решения является использование модуля Psyco. Работа этого модуля основывается на его внедрение в интерпретатор Python, где он выборочно заменяет части интерпретируемого байт-кода Python оптимизированным машинным кодом.

Psyco работает исключительно во время исполнения Python. Пока интерпретатор Python выполняет приложение, Psyco делает проверки, для попытки замены операции байт-кода Python отработанным машинным кодом [4].

Применять Psyco можно как для всего приложения, тогда модуль будет в автоматическом режиме производить вставки “по месту”. Еще имеется возможность указывать в параметрах модуля определенные методы и классы, с которыми нужно работать.

Однако, Psyco требует тестирования. Он полезен для обработки многократно выполняющихся циклов, и операций, в которых задействованы целые числа и числа с плавающей запятой. Для нециклических функций и для операций над другими типами объектов, Psyco обычно добавляет накладные расходы на анализ и внутреннюю компиляцию [4].

Для программ с большим числом методов и классов, включение Psyco для всего приложения является дополнительной нагрузкой. Кроме того, имеется существенный

недостаток - поддерживается только 32-битная версия Python версии не выше 2.6. Разработчики Psyco планировали выпуск второй версии модуля, но его выход так и не состоялся. Вместо этого, проект перерос в создание интерпретатора PyPy.

2. Использование различных интерпретаторов Python

2.1. PyPy

PyPy — это совместимый интерпретатор Python, с помощью которого можно получить заметное улучшение скорости. В PyPy встроен трассирующий JIT-компилятор, которой может превращать код на языке Python в машинный код прямо во время выполнения программы [5].

Оценка скорости работы PyPy приводится в Таблице 1, по ней видно, что ускорение кода происходит примерно в 50 раз. PyPy – это быстрая и эффективная альтернатива CPython. Запустив скрипт с его помощью, можно получить значительное улучшение скорости, не внося ни одного изменения в код.

Однако, следует отметить некоторые недостатки PyPy, такие как: обработка кода заново при мельчайших изменениях, поддержка не все библиотек Python, проблемы с jit-компилятором, а точнее с его уровнями абстракций, что в свою очередь влечет появление непредвиденных конфликтов и усложнение процесса поиска ошибок.

2.2. Numba

Numba — это JIT-компилятор с открытым исходным кодом. Суть его работы заключается в переводе функции Python в оптимизированный машинный код во время выполнения программы. Для этого используется стандартные библиотеки компилятора LLVM. Скомпилированные с помощью Numba алгоритмы могут приближаться к скорости выполнения программ на C или FORTRAN. Также Numba позволяет распараллеливать выполняемые задачи, что дает прирост в скорости работы [6].

Кроме того, Numba не требует замены интерпретатора Python или отдельного запуска. Для её использования необходимо применить один из декораторов Numba к функции.

Пример использования Numba продемонстрирован ниже:

```
from numba import jit
import random

@jit(nopython=True)
def numba_test(a):
    count = 0
    for i in range(a):
        x = random.random()
        y = random.random()
        if (x ** 2 + y ** 2) < 1.0:
            count += 1
    return count
```

Анализ эффективности использования данного программного модуля и сравнение его с другими происходит в Таблице 1. Из него видно, что, используя Numba достигается увеличение скорости выполнения программы примерно в 50 раз.

Подводя итоги анализа различных программных решений, стоит отметить, что каждый из модулей показывает скорость работы превосходящую скорость выполнения без их использования. Также следует учитывать, что их работа будет особенно сильно заметна при наличии большого количества циклов, математических вычислений и объема данных.

Для сравнения скорости работы с использованием дополнительных программных модулей и без, использовалась следующая функция:

```
total = 0
for i in range(1, 10000):
    for j in range(1, 10000):
        total += i + j
```

Полученные результаты сравнения представлены в Таблице 1.

Таблица 1 - Сравнение скорости выполнения кода с использованием различных программных модулей и без.

	Без использования дополнительных программных модулей	CPython	PyPy	Numba
Среднее время выполнения, сек	10	0.5	0.2	0.2

3. Оптимизация кода

Другой рассматриваемый метод ускорения кода на Python - его оптимизация.

3.1. Вынос кода программы в отдельные методы

Данный шаг помогает интерпретатору python проводить оптимизацию кода быстрее. Если изначальный коды выглядит так:

```
for Y in range(height):
    for X in range(width):
        #code here
```

то, изменив его на:

```
def my_func(height, width):
    for Y in xrange(height):
        for X in xrange(width):
            #code here
my_func(height, width)
```

время выполнения кода, согласно проверке через модуль time, сократится на 2.5 секунды, то есть 79%.

3.2. Профилирование

Стандартная библиотека Python'a, содержит модуль cProfile, который упрощает профилирование кода [7].

Модуль содержит множество команд и готовых решений, в статье рассматриваются лишь некоторые. Так, например, ключ интерпретатора -m позволяет запускать модули как отдельные программы, если сам модуль предоставляет такую возможность.

Пример:

```
python -m cProfile test.py
```

Результатом этой команды будет получение информации о количестве вызовах функции, время работы всех ее вызовов и каждого отдельного соответственно. С помощью этой информации можно определить места требующие оптимизации.

Также, добавив ключ “-s time”, можно отсортировать вывод профилировщика по времени выполнения.

3.3. Математика

По возможности необходимо преобразовывать сложные математические операции в более простые. Например, есть смысл заменить возведение в степень на умножение, если используется возведение в степень двойки. Такой простой шаг позволяет уменьшить временные затраты на выполнение кода примерно на 60%. Например, функция:

```
def power():
    a = 99999
    return a**2
```

выполняется в среднем за 0.002 мс, а за это же время, заменив возведение в квадрат на умножение, время выполнения функции получается равным в среднем 0.0012 мс, что может дать существенный прирост производительности, в случае большого количества подобных вычислений и усложнении операндов.

3.4. Особенности Python и решаемой задачи

При оптимизации кода следует учитывать особенности решаемой задачи - в некоторых случаях нет нужды выполнять вычисления до конца.

К общим методам оптимизации можно отнести отказ от использования глобальных переменных и глобальных объектов. Например, если в функции используется модуль math, импорт его функций можно производить в самой функции.

Пример листинга кода:

```
def my_func(height, width):
    from math import atan2, cos, sqrt
    pix = img.load()
```

Также, при написании кода, следует учитывать типизацию. Например, функция abs вернет число типа float. Поэтому, для ускорения кода, его надо сравнивать с float, а не int.

Пример оптимизации кода:

```
if abs(X) > 2:  --> if abs(X) > 2.0:
```

Заключение.

Подводя итоги, следует отметить, что в данной статье рассмотрены не все модули для оптимизации кода, а также алгоритмические решения. Однако, общий принцип работы модулей схож, а некоторые алгоритмические решения невозможно продемонстрировать без конкретных примеров. Применение модулей преимущественно носит характер оптимизации работы циклов и структур обрабатывающих большие объемы данных.

Анализируя примеры применения рассмотренных способов оптимизации кода, можно сделать вывод о том, что при правильном подходе можно добиться уменьшения времени работы программы более 80%.

Список литературы

1. Опрос разработчиков Stack Overflow. – 2019 / [Электронный ресурс]. – Режим доступа: URL: https://insights.stackoverflow.com/survey/2019.
2. Mark Lutz. Learning Python. — O’Reilly Media, 2009
3. Philipp Herron. Learning Cython Programming. — Packt, 2013
4. Psyco.Официальный сайт. – 2010 / [Электронный ресурс]. – Режим доступа: URL: http://psyco.sourceforge.net/.
5. PyPy.Официальный сайт. – 2021 / [Электронный ресурс]. – Режим доступа: URL: https://www.pypy.org/.
6. Numba. Официальный сайт. – 2021 / [Электронный ресурс]. – Режим доступа: URL: http://numba.pydata.org/.
7. Бизли Д. Python. Подробный справочник. — СПб.: Символ-Плюс, 2010

References

1. Stack Overflow Developer Poll. Available at: https://insights.stackoverflow.com/survey/2019.
2. Mark Lutz. Learning Python. — O’Reilly Media, 2009.
3. Philipp Herron. Learning Cython Programming. — Packt, 2013
4. Psyco. Official site. Available at: http://psyco.sourceforge.net/.
5. PyPy. Official site. Available at: https://www.pypy.org/.
6. Numba. Official site. Available at: http://numba.pydata.org/.
7. Beasley D. Python. Detailed reference. - Symbol-Plus, 2010

Балашов О.В., Букачев Д.С., Островская В.И. Концепция универсальной торговой платформы с программируемой стратегией // Международный журнал информационных технологий и энергоэффективности. – 2021
Т. 6 № 3(21) с. 31–37

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.42

КОНЦЕПЦИЯ УНИВЕРСАЛЬНОЙ ТОРГОВОЙ ПЛАТФОРМЫ С ПРОГРАММИРУЕМОЙ СТРАТЕГИЕЙ

[1]Балашов О.В., [2]Букачев Д.С., [3]Островская В.И.
[1]*Смоленский филиал АО «Радиозавод», Россия, (214027, г. Смоленск, улица Котовского,2),e-mail: smradio@mail.ru*
[2]*ФГБОУ ВО Смоленский государственный университет, Смоленск, Россия (214000, г. Смоленск, ул. Пржевальского, 4), e-mail: dsbuka@yandex.ru*
[3]*ФГБОУ ВО Смоленский государственный университет, Смоленск, Россия (214000, г. Смоленск, ул. Пржевальского, 4), e-mail: ostrovlada@mail.ru*

В статье рассматриваются задачи автоматизации алгоритмической торговли (алготрейдинга) цифровыми активами. Анализ рынка существующих программных решений в сфере биржевой торговли позволил определить основные причины, тормозящие использование алготрейдинга индивидуальными инвесторами. Представлена концепция универсальной торговой платформы с программируемой стратегией для биржевой торговли, позволяющая экономисту создавать и тестировать свои собственные торговые стратегии без участия программиста. Рассмотрены вопросы архитектуры и лицензирования торговой платформы.

Ключевые слова: цифровой актив, биржевая торговля, алготрейдинг, торговый скрипт.

THE CONCEPT OF A UNIVERSAL TRADING PLATFORM WITH A PROGRAMMABLE STRATEGY

[1]BalashovO.V., [2]Bukachev D.S., [3]Ostrovskaya V.I.
[1] *Smolensk branch of joint-stock company "Radio factory", Russia, (214027, Smolensk, street Kotovskogo, 2), e-mail: smradio@mail.ru*
[2] *Federal State Educational Institution of Higher Education Smolensk State University, Smolensk, Russia (214000, Smolensk, street Przewalski, 4), e-mail: dsbuka@yandex.ru*
[3] *Federal State Educational Institution of Higher Education Smolensk State University, Smolensk, Russia (214000, Smolensk, street Przewalski, 4), e-mail: ostrovlada@mail.ru*

This article discusses the tasks of automating algorithmic trading (algotrading) digital assets. Analysis of the market of existing software solutions in the field of exchange trading made it possible to determine the main reasons that hinder the use of algorithmic trading by individual investors. The concept of a universal trading platform with a programmable strategy for exchange trading is presented, which allows an economist to create and test his own trading strategies without the participation of a programmer. The issues of architecture and licensing of the trading platform are considered.

Keywords: digital asset, exchange trading, algotrading, trading script..

Введение.

Биржевая торговля вручную – это колоссальные затраты времени. Требуется постоянно анализировать графики курсов, использовать технические индикаторы. Поэтому сегодня всё

популярнее становится торговля ботами (её называют алгоритмическим трейдингом, коротко – алготрейдингом). Исполнителем торгового алгоритма в этом случае является специальная программа. Но, к сожалению, ни один из представленных на рынке IT продуктов не дает возможности экономисту создавать и тестировать свои торговые стратегии без участия программиста.

По официальным отчётам Московской биржи количество активных трейдеров в России стремительно возрастает (таблица 1, рисунок 1) [1].

Таблица 1 - Количество уникальных клиентов в Системе торгов Московской биржи

Группы клиентов	**2015 г.**	**2016 г.**	**2017 г.**	**2018 г.**	**2019 г.**	**2020 г.**
Физические лица	1 006 751	1 102 966	1 310 296	1 955 118	3 859 911	9 412 672
Юридические лица	20 753	18 622	17 766	16 631	17 695	19 074
Иностранные лица	8 729	9 215	10 211	11 453	14 011	17 041
Доверительное управление (ДУ)	3 836	10 694	22 564	29 262	41 535	82 193
Всего	**1 040 069**	**1 141 497**	**1 360 837**	**2 012 464**	**3 933 152**	**9 530 980**

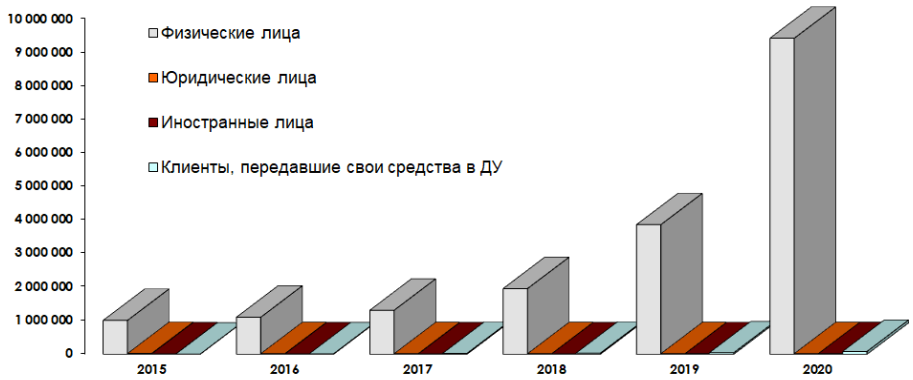

Рисунок 1 – Динамика количества активных трейдеров в России

Отчетность Московской биржи показывает, что в последнее время динамика роста только усиливается. Важно отметить, что на рынок заходят не только крупные игроки. В основном, приток осуществляется за счёт новых физических лиц, которые ставят своей целью не столько инвестировать, сколько активно торговать. По сравнению с инвестициями активная торговля на волатильных и падающих рынках выглядит более перспективной, поскольку зарабатывать можно и на падающем рынке.

Актуальная статистика торгов по интернет-биржам показывает, что Россия в настоящее время находится на 2-м месте в мире по количеству активных трейдеров (рисунок 2) [2]. Отсюда следует, что тематика цифровых активов сегодня очень популярна. Торговля ими – ещё популярнее.

Рисунок 2 – Место РФ в торговле цифровыми активами на интернет биржах

1. Постановка задачи

Алготрейдинг представляет собой удобную возможность автоматизации рутинных операций трейдера. В результате сокращается время, необходимое для анализа биржевой ситуации, математического расчета, размещения и контроля ордеров на закупку и продажу. Автоматизированные системы торговли помогают свести к минимуму влияние человеческого фактора — эмоций, паники, спешки, домыслов, которые зачастую делают убыточными даже профессиональные стратегии.

В настоящее время многие торговые платформы и терминалы позволяют торговать автоматизированно. Но, в основном, это готовые типовые стратегии и боты. Разрабатывать собственные торговые скрипты позволяют только такие гиганты, как MetaTrader [3] и QUIK [4]. Но в данном случае подразумевается торговля через брокера (требуется платить дополнительную комиссию, активы находятся у брокера). А чтобы разрабатывать торговые скрипты, нужно стать программистом и пройти серьезное обучение.

Чтобы продемонстрировать, что из себя представляют скриптовые языки для описания торговых стратегий, на рисунке 3 представлен фрагмент программного кода простейшего бота, торгующего по скользящим средним [5]. Объем кода – более 500 строк.

```
492  -- Возвращает ЗНАЧЕНИЕ БЫСТРОЙ скользящей по индексу свечи (по умолчанию: последняя)
493  function FastMA(Index)
494     -- Если индекс свечи не указан, то устанавливает индекс последней свечи
495     if Index == nil then Index = DS:Size(); end;
496     -- Сумма значений FAST_MA_SOURCE на FAST_MA_PERIOD свечах
497     local Sum = 0;
498     -- Перебирает последние FAST_MA_PERIOD свечей
499     for i=Index, Index - (FAST_MA_PERIOD - 1), -1 do
500        -- Считает сумму, исходя из выбранного источника для быстрой скользящей
501        if FAST_MA_SOURCE == 'O' then
502           Sum = Sum + DS:O(i);
503        elseif FAST_MA_SOURCE == 'C' then
504           Sum = Sum + DS:C(i);
505        elseif FAST_MA_SOURCE == 'H' then
506           Sum = Sum + DS:H(i);
507        elseif FAST_MA_SOURCE == 'L' then
508           Sum = Sum + DS:L(i);
509        else
510           message('Простой MA-робот:ОШИБКА! Не верно указан источник для быстрой скользящей!');
511           -- Останавливает скрипт
512           OnStop();
513        end;
514     end;
515     -- Возвращает значение
516     return Sum/FAST_MA_PERIOD;
517  end;
```

Рисунок 3 – Простейший торговый скрипт на языке QLUA для платформы QUIK

Даже простейший торговый алгоритм занимает несколько страниц программного кода. Это серьезно затрудняет использование алготрейдинга. Из-за высокой стоимости разработки, в основном, это могут позволить себе только крупные институциональные игроки.

В таблице 2 представлен сравнительный анализ популярных программных решений в сфере биржевой торговли (по материалам [3-4], [6-9]).

Таблица 2 - Сравнительный анализ торговых платформ

Платформа / терминал	Брокерская комиссия	Торговля ботами	Торговля скриптами	Конструктор торговых стратегий
CScalp	-	+	-	-
Margin	-	+	-	-
Coinigy	-	-	-	-
3commas	-	+	-	-
Traderbox	-	+	-	-
Capico	-	+	-	-
MetaTrader	+	+	+	-
QUIK	+	+	+	-
iTrader	+	+	-	примитивный

Сравнительный анализ популярных торговых платформ и терминалов показал, что на рынке IT не представлены программные решения, позволяющие экономисту разрабатывать собственные торговые скрипты без участия программиста. Поэтому задача сделать

алготрейдинг понятным, доступным и удобным является сегодня актуальной и привлекательной.

2. Предлагаемый подход

Для того, чтобы сделать алготрейдинг доступным для индивидуальных игроков без компетенций в сфере программирования, предлагается разработка универсальной торговой платформы с программируемой стратегией (назовём ее «ProStoTrader»), которая имеет следующие конкурентные преимущества:

1) простой и понятный скриптовый язык для описания торговых стратегий и интегрированный в систему интерпретатор торговых скриптов, позволяющий запускать торговых ботов, реализующих алгоритмы пользователя;

2) визуальный конструктор торговых скриптов, который позволит любому трейдеру генерировать торговые скрипты без привлечения программиста;

3) прямую работу через API с интернет-биржами без дополнительных брокерских комиссий и без возможности вывода средств с аккаунта пользователя.

Концепция торговой платформы «ProStoTrader» предполагает автоматизированные торги цифровыми активами на ведущих интернет-биржах, подключение и настройка торговых алгоритмов по каждому биржевому инструменту в отдельности и не имеет полных аналогов на рынке IT.

Предполагается, что потребителем предлагаемого программного продукта может стать как начинающий биржевой игрок, желающий попробовать себя в алготрейдинге, так и профессиональный трейдер, стремящийся автоматизировать свою деятельность.

На рисунке 4 показана предлагаемая архитектура торговой платформы.

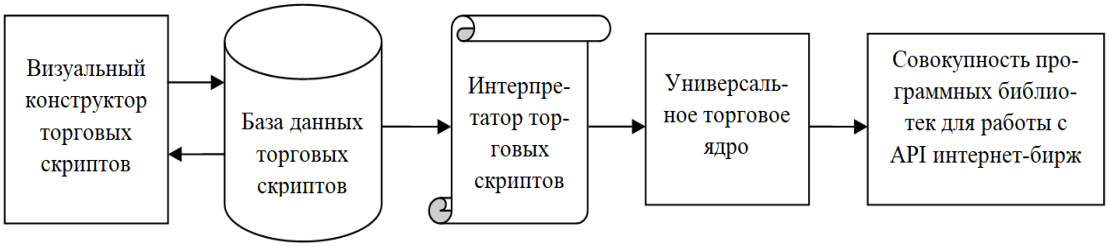

Рисунок 4 – Архитектура торговой платформы «ProStoTrader»

Для коммерциализации проекта предлагается использование классической бизнес-модели типа "Подписка" (рисунок 5) [10].

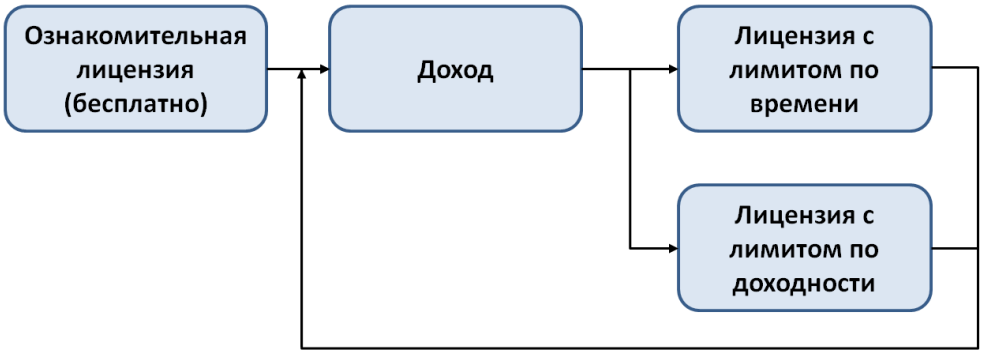

Рисунок 5 – Вариант схемы лицензирования торговой платформы «ProStoTrader»

Предлагается следующий механизм использования платформы:

Шаг 1. Клиент использует платформу бесплатно в рамках ознакомительной лицензии. Бесплатная лицензия позволяет использовать все инструменты платформы без ограничений, ограничение лишь одно: доходность. Лицензия перестанет действовать, когда пользователь в соответствии со статистикой своих сделок заработает определенную сумму.

Шаг 2. Заработанные средства пользователь вкладывает в один из двух типов лицензий (с ограничением по доходности или ограничением по времени). Для этого предлагается разработать соответствующие тарифные планы. Тарифы с ограничением по времени будут выгодны при высокой стоимости биржевого портфеля, для начинающих трейдеров со скромным портфелем более выгодной будет лицензия с лимитом по доходности (часть дохода можно снова вложить в приобретение новой лицензии).

Помимо технической значимости предлагаемого решения, следует обратить внимание на его научный потенциал. Предлагаемая платформа откроет прямой путь к алготрейдингу всем желающим, станет платформой разработки и тестирования новых торговых алгоритмов на различных рынках.

Заключение.

Разработка представленной торговой платформы поможет реализовать торговый потенциал тем, кто не обладает навыками программирования, станет вспомогательным инструментом проведения научных исследований в сфере электронных торгов, поможет привлечь дополнительные средства на рынок цифровых активов.

Список литературы

1. Статистическая информация по клиентам участников торгов фондового рынка Московской Биржи. URL: https://www.moex.com/s719.
2. Сравнительный анализ активности трейдеров маркетингового агентства BDCenter. URL: https://bdcenter.digital/ru/insights/cryptocurrency
3. Официальный сайт торгового терминала Metatrader 4. URL: https://www.metatrader4.com.
4. Официальный сайт торгового терминала QUIK. URL: https://arqatech.com/ru/products/quik.

5. Исходные программные коды простого торгового МА-бота. URL: https://github.com/koolvn/LUA_for_trading
6. Официальный сайт торгового терминала для скальпинга CScalp. URL: https://cscalp.net.
7. Официальный сайт торгового терминала Margin. URL: https://margin.de.
8. Официальный сайт торговой платформы Coinigy. URL: https://www.coinigy.com.
9. Официальный сайт торговой платформы для алготрейдинга 3commas. URL: https://3commas.io.
10. Clapp, S.L. The Beginnings of Subscription Publication in the Seventeenth Century // Modern Philology, 29 (1931), 199 - 224.

References

1. Statisticheskaya informaciya po klientam uchastnikov torgov fondovogo rynka Moskov-skoj Birzhi. URL: https://www.moex.com/s719.
2. Sravnitel'nyj analiz aktivnosti trejderov marketingovogo agentstva BDCenter. URL: https://bdcenter.digital/ru/insights/cryptocurrency.
3. Oficial'nyj sajt torgovogo terminala Metatrader 4. URL: https://www.metatrader4.com.
4. Oficial'nyj sajt torgovogo terminala QUIK. URL: https://arqatech.com/ru/products/quik.
5. Iskhodnye programmnye kody prostogo torgovogo MA-bota. URL: https://github.com/koolvn/LUA_for_trading
6. Oficial'nyj sajt torgovogo terminala dlya skal'pinga CScalp. URL: https://cscalp.net.
7. Oficial'nyj sajt torgovogo terminala Margin. URL: https://margin.de.
8. Oficial'nyj sajt torgovoj platformy Coinigy. URL: https://www.coinigy.com.
9. 9. Oficial'nyj sajt torgovoj platformy dlya algotrejdinga 3commas. URL: https://3commas.io.
10. Clapp, S.L. The Beginnings of Subscription Publication in the Seventeenth Century // Modern Philology, 29 (1931), 199 - 224.