http://dx.doi.org/10.35596/1729-7648-2023-29-3-64-74

Оригинальная статья
Original paper

УДК 004.032.26, 004.822.2

МОДЕЛЬ РЕШАТЕЛЯ ЗАДАЧ ИНТЕЛЛЕКТУАЛЬНОГО ФРЕЙМВОРКА ПО РАЗРАБОТКЕ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

М. В. КОВАЛЁВ

Белорусский государственный университет информатики и радиоэлектроники
(г. Минск, Республика Беларусь)

Поступила в редакцию 24.07.2023

Аннотация. В статье описан подход к реализации способности интеллектуальных систем к самообучению за счет интеллектуального фреймворка по разработке искусственных нейронных сетей. Предлагается метод взаимодействия между интеллектуальными системами и фреймворком на основании библиотеки многократно используемых компонентов, позволяющий автоматически проектировать и обучать искусственные нейронные сети с помощью заданной спецификации задачи. На основании анализа деятельности разработчиков искусственных нейронных сетей описана модель решателя задач такого фреймворка в виде иерархии действий по разработке искусственных нейронных сетей.

Ключевые слова: искусственная нейронная сеть, автоматизация, база знаний, решатель задач, интеллектуальный фреймворк.

Конфликт интересов. Автор заявляет об отсутствии конфликта интересов.

Для цитирования. Ковалёв, М. В. Модель решателя задач интеллектуального фреймворка по разработке искусственных нейронных сетей / М. В. Ковалёв // Цифровая трансформация. 2023. Т. 29, № 3. С. 64–74. http://dx.doi.org/10.35596/1729-7648-2023-29-3-64-74.

PROBLEM SOLVER MODEL OF INTELLIGENT FRAMEWORK FOR THE DEVELOPMENT OF ARTIFICIAL NEURAL NETWORKS

MIKHAIL V. KOVALEV

Belarusian State University of Informatics and Radioelectronics (Minsk, Republic of Belarus)

Submitted 24.07.2023

Abstract. The article describes an approach to implementing the ability of intelligent systems to self-learn through an intelligent framework for the development of artificial neural networks. A method of interaction between intelligent systems and a framework based on a library of reusable components is proposed, which allows to automatically design and train artificial neural networks based on a given problem specification. Based on the analysis of the activity of developers of artificial neural networks, a model of the problem solver of such a framework is described in the form of a hierarchy of actions for the development of artificial neural networks.

Keywords: artificial neural network, automation, knowledge base, problem solver, intelligent framework.

Conflict of interests. The author declares no conflict of interests.

For citation. Kovalev M. V. (2023) Problem Solver Model of Intelligent Framework for the Development of Artificial Neural Networks. *Digital Transformation.* 29 (3), 64–74. http://dx.doi.org/10.35596/1729-7648-2023-29-3-64-74 (in Russian).

Введение

Современное развитие всех направлений искусственного интеллекта направлено на построение интеллектуальных систем, автоматизирующих все более сложные виды человеческой деятельности. Ключевой особенностью таких систем является способность к самообучению, что, в свою очередь, подразумевает такие способности, как:

– постоянное повышение качества решения задач;

– приобретение навыков решения принципиально новых задач;

– обоснование своих решений;

– поиск и устранение ошибок в своих решения (способность к интроспекции).

Решение таких задач, как управление во время нештатных ситуаций, адаптация систем под уникальную ситуацию конкретных пользователей, принятие решений в условиях постоянно меняющейся окружающей обстановки, требует от интеллектуальных систем наличия способности к самообучению. Разработчики интеллектуальных систем не могут заранее предусмотреть в системе методы, решающие задачи, о которых они не догадываются во время разработки. Поэтому одной из основных проблем, стоящих перед разработчиками интеллектуальных систем нового поколения, является автоматизация процесса разработки методов решения задач на основании спецификации задачи, т. е. способность системы самой разрабатывать необходимые ей компоненты.

Самый популярный метод решения в настоящее время – искусственные нейронные сети (ИНС), что подтверждается последними достижениями в области использования больших языковых моделей [1, 2], систем компьютерного зрения [3], распознавания и синтеза речи [4] и т. д. Автоматизация процесса разработки именно ИНС представляется наиболее актуальной задачей для реализации способности к самообучению у современных интеллектуальных систем.

Цель исследований автора – описание принципов работы интеллектуального фреймворка по разработке ИНС, способного автоматически проектировать и обучать ИНС на основании спецификации задачи.

Современные инструменты разработки

Сегодня разработано большое количество различных библиотек, позволяющих успешно внедрять решения на основе интеллектуальных алгоритмов. Однако для этого необходимо иметь набор знаний и навыков, дающих возможность модифицировать и улучшать стандартные решения. Ниже приведен краткий обзор основных используемых в настоящее время фреймворков.

TensorFlow[1] – одна из самых популярных библиотек. Была разработана подразделением Google (2015). Позволяет запускать модели на нескольких CPU- и GPU-устройствах (хорошо масштабируется). Доступна для разных платформ, поддерживает различные языки программирования (C++, R, Python). Основные функции TensorFlow включают:

– поддержку нескольких видеокарт для выполнения вычислений;

– обучение с применением распределенных ресурсов (например, в облаке);

– визуализацию графа TensorFlow с помощью TensorBoard – специальной утилиты, которая поддерживает функции визуализации процесса обучения и визуализации тренировочной выборки;

– сохранение состояния модели – пользователи TensorFlow могут останавливать процесс обучения и продолжать его с определенной точки сохранения.

Caffe/*Caffe2*[2] – одна из первых библиотек глубокого обучения. Написана на C++, имеет интерфейс на Python. В основном ориентирована на обучение и использование моделей, построенных на сверхточных и многослойных сетях. Создано большое количество предобученных архитектур для Caffe. Facebook в 2017 году предложил новую версию Caffe2, которая дает большую гибкость в построении высокопроизводительных глубоких моделей. Может быть применена при разработке архитектур для использования на мобильных устройствах. Хорошо документирована.

Theano[3] – одна из первых библиотек, реализующих алгоритмы глубокого обучения. В определенной степени является низкоуровневой, имеет определенные проблемы с масштабируемостью

[1] TensorFlow. Available: https://tensorflow.org (Accessed 20 June 2023).

[2] Caffe2 | A New Lightweight, Modular, and Scalable Deep Learning Framework. Available: https://caffe2.ai/ (Accessed 20 June 2023).

[3] Theano. Available: https://github.com/Theano/Theano (Accessed 20 June 2023).

и вычислениями на кластере видеокарт. В целом пользуется устойчивой популярностью у специалистов в области глубокого обучения. Часто применяется как нижележащий фреймворк для многих высокоуровневых библиотек, которые предоставляют API-обертки (к примеру, Keras).

Keras[4] – предоставляет упрощенный интерфейс для работы с Theano, TensorFlow или CNTK. Очень легковесный и простой в изучении и использовании. Хорошо документирован. Позволяет описать создание и обучение ИНС в нескольких строчках кода. Написан на Python.

PyTorch[5] – очень простая библиотека, процесс построения моделей максимально упрощен. Возможно использование библиотек Python и CUDA.

Количество фреймворков постепенно растет, однако фактически каждый из них полностью повторяет другие, и особые различия включают лишь дополнительные модели и поддержку новых аппаратных возможностей. Основной недостаток всех разрабатываемых сегодня фреймворков – отсутствие возможности автоматической разработки ИНС на основании спецификации задачи. Перечисленные фреймворки являются инструментом разработчиков. Когда система сталкивается с ситуацией отсутствия метода решения поставленной перед ней задачи, она может только сообщить об этом пользователю. Пользователь может обратиться к разработчикам с просьбой разработать необходимый метод, и только тогда рассматриваемые фреймворки используются. В лучшем случае разработанные однажды ИНС смогут автоматически дообучаться на новых данных и повышать свою точность, однако этого недостаточно для реализации требуемой способности к самообучаемости интеллектуальных систем.

На рис. 1 представлена диаграмма последовательности, описывающая процесс решения задачи в случае отсутствия необходимого метода.

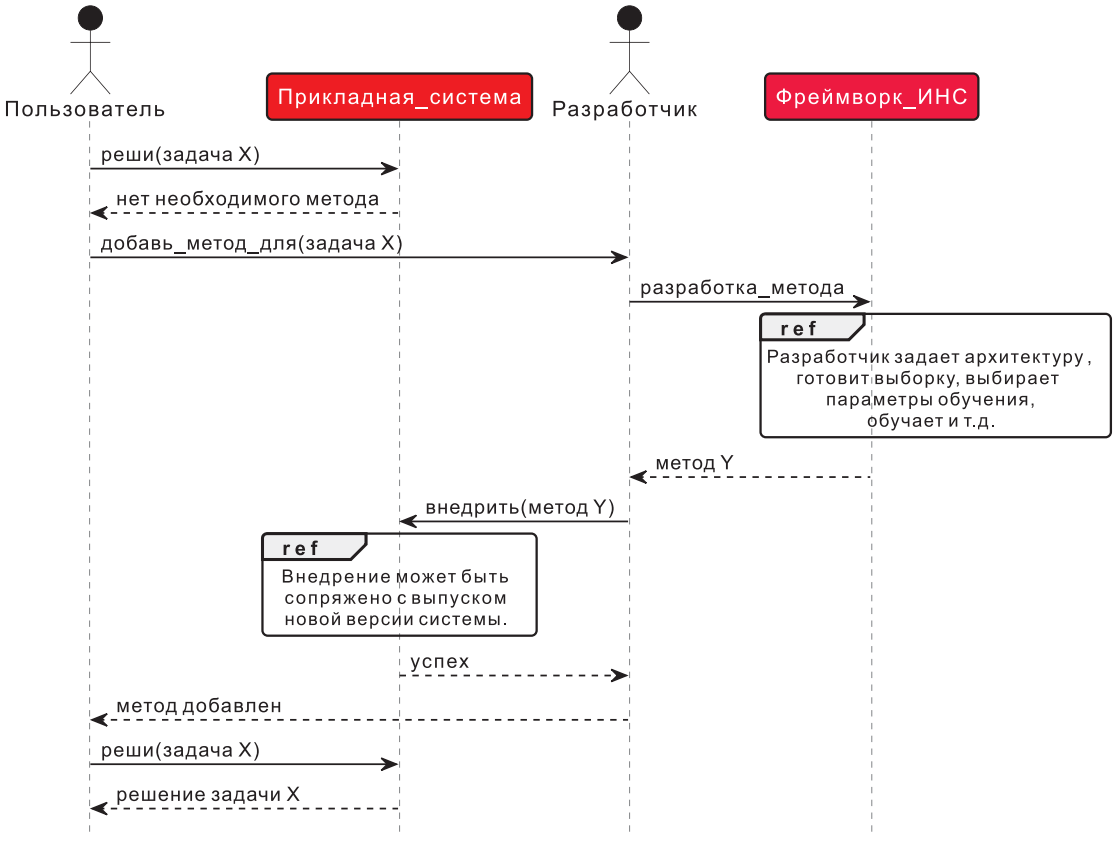

Рис. 1. Диаграмма последовательности решения задачи в условиях отсутствия необходимого метода ее решения
Fig. 1. Problem sequence diagram in the absence of the necessary method for solving it

Следует отметить, что серьезным недостатком является требование высокого уровня знаний о структурах моделей, которые должны использоваться для каждой конкретной задачи.

[4] Keras: Deep Learning for Humans. Available: https://keras.io/ (Accessed 20 June 2023).
[5] PyTorch. Available: https://pytorch.org/ (Accessed 20 June 2023).

Предлагаемый подход

Обеспечение способности к самообучению у интеллектуальных систем принципиально возможно в концепции, предложенной проектом OSTIS [5]. В первую очередь за счет унификации представления и онтологического структурирования знаний, описывающих задачи, предметные области, в рамках которых решаются задачи, и методы решения задач. Интеллектуальные системы, разработанные с применением технологии OSTIS, называются ostis-системами. Любая ostis-система состоит из базы знаний (БЗ), решателя задач и пользовательского интерфейса.

Представление различных методов решения задач в единой БЗ обеспечивает семантическую совместимость этих методов. Решая задачу с помощью таких методов, система не взаимодействует с ними по принципу «входов-выходов». Напротив, единая память позволяет отслеживать преобразование входных знаний в реальном времени с помощью любых имеющихся методов, что обеспечивает способность к интроспекции и способность объяснять решения системы.

Наличие в единой памяти не только экземпляров методов, но и понятий, их описывающих, создает основу для автоматизации процесса разработки ИНС. В памяти ostis-системы хранятся знания о том, методы какого класса могут решить задачу заданного класса, но экземпляров класса данного метода может не быть представлено в системе. На этот случай система должна иметь возможность самостоятельно разработать необходимый метод. Так как система хранит в единой памяти задачу и требования к методу ее решения, появляется возможность разработать нужный метод. Для этого необходимо наличие фреймворков по разработке методов соответствующих классов.

В случае ИНС речь идет об интеллектуальном фреймворке по разработке ИНС. Для этого фреймворка выдвинуты следующие функциональные требования:

– наличие семантической совместимости с ostis-системами;
– генерация ИНС на основании спецификации задачи;
– автоматизация этапов разработки ИНС;
– автоматизация процессов сравнения эффективности нейросетевых моделей;
– информационная поддержка пользователя на всех этапах разработки ИНС;
– использование библиотеки обученных и предобученных ИНС.

Семантическая совместимость достигается за счет использования ostis-системами одних и тех же многократно используемых компонентов [6]. Суть данного подхода заключается в том, что ostis-система, столкнувшаяся с отсутствием метода решения задач, делает запрос в интеллектуальный фреймворк на разработку необходимого метода и передает спецификацию задачи. Вместе со спецификацией передаются адреса и версии компонентов БЗ, наличие которых необходимо для того, чтобы фреймворк мог понять задачу (иметь онтологию всех понятий, использующихся в спецификации задачи). Если каких-то компонентов у фреймворка нет, он скачивает их по переданному адресу. Поняв задачу, фреймворк разрабатывает нужную ИНС (при необходимости обращаясь за дополнительной информацией к системе, поставившей задачу, или к разработчику) и публикует ее как компонент в библиотеке. Теперь обратившаяся к фреймворку система может скачать разработанную ИНС в виде компонента, скачать ее интерпретатор, если необходимо, и решить исходную задачу.

На рис. 2 представлена диаграмма последовательности, описывающая процесс взаимодействия ostis-системы и интеллектуального фреймворка при запросе разработки ИНС.

Модель решателя задач интеллектуального фреймворка по разработке искусственных нейронных сетей

Модель интеллектуального фреймворка по разработке ИНС, как и модель любой другой ostis-системы, задается моделью БЗ, решателя задач и интерфейса. Модель БЗ уже описана в [7]. Модель интерфейса интеллектуального фреймворка задается языком общения между ostis-системами [5]. Таким образом, для того, чтобы задать модель интеллектуального фреймворка по разработке ИНС, остается описать модель решателя задач.

Решатель задач занимается обработкой фрагментов БЗ. На операционном уровне обработка сводится к добавлению, поиску, редактированию и удалению конструкций БЗ. На семантическом же уровне такая операция является действием, выполняемым в памяти субъекта действия, где в общем случае субъект – это ostis-система, а БЗ – ее память.

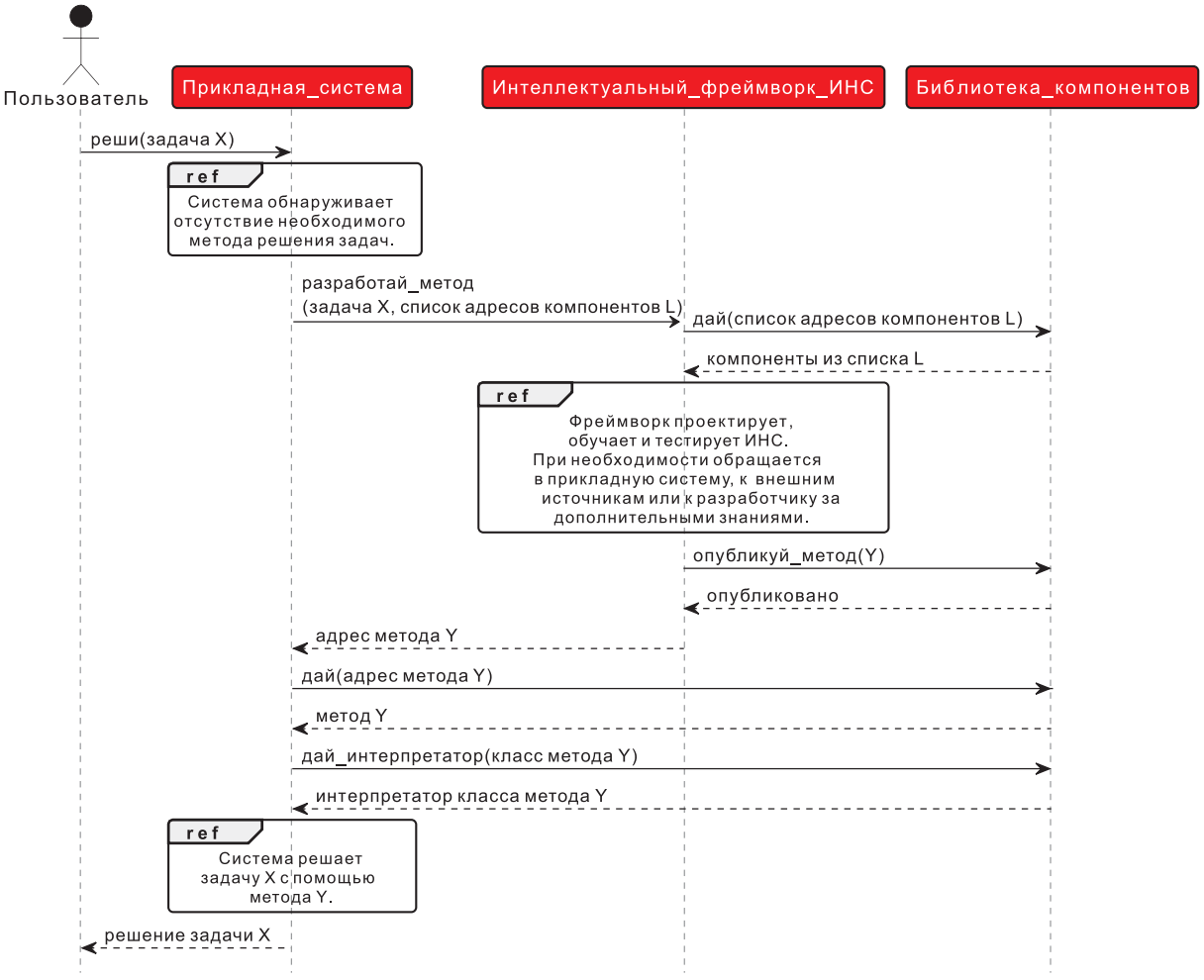

Рис. 2. Диаграмма последовательности решения задачи
в условиях отсутствия необходимого метода ее решения с помощью интеллектуального фреймворка
Fig. 2. Problem sequence diagram
in the absence of the necessary method for solving it with the help of an intelligent framework

Действие определяется как процесс воздействия одной сущности (или некоторого множества сущностей) на другую (или на некоторое множество других сущностей) в соответствии с некоторой целью. Модель решателя задач любой ostis-системы задается иерархией и спецификациями действий, которые он выполняет.

На основании действий, в общем случае выполняемых всеми разработчиками ИНС, выделена следующая иерархия действий решателя задач интеллектуального фреймворка.

1. *Действие спецификации задачи.* Включает в себя получение описания входных данных (изображения/видео, временные ряды, текст), выходных данных и требований к методу решения (скорость, затраты по памяти и т. д.). Также описывается дополнительная информация, которая может помочь в построении метода решения задачи (к примеру, спецификация обучающей выборки, если таковая имеется). На данном этапе определяется класс задачи, формируется требование к обучающей выборке, если она не предоставлена. Выполнение этого этапа фреймворком подразумевает осуществление следующих действий:

– трансляцию условия задачи;
– классификацию задачи;
– формирование требования к обучающей выборке;
– формирование обучающей выборки.

Формирование обучающей выборки может осуществляться в базе источников выборок, интегрированной в фреймворк, или в БЗ запрашивающей ostis-системы.

2. *Действие очистки выборки*. Обнаруживает признаки, которые имеют в общем случае некорректные значения. Например, для каких-то образов значение признака может иметь неопределенное значение, либо значение, не совпадающее по типу, либо аномально большое или очень маленькое значение, которое встречается в редком числе случаев. Действие выполняется при обработке выборки, ранее не представленной в памяти системы (к примеру, полученной от пользователя). Реализация интерпретатора (агента) данного действия требует описания в памяти классификации стратегий очистки данных и реализации методов применения этих стратегий.

3. *Действие выявления содержательных признаков*. Осуществляется так называемый инжиниринг признаков, состоящий в отборе признаков, влияющих на результат работы ИНС, а несодержательные признаки, которые никак не коррелируют с выходом ИНС, удаляются. Цель этого действия – уменьшение размерности пространства признаков для снижения влияния эффекта переобучения на ИНС. Реализация интерпретатора (агента) данного действия требует описания в памяти классификации стратегий снижения размерности признакового пространства и реализации методов применения этих стратегий.

4. *Действие трансформации признаков.* Трансформируются категориальные признаки, чаще всего заданные строковыми типами. Эти признаки могут быть номинальными и порядковыми. Для кодирования порядковых признаков чаще всего применяют последовательный числовой код (1, 2, 3, ...). Для кодирования номинальных такое решение неверно, поскольку эти признаки равноправны и не могут сравниваться по числовому коду (например, пол – 0/1). Для номинальных признаков применяется способ прямого кодирования, заключающийся в создании и использовании фиктивных признаков по количеству значений исходного. Например, признак пол (мужской, женский) преобразуется в два новых признака мужской и женский с соответствующими значениями для имеющихся образов. Реализация интерпретатора (агента) данного действия требует описания в памяти классификации методов масштабирования признаков и реализации методов применения этих стратегий.

5. *Действие разбиения выборки.* Производится разбиение всей выборки данных на обучающую, тестовую и в некоторых случаях валидационную. Валидационная выборка используется для оценки влияния изменения гиперпараметров на результат обучения. Обычно разбиение проводится в соотношении 3:1:1 – в процентах (60/20/20), если валидационная выборка не используется, то 70/30.

6. *Действие выбора класса ИНС.* На основании спецификации задачи решатель задач фреймворка осуществляет выбор архитектуры ИНС, которая будет применяться при обучении. Классификация ИНС описана в [7]. Нужно отметить, что этот выбор относительно условный, т. е. решатель задач фреймворка не ограничен использованием только одного класса ИНС для решения задачи (как, например, сверточной сети для изображений, поскольку изображения можно обрабатывать и обычным многослойным персептроном). Речь, скорее, идет о предположениях, которые впоследствии проверяются. Примерами таких предположений являются:

– изображения/видео – сверхточные нейронные сети;

– графы – графовые нейронные сети;

– временные ряды – многослойные персептроны или рекуррентные сети;

– текстовая информация – многослойные персептроны или рекуррентные сети;

– наборы характеристик некоторых объектов (например, спецификации автомобилей) – многослойный персептрон.

7. *Действие формирования спецификации входов и выходов ИНС.* Выполняются дополнительные преобразования данных, связанные с изменением структур хранения (например, преобразование многомерного массива в одномерный, конвертация типов).

8. *Действие выбора метода оптимизации.* Подбирается метод оптимизации ИНС как функции. В [7] описаны следующие методы оптимизации:

– стохастический градиентный спуск (SGD);

– метод Нестерова;

– адаптивный градиент (AdaGrad);

– адаптивная оценка момента (Adam);

– среднеквадратическое распространение (RMSProp).

9. *Действие выбора минимизируемой функции ошибки.* Задается функция ошибок, которая будет минимизироваться. К примеру, MSE лучше подходит для задач регрессии и кластеризации, CE – для классификационных задач. В [7] описана классификация таких функций.

10. *Действие начальной инициализации.* Инициализируются начальные значения весовых коэффициентов и пороги ИНС. В [7] уже описана классификация методов начальной инициализации ИНС.

11. *Действие выбора гиперпараметров ИНС.* Осуществляется подбор гиперпараметров, к которым относятся:

– параметры обучения ИНС (скорость обучения, моментный параметр, размер мини-батча);

– архитектурные параметры ИНС, опирающиеся на ранее сформулированные спецификации входных и выходных данных (например, количество нейронов в определенном слое (слоях) или конфигурации целых слоев).

Найти оптимальные гиперпараметры можно, например, с помощью метода сеточного поиска, который позволяет проверить значения гиперпараметров, взятые с определенным шагом или из определенного интервала (кортежа). Посредством этого метода выбирается оптимальный набор гиперпараметров, который дает лучшие результаты, он используется для последующего дообучения. Если же полученные результаты являются приемлемыми, процесс дальнейшего обучения вообще не проводится.

12. *Действие обучения ИНС.* Производится обучение ИНС до достижения выбранной точности (оценивается на тестовой выборке) или по другим заданным критериям (достижение заданного количества эпох обучения, неизменность точности на протяжении заданного количества эпох, падение точности на валидационной выборке и т. д.). Алгоритмы обучения описаны в [5].

13. *Действие оценки эффективности ИНС.* После выполнения обучения осуществляется оценка полученной модели с помощью метрик оценки качества. В случае неудовлетворяющих результатов фреймворк меняет гиперпараметры и проводит обучение еще раз.

Интерпретация описанных действий может осуществляться на основании любого популярного фреймворка для разработки ИНС, но только как вычислительная основа. Отличительной же особенностью такого интерпретатора является наличие в единой БЗ о контексте задачи истории разработки методов решения подобных задач, описание знаний о проектировании и обучении ИНС и т. д. Следует отметить, что для автоматизации разработки методов решения нетривиальных задач интеллектуальным фреймворком потребуется подсистема, накапливающая эмпирический опыт и принимающая решение об откате до предыдущих этапов, вплоть до переформирования выборки. Однако модель такой подсистемы представляется темой отдельных исследований. Реализация интерпретатора описанных действий и описания в БЗ экспертных знаний разработчиков ИНС позволят автоматически, исходя из описания задачи, генерировать ИНС, решающие целевые задачи.

Рассмотрим работу фреймворка с реальной прикладной системой. В [8] описан принцип работы диалговой системы, использующей интеграцию базы знаний и больших языковых моделей для повышения качества удержания контекста диалога на свободную тему. Такое повышение качества происходит за счет способности системы транслировать естественноязыковые сообщения пользователя в базу знаний на SC-коде и извлекать их, когда этого требует контекст диалога. Быстрый поиск знаний, релевантных контексту диалога, стал возможен благодоря использованию онтологического подхода к структурированию знаний, в рамках которого любое знание формализуется набором понятий, подробно описанных в некоторых онтологиях. Онтологии иерархически структурированы и соответствуют предметным областям.

Основным требованием к диалоговой системе является необходимость поддерживать диалог в рамках любой предметной области, а не только в тех, которые предварительно формализованы разработчиками. К примеру, когда пользователь сообщил системе: «Я умею разрабатывать системы на C++», то она должна протранслировать эти знания, несмотря на то, что изначально в ее базе знаний не содержалось предметных областей и онтологий о программировании. Таким образом, появилась необходимость решения задачи автоматического построения онтологий. Выяснилось, что большие языковые модели при правильной настройке могут неплохо выделять смысл сообщений в виде троек: <пользователь, разрабатывать, система>, <пользователь, разрабатывать на, C++>, <C++, есть, язык программирования>, <разрабатывать, есть, навык> и т. д.

Однако для того, чтобы протранслировать такие тройки в SC-код, нужно выделить из них понятия и экземпляры. Если понятий в базе знаний еще нет, надо их классифицировать, чтобы понять, к какой онтологии их отнести и каким образом использовать для формализации сообщений пользователя в будущем. К примеру, понятие «язык программирования» является абсолютным и обозначает множество, содержащее известные системе языки программирования, а понятие «разрабатывать» – это неролевое отношение, связывающее разработчика и систему. В рассматриваемой диалоговой системе эта задача решается с помощью фреймворка по разработке ИНС.

Далее приведено выполнение действий фреймворка на примере разработки ИНС для решения рассматриваемой задачи классификации.

1. *Действие спецификации задачи.* Постановка задачи сформулирована следующим образом: классифицировать понятия онтологий на основании их семантической окрестности, т. е. на основании их связей с другими сущностями в базе знаний. Выделяются следующие целевые классы:

– абсолютное понятие;
– ролевое отношение;
– неролевое отношение.

На рис. 3 представлена формулировка спецификации рассматриваемой задачи на языке SCg [5]. В качестве БЗ использовалась мета-система OSTIS[6]. Обучающей выборкой являлось множество всех понятий БЗ с их семантическими окрестностями. Размер выборки – 93 элемента.

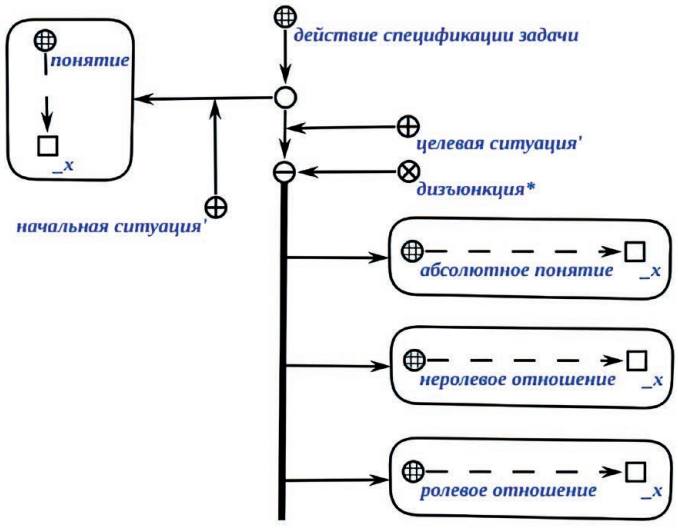

Рис. 3. Формулировка задачи классификации понятия на языке SCg
Fig. 3. Formulation of the concept classification problem in the SCg language

2. *Действие очистки выборки.* При решении рассматриваемой задачи это действие не выполнялось.

3. *Действие выявления содержательных признаков.* В том случае, когда элементы обучающей выборки представлены в виде семантической окрестности, фреймвор решает задачу выбора той части семантической окрестности понятия, которая наиболее характерна для класса данного понятия. В качестве такой части рассматривались:

– множества, в которых состоит понятие с указанием его роли;
– элементы, которые принадлежат понятию с указанием его роли;
– исходящие неролевые отношения с другими sc-элементами;
– входящие неролевые отношения с другими sc-элементами;
– различные сочетания вышеперечисленного.

Произвели несколько итераций обучения, и наилучшие результаты были получены при использовании той части семантической окрестности понятия, которая описывает входящие неролевые отношения с другими sc-элементами.

4. *Действие трансформации признаков.* В настоящее время в фреймворке не описаны ИНС, способные получать SC-код на вход, однако есть возможность обрабатывать классические гра-

[6] Intellegent Meta System. Available: https://ims.ostis.net (Accessed 22 July 2023).

фы с помощью графовых нейронных сетей. Поэтому был разработан алгоритм преобразования SC-кода в классический граф, суть которого состоит в замене инцидентности между дугами на вершину графа. На рис. 4 показан пример преобразования фрагмента SC-кода в классический граф.

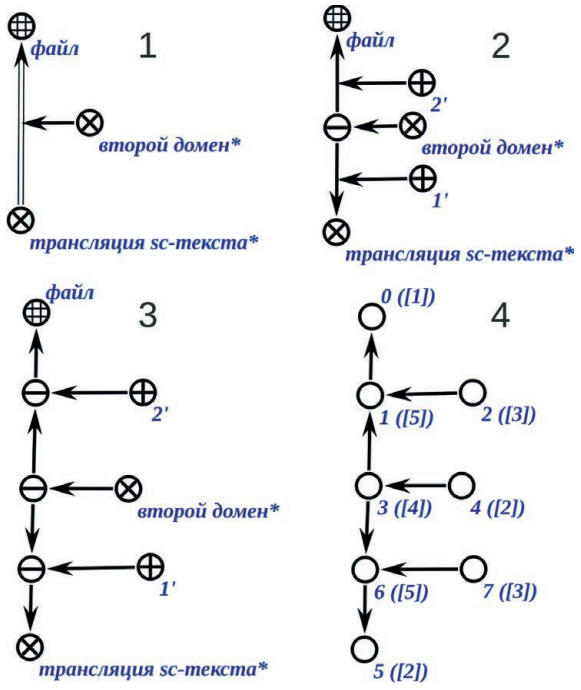

Рис. 4. Преобразование фрагмента SC-кода в классический граф
Fig. 4. Converting a fragment of SC-code into a classical graph

5. *Действие разбиения выборки.* На данный момент фреймвор не имеет специализированных правил для разбиения выборок, поэтому всегда используется разбиение 70/30 (размер обучающей выборки к размеру тестовой выборки).

6. *Действие выбора класса ИНС.* В рассматриваемой задаче тип входных данных задавал класс ИНС – графовая нейронная сеть (graph neural network, GNN). GNN – это семейство нейронных сетей, которые могут естественным образом работать с данными, структурированными на основе графа. Каждому элементу графа (вершине или ребру) ставится в соответствие некоторый эмбеддинг – вектор фиксированной размерности, характеризующий элемент графа. Каждый слой GNN преобразует эти эмбеддинги по некоторым правилам, которые учитывают эмбеддинги соседних элементов графа. Из эмбеддингов элементов графа высчитывается эмбеддинг всего графа. В фреймворке уже интегрирована реализация[7] предобученной графовой нейронной сети CapsGNN [9], которая используется в случае необходимости графовых нейронных сетей по умолчанию.

7. *Действие формирования спецификации входов и выходов ИНС.* Для SC-кода в ходе исследования в качестве начального эмбеддинга вершин получившегося графа рассматривались следующие характеристики:

– количество исходящих дуг;
– количество входящих дуг;
– расстояние до максимального класса в рамках ПрО sc-элемента;
– расстояние до понятий онтологии верхнего уровня;
– номер типа sc-элемента;
– различные сочетания вышеперечисленного.

В результате перебора наилучшим классифицирующим признаком был выбран номер типа sc-элемента, который в текущей реализации используется фреймворком по умолчанию.

[7] A PyTorch Implementation of Capsule Graph Neural Network. Available: https://github.com/benedekrozemberczki/CapsGNN (Accessed 22 July 2023).

8. *Действие выбора метода оптимизации.* Метод оптимизации фреймворком не выбирался, так как использовалась интегрированная реализация обучения CapsGNN.

9. *Действие выбора минимизируемой функции ошибки.* Минимизируемая функция ошибки не выбиралась, так как использовалась интегрированная реализация обучения CapsGNN.

10. *Действие начальной инициализации.* Поскольку взята предобученная ИНС, начальная инициализация весов не требуется.

11. *Действие выбора гиперпараметров ИНС.* Помимо параметров обучения, которые заданы интегрированной реализацией и на которые не влияет фреймворк, были заданы три значения количества эпох (100, 200, 300), по достижении которых оценивалась эффективность обучения.

12. *Действие обучения ИНС.* Интерпретация действия сводилась к вызову интегрированной реализации обучения CapsGNN.

13. *Действие оценки эффективности ИНС*. Для оценки эффективности высчитывали процент правильных классификаций.

В табл. 1 представлены результаты обучения GNN, которые можно назвать приемлемыми для решения поставленной задачи.

Таблица 1. Результаты обучения GNN
Table 1. Graph Neural Network training results

Количество эпох / Number of epochs	Размер выборки / Sample size		Среднее количество / Average quantity		Точность / Accuracy
	обучающей / teaching	тестовой / test	вершин / of peaks	дуг дуг / of arcs	
100	65	28	19	11	0,89
200	65	28	19	11	0,92
300	65	28	19	11	0,92

Заключение

1. Предложенный подход к взаимодействию интеллектуальных систем с интеллектуальным фреймворком по разработке искусственных нейронных сетей благодаря применению многократно используемых компонентов позволяет проектировать интеллектуальные системы, способные к самообучению.

2. Реализация описанных действий решателя задач интеллектуального фреймворка по разработке искусственных нейронных сетей позволит автоматически проектировать и обучать искусственные нейронные сети исходя из спецификации задачи. Однако выполнение некоторых действий требует большего количества экспертных знаний и эмпирического опыта разработки. Поэтому на начальном этапе эксплуатации предложенного фреймворка уместно рассматривать выполнение отдельных действий разработчиками с целью накопления необходимых знаний.

3. Направлением дальнейших исследованией представляется изучение процесса эксплуатации предложенного фреймворка разработчиками с целью разработки методов аккумуляции экспертных знаний и эмпирического опыта.

Список литературы

1. Attention is All You Need [Electronic Resource] / A. Vaswani [et al.]. Mode of access: https://doi.org/10.48550/arXiv.1706.03762. Date of access: 20.06.2023.
2. Training Language Models to Follow Instructions with Human Feedback [Electronic Resource] / L. Ouyang [et al.]. Mode of access: https://doi.org/10.48550/arXiv.2203.02155. Date of access: 20.06.2023.
3. Wiley, V. Computer Vision and Image Processing: a Paper Review / V. Wiley, T. Lucas // International J. of Artificial Intelligence Research. 2018. Vol. 2, No 1. P. 28–36. https://doi.org/10.29099/ijair.v2i1.42.
4. Hoy, M. B. Alexa, Siri, Cortana, and More: an Introduction to Voice Assistants / M. B. Hoy // Medical Reference Services Quarterly. 2018. Vol. 37, No 1. P. 81–88.
5. Технология комплексной поддержки жизненного цикла семантически совместимых интеллектуальных компьютерных систем нового поколения / под общ. ред. В. В. Голенкова. Минск: Бестпринт, 2023. Гл. 3.6. 1064 с.

6. Orlov, M. K. Comprehensive Library of Reusable Semantically Compatible Components of Next-Generation Intelligent Computer Systems / M. K. Orlov // Открытые семантические технологии проектирования интеллектуальных систем = Open Semantic Technologies for Intelligent Systems (OSTIS–2022): сб. науч. тр.; редкол. В. В. Голенков [и др.]. Минск: Белор. госуд. ун-т информ. и радиоэлектр, 2022. Вып. 6. С. 261–272.
7. Kovalev, M. V. Model for the Representation of Artificial Neural Networks and Actions for their Processing in the Knowledge Base / M. V. Kovalev // Открытые семантические технологии проектирования интеллектуальных систем = Open Semantic Technologies for Intelligent Systems (OSTIS–2021): сб. науч. тр.; редкол. В. В. Голенков [и др.]. Минск: Белор. госуд. ун-т информ. и радиоэлектр., 2021. Вып. 5. С. 93–100.
8. Integration of Large Language Models With Knowledge Bases of Intelligent Systems / K. Bantsevich [et al.] // Открытые семантические технологии проектирования интеллектуальных систем = Open Semantic Technologies for Intelligent Systems (OSTIS–2022): сб. науч. тр.; редкол.: В. В. Голенков [и др.]. Минск: Белор. госуд. ун-т информ. и радиоэлектр., 2023. Вып. 7. P. 213–218.
9. Xinyi, Z. Capsule Graph Neural Network [Electronic Resource] / Z. Xinyi, L. Chen // International Conference on Learning Representations, 2019. Mode of access: https://openreview.net/forum?id=Byl8BnRcYm. Date of access: 20.06.2023.

References

1. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. (2017) *Attention is All You Need*, *CoRR*. Available: https://doi.org/10.48550/arXiv.1706.03762 (Accessed 20 June 2023).
2. Ouyang L., Wu J., Jiang X., Almeida D., Wainwright C. L., Mishkin P., Zhang C., Agarwal S., Slama K., Ray A., Schulman J., Hilton J., Kelton F., Miller L. E., Simens M., Askell A., Welinder P., Christiano P. F., Leike J., Lowe R. J. (2022) *Training Language Models to Follow Instructions with Human Feedback*. Available: https://doi.org/10.48550/arXiv.2203.02155 (Accessed 20 June 2023).
3. Wiley V., Lucas T. (2018) Computer Vision and Image Processing: a Paper Review. *International Journal of Artificial Intelligence Research*. 2 (1), 28–36. https://doi.org/10.29099/ijair.v2i1.42.
4. Hoy M. B. (2018) Alexa, Siri, Cortana, and More: an Introduction to Voice Assistants. *Medical Reference Services Quarterly*. 37 (1), 81–88.
5. Golenkov V. (2023) *Technology for Comprehensive Life Cycle Support of Semantically Compatible, Next-Generation Intelligent Computer Systems*. Minsk, Bestprint. 1064 (in Russian).
6. Orlov M. K. (2022) Comprehensive Library of Reusable Semantically Compatible Components of Next-Generation Intelligent Computer Systems. *Open Semantic Technologies for Intelligent Systems*. Minsk, Belarusian State University of Informatics and Radioelectronics. 6, 261–272.
7. Kovalev M. V. (2021) Model for the Representation of Artificial Neural Networks and Actions for their Processing in the Knowledge Base. *Open Semantic Technologies for Intelligent Systems*. Minsk, Belarusian State University of Informatics and Radioelectronics. 5, 93–100.
8. Bantsevich K., Kovalev M., Tsishchanka V., Malinovskaya N., Andrushevich A. (2023) Integration of Large Language Models with Knowledge Bases of Intelligent Systems. *Open Semantic Technologies for Intelligent Systems*. Minsk, Belarusian State University of Informatics and Radioelectronics. 7, 213–218.
9. Xinyi Z., Chen L. (2019) Capsule Graph Neural Network. *International Conference on Learning Representations*. Available: https://openreview.net/forum?id=Byl8BnRcYm (Accessed 20 June 2023).

Сведения об авторе

Ковалёв М. В., ст. преподаватель кафедры интеллектуальных информационных технологий Белорусского государственного университета информатики и радиоэлектроники

Адрес для корреспонденции

220013, Республика Беларусь,
г. Минск, ул. П. Бровки, 6
Белорусский государственный университет
информатики и радиоэлектроники
Тел.: +375 29 721-60-63
E-mail: kovalev@bsuir.by
Ковалёв Михаил Владимирович

Information about the author

Kovalev M. V., Senior Lecturer at the Department of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics

Address for correspondence

220013, Republic of Belarus,
Minsk, P. Brovki St., 6
Belarusian State University
of Informatics and Radioelectronics
Tel.: +375 29 721-60-63
E-mail: kovalev@bsuir.by
Kovalev Mikhail Vladimirovich