Универсальность естественно-языкового интерфейса для взаимодействия с различными типами программных интерфейсов приложений

И.В. Евченко

***Аннотация* — В данной статье проанализирована универсальность естественно-языкового интерфейса для взаимодействия с различными типами программных интерфейсов приложений. Рассмотрены подходы к разработке семантической модели, позволяющей адаптировать интерфейс, и алгоритмы автоматического преобразования естественно-языковых запросов в вызовы программных интерфейсов приложений. Предложен метод, сочетающий современные технологии обработки естественного языка и традиционные подходы, что позволяет обеспечить высокую точность и полноту обработки запросов. Проведено экспериментальное исследование, результаты которого подтвердили универсальность разработанного интерфейса и его способность адаптироваться к различным сценариям взаимодействия. На основе полученных результатов можно утверждать, что предложенный подход обеспечивает повышение точности и удобства работы с API для пользователей, не обладающих технической подготовкой, а также может быть применён в различных отраслях, требующих автоматизации взаимодействия с информационными системами.**

***Ключевые слова* — Обработка естественного языка, естественно-языковой интерфейс, API.**

I. Введение

Современные информационные системы всё чаще основываются на использовании программных интерфейсов приложений (от англ. Application Programming Interface, API) для обеспечения взаимодействия между приложениями, сервисами и пользователями. Согласно отчету о состоянии API [1] наиболее используемыми архитектурами являются REST (от англ. Representational State Transfer) [2], SOAP (от англ. Simple Object Access Protocol) [3], GraphQL (от англ. Graph Query Language) [4] и gRPC (от англ. Remote Procedure Call) [5], каждый из которых обладает уникальными характеристиками, определяющими их применимость в различных сценариях.

Однако разнообразие API создаёт сложности для пользователей, особенно тех, кто не обладает технической подготовкой.

Статья получена 20 января 2025

Игорь Владимирович Евченко, Московский инженерно-физический институт (национальный исследовательский ядерный университет), (e-mail: t.foreli@ya.ru).

Также каждый тип API требует специфических знаний, что усложняет процесс взаимодействия и интеграции. Указанные трудности подчеркивают необходимость разработки более интуитивных способов взаимодействия, таких как естественно-языковые интерфейсы, которые позволяют преобразовывать пользовательские запросы в вызовы API, упрощая доступ к данным и операциям. Однако, несмотря на существование различных решений, большинство из них либо адаптированы для конкретного типа API, либо требуют значительных усилий по ручной настройке.

Создание универсального интерфейса, способного работать с API различных типов, остаётся актуальной задачей. В данной работе рассматриваются подходы к решению этой проблемы, направленные на повышение удобства взаимодействия с API и уменьшение сложности их использования.

II. Обзор существующих работ

Для успешной обработки естественного языка важно учитывать: универсальные аспекты, присутствующие во многих языках, и уникальные характеристики, присущие конкретным языковым системам. Для этого требуется разработка гибких методов, способных адаптироваться к различным языковым контекстам и обеспечивать высокую точность интерпретации текстов на естественном языке. Такой подход особенно важен при создании естественно-языковых интерфейсов, где необходимо учитывать специфические требования различных прикладных API.

Анализ существующих исследований показывает, что в работе [7] описаны подходы к созданию естественно-языкового интерфейса для командной строки Bash. Методы демонстрируют высокую точность преобразования естественно-языковых запросов в команды, но их применение ограничено спецификой командной оболочки Bash и затруднено при переносе на другие типы API.

В статье [6] представлен подход, направленный на улучшение генерализации моделей, используемых для преобразования запросов на естественном языке в вызовы веб-API. Авторы предложили методы, обеспечивающие гибкость и адаптивность интерфейсов для эффективного взаимодействия с различными Web

API. Основное внимание уделяется способности моделей эффективно обобщать знания, полученные на простых примерах, для обработки более сложных и комплексных запросов. В работе используется архитектура на основе трансформеров, таких как модель T5. Трансформеры позволяют учитывать контекст и взаимосвязи в последовательностях слов, что позволяет преобразовывать сложные запросы на естественном языке в структурированные API-запросы.

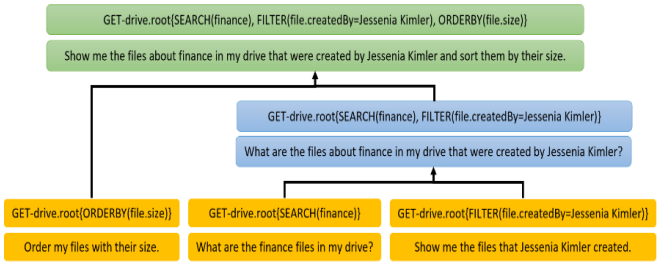

Рисунок 1. Пример комбинация вызовов API-запросов

Однако, несмотря на успехи, у работы есть ряд ограничений. Модели на основе T5, требуют значительных вычислительных ресурсов и объемов данных для эффективного обучения. Предложенный подход может привести к трудностям при обработке запросов, которые содержат неоднозначные или сильно контекстуально зависимые элементы, где требуется более тонкая настройка и обработка запросов пользователя.

Аналогично, исследование [8] фокусируется на разработке интерфейсов для взаимодействия с Web API, однако основное внимание уделяется английскому языку. По этой причине можно выделить определённые трудности при адаптации предложенных методов для русскоязычных пользователей, что ограничивает их универсальность.

В работе, посвященной методу автоматизированного формирования семантической модели базы данных диалоговой системы, представлены способы преобразования запросов. [9] Несмотря на то, что исследования были направлены на разработку методов для работы с базами данных, они могут быть адаптированы для взаимодействия с API.

Обзор существующих работ показал, что современные технологии обработки естественного языка, включая трансформеры, позволяют значительно улучшить понимание текстовых запросов. В свою очередь предложенный в данной статье комбинированный подход, объединяющий предобученные языковые модели и правила семантической интерпретации, позволяет минимизировать ограничения традиционных методов, основанных на шаблонах, и предоставляет гибкость при обработке запросов.

III. Методы обработки естественного языка

Предлагаемый подход основан на комбинированном применении методов обработки естественного языка и семантической модели, обеспечивающей универсальность интерфейса для различных типов API (REST, SOAP, GraphQL, gRPC), включающий в себя несколько этапов.

На первом этапе обрабатывается пользовательский запрос, введённый на естественном языке, путём: токенизации, лемматизации и извлечения сущностей. Текст запроса разбивается на отдельные лексемы (слова, знаки пунктуации и т. д.). Затем слова приводятся к их начальной (словарной) форме. Для русского языка применяется связка фреймворков spaCy и pymorphy2, которая позволяет повысить точность обработки с учётом русского языка. Затем происходит распознавания в тексте дат, имён собственных, географических названий, организаций и других значимых элементов.

В ходе синтаксического разбора строится дерево зависимостей, отражающее грамматические и семантические связи между словами, что необходимо для выделения ключевых сущностей и определения параметров запроса.

Данный этап формирует основу для метода комбинированного извлечения и интерпретации сущностей. В отличие от классических шаблонных решений, система может динамически адаптироваться к вариативной структуре предложений и терминологии, так как использует предобученные языковые модели.

Следующим шагом является построение семантической модели исходного текста, выделения типа действия (например, «получить», «добавить», «обновить»), целевых сущностей («поставщики холодильного оборудования», «список клиентов» и прочее) и их атрибутов («email», «телефон» и т.д.), условий или фильтров (например, «за последние 7 дней», «в конкретном регионе» и т. п.).

На этом этапе каждое понятие и действие сопоставляются с элементами, определёнными в семантической модели API. Для этой цели формируется промежуточное К-представление (концептуальное представление) [10], где указывается, какие именно параметры и методы API потенциально соответствуют тому или иному отрезку запроса.

В автоматизированном формировании семантической модели задействованы не только результаты морфологического и синтаксического анализа, но и обучающиеся методы для определения связей между сущностями.

В частности, семантическая модель описывает соответствие идентификатора категории определенному термину, что упрощает сопоставление данных. Кроме того, модель включает таблицы проекций, которые содержат информацию о синонимах и правилах преобразования терминов естественного языка в параметры и сущности API.

На третьем этапе используется сформированное К-представление для генерации запроса к API. Применение К-представления обеспечивает преобразование естественно-языковых запросов пользователя в универсальную форму, независимую от синтаксиса или операторов конкретного API.

Такая универсализация позволяет использовать К-

представление для формирования вызовов к различным типам API, включая REST, SOAP, GraphQL и gRPC, вне зависимости от их структурных особенностей.

Таблица 1. Пример таблицы проекций и терминов для запроса по поставщикам холодильного оборудования

Термины естественного языка	**Термины инфологической модели API**	**Термины даталогической модели API**
почта	контактная_ информация	contact.list + fields=email
поставщики	список_ поставщиков	contact.list
холодильное оборудование	категория_ оборудования	contact.list + category= "Холодильное оборудование"
Отношения между терминами		
P_{syn}*(поставщик,контакт)* P_{syn}*(холодильное оборудование, категория оборудования)* P_{link}*(supplier,category,category_id)* P_{rel}*(Поставщик,связан с,Категория)*		

Например, для запроса «Почта всех поставщиков холодильного оборудования» промежуточное К-представление запроса можно представить как объект с парой (ключ, значение), при этом запрос может быть представлен, как:

*Запрашиваемый объект(запрос1, все поставщики(Элемент, S1), Описание1 (категория * (Элемент, S1) : y1, категория (y1, Холодильное оборудование)))* (1)

В текущей реализации основной упор сделан на REST, однако благодаря модульной архитектуре система способна формировать вызовы SOAP, GraphQL и gRPC при подключении соответствующих шаблонов.

На основе выделенных атрибутов система определяет, какой именно метод API следует вызвать. В свою очередь для REST-запросов формируются HTTP-методы (GET, POST, PUT, DELETE) и URL с соответствующими параметрами.

GET /contact.list?category=Холодильное оборудование&fields=email (2)

При необходимости (например, POST или PUT) в тело запроса добавляются параметры в формате JSON или form-data.

Поддержка разных типов API осуществляется следующим образом, для формата SOAP генерируются XML-сообщения на основе К-представления. GraphQL – формируется структура GraphQL-запроса (полей, фильтров и вложенных объектов). gRPC – при наличии описания сервисов (protobuf) генерируются соответствующие RPC-вызовы.

На заключительном этапе результат выполнения запроса к API поступает в модуль формирования ответа. В этом модуле полученные данные преобразуются в удобочитаемый формат, учитывая контекст первоначального запроса пользователя, и отображаются в интерфейсе системы.

Таким образом, разработанная система обеспечивает эффективную обработку естественно-языковых запросов и их преобразование в корректные запросы к API, используя современные методы обработки естественного языка и структурированного представления данных.

Благодаря модульности и гибкости архитектуры, система может быть легко адаптирована для работы с различными типами API и расширена дополнительными функциональными возможностями в соответствии с требованиями пользователей и специфики предметной области.

IV. Результаты

Экспериментальные исследования проводились для проверки универсальности предложенного естественно-языкового интерфейса на примере четырех типов API: REST, SOAP, GraphQL и gRPC. Основной целью эксперимента было подтверждение точности, полноты и сбалансированности работы интерфейса (F-меры), а также демонстрация его адаптации к запросам разной сложности.

Эксперимент был организован следующим образом: пользовательский текст запроса проходил синтаксический и семантический анализ, сопоставлялся с семантической моделью API, преобразовывался в структурированный запрос и отправлялся в API. Результаты запросов представлялись пользователю.

Оценивались точность (Pr), полнота (Re) и F-мера для каждого типа API.

Типы запросов: запросы разделены на 4 категории сложности:

К1: простые запросы (выбор одного параметра).

К2: запросы с объединением данных.

К3: запросы с фильтрацией и вычислениями.

К4: сложные запросы с вложенными условиями.

Экспериментальные данные представлены в диаграммах, отражающих процент релевантных ответов, точность, полноты и F-меры интерфейсов по каждому типу API.

Проведённое исследование продемонстрировало различия в точности работы интерфейсов API различных типов при выполнении запросов, варьирующихся по степени сложности.

Наиболее высокий уровень точности наблюдается у REST, который сохраняет высокий показатель точности для всех категорий сложности запросов. В свою очередь SOAP показатели точности демонстрируют более значительное снижение с увеличением сложности запросов, где значение точности при выполнении сложных запросов К4 составляет 70,59%. GraphQL показывает аналогичное поведение. При изначальном уровне точности К1, наблюдается снижение, однако выполнении К4 на 2,74% выше, чем у предыдущего типа. Уровень обработки сложных запросов для gRPC меньше, чем у GraphQL.

$$Pr = \frac{|Drel \cup Dretr|}{|Dretr|} \quad (3)$$

При исследовании полноты выявлено преимущество REST-интерфейса, что объясняется его способностью обрабатывать сложные запросы с минимальными потерями точности, что связано с его устойчивой архитектурой и оптимальной поддержкой взаимодействия с клиентом.

$$\text{Re} = \frac{|Drel \cup Dretr|}{|Drel|} \quad (4)$$

Таблица 2. Полнота, точность и F-мера разработанного естественно-языкового интерфейса для типов API

Тип API	Категория	# запросов	|Dretr|	|Drel ∩ Dretr|	Pr	Re	F-мера
REST	K1	5	5	5	100,00%	100,00%	100,00%
	K2	27	27	26	96,30%	96,30%	96,30%
	K3	40	39	38	97,44%	95,00%	96,21%
	K4	20	19	18	94,74%	90,00%	92,34%
	Всего	92	90	87	96,67%	94,57%	95,61%
SOAP	K1	5	5	5	100,00%	100,00%	100,00%
	K2	27	25	22	88,00%	81,48%	84,68%
	K3	40	35	30	85,71%	75,00%	80,18%
	K4	20	17	12	70,59%	60,00%	65,08%
	Всего	92	82	69	84,15%	75,00%	79,44%
GraphQL	K1	5	5	5	100,00%	100,00%	100,00%
	K2	27	26	22	84,62%	81,48%	83,03%
	K3	40	32	26	81,25%	65,00%	72,67%
	K4	20	15	11	73,33%	55,00%	63,51%
	Всего	92	78	64	82,05%	69,57%	75,55%
gRPC	K1	5	5	5	100,00%	100,00%	100,00%
	K2	27	23	19	82,61%	70,37%	76,24%
	K3	40	30	23	76,67%	57,50%	66,40%
	K4	20	15	10	66,67%	50,00%	57,74%
	Всего	92	73	57	78,08%	61,96%	69,55%

SOAP и GraphQL, хотя и показывают приемлемые результаты на более простых уровнях запросов (81,48% и выше), теряют полноту при увеличении сложности, что ограничивает их применение в сценариях с высокими требованиями к обработке данных.

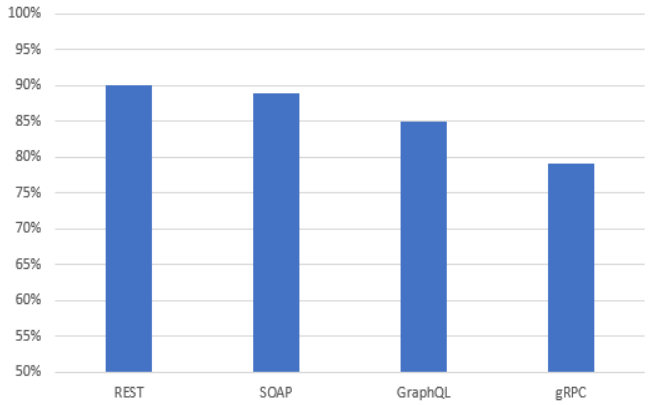

Рисунок 2. Средний процент релевантных ответов для различных типов API

F-мера, отражающая сбалансированность точности и полноты работы интерфейсов API, для REST-интерфейсы показала наивысший уровень и составила 92,34%.

$$\text{F} = \frac{2 * Pr * Re}{Pr + Re} \quad (5)$$

В свою очередь GraphQL продемонстрировал сходные показатели с SOAP (65,08%), но при этом F-мера для самых сложных запросов составила 63,51%, что на 1,57% ниже, чем предыдущий тип. gRPC показал подобные двум предыдущим типам показатели по F-мере на самом сложном уровне запросов, достигнув 57,74%, что на 5,77% ниже, чем у GraphQL.

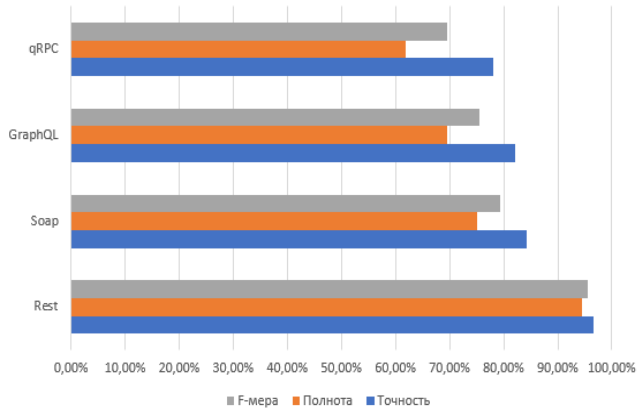

Рисунок 3. Сравнительный анализ точности, полноты и F-меры интерфейсов API разных типов

V. Заключение

В данной статье исследована универсальность естественно-языкового интерфейса для работы с различными типами API. Проведённый анализ и эксперименты подтвердили эффективность разработанного подхода, основанного на использовании семантической модели и методов обработки естественного языка.

Полученные данные открывают перспективы для

дальнейших исследований в данной области. В частности, выявленные закономерности снижения точности, полноты и F-меры для SOAP, GraphQL и gRPC при увеличении сложности запросов могут быть использованы для разработки новых методов оптимизации работы данных интерфейсов.

Эксперимент подтвердил возможность достигнуть универсальности естественно-языкового интерфейса для других типов API благодаря семантической модели. Алгоритм преобразования текста пользователя в API-запрос обеспечивает точность и полноту без ручной настройки для каждого API.

Полученные результаты подтверждают значимость предложенного решения для развития технологий взаимодействия с API и демонстрируют его потенциал для дальнейшей адаптации и внедрения в различные системы.

Библиография

[1] Postman. State of the API Report 2023. [Online]. Available: https://voyager.postman.com/pdf/2023-state-of-the-api-report-postman.pdf. 2023.

[2] Neumann A., Laranjeiro N., Bernardino J. An Analysis of Public REST Web Service APIs. IEEE Transactions on Services Computing. P.957-970. 2021.

[3] Ranga V., Soni A. API Features Individualizing of Web Services: REST and SOAP. International Journal of Innovative Technology and Exploring Engineering. 2019.

[4] Quiña-Mera A., Fernandez P., Garcia J., Ruiz-Cortés A. GraphQL: A Systematic Mapping Study. ACM Comput. Surv.55. 35 p. 2023.

[5] Bolanowski M., Żak K., Paszkiewicz A., Ganzha M., Paprzycki M., Sowiński P., Lacalle Ú., Carlos I., Carlos P. Eficiency of REST and gRPC realizing communication tasks in microservice-based ecosystems. 2022.

[6] Hosseini, S., Awadallah, A.H., Su, Y. Compositional generalization for natural language interfaces to web apis. arXiv, 2021.

[7] Lin, X.V., Wang, C., Zettlemoyer, L. and Ernst, M.D. Nl2bash: A corpus and semantic parser for natural language interface to the linux operating system. arXiv. 2018.

[8] Su Y., Hassan A., Khabsa M., Pantel P., Gamon M., Encarnacion M. Building Natural Language Interfaces to Web APIs. // In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17). Association for Computing Machinery, New York, NY, USA, P.177-186. 2017.

[9] Посевкин Р.В. Метод автоматизированного формирования семантической модели базы данных диалоговой системы // Программные продукты и системы. № 2. С. 291-294. 2018.

[10] Razorenov A.A., Fomichev V.A. A New Approach to Formalization of Instuctions` Semantic Processing Based on the Theory of K-representations // Information Technologies. Vol. 23, No. 1. P. 3-14. 2017.

Universality of a natural language interface for interaction with different types of application programming interfaces

Igor V. Evchenko

***Abstract* — This article examines the universality of a natural language interface for interaction with various types of application programming interfaces (APIs). It explores approaches to developing a semantic model that adapts the interface and algorithms for the automatic transformation of natural language queries into API calls. A method is proposed that combines modern natural language processing technologies with traditional approaches, ensuring high accuracy and completeness in query processing. An experimental study was conducted, and the results confirmed the universality of the developed interface and its ability to adapt to different interaction scenarios. Based on the findings, it can be concluded that the proposed approach enhances the accuracy and usability of APIs for users without technical expertise and can be applied in various fields that require automated interaction with information systems.**

***Keywords* — Natural language processing, natural language interface, API.**

REFERENCES

[1] Postman. State of the API Report 2023. [Online]. Available: https://voyager.postman.com/pdf/2023-state-of-the-api-report-postman.pdf. 2023.

[2] Neumann A., Laranjeiro N., Bernardino J. An Analysis of Public REST Web Service APIs. IEEE Transactions on Services Computing. P.957-970. 2021.

[3] Ranga V., Soni A. API Features Individualizing of Web Services: REST and SOAP. International Journal of Innovative Technology and Exploring Engineering. 2019.

[4] Quiña-Mera A., Fernandez P., Garcia J., Ruiz-Cortés A. GraphQL: A Systematic Mapping Study. ACM Comput. Surv.55. 35 p. 2023.

[5] Bolanowski M., Żak K., Paszkiewicz A., Ganzha M., Paprzycki M., Sowiński P., Lacalle Ú., Carlos I., Carlos P. Eficiency of REST and gRPC realizing communication tasks in microservice-based ecosystems. 2022.

[6] Hosseini, S., Awadallah, A.H., Su, Y. Compositional generalization for natural language interfaces to web apis. arXiv, 2021.

[7] Lin, X.V., Wang, C., Zettlemoyer, L. and Ernst, M.D. Nl2bash: A corpus and semantic parser for natural language interface to the linux operating system. arXiv. 2018.

[8] Su Y., Hassan A., Khabsa M., Pantel P., Gamon M., Encarnacion M. Building Natural Language Interfaces to Web APIs. // In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17). Association for Computing Machinery, New York, NY, USA, P.177-186. 2017.

[9] Posevkin R.V. Method of automated formation of a semantic model of a dialog system database // Software products and systems. No. 2. P. 291-294. 2018.

[10] Razorenov A.A., Fomichev V.A. A New Approach to Formalization of Instuctions` Semantic Processing Based on the Theory of K-representations // Information Technologies. Vol. 23, No. 1. P. 3-14. 2017.