ВЕСТНИК НАЦИОНАЛЬНОГО ИССЛЕДОВАТЕЛЬСКОГО ЯДЕРНОГО УНИВЕРСИТЕТА “МИФИ”, 2022, том 11, № 2, с. 153–163

ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

УДК 004.912

НЕЙРОСЕТЕВОЙ ИНТЕРФЕЙС КОНВЕРТАЦИИ СЛОЖНЫХ РУССКОЯЗЫЧНЫХ ТЕКСТОВЫХ КОМАНД В ФОРМАЛИЗОВАННЫЙ ГРАФОВЫЙ ВИД ДЛЯ УПРАВЛЕНИЯ РОБОТЕХНИЧЕСКИМИ УСТРОЙСТВАМИ

© 2022 г. А. Г. Сбоев[1,2,*], А. В. Грязнов[1], Р. Б. Рыбка[1], М. С. Скороходов[1], И. А. Молошников[1]

[1] НИЦ “Курчатовский институт”, Москва, 123182 Россия

[2] Национальный исследовательский ядерный университет “МИФИ”, Москва, 115409 Россия

**e-mail: sag111@mail.ru*

Поступила в редакцию 24.06.2022 г.
После доработки 25.06.2022 г.
Принята к публикации 28.06.2022 г.

Представлен нейросетевой интерфейс конвертации сложных русскоязычных текстовых команд для роботехнических устройств в RDF-формат. В составе интерфейса используются нейросетевые модели для восстановления пропущенных глаголов, разделения составных команд на единичные и разбора единичных команд. Для обучения этих моделей сформированы обучающие и тестировочные выборки, в частности: для создания набора данных для обучения разделению составных команд на единичные и анализа единичных команд был разработан генератор текстовых команд, используя который были сгенерированы по специальным шаблонам 55 тыс. простых команд и 16 тыс. составных; для задачи восстановления глаголов был использован корпус с конференции Dialog-21, содержащий 16 тыс. предложений, из которых 5 тыс. имели пропущенные глаголы; тестовое множество было собрано с помощью технологии краудсорсинга и содержит 1.3 тыс. примеров. В основе используемых методов лежат языковые модели и нейросети с архитектурой трансформер. В работе оцениваются по точности применения ресурсоемкой языковой модели (MultilingualBERT) и более экономичной по ресурсам дистиллированной версии (RuBERT-tiny). Представлены результаты влияния на точность разбора команд наличия знаков препинания и заглавных букв в тексте.

Ключевые слова: машинное обучение, анализ естественного языка, трансформеры, многозначная классификация, робототехника

DOI: 10.56304/S2304487X22020092

1. ВВЕДЕНИЕ

Проблема управления мобильными робототехническими устройствами на естественном языке в сложной среде чрезвычайно актуальна, как в различных отраслях и областях науки, так и в повседневной жизни. Разнообразие ситуаций, в которых необходимо использовать такие устройства, диктует необходимость создания точного и в то же время гибкого алгоритма распознавания команд, передаваемых пользователем на устройство с преобразованием их во внутреннее представление робота. Представление команды в понятном для робототехнического устройства виде может быть описано в форме логического представления данных или графа, отражающего семантические отношения между сущностями внутри закодированного предложения. Существует важное отличие текста естественного языка от таких формализованных представлений роботехнических систем: для последних не допускаются омонимия и перефразирование, структура данных не должна нарушаться, чтобы не допускать неоднозначность толкования. Это делает особо актуальной задачу построения системы преобразования русскоязычных текстовых команд во внутреннее представление робота в связи со свободным порядком слов в предложении. На эту тему существует ряд зарубежных работ. Например, в работе [1] представлено решение для конвертации текстовых команд на английском языке в действия агента в компьютерной игре. Ввиду ограниченности мира компьютерной игры результаты статьи не применимы к области робототехники. Другие работы [2, 3] имплементируют подходы на основе моделей машинного обучения, однако они также апробированы в специализированных условиях задачи и эмуляционных средах. В данной статье предлагается вариант реализации человеко-машинного интерфейса для преобразования текстовых команд на естественном русском языке во

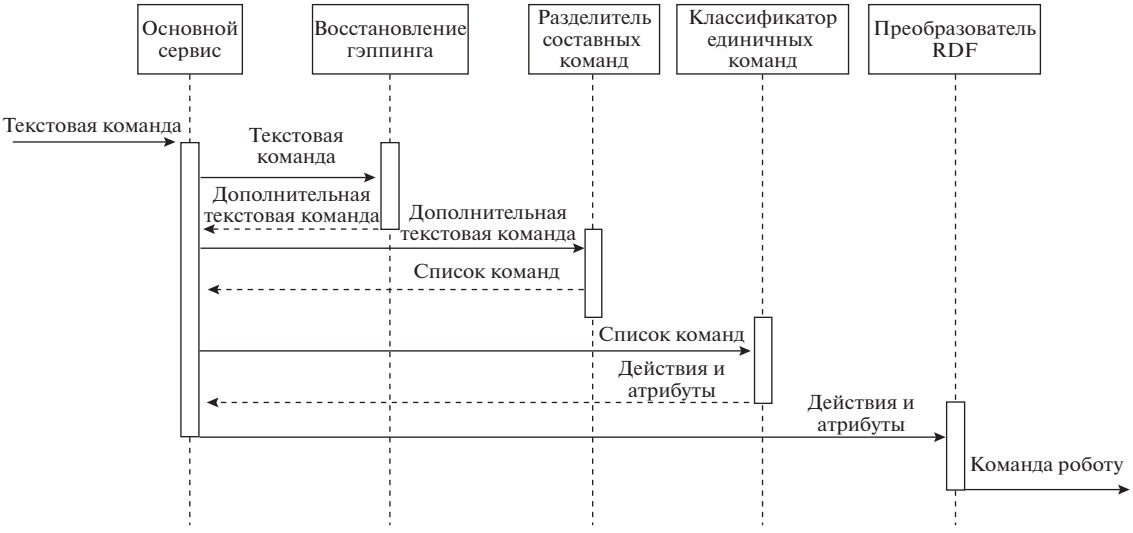

Рис. 1. Схема взаимодействия между модулями сервисом обработки команд.

внутреннее представление робота с отработкой на задаче управления агентом в мире с различными объектами. При этом используется гибкий формат формализованного представления команд, который может быть расширен новыми объектами, типами действий и т.д., не прибегая к дополнительному сбору данных. Основой подхода является комплекс нейросетевых моделей для классификации текстов. Для настройки моделей использовались как наборы данных, собранные с привлечением аннотаторов, так и синтетически-сгенерированные команды. В разделе 2 представлен используемый подход к конвертации сложного текста в формализованное представление. В разделе 3 описаны методы в составе развиваемого подхода. Далее в разделе 4 описаны используемые для обучения нейросетей корпуса примеров и этапы их создания. Результаты экспериментов по выбору нейросетевой модели конвертации и набор дополнительных инструментов для построения интерфейса представлены в разделе 5. В результате сформированный и апробированный комплекс методов расширяет набор подходов анализа персонифицированных данных на основе методов машинного обучения [4, 5] и глубоких нейронных сетей [6–8].

2. ОПИСАНИЕ ПОДХОДА КОНВЕРТАЦИИ

Рассматривается набор команд на перемещение агента к заданной точке мира нахождения агента. В общем случае команда состоит из действия и атрибутов действия. Действия можно разделить на оперативные и верхнеуровневые. Первые позволяют управлять агентом напрямую без привязки к элементам мира робота: поворот на заданное число градусов, проход вперед или назад на заданное число метров. В этом случае атрибутами действия являются количество градусов поворота, число метров и т.д. Действия второго типа используют в качестве атрибутов объекты мира, а также взаимосвязи между ними. Задаваемые оператором команды условно можно разделить на односоставные (единичные) и многосоставные (составные). В составе составных команд частым случаем является пропуск глаголов (гэппинг), например: “Езжай к лесу, а потом к озеру”. Этот текст включает две единичные команды, однако во второй части пропущен глагол “езжай”. В работе предлагается подход с разделением текста сложной команды на последовательность единичных с восстановлением пропущенных глаголов и последующим разбором. Этапы конвертации, представленные на рис. 1, включают:

1) восстановление пропущенных глаголов, отвечающих за указание действия;

2) разделение составной команды на единичные;

3) разбор единичных команд;

4) преобразование полученных разборов команд в RDF-представление.

В результате текстовая команда конвертируется в формализованный вид в формате RDF-вида “субъект” – “предикат” – “объект”.

3. ДАННЫЕ

3.1. Корпус текстов для задачи восстановления глаголов

Для обучения модели для задачи гэппинга был использован корпус с конференции Dialog-21 [9]. Он был составлен из новостных, художественных и технических текстов, а также текстов из социальных сетей. Обучающее множество содержало 16406 предложений из которых 5542 имели пропущенные глаголы. Валидационное множество содержало 4142 предложения, 1382 из которых

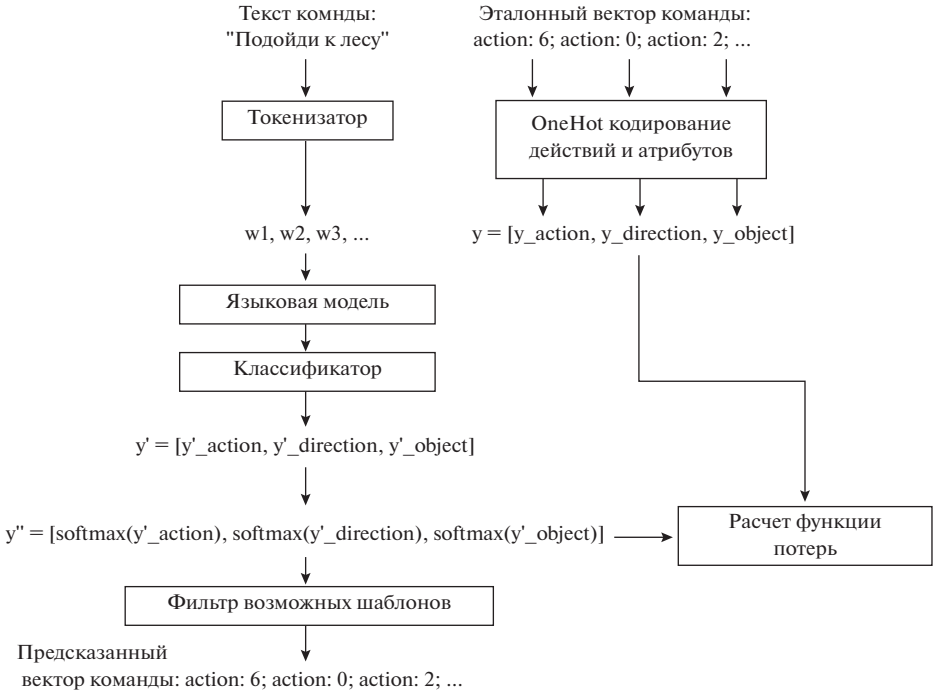

Рис. 2. Схема обучения модели классификации единичных команд.

имели пропущенные глаголы. Тестовое множество состояло из 2045 текстов, 680 с пропущенными глаголами. Также корпус включает 115563 предложения с автоматической разметкой, которая не была проверена экспертами в качестве дополнительного обучающего множества. Автоматическая разметка была получена системой Compreno [10]. Для каждого предложения имелась метка класса (0 – нет пропущенных глаголов, 1 – есть пропущенный глагол), а также размечены фрагменты текста, соответствующие ремнантам (R1, R2), коррелятам (cR1, cR2), пропущенным глаголам (cV), и местам пропуска (V).

3.2. Корпус текстовых сообщений для задачи разбора команд

Для создания набора данных для обучения нейросетевых моделей разделения составных команд на единичные и анализа единичных команд разработан генератор текстовых команд, который выполняет следующие функции:

– создает множество команд на естественном языке на основе заданных шаблонов с переменными синонимами и вариациями речевых оборотов русского языка;

– составляет вектор эталонных шаблонов действий и атрибутов для каждой сгенерированной команды на основе шаблона ее генерации.

Если в команде содержится какой-либо атрибут, его значение подставляется в соответствии со словарем, характеризующим возможные конфигурации задаваемых оператором команд (см. табл. 1). Пример векторного представления для команды “иди к дому” с атрибутами представлен в табл. 2.

Используемые для генерации текстовой команды шаблоны делятся на 2 группы: 1) в которых упоминается одна команда, т.е. единичные, и 2) составные, в которых упоминается 2–3 команды. Команды и их вектор атрибутов, сгенерированные с использованием первого набора шаблонов используются для обучения модели разбора единичной команды с последующей конвертацией в RDF-вид.

Второй набор шаблонов используется для генерации составных команд для обучения модели разделения сложной команды на простые.

Шаблоны с примерами, по которым происходит генерация единичных команд, представлены в табл. 3. Всего было определено 11 примеров на различные команды, которые поддерживаются робототехнической платформой. Каждый шаблон единичной команды описывается конкретным RDF-представлением, в котором содержатся переменные атрибуты типа команды, направления движения, типа объекта и т.д.

Для генерации по этому типу шаблонов используется набор словарей с синонимами для определенного атрибута действия, которые отличаются по составу в зависимости от используемого шаблона.

Шаблоны с примерами, по которым происходит генерация составных команд представлены в табл. 4. Составные команды представляют собой последовательное выполнение команд, описанных в табл. 3. Всего было определено 4 шаблона составных шаблонов.

Таблица 1. Словарь для составления вектора атрибутов единичной команды

Название ячейки вектора	Возможные значения
y – метка шаблона	0: простые (без атрибутов) команды 1: движение в направлении 2: движение на кол-во метров 3: движение на кол-во градусов 4: движение на кол-во часов 5: движение к объектам 6: движение к близкому объекту 7: движение с одним отношением 8: движение с двумя отношениями 9: патрулирование по кругу 10: движение к объектам относительно робота 11: движение по взгляду
act – тип действия	0: patrol, (патрулировать) 1: stop, (остановиться) 2: move_dir, (двигаться) 3: rotate_dir, (поворачивать) 4: move_on, (двигаться) 5: rotate_on, (поворачивать) 6: move_to, (двигаться) 7: find, (искать) 8: go_around, (обходить) 9: monitor, (осматривать) 10: rotate_to, (поворачивать) 11: pause, (прерваться) 12: continue (продолжать)
dir – направление	0: 0, 1: dir_forward, (вперед) 2: dir_backward, (назад) 3: dir_right, (направо) 4: dir_left, (налево) 5: dir_north, (на север) 6: dir_south, (на юг) 7: dir_east, (на восток) 8: dir_west (на запад)
m – метры	0: None, 1: 1, 2: 3, 3: 5, 4: 7, 5: 10, 6: 15, 7: 20, 8: 25, 9: 35, 10: 50, 11: 75, 12: 100
dh – часы/градусы	0: None, 1: 15, 2: 30, 3: 45 ... 24: 360
o1, o2, o3 – атрибуты объектов команды	0: 0 1: house, (дом) 2: tree, (дерево) 3: broken_tree, (сломанное дерево) 4: forest, (лес) 5: pit, (яма) 6: human, (человек) 7: hill, (холм)

Таблица 1. Окончание

Название ячейки вектора	Возможные значения
	8: fissure, (трещина) 9: man_lay, (лежащий человек) 10: rock, (камень) 11: butte, (останец) 12: barrier, (барьер) 13: lamp_post, (фонарь) 14: car (автомобиль)
n – метка близкого объекта	0: None, 1: nearest_from_type (ближайший)
r1 – отношение между o1 и o2 r2 – отношение между o2 и o3 s – направление относительно робота	0: 0, 1: near, (около) 2: behind_of, (позади) 3: in_front_of, (перед) 4: to_left_from, (слева от) 5: to_right_from (правее)
g – метка направления взгляда	0: None, 1: has_gaze_focus_on (туда, этот)
del – указание на номер в последовательности команд, если команда составная	0: 0 1: 1 2: 2

Каждый вектор с атрибутами однозначно переводится в формат команды робототехнического устройства – RDF – с использованием специального конвертера.

3.3. Данные для тестирования модели разбора единичных команд

Тестовое множество было собрано с помощью технологии краудсорсинга. Из сгенерированных команд было выбрано 250 образцовых команд, после чего пользователям краудсорсинговой платформы было дано задание перефразировать эти команды пятью разными способами. Результирующий набор содержал следующее распределение текстов команд по типам действий:

– простая команда (51 пример);
– движение в направлении (89 примеров);
– движение к объекту (53 примера);
– движение к объекту с указанием его положения относительно другого объекта (130 примеров);
– движение к объекту с указанием его положения относительно двух объектов (995 примеров).

4. МЕТОДЫ

4.1. Метод восстановления пропущенных глаголов

В общем случае, эта задача подразумевает распознавание и заполнение пропущенных слов в предложении. В данной работе для решения этой задачи был выбран метод [11], показавший лучшие результаты на соревновании Automatic gapping resolution for russian (AGRR-2019) в рамках конференции Dialogue-21 [9].

Основой метода является нейросетевая модель обработки последовательностей (seq2seq), выполняющая классификацию токенов входного текста по пяти классам: cV, cR1, cR2, R1, R2. Здесь cV – глагол (или предикат), который пропущен в следующих простых предложениях в составе сложного. cR1 и cR2 – корреляты из предложения без пропуска, которые синтаксически и семантически схожи с R1 и R2 – ремнантами из предложений с пропуском (см. пример ниже).

Пример: индекс [cR1 промышленного производства cR1] за январь-февраль 2008 г. [cV составил cV] [cR2 106.0% cR2], [R1 инвестиций в основной капитал R1] – [R2 120.2% R2] и [R1 оборота розничной торговли R1] – [R2 116.3% R2].

Местом пропуска глагола, отмеченного как cV, считается начало либо R2, либо R1 в случае, когда R2 не был обнаружен в тексте.

Таблица 2. Векторное представление команды “иди к дому” с атрибутами

y	act	dir	m	dh	o1	n	r1	o2	r2	o3	s	g	del
5	6	0	0	0	1	0	0	0	0	0	0	0	0

Таблица 3. Шаблоны и примеры команд генератора для единичных команд

№	Описание шаблона и типов команд	Пример сгенерированных текстов команд	RDF-представление для примеров команд (для первой команды из поля примеров)
0	Простые (без атрибутов) команды: patrol, stop, pause, continue	стой, патрулируй, продолжай	<KI: F0> <ki: type> <ki: action> . <KI: F0> <ki: actor> <ki: me> . <KI: F0> <ki: action_type> <ki: stop> .
1	Движение в направлении: move_dir, rotate_dir	двигайся вперед, направляйся на север, двигайся назад, поворачивай налево	<KI: F0> <ki: type> <ki: action> . <KI: F0> <ki: actor> <ki: me> . <KI: F0> <ki: action_type> <ki: move_dir> . <KI: F0> <ki: patient> <KI: F0_V0> . <KI: F0_V0> <ki: type> <ki: direction> . <KI: F0_V0> <ki: value> <ki: dir_forward> .
2	Движение на заданное число метров: move_on	двигайся вперед на 5 метров	<KI: F0> <ki: type> <ki: action> . <KI: F0> <ki: actor> <ki: me> . <KI: F0> <ki: action_type> <ki: move_on> . <KI: F0> <ki: patient> <KI: F0_V0> . <KI: F0_V0> <ki: type> <ki: direction> . <KI: F0_V0> <ki: value> <ki: dir_forward> . <KI: F0_V0> <ki: numeric_value> “2” .
3	Поворот на заданное число градусов: rotate_on	разворачивайся, поворачивай направо на 30 градусов	<KI: F0> <ki: type> <ki: action> . <KI: F0> <ki: actor> <ki: me> . <KI: F0> <ki: action_type> <ki: rotate_on> . <KI: F0> <ki: patient> <KI: F0_V0> . <KI: F0_V0> <ki: type> <ki: direction> . <KI: F0_V0> <ki: value> <ki: dir_right> . <KI: F0_V0> <ki: numeric_value> “180” .
4	Поворот по “часам”: rotate_on	поверни на 6 часов, поверни на 6:00	<KI: F0> <ki: type> <ki: action> . <KI: F0> <ki: actor> <ki: me> . <KI: F0> <ki: action_type> <ki: rotate_on> . <KI: F0> <ki: patient> <KI: F0_V0> . <KI: F0_V0> <ki: type> <ki: direction> . <KI: F0_V0> <ki: value> <ki: dir_right> . <KI: F0_V0> <ki: numeric_value> “180” .
5	Движение к объектам: move_to, rotate_to, find, go_around, monitor	иди к дому, найди человека, обойди дерево, осмотри камень, повернись к барьеру	<KI: F0> <ki: type> <ki: action> . <KI: F0> <ki: actor> <ki: me> . <KI: F0> <ki: action_type> <ki: move_to> . <KI: F0> <ki: patient> <KI: F0_V0> . <KI: F0_V0> <ki: type> <ki: object> . <KI: F0_V0> <ki: type> <ki: house> .
6	Движение к ближайшему объекту: move_to, rotate_to, find, go_around, monitor	подойди к ближайшему дому, отыщи близкий дом, обойди ближайший камень, осмотри близкое строение, повернись к ближнему человеку	<KI: F0> <ki: type> <ki: action> . <KI: F0> <ki: actor> <ki: me> . <KI: F0> <ki: action_type> <ki: move_to> . <KI: F0> <ki: patient> <KI: F0_V0> . <KI: F0_V0> <ki: type> <ki: object> . <KI: F0_V0> <ki: type> <ki: house> . <KI: F0_V0> <ki: nearest_from_type> <ki: house> .
7	Движение с одним отношением между объектами: move_to, rotate_to, find, go_around, monitor	направляйся к человеку около дома; за деревом находится человек, к которому надо подойти	<KI: F0> <ki: type> <ki: action> . <KI: F0> <ki: actor> <ki: me> . <KI: F0> <ki: action_type> <ki: move_to> . <KI: F0> <ki: patient> <KI: F0_V0> . <KI: F0_V0> <ki: type> <ki: object> . <KI: F0_V0> <ki: type> <ki: house> . <KI: F0_V0> <ki: near> <KI: F0_V1> . <KI: F0_V1> <ki: type> <ki: object> . <KI: F0_V1> <ki: type> <ki: human> .

Таблица 3. Окончание

№	Описание шаблона и типов команд	Пример сгенерированных текстов команд	RDF-представление для примеров команд (для первой команды из поля примеров)
8	Движение с двумя отношениями между объектами: move_to, rotate_to, find, go_around, monitor	осмотри дом, который рядом с человеком, неподалеку от дерева	<KI: F0> <ki: type> <ki: action> . <KI: F0> <ki: actor> <ki: me> . <KI: F0> <ki: action_type> <ki: move_to> . <KI: F0> <ki: patient> <KI: F0_V0> . <KI: F0_V0> <ki: type> <ki: object> . <KI: F0_V0> <ki: type> <ki: house> . <KI: F0_V0> <ki: near> <KI: F0_V1> . <KI: F0_V1> <ki: type> <ki: object> . <KI: F0_V1> <ki: type> <ki: rock> . <KI: F0_V1> <ki: near> <KI: F0_V2> . <KI: F0_V2> <ki: type> <ki: object> . <KI: F0_V2> <ki: type> <ki: tree> .
9	Патрулирование по кругу: patrol	патрулируй по кругу с радиусом 10 метров	<KI: F0> <ki: type> <ki: action> . <KI: F0> <ki: actor> <ki: me> . <KI: F0> <ki: action_type> <ki: patrol> . <KI: F0> <ki: patient> <KI: F0_V0> . <KI: F0_V0> <ki: type> <ki: circle> . <KI: F0_V0> <ki: numeric_value> “10” .
10	Движение к объектам, располагающимся относительно робота: move_to, rotate_to, find, go_around, monitor	двигайся к дереву справа от тебя, найди человека слева	<KI: F0> <ki: type> <ki: action> . <KI: F0> <ki: actor> <ki: me> . <KI: F0> <ki: action_type> <ki: move_to> . <KI: F0> <ki: patient> <KI: F0_V0> . <KI: F0_V0> <ki: type> <ki: object> . <KI: F0_V0> <ki: type> <ki: house> . <KI: F0_V0> <ki: in_front_of> <ki: me> .
11	Движение по направлению взгляда: move_to, rotate_to, go_around, monitor	подойди к тому дереву; иди туда; двигайся туда, к дому	<KI: F0> <ki: type> <ki: action> . <KI: F0> <ki: actor> <ki: me> . <KI: F0> <ki: action_type> <ki: move_to> . <KI: F0> <ki: patient> <KI: F0_V0> . <KI: F0_V0> <ki: type> <ki: object> . <ki: me> <ki: has_gaze_focus_on> <KI: F0_V0> . <KI: F0_V0> <ki: type> <ki: tree> .

*курсивом выделены переменные атрибуты;

**жирным шрифтом специфические постоянные предикаты для конкретного шаблона.

Таблица 4. Шаблоны и примеры команд генератора для составных команд

Описание шаблона составной команды	Пример генератора
2 команды, разделенные ключевыми словами	двигайся к дому, потом иди к дереву; поверни направо на 3 часа, после этого иди к человеку слева от тебя
3 команды разделенные ключевыми словами	направляйся к человеку около дома, далее развернись, наконец найди ближайшее дерево
Команды типа “найди–иди”	Найди дерево. За деревом располагается человек, к которому надо подойти. Найди дом. За ним находится человек, к которому требуется подойти.
2 команды со связью при помощи местоимения	Найди дом и обходи его

4.2. Метод разделения составной команды на единичные

Разделение составных команд на единичные осуществляется на основе классификатора токенов. В качестве классификатора используется языковая нейросетевая модель, которая для каждого токена определяет классы:

– O, который означает, что токен не относится ни к одной из команд;

– [SEP] – означает, что токен является частью текущей команды;

– [CMD] – означает, что токен относится к текущей команде, но все последующие токены отмеченные как [CMD] или [SEP] относятся к следующей команде.

4.3. Метод разбора единичной команды

Обработка текста единичной команды выполняется на основе классификационной модели, позволяющей определить одновременно несколько классов, отражающих: класс действия и классы соответствующих действию атрибутов. Для каждой команды формируется вектор из 0 и 1, где первые 13 компонент соответствуют типам действий (см. табл. 1), а следующие компоненты соответствуют атрибутам действия и классам этих атрибутов. Команды подаются в классификатор на базе языковой модели, задача которого определить этот вектор. В качестве функции потерь используется кросс-энтропия.

Для исключения несовместимых между собой наборов атрибутов и действий реализована процедура сопоставления кортежей атрибутов и действий. Если, отбирая классы действий и их атрибутов получается комбинация, которой не было в обучающей выборке, такой кортеж игнорируется и выбирается следующий по суммарной активности модели на каждый выходной класс.

4.4. Языковые модели

Основой используемых методов являются языковые модели на основе нейросетей архитектуры трансформер. Современные высокоточные языковые модели достаточно ресурсоемки и для их использования в сервисах онлайн-обработки требуются значительные вычислительные мощности. В связи с этим в данной работе исследовались применения ресурсоемкой языковой модели (MultilingualBERT) и облегченной-дистиллированной версии (RuBERT-tiny).

MultilingualBERT – это нейросетевая языковая модель глубокого обучения на основе слоев multi head attention, представлена в работе [12]. Обучение проводилось на корпусе дампов Википедии для 104 языков. При формировании корпуса, производилось сглаживание по представительности текстов для каждого языка. Модель состоит из 12 трансформер блоков, размерность скрытого слоя 768 (есть еще размерность около 3000), 12 голов в multi head attention.

RuBERT-tiny [13] архитектурно соответствует модели BERT со следующими изменениями: размер входного словаря уменьшен с 119 тыс. до 30 тыс. наиболее часто встречающихся слов (токенов) в русском и английском языках, размер слоя векторного представления слов уменьшен с 768 до 312, число слоев трансформер уменьшено с 12 до 3. Модель получена в результате процедуры обучения с использованием выходов уже обученных нейросетевых моделей большего размера, в т.ч.: RuBERT [14], LaBSE [15], Laser [16] и USE [17]. В качестве обучающих данных были использованы следующие корпуса: Yandex Translate (https://translate.yandex.ru/corpus), OPUS-100 [18] и Tatoeba [19]. Количество параметров для RuBERT-tiny модели, рассчитанное по оценке, представленной для RuBERT, составляет 11.5 M (L = 3, H_m = 312; H_ff = 600; V = 29 тыс.).

RuBERT-tiny2 – является улучшенным вариантом RuBERT-tiny. Имеет больший размер словаря (83828 токенов вместо 29564); поддерживает последовательности большей длины (2048 вместо 512); для обучения модели использованы преимущественно тексты на русском языке; проведено дообучение на задаче логического следования предложений [20].

5. ЭКСПЕРИМЕНТЫ

Конвертация речевых сообщений команд на естественном языке осложняется отсутствием в текстах знаков пунктуации и заглавных букв, что особенно наблюдается при получении текста из сервисов распознавания речи. В связи с этим в проводимых экспериментах исследовалась работа моделей для текстов такого вида.

5.1. Точности метода восстановления глаголов

По метрике f1, используемой на соревновании Automatic gapping resolution for russian (AGRR-2019), оценка точности решения задачи восстановления глаголов при использовании модели MultilingualBERT составляет: 0.90. Точность после замены языковой модели MultilingualBERT на RuBERT-tiny2 составляет 0.69. При условии отсутствия в анализируемом тексте знаков пунктуации и заглавных букв точности снижаются до 0.77 и 0.4 соответственно.

5.2. Точность метода разделения составных команд

После обучения RuBERT-tiny2 в течение 10 эпох была достигнута 99.9% точность. Из 3033 валидационных примеров, была ошибка определения границ команд только в 7 случаях. В связи с высокой точностью этой модели и ее значительно

Таблица 5. Процент правильно предсказанных команд при использовании языковой модели RuBERT-tiny2

Эксперимент	Фолды					
	0	1	2	3	4	Среднее
Первая постановка с сохранением пунктуации и заглавных букв	77.6	68.9	79.9	67.7	47.5	68.3
Вторая постановка с сохранением пунктуации и заглавных букв	81.8	89.8	86.7	84.8	76.1	83.8
Вторая постановка. Тексты в нижнем регистре и без пунктуации	81.8	90.2	85.9	84.8	78.3	84.2
Вторая постановка. Проверка на совместимость атрибутов в составе вектора.	82.6	90.9	87.1	85.6	78.7	84.9
Вторая постановка. Тексты в нижнем регистре и без пунктуации. Увеличение эпох обучения с 10 до 20	83.3	91.3	87.5	85.9	80.2	85.6

более низкой сложности по сравнению с остальными, дальнейшие эксперименты не проводились.

5.3. Точность метода разбора единичных команд

В рамках экспериментов тестовое множество было разделено на 5 подчастей. При проведении экспериментов 4 подчасти объединялись и добавлялись к обучающей выборке, полученной с использованием разработанного генератора текстов единичных команд. Оставшаяся часть использовалась для тестирования.

Эксперименты проводились в двух постановках:

1. модель обучается предсказывать весь вектор команды;

2. модель предсказывает весь вектор, но выходные значения нормализуются функцией softmax внутри каждого атрибута и внутри действия. Значения кросс энтропии считается для них также отдельно и итоговая функция потерь является суммой функции потерь для действия и атрибутов.

В табл. 5 представлены точности определения векторов команд с учетом сложных условий.

Как видно из табл. 5 лучшие результаты достигаются во второй постановке эксперимента с увеличением числа эпох и добавлением проверки на совместимость атрибутов в составе результирующего вектора классов.

6. СЕРВИС

Сервис обработки команд состоит из четырех сервисов.

Сервис гэппинга. Принимает http post запрос с текстом для анализа. Текст токенизируется, токены размечаются классами cV, cR1, cR2, R1, R2. Если есть последовательности токенов, помеченные как R2, тогда токен, помеченный как cV вставляется перед началом этих последовательностей, иначе он вставляется перед началом последовательностей, размеченных классом R1. После детокенизации получившаяся строка возвращается ответом на post запрос.

Сервис разделения составных команд. Принимает http post запрос с текстом для анализа. Текст токенизируется и размечается классами [SEP], [CMD], O. С учетом разделения по токенам, отмеченным классом [SEP], из токенов, отмеченных как [CMD] или [SEP] составляются отдельные команды. Ответом на post-запрос будет список из единичных команд, на которые была разделена входная составная команда.

Сервис разбора единичных команд. Принимает http post-запрос со списком текстов команд для анализа. Подает этот список в классификатор, который определяет действие, указанное в команде и его атрибуты. Ответом на Post-запрос является список, в котором каждой команде соответствует список пар: первой парой всегда будет ключевое слово “action” и класс действия, которое было определено в команде, последующие пары – названия атрибутов и их классы, если атрибут был определен как активный.

Внешний сервис. Является внешним интерфейсом и переадресует принимаемые команды рабочим сервисам обработки. Принимает http post с текстом команд, который переадресуется последовательно сервисам обработки. Ответом внешнего сервиса является разбор всех единичных команд в составе входного текста. Общее время обработки занимает ~0.2 секунды при использовании модели RuBERT-tiny2.

7. ЗАКЛЮЧЕНИЕ

В результате проведенного комплекса работ создан программный комплекс разбора сложных команд роботехническому устройству с переводом их в RDF-формат, позволяющий:

– восстанавливать пропущенные в команде глаголы (достигаемая точность до 90%), однако зависит от качества текста команды (наличия знаков препинания и заглавных букв и т.д.);

– разбирать составные команды на единичные (достигаемая точность до 99.9%);

– классификация единичных команд в формализованное представление (достигаемая точность до 86%).

В перспективе планируется провести исследования по сокращению временных затрат сервиса на разбор команды без существенного снижения точности.

БЛАГОДАРНОСТИ

Работа выполнена при поддержке НИЦ "Курчатовский институт" (приказ № 2754 от 28.10.2021).

СПИСОК ЛИТЕРАТУРЫ

1. *Gray J., Srinet K., Jernite Y., Yu H., Chen Z., Guo D., Szlam A.* Craftassist: A framework for dialogue-enabled interactive agents // arXiv preprint arXiv: 1907.08584, 2019.
2. *Adiwardana D., Luong M.T., So D.R., Hall J., Fiedel N., Thoppilan R., Le Q.V.* Towards a human-like open-domain chatbot // arXiv preprint arXiv: 2001.09977, 2020.
3. *Dong L., Lapata M.* Language to logical form with neural attention // arXiv preprint arXiv: 1601.01280, 2016.
4. *Сбоев А.Г., Гудовских Д.В., Молошников И.А., Рыбка Р.Б.* Определение пола автора текста в коллекции русских многожанровых текстов с искажениями с использованием моделей машинного обучения // Вестник НИЯУ МИФИ, 2018. Т. 7 № 6. С. 531–536.
5. *Сбоев А.Г., Молошников И.А., Рыбка Р.Б., Наумов А.В.* Интерпретация результатов модели определения типа имитации возраста в тексте // Вестник НИЯУ МИФИ, 2020. Т. 9. № 2. С. 189–196.
6. *Молошников И.А., Грязнов А.В., Власов Д.С., Рыбка Р.Б., Сбоев А.Г.* Применение мультизадачной модели для практических задач генерации заголовка, определение лемм и ключевых слов // Вестник НИЯУ МИФИ, 2020. Т. 9 № 3. С. 236–244.
7. *Сбоев А.Г., Рыбка Р.Б., Грязнов А.В., Молошников И.А.* Генеративно-дискриминативная нейросетевая модель для задачи авторского профилирования // Вестник НИЯУ МИФИ, 2020. Т. 9. № 1. С. 50–57.
8. *Сбоев А.Г., Селиванов А.А., Рыбка Р.Б., Молошников И.А., Богачев Д.С.* Модель нейронной сети для включения синтаксической структуры предложения в задачу классификации пола автора русского текста // Вестник НИЯУ МИФИ. 2019. Т. 8. № 6. С. 569–576.
9. *Smurov I.M., Ponomareva M., Shavrina T.O., Droganova K.* Agrr-2019: Automatic gapping resolution for Russian // Computational Linguistics and Intellectual Technologies. 2019. С. 561–575.
10. *Anisimovich K.V., Druzhkin K.Ju., Minlos F.R., Petrova M.A., Selegey V.P., Zuev K.A.* Syntactic and semantic parser based on ABBYY Compreno linguistic technologies // Komp'iuternaia Lingvistika i Intellektual'nye Tehnologii: Trudy Mezhdunarodnoj Konferentsii "Dialog", 2012. P. 91–103.
11. *Belkin I.* BERT finetuning and graph modeling for gapping resolution // Komp'juternaja Lingvistika i Intellektual'nye Tehnologii, 2019. P. 63–71.
12. *Devlin J., Chang M.W., Lee K., Toutanova K.* Bert: Pre-training of deep bidirectional transformers for language understanding // arXiv preprint arXiv: 1810.04805, 2018.
13. *Дейл Д.* Маленький и быстрый БЕРТ для русского языка. [Электронный ресурс]. https://habr.com/ru/post/562064/. (дата обращения 24.06.2022)
14. *Kuratov Y., Arkhipov M.* Adaptation of deep bidirectional multilingual transformers for russian language // arXiv preprint arXiv: 1905.07213., 2019.
15. *Feng F., Yang Y., Cer D., Arivazhagan N., Wang W.* Language-agnostic bert sentence embedding // arXiv preprint arXiv: 2007.01852, 2020.
16. *Artetxe M., Schwenk H.* Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond // Transactions of the Association for Computational Linguistics, 2019. V. 7. P. 597–610.
17. *Cer D., Yang Y., Kong S.Y., Hua N., Limtiaco N., John R.S., Kurzweil R.* Universal sentence encoder // arXiv preprint arXiv: 1803.11175, 2018.
18. *Zhang B., Williams P., Titov I., Sennrich R.* Improving massively multilingual neural machine translation and zero-shot translation // arXiv preprint arXiv: 2004. 11867, 2020.
19. *Tiedemann J.* Parallel data, tools and interfaces in OPUS // Lrec., 2012. С. 2214–2218.
20. *Williams A., Nangia N., Bowman S.R.* A broad-coverage challenge corpus for sentence understanding through inference // arXiv preprint arXiv: 1704.05426., 2017.

Vestnik Natsional'nogo Issledovatel'skogo Yadernogo Universiteta "MIFI", 2022, vol. 11, no. 2, pp. 153–163

Neural Network Interface for Converting Complex Russian-Language Text Commands into a Formalized Graph to Control Robotic Devices

A. G. Sboev[a,b,#], A. V. Gryaznov[a], R. B. Rybka[a], M. S. Skorokhodov[a], and I. A. Moloshnikov[a]

[a] *National Research Center Kurchatov Institute, Moscow, 123182 Russia*

[b] *National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Moscow, 115409 Russia*

[#] *e-mail: sag111@mail.ru*

Received June 24, 2022; revised June 25, 2022; accepted June 28, 2022

Abstract—A neural network interface for converting complex Russian-language text commands for robotic devices into the RDF-format is presented. The interface involves neural network models to recover missing

verbs, split compound commands into single ones, and parse single commands. In order to train these models, training and testing samples have been formed: a data set for learning to divide compound commands into single commands and to analyze single commands has been created with the help of a specially-developed text command generator, using which 55000 simple commands and 16000 composite commands have been generated using special templates; for the task of recovering missing verbs, the corpus from the Dialog-21 conference is used, containing 16000 sentences, of which 5000 have missing verbs; the test set is assembled using crowdsourcing technology and contains 1300 examples. The methods used for text analysis are based on language models and neural networks with Transformer architecture. The accuracy of using a resource-intensive language model (MultilingualBERT) and the resource-efficient distilled version RuBERT-tiny of RuBERT has been evaluated. The results of the dependence of the command parsing accuracy on the presence of punctuation marks and capital letters in the text are presented.

Keywords: machine learning, natural language analysis, transformers, multi-label classification, robotics

DOI: 10.56304/S2304487X22020092

REFERENCES

1. Gray J., Srinet K., Jernite Y., Yu H., Chen Z., Guo D., Szlam A. Craftassist: A framework for dialogue-enabled interactive agents. *arXiv preprint arXiv: 1907.08584*, 2019.
2. Adiwardana D., Luong M.T., So D.R., Hall J., Fiedel N., Thoppilan R., Le Q.V. Towards a human-like open-domain chatbot. *arXiv preprint arXiv: 2001.09977.,* 2020.
3. Dong L., Lapata M. Language to logical form with neural attention. *arXiv preprint arXiv: 1601.01280*, 2016.
4. Sboev A.G., Gudovskih D.V., Moloshnikov I.A., Rybka R.B. Opredelenie pola avtora teksta v kollekcii russkih mnogozhanrovyh tekstov s iskazheniyami s ispol'zovaniem modelej mashinnogo obucheniya. *Vestnik NIYaU MIFI*, 2018, vol. 7, no. 6, pp. 531–536. (in Russian)
5. Sboev A.G., Moloshnikov I.A., Rybka R.B., Naumov A.V. Interpretaciya rezul'tatov modeli opredeleniya tipa imitacii vozrasta v tekste. [Interpretation of the results of the model for determining the type of age simulation in the text]. *Vestnik NIYaU MIFI*, 2020, vol. 9, no. 2, pp. 189–196. (in Russian)
6. Moloshnikov I.A., Gryaznov A.V., Vlasov D.S., Rybka R.B., Sboev A.G. Primenenie mul'tizadachnoj modeli dlya prakticheskih zadach generacii zagolovka, opredeleniya lemm i klyuchevyh slov [Application of the multitasking model for practical tasks of title generation, definition of lemmas and keywords]. *Vestnik NIYaU MIFI*, 2020, vol. 9, no. 3, pp. 236–244. (in Russian)
7. Sboev A.G., Rybka R.B., Gryaznov A.V., Moloshnikov I.A. Generativno-diskriminativnaya nejrosetevaya model' dlya zadachi avtorskogo profilirovaniya [Generative-discriminative neural network model for the problem of author's profiling]. *Vestnik NIYaU MIFI.* 2020, vol. 9, no. 1, pp. 50–57. (in Russian)
8. Sboev A.G., Selivanov A.A., Rybka R.B., Moloshnikov I.A., Bogachev D.S. Model' nejronnoj seti dlya vklyucheniya sintaksicheskoj struktury predlozheniya v zadachu klassifikacii pola avtora russkogo teksta. [A neural network model for including the syntactic structure of a sentence in the task of classifying the gender of the author of a Russian text]. *Vestnik NIYaU MIFI*, 2019, vol. 8, no. 6, pp. 569–576. (in Russian)
9. Smurov I.M., Ponomareva M., Shavrina T.O., Droganova K. Agrr-2019: Automatic gapping resolution for russian. *Computational Linguistics and Intellectual Technologies*, 2019, pp. 561–575.
10. Anisimovich K.V., Druzhkin K.Ju., Minlos F.R., Petrova M.A., Selegey V.P., Zuev K.A. Syntactic and semantic parser based on ABBYY Compreno linguistic technologies. *Komp'iuternaia Lingvistika i Intellektual'nye Tehnologii: Trudy Mezhdunarodnoj Konferentsii "Dialog"*, 2012, pp. 91–103.
11. Belkin I. BERT finetuning and graph modeling for gapping resolution. *Komp'juternaja Lingvistika i Intellektual'nye Tehnologii*, 2019, pp. 63–71.
12. Devlin J., Chang M.W., Lee K., Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv: 1810.04805*, 2018.
13. Dale D. *Malen'kij i bystryj BERT dlya russkogo yazyka* [Small and fast BERT for Russian]. Available at: https://habr.com/ru/post/562064/. (accessed 24.06.2022).
14. Kuratov Y., Arkhipov M. Adaptation of deep bidirectional multilingual transformers for russian language. *arXiv preprint arXiv: 1905.07213,* 2019.
15. Feng F., Yang Y., Cer D., Arivazhagan N., Wang W. Language-agnostic bert sentence embedding. *arXiv preprint arXiv: 2007.01852,* 2020.
16. Artetxe M., Schwenk H. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 2019, vol. 7, pp. 597–610.
17. Cer D., Yang Y., Kong S.Y., Hua N., Limtiaco N., John R.S., Kurzweil R. Universal sentence encoder. *arXiv preprint arXiv: 1803.11175,* 2018.
18. Zhang B., Williams P., Titov I., Sennrich R. Improving massively multilingual neural machine translation and zero-shot translation. *arXiv preprint arXiv: 2004.11867,* 2020.
19. Tiedemann J. Parallel data, tools and interfaces in OPUS. *Lrec.*, 2012, pp. 2214–2218.
20. Williams A., Nangia N., Bowman S.R. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv: 1704.05426*, 2017.