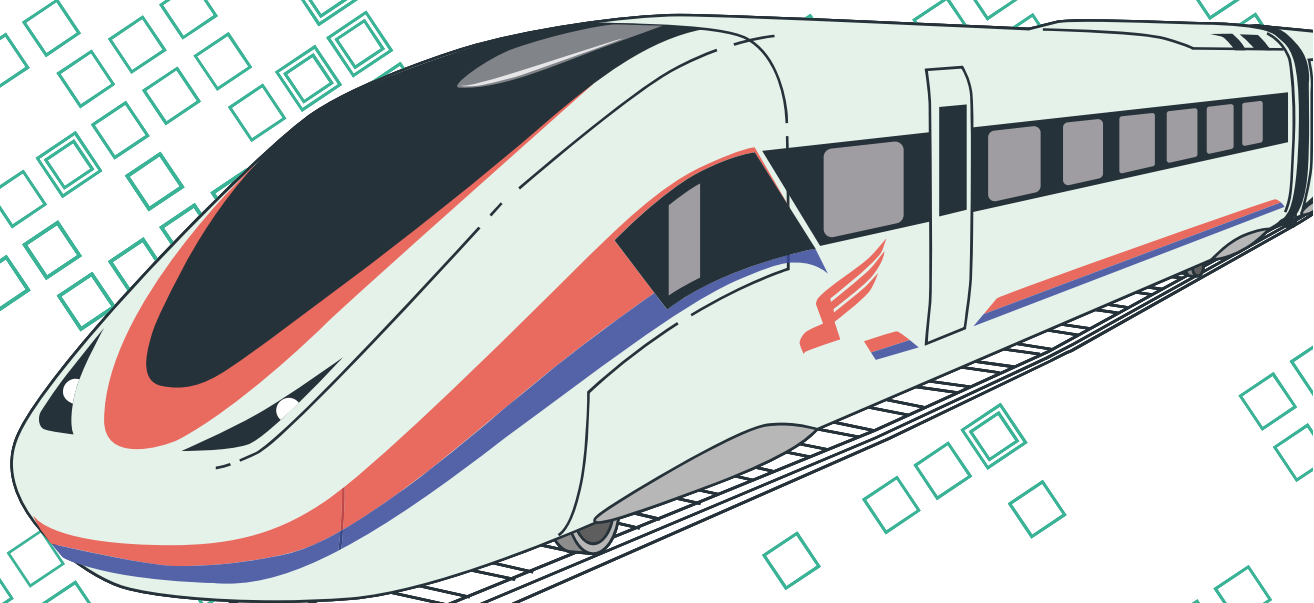


Интеллектуальные технологии на транспорте

Intellectual Technologies
on Transport



Выпуск 4
2023

**Интеллектуальные технологии на транспорте
(сетевой электронный научный журнал)
Выпуск 4 (36), 2023**

ISSN 2413-2527

Сетевой электронный научный журнал, свободно распространяемый через Интернет.
Публикуются статьи на русском и английском языках с результатами исследований
и практических достижений в области интеллектуальных технологий
и сопутствующих им научных исследований.

Журнал основан в 2015 году.

Учредитель и издатель

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Петербургский государственный университет путей сообщения Императора Александра I» (ФГБОУ ВО ПГУПС)

Главный редактор

Хомоненко А. Д., д.т.н., проф., С.-Петербург, РФ

Сопредседатели редакционного совета

Валинский О. С., к.т.н., ректор ПГУПС, С.-Петербург, РФ
Чаркин Е. И., зам. ген. директора по ИТ ОАО «РЖД», Москва, РФ

Редакционный совет

Ададулов С. Е., д.т.н., проф., ВНИИЖТ, Москва, РФ
Дудин А. Н., д.ф.-м.н., проф., БГУ, Минск, Беларусь
Корниенко А. А., д.т.н., проф., ПГУПС, С.-Петербург, РФ
Макаренко С. И., д.т.н., доц., ПАО «Интелтех»,
С.-Петербург, РФ

Меркурьев Ю. А., Dr. Habil., проф., член Латвийской АН,
РТУ, Рига, Латвия
Титова Т. С., д.т.н., проф., первый проректор ПГУПС,
С.-Петербург, РФ
Юсупов Р. М., д.т.н., проф., чл.-корр. РАН, СПб ФИЦ РАН,
С.-Петербург, РФ

Редакционная коллегия

Божко Л. М., д.э.н., доц., ПГУПС, С.-Петербург, РФ –
заместитель главного редактора
Баталов Д. И., к.т.н., ПГУПС, С.-Петербург, РФ –
научный редактор
Александрова Е. Б., д.т.н., проф., ПГУПС, С.-Петербург, РФ
Басыров А. Г., д.т.н., проф., ВКА, С.-Петербург, РФ
Безродный Б. Ф., д.т.н., проф., НИИАС, Москва, РФ
Благовещенская Е. А., д.ф.-м.н., проф., ПГУПС,
С.-Петербург, РФ
Бубнов В. П., д.т.н., проф., С.-Петербург, РФ
Булавский П. Е., д.т.н., доц., ПГУПС, С.-Петербург, РФ
Василенко М. Н., д.т.н., проф., ПГУПС, С.-Петербург, РФ
Глухов А. П., д.т.н., ПГУПС, С.-Петербург, РФ
Гуда А. Н., д.т.н., проф., РГУПС, Ростов-на-Дону, РФ
Ермаков С. Г., д.т.н., проф., ПГУПС, С.-Петербург, РФ
Заборовский В. С., д.т.н., проф., СПбПУ, С.-Петербург, РФ
Канаев А. К., д.т.н., проф., ПГУПС, С.-Петербург, РФ
Котенко А. Г., д.т.н., проф., ВНИИЖТ, Москва, РФ

Куренков П. В., д.э.н., к.т.н., проф., РУТ (МИИТ), Москва, РФ
Лецкий Э. К., д.т.н., проф., РУТ (МИИТ), Москва, РФ
Наседкин О. А., к.т.н., доц., ПГУПС, С.-Петербург, РФ
Никитин А. Б., д.т.н., проф., ПГУПС, С.-Петербург, РФ
Новиков Е. А., д.т.н., доц., ВКА, С.-Петербург, РФ
Охтилев М. Ю., д.т.н., проф., НИО ЦИТ «Петрокомета»,
С.-Петербург, РФ
Привалов А. А., д.воен.н., проф., С.-Петербург, РФ
Соколов Б. В., д.т.н., проф., СПб ФИЦ РАН,
С.-Петербург, РФ
Таранцев А. А., д.т.н., проф., ИПТ РАН, С.-Петербург, РФ
Утепбергенов И. Т., д.т.н., проф., АУЭС, Алматы, Казахстан
Фазылов Ш. Х., д.т.н., проф., НИИ развития цифровых
технологий и ИИ, Ташкент, Узбекистан
Хабаров В. И., д.т.н., проф., СГУПС, Новосибирск, РФ
Ходаковский В. А., д.т.н., проф., ПГУПС, С.-Петербург, РФ
Чехонин К. А., д.ф.-м.н., доц., ХВИЦ ДВО РАН,
Хабаровск, РФ

Адрес редакции:

190031, Санкт-Петербург, Московский пр., 9
e-mail: itt-pgups@yandex.ru
Телефон: +7 (812) 457-86-06

Сетевое издание «Интеллектуальные технологии на транспорте (сетевой электронный научный журнал),
Intellectual Technologies on Transport» зарегистрировано Федеральной службой по надзору в сфере связи,
информационных технологий и массовых коммуникаций.
Свидетельство Эл № ФС77–61707 от 07 мая 2015 г.

Журнал зарегистрирован в Российском индексе научного цитирования (РИНЦ).

Периодичность выхода – 4 номера в год. Выпуски журнала доступны на сайте <http://itt-pgups.ru>.

Информация предназначена для детей старше 12 лет.

Intellectual Technologies on Transport

Issue 4 (36), 2023

ISSN 2413-2527

Network electronic scientific journal, open access. It publishes articles in Russian and English with the results of research and practical achievements in the field of intelligent technologies and associated research.

Founded in 2015.

Founder and Publisher

Federal State Budgetary Educational Institution of Higher Education
«Emperor Alexander I St. Petersburg State Transport University»

Editor-in-Chief

Khomonenko A. D., Prof., St. Petersburg, Russia

Co-chairs of the Editorial Council

Valinsky O. S., rector of PSTU, St. Petersburg, Russia
Charkin E. I., CIO of JSC «Russian Railways», Moscow, Russia

Editorial Council Members

Adadurov S. E., Prof., VNIIZHT, Moscow, Russia
Dudin A. N., Prof., BSU, Minsk, Belarus
Kornienko A. A., Prof., PSTU, St. Petersburg, Russia
Makarenko S. I., As. Prof., Inteltech, St. Petersburg, Russia

Merkuryev Yu. A., Prof., Academician of the Latvian Academy
of Sciences, RTU, Riga, Latvia
Titova T. S., Prof., First Vice-Rector PSTU, St. Petersburg, Russia
Yusupov R. M., Prof., Corr. Member of RAS, SPC RAS,
St. Petersburg, Russia

Editorial Board Members

Bozhko L. M., As. Prof., PSTU, St. Petersburg, Russia –
Deputy Editor-in-Chief
Batalov D. I., PSTU, St. Petersburg, Russia –
Science Editor
Aleksandrova E. B., Prof., SPbPU, St. Petersburg, Russia
Basyrov A. G., Prof., MSA, St. Petersburg, Russia
Bezrodny B. F., Prof., NIIAS, Moscow, Russia
Blagoveshchenskaya E. A., Prof., PSTU,
St. Petersburg, Russia
Bubnov V. P., Prof., St. Petersburg, Russia
Bulavsky P. E., As. Prof., PSTU, St. Petersburg, Russia
Vasilenko M. N., Prof., PSTU, St. Petersburg, Russia
Glukhov A. P., PSTU, St. Petersburg, Russia
Guda A. N., Prof., RSTU, Rostov-on-Don, Russia
Ermakov S. G., Prof., PSTU, St. Petersburg, Russia
Zaborovsky V. S., Prof., SPbPU, St. Petersburg, Russia

Kanaev A. K., Prof., PSTU, St. Petersburg, Russia
Kotenko A. G., Prof., VNIIZHT, Moscow, Russia
Kurenkov P. V., Prof., RUT (MIIT), Moscow, Russia
Letsky E. K., Prof., RUT (MIIT), Moscow, Russia
Nasedkin O. A., As. Prof., PSTU, St. Petersburg, Russia
Nikitin A. B., Prof., PSTU, St. Petersburg, Russia
Novikov E. A., As. Prof., MSA, St. Petersburg, Russia
Okhtilev M. Yu., Prof., JSC «Petrokometa», St. Petersburg, Russia
Privalov A. A., Prof., PSTU, St. Petersburg, Russia
Sokolov B. V., Prof., SPC RAS, St. Petersburg, Russia
Tarantsev A. A., Prof., IPT RAS, St. Petersburg, Russia
Utepbergenov I. T., Prof., AUPET, Almaty, Kazakhstan
Fazilov Sh. X., Prof., AIRI, Tashkent, Uzbekistan
Khabarov V. I., Prof., STU, Novosibirsk, Russia
Khodakovskiy V. A., Prof., PSTU, St. Petersburg, Russia
Chekhonin K. A., As. Prof., Khabarovsk FRC RAS, Khabarovsk, Russia

Editorial address:

190031, St. Petersburg, Moskovsky ave., 9
e-mail: itt-pgups@yandex.ru
Phone: +7 812 457 86 06

The online journal «Intellectual Technologies on Transport» is registered by the Federal Service for Supervision of Communications, Information Technologies, and Mass Media.
EI No. FS77-61707 Testimony from May 7, 2015.

The journal is registered in the Russian Science Citation Index (RSCI).

Frequency of release - 4 issues per year. Issues of the magazine are available at <http://itt-pgups.ru>

The content is for children over the age of 12.

© Federal State Budgetary Educational Institution of Higher Education
«Emperor Alexander I St. Petersburg State Transport University», 2023

Содержание

Интеллектуальные системы и технологии на транспорте

<i>Блюм В. С., Космачев В. М., Поляков О. М.</i> Семантические принципы разработки текстовых систем искусственного интеллекта	5
<i>Бобриков Д. А., Калиневич Н.</i> Внедрение системы работы с машиночитаемыми доверенностями через средства API в системе 1С:Документооборот	13
<i>Михайлова С. А., Аникеев К. Г.</i> Применение технологий обработки естественного языка для голосового управления на основе открытого словаря	19

Системный анализ и моделирование сложных систем

<i>Зыкова С. С.</i> Модель и алгоритм планирования параллельной обработки информации в отказоустойчивой бортовой вычислительной системе на основе временной избыточности вычислительного процесса	28
<i>Нельин А. А., Забродин А. В.</i> Разработка и практическое применение модели интеграции оператора импликации и перегруженного контейнера «Вектор» на языке программирования C++	34

Информационная безопасность и кибербезопасность

<i>Слесарев Р. М., Добровольская М. А., Забродин А. В.</i> Разработка алгоритма сжатия данных на основе метода Хаффмана: практический аспект	42
---	----

Телекоммуникации

<i>Сыркин И. С., Пашков Д. А., Дубинкин С. Д.</i> Тенденции развития создания интеллектуальной собственности в области разработки цифровых систем диспетчеризации угольного карьера	47
---	----

Электронное обучение и дистанционные образовательные технологии

<i>Герасименко П. В., Вертешев С. М.</i> Об исследовании результатов подготовки бакалавров направления ИВТ ПсковГУ по очной и смешанной формам обучения	53
---	----

.....

Памяти коллеги	58
----------------------	----

Contents

Intellectual systems and transport technologies

Blyum V. S., Kosmachev V. M., Polyakov O. M.
Semantic Principles of Text Development Artificial Intelligence Systems 5

Bobrikov D. A., Kalinevich N.
Implementation of the Work System with Machine-Readable Powers of Attorney
Through API Tools in the 1C:Document Management 13

Mikhaylova S. A., Anikeev K. G.
Application of Natural Language Processing Technologies for Voice Control
Based on an Open Dictionary 19

System analysis and modelling of complex systems

Zykova S. S.
A Model and Algorithm for Planning Parallel Information Processing in a Fault-Tolerant
On-Board Computing System Based on the Time Redundancy of the Computing Process 28

Nelin A. A., Zabrodin A. V.
Development and Practical Application of the Integration Model of the Implication Operator
and the Overloaded «Vector» Container in the C++ Programming Language 34

Information security and cybersecurity

Slesarev R. M., Dobrovolskaya M. A., Zabrodin A. V.
Development of a Data Compression Algorithm Based on the Huffman Method 42

Telecommunication

Syrkin I. S., Pashkov D. A., Dubinkin S. D.
Trends in the Development of Intellectual Property Creation in the Field of Development
of Digital Coal Pit Dispatch Systems 47

E-learning and distant educational technologies

Fedorov D. Yu., Zabrodin A. V.
About the Study of the Results of Bachelor's Degree Training in the IVT Direction
in Pskov State University in Full-Time and Mixed Forms of Education 53

.....

In Memory of Colleague 58

Семантические принципы разработки текстовых систем искусственного интеллекта

к.т.н. В. С. Блюм, к.т.н. В. М. Космачёв, к.т.н. О. М. Поляков

Санкт-Петербургский государственный университет аэрокосмического приборостроения

Санкт-Петербург, Россия

vladblum7@gmail.com, kvm@aanet.ru, polyakovoleg@yandex.ru

Аннотация. Целью статьи является определение теоретических и технологических основ вычисления семантики текстов. Доказано, что любое предложение, имеющее безусловное правое отрицание, не имеет смысла. Рассмотрены принципы построения основных современных генераторов «осмысленных» текстов, последовательных систем искусственного интеллекта и способы их применения. Рассмотрены структура и алгоритмы рекуррентных нейронных сетей и универсального нейросетевого вычислительного механизма Transformer. Новизна подхода состоит в том, что в основу вычисления семантики генерируемого текста положена большая языковая модель.

Ключевые слова: семантика, языковая модель, искусственный интеллект.

ВВЕДЕНИЕ

Синтез лингвистической модели является одной из центральных проблем обработки естественного языка (Natural Language Processing, NLP) [1] — общего направления искусственного интеллекта (ИИ) и математической лингвистики.

Одной из основных теорий лингвистики, используемых в синтезе лингвистической модели искусственного интеллекта, является теория формальных грамматик, определяющая правила построения предложений и языковых конструкций, а также правила определения значения слов и действий внутри предложения [2].

Для синтеза лингвистической модели искусственного интеллекта используются такие методы компьютерных наук, как алгоритмы машинного обучения и нейронные сети [3]. Эти методы позволяют обучать модель распознаванию и анализу языковых конструкций и текстов.

В целом синтез лингвистической модели искусственного интеллекта основывается на изучении естественных языков и применении теории компьютерных наук для создания интеллектуальных систем, способных анализировать и обрабатывать текстовую информацию на естественном языке.

СЕМАНТИЧЕСКИЕ АСПЕКТЫ И ПРОБЛЕМЫ ПОСЛЕДНИХ ВЕРСИЙ ИИ

Для того, чтобы понять возможности и недостатки систем ИИ типа большой языковой модели GPT 4 (Generative Pre-trained Transformer 4), необходимо разобраться с общим вопросом формирования семантики.

Заметим, что в качестве объекта обучения в таких системах выступает рекуррентная нейросеть. Это означает, что прогноз в таких моделях производится не только на основе поступающих извне параметров, но и на основе предыдущего состояния сети. Это происходит независимо

от числа используемых параметров, выделенных системой вниманием.

Итак, пусть в нашем распоряжении есть некоторый естественный язык и \mathcal{L} — множество предложений в этом языке. Хотя в GPT 4 прогнозу подвергаются слова, предположение о характере множества \mathcal{L} не является существенным, поскольку при этом сохраняется последовательно-рекуррентная структура интерпретации текста.

Под предложением мы понимаем последовательность слов языка, которая может быть интерпретирована (отражена в модели). Это означает существование некоторого оператора интерпретации (осмысления) Ψ , который перерабатывает предложения в семантику или смысл. Мы фактически называем предложением такие конечные последовательности слов языка, которые входят в область определения оператора Ψ .

Оператор интерпретации Ψ интерпретирует предложение $s \in \mathcal{L}$ с учетом уже накопленной к моменту интерпретации s семантики C (предшествующие события или предложения могут изменять или уточнять интерпретацию s) и формирует новый смысл C' , включающий и смысл предложения s . Итак, «текущая» семантика после осмысления предложения s определяется как $C' = \Psi(C, s)$.

В частности, при отдельной интерпретации предложения без предварительных смыслов получаем $C' = \Psi(\emptyset, s)$. Под отдельной интерпретацией предложения s здесь понимается такая интерпретация, которой не предшествует некоторая семантика в отношении предложения s . Например, когда мы приступаем к чтению СМС-сообщения с неизвестного номера телефона, мы имеем нулевую начальную семантику.

К этому моменту мы не знаем, как работает эта интерпретация и из чего состоит семантика, поэтому C' пока обозначает не множество, а некую семантическую структуру, которая получается после интерпретации s . В частности, \emptyset обозначает здесь не пустое множество, а пустую семантическую структуру, которая соответствует отсутствию смысла. Разумеется, у каждого человека или ИИ в конкретный момент времени свой оператор интерпретации. Он зависит от модели мира, от настроения, желаний и т. п.

Например, интерпретация одного и того же текста человеком в состоянии сильного испуга может существенно отличаться от интерпретации в благодушном расположении духа. Но все перечисленные факторы на конкретный момент интерпретации фиксированы, изменяется только накапливаемый смысл, так что на момент интерпретации предложения у каждого интеллекта действует какой-то конкретный оператор Ψ .

Мы перевели проблему множественности факторов, влияющих на формирование смысла, на оператор Ψ . Например, формируемый смысл зависит от действующей на момент интерпретации модели мира M , что отмечено в [1]: «Мы не можем судить о том, что человек хочет сказать, пока мы не знаем его взглядов, в то же время мы не можем судить, каковы взгляды человека, пока мы не знаем того, что он говорит». Тот факт, что мы перенесли эту зависимость на Ψ , означает: мы полагаем, что за время интерпретации текста модель мира не меняется. Это означает, что в мозгу человека или в модели ИИ имеется долговременная память (результат обучения), в которой хранится модель мира (для GPT 4 — модель языка), и оперативная память, в которой накапливаются текущие смыслы.

Кроме того, существуют временные периоды, когда накопленные смыслы пересматриваются, чтобы скорректировать модель мира. Например, у человека это может происходить во время сна, а у ИИ — в период обучения. После обучения ИИ имеет обновленную модель мира M' , что соответствует обновленному оператору Ψ' .

Иногда возникают моменты, когда человеку как бы «открывают глаза» на происходящее. Другими словами, он получает такие важные смыслы, что возникает необходимость срочно скорректировать модель мира, после чего интерпретация им того же текста станет совершенно другой. Но это бывает редко, и мы не будем рассматривать случай изменения оператора Ψ в процессе интерпретации.

Описанный выше механизм вполне соответствует «герменевтическому кругу», суть которого сформулировал один из родоначальников герменевтики Фридрих Шлейермахер (Friedrich Schleiermacher) [1]: «...как целое понимается из отдельного, но и отдельное может быть понято только из целого, имеет такую важность для данного искусства и столь неоспоримо, что уже первые же операции невозможно проделать без применения его, да и огромное число герменевтических правил в большей или в меньшей степени основывается на нем...». Другими словами, получаемые «частные смыслы» через оператор интерпретации зависят от модели мира, которая изменяется этими смыслами и изменяет оператор интерпретации, что приводит к изменению получаемых смыслов от тех же текстов и так далее по кругу.

На множестве предложений языка \mathcal{L} определена операция конкатенации (приписывания — \bullet), которая к одному предложению приписывает другое так, что в результате получается некоторый текст. Под текстом $s = s_1 \bullet \dots \bullet s_n$ будем понимать конечную последовательность предложений из \mathcal{L} .

Будем считать, что интерпретация текста $s = s_1 \bullet \dots \bullet s_n$ происходит следующим образом. Сначала интерпретируется первое предложение $C_1 = \Psi(\emptyset, s_1)$. Предполагается, что предварительного смысла в отношении текста перед началом интерпретации нет. На основе интерпретации первого предложения интерпретируется второе $C_2 = \Psi(C_1, s_2)$, на основе интерпретации первых двух интерпретируется третье предложение и так далее до C_n .

Здесь необходимо сделать одно замечание. Несмотря на то, что мы определили текст как последовательность предложений, все же правильнее было бы понимать под текстом абзац. Однако неопределенность, связанная с этим семантическим понятием, не позволяет дать его строгое определение, не сводящееся к тавтологии типа «абзац — отрезок

письменной речи между двумя красными строками», что фактически означает: абзац — это то, что выделено как абзац. Обычно интерпретация происходит по абзацам, которые в устной речи выделяются более длительными паузами. Разбиение устной речи на абзацы хорошо видно, когда человек говорит под перевод, делая паузу и как бы приглашая переводчика приступить к переводу абзаца.

Если предложение является единицей интерпретации, то абзац — единицей завершенной мысли. Завершение абзаца обычно означает, что говорящий дал на его взгляд достаточно сведений, чтобы слушающий мог завершить интерпретацию, задать недостающие для интерпретации вопросы и осуществить логический вывод.

Для теоретической семантики представляется крайне важным понимание оснований (правил), по которым говорящий определяет конец абзаца. В частности, для экспертных систем это сигнал к началу вывода, а точнее — полноценного вывода. Здесь уместно также напомнить о различиях между синхронным и последовательным переводами. Разница между ними заключается именно в отсутствии для синхронного переводчика абзацев как завершенных мыслей.

Как бы ни выглядела семантическая структура C , для естественных языков выполняются две аксиомы.

Первая аксиома утверждает, что в языке существует пустое предложение e , приписывание которого к любому тексту справа и слева не меняет интерпретации текста. Это означает, что e само имеет пустую семантику ($\Psi(C, e) = C$ для любого C) и не изменяет семантику любого текста: $\Psi(C, e \bullet s) = \Psi(C, s \bullet e) = \Psi(C, s)$.

Например, если после текста (перед текстом) идет чистая бумага, потому что текст закончился (не начался), то семантика текста не изменяется. Это семантическое правило в языке отображается как $s \bullet e = e \bullet s = s$. В языке может существовать только одно пустое предложение в семантическом смысле, поскольку, если бы их было более одного (например, e и e'), то $e = e \bullet e' = e'$. Отметим, что по определению пустое предложение интерпретируемо и имеет нулевой смысл (соответствует нулевой семантической структуре). Именно поэтому Word выделяет как ошибку появление в тексте лишних пробелов (которые не выполняют функцию разделения слов) и предлагает удалять эти пустые предложения.

Вторая аксиома (аксиома идемпотентности) утверждает, что повторение одного и того же текста s (предложения) не изменяет семантики текста: $\Psi(C, s) = C' = \Psi(C', s)$ для любой семантики C . В языке это свойство интерпретации отражается равенством $s = s \bullet s$. Например, мы пропускаем повторно пропечатанный текст, поскольку он не несет дополнительной информации. Недаром Word выделяет как ошибку дублирование слова (предложения) в тексте.

На первый взгляд аксиома идемпотентности противоречит пословице «повторение — мать учения». Однако эта пословица означает, что результаты интерпретации впоследствии могут изменить модель и за счет этого скорректировать оператор интерпретации, так что в дальнейшем результаты интерпретации того же текста могут измениться (герменевтический круг). Однако в силу сделанного ранее замечания, на этапе интерпретации текста мы полагаем оператор Ψ неизменным. Это означает, что *внимание*

воспринимающего текст человека или ИИ в процессе интерпретации остается неизменным.

Определение 1. Будем говорить, что предложение s имеет правое отрицание s^{-1} , если интерпретация текста $s \bullet s^{-1}$ при нулевом начальном смысле порождает пустой смысл («Я пойду в магазин. Я не пойду в магазин»). Другими словами, если $C = \Psi(\emptyset, s)$, то $\Psi(C, s^{-1}) = \emptyset$.

В языке обычно мы получаем отрицание предложения, когда ставим «не» перед глаголом. Семантическое свойство правого отрицания в языке выражается правилом $s \bullet s^{-1} = e$, где s — предложение и начальный смысл — нулевой.

Определение 2. Будем говорить, что предложение s имеет безусловное правое отрицание, если для любого смысла C имеют место равенства: если $\Psi(C, s) = C'$, то $\Psi(C', s^{-1}) = C$. Другими словами, $\Psi(C, s \bullet s^{-1}) = C$.

Это семантическое свойство предложения означает, что операция \bullet ассоциативна для текста $s \bullet s^{-1}$, то есть для любого текста t выполняется равенство $(t \bullet s) \bullet s^{-1} = t \bullet (s \bullet s^{-1})$. Разумеется, если операция \bullet в целом ассоциативна, то она порождает языки с безусловными правыми отрицаниями.

Пример 1. В естественных языках свойство безусловности правого отрицания в общем случае не выполняется. Так, в известном фильме «Берегись автомобиля» следователь Максим Подберезовиков, выступая в качестве свидетеля дежний Деточкина, заявляет: «Он, конечно, виноват, но он... не виноват». Эта фраза не вызывает у слушателя ощущения нулевого смысла, поскольку кроме самого фильма, формирующего определенную семантику, Подберезовиков перед этой фразой произносит предложения, которые создают семантику, сохраняющую непустую семантику этой фразы, а именно: он наделяет положительную часть фразы смыслом незаконности действий Деточкина, а отрицательную — смыслом справедливости его действий. В итоге смысл справедливости не может нейтрализовать смысл незаконности и наоборот.

Кроме того, из определения 2 не следует, что у каждого предложения языка должно быть отрицание. Например, у предложения «ну, вот» нет отрицания. В этом также одно из отличий отрицания определения 2 от отрицания в логике.

Безусловность правого отрицания обеспечивает закон двойного отрицания в языке. В самом деле, $(s^{-1})^{-1} = e \bullet (s^{-1})^{-1} = (s \bullet s^{-1}) \bullet (s^{-1})^{-1} = s \bullet (s^{-1} \bullet (s^{-1})^{-1}) = s \bullet e = s$. Если предположить, что логика была абстрагирована из языка, то, скорее всего, прототипом операции дизъюнкции была операция приписывания, а прототипом логического отрицания — операция безусловного отрицания. Как известно, операций дизъюнкции и отрицания достаточно для выражения всех функций алгебры логики. Вместе с тем, имеет место следующая теорема.

Теорема 1. Любое предложение, имеющее безусловное правое отрицание, не имеет смысла.

Доказательство. Пусть предложение s имеет безусловное правое отрицание s^{-1} , тогда $s = s \bullet e = s \bullet (s \bullet s^{-1}) = (s \bullet s) \bullet s^{-1} = s \bullet s^{-1} = e$.

Следствие. Если текст состоит из предложений, имеющих безусловные правые отрицания, то семантика первого предложения равна $\Psi(C, s) = \Psi(C, e) = C$.

То же получим для второго и последующего предложения текста. В частности, для $C = \emptyset$ получаем пустую интер-

претацию текста. Если какие-либо предложения языка имеют безусловные правые отрицания, то верно следующее утверждение: в любом языке любой текст, составленный из предложений с безусловными правыми отрицаниями, не имеет смысла.

Поскольку предложения естественных языков имеют смысл, то безусловные правые отрицания в языке отсутствуют, а с ними не выполняется и правило двойного отрицания в том смысле, что два отрицания эквивалентны положительному утверждению.

Например, известно [4], что «не» с подлежащего или дополнения можно выносить на глагол, так что из предложения «я не пойду не в кино» следует «я не не пойду в кино». Из первого предложения следует, что человек может пойти в кино или куда не пойти, но, в соответствии со вторым предложением, при допущении двойного отрицания один из вариантов отпадает.

Проверим это в диалоге с виртуальным собеседником Chat GPT 4 (ИИ).

Человек. Что означает фраза «Я не пойду не в кино»?

GPT 4. Эта фраза имеет двойное отрицание, что означает, что человек пойдет в кино.

Итак, ИИ допускает двойное отрицание и, значит, в силу теоремы 1 может создавать пустую семантику даже там, где ее нет.

Возьмем теперь два предложения, которые формально отрицают друг друга. Как ИИ оценит семантику в этом случае?

Человек. Какая из фраз истинна: «У Кутузова был один глаз», «У Кутузова не было одного глаза»?

GPT 4. Как ИИ, я могу подтвердить, что обе фразы описывают отсутствие одного глаза у Кутузова. Разница только в формулировке: одна фраза утверждает, что у него был один глаз, а другая — что не было одного глаза. Но обе выражают одно и то же.

Итак, в случае с одинарным отрицанием GPT 4 справился отлично. И последняя проверка — на скрытые отрицания.

Человек. Что означает фраза «В моей практике не было такого, чтобы такого не было»?

GPT 4. Эта фраза означает, что в опыте говорящего не было случаев, которые бы полностью исключали возможность определенного события или явления. Другими словами, говорящий утверждает, что все возможно в его практике.

И мы опять видим, что GPT 4 при нескольких отрицаниях семантически «плывет». Это имеет прямое отношение к его способу формирования смыслов.

Теорема 1 очень важна в понимании семантики языка. Семантика «склеивает» предложения в единое целое. В этом ее важнейшая функция. Даже самая незначительная попытка разорвать (упростить) эту связь приводит к ликвидации семантики, а значит, и возможности интерпретации языка в некоторой модели. Действительно, наличие безусловного правого отрицания всего лишь позволяет проинтерпретировать текст $s \bullet s^{-1}$ отдельно, независимо от контекста. Сохраняется порядок предложений в тексте, его последовательная интерпретация и т. д. И все же это разрушает семантику текста. В самом деле, пусть предложение s имеет семантику C . Аксиома идемпотентности

утверждает, что предложение $s*s$ имеет такую же семантику S . То есть семантики предложений как бы накладываются друг на друга («сцепляются») и, если они совпадают, дают тот же результат. Но тогда приписка предложения-отрицания s^{-1} должна уничтожить семантику S независимо от того, как она была сформирована: через s или $s*s$. В итоге мы получаем нулевую семантику, которая соответствует пустому предложению. Но та же самая последовательность, вычисленная как $s*(s*s^{-1})$, порождает семантику S , а это значит, что семантика S нулевая. Иначе нам пришлось бы признать, что либо у одного и того же предложения могут быть разные семантики (при одинаковом Ψ), либо семантики не «накладываются» друг на друга. То есть это бы означало, что семантика не «склеивает» смыслы предложений. Кроме того, это означает, что приписка s^{-1} формально воздействует не только на ближайшее слева предложение s , но и на предшествующее предложение s . Другими словами, приписывание отрицательных предложений может иметь эффект распространения влияния на интерпретацию предшествующего текста, а не только ближайшего предложения, что является одной из причин множества параметров в GPT 4, которые учитывают отдаленный уже прошедший текст.

Все сказанное ранее в полной мере относится к ассоциативным языкам, а предположение об ассоциативности сплошь и рядом встречается в лингвистике, хотя такие языки не могут иметь никакой интерпретации, если конечно в этих языках существует отрицание предложений.

Завершая семантический обзор для языковых моделей отметим еще два обстоятельства. Поскольку GPT 4 фактически не имеет модели мира, а имеет модель языка, которая является промежуточным, обслуживающим элементом для модели мира, у него отсутствует семантика, связанная с разным мышлением [5]. Поясним это на примере.

Человек. Что находится севернее Северного полюса?

GPT 4. Северный полюс является самой северной точкой на Земле, поэтому севернее него — это уже вода: Арктический океан с островами и архипелагами, такими как Шпицберген, Северная Земля, Земля Франца-Иосифа, Новая Земля и др.

По этой же причине GPT 4, как правило, плохо отгадывает загадки.

Человек. Отгадай загадку: без окон, без дверей — полна горница людей.

GPT 4. Попробуйте задать вопрос снова.

В дальнейшем при различных вариантах формулировки этого вопроса был получен тот же ответ.

Второе обстоятельство связано с коммуникативной семантикой, под которой понимаются смыслы, связанные со взаимодействием различных моделей у разных людей для реализации совместной деятельности. Поясним это на примере следующего диалога:

— Ну так как там, что там? Давай уже... Я тут уже вообще не знаю...

— Ну, понимаешь, там не очень... Там такое дело, что даже не знаю теперь...

— Что, совсем-совсем никак? Да?

— Ну, в общем, честно говоря, да!

— Ну и чего ж теперь? А? Вот ведь!

— Правда, знаешь, там может быть один еще вариант, но это если только Петр Михайлович сам...

— А! Ну так и ничего! Пускай Петр Михайлович... Не вопрос... Нет. А что такого? Серьезно. Это как-то даже и лучше будет.

В приведенном диалоге у двух собеседников есть модели обсуждаемой ситуации, и эти модели практически не отличаются. Первый собеседник интересуется, произошло ли событие, которое они прогнозировали исходя из своих моделей. Его интерес эмоциональный, потому что для него значимо достижение цели развития этой ситуации. По этой причине он недоволен затягиванием диалога вторым собеседником. Далее собеседники обсуждают коррекцию технологии достижения целевого состояния. Заметим, что здесь не обсуждается ни ситуация, ни ее признаки, ни модель, ни целевое состояние. Обсуждается неизвестная нам (но известная собеседникам) технология достижения цели, которая в какой-то момент реализации дала сбой из-за расхождения модельных прогнозов и реальной действительности.

Поскольку GPT 4 не имеет собственной модели мира, а только модель языка, он «понимает» коммуникативную семантику лишь в языковом смысле.

Человек. Ну ты даешь! Что, теперь вообще никак? И даже как раньше нельзя?

GPT 4. Извините, я не совсем понимаю ваш вопрос. Можете уточнить, о чем именно вы спрашиваете?

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ ИИ ДЛЯ ИЗВЛЕЧЕНИЯ СМЫСЛОВ

К числу современных инструментов для извлечения смыслов из текстов (вычисления семантики предложений) следует отнести последовательные системы искусственного интеллекта (ПСИИ) — это системы, в которых некоторые действия следуют за другими, и результат вычислений предыдущего шага служит входом для следующего.

Обобщенная схема применения ПСИИ включает:

1. Разбиение задачи на последовательность подзадач.

2. Использование специализированных алгоритмов (методов) для решения каждой подзадачи.

3. Организацию последовательности действий таким образом, чтобы результат выполнения каждой подзадачи стал входом для следующей.

4. Непрерывное тестирование и улучшение системы на основе полученных результатов.

Для построения эффективной ПСИИ важно правильно выбрать алгоритмы и методы решения каждой подзадачи, чтобы обеспечить наилучшие результаты при минимальных затратах ресурсов.

ПСИИ могут применяться для различных целей, таких как обработка естественного языка, распознавание образов, компьютерное зрение, управление роботами, принятие решений в режиме реального времени и т. д. Результатом успешной работы ПСИИ является повышение эффективности и точности выполнения задач, автоматизация процессов и увеличение производительности.

Разработка ПСИИ может включать следующие шаги:

1. Определение целей. Перед началом работы над последовательной системой ИИ необходимо четко определить цели, которых эта система должна достигнуть.

2. Подбор датасетов. Для обучения последовательной системы ИИ необходимо отобрать соответствующие датасеты с помощью анализа данных, связанных с основными задачами системы.

3. Выделение признаков. После выбора соответствующего датасета требуется определить самые значимые признаки для обучения системы.

4. Обучение модели. На этом этапе данные используются для обучения системы на основе алгоритмов машинного обучения, таких как нейронные сети, метод опорных векторов и решающие деревья.

5. Тестирование. После завершения обучения системы необходимо проверить ее работоспособность на датасетах, которые не были использованы в процессе обучения.

6. Анализ результатов. На этом этапе анализируются результаты тестирования, и, при необходимости, система дорабатывается и оптимизируется.

7. Применение. После завершения всех предыдущих этапов система готова к применению в практических задачах.

Одной из возможных реализаций ПСИИ является рекуррентная архитектура ChatGPT, состоящая из комбинации двух архитектур: Transformer и рекуррентных нейронных сетей (Recurrent Neural Network, RNN).

В отличие от классических рекуррентных нейронных сетей, которые могут столкнуться с проблемой затухающе-

го градиента и забывать далекие зависимости, ChatGPT использует «длинную краткосрочную память» (Long Short-Term Memory, LSTM), которая позволяет модели запоминать длинные зависимости и глубокие контекстные связи между словами.

Таким образом, рекуррентная архитектура ChatGPT сочетает в себе преимущества Transformer и LSTM, что позволяет модели изучать контекст, создавать более точные представления текста и генерировать более интуитивные ответы на основе понимания контекста.

РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ

Рекуррентные нейронные сети являются классом искусственных нейронных сетей, которые состоят из повторяющихся блоков и используются для анализа последовательностей данных, где входы могут быть произвольной длины [6, 7]. В отличие от обычных нейронных сетей, которые могут работать только с фиксированным количеством входных данных, RNN могут обрабатывать и моделировать последовательности переменной длины, например текст или звуковые сигналы (рис. 1).

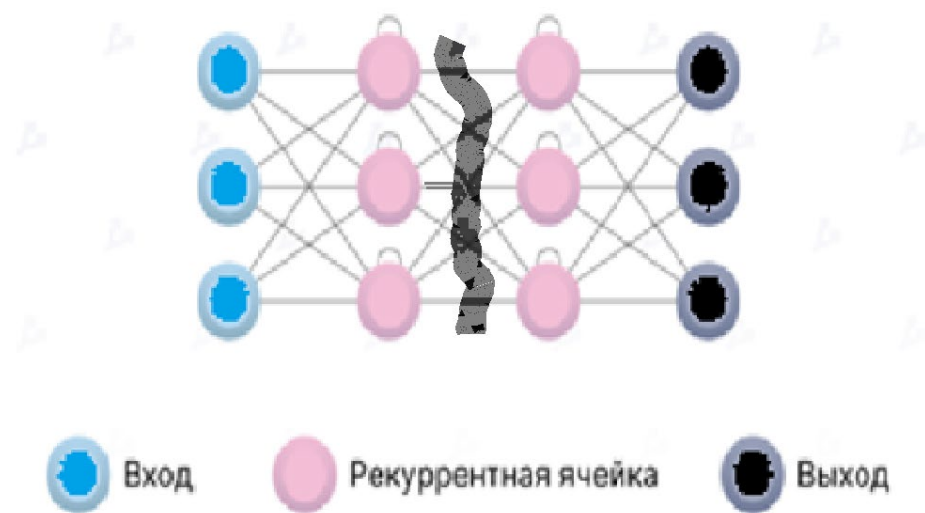


Рис. 1. Рекуррентная нейронная сеть

В отличие от других типов нейронных сетей, RNN обладает возможностью запоминания прошлой информации и использования ее при анализе последующих данных. Они обрабатывают входные данные порциями, сохраняют и передают внутреннее состояние (память) между последовательностями, что позволяет учитывать историю при принятии решений в текущем моменте.

Основными компонентами RNN являются скрытые слои, которые принимают на вход данные от предыдущего шага и вычисляют новое скрытое состояние (память) для передачи на следующий шаг. При этом веса каждого слоя являются общими для всей последовательности данных. Другими словами, каждый слой не просто обрабатывает данные, но и совместно с ранее просмотренными слоями использует их, чтобы создать более осмысленный анализ и вывод.

RNN успешно применяются в разных задачах: в обработке текста (например, в задачах машинного перевода), распознавании речи, создании рекомендательных систем, генерации текстов и т. д. Однако у RNN есть свои проблемы, такие как «затухающий градиент», при котором

влияние более ранних данных на конечный результат исчезает с каждым последующим шагом. Для решения этих и других проблем рекуррентных нейронных сетей были разработаны более совершенные и эффективные модели, такие как LSTM и GRU (Gated Recurrent Unit).

ТРАНСФОРМЕР — УНИВЕРСАЛЬНЫЙ ВЫЧИСЛИТЕЛЬНЫЙ МЕХАНИЗМ

Трансформер (Transformer) — это название архитектуры нейросети, предложенной исследователями Google в 2017 году.

Transformer используется для извлечения признаков из текстовых данных и создания представления контекста. Он работает путем многократной обработки последовательности входных данных, называемой энкодером, и создания скрытых представлений для каждого элемента последовательности [8].

Transformer отличается от RNN тем, что он не использует рекуррентность для моделирования последовательностей (рис. 2).

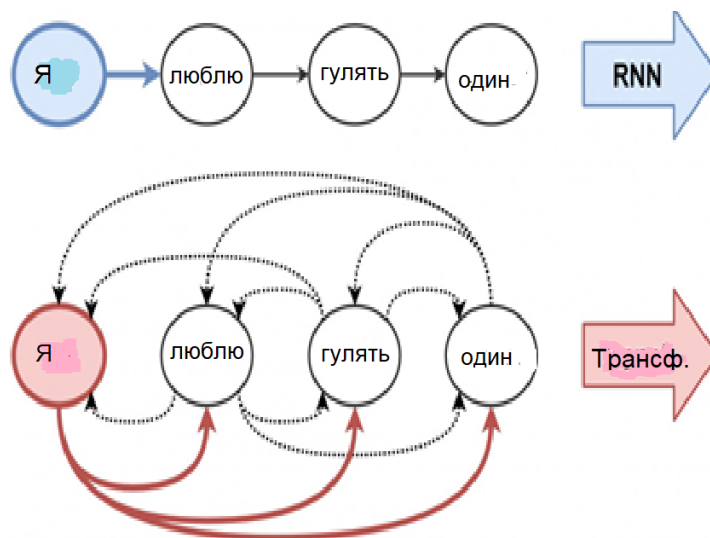


Рис. 2. Отличие Трансформера от рекуррентной нейронной сети

Transformer использует механизм внимания (attention mechanism, attention model) для вычисления весов, которые моделируют отношения между элементами последовательности, и позволяют модели более эффективно использовать контекст. Он состоит из двух основных компонентов — энкодера и декодера.

Универсальный вычислительный механизм Трансформера просто описать: он принимает на вход один набор последовательностей (данных) и выдает на выходе тоже набор последовательностей, но уже другой, преобразованный по некоему алгоритму [9].

Так как текст, картинки и звук можно представить в виде последовательностей чисел, то с помощью Трансформера можно решать практически любые задачи.

Главная особенность Трансформера заключается в его удобстве и гибкости: он состоит из простых модулей-блоков, которые очень легко масштабировать. Если старые, дотрансформерные языковые модели начинали требовать слишком много ресурсов, когда их пытались заставить работать быстро и много слов за один раз, то нейросети-трансформеры справляются с этой задачей гораздо экономнее.

Нейронная сеть Transformer состоит из нескольких блоков, каждый из которых имеет свою структуру. Общая структура нейронной сети может быть описана следующим образом:

1. Входной слой, который принимает входные данные и векторизует их.
2. Предобработка, включающая в себя нормализацию и преобразование данных.
3. Энкодер, который разбивает входные данные на несколько последовательностей (например, токенов) и преобразует каждую последовательность в вектор фиксированной длины.
4. Декодер, который генерирует выходные данные на основе векторов, созданных энкодером.
5. Выходной слой, который генерирует финальный ответ.

Блоки энкодера и декодера в нейронной сети Transformer включают в себя слои многоканальной свертки, слои само-внимания и полносвязные слои. Все эти

блоки работают вместе для достижения высокой точности в задачах обработки текста и генерации языка.

АЛГОРИТМ ОБУЧЕНИЯ ТРАНСФОРМЕРА

Алгоритм обучения Трансформера включает в себя следующие шаги [10]:

1. Подготовка данных. На этом шаге данные должны быть представлены в виде последовательности токенов, каждый из которых имеет свой уникальный идентификатор.
2. Построение модели. Трансформер состоит из нескольких слоев, каждый из которых содержит множество механизмов внимания. Модель может быть построена с использованием открытых библиотек машинного обучения, таких как TensorFlow или PyTorch.
3. Инициализация модели. Веса модели инициализируются случайными значениями.
4. Определение функции потерь. Функция потерь определяет, насколько хорошо модель предсказывает правильный ответ. Наиболее распространенными функциями потерь для задач обработки естественного языка являются перекрестная энтропия и среднеквадратичная ошибка.
5. Обучение модели. Модель обучается на тренировочных данных с помощью метода обратного распространения ошибки. Во время обучения модель пытается минимизировать функцию потерь с помощью градиентного спуска.
6. Оценка модели. После обучения модель оценивается на тестовых данных, чтобы определить ее точность и эффективность.
7. Тюнинг модели. Если модель недостаточно эффективна, ее можно доработать, изменяя гиперпараметры.
8. Использование модели. После обучения модель может быть использована для предсказания ответов на новых данных.

Трансформерный блок и токен являются разными элементами в машинном обучении.

Трансформерный блок — это алгоритм, который преобразует входные данные в выходные, обычно с помощью нейронной сети. Он может использоваться, например, для решения задачи классификации или регрессии.

Токен — это отдельный элемент входных данных, который может быть вектором, числом или символом. Он может быть использован для представления слов, символов или фраз в тексте.

Таким образом, трансформенный блок используется для обработки всей входной последовательности, в то время как токены используются для представления отдельных элементов входных данных.

ЗАКЛЮЧЕНИЕ

Модель GPT — одна из наиболее эффективных нейронных сетей для вычисления смыслов. Модель обучена на огромных объемах текстовых данных.

Эффект возникновения «осмысленных» текстов в данной модели, очевидно, связан с преодолением известного порога в количестве параметров нейронной сети. Этот эффект требует дальнейших исследований и доказательств.

Отметим некоторые ключевые перспективы развития модели GPT:

1. Увеличение размера модели. Последние версии модели GPT имеют уже достаточно большое количество (миллиарды) параметров, но можно ожидать что они будут еще больше в будущем.

2. Увеличение контекста. Существующие версии модели GPT используют контекст порядка нескольких сотен слов, однако в будущем, вероятно, контекст будет расширяться.

3. Улучшение архитектурных возможностей. Последовательная подача данных между слоями модели типа GPT ограничивает работу сети с длинными последовательностями данных. Будущее развитие модели GPT потребует продвинутых архитектурных решений, например гибридных моделей.

В целом возможности для развития модели GPT в будущем огромны и будут включать в себя техническое совершенствование, изучение дополнительных интересных возможностей и применение в разных задачах.

В дальнейшем ожидается, что развитие модели GPT приведет к более точной генерации текстов, нейронные сети GPT смогут стать более компактными и легкими в работе, произойдет увеличение размера обучаемой модели GPT, развитие модели GPT будет значимо связано с поиском и развитием новых подходов к прикладным задачам.

ЛИТЕРАТУРА

1. Шлейермахер, Ф. Д. Э. Академические речи 1829 года / пер. с нем. Е. М. Ананьевой // *Метафизические исследования*. — Санкт-Петербург: Алетейя, 1997. — Вып. 3. — С. 242–260; Вып. 4. — С. 147–163.

2. Худяков, А. А. Теоретическая грамматика английского языка: Учебное пособие. — Москва: Академия, 2005. — 256 с. — (Высшее профессиональное образование).

3. Внимание — все, что вам нужно: как работает attention в нейросетях // Системный Блок. — 2020. — 24 июня. URL: <http://sysblok.ru/knowhow/vnimanie-vse-cto-vam-nuzhno-kak-rabotaet-attention-v-nejrosetjah> (дата обращения 16.05.2023).

4. Polyakov, O. M. Linguistic Data Model for Natural Languages and Artificial Intelligence. Part 7. Internal Logic 2 // *Дискурс*. 2022. Т. 8, № 2. С. 98–112.

DOI: 10.32603/2412-8562-2022-8-2-98-112.

5. Блюм, В. С. Диалог о диалоге с виртуальным собеседником / В. С. Блюм, О. М. Поляков // *Актуальные проблемы экономики и управления*. 2022. № 1 (33). С. 81–88.

6. A Novel Connectionist System for Unconstrained Handwriting Recognition / A. Graves, M. Liwicki, S. Fernández, [et al.] // *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2009. Vol. 31, Is. 5. Pp. 855–868. DOI: 10.1109/TPAMI.2008.137.

7. Recurrent Neural Network Tutorial, Part 4 — Implementing a GRU and LSTM RNN with Python and Theano // Denny's Blog. — 2015. — 27 October. URL: <http://dennybritz.com/posts/wildml/recurrent-neural-networks-tutorial-part-4> (дата обращения 18.05.2023).

8. Брагин, А. Большие языковые модели как новый инструмент в научной работе // Хабр. — 2023. — 03 апреля. URL: <http://habr.com/ru/articles/726606> (дата обращения 22.05.2023).

9. Трансформер // Викиконспекты. — Обновлено 04.09.2022.

URL: <http://neerc.ifmo.ru/wiki/index.php?title=Трансформер> (дата обращения 14.10.2023).

10. Алгоритмы обучения нейронной сети: наиболее распространенные варианты // GeekBrains. — 2022. — 11 января. URL: <http://gb.ru/blog/algorithmy-obucheniya-nejronnoj-seti> (дата обращения 21.10.2023).

Semantic Principles of Text Development Artificial Intelligence Systems

PhD V. S. Blyum, PhD V. M. Kosmachev, PhD O. M. Polyakov
Saint Petersburg State University of Aerospace Instrumentation
Saint Petersburg, Russia
vladblum7@gmail.com, kvm@aanet.ru, polyakovoleg@yandex.ru

Abstract. The purpose of the article is to determine the theoretical and technological foundations for calculating the semantics of texts. It has been proven that any sentence that has an unconditional right negation does not make sense. The principles of constructing the main modern generators of «meaningful» texts, sequential artificial intelligence systems, and methods of their application are considered. The structure and algorithms of recurrent neural networks and the universal neural network computing mechanism Transformer are considered. The novelty of the approach lies in the fact that the calculation of the semantics of the generated text is based on a large language model.

Keywords: semantics, language model, artificial intelligence.

REFERENCES

1. Schleiermacher F. D. E. Academic speeches of 1829 [Akademicheskie rechi 1829 goda], *Metaphysical Research [Metafizicheskie issledovaniya]*. St. Petersburg, Aletheia Publishing House, 1997. Is. 3, Pp. 242–260; Is. 4, Pp. 147–163.
2. Khudyakov A. A. Theoretical grammar of the English language: Study guide [Teoreticheskaya grammatika angliyskogo yazyka: Uchebnoe posobie]. Moscow, Academia Publishing House, 2005, 256 p.
3. Attention Is All You Need: How Attention Works in Neural Networks [Vnimanie — vse, chto vam nuzhno: kak rabotaet attention v neyrosetyakh], *System Unit [Sistemnyy Blok]*. Published online at June 24, 2020. Available at: <http://sysblok.ru/knowhow/vnimanie-vse-chto-vam-nuzhno-kak-rabotaet-attention-v-nejrosetjah> (accessed 16 May 2023).
4. Polyakov O. M. Linguistic Data Model for Natural Languages and Artificial Intelligence. Part 7. Internal Logic 2, *Discourse [Diskurs]*, 2022, Vol. 8, No. 2, Pp. 98–112. DOI: 10.32603/2412-8562-2022-8-2-98-112.

5. Blyum V. S., Polyakov O. M. Dialogue About Dialogue with a Virtual Speaker [Dialog o dialoge s virtualnym sobesednikom], *Actual Problems of Economics and Management [Aktualnye problemy ekonomiki i upravleniya]*, 2022, No. 1 (33), Pp. 81–88.

6. Graves A., Liwicki, M., Fernández S., et al. A Novel Connectionist System for Unconstrained Handwriting R, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009, Vol. 31, Is. 5, Pp. 855–868. DOI: 10.1109/TPAMI.2008.137.

7. Recurrent Neural Network Tutorial, Part 4 — Implementing a GRU and LSTM RNN with Python and Theano, *Denny's Blog*. Published online at October 27, 2015. Available at: <http://dennybritz.com/posts/wildml/recurrent-neural-networks-tutorial-part-4> (accessed 18 May 2023).

8. Bragin A. Large Language Models as a New Tool in Scientific Work [Bolshie yazykovye modeli kak novyy instrument v nauchnoy rabote], *Habr [Khabr]*. Published online at April 03, 2023. Available at: <http://habr.com/ru/articles/726606> (accessed 22 May 2023).

9. Transformer, *Wikinotes [Vikikonspekty]*. Last update at September 04, 2022. Available at: <http://neerc.ifmo.ru/wiki/index.php?title=Трансформер> (accessed 14 Oct 2023).

10. Neural Network Learning Algorithms: The Most Common Options [Algoritmy obucheniya neyronnoy seti: naibolee rasprostrannyye varianty], *GeekBrains*. Last update at January 11, 2022. Available at: <http://gb.ru/blog/algoritmy-obucheniya-nejronnoj-seti> (accessed 21 Oct 2023).

Внедрение системы работы с машиночитаемыми доверенностями через средства API в системе 1С:Документооборот

Д. А. Бобриков, Н. Калиневич

Петербургский государственный университет путей сообщения Императора Александра I
Санкт-Петербург, Россия
daniil20001210@mail.ru, Kalinevich2015@yandex.ru

Аннотация. Рассматривается метод внедрения системы работы с машиночитаемыми доверенностями с использованием API в системе 1С. Исследуются технические и организационные аспекты интеграции, а также выявляются преимущества и потенциальные вызовы данного подхода.

Ключевые слова: машиночитаемая доверенность, 1С:Документооборот, интеграция, управление доверенностями, API.

ВВЕДЕНИЕ

Внедрение системы работы с машиночитаемыми доверенностями через средства API в систему 1С представляет собой важный шаг в совершенствовании управления доверенностями и автоматизации бизнес-процессов. Система 1С широко используется в различных организациях и играет ключевую роль в учете, финансах, логистике и других аспектах деятельности. Однако, управление доверенностями и авторизацией часто остается слабым местом в этих системах, что может привести к утечкам данных, ошибкам и низкой эффективности.

В условиях растущей важности цифровой безопасности и требований к защите данных, возникает потребность в разработке более надежных и гибких механизмов управления доверенностями. Использование API для работы с машиночитаемыми доверенностями позволяет повысить уровень безопасности, автоматизировать процессы и сделать систему более гибкой и отзывчивой на потребности бизнеса.

Также в связи с тем, что в 2022 году вступили в силу поправки в Федеральный закон от 06.04.2011 № 63-ФЗ, которые вводят новые правила работы с электронными документами. Сотрудники должны подписывать документы с помощью сертификата электронной подписи (ЭП) физического лица. Для подтверждения полномочий сотрудника и его принадлежности к компании важно при подписании указать машиночитаемую доверенность (МЧД). Соответственно появляется необходимость интеграции системы работы с МЧД в системах 1С.

Целью данного исследования является разработка методики и оценка эффективности внедрения системы работы с машиночитаемыми доверенностями через средства API в систему 1С. Для достижения этой цели были сформулированы следующие задачи:

1. Рассмотреть существующие методы управления доверенностями в системе 1С и выявить их ограничения.
2. Разработать API для работы с машиночитаемыми доверенностями, учитывая специфику системы 1С.
3. Интегрировать разработанное API в существующие информационные процессы и бизнес-приложения на базе 1С.
4. Провести анализ эффективности внедрения API, оценив преимущества и вызовы данного подхода.

ОБЗОР СИСТЕМЫ 1С

Система 1С:Документооборот представляет собой высокопроизводительное программное обеспечение, разработанное компанией 1С для автоматизации управления документами и бизнес-процессами. Эта система обеспечивает упорядоченное создание, обработку, передачу и архивирование документов, что делает ее важной частью информационной инфраструктуры многих организаций. Система может быть адаптирована под разнообразные нужды и предоставляет широкий спектр функций, включая учет и управление документами, автоматизацию рабочих процессов и совместную работу над документами [1].

Роль системы в управлении доверенностями

Управление доверенностями в контексте системы 1С означает управление доступом пользователей к различным функциональным элементам системы. Это может включать в себя:

1. Определение прав доступа. Установление, какие пользователи или группы пользователей имеют доступ к определенным разделам системы, к данным и функциям.
2. Аутентификация и авторизация. Проверка подлинности пользователей и установление их прав на выполнение определенных операций.
3. Мониторинг действий. Система 1С может регистрировать действия пользователей, что помогает в аудите безопасности и обнаружении некорректного использования.
4. Управление сессиями. Контроль и управление сессиями пользователей для обеспечения безопасности.

Система 1С играет важную роль в управлении доверенностями, так как она определяет, кто и как может взаимодействовать с данными и процессами внутри системы. Это имеет решающее значение для обеспечения безопасности и конфиденциальности информации, а также для

предотвращения несанкционированного доступа и изменения данных [2].

Машиночитаемая доверенность

Машиночитаемая доверенность — это электронный документ в формате XML, подписанный электронной подписью руководителя организации. Документ содержит данные о доверителе (организация, которая выдала доверенность), доверенном лице (физическое лицо, которое уполномочено совершать действие) и полномочиях. МЧД необходимо передавать вместе с каждым документом, который был подписан сертификатом физического лица [3].

Для руководителей организаций сертификат физического лица и МЧД не требуются.

В машиночитаемой доверенности используются два основных понятия:

- **доверитель** — ответственное лицо, разрешающее выполнение операций другим организациям или физическим лицам по доверенности;

- **представитель** — организация или физическое лицо, принимающее на себя ответственность действовать от лица доверителя в рамкахверяемых ему полномочий по доверенности. Им может быть как штатный сотрудник организации, так и сторонний представитель (физическое или юридическое лицо или ИП) [4].

МЧД будет использоваться при подписании документа и действий с документом, таких как:

- отказ в подписи;
- запрос аннулирования;
- отказ в аннулировании;
- аннулирование;
- запрос уточнения;
- подписание извещения о получении документа.

Текущие вызовы и ограничения

Несмотря на важность системы IC в управлении доверенностями и документооборотом, существуют ряд вызовов и ограничений:

1. Сложность настройки прав доступа. В больших организациях с множеством пользователей и разнообразными ролями может возникнуть сложность при настройке и управлении правами доступа. Это может потребовать значительных усилий и времени.

2. Ограниченная автоматизация. Несмотря на широкий функционал IC:Документооборот, автоматизация управления доверенностями и секретными операциями часто остается ограниченной и требует дополнительной работы.

3. Совместимость и интеграция. Интеграция системы IC с другими системами и стандартами безопасности может стать вызовом, особенно в больших экосистемах информационных технологий.

4. Безопасность данных. Защита данных и управление рисками важны, но система IC может оказаться недостаточно гибкой для некоторых требований по безопасности.

Учитывая эти трудности и ограничения, становится актуальным углубление в изучение и реализацию механизма использования машиночитаемых доверенностей через инструменты API. Такое внедрение позволит улучшить управление доверенностями и одновременно снизить потенциальные опасности, связанные с доступностью данных и работоспособностью системы.

ИНТЕГРАЦИЯ API В СИСТЕМУ IC

Рассмотрим процесс интеграции API для работы с машиночитаемыми доверенностями в систему IC:Документооборот. Интерфейс прикладного программирования (API) представляет собой набор программных методов и средств, позволяющих взаимодействовать с системой и выполнять различные операции. API для работы с доверенностями включает в себя следующие ключевые функции:

1. Аутентификация и авторизация. API позволяет проводить проверку подлинности пользователей и управлять их правами доступа в системе IC:Документооборот.

2. Управление доверенностями. API предоставляет возможность создания, редактирования, удаления и управления доверенностями в системе. Это включает в себя установку прав доступа, а также определение сроков действия и других параметров доверенности.

3. Мониторинг и аудит. API позволяет регистрировать действия пользователей и проводить мониторинг безопасности. Это важно для отслеживания всех операций, связанных с доверенностями.

4. Управление сессиями. API обеспечивает контроль над сессиями пользователей, что позволяет управлять активностью пользователей и их доступом к системе [5].

Технические аспекты интеграции

Интеграция API в систему IC:Документооборот требует внимания к ряду технических аспектов. Эти аспекты включают в себя:

1. Типы API. Определение, какие виды API будут использоваться для работы с доверенностями. Это может включать RESTful API, SOAP API, GraphQL или другие технологии, в зависимости от требований проекта.

2. Аутентификация и безопасность. Разработка механизмов аутентификации и обеспечение безопасности API для защиты данных и операций. Это может включать в себя использование токенов, OAuth, SSL-шифрования и других методов.

3. Интеграция с базой данных. Взаимодействие API с базой данных системы IC, чтобы обеспечить доступ к данным о доверенностях и пользователях.

4. Обработка ошибок и исключений. Разработка механизмов для обработки ошибок и исключений, которые могут возникнуть в процессе интеграции. Это позволяет предотвратить сбои и недоступность системы [6].

Проектирование и разработка API

Проектирование API включает в себя следующие этапы:

1. Определение функциональности. Определение всех функций и методов, которые будут доступны через API. Это включает в себя создание спецификации, которая описывает, какие операции могут быть выполнены.

2. Определение формата данных. Определение формата данных, который будет использоваться для передачи информации между системой IC и внешними приложениями через API. Это может включать в себя форматы JSON, XML или другие.

3. Разработка API. Разработка программного кода, который реализует API, включая логику работы методов, аутентификацию, авторизацию и обработку данных.

4. Документирование API. Создание документации, которая описывает методы, форматы данных, примеры использования и другие сведения о API. Это помогает другим разработчикам понять, как использовать API [6].

Интеграция API в систему 1С:Документооборот имеет важное значение для обеспечения безопасности и эффективности работы с доверенностями и документами. Эти технические элементы оказывают решающее влияние на бесперебойную работу системы, которая облегчает работу с машиночитаемыми доверенностями с использованием инструментов API.

ПРАКТИЧЕСКИЙ ОПЫТ ВНЕДРЕНИЯ В СИСТЕМУ 1С

В процессе интеграции 1С и API в первую очередь необходимо изучить информацию о сервисе, к которому будет подключение, какие методы будем использовать в работе и необходимость интеграции. После изучения основных вопросов нужно получить доступ к сервису. Далее подключившись к тестовому сервису (если он имеется), попытаться отправить первые данные.

В процессе разработки API, обращаясь к документации, получаем ссылку на метод, а также тип метода (Get и Post), к которому будем обращаться. Для начала получаем адрес сервера и с помощью Метода HTTP Соединение создаем соединение с сервером [7]. После чего обращаемся по ссылке метода из документации в HTTP Запросе в виде заголовка вместе с токеном, по необходимости передаем другие параметры в заголовке и теле запроса. После успешной отправки запроса на сервер получаем ответ по успешной или не успешной операции.

Интеграция с системой осуществляется на уровне роботов или запросов при определенных действиях. Поскольку реализуется интеграция 1С Документооборот с API для МЧД, процесс создания файла осуществляется через API, также как процессы регистрации, валидации, импорта, скачивания и отзыва.

В процессе тестирования необходимо проверить: корректность получаемых и отправляемых данных, надежность к изменениям в системе и валидность использования API.

Примеры использования API в реальных сценариях

В качестве примера приведем интеграцию системы 1С:Документооборот и Контур.Доверенность.

В первом методе получаем уникальный номер организации, к которой будут привязаны МЧД в системе Контур.Доверенность:

1. Получаем основную информацию для обращения по API: адрес сервера, токен, адрес метода к которому будем обращаться:

```
АдресСервера = ВозвратАдресаСервера();
КлючРазработчика = ВозвратКлючаРазработчика();
АдресРесурса = "/v1/organizations";
```

2. Обращаемся в запросе к нашему методу:

```
Запрос = Новый HTTPЗапрос(АдресРесурса);
Запрос.Заголовки.Вставить("X-Kontur-APIkey",
    КлючРазработчика);
Результат = Соединение.Получить(Запрос);
```

3. Обрабатываем ответ (рис. 1–3):

```
Если Результат.КодСостояния = 200 Тогда
    Организация = "Организация существует";
Иначе
    Организация = "Ошибка в получения организации";
КонецЕсли;
```



Рис. 1. Строка до получения результата

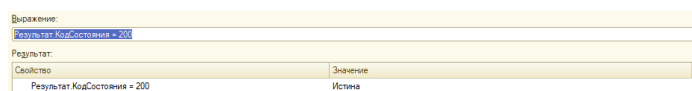


Рис. 2. Получение успешного ответа от сервера

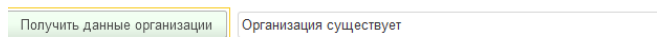


Рис. 3. Строка после получения результата

Во втором методе получаем данные передаваемой доверенности организации:

1. Получаем основную информацию для обращения по API: адрес сервера, Токен, адрес метода к которому будем обращаться:

```
АдресСервера = ВозвратАдресаСервера();
КлючРазработчика = ВозвратКлючаРазработчика();
АдресРесурса = СтрЦльблон("/v1/organizations/%1/poas", OrganizationId);
```

2. Получаем ID организации (из прошлого метода):

```
действОрг = ПолучениеОрганизации().ОтветСервера.organizations.items;
Для каждого стр Из действОрг Цикл
    OrganizationId = стр.id;
КонецЦикла;
```

3. Далее также передаем данные в заголовки и обрабатываем ответ сервера (рис. 4).

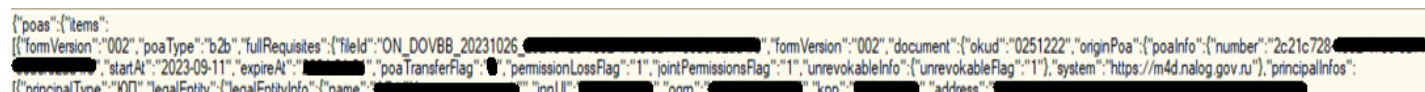


Рис. 4. Тело ответа от сервера при успешном обращении

ПРЕИМУЩЕСТВА И ПРАКТИЧЕСКАЯ ПОЛЬЗА

Интеграция системы работы с машиночитаемыми доверенностями через средства API в систему 1С:Документооборот приносит множество практических польз и выгод, например:

1. Повышение эффективности управления доверенностями. Интеграция системы работы с машиночитаемыми доверенностями через средства API в систему 1С:Документооборот приносит значительные преимущества в управлении доверенностями. Эффективность управления доверенностями повышается благодаря следующим факторам:

- быстрый доступ и авторизация. Пользователи получают быстрый и безопасный доступ к данным и ресурсам, что снижает временные затраты на ожидание разрешения доступа;

- гибкость и адаптивность. API позволяют быстро адаптировать систему к изменяющимся требованиям и ролям пользователей, что способствует более эффективной управляемости доверенностями;

- снижение бюрократии. Процессы авторизации и управления доверенностями становятся менее бюрократичными и меньше зависят от ручных вмешательств.

2. Увеличение автоматизации процессов. Интеграция API в систему 1С:Документооборот содействует увеличению автоматизации бизнес-процессов и управления доверенностями:

- автоматическая выдача доверенностей. Процессы выдачи и обновления доверенностей могут быть автоматизированы, что устраняет необходимость вручную управлять доступом;

- система уведомлений и рассылок. API может включать в себя функциональность оповещения и уведомлений, что сокращает вероятность пропуска важных событий;

- интеграция с другими системами. Интеграция API позволяет системе 1С взаимодействовать с другими системами, что дополнительно увеличивает автоматизацию бизнес-процессов.

3. Уменьшение рисков и ошибок. Использование API для управления доверенностями в системе 1С:Документооборот снижает риски и вероятность ошибок:

- большая надежность. API обеспечивает точное и надежное выполнение операций, устраняя человеческие ошибки;

- легкий мониторинг и аудит. Интегрированные механизмы мониторинга и аудита позволяют оперативно выявлять и устранять возможные проблемы и нарушения безопасности;

- быстрая реакция на угрозы. Система с API может реагировать на потенциальные угрозы и автоматически принимать меры для их предотвращения.

4. Экономические и временные выгоды. Внедрение системы работы с машиночитаемыми доверенностями через средства API в систему 1С:Документооборот также приносит экономические и временные выгоды:

- сокращение операционных расходов. Автоматизация процессов снижает необходимость в ручной работе и уменьшает операционные расходы;

- сокращение времени на обработку доверенностей. Быстрый доступ и автоматизированные процессы позволяют сократить время, затрачиваемое на обработку доверенностей и управление ими;

- улучшение клиентского опыта. Более быстрое и точное управление доверенностями способствует улучшению клиентского опыта и довольства клиентов;

- повышение конкурентоспособности. Эффективное управление доверенностями и бизнес-процессами делает организацию более конкурентоспособной на рынке.

ВЫЗОВЫ И ПОТЕНЦИАЛЬНЫЕ РИСКИ

Кроме очевидных преимуществ интеграция системы работы с машиночитаемыми доверенностями через средства API в систему 1С:Документооборот также может столкнуться с проблемами совместимости, безопасности и другими потенциальными рисками и проблемами:

1. Проблемы совместимости и стандартизации:

- совместимость API. API, разработанные для управления доверенностями, могут быть несовместимыми с текущей инфраструктурой и версией системы 1С, что может потребовать дополнительной работы по адаптации и совместимости;

- стандартизация. Отсутствие единого стандарта для работы с машиночитаемыми доверенностями может создать вызовы в обеспечении совместимости между разными системами и API.

2. Вопросы безопасности и конфиденциальности:

- защита данных. Обеспечение безопасности данных, передаваемых через API, требует разработки механизмов шифрования, управления доступом и аутентификации;

- конфиденциальность информации. Важно уделять внимание сохранению конфиденциальности данных, особенно в случае машиночитаемых доверенностей, которые могут содержать чувствительную информацию;

- угрозы безопасности. Внедрение API может сделать систему более уязвимой к угрозам безопасности, таким как атаки на API, фишинг и другие виды атак.

3. Возможные проблемы при обновлении системы 1С:

- совместимость с обновлениями. Новые версии системы 1С могут быть несовместимы с текущей интеграцией API, что потребует пересмотра и адаптации интеграции;

- зависимость от сторонних разработчиков. Если интеграция осуществляется с помощью сторонних разработчиков, то возможны проблемы с обновлением API и поддержкой.

4. Взаимодействие с другими системами. Интеграция API в систему 1С:Документооборот может внести сложности во взаимодействие с другими системами:

- интеграция с внешними системами. Перед интеграцией следует учесть, как система будет взаимодействовать с другими системами, включая сопряженные процессы и обмен данными;

- совместимость с API других систем. Важно обеспечить совместимость и возможность интеграции с API других систем, с которыми система 1С взаимодействует.

Все эти вызовы и риски требуют тщательного планирования и управления в процессе интеграции системы работы с машиночитаемыми доверенностями через сред-

ства API в систему 1С:Документооборот. Только при условии правильного подхода и решения этих вызовов можно добиться успешной интеграции и получения выгод от новой системы.

ЗАКЛЮЧЕНИЕ

В ходе исследования и разработки системы работы с машиночитаемыми доверенностями через средства API в систему 1С:Документооборот были сделаны следующие выводы:

1. Интеграция API в систему 1С:Документооборот способствует повышению эффективности управления доверенностями и снижению рисков ошибок и нарушений безопасности.

2. Автоматизация процессов управления доверенностями позволяет сократить временные затраты и повысить гибкость системы.

3. Система работы с машиночитаемыми доверенностями через средства API принесла экономические и временные выгоды организации.

4. Однако, внедрение системы столкнулось с вызовами в виде проблем совместимости, вопросами безопасности и конфиденциальности, а также потенциальными проблемами при обновлении системы 1С и взаимодействием с другими системами.

Дальнейшее развитие системы работы с машиночитаемыми доверенностями через средства API в системе 1С:Документооборот представляет собой перспективное направление:

1. Развитие стандартизации. Стандартизация форматов и протоколов для работы с машиночитаемыми доверенностями может содействовать более простой интеграции и обмену данными между системами.

2. Усовершенствование безопасности. Дальнейшее улучшение механизмов безопасности и конфиденциальности позволит повысить уровень защиты данных и предотвращать угрозы безопасности.

3. Интеграция с более широким спектром систем. Развитие системы позволит интегрировать ее с большим чис-

лом других систем и платформ, что расширит область применения.

В целом система работы с машиночитаемыми доверенностями через средства API в системе 1С:Документооборот имеет значительный потенциал для улучшения управления доверенностями и оптимизации бизнес-процессов. Правильное развитие и управление этой системой может принести организации значительные преимущества и стать ключевым элементом ее цифровой трансформации.

ЛИТЕРАТУРА

1. Аганесян, Р. Л. 1С:Документооборот 8 КОРП, 1С:Архив. Комплект нормативных документов–2023. Методика управления документами и совместной работой / Р. Л. Аганесян, А. Е. Безбородов, А. Н. Гацков. — Москва: 1С-Паблишинг, 2023. — 505 с.

2. 1С:Документооборот КОРП. Описание. Редакция 2.1 // Информационная система 1С:ИТС. URL: <http://its.1c.ru/db/doccorp21> (дата обращения 25.10.2023).

3. Диадок: 1С 7.7. Машиночитаемые доверенности // Контур — экосистема для бизнеса. URL: <http://support.kontur.ru/diadoc-1c7x/mashinochitaemye-doverennosti> (дата обращения 25.10.2023).

4. Машиночитаемая доверенность // Информационная система 1С:ИТС. — 2023. — 16 июня. URL: <http://its.1c.ru/db/eldocs/content/109/hdoc> (дата обращения 25.10.2023).

5. Хрусталева, Е. Ю. Технологии интеграции 1С:Предприятия 8.3. — Москва: 1С-Паблишинг, 2020. — 503 с.

6. Готовим API правильно (ну или почти правильно) // Инфостарт. База знаний для 1С. — 2023. — 13 июня. URL: <http://infostart.ru/1c/articles/1879577> (дата обращения 25.10.2023).

7. API для обслуживающих организаций // Информационная система 1С:ИТС. — Обновлено 15.12.2022. URL: <http://its.1c.ru/db/freshsd#content:48727915:hdoc> (дата обращения 25.10.2023).

Implementation of the Work System with Machine-Readable Powers of Attorney Through API Tools in the 1C:Document Management

D. A. Bobrikov, N. Kalinevich

Emperor Alexander I St. Petersburg State Transport University
Saint Petersburg, Russia
daniil20001210@mail.ru, Kalinevich2015@yandex.ru

Abstract. A method for implementing a system for working with machine-readable powers of attorney using API in the 1C system is considered. The technical and organizational aspects of integration are explored, and the advantages and potential challenges of this approach are identified.

Keywords: machine-readable power of attorney, 1C:Document Management, integration, power of attorney management, API.

REFERENCES

1. Aganesyan R. L., Bezborodov A. E., Gatskov A. N. 1C:Document Management CORP, 1C:Archive. A set of regulatory documents—2023. Methodology for document management and collaboration [1C:Dokumentooborot 8 KORP, 1C:Arkhiv. Komplekt normativnykh dokumentov—2023. Metodika upravleniya dokumentami i sovmestnoy rabotoy]. Moscow, 1C Publishing, 2023, 505 p.
2. 1C:Document Management CORP. Description. 2.1 Edition [1C:Dokumentooborot KORP. Opisanie. Redaktsiya 2.1], *1C:ITS Information System [Informatsionnaya sistema 1C:ITS]*. Available at: <http://its.1c.ru/db/doccorp21> (accessed 25 Oct 2023).
3. Diadoc: 1C 7.7. Machine-Readable Power of Attorney [Diadoc: 1C 7.7. Mashinochitaemye doverennosti], *Kontur Group [Gruppa kompaniy SKB Kontur]*. Available at: <http://support.kontur.ru/diadoc-1c7x/mashinochitaemye-doverennosti> (accessed 25 Oct 2023).
4. Machine-Readable Power of Attorney [Mashinochitaemaya doverennost], *1C:ITS Information System [Informatsionnaya sistema 1C:ITS]*. Published online at June 16, 2023. Available at: <http://its.1c.ru/db/eldocs/content/109/hdoc> (accessed 25 Oct 2023).
5. Khrustaleva E. Yu. Integration technologies 1C:Enterprise 8.3 [Tekhnologii integratsii 1C:Predpriyatiya 8.3]. Moscow, 1C Publishing, 2020, 503 p.
6. Preparing the API Correctly (or Almost Correctly) [Gotovim API pravilno (nu ili pochti pravilno)], *Infostart Knowledge Base [Infostart. Baza znaniy]*. Published online at June 13, 2023. Available at: <http://infostart.ru/1c/articles/1879577> (accessed 25 Oct 2023).
7. API for Service Organizations [API dlya obsluzhivayushchikh organizatsiy], *1C:ITS Information System [Informatsionnaya sistema 1C:ITS]*. Last update at December 15, 2022. Available at: <http://its.1c.ru/db/freshsd#content:48727915:hdoc> (accessed 25 Oct 2023).

Применение технологий обработки естественного языка для голосового управления на основе открытого словаря

С. А. Михайлова, К. Г. Аникеев

Военно-космическая академия имени А. Ф. Можайского
Санкт-Петербург, Россия
mikhaylova_sa@mail.ru

Аннотация. Рассмотрена возможность применения технологий обработки естественного языка для решения задачи голосового управления путем выделения в естественной речи фраз, имеющих сходный смысл с командами управления. Рассмотрены возможности библиотек и сервисов распознавания речи. Предлагается структура системы голосового управления на базе библиотеки распознавания речи Vosk. Данная структура позволяет пополнять словарь фраз, соответствующих командам управления, на основе оценки семантической близости.

Ключевые слова: распознавание речи, голосовое управление, обработка естественного языка.

ВВЕДЕНИЕ

В настоящее время существует проблема, связанная с упрощением управления программным обеспечением, например презентацией, во время выступления докладчиков. Одним из решений данной проблемы можно считать создание системы речевого управления. Это позволит докладчикам самостоятельно регулировать темп выступления и управлять переключением мультимедийного сопровождения, не отвлекаясь на специальные устройства, такие как клавиатура, мышь или пульт дистанционного управления, без привлечения человека-ассистента.

Широкое распространение мультимедийных устройств совместно с быстрым развитием технологий искусственного интеллекта и предоставлением доступа к моделям машинного обучения привели к стадии активного внедрения технологий распознавания речи для решения различных задач во многих отраслях: в банковской сфере, промышленности, логистике, медиасфере, медицине, науке и образовании [1, 2]. Имеется множество примеров использования голосового управления различными объектами на основе умных колонок Amazon Echo или Google Home, голосовых ассистентов, таких как Siri от Apple или Google Assistant, голосовых переводчиков, например Google Translate. Большинство подобных систем построено на основе четко определенного командного языка управления объектами, или ограниченного подмножества фразеологизмов [3], или закрытого словаря команд управления.

В случае отсутствия строго формализованных команд требуется учитывать наличие множества слов или фраз, являющихся смысловыми синонимами. Особенно ярко эта проблема проявляется в случае выделения неявных команд управления в естественной речи, например в речи докладчика для управления мультимедийным сопровождением. Для решения указанной задачи необходимо соче-

тание технологий распознавания речи и методов обработки естественного языка.

Целью работы является разработка программы распознавания в естественной речи человека фраз, семантически сходных с командами управления на основе открытого словаря.

ГОЛОСОВОЕ УПРАВЛЕНИЕ

Голосовое управление — это способ управления различными устройствами или системами с помощью голосовых команд. Эта технология позволяет пользователю взаимодействовать с устройствами, не используя физические кнопки или сенсорные экраны.

В основе голосового управления лежат технологии распознавания речи. Широко распространены технологии и сервисы на их основе от крупных игроков IT-индустрии, такие как Google Speech Recognition, Apple Siri, Amazon Alexa, Nuance Dragon, Kaldi, TensorFlow.

Google Speech Recognition — технология распознавания речи от Google, используемая в таких продуктах, как Google Assistant, Google Home и Google Translate. Она позволяет пользователям голосовых устройств взаимодействовать с устройствами на естественном языке. Технология Google доступна на нескольких языках, включая русский. Развитие технологии происходит с помощью машинного обучения.

Apple Siri — технология распознавания речи от Apple, довольно популярная и интуитивно понятная. Она используется для управления устройствами Apple. Siri использует технологию обработки естественного языка для понимания запросов пользователей и предоставления ответов.

Amazon Alexa — технология распознавания речи, используемая в продуктах Amazon, таких как Amazon Echo и Alexa-enabled устройства. Она позволяет пользователям управлять устройствами и задавать вопросы на естественном языке. Пользователи могут использовать Amazon Alexa, чтобы интегрировать его со своими устройствами, установить напоминания и делать покупки.

Яндекс.Облако Контур.Распознавание — технология распознавания речи, созданная компанией Яндекс. Она может быть использована для распознавания речи на нескольких языках, включая русский, а также используется для автоматического распознавания голосов в телефонных автоответчиках. Яндекс.Облако Контур. Распознавание также используется для распознавания речевых команд в умных домах и системах безопасности.

Nuance Dragon — технология распознавания речи для профессионального использования, включая медицинские и юридические сферы. Она позволяет пользователям диктовать текст и управлять компьютером через голос. Nuance Dragon доступен на нескольких языках и имеет возможность обучения для конкретного пользователя.

Kaldi — библиотека открытого исходного кода для распознавания речи на различных языках, включая русский. Она может быть использована для создания собственной системы распознавания речи и дообучения уже существующих моделей на конкретный голос. Kaldi используется в таких областях, как биометрия, помощь водителю, медицина и образование.

TensorFlow — открытая библиотека машинного обучения, которая используется для создания и обучения нейронных сетей для распознавания речи и других задач, связанных с голосом. TensorFlow может быть использован для создания встроенной в устройства системы распознавания речи, примерами таких устройств могут быть умные наушники или колонки.

Чаще всего при взаимодействии с голосовым устройством используется включение режима восприятия команды механически (с помощью специальной кнопки) или с помощью ключевого слова при непрерывном прослушивании. Для голосовой активации режима восприятия команд используются модели, обученные на распознавание пробуждающего слова. Для распознавания самих команд используется комбинация технологий преобразования речи в текст Speech-to-Text (STT) и понимания естественного языка Natural Language Understanding (NLU).

Speech-to-Text — компонент обработки голоса, получающий от пользователя входные данные в аудиоформате и преобразующий этот фрагмент в текст.

Natural Language Understanding — компонент, определяющий намерения пользователя и основные элементы информации в команде, необходимые для ее выполнения [4]. В зависимости от намерений активируется нужный класс в модуле исполнения команд, который выполняет требование пользователя. Элементы команды, необходимые для ее выполнения, выделяются с помощью технологии выделения именованных сущностей Named Entity Recognition (NER).

В случае идентификации команды компонент управления выполняет соответствующее действие, в противном случае возможно уточнение запроса или конкретизация команды пользователем [5, 6].

Подсистема голосового управления требует предварительного создания словаря распознавания, содержащего все возможные слова, которые могут встречаться в подаваемых командах, и их транскрипции. Команды содержат указанные слова в строго определенной последовательности, другой порядок слов не распознается в качестве команды. Словарь преобразуется в форму, необходимую для работы алгоритмов распознавания, путем составления файлов грамматики, содержащих описание всех возможных команд [7].

В качестве возможной реализации управления презентацией были рассмотрены голосовые модели и сервисы Vosk, Assembly AI, PocketSphinx, Speech Recognition, Api.ai, Kaldi (табл. 1). Тестирование моделей осуществлялось на персональном компьютере с процессором на базе 11th Gen Intel(R) Core(TM) (i3-1125G4@2,00 GHz, 8 Gb RAM, Win 10).

Все модели имеют возможность создания и управления собственным словарем, в том числе путем добавления специфичных фраз и соответствующих им действий.

Таблица 1

Сравнение голосовых моделей и сервисов

Модель	Технология	Поддержка русского языка	Время распознавания голоса и вывода текста фразы в терминал	Время инициализации модели	Возможность локального подключения
Vosk	Carpathian	Vosk-Model-RU	2 с	2,3 с	Да
Assembly AI	Глубокое обучение	Обучение собственной модели	Мгновенно	Менее 1 с	Нет
PocketSphinx	HMM	Обучение собственной модели	3,2 с	43 с	Да
Speech Recognition	Google Web Speech API	Сервисы Google Cloud и IBM Watson	Мгновенно	Мгновенно	Нет
Api.ai (Dialogflow)	Google	Имеется	3,2 с	Мгновенно	Нет
Kaldi	Carpathian	Имеется	-	-	Да

Модель Vosk предназначена для работы в реальном времени [8]. Для использования Vosk необходим доступ к определенным библиотекам и зависимостям в операционной системе, таким как Python, Kaldi, и другие. Хотя модель Vosk обеспечивает высокую точность распознавания речи, она все равно может иметь ограничения в определенных сложных сценариях, таких как сильный шум или различия в диалектах. Vosk — это автономный инструмент для распознавания речи с открытым исходным кодом. Он позволяет использовать модели для 17 языков и диалектов. Для использования Vosk с новыми языками или диалектами может потребоваться предварительное обучение модели. Модели Vosk малы (50 Мб). Существуют и более точные модели, их размер достигает 2 Гб. Существует реализация библиотеки на Python, Java, Node JS, C#, C++ и др. Возможен запуск на операционных системах Windows, Linux, Android. На данный момент для русского языка доступны две модели: `vosk-model-small-ru-0.4` (50 Мб) и `vosk-model-ru-0.10` (2 Гб). Стоит отметить, что данная библиотека распознавания речи не обучена определять жаргонизмы и ненормативную лексику, но позволяет проводить дообучение моделей на пользовательской выборке.

Assembly AI предоставляет API для распознавания речи. Для создания голосовой модели необходим доступ к достаточному объему голосовых данных для обучения модели. Для облачного обучения и использования голосовой модели необходимо стабильное и быстрое интернет-соединение. Низкокачественные или зашумленные данные могут ухудшить производительность модели. При использовании облачного сервиса для создания голосовых моделей необходимо учитывать конфиденциальность голосовых данных и обеспечить их защиту.

PocketSphinx подходит для встроенных устройств и систем с ограниченными ресурсами. Для точного распознавания речи голосовая модель PocketSphinx требует четкого произношения. Звуковое окружение должно также быть достаточно тихим, чтобы предотвратить ошибки. Выполнение голосовой модели PocketSphinx требует больших вычислительных мощностей, что может быть вызвано проблемами при использовании на медленных или устаревших устройствах. Для работы голосовой модели PocketSphinx необходимо наличие дополнительного программного обеспечения для обработки звуковых и текстовых данных, например для работы с акустическими моделями и словарями. Голосовая модель PocketSphinx ограничена объемом словаря, который может быть использован при распознавании речи. Это означает, что словарь должен быть заранее создан и загружен в приложение, в него можно добавлять только ограниченное количество новых слов. Хотя голосовая модель PocketSphinx обеспечивает точное распознавание речи в большинстве случаев, она не всегда может давать точные результаты при сложных условиях, таких как шумное звуковое окружение или нечеткое произношение слов.

Speech Recognition использует веб-интерфейс Google для распознавания речи. Требует подключения к интернету для работы. Голосовая модель Speech Recognition способна обрабатывать аудиозаписи различного качества,

включая записи с низким уровнем шума и различными акцентами.

Api.ai предоставляет возможности обработки естественного языка для создания чат-ботов и интерфейсов разговора. Голосовая модель Api.ai работает через интернет, поэтому требуется постоянное подключение к сети. Для использования голосовой модели Api.ai пользователь должен иметь доступ к современным веб-браузерам, таким как Google Chrome, Mozilla Firefox, Safari или Microsoft Edge. Для доступа к Api.ai требуется создание аккаунта Google. Это позволит пользователю сохранять и управлять своими голосовыми моделями. Api.ai имеет ограничения на количество запросов, которые могут быть отправлены в течение определенного периода времени. Существует ограничение на количество символов, которое может содержать один запрос к голосовой модели. Api.ai может не обрабатывать некоторые виды аудиофайлов из-за их качества или формата.

Kaldi — мощный и гибкий инструмент для распознавания речи, часто используется в исследованиях и проектах с большими объемами данных. Kaldi требует большого объема свободного места на жестком диске для обработки аудиофайлов и обучения моделей. Kaldi требует достаточно быстрый процессор для обработки аудиофайлов и обучения моделей. Низкопроизводительный процессор может снизить скорость обработки. Kaldi требует обученной акустической модели для обработки аудиофайлов. Если модель плохо обучена, это может снизить точность распознавания речи.

Одни из главных критериев проекта — автономность и производительность. Этим критериям отвечала только одна из вышеупомянутых голосовых моделей, Vosk. Благодаря ей стала возможной обработка голоса в автономном режиме, а ее модели весили гораздо меньше, чем ее соперники.

ОПРЕДЕЛЕНИЕ СЕМАНТИЧЕСКОЙ БЛИЗОСТИ ТЕКСТОВ

Понятие семантической близости изучалось многими авторами [9, 10]. Под мерой (степенью) смысловой (семантической) близости и схожести (далее — «близость», «сходство») понимается показатель семантического сходства пары рассматриваемых слов или пары наборов слов естественного языка.

Для выделения запроса из распознанного текста производится разбиение текста на отдельные фразы для дальнейшего определения семантического сходства с командами управления. Каждую команду управления презентацией можно представить в виде метки класса семантически сходных слов/фраз, активирующих ее. Класс может пополняться на основе ключевых слов. Ключевые слова (теги) класса команд управления — набор слов естественного языка или терминов, обозначающих действие, выполняемое командой.

Семантическая близость пар слов/фраз может быть определена с помощью готовых тезаурусов, словарей или других семантических сетей [11–15], либо на основе векторного представления для текущего слова (Word Embeddings) [16]. Одними из самых известных методов определения семантической близости слов являются модели `word2vec`[17], `GloVe`[18], `StarSpace` [16] и языковая модель `BERT` [19].

Обученная модель строит векторное пространство, позволяющее определить семантическую близость понятий, векторные представления которых похожи. Таким образом, любую пару слов из словаря можно сравнить, используя, например, косинусное расстояние между векторами [20].

Насколько два текста близки по содержанию можно определить, путем вычисления косинусной меры близости двух векторов, представляющих тексты:

$$\text{similarity} = \cos(\theta) = \frac{x \times y}{|x| \times |y|},$$

где x – запрос пользователя, y – категория запроса.

Матрица сходства S представляет собой значения сходства между извлеченными векторными признаками для всех пар корпуса текстов. Таким образом, S_{ij} является величиной степени сходства текстов i, j .

УПРАВЛЕНИЕ ПРЕЗЕНТАЦИЕЙ

Проект с голосовым управлением возможно запустить множеством способов. Например, при помощи работы с периферийными устройствами, подключенными к ЭВМ, такими как мышь, клавиатура, устройствами для манипулирования работой курсора на экране ЭВМ, например пульт дистанционного управления, а также при помощи сенсорного экрана. Последняя технология упрощает работу пользователя с манипуляцией курсором в самой операционной системе, но требует для себя дополнительного дорогостоящего аппаратного дополнения.

В некоторых случаях, например при контроле аппаратной конфигурации ПЭВМ, отсутствует возможность подключения дополнительных специализированных устройств для обеспечения взаимодействия пользователя с программами. В подобных ситуациях возможно применение стандартных микрофонов, встроенных в акустические системы вычислительной техники.

Рассмотрим применение голосовых моделей на примере управления презентацией Microsoft PowerPoint.

К системе голосового управления презентацией предъявляются следующие требования:

- распознавание нестандартных формулировок команд в разговорном языке;
- динамическое пополнение словаря;
- независимость от скорости произношения, особенностей дикции, диалектов и посторонних шумов;
- локальная обработка голосовых команд;
- режим постоянного отслеживания команд.

В настоящее время разработаны системы распознавания речи, не зависящие от диктора, доступные для применения в виде сервисов и библиотек. Для реализации голосового управления презентацией в ходе изложения, например, учебного материала на естественном языке необходимо обеспечить выполнение первых двух требований. Для этого необходимо решить задачу выделения и идентификации команд управления презентацией в естественной речи человека, что соответствует построению системы управления на основе бесконечного словаря команд управления с динамическим пополнением.

С целью голосового управления презентацией сформирована система команд управления, представленная в таблице 2.

Фразы, семантически сходные с командами управления, могут быть различной длины. Например, для перехода к следующему слайду могут использоваться фразы: «далее», «следующий слайд», «на следующей схеме», «перейдем к следующему слайду», «перейдем к следующему вопросу». Также возможны нелинейные переходы: «на слайде 5», «на схеме вариантов использования», которые должны осуществлять переход по номеру или названию слайда.

С другой стороны, некоторые фразы, которые могут указывать на необходимость применения управляющей команды, при более детальном рассмотрении таковыми не являются. Например, фразы «в следующем вопросе рассмотрим...», «далее рассмотрим...», «на следующем занятии...» могут быть распознаны как команды перехода к следующему слайду или слайду с определенным названием, но не являются семантически эквивалентными ей.

Приведенные примеры можно отнести к индивидуальным особенностям построения речи, что позволяет адаптировать виртуального ассистента к речи конкретного докладчика путем формирования персонализированного словаря и/или словаря исключений. В случае выполнения неверно распознанной команды возможна ее отмена с добавлением исходной фразы, рассматриваемой как семантический эквивалент команды, в словарь исключений.

Таблица 2

Система команд

Ключевые слова/фразы	Назначение команды
Следующий слайд / Вперед	Переключение на следующий слайд
Предыдущий слайд / Назад	Переключение на предыдущий слайд
Кисть	Выбрать инструмент «кисть»
Стереть	Выбрать инструмент «ластик»
Отмена	Убирает инструмент «кисть» или «ластик», автоматическое переключение на стандартный курсор
Запуск	Запускает файл формата .pptx, путь к которому был указан при запуске программы
Открой слайд (выбери страницу) [номер_страницы]	Автоматическое переключение на слайд, порядковый номер которого был назван в команде
Открой слайд (выбери страницу, открой рисунок, открой список) [название]	Автоматическое переключение на слайд, содержащий заголовок, который был назван в команде

ПРОГРАММА УПРАВЛЕНИЯ ПРЕЗЕНТАЦИЕЙ

Программа управления презентацией построена на основе сочетания технологий распознавания речи и обработки естественного языка (рис. 1). Программа написана на языке Python 3.10 с использованием библиотеки PyQt5.

Захват аудио выполняется в режиме постоянного прослушивания на основе библиотеки PyAudio. Голосовой поток передается в модуль распознавания речи и преобразования ее в текст, построенный на базе модели Vosk. Распознанный текст поступает на вход модуля поиска команд в тексте, в котором разбивается на отдельные фразы, для последующей оценки семантической близости фраз с ко-

мандами управления. Диаграмма деятельности модуля поиска команд в тексте (рис. 2) более подробно описывает его работу.

В случае выявления команд, подлежащих выполнению, они передаются в модуль исполнения команд, который реализует управление презентаций или функциями операционной системы, такими как открытие файла.

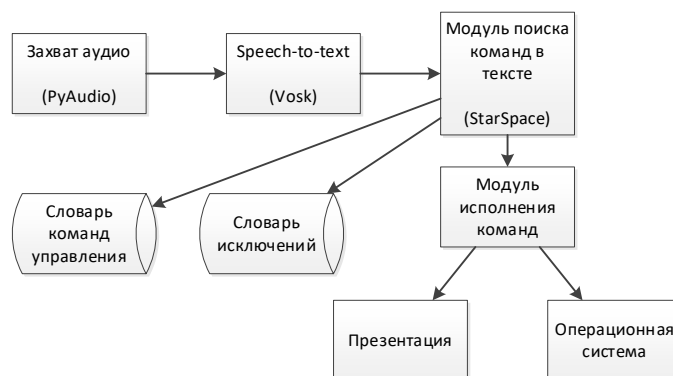


Рис. 1. Структура программы голосового управления

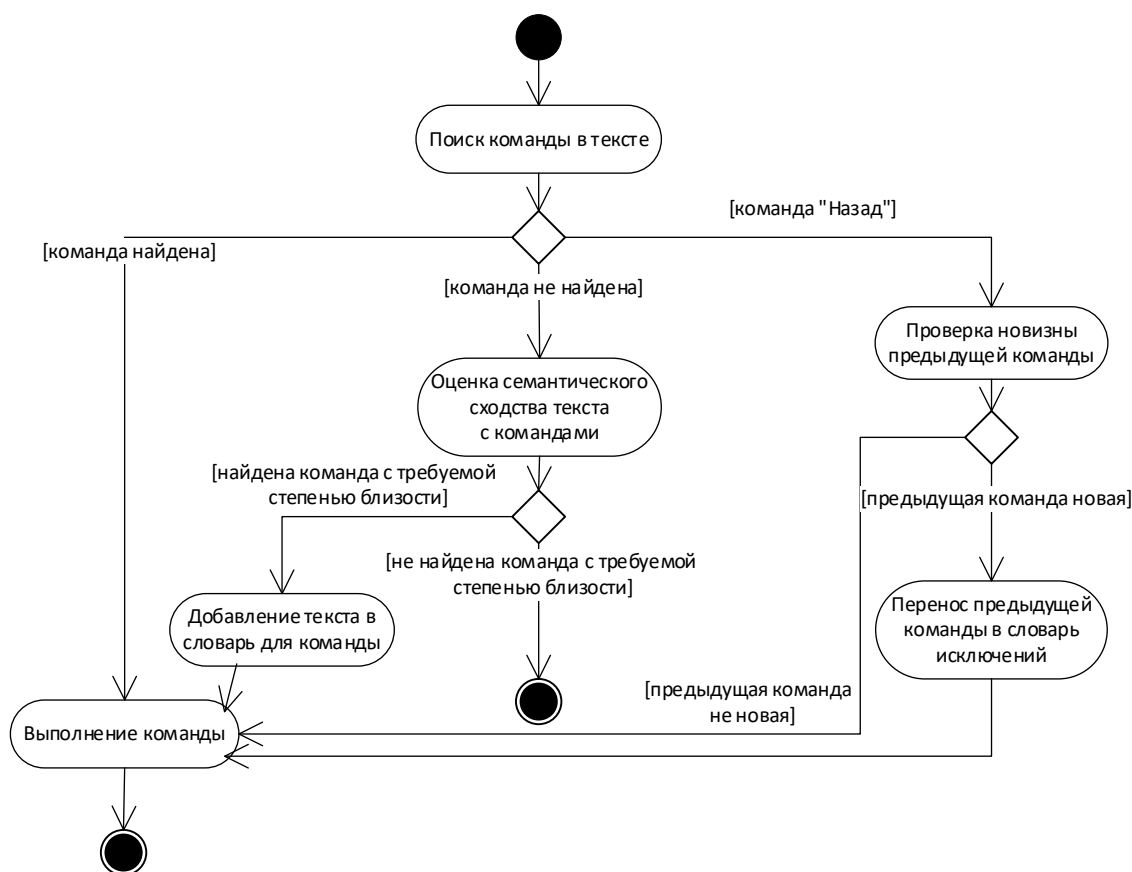


Рис. 2. Диаграмма деятельности модуля поиска команд в тексте

Работа с голосовым ассистентом проста и интуитивно понятна. Пользователь при помощи голоса подает системе команду, которая воспринимается и выполняет определенный алгоритм с презентацией (например, запускает файл формата .pptx). Далее, изучив документацию к программе, пользователь способен выполнять различные манипуляции с презентацией.

При запуске программы на экране появляется интерфейс взаимодействия с ней, содержащий:

- поле ввода информации;
- кнопку «Файловая система»;

- кнопку «Старт!»;
- кнопку «Информация».

Интерфейс программы показан на рисунке 3.

Кнопка «Файловая система» связана с голосовой командой «Открыть» и открывает проводник, в котором выбирается нужная презентация. После выбора элемента и подтверждения команды «Открыть» в поле ввода информации появляется путь к данному файлу.

Кнопка «Информация» связана с соответствующей голосовой командой, открывающей документацию к программе в формате html-файла. В ней описаны поддержива-

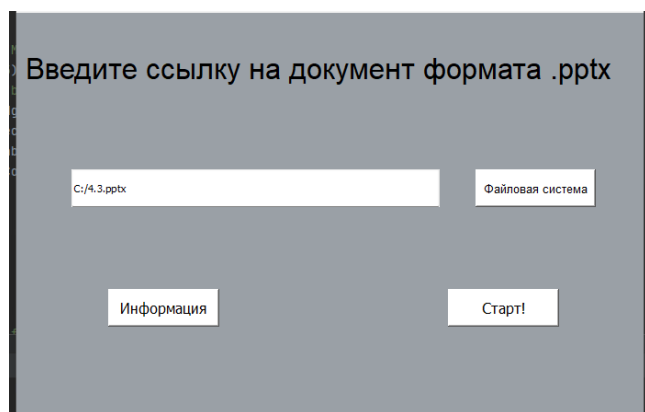


Рис. 3. Интерфейс программы голосового управления

емые в проекте голосовые команды и перечислены заранее добавленные в словарь этих команд фразы.

Кнопка «Старт!» и соответствующая голосовая команда запускает проект. Обязательное условие использования данного элемента интерфейса — наличие и правильное написание абсолютного пути к файлу презентации. Если данное условие соблюдено — программа запускается.

Сразу после запуска идет инициализация голосовой модели Vosk. Осуществляется подгрузка словаря и всех других элементов, необходимых для нормальной работы программы.

Следующий шаг — запуск приема голосового потока. Звуковая информация, попадающая в программу, преобразуется в формат файла JSON, после чего парсится и обрабатывается с целью поиска словосочетания команды или семантически сходной фразы в нем. Если команда в тексте определена, программа начинает выполнять алгоритм, соответствующий данной команде.

ЗАКЛЮЧЕНИЕ

Предлагаемое решение позволяет сократить размерность закрытого словаря команд управления за счет введения классов семантически сходных фраз, а также добавления для каждого класса словаря исключений. Подобный подход позволяет снизить количество вызовов алгоритма определения семантически эквивалентных фраз, что повышает скорость обработки речи при задействовании меньших вычислительных ресурсов.

Решение построено на базе свободно распространяемых библиотек, содержащих предобученные модели, доступных для локальной установки, что позволяет использовать данный продукт без доступа к интернет-сервисам.

Дальнейшее развитие приложения возможно за счет повышения точности распознавания команд в естественной речи, удобства использования и интеграции с другими подходами управления, например жестами, что сделает его еще более полезным и привлекательным для широкого круга пользователей.

ЛИТЕРАТУРА

1. Кипяткова, И. С. Автоматическая обработка разговорной русской речи: Монография / И. С. Кипяткова, А. Л. Ронжин, А. А. Карпов; Санкт-Петербургский ин-т информатики и автоматизации РАН. — Санкт-Петербург: ГУАП, 2013. — 314 с.

2. Обработка естественного языка, распознавание и синтез речи. Применения // Искусственный интеллект. 2019. № 2. С. 67–100.

3. Ротов, А. П. Внедрение средств распознавания речи в тренажерные комплексы управления воздушным движением / А. П. Ротов, А. Ю. Княжский // Аэрокосмическое приборостроение и эксплуатационные технологии: Сборник докладов Четвертой Международной научной конференции (Санкт-Петербург, Россия, 04–21 апреля 2023 г.). Часть 1. — Санкт-Петербург: ГУАП, 2023. — С. 72–76.

4. Petraitytė, J. *Cómo construir un asistente de voz con herramientas de código abierto como Rasa y Mozilla* // *Planeta Chatbot*. — 2019. — 30 September. URL: <http://planetachatbot.com/tutorial-como-construir-asistente-voz-con-herramientas-de-codigoabierto-rasa-y-mozilla> (дата обращения 01.12.2023).

5. On the Record. Exploring the ethical, technical and legal issues of voice assistants (CNIL White Paper Collection No. 1). — Commission Nationale de l'Informatique et des Libertés, 2020. — 84 p. URL: http://www.cnil.fr/sites/cnil/files/atoms/files/cnil_white-paper-on_the_record.pdf (дата обращения 03.12.2023).

6. Campoy, A. *Voice Assistants 101: A Look at How Conversational AI Works* / A. Campoy, S. Sassi // *Sophilabs*. — 2019. — 28 August. URL: <https://sophilabs.com/blog/voice-assistants-101> (дата обращения 28.11.2023).

7. Топорин, А. А. Подсистема голосового управления системы интеллектуального управления мобильным роботом // *Вестник науки и образования*. 2020. № 14-4 (92). С. 9–13.

8. Vosk. Распознавание речи без сети // *Alpha Cephei Speech Recognition*. URL: <http://alphacephei.com/vosk/index.ru> (дата обращения 28.11.2023).

9. Frawley, W. *Linguistic Semantics*. — New York: Routledge, 1992. — 552 p.

10. *Semantic Similarity from Natural Language and Ontology Analysis* / S. Harispe, S. Ranwez, S. Janaqi, J. Montmain. — Cham: Springer Nature, 2015. — 252 p. — (Synthesis Lectures on Human Language Technologies). DOI: 10.1007/978-3-031-02156-5.

11. RussNet: Building a Lexical Database for the Russian Language / I. Azarova, O. Mitrofanova, A. Sinopalnikova, [et al.] // *Third International Conference on Language Resources and Evaluation (LREC 2002): Proceedings of Workshop on WordNet Structures and Standardisation, and How this Affect Wordnet Applications and Evaluation (Las Palmas, Spain, 29–31 May 2002)*. — Pp. 60–64.

12. Braslavski, P. *A Spinning Wheel for YARN: User Interface for a Crowdsourced Thesaurus* / P. Braslavski, D. Ustalo, M. Mukhin // *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics (Gothenburg, Sweden, 26–30 April 2014)*. — Association for Computational Linguistics, 2014. — Pp. 101–104. DOI: 10.3115/v1/E14-2026.

13. *YARN: Spinning-in-Progress* / P. Braslavski, D. Ustalo, M. Mukhin, Y. Kiselev // *Proceedings of the Eighth Global WordNet Conference (GWC 2016), (Bucharest, Romania, 27–30 January 2016)* / V. B. Mititelu, [et al.] (eds.). — Global WordNet Association, 2016. — Pp. 58–65.

14. Лукашевич, Н. В. Тезаурусы в задачах информационного поиска. — Москва: Изд-во Московского ун-та, 2011. — 512 с.

15. Creating Russian WordNet by Conversion / N. V. Loukachevitch, G. Lashevich, A. A. Gerasimova, [et al.] // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной международной конференции «Диалог» (Москва, Россия, 01–04 июля 2016 г.). Выпуск 15 (22). — Москва: Изд-во РГГУ, 2016. — С. 405–415.

16. StarSpace: Embed All the Things! / L. Wu, A. Fisch, S. Chopra, [et al.] // Proceedings of the AAAI Conference on Artificial Intelligence. 2018. Vol. 32. Pp. 5569–5577. DOI: 10.1609/aaai.v32i1.11996.

17. Distributed Representations of Words and Phrases and Their Compositionality / T. Mikolov, I. Sutskever, K. Chen, [et al.] // Advances in Neural Information Processing Systems 26 (NIPS 2013): Proceedings of the 27th Annual Conference on Neural Information Processing Systems (Stateline, NV, USA, 05–10 December 2013) / C. J. C. Burges, [et al.] (eds.). — Curran Associates, 2013. — Pp. 3111–3119.

18. Pennington, J. Glove: Global Vectors for Word Representation / J. Pennington, R. Socher, C. D. Manning // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), (Doha, Qatar, 25–29 October 2014). — Association for Computational Linguistics, 2014. — Pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

19. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin, M.-W. Chang, K. Lee, K. Toutanova // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), (Minneapolis, MN, USA, 02–07 June 2019). Volume 1 / J. Burstein, [et al.] (eds.). — Association for Computational Linguistics, 2019. — Pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

20. Бермудес, С. Х. Г. Метод измерения семантического сходства текстовых документов // Известия Южного федерального университета. Технические науки. 2017. № 3 (188). С. 17–29. DOI: 10.23683/2311-3103-2017-3-17-29.

Application of Natural Language Processing Technologies for Voice Control Based on an Open Dictionary

S. A. Mikhaylova, K. G. Anikeev

Military Aerospace Academy
Saint Petersburg, Russia
mikhaylova_sa@mail.ru

Abstract. The possibility of using natural language processing technologies to solve the problem of voice control by identifying phrases in natural speech that have a similar meaning to control commands is considered. The capabilities of speech recognition libraries and services are considered. The structure of a voice control system based on the Vosk speech recognition library is proposed. This structure allows you to replenish the dictionary of phrases corresponding to control commands based on an assessment of semantic proximity.

Keywords: speech recognition, voice control, natural language processing.

REFERENCES

1. Kipyatkova I. S., Ronzhin A. L., Karpov A. A. Automatic processing of spoken Russian speech: Monograph [Avtomaticheskaya obrabotka razgovornoy russkoy rechi: Monografiya]. Saint Petersburg, St. Petersburg State University of Aerospace Instrumentation, 2013, 314 p.
2. Natural Language Processing, Speech Recognition and Synthesis. Applications [Obrabotka estestvennogo yazyka, raspoznavanie i sintez rechi. Primeneniya], *Artificial Intelligence [Iskusstvennyy intellekt]*, 2019, No. 2, Pp. 67–100.
3. Rotov A. P., Knyazhsky A. Yu. Implementation of Speech Recognition Tools in Air Traffic Control a Training Complexes [Vnedrenie sredstv raspoznavaniya rechi v trenazhernye komplekсы upravleniya vozdušnym dvizheniem], *Aerospace Instrumentation and Operational Technologies: Proceedings of the Fourth International Scientific Conference [Aerokosmicheskoe priborostroenie i ekspluatatsionnye tekhnologii: Sbornik dokladov Chetvertoy Mezhdunarodnoy nauchnoy konferentsii]*, Saint Petersburg, Russia, April 04–21, 2023. Part 1. Saint Petersburg, St. Petersburg State University of Aerospace Instrumentation, 2023, Pp. 72–76.
4. Petraytite J. Cómo construir un asistente de voz con herramientas de código abierto como Rasa y Mozilla, *Planeta Chatbot*. Published online at September 30, 2019. Available at: <http://planetachatbot.com/tutorial-como-construir-asistente-voz-con-herramientas-de-codigo-abierto-rasa-y-mozilla> (accessed 01 Dec 2023).
5. On the Record. Exploring the ethical, technical and legal issues of voice assistants (CNIL White Paper Collection No. 1). Commission Nationale de l'Informatique et des Libertés, 2020, 84 p. Available at: http://www.cnil.fr/sites/cnil/files/atoms/files/cnil_white-paper-on_the_record.pdf (accessed 03 Dec 2023).
6. Campoy A., Sassi S. Voice Assistants 101: A Look at How Conversational AI Works, Sophilabs. Published online at August 28, 2019. Available at: <http://sophilabs.com/blog/voice-assistants-101> (accessed 28 Nov 2023).
7. Toporin A. A. Voice Control Subsystem of Intelligent Mobile Robot Control System [Podsystema golosovogo upravleniya sistemy intellektualnogo upravleniya mobilnym robotom], *Vestnik Nauki i Obrazovaniya*, 2020, No. 14-4 (92), Pp. 9–13.
8. Vosk. Offline Speech Recognition API [Vosk. Raspoznavanie rechi bez seti], *Alpha Cephei*. Available at: <http://alphacephei.com/vosk/index> (accessed 28 Nov 2023).
9. Frawley, W. Linguistic Semantics. New York, Routledge, 1992, 552 p.
10. Harispe S., Ranwez S., Janaqi S., Montmain J. Semantic Similarity from Natural Language and Ontology Analysis. Cham, Springer Nature, 2015, 252 p. DOI: 10.1007/978-3-031-02156-5.
11. Azarova I., Mitrofanova O., Sinopalnikova A., et al. RussNet: Building a Lexical Database for the Russian Language, *Third International Conference on Language Resources and Evaluation (LREC 2002): Proceedings of Workshop on WordNet Structures and Standardisation, and How this Affect Wordnet Applications and Evaluation, Las Palmas, Spain, May 29–31, 2002*, Pp. 60–64.
12. Braslavski P., Ustalov D., Mukhin M. A Spinning Wheel for YARN: User Interface for a Crowdsourced Thesaurus, *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics, Gothenburg, Sweden, April 26–30, 2014*. Association for Computational Linguistics, 2014, Pp. 101–104. DOI: 10.3115/v1/E14-2026.
13. Braslavski P., Ustalov D., Mukhin M., Kiselev Y. YARN: Spinning-in-Progress. In: Mititelu V. B., et al. (eds.) *Proceedings of the Eighth Global WordNet Conference (GWC 2016), Bucharest, Romania, January 27–30, 2016*. Global WordNet Association, 2016, Pp. 58–65.
14. Lukashovich, N. Thesauruses in information retrieval problems [Tezaurusy v zadachakh informatsionnogo poiska], Lomonosov Moscow State University, 2011, 512 p.
15. Loukachevitch N. V., Lashevich G., Gerasimova A. A., et al. Creating Russian WordNet by Conversion, *Computational Linguistics and Intellectual Technologies: Proceedings of the 2016 Annual International Conference «Dialogue» [Kompyuternaya lingvistika i intellektualnye tekhnologii: Po materialam ezhegodnoy mezhdunarodnoy konferentsii «Dialogue»]*, Moscow, Russia, July 01–04, 2016. Issue 15 (22). Moscow, Russian State University for the Humanities, 2016, Pp. 405–415.

16. Wu, L., Fisch A., Chopra S., et al. StarSpace: Embed All the Things!, *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, Vol. 32, Pp. 5569–5577.

DOI: 10.1609/aaai.v32i1.11996.

17. Mikolov T., Sutskever I., Chen K., et al. Distributed Representations of Words and Phrases and Their Compositionality. In: *Burges C. J. C., et al. (eds.) Advances in Neural Information Processing Systems 26 (NIPS 2013): Proceedings of the 27th Annual Conference on Neural Information Processing Systems, Stateline, NV, USA, December 05–10, 2013*. Curran Associates, 2013, Pp. 3111–3119.

18. Pennington J., Socher R., Manning C. D. Glove: Global Vectors for Word Representation, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, October 25–29, 2014*. Association for Computational Linguistics, 2014, Pp. 1532–1543.

DOI: 10.3115/v1/D14-1162.

19. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *Burstein J., et al. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Minneapolis, MN, USA, June 02–07, 2019. Volume 1*. Association for Computational Linguistics, 2019, Pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

20. Bermudez S. J. G. Method for Measuring the Semantic-Similarity of Textual Documents [Metod izmereniya semanticheskogo skhodstva tekstovykh dokumentov], *Izvestiya of the Southern Federal University. Engineering Science [Izvestiya Yuzhnogo federalnogo universiteta. Tekhnicheskie nauki]*, 2017, No. 3 (188), Pp. 17–29. DOI: 10.23683/2311-3103-2017-3-17-29.

Модель и алгоритм планирования параллельной обработки информации в отказоустойчивой бортовой вычислительной системе на основе временной избыточности вычислительного процесса

С. С. Зыкова

Военно-космическая академия имени А. Ф. Можайского
Санкт-Петербург, Россия
swetlanca.zykova@yandex.ru

Аннотация. Представлена модель параллельного вычислительного процесса с временной избыточностью в отказоустойчивой бортовой вычислительной системе при внешних воздействиях. Описан вычислительный процесс, включающий периодическое сохранение текущего состояния вычислений с возможностью их восстановления.

Решена частная оптимизационная задача выбора периодичности сохранения текущего состояния вычислительного процесса в условиях возможных внешних воздействий.

Описан алгоритм планирования параллельной обработки информации в отказоустойчивой БВС с периодическим сохранением состояния вычислительного процесса и его восстановлением при деградации БВС.

Представлена имитационная модель функционирования отказоустойчивой БВС при решении целевых задач в условиях внешних воздействий, приведен анализ результатов моделирования параллельной обработки данных.

Ключевые слова: отказоустойчивая бортовая вычислительная система, параллельная обработка информации, временная избыточность.

ВВЕДЕНИЕ

Большую роль в процессах сбора и обработки целевой информации в автономных комплексах играют бортовые вычислительные системы (БВС). Анализ современных БВС показал, что сейчас их возможности по высокопроизводительной обработке информации весьма ограничены. Однако новые проекты бортовых комплексов, построенных на сетевых технологиях, позволяют наращивать вычислительную мощность бортовых компьютеров на основе реализации на них параллельных вычислений.

Перспективы развития средств вычислительной техники, интеллектуализации бортовых систем свидетельствуют о целесообразности и возможности в перспективе реализации бортовой обработки данных на основе технологий параллельных вычислений [1]. Причем при решении на борту задач могут быть одинаково эффективно реализованы как параллелизм данных, так и параллелизм задач. При этом возникает новый класс научных задач планирования параллельных вычислений в условиях внешних воздействий на БВС.

Вопросы методологии организации отказоустойчивых параллельных вычислений в вычислительных системах в

условиях их деградации являются новыми и лежат на стыке теории расписаний, теории вычислительных систем, теории параллельных вычислений, теории надежности и теории вероятностей.

Концепция работы предполагает перевод процессов функционирования БВС в угрожаемый период, предполагающий возможность частичного разрушения БВС в результате внешних воздействий, в особый режим, который заключается в превентивном планировании и реализации параллельных вычислений с периодическим сохранением их состояния на общем внешнем носителе на основе временной избыточности вычислительного процесса.

Основным показателем качества функционирования БВС в этих условиях является остаточная производительность функционирования БВС при внешних воздействиях — количество выполненных БВС целевых задач за заданное время при воздействиях, не предусмотренных условиями эксплуатации, приведших к деградации БВС (отказом части вычислительных модулей).

Штатное решение целевых задач может быть прервано (нештатно завершено) внешним воздействием. Предполагается такая организация вычислений, при которой задачи, спланированные на вышедшие из строя вычислительные модули (ВМ), выполняются на исправных ВМ либо с самого их начала, либо с момента, определяемого созданием последней «контрольной точки» вычислительного процесса прерванной задачи.

На рисунке 1 показан пример параллельного выполнения четырех задач на двух ВМ БВС. На рисунке 1, *а* представлен стандартный план вычислительного процесса, на рисунке 1, *б* — тот же план, но с созданием двух «контрольных точек» для задач 1 и 4 и по одной — для задач 2 и 3. Временные затраты на создание «контрольных точек», определяющие временную избыточность решения задач, представлены заштрихованной областью после завершения каждой задачи. В случае отказа первого ВМ в момент времени, изображенный стрелкой, выполнение задач 1 и 3 будет реализовано на исправном втором ВМ после завершения на нем задач 2 и 4.

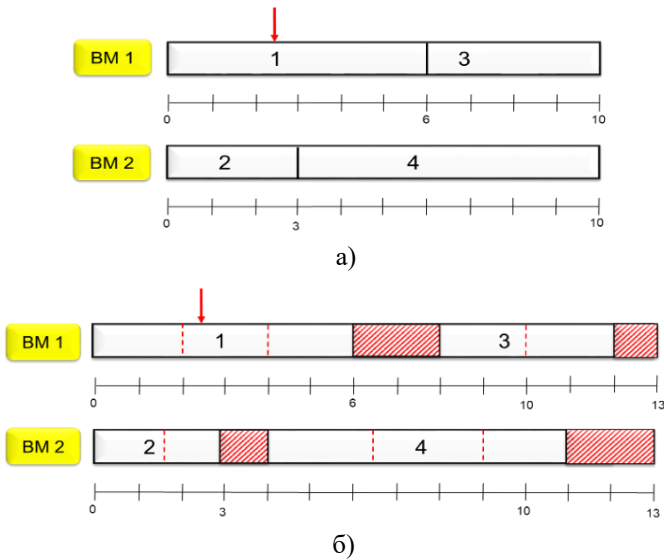


Рис. 1. Схема вычислительного процесса
 а — стандартный план вычислительного процесса;
 б — план вычислительного процесса с «контрольными точками»

При этом прерванная отказом VM задача 1 в стандартном варианте (рис. 2, а) будет выполняться с начала, а в предлагаемом варианте (рис. 2, б) — с момента создания ее первой «контрольной точки», что позволит сократить время на выполнение этой задачи на втором VM на две единицы времени.

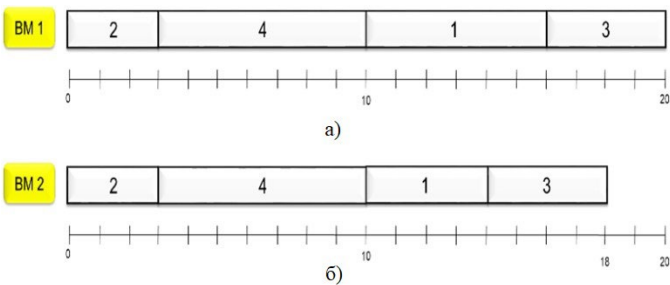


Рис. 2. Схема вычислительного процесса после отказа первого VM
 а — вычислительный процесс без временного резервирования;
 б — план вычислительного процесса с временным резервированием

Таким образом, за счет определенной временной избыточности, вводимой при решении каждой задачи, можно минимизировать общее время ее решения при отказе основного VM, на котором она выполнялась, и завершении ее выполнения на другом VM.

Преимущество такого подхода зависит от количества создаваемых при решении каждой задачи «контрольных точек» и временных затрат на этот процесс, а также момента времени отказа вычислительного модуля БВС.

Постановка задачи планирования параллельной обработки информации в отказоустойчивой БВС на основе временной избыточности вычислительного процесса

По заданным параметрам задач, параметрам БВС, параметрам возможного воздействия на БВС требуется найти

план (расписание) вычислительного процесса, обеспечивающий максимальное значение остаточной производительности функционирования БВС при внешних воздействиях.

При этом будем придерживаться трех допущений:

1. Любой VM может выполнить любую задачу.
2. Решение целевой задачи, прерванной на одном VM, может быть завершено на другом VM.
3. Все задачи независимые, т. е. результат выполнения одной задачи не влияет на решение другой.
4. Внешнее воздействие на VM БВС прекращает его функционирование на период реализации плана вычислений.

Дано:

1. Множество Z задач с параметрами $\langle \tau_i \rangle, i = 1, \dots, n$, где τ_i — время решения задачи, n — количество задач.
2. Количество m вычислительных модулей.
3. Функция $F(\xi)$ распределения моментов времени ξ реализации воздействия на БВС.

Найти:

1. Зависимость $S_{\Xi}(F(\xi))$ остаточной производительности функционирования БВС от плана $\Xi(Z, m)$ вычислительного процесса и функции $F(\xi)$ распределения моментов времени ξ реализации внешнего воздействия.
2. План $\Xi = \{t_1, \dots, t_n, k_1, \dots, k_m\}$ вычислительного процесса, содержащий «контрольные точки» ($t_i, i = 1, \dots, n$ — планируемые моменты времени завершения выполнения задач; $k_i, i = 1, \dots, m$ — количество «контрольных точек»), такой, что

$$\Xi^* = \arg \max_{\Xi \in \Xi^{\text{доп}}} s_{\Xi}(F(\xi)),$$

где s_{Ξ} — остаточная производительность БВС при реализации плана Ξ ;

$\Xi^{\text{доп}}$ — множество допустимых планов вычислительного процесса.

МОДЕЛЬ ПАРАЛЛЕЛЬНОГО ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА С ВРЕМЕННОЙ ИЗБЫТОЧНОСТЬЮ В ОТКАЗОУСТОЙЧИВОЙ БВС ПРИ ВНЕШНИХ ВОЗДЕЙСТВИЯХ

Рассмотрим общую схему БВС (рис. 3), в которой несколько вычислительных модулей по заявкам на решение задач в соответствии с некоторым расписанием вычислительного процесса решают целевые задачи. С некоторой периодичностью каждый VM записывает на общее внешнее запоминающее устройство состояние процесса выполнения текущей программы решения задачи (содержимое рабочей памяти, общих регистров, слово состояния программы).

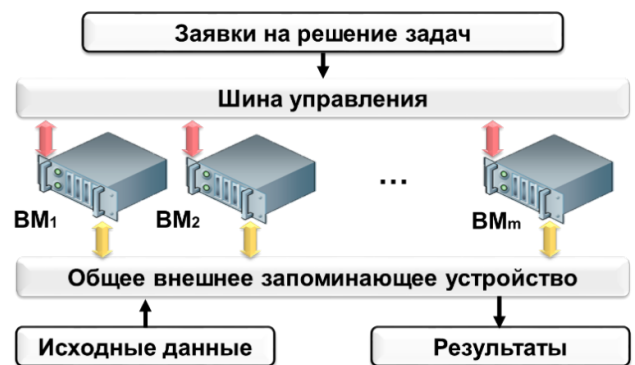


Рис. 3. Общая схема многомодульной БВС

В случае отказа некоторого ВМ спланированные для решения на нем задачи ставятся на решение после освобождения исправных ВМ. При этом те задачи, которые еще не начинали выполнение на отказавшем ВМ, будут выполнены на одном из исправных ВМ с самого их начала, а задача, выполнение которой было прервано отказом ВМ, будет довыполнена с момента создания ее последней «контрольной точки» [2].

План вычислительного процесса представлен множеством моментов времени завершения задач. При рассматриваемом подходе период решения задачи $[0, \gamma]$ состоит из k интервалов, каждый из которых, кроме последнего, включает интервал τ выполнения задачи и интервал Δ записи промежуточных результатов и состояния программы в память, то есть создания «контрольной точки» решения задачи. Предположим, что отказ одного ВМ происходит в момент времени ξ (рис. 4).

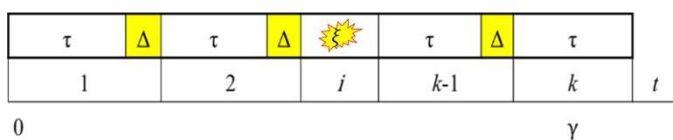


Рис. 4. Пример вычислительного процесса на одном ВМ

Таким образом, общее время решения задачи составит

$$\gamma = k\tau + (k - 1)\Delta = T + (k - 1)\Delta,$$

где T — «чистое» время, необходимое для решения задачи.

Если в момент времени ξ произошел отказ ВМ на i -м интервале решения задачи, $1 \leq i \leq k$, то в этот момент времени в памяти находится контрольная точка решения задачи, реализованного на $i - 1$ интервалах. Тогда становится возможным завершить решение задачи на другом вычислительном модуле, начиная с контрольной точки, созданной на $i - 1$ интервале.

Длительность интервала τ выполнения задачи, а следовательно, и значение времени θ выполнения прерванной задачи, зависят от величины k .

Если известна плотность распределения $f(\xi)$ момента времени отказа ВМ, то математическое ожидание $\bar{\theta}$ времени θ выполнения задачи составит

$$\bar{\theta} = \int_0^\gamma f(\xi)\theta(k, \xi)d\xi.$$

Задача состоит в оптимизации частоты создания «контрольных точек» при решении задачи с учетом вероятности отказа вычислительного модуля и возможностью ее завершения на другом ВМ.

На рисунке 5 представлено параллельное выполнение двух одинаковых по времени выполнения задач на двух ВМ, вероятность отказа каждого из которых равна q .



Рис. 5. Пример вычислительного процесса на двух ВМ

Обе задачи будут выполнены в случае, если откажет только один из двух ВМ или оба останутся работоспособными. Тогда среднее время, затраченное на выполнение задач, представлено выражением

$$\bar{\theta} = (1 - q)^2(T + (k - 1)\Delta) + q(1 - q)\left(3T + (k - 1)\Delta - \frac{T(T + k\Delta)(k - 1)}{k(T + (k - 1)\Delta)}\right). \quad (1)$$

Оптимальное количество «контрольных точек» для обеспечения восстановления вычислительного процесса после отказа ВМ находится из следующего уравнения, которое может быть решено как аналитически, так и численно.

$$\Delta k^2(k^2\Delta^2 + 2k\Delta T - 2k\Delta^2 - 2\Delta T + \Delta^2 + T^2) - qT^2(2k\Delta - \Delta + T) = 0.$$

При этом в общем случае требуется округление значения искомого корня уравнения до натурального числа.

АЛГОРИТМ ПЛАНИРОВАНИЯ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ИНФОРМАЦИИ В ОТКАЗОУСТОЙЧИВОЙ БВС С ПЕРИОДИЧЕСКИМ СОХРАНЕНИЕМ СОСТОЯНИЯ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА И ЕГО ВОССТАНОВЛЕНИЕМ ПРИ ДЕГРАДАЦИИ БВС

Вопросам планирования параллельных вычислений в научной литературе уделено достаточно много внимания. Известно множество алгоритмов как оптимального [3, 4], так и приближенного планирования [5–7]. Применительно к рассматриваемым условиям функционирования БВС целесообразно применение приближенных алгоритмов планирования.

Идея алгоритма заключается в восстановлении прерванного из-за отказа ВМ процесса решения задачи на исправном ВМ не с самого его начала, а с последнего сохраненного состояния выполнения программы («контрольной точки») (рис. 6).

При этом целесообразно оптимизировать количество «контрольных точек», так как, с одной стороны, увеличение их количества позволяет затрачивать меньше времени на завершение вычислений прерванной отказом ВМ задачи, с другой — создание каждой «контрольной точки» вносит временную избыточность в вычислительный процесс.

Суть алгоритма: при отказе ВМ прерванная решаемая на нем задача назначается для решения с момента времени последней «контрольной точки» на исправный ВМ. Остальные, еще не решенные, задачи, назначенные на отказавший ВМ, перераспределяются на исправные модули алгоритмом FFD («первый подходящий с упорядочением») [8].

Предложенный алгоритм имеет полиномиальную сложность и может быть реализован в режиме реального времени.

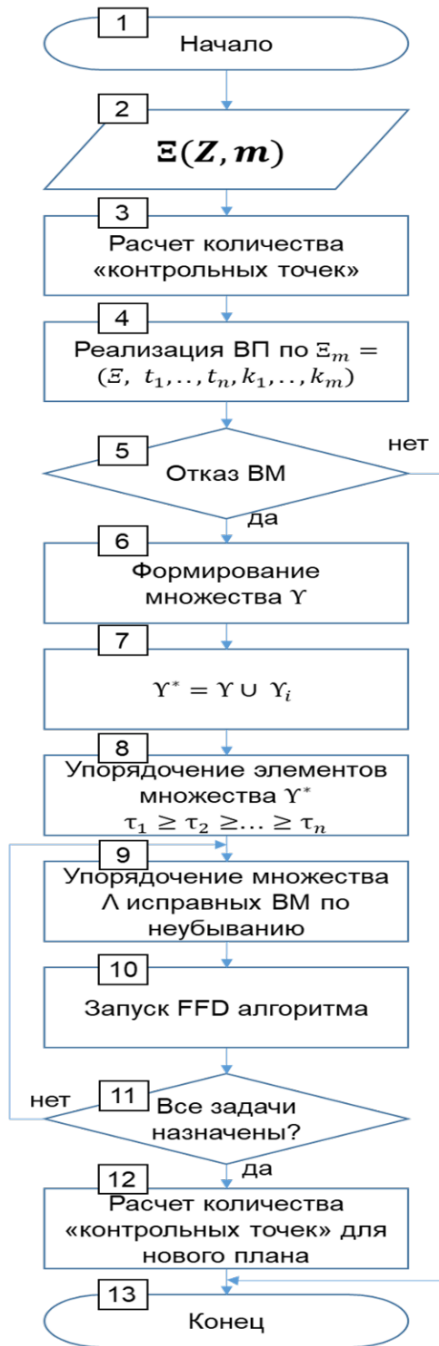


Рис. 6. Алгоритм планирования параллельной обработки информации в отказоустойчивой БВС с периодическим сохранением состояния вычислительного процесса и его восстановлением при деградации БВС

ИМИТАЦИОННАЯ МОДЕЛЬ ФУНКЦИОНИРОВАНИЯ ОТКАЗОУСТОЙЧИВОЙ БВС ПРИ РЕШЕНИИ ЦЕЛЕВЫХ ЗАДАЧ В УСЛОВИЯХ ВНЕШНИХ ВОЗДЕЙСТВИЙ

Для оценивания эффективности применения отказоустойчивой БВС с временной избыточностью реализации параллельных вычислений была построена имитационная модель процесса функционирования этой БВС.

Имитационное моделирование заключается в следующем:

1. Генерируется массив значений времени выполнения каждой задачи.

2. Строится план параллельного вычислительного процесса для заданного количества ВМ БВС.

3. Рассчитывается требуемое количество «контрольных точек» для вычислительного процесса на каждом ВМ.

4. Планируемое время вычислений на каждом ВМ увеличивается на суммарную величину значений интервалов создания «контрольных точек».

5. Случайным образом формируется номер отказавшего ВМ и момент времени его отказа.

6. Задача, выполняющаяся на отказавшем ВМ, считается прерванной в момент отказа ВМ, и ее дообработка проводится на одном из исправных ВМ с момента создания последней «контрольной точки».

7. Задачи, спланированные к выполнению на отказавшем ВМ после момента времени его отказа, переназначаются для обработки на исправные ВМ.

8. Оценивается общее время w вычислительного процесса как максимум из моментов времени завершения вычислений на каждом ВМ БВС.

9. Пункты 1–8 выполняются заданное количество раз.

10. По формуле (1) оценивается среднее время $\bar{\theta}$ вычислительного процесса.

На основе разработанных модели и алгоритма проведено имитационное моделирование, получены и исследованы зависимости, показанные на рисунках 7–9.

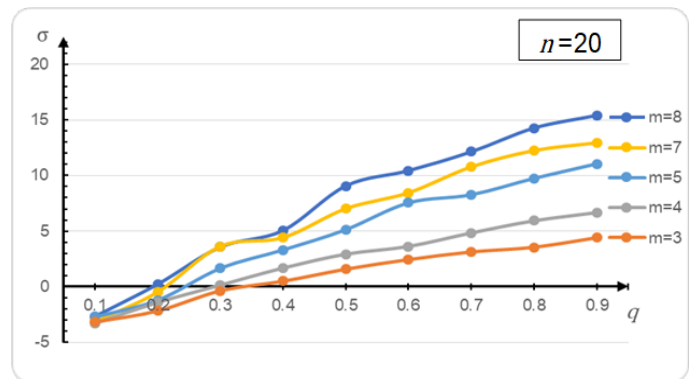


Рис. 7. Зависимость относительного выигрыша от реализации отказоустойчивого вычислительного процесса с временной избыточностью от вероятности внешнего воздействия на БВС

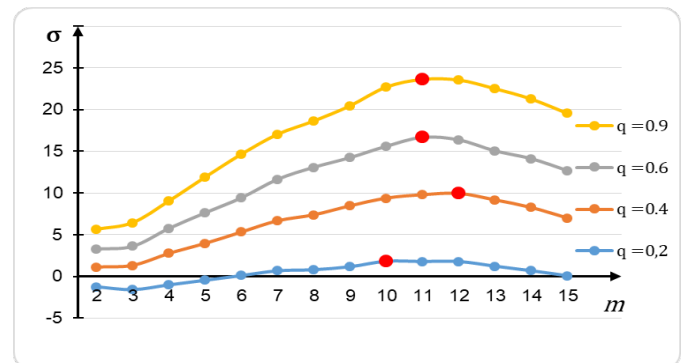


Рис. 8. Зависимость относительного выигрыша от реализации отказоустойчивого вычислительного процесса с временной избыточностью от количества ВМ в БВС

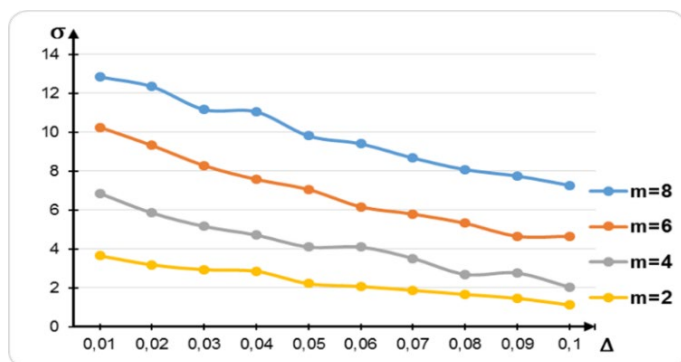


Рис. 9. Зависимость относительного выигрыша от реализации отказоустойчивого вычислительного процесса с временной избыточностью от затрат времени на создание «контрольной точки»

Представленные зависимости демонстрируют преимущество предлагаемого способа введения временной избыточности (при возрастании угрозы деградации БВС), которое увеличивается:

- с увеличением (до некоторого значения) количества ВМ;
- с увеличением вероятности ДВ на БВС;
- с сокращением временных затрат на создание «контрольных точек».

ЗАКЛЮЧЕНИЕ

Разработанная модель, в отличие от известных, учитывает временную избыточность вычислительного процесса и позволяет оценить значение показателя остаточной производительности БВС в условиях возможной ее деградации.

Представленный алгоритм предполагает проактивное сохранение состояния вычислительного процесса с возможностью его восстановления при отказах вычислительных модулей, что обеспечивает снижение потерь производительности (повышение значения остаточной производительности) при деградации вычислительной системы.

Использование предлагаемого подхода к планированию параллельных вычислений в условиях внешних воздействий позволяет снизить потери остаточной производительности БВС на величину до 15-25 %.

ЛИТЕРАТУРА

1. Кустов, В. Н. Основы теории ограниченного структурного параллелизма. — Санкт-Петербург: Министерство обороны РФ, 1992. — 246 с.
2. Метод отказоустойчивой параллельной обработки информации в бортовых вычислительных системах летательных аппаратов на основе временной избыточности вычислительного процесса / А. Г. Басыров, С. С. Зыкова, И. Н. Кошель, В. В. Кузнецов // Авиакосмическое приборостроение. 2023. № 6. С. 33–39. DOI: 10.25791/aviakosmos.6.2023.1345.
3. Crainic, T. G. Parallel Branch-and-Bound Algorithms / T. G. Crainic, B. Le Cun, C. Roucairol // Parallel Combinatorial Optimization / E.-G. Talbi (ed.). — Hoboken (NJ): Wiley-Interscience, 2006. — Pp. 1–28. — (Wiley Series on Parallel and Distributed Computing). DOI: 10.1002/9780470053928.ch1.
4. Lenstra, J. K. Sequencing by enumerative methods. — Amsterdam: Mathematisch Centrum, 1977. — 208 p. — (Mathematical Centre Tracts; Vol. 69).
5. Černý, V. Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm // Journal of Optimization Theory and Applications. 1985. Vol. 45, Is. 1. Pp. 41–51. DOI: 10.1007/BF00940812.
6. Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs. Third, Revised and Extended Edition. — Heidelberg: Springer-Verlag, 1998. — 407 p.
7. Sandnes, F. E. A Hybrid Genetic Algorithm Applied to Automatic Parallel Controller Code Generation / F. E. Sandnes, G. M. Megson // Proceedings of the Eighth Euromicro Workshop on Real-Time Systems (L'Aquila, Italy, 12–14 June 1996). — Institute of Electrical and Electronics Engineers, 1996. — Pp. 70–75. DOI: 10.1109/EMWRTS.1996.557799.
8. Adan, J. M. Meeting Hard-Real-Time Constraints Using a Client-Server Model of Interaction / J. M. Adan, M. F. Magalhaes, K. Ramamritham // Proceedings of the Seventh Euromicro Workshop on Real-Time Systems (Odense, Denmark, 14–16 June 1995). — Institute of Electrical and Electronics Engineers, 1995. — Pp. 286–293. DOI: 10.1109/EMWRTS.1995.514323.

A Model and Algorithm for Planning Parallel Information Processing in a Fault-Tolerant On-Board Computing System Based on the Time Redundancy of the Computing Process

S. S. Zykova

Military Aerospace Academy
Saint Petersburg, Russia
swetlanca.zykova@yandex.ru

Abstract. A model of a parallel computing process with temporary redundancy in a fault-tolerant on-board computing system under external influences is presented. A computational process is described, including periodic preservation of the current state of calculations with the possibility of their restoration.

A particular optimization problem of choosing the periodicity of maintaining the current state of the computing process under possible external influences is solved.

An algorithm is described for planning parallel processing of information in a fault-tolerant UAV with periodic preservation of the state of the computing process and its restoration during degradation of the UAV.

A simulation model of the functioning of a fault-tolerant UAV is presented when solving target tasks under conditions of external influences, an analysis of the results of modeling parallel data processing is given.

Keywords: fault-tolerant onboard computing system, parallel processing of information, temporary redundancy.

REFERENCES

1. Kustov V. N. Fundamentals of the theory of bounded structural parallelism [Osnovy teorii ogranichennogo strukturnogo parallelizma]. Saint Petersburg, Ministry of Defense of the Russian Federation, 1992, 246 p.
2. Basyrov A. G., Zykova S. S., Koshel I. N., Kuznecov V. V. The Method of Fault-Tolerant Parallel Processing of Information in On-Board Computing Systems of Aircraft Based on the Temporary Redundancy of the Computing Process [Metod otkazoustoychivoy paralelnoy obrabotki informatsii v bortovykh vychislitelnykh sistemakh letatelnykh apparatov na osnove vremennoy izbytochnosti vychislitelnogo protsesssa], *Aerospace Instrumentation [Aviakosmicheskoe priborostroenie]*, 2023, No. 6, Pp. 33–39. DOI: 10.25791/aviakosmos.6.2023.1345.

3. Crainic T. G., Le Cun B., Roucairol C. Parallel Branch-and-Bound Algorithms. In: *Talbi G. (ed.) Parallel Combinatorial Optimization*. Hoboken (NJ), Wiley-Interscience, 2006, Pp. 1–28. DOI: 10.1002/9780470053928.ch1.

4. Lenstra J. K. Sequencing by enumerative methods. Amsterdam, Mathematisch Centrum, 1977, 208 p.

5. Černý V. Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm, *Journal of Optimization Theory and Applications*, 1985, Vol. 45, Is. 1, Pp. 41–51. DOI: 10.1007/BF00940812.

6. Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs. Third, Revised and Extended Edition. Heidelberg, Springer-Verlag, 1998, 407 p.

7. Sandnes F. E., Megson G. M. A Hybrid Genetic Algorithm Applied to Automatic Parallel Controller Code Generation, *Proceedings of the Eighth Euromicro Workshop on Real-Time Systems, L'Aquila, Italy, June 12–14, 1996*. Institute of Electrical and Electronics Engineers, 1996, Pp. 70–75. DOI: 10.1109/EMWRTS.1996.557799.

8. Adan J. M., Magalhaes M. F., Ramamritham K. Meeting Hard-Real-Time Constraints Using a Client-Server Model of Interaction, *Proceedings of the Seventh Euromicro Workshop on Real-Time Systems, Odense, Denmark, June 14–16, 1995*. Institute of Electrical and Electronics Engineers, 1995, Pp. 286–293. DOI: 10.1109/EMWRTS.1995.514323.

Разработка и практическое применение модели интеграции оператора импликации и перегруженного контейнера «Вектор» на языке программирования C++

А. А. Нельин, к.и.н. А. В. Забродин

Петербургский государственный университет путей сообщения Императора Александра I
Санкт-Петербург, Россия
puppi2016@mail.ru, teach-case@yandex.ru

Аннотация. Рассматриваются теоретические и практические аспекты, необходимые для реализации новой интеграции из ключевых компонентов — оператора импликации и шаблонного контейнера «Вектор». Эти компоненты будут задействованы при разработке игры на графическом движке OpenGL с использованием дополнительной библиотеки GLUT и самостоятельно написанных шейдеров на C-подобном языке GLSL. Данная реализация может послужить источником вдохновения для программистов, побуждая их к активному развитию будущих идей. Практические разделы статьи включают тестовые примеры, демонстрирующие правильную работу моделей интеграции.

Ключевые слова: язык программирования C++, импликация, перегрузка операторов, вектор, выделение памяти, классы, интеграция, механика, OpenGL, шейдеры.

ВВЕДЕНИЕ

C++ является одним из самых популярных языков программирования среди разработчиков. Этот язык унаследовал множество особенностей языка C [1]. Одним из ключевых аспектов в C++ является перегрузка операторов (+, -, * и т. д.), что упрощает взаимодействие с объектами, созданными пользовательскими классами. Хотя создание собственного оператора в C++ недоступно обычным пользователям в явном виде, перегрузка операторов может рассматриваться как неявная реализация новых операторов с целью облегчить взаимодействие с объектами.

Язык C++ предоставляет возможность не только перегружать операторы, но и использовать шаблоны для создания разнообразных контейнеров. Одним из самых устоявшихся и широко используемых контейнеров является вектор, представляющий собой динамический массив, гибко изменяющий свой размер при добавлении или удалении элементов. В отличие от обычных массивов, где программисту приходится управлять выделенной памятью вручную, используя оператор delete, в векторе этот процесс автоматизирован благодаря деструкторам контейнеров библиотеки STL. Это значительно упрощает работу программиста и обеспечивает безопасное управление памятью в приложениях на C++.

В языке программирования C++ представлены все необходимые средства для воплощения в жизнь любых идей. Путем сочетания перегрузки операторов и использования

векторов в качестве отправной точки можно создать структуры данных с улучшенной производительностью и расширенной функциональностью.

В статье предлагается провести исследование с целью модернизации широко известной структуры данных — вектора. В рамках этого исследования будет рассмотрено создание нового оператора путем перегрузки нескольких других пользовательских операторов в правильном порядке. Далее, используя эти компоненты, предполагается разработать основные элементы игровой механики, которые будут реализованы на графическом движке OpenGL.

Основная цель статьи заключается в проверке возможности реализации предложенных идей в контексте языка программирования C++. Помимо этого, значительное внимание уделяется выяснению практических выгод, которые могут возникнуть в результате этих творческих усилий.

НОВЫЙ ПОЛЬЗОВАТЕЛЬСКИЙ ОПЕРАТОР

Количество пользовательских операторов в языке программирования C++ на данный момент насчитывает около трех десятков. При этом каждый обладает своим уровнем и группой приоритетности, символьным и словесным описанием, а также определенной группировкой значений при использовании. Во всех операторах группировка осуществляется слева направо [2, 3].

Основные операторы в языке C++:

- инкремент/декремент;
- функциональный (для определения функции);
- индексирование;
- логические (НЕ, И, ИЛИ, эквивалентность (равенство), импликация (следование)).

Приоритет операций в алгебре логики такой же, как и в математике: вначале выполняются операции в скобках, далее идет инвертирование операндов. Следом по приоритетности идет конъюнкция, далее дизъюнкция. В последнюю очередь идет импликация (логическое следование) и самой последней — эквивалентность. Все логические операторы, упомянутые выше, реализованы на языке C++ как пользовательские, за исключением импликации. На ней как раз стоит остановиться.

Основные свойства импликации:

- 1) x достаточное условие для y ;
- 2) y необходимое условие для x ;
- 3) результат операции ложный, если из 1 следует 0;

4) результат операции — истина, если выражение $x \leq y$;

5) эквивалентна выражению $(!x \vee y)$.

В языке программирования C++ операция следования выполняется как выражение в пункте 5. На первый взгляд, оно может показаться компактным. Однако что произойдет, если эту операцию нужно будет повторить десятки, сотни раз? В таком случае выражение станет чрезмерно громоздким, что ухудшит читабельность программного кода не только для людей, читающих его, но и для тех, кто его написал.

Создание нового оператора в языке программирования C++ представляет собой непростую задачу, так как напрямую этого сделать нельзя. Существуют два альтернативных способа для его реализации: через обыкновенную функцию и через перегрузку других операторов. Первый вариант довольно прост, но имеет недостаток в громоздкости при вызове функции каждый раз. Это может затруднить восприятие кода для читателя. Второй вариант предполагает создание оператора через перегрузку нескольких пользовательских операторов. Главное в этом случае — правильно определить «шаблон перегрузки»: параметры, которые он принимает, и возвращаемое значение, если таковое имеется. Возвращаемым значением должна быть ссылка на объект. А что будет происходить в теле оператора — решать уже нам: можно спокойно перегрузить оператор инкремента, при этом будет выполняться операция умножения.

Этапы реализации оператора следования:

1. Объявить шаблонную структуру для левой части оператора:

```
template<class T, class new_operator>
struct Left_side {
    T left;
};
```

В шаблоне указывается два абстрактных типа данных. Внутри структуры будет храниться единственное поле left. Внесем название оператора в конструкцию перечисления enum.

2. Перегрузить оператор «меньше» для левой части (до знака <) оператора следования:

```
template<class T>
Left_side<T, decltype(sledovanie)> operator<(const T& levaya,
decltype(sledovanie))>{
    return {levaya};
}
```

Возвращаемым значением будет ранее созданная структура, в которую будет передаваться значение параметра levaya.

3. Перегрузить оператор «больше» — правую часть оператора импликации:

```
template<class T1, class T2>
bool operator>(Left_side<T1, decltype(sledovanie)> levaya, T2
right_part){
    T1& left_part = levaya.left;
    return (!left_part || right_part); // реализация оператора.
}
```

Для обеспечения корректной работы оператора следования необходимо применить перегруженные операторы < и >. Используем директиву define для создания макроса, который сокращает объем оператора до одного символа. Таким образом, оператор импликации готов к использованию.

ШАБЛОННЫЙ КОНТЕЙНЕР «ВЕКТОР»

Контейнер является динамическим массивом, но, в отличие от статического, в него можно легко добавлять новые значения благодаря автоматическому увеличению пространства. Наглядным примером контейнера является всеми известная шаблонная структура — вектор.

Исследование [4] показывает, что шаблонные функции и классы устраняют необходимость в перегрузке функций для различных типов данных. Это достигается благодаря уникальному типу T в шаблонах, который можно заменить на любой тип данных языка C++ [5].

В объектно-ориентированном программировании рекомендуется называть функции, определенные внутри классов, методами, а данные — полями. Важно отметить разницу между классами и объектами. Документация [6] подробно описывает объекты как экземпляры класса, которые могут быть инициализированы как вручную путем прямого присвоения значений их полям, так и с использованием методов класса.

Основные методы шаблонного контейнера включают в себя:

- clear: очищает содержимое контейнера от всех элементов;
- push_back(новый_элемент): добавляет новый элемент в конец контейнера;
- pop_back(): удаляет последний элемент из контейнера;
- size(): возвращает количество элементов в контейнере, позволяя определить его размер;
- capacity(): возвращает текущую емкость контейнера, то есть количество элементов, которые могут быть сохранены в векторе без необходимости увеличения выделенной памяти.

Емкость должна быть больше или равна фактическому количеству элементов в векторе. Когда вектор заполняется, его емкость автоматически увеличивается для дополнительного хранения данных.

Для доступа к элементу вектора по индексу можно воспользоваться не только квадратными скобками, но и методом at, который позволяет найти элемент по указанному порядковому номеру.

Важность конструкторов и деструкторов в языке программирования C++ заключается в обеспечении корректного и безопасного управления ресурсами при создании и удалении объектов. Правильное использование этих элементов структуры классов является ключевым аспектом для обеспечения надежности и эффективности программного кода. Пример реализации конструктора и деструктора:

```
template <class T>
new_vector<T>::new_vector() {
    massive = new T[1];
    volume = 1;
    elements = 0;
}
template <class T>
new_vector<T>::~~new_vector() {
    delete[] massive;
}
```


Конструктор по умолчанию выделяет память, устанавливает переменную объема в 1 и количество элементов в 0. Деструктор освобождает динамическую память, выделенную для массива вектора в процессе работы программы. Важно отметить, что `new_vector<T>` представляет собой пространство имен. Методы, используемые в этом пространстве имен (в данном случае, в классе), должны начинаться с указания имени пространства и разделяться символом «:».

Использование пространства имен (`using namespace`) является способом избежать конфликтов именованных, позволяя локализовать идентификаторы. Как указано в [7], использование пространства имен является эффективным способом предотвратить конфликты идентификаторов.

Обращение к пространству имен `ns-name::member-name` [8], представляет собой важный пример, используемый в данной работе.

Реализация основных методов работы с вектором включает в себя операции добавления, удаления, доступа к элементам и изменения размера. В рамках данной задачи предполагается создание функций, позволяющих добавлять элементы в вектор, удалять их, получать доступ к определенному элементу по индексу, а также изменять размер вектора в соответствии с требованиями приложения.

Для успешной реализации шаблонного контейнера нам предстоит выполнить следующие действия:

1. Реализовать метод добавления элемента в вектор. Чтобы не выйти за границы области памяти, выделенной для вектора, нужно каждый раз проверять, не заполнена ли она целиком. Если окажется, что места нет, предстоит выделить для массива памяти для хранения данных в 2 раза больше старого, то есть расширить контейнер вдвое. Но почему именно 2 раза?

Процесс расширения массива включает в себя выделение новой памяти с помощью оператора `new` (`new тип[размер]`) [9]. Этот процесс может быть довольно ресурсоемким, если выполнять его при каждой операции добавления новых элементов.

Из-за значительной нагрузки операция расширения динамического массива будет замедлять выполнение программы на порядки. Чтобы избежать этого и не занимать излишне много памяти, оптимальным решением является увеличение выделенного пространства в два раза при каждом последующем заполнении, по сравнению с предыдущим объемом.

Выделение памяти под вектор нового размера:

```
if (elements == volume) {
    T* new_massive = new T[2 * volume]; // процесс пересоздания массива
    for (int i = 0; i < volume; i++)
        new_massive[i] = massive[i]; // копирование по циклу значений в новый массив
    delete[] massive; // очистка памяти от старого массива.
    volume *= 2;
    massive = new_massive;
}
```

После выделения памяти под массив происходит копирование элементов из старого массива в новый. Далее память очищается, объем хранилища увеличивается в 2 раза,

а старому указателю присваивается новый на последний созданный массив. На место после крайнего элемента происходит добавление нового значения.

2. Реализовать метод, позволяющий получить конкретный элемент вектора на основе заданного индекса. Этот функционал необходим для обеспечения удобного доступа к элементам контейнера и последующей их обработки.

Для того чтобы написать метод получения элемента по индексу, нужно передавать параметр целочисленного индекса. Единственное условие, которое нужно проверять — выход индекса за границы массива: если индекс вышел за границы массива, ничего не возвращается.

3. Реализовать метод удаления последнего элемента из массива, обеспечивая гибкость управления содержимым. Этот метод необходим для операций, требующих динамического изменения размера массива.

Этот метод уменьшает количество элементов в векторе на один, не принимая на вход никаких параметров. Удаленный элемент будет заменен при добавлении нового элемента в будущем. Дополнительно, в классе реализован деструктор, который освободит память, выделенную для удаленного элемента. Удаленный элемент не отображается при выводе содержимого вектора на экран, так как его индекс превышает максимально допустимый. Метод возвращает текущий размер вектора, подсчитанный после удаления элемента.

Далее перегружаем операторы.

4. Перегрузить оператор равенства. Реализация оператора проверки на равенство векторов:

```
template <class T>
bool new_vector<T>::operator == (const new_vector<T>& new_v) {
    if (this->elements != new_v.elements)
        return false;
    for (size_t i = 0; i < this->elements; i++) {
        if (this->massive[i] != new_v.massive[i])
            return false;
    }
    return true;
}
```

В случае оператора равенства для векторов они могут быть неравными, если они различаются по размеру или хотя бы один из их элементов отличается при сравнении. Если хотя бы одно из этих условий не выполняется, оператор возвращает `false`.

5. Перегрузить оператор неравенства.

Операторы — это фактически функции, которые могут быть перегружены, поэтому они принимают определенные параметры. В данном случае можно вызвать ранее определенный оператор равенства рекурсивно и инвертировать его результат.

6. Перегрузить оператор индексации (`[]`), который позволяет обращаться к элементам контейнера (например, массива или вектора) по их индексам, что делает работу с данными структурами более удобной и эффективной. Перегрузка этого оператора позволяет определить специфическое поведение при обращении к элементам контейнера:

```
template <class T>
T& new_vector<T>::operator[](int index) {
    assert(index >= 0 && index < elements);
    return massive[index];
}
```

Сначала индекс проверяется с использованием функции `assert` для предотвращения ошибок, связанных с памятью. Если индекс находится в пределах массива, возвращается соответствующий элемент. В случае выхода за границы массива вызывается метод `abort()`, прекращающий выполнение программы и выводящий сообщение об ошибке.

7. Для удобства вывода содержимого вектора необходимо перегрузить оператор потокового вывода.

В стандартном векторе этот оператор не был перегружен, что создавало неудобство при выводе содержимого и требовало использования цикла `for` для этой цели в рамках рассматриваемой задачи. Реализация перегруженного оператора вывода:

```
template <class T9>
ostream& operator <<(ostream& vyvod, const new_vector<T9>&
new_v) {
    for (int i = 0; i < new_v.elements; i++)
        vyvod << new_v.massive[i] << " ";
    return vyvod;
}
```

Элементы массива выводятся с использованием цикла от 0 до последнего индекса массива, что типично для вывода содержимого обычного вектора. В конце оператор возвращает ссылку на поток для поддержки цепочки операций.

На этом реализация структуры `new_vector` с ее базовыми методами завершена.

ДВЕ НОВЫЕ ИНТЕГРАЦИИ – ЧАСТЬ ЧЕГО-ТО БОЛЬШЕГО?

Оператор импликации и перегруженный контейнер «Вектор» представляют собой мощные инструменты в мире программирования на C++. Первый используется для логических выражений, позволяя устанавливать условия и связи между данными. Второй — это гибкая структура данных, способная хранить и управлять коллекциями элементов, облегчая работу с данными и их обработку в программе. Их интеграция открывает двери для оптимизации и эффективного управления информацией в различных алгоритмах, например:

1. Упрощение программирования на C++. Контейнер «Вектор» помогает удобно хранить и управлять данными, упрощая создание новых структур данных или классов.

2. Оптимизация работы с данными. Оператор следования может использоваться для более эффективной обработки системных данных, улучшая алгоритмы работы с ними.

3. Разработка игровых механик. В C++ оператор импликации и «Вектор» подходят для создания интересных игровых механик и управления игровыми объектами.

При этом следует подчеркнуть, что оператор импликации («если *A*, то *B*») прекрасно используется в алгоритмах, обрабатывающих различные сценарии в играх. И его интеграция с контейнером «Вектор» позволяет создавать сложные игровые логики и управлять игровыми элементами.

Помимо этого, мы можем поднять планку. Вместо того, чтобы просто создавать игру на языке C++, мы можем воспользоваться мощью OpenGL и библиотеки Glut для разработки качественных графических сцен. А дополнительные возможности по созданию собственных шейдеров на языке GLSL придадут игре уникальный вид. Таким образом, C++ становится не только основой, но и ключом к созданию захватывающего игрового мира. Но перед тем, как углубиться в детали механики игры, рассмотрим основы работы с OpenGL, Glut и GLSL.

Однако стоит учитывать, что у нас еще есть реализованный контейнер «Вектор», который может хранить значения обработанных условий. Пункт 2, безусловно, подходит под практическое применение моделей интеграции, однако стоит ли останавливаться на одном алгоритме, если и он может быть только частью чего-то большего?

А что, если сделать собственную игру, которая будет реализована на базе языка C++? А что, если написать игру на движке OpenGL, взаимодействуя с одной из поддерживаемых библиотек — Glut? А что, если вдобавок ко всему этому не использовать стандартные шейдеры, заложенные программой, а реализовать их самому на C-подобном языке GLSL? Каков же будет результат? В таком случае использовать только язык программирования C++ не выйдет, однако он останется основой для написания программного кода к игре. Перед тем, как углубиться в идеи для механики игры, стоит немного поговорить про OpenGL, Glut и GLSL.

OpenGL (Open Graphics Library) — это современный кроссплатформенный API для работы с двухмерной и трехмерной графикой. Он функционирует по принципу клиент-сервер, где клиентом является пользователь, использующий движок OpenGL, а сервером выступает графическое ядро (GPU) и его драйвер [10].

Однако напрямую работать с чистым OpenGL не всегда удобно. Для более удобного и эффективного программирования следует воспользоваться дополнительными библиотеками:

1. `glut.h` (OpenGL Utility Toolkit) — библиотека для различных утилит, предназначенная для приложений на OpenGL. Она упрощает возможность создания графического окна, позволяет добавлять текст, определять функции отрисовки и даже устанавливать таймеры. Благодаря кроссплатформенной поддержке, код, написанный для одной операционной системы, будет работать и на других.

Пример использования библиотеки GLUT:

```
glutInitWindowSize(666, 666);
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowPosition(280, 80);
glutCreateWindow("");
glutDisplayFunc(displayMe);
glutReshapeFunc(reshape);
```

2. `glew.h` (OpenGL Extension Wrangler). Эта библиотека расширяет функциональность OpenGL для работы с шейдерами. Шейдеры — это специальные программы, которые выполняют графические вычисления на графическом процессоре (GPU). Они позволяют контролировать различные аспекты графики, такие как освещение, тени, отражения и преломления света.

При этом важно проверять статус функции инициализации библиотеки GLew:

```
if(glewInit() != GLEW_OK){
std::cout << "Error in glewInit\n";
return 1;
}
```

Для реализации предлагаемой модели интеграции нам понадобится GLSL (OpenGL Shading Language): GLSL — это язык программирования, который используется для написания шейдеров в OpenGL. GLSL напоминает язык программирования C и C++, что делает его относительно легким для изучения и использования программистами. Он предоставляет широкий набор функций и возможностей для создания сложных визуальных эффектов в компьютерной графике. Шейдеры пишутся на языке GLSL и компилируются на графическом процессоре в реальном времени во время выполнения программы OpenGL. В контексте OpenGL, GLSL позволяет программистам создавать сложные и красивые сцены, делая компьютерную графику более реалистичной и захватывающей.

При этом, в отличие от языка C++, в GLSL отсутствуют:

- строки;
- указатели;
- битовые поля;
- рекурсия;
- подключение каких-либо библиотек из-за отсутствия необходимости.

В GLSL есть векторные и матричные типы, которые позволяют работать с шейдерами:

- vec2, vec3, vec4 — двух-, трех- и четырехмерные векторы из компонент типа float;
- ivec2, ivec3, ivec4 — двух-, трех- и четырехмерные векторы из компонент типа int;
- uvec2, uvec3, uvec4 — двух-, трех- и четырехмерные векторы из компонент типа uint;
- bvec2, bvec3, bvec4 — двух-, трех- и четырехмерные векторы из компонент типа bool;
- mat2, mat3, mat4 — двух-, трех- и четырехмерные квадратные матрицы из компонент типа float;
- mat2x3, mat2x4, mat3x2, mat3x4, mat4x1, mat4x2, mat4x3 — матрицы из компонент типа float.

В статье рассматривается GLSL версии 3.30. Эта версия поддерживает входные и выходные типы данных, а также специальный спецификатор — uniform. Эти особенности позволяют более гибко управлять данными и параметрами, необходимыми для реализации шейдеров в контексте предлагаемой модели интеграции.

В контексте рассматриваемой версии GLSL входной тип (in) представляет собой данные, с которыми шейдер будет работать внутри программы, а выходной тип (out) возвращает полученное значение из шейдера в программу. Тип uniform задается пользователем в основной программе и передается шейдеру с помощью специальной функции glGetUniformLocation().

При этом существуют два основных способа загрузки шейдеров в программу:

- через передачу считанной строки из файла;

– через строку символов, написав код прямо в программе на C++.

Дополнительные сведения об основных типах шейдеров можно найти в [11, 12].

Примеры чтения шейдеров на языке C++:

1. Функция чтения файла в строку:

```
string txtFileRead(string filename){
string startStr, returnStr;
ifstream in;

in.open(filename);
if(in.is_open()){
getline(in, startStr);
while(in){
returnStr += startStr + "\n";
getline(in, startStr);
}
}
in.close();

return returnStr;
}
```

2. Передача кода шейдера на языке GLSL напрямую в строку символов:

```
const char* vertexShaderSource =
"#version 330 core\n"
"layout(location = 0) in vec3 position;\n"
"layout(location = 1) in vec3 Norm;\n"
"out float light;\n"
"uniform mat4 model;\n"
"uniform mat4 view;\n"
"uniform mat4 projection;\n"
"uniform vec3 lightPos;\n"
"void main(){\n"
"vec3 p = vec3(view*model* vec4(position, 1.0));\n"
"vec3 l = lightPos-p;\n"
"vec3 n = vec3(model * vec4(Norm, 1.0));\n"
"light = max(dot(n,l),0.0);\n"
"gl_Position = projection * view * model * vec4(position, 1.0);\n"
"}\n0";
```

Для успешной компиляции шейдеров в программе на языке C++, обе методики требуют использования библиотеки GLew. После этого необходимо воспользоваться рядом функций:

- loadShader() — для загрузки шейдера;
- glGetShaderiv() — для проверки ошибок компиляции;
- glGetShaderInfoLog() — для получения информации об ошибках;
- glCreateProgram() — для создания программы, к которой будут присоединены шейдеры (glAttachShader());
- glLinkProgram() — для линковки программы;
- glGetProgramiv() и glGetProgramInfoLog() — для проверки ошибок линковки;
- glUseProgram() — отвечает за отрисовку данных, обработанных шейдерами.

Как упоминалось ранее, стандартные шейдеры в OpenGL предоставляют множество удобных функций для работы с графическими объектами, избавляя пользователя от необходимости работать с матрицами и векторами. Однако при подключении собственных шейдеров пользователь сам отвечает за правила взаимодействия с объектами. Это означает, что пользователю придется работать с матрицами (двумерными массивами) вручную!

Теперь, когда мы познакомились с основами OpenGL, рассмотрим, каким образом концепция будущей игры будет построена и на каких принципах она будет основана. Оператор импликации, о котором мы упоминали ранее, работает с условиями вида «если A , то B ». В алгебре логики он представляется формулой $A \implies B$ (не A или B). Допустим, оператор следования объединяет как минимум два действия в игре, произведенные пользователем. В этом случае вектор будет хранить значения, полученные в результате этих действий. После сохраненные значения будут проверяться в нашем шаблонном контейнере, и на основе этой проверки будет приниматься решение — true (истина) или false (ложь).

Конечная цель нашей игровой механики состоит в том, чтобы оператор импликации мог объединять различные действия игрока и принимать решения на основе этих действий.

Приведем простой пример: когда пользователь выполняет действия, например, подбирает ключ и затем пытается открыть дверь, значения этих действий сохраняются в нашем векторе. Далее происходит проверка сохраненных значений в шаблонном контейнере. Если оба условия выполнены (пользователь подобрал ключ и пытается открыть дверь), оператор импликации возвращает true (истина), указывая, что дверь должна открыться. В противном случае, если хотя бы одно из условий не выполнено, оператор импликации вернет false (ложь), и дверь останется закрытой. В ситуации с целой локацией дверей значения действий игрока будут храниться в векторе до тех пор, пока он не пройдет все двери без ошибок или хотя бы раз не произойдет просчет. В таких случаях значения шаблонного контейнера обнуляются, чтобы избежать неправильной работы игровой механики.

Таким образом, наш вектор хранит значения, полученные в результате различных действий игрока, и оператор импликации использует эти значения для принятия решений в игре, что делает механику более интересной и интерактивной для игрока.

Предложенная механика может быть весьма эффективна при создании игр-головоломок различной сложности. Эти логические игры могут быть самостоятельными проектами с увлекательным сюжетом или дополнительными элементами в более крупных играх. Особенно ценным является второй вариант, где головоломки становятся ключевым компонентом для развития сюжета и принятия игровых решений. В таких случаях игроки могут взаимодействовать с головоломками, чтобы открыть сюжетные ответвления и раскрывать новые аспекты игровой вселенной. Это подчеркивает важность данной механики в обогащении геймплея и создании увлекательного игрового опыта.

Общая диаграмма структуры игрового приложения представлена на рисунке 1.

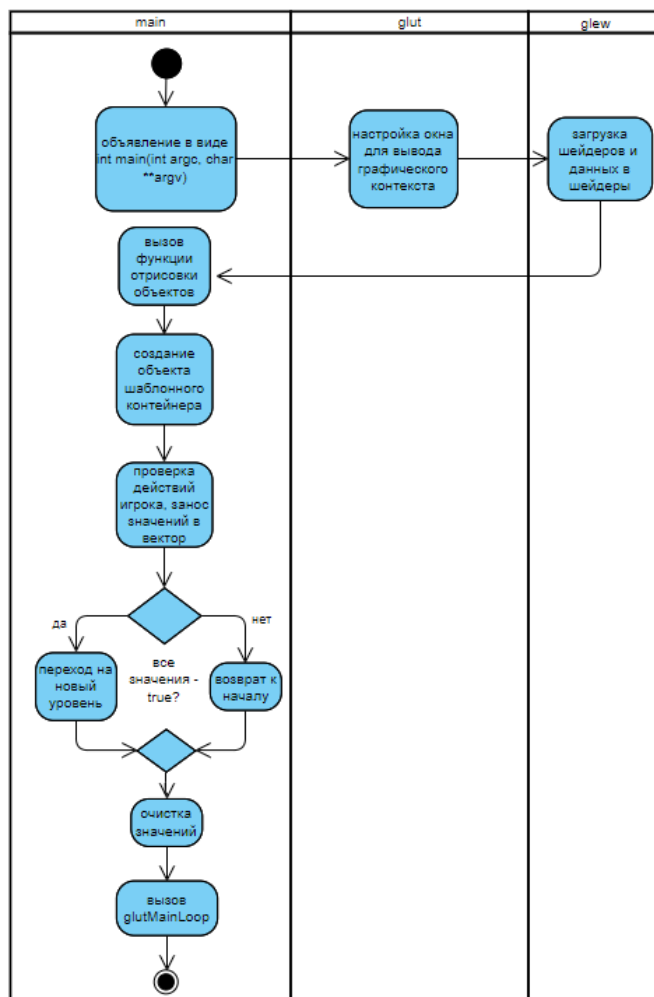


Рис. 1. Структура игрового приложения

Следует отметить, что OpenGL предоставляет возможность создавать как 2D, так и 3D приложения, что, в свою очередь, позволяет интегрировать головоломки как в двумерные, так и в трехмерные игры. Интересные головоломки могут усилить слабую механику игры, делая ее более увлекательной и уникальной. С другой стороны, сложные головоломки могут сделать простые игровые механики интересными и вызывающими. Помимо этого, можно экспериментировать с различными механиками, используя векторы и операторы импликации. Но следует помнить, что разработка таких механик может быть сложной задачей, требующей внимательного проектирования.

ЗАКЛЮЧЕНИЕ

C++ — язык с бесконечными возможностями, позволяющий создавать уникальные модификации благодаря шаблонам, классам и перегрузке операторов. В данной статье представлена инновационная связка моделей интеграции: контейнера «Вектор» и оператора импликации. Эти компоненты стали основой для разработки механики игры на движке OpenGL, используя библиотеки GLUT и GLew, а также самописные шейдеры. Полученный опыт открывает новые перспективы для будущих проектов в области компьютерных игр и графики.

Представленная в статье механика игрового опыта предоставляет новаторские возможности для индустрии

развлечений, добавляя глубину и оригинальность игровым проектам. Однако, как и с любой новой идеей, данную модификацию нужно будет адаптировать под постоянно изменяющиеся технологии. В случае успешной реализации этой концепции планируется более подробное рассмотрение создания 2D или 3D игровых приложений с использованием предложенной на рисунке 1 структуры.

Представленные концепции могут послужить источником вдохновения для дальнейшего творчества. Все, что было рассмотрено в данной статье, является всего лишь крошечной частью богатства языка программирования C++ и современного кроссплатформенного API OpenGL. Возможности и потенциал этих технологий остаются практически безграничными, ожидая новых исследований и творческих решений.

ЛИТЕРАТУРА

1. Рейзлин, В. И. Язык C++ и программирование на нем: Учебное пособие. — 3-е изд., перераб. — Томск: Изд-во Томского политехнического университета, 2021. — 208 с.
2. Литвиненко, Н. А. Технология программирования на C++. Начальный курс. — Санкт-Петербург: БХВ-Петербург, 2005. — 288 с.
3. Введение в математическую логику. Нотации // Хекслет. URL: http://ru.hexlet.io/courses/logic/lessons/notation/theory_unit (дата обращения 08.12.2023).
4. Семякин, В. С. Просто о шаблонах C++ // Хабр. — 2022. — 10 января. URL: <http://habr.com/ru/articles/599801> (дата обращения 22.11.2023).
5. Абрамян, М. Э. Введение в стандартную библиотеку шаблонов C++: описание, примеры использования, учебные задачи: Учебник. — Ростов-на-Дону; Таганрог: Изд-во Южного федерального университета, 2017. — 177 с.
6. Классы в C++ // Язык программирования C++: Учебник. URL: <http://cppstudio.com/post/439> (дата обращения 26.11.2023).
7. Шилдт, Г. C++: базовый курс. Третье издание = C++ from the Ground Up. Third Edition / Пер. с англ. и ред. Н. М. Лучко. — Москва: Издательский дом «Вильямс», 2010. — 624 с.
8. Namespaces // C++ Reference. — Обновлено 26.11.2023. URL: <http://en.cppreference.com/w/cpp/language/namespace> (дата обращения 29.11.2023).
9. Динамическое выделение памяти в C++ // Программирование. URL: <http://prog-cpp.ru/cpp-newdelete> (дата обращения 06.12.2023).
10. Васильев, С. А. OpenGL. Компьютерная графика: Учебное пособие. — Тамбов: Изд-во Тамб. гос. техн. ун-та, 2005. — 80 с.
11. Боресков, А. В. Программирование компьютерной графики. Современный OpenGL. — Москва: ДМК Пресс, 2019. — 372 с.
12. OpenGL ES 3.0. Руководство разработчика = OpenGL ES 3.0 Programming Guide. Second Edition / Д. Гинсбург, Б. Пурномо, Д. Шрейнер, А. Мунши; пер. с англ. А. В. Борескова. — Москва: ДМК Пресс, 2015. — 448 с.

Development and Practical Application of the Integration Model of the Implication Operator and the Overloaded «Vector» Container in the C++ Programming Language

A. A. Nelin, PhD A. V. Zabrodin

Emperor Alexander I St. Petersburg State Transport University
Saint Petersburg, Russia
puppi2016@mail.ru, teach-case@yandex.ru

Abstract. The article discusses the theoretical and practical aspects necessary for the implementation of a new integration of the key components — the implication operator and the template container «Vector». These components will serve in the development of the game on the OpenGL graphics engine using the additional GLUT library and independently written shaders in the GLSL language. This implementation can serve as a source of inspiration for programmers, encouraging them to actively develop future ideas. The practical sections of the article include test cases demonstrating the correct operation of integration models.

Keywords: C++ programming language, implication, operator overloading, vector, memory allocation, classes, integration, mechanics, OpenGL, shaders.

REFERENCES

1. Reyzlin V. I. The C++ language and programming on it: Study guide [Yazyk C++ i programmirovaniye na nem: Uchebnoye posobie]. Tomsk, Tomsk Polytechnic University, 2021, 208 p.
2. Litvinenko N. A. C++ programming technology. The initial course [Tekhnologiya programmirovaniya na C++. Nachalnyy kurs]. Saint Petersburg, BHV-Peterburg Publishing House, 2005, 288 p.
3. Introduction to Mathematical Logic. Notations [Vvedenie v matematicheskuyu logiku. Notatsii], *Hexlet [Khekslet]*. Available at: http://ru.hexlet.io/courses/logic/lessons/notation/theory_unit (accessed 08 Dec 2023).
4. Semenyakin V. S. Just About C++ Templates [Prosto o shablonakh C++], *Habr [Khabr]*. Published online at January 10, 2022. Available at: <http://habr.com/ru/articles/599801> (accessed 22 Nov 2023).
5. Abramyan M. E. Introduction to the standard C++ template library: Description, examples of using, training tasks: Textbook [Vvedenie v standartnyuyu biblioteku shablonov C++: opisanie, primery ispolzovaniya, uchebnye zadachi: Uchebnik]. Rostov-on-Don, Taganrog, Southern Federal University, 2017, 177 p.
6. Classes in the C++ [Klassy v C++], *C++ programming language: Textbook [Yazyk programmirovaniya C++: Uchebnik]*. Available at: <http://cppstudio.com/post/439> (accessed 26 Nov 2023).
7. Schildt H. C++ from the Ground Up. Third Edition [C++: bazovyy kurs. Tret'ye izdanie]. Moscow, Williams Publishing House, 2010, 624 p.
8. Namespaces, *C++ Reference*. Last update at November 26, 2023. Available at: <http://en.cppreference.com/w/cpp/language/namespace> (accessed 29 Nov 2023).
9. Dynamic Memory Allocation in C++ [Dinamicheskoe vydelenie pamyati v C++], *Programming [Programmirovaniye]*. Available at: <http://prog-cpp.ru/cpp-newdelete> (accessed 06 Dec 2023).
10. Vasilyev S. A. OpenGL. Computer graphics: Study guide [OpenGL. Kompyuternaya grafika: Uchebnoye posobie]. Tambov, Tambov State Technical University, 2005, 80 p.
11. Borekov A. V. Computer graphics programming. Modern OpenGL [Programmirovaniye kompyuternoy grafiki. Sovremennyy OpenGL]. Moscow, DMK Press Publishing House, 2019, 372 p.
12. Ginsburg D., Purnomo B., Shreiner D., Munshi A. OpenGL ES 3.0 Programming Guide. Second Edition [OpenGL ES 3.0. Rukovodstvo razrabotchika]. Moscow, DMK Press Publishing House, 2015, 448 p.

Разработка алгоритма сжатия данных на основе метода Хаффмана: практический аспект

Р. М. Слесарев, М. А. Добровольская, к.и.н. А. В. Забродин

Петербургский государственный университет путей сообщения Императора Александра I
Санкт-Петербург, Россия

romsy2002@gmail.com, dobro-7878@mail.ru, teach-case@yandex.ru

Аннотация. Статья посвящена разработке алгоритма сжатия данных различных типов, представленных в виде файлов во внешней памяти компьютера, на основе метода Хаффмана. Подробно рассмотрен алгоритм Хаффмана минимального префиксного кодирования. Целью работы является изучение и сравнение между собой различных алгоритмов сжатия данных, а также практическое применение алгоритма на основе метода Хаффмана для написания программы-компрессора файлов.

Ключевые слова: сжатие без потерь, сжатие с потерями, алгоритмы сжатия, кодирование, кодовое дерево.

ВВЕДЕНИЕ

С увеличением объема данных в современном мире возникает настоятельная потребность в эффективных методах хранения и передачи информации. Для оптимизации использования памяти было разработано множество алгоритмов сжатия данных. Существуют две основные категории таких алгоритмов: сжатие данных с потерями и без потерь. Первая используется для уменьшения размера фотографий и видео, в то время как вторая включает в себя методы и алгоритмы, направленные на сжатие файлов без утраты информации. Среди методов без потерь выделяются алгоритмы кодирования Хаффмана, которые являются одними из самых старых, но, несмотря на свою долгую историю, остаются востребованными и эффективными. Основная цель работы заключается в исследовании и практическом применении алгоритма на основе метода Хаффмана в контексте сжатия текстовых данных.

ОСНОВНЫЕ ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Сжатие данных представляет собой процесс преобразования данных с целью уменьшения их объема путем удаления избыточности. Главная цель сжатия данных заключается в уменьшении занимаемого ими пространства в памяти и повышении скорости их обработки [1].

Избыточность является ключевым понятием в теории сжатия информации, поскольку сжатие возможно только для данных, содержащих избыточную информацию. Данные, не содержащие избыточности (например, файлы, состоящие из случайных равновероятных байтов), невозможно сжать [2].

Методы сжатия данных представляют собой специфический случай методов кодирования информации. Код представляет собой взаимно-однозначное отображение конечного упорядоченного множества символов из определенного алфавита на другое множество символов.

С точки зрения возможности полного восстановления данных методы сжатия данных можно разделить на два основных класса [3]: сжатие без потерь (обратимое) и сжатие с потерями (необратимое).

Сжатие без потерь позволяет снизить объем данных без потери информации, то есть позволяет восстановить закодированную порцию данных без изменений.

Сжатие с потерями не позволяет путем декодирования восстановить исходные данные в полном объеме, но при этом обеспечивается максимальная степень сжатия данных [4].

Методы сжатия данных с точки зрения возможности их использования для сжатия информации различных типов (текста, изображений, звука или видео) можно разделить на два типа [5]: специальные и универсальные.

Универсальные методы представляют информацию как простую последовательность битов, поэтому могут использоваться для сжатия любой цифровой информации. Такие методы являются обратимыми, так как позволяют добиться полного восстановления сжатых данных.

Специальные методы предназначены для сжатия информации только определенного типа. Такие методы учитывают специфику восприятия изображений, звука или видео человеком. Благодаря удалению малозначимой для восприятия информации удается достичь высокой степени сжатия. Таким образом, специальные методы являются необратимыми.

Для оценки производительности алгоритмов сжатия применяют следующие величины [6]:

1. Коэффициент сжатия:

$$\text{Коэффициент сжатия} = \frac{\text{размер выходного файла}}{\text{размер входного файла}}$$

2. Величина, обратная коэффициенту сжатия, называется фактором сжатия. Если эта величина меньше 1, то идет сжатие, если больше 1 — расширение.

$$\text{Фактор сжатия} = \frac{\text{размер входного файла}}{\text{размер выходного файла}}$$

3. Качество сжатия: $100 * (1 - k)$, где k - коэффициент сжатия, отражает качество сжатия.

$$\text{Качество сжатия} = 100 * (1 - k),$$

где k — коэффициент сжатия.

ОПИСАНИЕ АЛГОРИТМА ХАФФМАНА

Простой алгоритм минимального префиксного кодирования был предложен в 1952 году Дэвидом Хаффманом (David Huffman). Он позволяет сжимать данные до их энтропии, что является оптимальным методом для сжатия без потерь. Алгоритм работает путем построения кодового дерева снизу вверх [7]:

1. Каждому символу сопоставляется дерево, состоящее из одной вершины, содержащей код символа и его частоту.
2. Составляется список всех вершин в порядке убывания их вероятностей.
3. На каждом шаге выбираются две вершины с наименьшими вероятностями, удаляются из списка и заменяются вспомогательной вершиной, представляющей эти два символа. Вспомогательной вершине приписывается вероятность, равная сумме вероятностей выбранных на этом шаге символов.
4. Когда список сокращается до одного вспомогательного символа, представляющего весь алфавит, дерево объявляется построенным. Алгоритм завершается.

Построенное дерево кодов Хаффмана показано на рисунке 1.

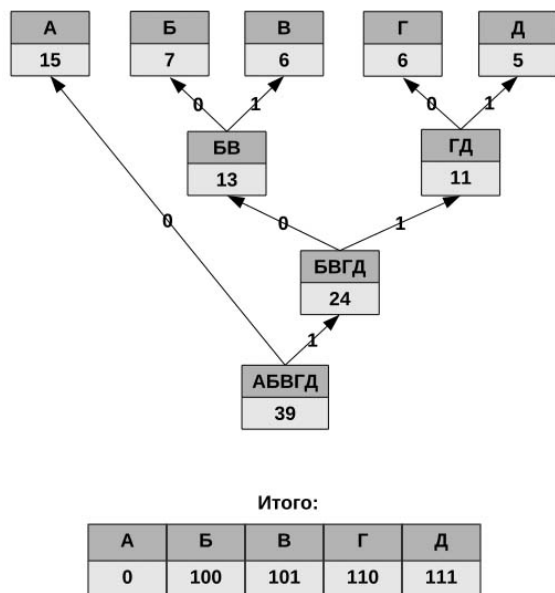


Рис. 1. Дерево кодов Хаффмана

Математическая модель данного алгоритма будет выглядеть таким образом [8]:

Ввод.

Алфавит $A = \{a_1, a_2, \dots, a_n\}$, представляющий собой алфавит символов размера n .

Кортеж $W = \{w_1, w_2, \dots, w_n\}$, представляющий собой кортеж положительных весов символов (обычно пропорциональных вероятностям). *Вывод.*

Код $C(W) = \{c_1, c_2, \dots, c_n\}$, представляющий собой набор двоичных кодовых слов, где c_i — кодовое слово для $a_i, i = \{1, 2, \dots, n\}$.

На практике наиболее часто применяют два алгоритма, основанных на методе Хаффмана: полуадаптивный и адаптивный (динамический).

При использовании полуадаптивного кодирования Хаффмана, кодировщик считывает сжимаемый файл дважды. Во время первого чтения вычисляются частоты встречаемости символов, а во время второго чтения происходит непосредственное сжатие файла. Для обеспечения возможности декодирования кодировщик добавляет в начало сжатого файла таблицу частот, занимающую несколько сотен байт, которая позволяет восстановить дерево кодирования и воссоздать исходный сжатый файл.

Основная идея адаптивного кодирования заключается в том, что компрессор и декомпрессор начинают работать с «пустого» дерева Хаффмана, а потом модифицируют его по мере чтения и обработки символов. Соответственно, кодер и декодер должны модифицировать дерево одинаково, чтобы все время использовать один и тот же код, который может меняться по ходу процесса. При таком кодировании нет необходимости сохранять таблицу частот в сжатый файл, а для кодирования требуется всего одно прочтение исходного файла (вместо двух), но при этом сжатие получается менее эффективным, чем при использовании полуадаптивного алгоритма [9, 10].

В данной работе будет реализован полуадаптивный вариант метода Хаффмана, который позволяет достичь баланса между эффективностью сжатия и вычислительной сложностью алгоритма.

ОПИСАНИЕ ПРОБЛЕМЫ

Практической задачей для оценки разрабатываемого алгоритма является обработка информации, передаваемой студентами в систему дистанционного образования. Группа из 30 студентов бакалавриата за один семестр загружает в систему не менее 120 мегабайт информации (примерно 4 мегабайта на каждого студента). Учитывая, что каждый студент в течение учебы может загрузить до 1 гигабайта данных, важно иметь эффективный алгоритм сжатия. Он должен обеспечить оптимальное использование ресурсов университета для хранения и обработки этой информации. Помимо сжатия, также важно учесть долгосрочное хранение данных после выпуска студентов, что подчеркивает значимость эффективности и надежности разработанного алгоритма.

В одном университете в среднем насчитывается 6 факультетов, каждый из них может иметь 5 и более кафедр, ответственных за группы студентов (причем на каждой кафедре более 3 групп). Если подсчитать объем информации, проходящей через систему дистанционного образования за один год со всех групп и факультетов, то количество информации составляет более 21 гигабайта, не считая значительного объема других данных, циркулирующих в системе.

Этот аспект может вызвать сложности в эффективном управлении большим объемом информации, передаваемой через систему дистанционного образования за год, что в свою очередь создает трудности в хранении, обработке и мониторинге всех работ, отправленных различными группами и факультетами.

Решением данной проблемы является кодирование информации, что поможет уменьшить объем, занимаемый лабораторными и практическими работами студентов.

ОПИСАНИЕ ПРОГРАММЫ-КОМПРЕССОРА

Программа включает в себя классы и функции, которые можно увидеть на диаграмме классов (рис. 2).

Описание классов:

1. Классы побитового чтения/побитовой записи файлов (BitFileWriter, BitFileReader). Классы, обеспечивающие побитовый ввод данных в файл и вывод данных из файла.

2. Класс кода Хаффмана (HuffmanCode). Класс, предназначенный для хранения сколь угодно длинных последовательностей нулей и единиц.

3. Класс кодового дерева (HuffmanTree). Класс, предназначенный для кодирования информации с помощью алгоритма Хаффмана.

4. Функция сжатия файла (compressFile). Функция, предназначенная для создания двух файлов:

- файла, содержащего последовательность кодов Хаффмана, полученного из исходного файла;
- текстового файла, содержащего расширение исходного файла и массив частот байтов, необходимый для построения дерева Хаффмана при декомпрессии.

5. Функция восстановления исходного файла (decompressFile). Функция предназначена для восстановления исходного файла.

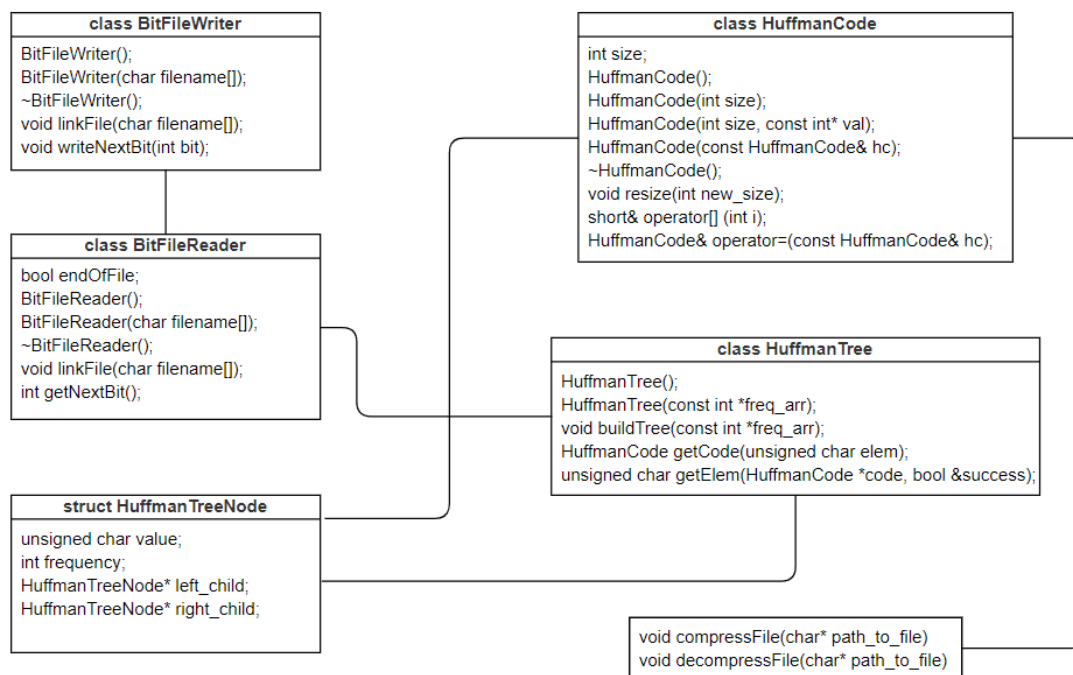


Рис. 2. Диаграмма классов программы

ОЦЕНКА ЭФФЕКТИВНОСТИ АЛГОРИТМА

Дать асимптотическую оценку эффективности работы созданной программы в зависимости от размера входных данных — достаточно сложная задача. Скорость работы разработанной реализации алгоритма Хаффмана зависит от размера исходного файла, а также его содержания (частот байтов).

Для изучения эффективности была проведена серия тестов, включающая использование различных файлов с лабораторными и практическими работами по различным дисциплинам. Результаты исследования представлены в таблице 1.

Таблица 1

Эффективность сжатия текстовых файлов в зависимости от их размера

Объем исходного файла, КБ	Объем сжатого файла, КБ	Объем файла с частотами, КБ	Суммарный объем после сжатия, КБ	Фактор сжатия
2	1,17	0,81	1,98	0,99
4	2,37	0,83	3,20	0,80
8	4,57	0,84	5,41	0,67
16	9,14	0,87	10,01	0,62
32	17,85	0,90	18,75	0,58
64	35,67	0,93	36,60	0,57
128	70,30	0,97	71,27	0,56
256	138,80	1,00	139,80	0,55
512	281,70	1,00	282,70	0,55
1 024	553,40	1,10	554,50	0,54

Из данных в таблице 1 следует, что эффективность сжатия текстовых файлов растет с увеличением объема сжимаемого файла. При достаточно больших объемах достигается сжатие примерно в два раза.

Измерения временных затрат на компрессию/декомпрессию текстовых файлов разного объема приведены в таблице 2.

Таблица 2

Время компрессию/декомпрессию текстовых файлов

Объем исходного файла, КБ	Время компрессии, мс	Время декомпрессии, мс
2	6	1
4	6	3
8	10	3
16	20	5
32	33	8
64	65	22
128	134	29
256	260	66
512	523	120
1 024	1 051	233

Из данных таблицы 2 можно выявить линейную зависимость времени сжатия и восстановления файлов программой от объема исходного файла.

ЗАКЛЮЧЕНИЕ

Разработанная программа оперирует байтами в качестве кодируемых символов и, следовательно, эффективна лишь для сжатия текстовой информации. Аудио-, графические и видеоданные содержат байты с приблизительно одинаковой частотой, поэтому для сжатия нетекстовой информации предпочтительнее применять специализированные методы.

При кодировании текстовых файлов, содержащих осмысленный текст на русском языке, при достаточно большом объеме сжимаемых файлов достигается сжатие примерно в два раза, что поможет уменьшить объем, занимаемый работами студентов.

Одним из недостатков реализованной программы является необходимость использования отдельного файла для хранения расширения исходного файла и таблицы частот символов. Этот недостаток можно исправить, включив указанную информацию в сжатый файл, однако это усложнит процесс чтения файла. При обработке больших объемов данных (более 2-3 МБ) скорость работы программы становится низкой из-за неэффективного метода получения кода символа из дерева кодов.

ЛИТЕРАТУРА

- Garg, N. Computer Sc — Data Structures and Algorithms: Lecture Series on Data Structures and Algorithms. Lecture 19. Data Compression. URL: <http://www.youtube.com/watch?v=5wRPin4oxCo&list=PLBF3763AF2E1C572F&index=19&t=251s> (дата обращения 18.11.2023).
- Виноградова, М. С. Сжатие данных. Алгоритм Хаффмана / М. С. Виноградова, О. С. Ткачева // *Modern European Researches*. 2022. Is. 3-1. Pp. 60–69.

- Сэлмон, Д. Сжатие данных, изображений и звука = A Guide to Data Compression Methods / Пер. с англ. В. В. Чепыгова. — Москва: Техносфера, 2004. — 368 с. — (Мир программирования. Цифровая обработка сигналов).

- Петрянин, Д. Л. Сжатие текстовых данных / Д. Л. Петрянин, Е. А. Сидорова, А. В. Зорькин // Новые информационные технологии в автоматизированных системах: Материалы семнадцатого научно-практического семинара. — Москва: ИПМ им. М. В. Келдыша, 2014. — С. 505–507.

- Марков, А. А. Введение в теорию кодирования. — Москва: Наука. Главная редакция физико-математической литературы, 1982. — 192 с.

- Variable-Length Code // Wikipedia. — Обновлено 11.11.2023. URL: http://en.wikipedia.org/wiki/Variable-length_code (дата обращения 19.11.2023).

- Huffman Coding // Wikipedia. — Обновлено 20.11.2023. URL: http://en.wikipedia.org/wiki/Huffman_coding (дата обращения 22.11.2023).

- Охотин, А. Математические основы алгоритмов. Курс лекций. URL: http://users.math-cs.spbu.ru/~okhotin/teaching/algorithms_2019 (дата обращения 22.11.2023).

- Алгоритм Хаффмана // Викиконспекты. — Обновлено 04.09.2022. URL: http://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_Хаффмана (дата обращения 22.11.2023).

- Левитин, А. В. Алгоритмы. Введение в разработку и анализ = Introduction to The Design and Analysis of Algorithms / пер. с англ. и ред. С. Г. Тригуб, И. В. Красикова. — Москва: Вильямс, 2006. — 576 с.

Development of a Data Compression Algorithm Based on the Huffman Method

R. M. Slesarev, M. A. Dobrovolskaya, PhD A. V. Zabrodin

Emperor Alexander I St. Petersburg State Transport University

Saint Petersburg, Russia

romsy2002@gmail.com, dobro-7878@mail.ru, teach-case@yandex.ru

Abstract. The article is devoted to the development of an algorithm for compressing data of various types, represented as files in the external memory of a computer, based on the Huffman method. The Huffman algorithm of minimal prefix coding is considered in detail. The purpose of writing this paper is to study and compare various data compression algorithms, as well as the practical application of the algorithm based on the Huffman method for writing a file compressor program.

Keywords: lossless compression, lossy compression, compression algorithms, coding, code tree.

REFERENCES

1. Garg, N. Computer Sc — Data Structures and Algorithms: Lecture Series on Data Structures and Algorithms. Lecture 19. Data Compression. URL: <http://www.youtube.com/watch?v=5wRPin4oxCo&list=PLBF3763AF2E1C572F&index=19&t=251s> (accessed 18 Nov 2023).
2. Vinogradova M. S., Tkacheva O. S. Data Compression. Huffman Algorithm [Szhatie dannykh. Algoritm Khaffmana], *Modern European Researches*, 2022, Is. 3-1, Pp. 60–69.
3. Salomon D. A Guide to Data Compression Methods [Szhatie dannykh, izobrazheniy i zvuka], Moscow, Tekhnosfera Publishing House, 2004, 368 p.
4. Petryanin D. L., Sidorova E. A., Zorkin A. V. Text Data Compression [Szhatie tekstovyykh dannykh], *New Information Technologies in Automated Systems: Materials of the 17th Scientific and Practical Seminar [Novye informatsionnye tehnologii v avtomatizirovannykh sistemakh: Materialy semnadsatogo nauchno-prakticheskogo seminar]*. Moscow, Keldysh Institute of Applied Mathematics of RAS, 2014, Pp. 505–507.
5. Markov A. A. Introduction to coding theory [Vvedenie v teoriyu kodirovaniya]. Moscow, Nauka Publishers, 1982, 192 p.
6. Variable-Length Code, *Wikipedia*. Last update at November 11, 2023. Available at: http://en.wikipedia.org/wiki/Variable-length_code (accessed 19 Nov 2023).
7. Huffman Coding, *Wikipedia*. Last update at November 20, 2023. Available at: http://en.wikipedia.org/wiki/Huffman_coding (accessed 22 Nov 2023).
8. Okhotin A. Mathematical foundations of algorithms: A course of lectures [Matematicheskie osnovy algoritmov. Kurs lektsiy] Available at: http://users.math-cs.spbu.ru/~okhotin/teaching/algorithms_2019 (accessed 22 Nov 2023).
9. Huffman Algorithm [Algoritm Khaffmana], *Wikinotes [Vikikonspekty]*. Last update at September 04, 2022. Available at: http://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_Хаффмана (accessed 22 Nov 2023).
10. Levitin A. V. Introduction to The Design and Analysis of Algorithms [Algoritmy. Vvedenie v razrabotku i analiz]. Moscow, Williams Publishing House, 2006, 576 p.

Тенденции развития создания интеллектуальной собственности в области разработки цифровых систем диспетчеризации угольного карьера

к.т.н. И. С. Сыркин, к.т.н. Д. А. Пашков, С. Д. Дубинкин
Кузбасский государственный технический университет имени Т. Ф. Горбачева
Кемерово, Россия
pashkovda@kuzstu.ru

Аннотация. Представлены тенденции развития создания интеллектуальной собственности в области разработки цифровых систем диспетчеризации угольного карьера, выявленные при проведении патентных исследований по мероприятию «Разработка и создание беспилотного карьерного самосвала челночного типа грузоподъемностью 220 тонн». На основании динамики патентования по годам дан прогноз развития конструкций цифровых систем диспетчеризации. Отмечаются прослеживаемые тенденции в патентовании конструкций цифровых систем диспетчеризации.

Ключевые слова: добыча полезных ископаемых, горные машины, карьерный самосвал, беспилотный карьерный самосвал, цифровая система диспетчеризации.

ВВЕДЕНИЕ

Горнодобывающие предприятия являются высокозатратными. Наибольшие затраты несут предприятия, добывающие полезные ископаемые открытым способом (карьерным) [1–4].

На карьерах основным потребителем затрат является погрузочно-транспортный процесс, который направлен на извлечение и транспортировку извлеченной горной массы из забоев в различные места складирования.

В исследованиях [5–7] отмечается, что 50 % эксплуатационных расходов на карьерах, а в некоторых случаях, особенно в крупных карьерах, даже до 60 % приходится на погрузочно-транспортные работы. Таким образом, улучшение транспортных операций и последующее снижение затрат на них даже на 2–3 % приведет к улучшению экономической ситуации на предприятии.

Существует два основных способа повышения эффективности транспортировки горной массы в карьерах:

– внедрение в автопарк самосвалов особо большой грузоподъемности, способных перевозить больше горной массы с каждой загрузкой;

– проведение исследования операций для повышения производительности работ [8–10].

В первом способе производители, видимо, достигли своего предела, создав карьерный самосвал грузоподъемностью 450 тонн (БелАЗ-75710). Дальнейшие ограничения

связаны с неготовностью дорог на карьерах, что в свою очередь ведет к необходимости дополнительных затрат, а также с пределом грузоподъемности крупногабаритных шин.

Для использования второго способа необходимо внедрить систему диспетчеризации на предприятии.

В рамках Комплексной научно-технической программы полного инновационного цикла (КНТП) по мероприятию «Разработка и создание беспилотного карьерного самосвала челночного типа грузоподъемностью 220 тонн» [11–15] необходимо разработать и создать прототип цифровой системы диспетчеризации (ЦСД) угольного карьера (УК).

При выполнении проекта были проведены патентные исследования уровня техники в части конструкций ЦСД УК.

Проведение исследований выполнялось по патентным документам (патентам, патентным заявкам), опубликованным различными патентными ведомствами мира:

- база данных международных заявок PATENTSCOPE (бесплатный поисковый сервис Всемирной организации интеллектуальной собственности);
- мультинациональная патентная база данных Европейского патентного ведомства Global Patent Index (GPI);
- базы данных Федерального института промышленной собственности (ФИПС);
- базы данных Роспатента (RUPAT);
- Евразийская патентно-информационная система (Eurasian Patent Information System, EAPATIS).

Классификационные рубрики международной патентной классификации (МПК) определялись по следующему отобранному ключевому термину: диспетчерский пункт (control room).

В соответствии с алфавитно-предметным указателем к МПК были выбраны следующие рубрики, которые в совокупности определяют требуемую область патентного поиска: B60, G01, G05, G06, G08G.

Глубина патентного поиска составила 25 лет (1998–2022 гг.).

Цель — определить тенденции развития создания интеллектуальной собственности в области разработки цифровых систем диспетчеризации угольного карьера.

Работа выполнена при финансовой поддержке Министерства науки и высшего образования Российской Федерации по соглашению от 30.09.2022 № 075-15-2022-1198 с ФГБОУ ВО «Кузбасский государственный технический университет имени Т. Ф. Горбачева» Комплексной научно-технической программы полного инновационного цикла «Разработка и внедрение комплекса технологий в областях разведки и добычи твердых полезных ископаемых, обеспечения промышленной безопасности, биоремедиации, создания новых продуктов глубокой переработки из угольного сырья при последовательном снижении экологической нагрузки на окружающую среду и рисков для жизни населения» (КНТП «Чистый уголь — Зеленый Кузбасс») в рамках реализации мероприятия «Разработка и создание беспилотного карьерного самосвала челночного типа грузоподъемностью 220 тонн» в части выполнения научно-исследовательских и опытно-конструкторских работ.

АНАЛИЗ ПАТЕНТНОГО ПОИСКА

По результатам патентного поиска выявлен 71 патентный документ.

Для определения прогноза развития конструкций ЦСД рассмотрена динамика патентования по годам, представленная на рисунке 1.

До 2001 года не было найдено ни одного патентного документа в части конструкции ЦСД, поэтому область с 1998 по 2001 год на рисунке не показана.

Как видно из рисунка 1, отмечается стабильное повышение патентования конструкций ЦСД.

ЦСД активно начали патентовать только с 2008 года, что в свою очередь говорит о начале работ в данном

направлении. Однако первые два технических решения ЦСД в 2001 и 2002 году были запатентованы сотрудниками университета Карнеги — Меллона (Carnegie Mellon University). К 2022 году наблюдается повышенный интерес к рассматриваемому объекту патентных исследований.

Анализ выявленных охраняемых технических решений дает основание выстроить краткосрочный (на 5 лет) прогноз развития конструкций ЦСД, а именно: патентование конструкций ЦСД будет активно увеличиваться.

Из выявленных патентных документов 69 — зарубежных, 2 — российских. Подробное распределение по странам представлено на рисунке 2.

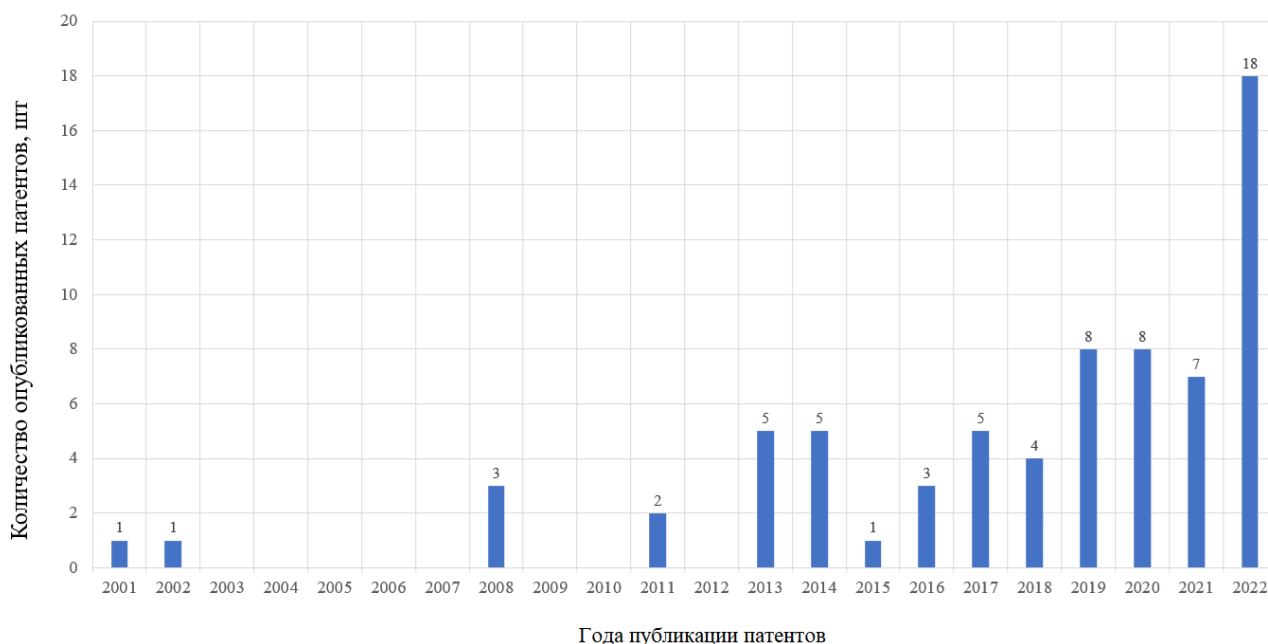


Рис. 1. Динамика патентования по годам

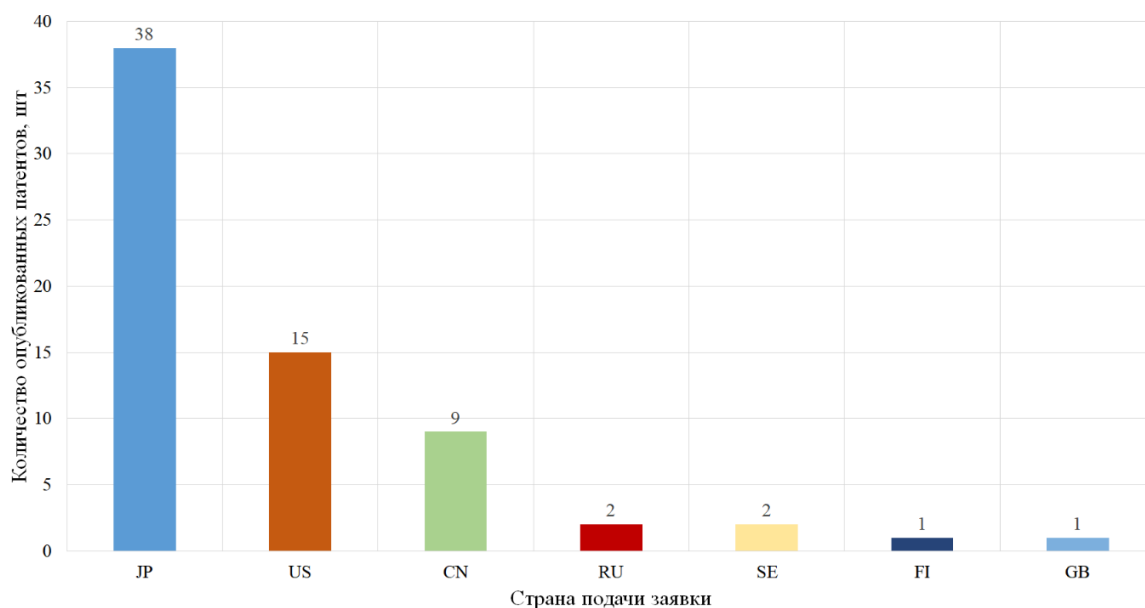


Рис. 2. География патентования в рассматриваемой области поиска

Стоит отметить, что в части патентования конструкций ЦСД лидирует Япония. Также активно патентуют технические решения в США, но в 2,5 раза меньше, чем в Японии.

География патентования по годам представлена на рисунке 3.

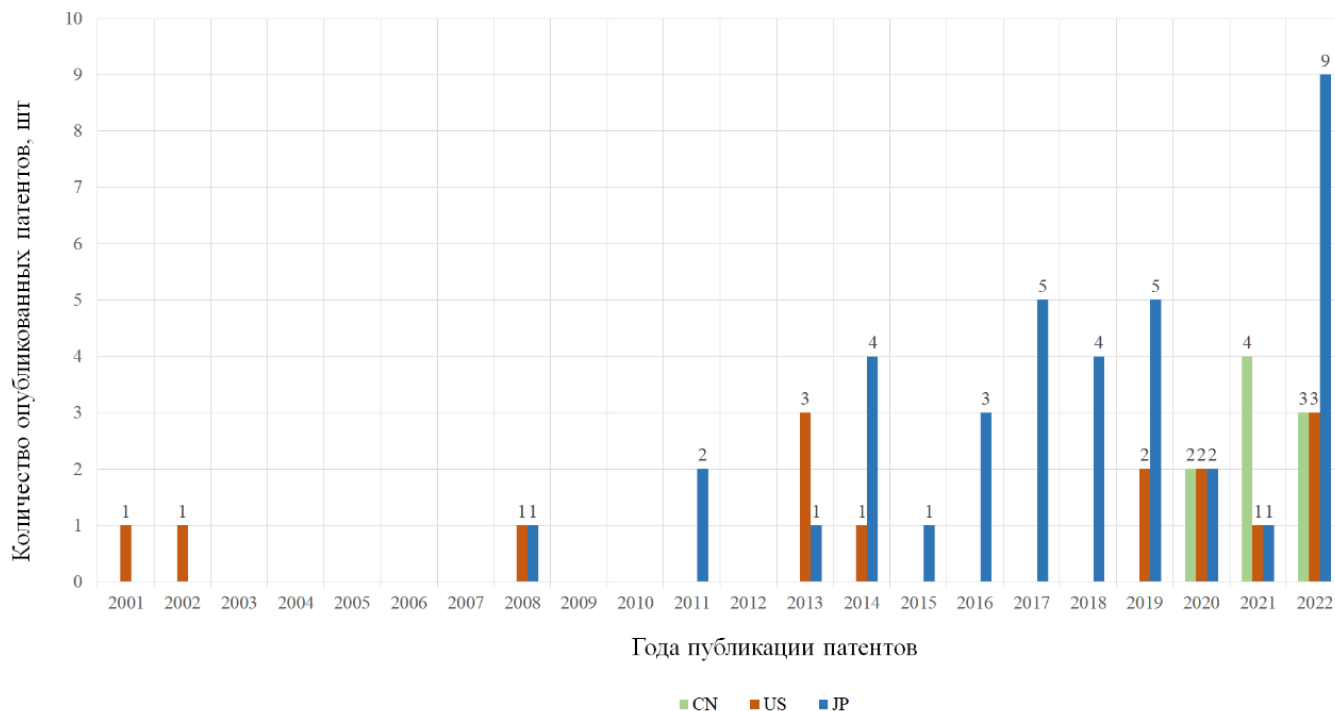


Рис. 3. География патентования в части конструкций цифровых систем диспетчеризации по годам

В патентовании технических решений конструкций ЦСД лидирует и активно набирает обороты Япония. Активные работы в этом направлении с 2020 года проводятся в Китае. Однако первые технические решения конструкций ЦСД были запатентованы в США в 2001 и 2002 годах.

Анализ патентообладателей выявленных патентных документов показал, что 59 % от всей полученной базы патентов принадлежит трем крупным мировым компаниям по производству карьерных самосвалов (рис. 4): Komatsu (Япония) — 17 патентов, Hitachi (Япония) — 16 патентов, Caterpillar Inc. (США) — 7 патентов.

ЗАКЛЮЧЕНИЕ

Прослеживая тенденции патентования в области конструкций ЦСД УК, можно отметить следующее:

- стабильное повышение патентования конструкций ЦСД;
- активно ЦСД начали патентоваться только с 2008 года, что говорит о начале работ в данном направлении;
- в ближайшие 5 лет прогнозируется заметное увеличение патентования конструкций ЦСД;
- в части конструкций ЦСД лидирует Япония.

ЛИТЕРАТУРА

1. Дубинкин, Д. М. Инженерные решения в повышении экологической безопасности карьерного транспорта / Д. М. Дубинкин, Н. Н. Голофастова // Экология и промышленность России. 2022. Т. 26, № 11. С. 8–12. DOI: 10.18412/1816-0395-2022-11-8-12.
2. Применение модульного метода для расчета показателей разработки угленасыщенной зоны на разрезах / А. А. Хорешок, А. В. Кацубин, Д. М. Дубинкин, [и др.] // Уголь. 2022. № S12 (1162). С. 76–81. DOI: 10.18796/0041-5790-2022-S12-76-81.
3. Дубинкин, Д. М. Основы цифрового создания автономных карьерных самосвалов // Горное оборудование и электромеханика. 2022. № 2 (160). С. 39–50. DOI: 10.26730/1816-4528-2022-2-39-50.
4. Дубинкин, Д. М. Перспективы высокотехнологичного производства карьерных самосвалов / Д. М. Дубинкин, Н. Н. Голофастова // Конкурентоспособность в глобальном мире: экономика, наука, технологии. 2022. № 5. С. 180–184.

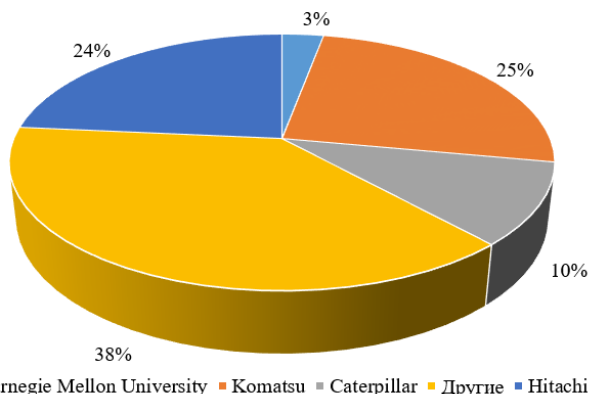


Рис. 4. Доля патентообладателей в рассматриваемой области поиска

5. Разработка программы и методики предварительных испытаний автономного карьерного самосвала / Д. М. Дубинкин, А. Б. Карташов, Г. А. Арутюнян, [и др.] // Горное оборудование и электромеханика. 2021. № 6 (158). С. 59–65. DOI: 10.26730/1816-4528-2021-6-59-65.

6. Об интенсивности изменения производительности автономной тяжелой платформы / М. А. Тюленев, С. О. Марков, Д. М. Дубинкин, В. В. Аксенов // Вестник Кузбасского государственного технического университета. 2021. № 1 (143). С. 97–108. DOI: 10.26730/1999-4125-2021-1-97-108.

7. Воронов, А. Ю. Обзор моделей диспетчеризации карьерного автотранспорта / А. Ю. Воронов, Д. М. Дубинкин, Ю. Е. Воронов // Горная промышленность. 2022. № 6. С. 111–121. DOI: 10.30686/1609-9192-2022-6-111-121.

8. Оптимизация параметров экскаваторно-автомобильных комплексов разрезов / А.Ю. Воронов, А. А. Хорешок, Ю. Е. Воронов, [и др.] // Горная промышленность. 2022. № 5. С. 92–98. DOI: 10.30686/1609-9192-2022-5-92-98.

9. Диспетчеризация в карьерных экскаваторно-автомобильных комплексах с беспилотным транспортом / Ю. Е. Воронов, А. Ю. Воронов, Д. М. Дубинкин, О. С. Максимова // Уголь. 2023. № 9 (1171). С. 75–83. DOI: 10.18796/0041-5790-2023-9-75-83.

10. Обзор систем безлюдных грузовых перевозок на карьерах / Ю. Е. Воронов, А. Ю. Воронов, И. С. Сыркин, [и др.] // Уголь. 2022. № S12 (1162). С. 30–36. DOI: 10.18796/0041-5790-2022-S12-30-36.

11. Дубинкин, Д. М. Тенденции развития грузовых платформ карьерных самосвалов / Д. М. Дубинкин, А. В. Ялышев, Ш. Я. Исмаилова // Горная промышленность. 2023. № 3. С. 72–76. DOI: 10.30686/1609-9192-2023-3-72-76.

12. Дубинкин, Д. М. Тенденции развития беспилотных карьерных самосвалов / Д. М. Дубинкин, В. В. Аксенов, Д. А. Пашков // Уголь. 2023. № 6 (1168). С. 72–79. DOI: 10.18796/0041-5790-2023-6-72-79.

13. Дубинкин, Д. М. Импортонезависимость производства беспилотных карьерных самосвалов / Д. М. Дубинкин, Д. А. Пашков // Уголь. 2023. № 4 (1166). С. 42–48. DOI: 10.18796/0041-5790-2023-4-42-48.

14. Обоснование параметров выемочно-погрузочного оборудования для опережающей выемки угольных пластов на разрезах / А. А. Хорешок, А. В. Кацубин, Д. М. Дубинкин, [и др.] // Уголь. 2022. № S12 (1162). С. 82–87. DOI: 10.18796/0041-5790-2022-S12-82-87.

15. Дубинкин, Д. М. Определение статических нагрузок на борт грузовой платформы карьерного самосвала / Д. М. Дубинкин, А. В. Ялышев // Горная промышленность. 2022. № 6. С. 137–144. DOI: 10.30686/1609-9192-2022-6-137-144.

Trends in the Development of Intellectual Property Creation in the Field of Development of Digital Coal Pit Dispatch Systems

PhD I. S. Syrkin, PhD D. A. Pashkov, S. D. Dubinkin
T. F. Gorbachev Kuzbass State Technical University
Kemerovo, Russia
pashkovda@kuzstu.ru

Abstract. The article presents trends in the development of digital coal pit dispatch system, identified in the course of patent research on the event on the topic «Development and Creation of an Unmanned Mine Dump Truck of Shuttle Type with a Carrying Capacity of 220 Tons». Based on the dynamics of patenting over the years, a forecast of the development of digital dispatch systems designs is given. Traceable trends in patenting designs of digital dispatch systems are noted.

Keywords: mining, mining machines, quarry dump truck, unmanned quarry dump truck, digital dispatching system.

REFERENCES

- Dubinkin D. M., Golofastova N. N. Engineering Solutions to Improve the Environmental Safety of Quarry Transport [Inzhenernye resheniya v povyshenii ekologicheskoy bezopasnosti karyernogo transporta], *Ecology and Industry of Russia [Ekologiya i promyshlennost Rossii]*, 2022, Vol. 26, No. 11, Pp. 8–12. DOI: 10.18412/1816-0395-2022-11-8-12.
- Khoreshok A. A., Katsubin A. V., Dubinkin D. M., et al. Using the Modular Method to Calculate the Indicators of Mining of the Coal-Bearing Zone at Opencast Mines [Primenenie modulnogo metoda dlya rascheta pokazatelya razrabotki ugle-nasyschennoy zony na razrezakh], *Ugol' — Russian Coal Journal [Ugol]*, 2022, No. S12 (1162), Pp. 76–81. DOI: 10.18796/0041-5790-2022-S12-76-81.
- Dubinkin D. M. Justification of the Need to Create Heavy Platforms for Open-Pit Mining [Osnovy tsifrovogo sozdaniya avtonomnykh karyernykh samosvalov], *Mining Equipment and Electromechanics [Gornoe oborudovanie i elektromekhanika]*, 2022, No. 2 (160), Pp. 39–50. DOI: 10.26730/1816-4528-2022-2-39-50.
- Dubinkin D. M., Golofastova N. N. Prospects for High-Tech Production of Quarry Dump Trucks [Perspektivy vysokotekhnologichnogo proizvodstva karyernykh samosvalov], *Competitiveness in the Global World: Economics, Science, Technology [Konkurentosposobnost v globalnom mire: ekonomika, nauka, tekhnologii]*, 2022, No. 5, Pp. 180–184.
- Dubinkin D. M., Kartashov A. B., Arutyunyan G. A., et al. Development of a Program and Methodology for Preliminary Testing of an Autonomous Quarry Dump Truck [Razrabotka programmy i metodiki predvaritelnykh ispytaniy avtonomnogo karyernogo samosvala], *Mining Equipment and Electromechanics [Gornoe oborudovanie i elektromekhanika]*, 2021, No. 6 (158), Pp. 59–65. DOI: 10.26730/1816-4528-2021-6-59-65.
- Tyulenev M. A., Markov S. O., Dubinkin D. M., Aksenov V. V. On the Intensity of Changing the Performance of the Autonomous Heavy Platform [Ob intensivnosti izmeneniya proizvoditelnosti avtonomnoy tyazhely platformy], *Bulletin of the Kuzbass State Technical University [Vestnik Kuzbasskogo gosudarstvennogo tekhnicheskogo universiteta]*, 2021, No. 1 (143), Pp. 97–108. DOI: 10.26730/1999-4125-2021-1-97-108.
- Voronov A. Y., Dubinkin D. M., Voronov Y. E. An Overview of Models for Truck Dispatching in Open-Pit Mines [Obzor modeley dispetcherizatsii karyernogo avtotransporta], *Russian Mining Industry Journal [Gornaya promyshlennost]*, 2022, No. 6, Pp. 111–121. DOI: 10.30686/1609-9192-2022-6-111-121.
- Voronov A. Y., Khoreshok A. A., Voronov Y. E., et al. Optimization of Parameters of Shovel-Truck Systems at Open-Pit Coal Mines [Optimizatsiya parametrov ekskavatorno-avtomobilnykh kompleksov razrezov], *Russian Mining Industry Journal [Gornaya promyshlennost]*, 2022, No. 5, Pp. 92–98. DOI: 10.30686/1609-9192-2022-5-92-98.
- Voronov Yu. E., Voronov A. Yu., Dubinkin D. M., Maksimova O. S. Dispatching in Truck-Shovel Systems with Unmanned Transport at Open-Pit Mines [Dispetcherizatsiya v karyernykh ekskavatorno-avtomobilnykh kompleksakh s bespilotnym transportom], *Ugol' — Russian Coal Journal [Ugol]*, 2023, No. 9 (1171), Pp. 75–83. DOI: 10.18796/0041-5790-2023-9-75-83.
- Voronov A. Yu., Voronov Yu. E., Syrkin I. S., et al. A Review of Unmanned Haulage Systems at Open-Pit Mines [Obzor sistem bezlyudnykh gruzovykh perevozok na karyerakh], *Ugol' — Russian Coal Journal [Ugol]*, 2022, No. S12 (1162), Pp. 30–36. DOI: 10.18796/0041-5790-2022-S12-30-36.
- Dubinkin D. M., Yalyshev A. V., Ismailova Sh. Ya. Trends in the Development of Unmanned Mining Dump Trucks [Tendentsii razvitiya gruzovykh platform karyernykh samosvalov], *Russian Mining Industry Journal [Gornaya promyshlennost]*, 2023, No. 3, Pp. 72–76. DOI: 10.30686/1609-9192-2023-3-72-76.
- Dubinkin D. M., Aksenov V. V., Pashkov D. A. Trends in the Development of Unmanned Mining Dump Trucks [Tendentsii razvitiya bespilotnykh karyernykh samosvalov], *Ugol' — Russian Coal Journal [Ugol]*, 2023, No. 6 (1168), Pp. 72–79. DOI: 10.18796/0041-5790-2023-6-72-79.

13. Dubinkin D. M., Pashkov D. A. Import-Independent Production of Unmanned Dump Trucks [Importonezavisimost proizvodstva bespilotnykh karyernykh samosvalov], *Ugol' — Russian Coal Journal [Ugol]*, 2023, No. 4 (1166), Pp. 42–48. DOI: 10.18796/0041-5790-2023-4-42-48.

14. Khoreshok A. A., Katsubin A. V., Dubinkin D. M., et al. Justification of Parameters of Excavation and Loading Equipment for Outpacing Excavation of Coal Seams at Opencast Mines [Obosnovanie parametrov vyemochno-pogruzochnogo oborudovaniya dlya operezhayushchey vyemki ugolnykh plastov na razrezakh], *Ugol' — Russian Coal Journal [Ugol]*, 2022, No. S12 (1162), Pp. 82–87. DOI: 10.18796/0041-5790-2022-S12-82-87.

15. Dubinkin D. M., Yalyshev A. V. Determination of Static Loads on the Body Sides of a Mining Dump Truck [Opredelenie staticheskikh nagruzok na bort gruzovoy platformy karyernogo samosvala], *Russian Mining Industry Journal [Gornaya promyshlennost]*, 2022, No. 6, Pp. 137–144. DOI: 10.30686/1609-9192-2022-6-137-144.

Об исследовании результатов подготовки бакалавров направления ИВТ ПсковГУ по очной и смешанной формам обучения

д.т.н. П. В. Герасименко

Петербургский государственный университет
пути сообщения Императора Александра I
Санкт-Петербург, Россия
pv39@mail.ru

д.т.н. С. М. Вертешев

Псковский государственный университет
Псков, Россия
president@pskgu.ru

Аннотация. Проведено исследование уровней междисциплинарных связей блоков математических, общинженерных и специальных дисциплин, достигнутых студентами по очной, дистанционной и смешанной формам обучения на четырех курсах Псковского государственного университета. В качестве показателя тесноты связей использован коэффициент корреляции, вычисляемый между сформированными векторами семестровых оценок студентов двух разных дисциплин. Выявлена достаточно слабая связь между уровнем математической подготовки выпускников школы и знаниями математических дисциплин вуза, что определило в последующем аналогичную связь между ними и общинженерными, а также специальными дисциплинами. С целью обеспечения успешного обучения студентов по дистанционной и смешанной формам рекомендовано знакомить студентов с информационными технологиями на первом курсе.

Ключевые слова: математические дисциплины, блоки дисциплин, очная форма обучения, дистанционная и смешанная формы обучения, средние оценки, коэффициент корреляции, пандемия.

Сегодня важнейшие задачи, которые стоят перед высшей школой, направлены на фундаментальную подготовку кадров, на развитие рационального соотношения между преподавательской и научно-исследовательской деятельностью с учетом достижений в областях педагогического образования и науки [1–4]. В Псковском государственном университете (ПсковГУ) аналогичные задачи стоят перед обучением бакалавров по направлению «Информатика и вычислительная техника» (ИВТ), которое связано с подготовкой IT-специалистов различного профиля. Области профессиональной деятельности выпускников в соответствии с учебным планом и профессиональными стандартами являются следующие:

- связь, информационные и коммуникационные технологии (код 06);
- администратор баз данных (код 06.011);
- специалист по администрированию сетевых устройств информационных и коммуникационных систем (код 06.027);
- программист (код 06.001);
- системный программист (код 06.028).

Следует отметить, что в последние годы в России продолжает сохраняться низкий уровень знаний выпускников школы. В дополнение к этому в России, как и во всем мире, разразилась пандемия COVID-19. Это существенно

усложнило учебный процесс в вузах. В рассматриваемый период ПсковГУ осуществлял обучение студентов очно, дистанционно и в смешанной формах.

Цель настоящего исследования — на основе анализа семестровых оценок студентов установить корреляционные связи между отдельными дисциплинами из блоков математических, общинженерных и специальных дисциплин при разных формах образовательного процесса студентов. Анализ выполнен на основании статистических данных образовательного процесса студентов направления ИВТ ПсковГУ.

Как известно, весной 2020 года все студенты в соответствии с решениями Минобрнауки и руководства Псковской области были полностью переведены на дистанционную форму обучения. В дальнейшем ограничительные меры были ослаблены и осенью 2021 г. повсеместно стала использоваться смешанная (очно-дистанционная) форма обучения.

В работе были рассмотрены две учебные группы набора 2015 и 2018 годов, выпуск которых состоялся в 2019 и 2022 году. Исследованы следующие блоки дисциплин: математические, общинженерные и специальные. Изучение дисциплин специального блока проходило в смешанной (дистанционной и частично очной) форме на третьем курсе в условиях распространения пандемии. После завершения пандемии на четвертом курсе студенты обучались очно.

Несмотря на сложные условия, в которых оказались вузы, в ПсковГУ особое внимание по-прежнему уделялось фундаментальной подготовке. Это обеспечивало возможность формирования у выпускника твердых знаний и способности быстро реагировать на изменения требований экономики.

В связи с этим учебный план предусматривал строгое соблюдение последовательности изучения дисциплин. Это требование означает, что последующие дисциплины должны опираться на дисциплины предыдущих семестров, а дисциплины первого курса — прежде всего на школьную математику и физику.

Обоснование такого требования можно наглядно проиллюстрировать сравнением этапов образовательного процесса в вузе с этапами строительства производственного здания. Схематично это сравнение представлено в таблице 1.

Таблица 1

Схема, характеризующая этапы строительного и образовательного процессов

Этап	Характеристики этапов	
	Строительство здания	Образовательный процесс в вузе
1	Выбор площадок под строительство из возможности грунта местности: болотистая, глинистая, твердая грунтовая, скалистая	Уровни знаний базовых предметов по завершению обучения в школе: удовлетворительный, хороший, отличный
2	Создание фундамента под производственное здание: точечное, ленточное, сплошное заливкой бетоном	Изучение дисциплин, определяющих фундаментальную подготовку, из числа которых, прежде всего для инженерных специальностей: математика и физика
3	Создание стен, крыши с формированием инфраструктуры	Изучение базовых общеинженерных дисциплин, базирующихся на фундаментальных дисциплинах
4	Установка оборудования для производственной деятельности	Изучение специальных дисциплин, формирующих направление профессионально ориентированного образования

Анализ таблицы 1 показывает, что процесс подготовки бакалавров с учетом современных требований должен предусматривать правильную последовательность изучения дисциплин. При этом необходимо обеспечивать высокий уровень тесноты междисциплинарных связей. Приобретение фундаментальных знаний позволит выпускнику согласно требованиям федерального государственного образовательного стандарта высшего образования не только овладеть своей профессией, но и разбираться в смежных областях деятельности [1].

Следует отметить, что вопрос об успешности обучения студентов по разным дисциплинам, а также взаимосвязь их успешности между разными дисциплинами является важным вопросом как с точки зрения оптимизации содержания учебных дисциплин, так и с точки зрения улучшения общего конечного результата — подготовки квалифицированных специалистов.

Для организации контроля качества освоения разных дисциплин в ПсковГУ была предусмотрена не только аттестация по итогам изучения дисциплины, но и так называемый входной контроль.

Помимо входных и выходных испытаний, успешность освоения дисциплин целесообразно контролировать путем текущей аттестации в течение семестра: короткие ответы на контрольные вопросы, выполнение несложных практических заданий и т. п. Форма проведения входных, выходных и текущих испытаний проходила как в письменной, так и в устной формах. Задания формулировались в виде контрольных вопросов и задач, в виде тестов или в виде экзаменационных билетов.

Существуют различные мнения относительно эффективности разных форм испытаний и видов заданий. Однако каждый из вариантов имеет свои достоинства и недостатки. Профессионализм преподавателя проявляется в

умении правильно сочетать разные формы испытаний и виды заданий, используя достоинства каждого варианта. В таблице 2 представлены результаты входного контроля знаний элементарной школьной математики будущих бакалавров, поступивших на направление ИВТ.

Таблица 2

Процент студентов, набравших количество баллов ЕГЭ

Интервал баллов ЕГЭ по элементарной математике	Количество студентов, %	
	Набор 2015 года	Набор 2018 года
27–60	77,8	66,9
61–80	24,6	29,3
81–90	1,6	3,8
91–100	0,0	0,0

Из таблицы 2 следует, что поступившие в вуз школьники как в 2015, так и в 2018 году практически не изменили свой уровень знаний по элементарной математике, так как достигли некоторого минимума, который сформировался в России [5–7]. Этот низкий уровень подтвердился в ПсковГУ в процессе обучения большим количеством отчисленных: в 2015 году завершило обучение 53 %, а в 2018 году — 49 % от числа поступивших. В таблице 3 представлены средние значения оценок по 6 математическим и 11 инженерным дисциплинам.

Таблица 3

Средние оценки по математическим и инженерным дисциплинам

№	Дисциплина	Средние оценки	
		Набор 2015 года	Набор 2018 года
1	Математическая логика	4,7	4,8
2	Алгебра и геометрия	3,7	3,3
3	Математический анализ	3,3	3,1
4	Теория вероятностей	3,6	4,1
5	Дискретная математика	3,6	3,9
6	Вычислительная математика	3,6	3,8
7	Физика	3,3	3,8
8	Программирование	4,1	4,2
9	Информатика	4,3	4,3
10	Теория алгоритмов	3,7	4,1
11	Теория кодирования	4,1	4,4
12	Электроника	3,7	3,9
13	Моделирование	3,6	3,8
14	Техника программирования	3,7	3,8
15	Основы теории управления	3,6	3,9
16	Ориентированное программирование	3,9	3,9
17	Инженерная и компьютерная графика	4,6	4,5

Из сравнения представленных в таблице 3 средних оценок видно, что уровни знаний по приведенным дисциплинам значимых отличий у поступивших в 2015 году и в 2018 году не имеют. Так, у набора 2015 года средняя оценка по математическим дисциплинам составляет 3,75, а по инженерным — 3,87, в то время как у набора 2018 года эти величины равны соответственно 3,83 и 4,06.

Низкий уровень знаний по элементарной математике в дальнейшем негативно отразился на процессе подготовки

студентов по вузовским математическим и инженерным дисциплинам.

Как показали дальнейшие исследования, низкий уровень подготовки по школьной математике вызвал также низкий уровень междисциплинарной связи между математическими и общинженерными дисциплинами.

Средние оценки каждого студента двух дисциплин математического и инженерного блоков позволили определить коэффициенты корреляции между этими дисциплинами, которые использованы в работе в качестве показателей тесноты междисциплинарных связей [8].

Вычисленные коэффициенты позволили представить их в виде корреляционной матрицы. Такие матрицы дают возможность провести анализ и установить тесноту междисциплинарных связи между блоками дисциплин [9].

Для набора 2018 года такая матрица построена для математических и общинженерных дисциплин. Она представлена в таблице 4.

Корреляционная матрица междисциплинарных связей для блоков математических и специальных дисциплин представлена в таблице 5.

Таблица 4

Корреляционная матрица связи математических и общинженерных дисциплин

Дисциплины	Математическая логика	Алгебра и геометрия	Математический анализ	Теория вероятностей	Дискретная математика	Вычислительная математика
Физика	0,57	0,62	0,49	0,51	0,60	0,58
Программирование	0,42	0,51	0,24	0,68	0,43	0,56
Информатика	0,37	0,13	0,03	0,78	0,34	0,54
Теория алгоритмов	0,63	0,52	0,26	0,73	0,55	0,66
Теория кодирования	0,44	0,37	0,24	0,45	0,53	0,64
Электроника	0,38	0,50	0,23	0,48	0,47	0,47
Моделирование	0,57	0,62	0,31	0,59	0,60	0,66
Основы теории управления	0,73	0,26	0,17	0,56	0,10	0,41
Инженерная и компьютерная графика	0,53	0,66	0,43	0,52	0,55	0,56

Таблица 5

Корреляционная матрица математических и специальных дисциплин

Дисциплины	Математическая логика	Алгебра и геометрия	Математический анализ	Вычислительная математика	Теория вероятностей	Дискретная математика
Программная инженерия	0,59	0,13	0,27	0,76	0,48	0,51
Технология программирования	0,32	0,22	0,20	0,83	0,38	0,46
Объектно-ориентированное программирование	0,47	0,18	0,18	0,60	0,58	0,40
Схемотехника ЭВМ	0,53	0,08	0,06	0,56	0,37	0,24
Операционные системы	0,45	0,20	0,14	0,67	0,51	0,41
Программирование в графических средах	0,50	0,13	0,12	0,48	0,48	0,27
Основы сетевых технологий	0,62	0,37	0,20	0,56	0,52	0,49
Управление данными	0,55	0,21	0,21	0,47	0,49	0,45
Системное ПО	0,51	0,23	0,11	0,52	0,54	0,29
Надежность вычислительных систем	0,59	0,36	0,26	0,59	0,54	0,46

Анализ результатов исследований, представленных в таблицах 4 и 5, свидетельствует о низком уровне тесноты как между дисциплинами математического и инженерного блоков, так и между блоками математических и специальных дисциплин. Средние коэффициенты корреляции между дисциплинами математического блока и в блоке между математическими и специальными дисциплинами равны соответственно 0,47 и 0,38. Если следовать результатам многих работ, посвященных влиянию школьной математики на изучение специальных дисциплин в разных вузах, то уровень полученных знаний по специальным дисциплинам, которые исследованы в данной работе, является достаточно низким, что и подтверждается работами.

Вместе с тем анализ массива коэффициентов корреляции позволяет заключить, что между отдельными дисциплинами существует тесная связь (коэффициент более 0,7), однако практически отсутствует линейная связь (коэффициент равен 0,03).

На основании проведенных исследований выявлены особенности образовательного процесса в ПсковГУ, по результатам которого следует сделать ряд замечаний, характерных для многих вузов, поскольку они зависят от общих недостатков современной средней школы. Результаты проведенных исследований позволяют сформулировать следующие выводы:

1. Работа со студентами в период пандемии COVID-19 показывает, что не все виды учебных занятий одинаково успешно реализуются в дистанционном режиме. При вынужденном полном переходе на дистанционный режим, а затем и на смешанный режим наименьший ущерб понесли студенты направления ИВТ, так как ко времени возникновения пандемии они были профессионально подготовлены к использованию средств и информационных технологий для дистанционного обучения. Студенты других образовательных направлений хуже справились с вынужденной изоляцией и дистанционной формой обучения. Несмотря на это, целесообразно разумно сочетать очную и дистанционную форму, так как это, во-первых, уменьшает трудоемкость работы преподавателей, во-вторых, обеспечивает определенную гибкость учебного процесса и возможность его адаптации к изменениям внешних условий, в-третьих, стимулирует студентов уделять больше времени самостоятельной работе над освоением дисциплин.

2. Признано, что во время возникновения угрозы массовых заболеваний обучаемых и преподавателей целесообразно организовывать работу мониторинговой службы на уровне вуза и региона. Целью этого является обеспечение своевременного изменения формы занятий. Во-первых, это исключает экстренные (авральные) действия по переходу на дистанционное обучение, во-вторых, своевременное введение дистанционного обучения способствует замедлению распространения болезни и скорейшей ликвидации эпидемии в регионе.

3. Анализ междисциплинарных связей позволяет корректировать содержание учебных дисциплин и последовательность их изучения. Для этого необходимо сочетать различные формы обучения (очная, дистанционная и смешанная), а также различные виды контроля успешности освоения учебных дисциплин студентами.

4. Входной контроль и семестровая аттестация студентов позволяет более четко установить междисциплинар-

ные связи и выявить пробелы в подготовке студентов по последующим дисциплинам относительно предыдущих. Тем самым появляется возможность выполнить необходимую корректировку содержания учебных дисциплин и последовательность их изучения.

ЛИТЕРАТУРА

1. Виноградов, Б. А. Системный подход в процедурах оценки качества подготовки персонала для ОПК / Б. А. Виноградов, В. Г. Пальмов, Г. П. Мещерякова // *Инновации*. 2014. № 10 (192). С. 70–78.

2. Поличка, А. Е. Особенности проектирования инновационной инфраструктуры подготовки кадров информатизации региональной системы образования в условиях функционирования информационно-коммуникационной предметной среды: Монография. — Хабаровск: ДВГУПС, 2015. — 86 с.

3. Ганичева, А. В. Математическая модель оценки качества обучения // *В мире научных открытий*. 2015. № 6.1 (66). С. 313–326. DOI: 10.12731/wsd-2015-6.1-313-326.

4. Ганичева, А. В. Оценка эффективности процесса обучения // *Интеллект. Инновации. Инвестиции*. 2011. № 2. С. 134–137.

5. Уразаева, Л. Ю. Проблемы математического образования и их решение / Л. Ю. Уразаева, Н. Н. Дацун // *Вестник Пермского университета. Математика. Механика. Информатика*. 2015. № 3 (30). С. 57–63.

6. Герасименко, П. В. Математическое моделирование процесса изучения учебных многосеместровых дисциплин в технических вузах / П. В. Герасименко, Е. А. Благовещенская, В. А. Ходаковский // *Известия Петербургского университета путей сообщения*. 2017. Т. 14, Вып. 3. С. 513–522.

7. Вертешев, С. М. Роль математики и информатики в подготовке инженеров для инновационной деятельности / С. М. Вертешев, П. В. Герасименко, С. Н. Лехин // *Перспективы развития высшей школы: Материалы X Международной научно-методической конференции (Гродно, Беларусь, 04–05 мая 2017 г.)*. — Гродно: Гродненский гос. аграрный ун-т, 2017. — С. 223–226.

8. Герасименко, П. В. Методика оценивания качества знаний выпускников вузов по уровню плотности межпредметных корреляционных связей экзаменационных оценок // *Эксперт: теория и практика*. 2022. № 3 (18). С. 75–78. DOI: 10.51608/26867818_2022_3_75.

9. Вертешев, С. М. Моделирование зависимости показателей знаний инженерных дисциплин от математических дисциплин при подготовке студентов по направлению ИВТ в Псковском государственном университете / С. М. Вертешев, П. В. Герасименко, С. Н. Лехин // *Инженерное образование*. 2019. № 25. С. 82–91.

About the Study of the Results of Bachelor's Degree Training in the IVT Direction in Pskov State University in Full-Time and Mixed Forms of Education

Grand PhD P. V. Gerasimenko
Emperor Alexander I St. Petersburg
State Transport University
Saint Petersburg, Russia
pv39@mail.ru

Grand PhD S. M. Verteshev
Pskov State University
Pskov, Russia
president@pskgu.ru

Abstract. A study of the close connections between blocks of mathematical, general engineering and special disciplines was carried out. The level of knowledge achieved by students over four years was analyzed. At the same time, training was carried out in full-time, distance and mixed forms. The correlation coefficient calculated between the vectors of semester grades was used as an indicator of the closeness of connections. The relationship between the level of mathematical training of school graduates and the level of training of students in mathematical disciplines of the university is shown. A connection was also revealed with the level of training in general engineering and special disciplines. In order to ensure successful learning in any form (full-time, distance and mixed), it is recommended to introduce all students to information technologies already in the first year.

Keywords: mathematical disciplines, blocks of disciplines, full-time form of education, distance and mixed forms of education, average grades, correlation coefficient, pandemic.

REFERENCES

1. Vinogradov B. A., Palmov V. G., Meshcheryakova G. P. A System Approach in Procedure of Quality Assessment of Personal Training for Military-Industrial Complex [Sistemnyy podkhod v protsedurakh otsenki kachestva podgotovki personala dlya OPK], *Innovations [Innovatsii]*, 2014, No. 10 (192), Pp. 70–79.
2. Polichka A. E. Features of designing an innovation infrastructure for training personnel for informatization of the regional education system in the context of the functioning of the information and communication subject environment: Monograph [Osobennosti proektirovaniya innovatsionnoy infrastruktury podgotovki kadrov informatizatsii regionalnoy sistemy obrazovaniya v usloviyakh funktsionirovaniya informatсионно-kommunikatsionnoy predmetnoy sredy: Monografiya]. Khabarovsk, Far Eastern State Transport University, 2015, 86 p.
3. Ganicheva A.V. Mathematical Model of the Assessment of Quality of Training [Matematicheskaya model otsenki kachestva obucheniya], *In the World of Scientific Discoveries [V mire nauchnykh otkrytiy]*, 2015, No. 6.1 (66), Pp. 313–326. DOI: 10.12731/wsd-2015-6.1-313-326.
4. Ganicheva A.V. Efficiency Evaluation of Education Process [Otsenka effektivnosti protsessy obucheniya], *Intellect. Innovations. Investments [Intellekt. Innovatsii. Investitsii]*, 2011, No. 2, Pp. 134–137.

5. Urazaeva L. Yu., Datsun N. N. Problems of Mathematical Education and Their Decision [Problemy matematicheskogo obrazovaniya i ikh reshenie], *Bulletin of Perm University. Mathematics. Mechanics. Computer Science [Vestnik Permskogo universiteta. Matematika. Mekhanika. Informatika]*, 2015, No. 3 (30), Pp. 57–63.

6. Gerasymenko P. V., Blagoveshenskaya Ye. A., Khodakovskiy V. A. Mathematical Simulation of Studying Academic Multi-Semestrial Disciplines in Technical Colleges [Matematicheskoe modelirovanie protsessy izucheniya uchebnykh mnogosemestrovnykh distsiplin v tekhnicheskikh vuzakh], *Proceedings of Petersburg Transport University [Izvestiya Peterburgskogo universiteta putey soobshcheniya]*, 2017, Vol. 14, Is. 3, Pp. 513–522.

7. Verteshev S. M., Gerasimenko P. V., Lekhin S. N. The Role of Mathematics and Informatics in the Training of Engineers for Innovative Activities [Rol matematiki i informatiki v podgotovke inzhenerov dlya innovatsionnoy deyatel'nosti], *Prospects for the Development of Higher Education: Proceedings of the X International Scientific and Methodological Conference [Perspektivy razvitiya vysshey shkoly: Materialy X Mezhdunarodnoy nauchno-metodicheskoy konferentsii]*, Grodno, Belarus, May 04–05, 2017. Grodno, Grodno State Agrarian University, 2017, Pp. 223–226.

8. Gerasimenko P. V. Methods for Assessing the Quality of Graduates Knowledge by the Level of Density of Inter-Meth Correlation Relationships of Examination Assessments [Metodika otsenivaniya kachestva znaniy vypusnikov vuzov po urovnyu plotnosti mezhpredmetnykh korrelyatsionnykh svyazey ekzamenatsionnykh otsenok], *Expert: Theory and Practice [Ekspert: teoriya i praktika]*, 2022, No. 3 (18), Pp. 75–78. DOI: 10.51608/26867818_2022_3_75.

9. Verteshev S. M., Gerasimenko P. V., Lekhin S. N. Modeling of the Dependence of Indicators of Knowledge of Engineering Disciplines on Mathematical Disciplines When Preparing Students in the Direction «Informatics and Computing Technology» at Pskov State University [Modelirovanie zavisimosti pokazateley znaniy inzhenernykh distsiplin ot matematicheskikh distsiplin pri podgotovke studentov po napravleniyu IVT v Pskovskom gosudarstvennom universitete], *Engineering Education [Inzhenernoe obrazovanie]*, 2019, No. 25, Pp. 82–91.

ПАМЯТИ КОЛЛЕГИ



Добрую память о себе в умах преподавателей, ученых, коллег и студентов оставил профессор, доктор технических наук, профессор кафедры «Информатика и информационная безопасность», КУСТОВ Владимир Николаевич (29.09.1948–11.12.2023).

Выпускник Свердловского суворовского военного училища, которое закончил в 1967 году с золотой медалью, и Пермского высшего командно-инженерного училища, которое окончил также с золотой медалью.

Владимир Николаевич с 1972 по 1998 год служил в академии имени А. Ф. Можайского на разных должностях от инженера лаборатории до начальника кафедры электронной вычислительной техники, а после увольнения в запас в 1999–2001 годах продолжил работу в академии профессором кафедры электронной вычислительной техники.

В 2002–2004 годах — начальник сектора информационной безопасности информационных систем, далее — заместитель генерального директора Санкт-Петербургского филиала «Внедренческий центр» ГУП АЦ «Желдоринформзащита» МПС России.

С 2003 года Владимир Николаевич являлся профессором кафедры «Информатика и информационная безопасность» Петербургского государственного университета путей сообщения Императора Александра I. Одновременно с 2004 до последних дней жизни Владимир Николаевич трудился на руководящих должностях в группе компаний «Газинформсервис».

Активно участвовал в формировании и работе научной школы информационной безопасности и защиты информации в информационно-управляющих и телекоммуникационных системах транспортного и космического назначения. Подготовил 6 кандидатов технических наук, 2 его ученика стали докторами наук. Член диссертационного совета ПГУПС по специальности 2.3.6, член диссертационного совета ИТМО по специальности 2.3.2.

Автор более 300 научных и учебно-методических трудов, в том числе монографий, учебников, научных статей, авторских свидетельств и патентов. Внес огромный вклад в развитие теории расписаний и стеганографии.

Ушел из жизни профессионал, ученый, искренне преданный науке, педагог, наставник, умело применяющий свои глубокие научные знания на практике, светлый, добрый, честный, искренний человек.

In Memory of Colleague

A good memory of himself in the minds of teachers, scientists, colleagues and students was left by Professor, Doctor of Technical Sciences, Professor of the Department of Information Technology and IT Security KUSTOV Vladimir Nikolaevich (29.09.1948–11.12.2023).

He is a graduate of the Sverdlovsk Suvorov Military School, which he graduated in 1967 with a gold medal, and the Perm Higher Command and Engineering School, which he also graduated with a gold medal.

Vladimir Nikolaevich from 1972 to 1998 served at the Mozhaisky Military Space Academy in various positions from laboratory engineer to Head of the Department of Electronic Computer Technology, and after his retirement in 1999–2001, he continued to work at the Academy as a professor of the Department of Electronic Computer Technology.

In 2002–2004 — Head of the Information Security Sector of Information Systems and then Deputy General Director of the St. Petersburg Branch «Implementation Center» State Unitary Enterprise Certification Center «Zheldorinformzashchita» of the Ministry of Railways of Russia. Since 2003, Vladimir Nikolaevich has been a Professor at the Department of Information Technology and IT Security of the Emperor Alexander I St. Petersburg State Transport University. At the same time, from 2004 to the last days of his life, Vladimir Nikolaevich worked in senior positions in the Gazinformservis Group of Companies.

Actively participated in the formation and work of the scientific school of information security and information protection in information management and telecommunications systems for transport and space purposes. Prepared 6 candidates of technical sciences, 2 of his wards became doctors of science. Member of the Dissertation Council of PGUPS in specialty 2.3.6, member of the Dissertation Council of ITMO University in specialty 2.3.2.

Author of more than 300 scientific and educational works, including monographs, textbooks, scientific articles, author's certificates and patents. He made a huge contribution to the development of the theory of schedules and steganography.

A professional, a scientist sincerely devoted to science, a teacher, a mentor who skillfully applies his deep scientific knowledge in practice, a bright, kind, honest, sincere person, has passed away.