НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

На правах рукописи

ШПАКОВ ВЛАДИСЛАВ КОНСТАНТИНОВИЧ

**ГИБРИДНАЯ ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА УПРАВЛЕНИЯ МОБИЛЬНЫМ РОБОТОМ**

Научный руководитель:

к.т.н., доцент кафедры ВТ Гаврилов А.В.

. Новосибирск 2020

Содержание

[Введение 3](#_Toc39843346)

[1. Анализ существующих систем с естественно-языковым интерфейсом 6](#_Toc39843347)

[1.1. Естественный язык в ЭВМ 6](#_Toc39843348)

[1.2. Основные подходы к обработке естественного языка 10](#_Toc39843349)

[1.3. Обзор систем с ЕЯ-интерфейсом 15](#_Toc39843350)

[2. Разработка интерпретатора естественного языка 36](#_Toc39843351)

[2.1. Эмбединг слов 36](#_Toc39843352)

[2.2. Архитектура используемой нейронной сети 38](#_Toc39843353)

[2.3. Архитектура ГИСУМР и гибридный подход к решению задач 41](#_Toc39843354)

[3. Разработка программной системы, реализующей ГИСУМР 45](#_Toc39843355)

[3.1. Общая информация о Unreal Engine 4 45](#_Toc39843356)

[3.2. Построение тренировочной выборки для нейронной сети интерпретатора 46](#_Toc39843357)

[3.3. Анализ обучения моделей LSTM в различных индексациях словаря 49](#_Toc39843358)

[3.4. Итеграция ГИСУМР в Unreal Engine 4 50](#_Toc39843359)

[4. Проведение экспериментов в среде моделирования Unreal Engine 4 54](#_Toc39843360)

[Заключение 55](#_Toc39843361)

[Список литературы 56](#_Toc39843362)

# Введение

***Актуальность***

Одной из наиболее важных проблем в робототехнике является создание системы управления роботом, обеспечивающей общение с роботом на естественном языке со всеми его особенностями, главными из которых являются неоднозначность и зависимость от контекста.

В последнее время все чаще применяется подход создания интеллектуального диалогового интерфейса, с помощью которого человек может давать команды роботу на естественном языке(ЕЯ). Он основан на применении логической обработки данных классическими методами инженерии знаний и ассоциативной обработки данных нейронными сетями []. Для этого необходимо закодировать все слова, используемые в человеко-машинном взаимодействии, таким образом, чтобы они обладали семантическими связями с другими словами по контексту диалога. При использовании данной системы в робототехнике, контекст диалога должен распознаваться по предыдущему диалогу человеко-машинного взаимодействия, в сочетании с данными, поступаемыми с сенсоров робота.

***Целью магистерской диссертации*** является разработка гибридной интеллектуальной системы управления мобильным роботом(ГИСУМР), которая бы понимала ЕЯ, была способна обрабатывать русский язык, и использовала гибридные подходы к решению задач. Основные задачи, которые должны быть решены, связаны, прежде всего, с интерпретатором ЕЯ, в терминологии понятные системе, и разработка сценариев поведения робота. Система должна на вход получать речевые команды, переведенные в текст, и принимать решение какую последовательность действий необходимо выполнить. Все результаты должны быть протестированы в виртуальной среде моделирования.

Для достижения поставленной цели в рамках магистерской диссертации решаются следующие задачи:

1. Разработка архитектуры ГИСУМР, ориентированной на использование ЕЯ;
2. Разработка интерпретатора ограниченного ЕЯ для управления роботом;
3. Разработка подсистемы сценариев поведения робота, обеспечивающей представление сценариев и процесс обучения системы новым сценариям;
4. Разработка подсистемы управления приводами для реализации сценариев;
5. Разработка программной модели робота
6. Проведение экспериментов с целью проверки и оценки реализованных методов на виртуальной модели робота в среде моделирования Unreal Engine 4.

***Объектом исследования*** являетсяпрограммная система управления мобильным роботом.

***Предметом исследования*** является обработка естественного языка, основанная на комбинировании методов машинного обучения и шаблонных алгоритмов.

***Научная новизна*** полученных результатов заключается в разработке контекстно-зависимой гибридной системы управления мобильным роботом, ориентированной на использование речевых команд на ЕЯ

***Практическая значимость*** работы заключается в следующем:

* Предложенные методы и разработанные алгоритмы могут стать основой интерфейса для персональных роботов, выполняющих роль помощников для людей с ограниченными возможностями.
* Предложенные методы могут быть использованы в проектировании иных программных систем с естественно-языковым интерфейсом или моделей искусственного интеллекта в видеоиграх.
* Разработанное ПО планируется использовать при постановке лабораторной работы по дисциплине «Интеллектуальные системы и технологии»
* Разработанные алгоритмы и ПО планируется использовать для дальнейшей научно-исследовательской работы по использованию нейронных сетей и ЕЯ в робототехники, в частности, с использованием кафедрального робота-манипулятора Дельта 5Х.

# Анализ существующих систем с естественно-языковым интерфейсом

## Естественный язык в ЭВМ

Задача исследований ествественного языка - создание компьютерно-эффективных моделей коммуникации на ЕЯ. Именно такая постановка задачи отличает NLP (от анг. «Natural Language Processing» - обработка естественного языка) от задач традиционной лингвистики и других дисциплин, изучающих ЕЯ, и позволяет отнести ее к области ИИ. Проблемой NLP занимаются две дисциплины: компьютерная лингвистика и когнитивная психология.

Традиционно лингвисты занимались созданием формальных, общих, структурных моделей ЕЯ, и поэтому отдавали предпочтение тем из них, которые позволяли извлекать как можно больше языковых закономерностей и делать обобщения. Практически никакого внимания не уделялось вопросу о пригодности моделей с точки зрения компьютерной эффективности их применения. Таким образом, оказалось, что лингвистические модели, характеризуя собственно язык, не рассматривали механизмы его порождения и распознавания. Хорошим примером тому служит порождающая грамматика Хомского, которая оказалась абсолютно непригодной на практике в качестве основы для компьютерного распознавания ЕЯ.

Задачей же когнитивной психологии является моделирование не структуры языка, а его использования. Специалисты в этой области также не придавали большого значения вопросу о компьютерной эффективности.

Различаются общая и прикладная NLP. Задачей общей NLP является разработка моделей использования языка человеком, являющихся при этом компьютерно-эффективными. Основой для этого является общее понимание текстов, как это подразумевается в работах Чарняка, Шенка, Карбонелла и др. Несомненно, общая NLP требует огромных знаний о реальном мире, и большая часть работ сосредоточена на представлении таких знаний и их применении при распознавании поступающего сообщения на ЕЯ. На сегодняшний день ИИ еще не достиг того уровня развития, когда для решения подобных задач в большом объеме использовались бы знания о реальном мире, и существующие системы можно называть лишь экспериментальными, поскольку они работают с ограниченным количеством тщательно отобранных шаблонов на ЕЯ.

Прикладная NLP занимается обычно не моделированием, а непосредственно возможностью коммуникации человека с ЭВМ на ЕЯ. В этом случае не так важно, как введенная фраза будет понята с точки зрения знаний о реальном мире, а важно извлечение информации о том, чем и как ЭВМ может быть полезной пользователю (примером может служить интерфейс экспертных систем). Кроме понимания ЕЯ, в таких системах важно также и распознавание ошибок и их коррекция.

* + 1. ***Проблема обработки ЕЯ в англоязычных системах***

Основной проблемой NLP является языковая неоднозначность. Существуют разные виды неоднозначности:

* Синтаксическая (структурная) неоднозначность: во фразе Time flies like an arrow для ЭВМ неясно, идет ли речь о времени, которое летит, или о насекомых, т.е. является ли слово flies глаголом или существительным.
* Смысловая неоднозначность: во фразе The man went to the bank to get some money and jumped in слово bank может означать как банк, так и берег.
* Падежная неоднозначность: предлог in в предложениях He ran the mile in four minutes/He ran the mile in the Olympics обозначает либо время, либо место, т.е. представлены совершенно различные отношения.
* Референциальная неоднозначность: для системы, не обладающей знаниями о реальном мире, будет затруднительно определить, с каким словом - table или cake - соотносится местоимение it во фразе I took the cake from the table and ate it.
* Литерация (Literalness): в диалоге Can you open the door? — I feel cold ни просьба, ни ответ выражены нестандартным способом. В других обстоятельствах на вопрос может быть получен прямой ответ yes/no, но в данном случае в вопросе имплицитно выражена просьба открыть дверь.

Центральная проблема как для общей, так и для прикладной NLP - разрешение такого рода неоднозначностей - решается с помощью перевода внешнего представления на ЕЯ в некую внутреннюю структуру. Для общей NLP такое превращение требует набора знаний о реальном мире. Так, для анализа фразы Jack took the bread from the supermarket shelf, paid for it, and left и для корректного ответа на такие вопросы, как What did Jack pay for?, What did Jack leave? и Did Jack have the bread with him when he left? необходимы знания о супермаркетах, процессах покупки и продажи и некоторые другие.

Прикладные системы NLP имеют преимущество перед общими, т.к. работают в узких предметных областях. К примеру, системе, используемой продавцами в магазинах по продаже компьютеров, не нужно ”раздумывать” над неоднозначностью слова terminals в вопросе How many terminals are there in the order?.

Тем не менее, создание систем, имеющих возможность общения на ЕЯ в широких областях, возможно, хотя пока результаты далеки от удовлетворительных.

* + 1. ***Проблемы обработки ЕЯ в русском языке***

Качество понимания зависит от множества факторов: от языка, от национальной культуры, от самого собеседника и т. д. Вот некоторые примеры сложностей, с которыми сталкиваются системы понимания текстов.

* Предложения «Мы отдали бананы обезьянам, потому что они были голодные» и «Мы отдали бананы обезьянам, потому что они были перезрелыми» похожи по синтаксической структуре. В одном из них местоимение они относится к обезьянам, а в другом — к бананам. Правильное понимание зависит от знаний компьютера, какими могут быть бананы и обезьяны. По нормам русского языка второе предложение некорректно, потому что в нем местоимение ссылается не на последнее подходящее слово, однако в живой речи такое предложение очень даже может встретиться.
* Свободный порядок слов может привести к совершенно иному толкованию фразы: «Бытие определяет сознание» — кто кого определяет?
* В русском языке свободный порядок компенсируется развитой морфологией, служебными словами и знаками препинания, но в большинстве случаев для компьютера это представляет дополнительную проблему.
* В речи могут встретиться неологизмы, например, глагол «Пятидесятирублируй» — то есть высылай 50 рублей. Система должна уметь отличать такие случаи от опечаток и правильно их понимать.
* Правильное понимание омонимов — ещё одна проблема. При распознавании речи, помимо прочих, возникает проблема фонетических омонимов. Во фразе «Серый волк в глухом лесу встретил рыжую лису» выделенные слова слышатся одинаково, и без знания, кто глухой, а кто рыжий, не обойтись (Кроме того, что лиса может быть рыжей, а лес — глухим, лес также может быть рыжим (характеристика, в данном случае обозначающая преобладающий цвет листвы в лесу), в то время как лиса может быть глухой, что порождает дополнительную проблему, вытекающую из предыдущей.)

## Основные подходы к обработке естественного языка

* + 1. ***Распознавание речи***

В задачи большинства NLP систем входит распознавание речи, которая решается восстановлением по звуковому сигналу слов естественного языка (из ограниченного словаря), произнесением которого является этот звуковой сигнал. Обычно задаются эталоны слов словаря для последующего сравнения звуковых сигналов с этими эталонами. Звуковой сигнал представляет из себя целочисленный вектор значений звукового давления, измеренного в равноотстоящие друг от друга моменты времени.

Известно, что речь включает в себя несколько видов информации. Основной вид информации – семантическая, которая передает содержание сообщения, его смысл. Однако данным видом информации роль речи в общении людей не исчерпывается: большое значение имеет просодия – эмоциональная окраска речи (интонация). Просодия включает информацию о физическом и эмоциональном состоянии говорящего (ирония, шутка и т.п.). И, наконец, речь, как и любой акустический звук, содержит информацию о положении источника звука в пространстве.

На рисунке 1 представлена классическая функциональная схема системы распознавания речи, состоящая из следующих функциональных узлов:

– микрофон;

– блок обработки;

– блок анализа.

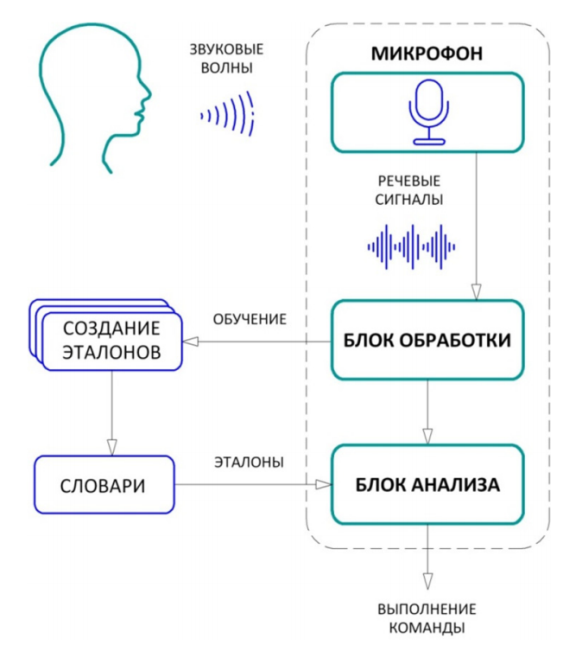


Рисунок 1. Классическая функциональная схема системы распознавания речи

Важно отметить, что с учетом программного и аппаратного обеспечения внутреннее устройство блоков системы распознавания речи гораздо сложнее.

* + 1. ***Классификация распознавателей речи***

Для того чтобы дать полную оценку современному состоянию систем распознавания речи, автором была представлена классификация систем по следующим основным параметрам [1,5].

*Техническое исполнение.* Все системы распознавания речи по техническому исполнению можно разделить на программные продукты и программно-аппаратные средства. Первые реализуются в виде программного обеспечения, инсталлируемого на компьютеризированную технику, вторые представляют собой законченное устройство.

*Назначение.* От назначения системы зависит «глубина распознавания». Под термином «глубина распознавания» условно подразумевается качество, которое в зависимости от назначения системы можно изменять с помощью подключения или отключения дополнительных средств (фонетический анализ, лексический анализ, грамматический анализ и др.). По этому параметру все системы распознавания подразделяются на три вида:

– командные системы;

– системы диктовки;

– системы распознавания.

*Персонализация*. Под этим параметром скрывается зависимость (или независимость) системы распознавания от диктора. Все системы распознавания речи делятся на дикторозависимые и дикторонезависимые системы. Первые предназначены для работы только с одним пользователем (система обучена для одного человека), вторые предназначены для работы с любым пользователем.

*Тип речи.* Речь пользователя условно можно разделить на слитную и раздельную. Если слова в речи разделены между собой участками тишины, то такая речь считается раздельной. К слитной речи относятся естественно произнесенные предложения. По типу речи системы распознавания подразделяются на системы, распознающие раздельную речь, и системы, распознающие слитную речь

*Размер словаря.* Под размером словаря систем распознавания понимается количество слов, которое система может распознать. Чем больше размер словаря, тем больше вероятность появления ошибок при распознавании и, соответственно, наоборот. Например, словарь, состоящий из десяти слов, может быть распознан практически без ошибок, тогда как в словаре из десяти тысяч слов вероятность ошибки достигает 45 %. Таким образом, системы распознавания делятся на системы с ограниченным словарем и системы со словарем большого размера.

*Тип структурной единицы.* При распознавании речи в качестве структурной единицы могут быть выбраны отдельные слова или части произнесенных слов, такие как фонемы, аллофоны, дифоны и трифоны. Системы, использующие при распознавании слова целиком или фразы, называются системами распознавания по эталону. Создание таких систем менее трудоемко в отличие от систем, распознающих минимальные структурные единицы речи: фонемы, аллофоны, дифоны и трифоны. Таким образом, системы распознавания по типу структурной единицы делятся на системы распознавания по эталону и системы распознавания по структурной единице.

*Принцип выделения*. В настоящее время в системах распознавания речи используются несколько методов выделения из речи структурных единиц:

– анализ Фурье;

– кепстральный анализ;

– вейвлет-анализ.

Механизм функционирования. В современных системах распознавания широко используются различные подходы к механизму функционирования, среди которых наибольшую популярность получили следующие: – скрытые марковские модели; – динамическое программирование; – нейросетевой метод; – экспертные системы; – простейшие детекторы. Обобщив все вышеперечисленное, можно представить классификацию систем распознавания речи (рисунок 2).

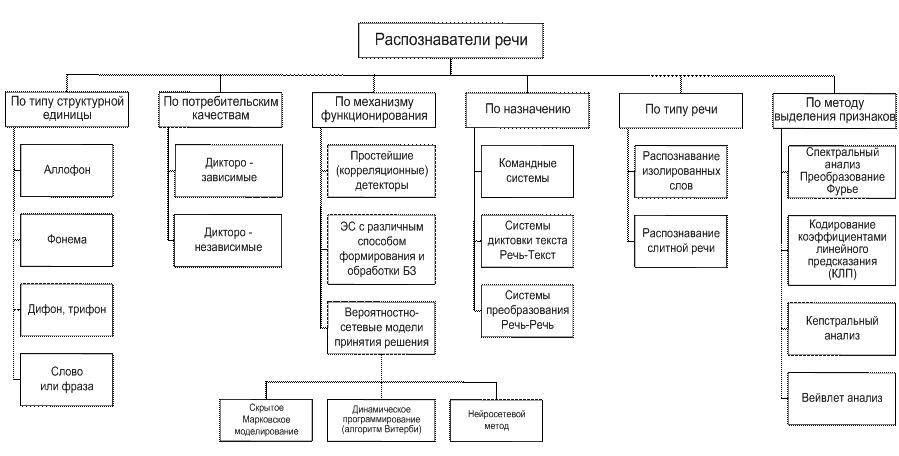


Рисунок 2. Классификация систем распознавания

* + 1. ***Извлечение семантики из естественного языка***

После кодирования входной команды требуется извлечь ее семантику.

Извлечение семантики из текстов включает в себя технологии, направленные на извлечение смысловых единиц, фактов, атрибутов, концепций и событий для того, чтобы на выходе аналитик получил структурированный набор данных, с которым можно работать при помощи обычных методов data mining. Или, например, для совершенствования систем машинного перевода и искусственного интеллекта.

Прежде, чем говорить о частных алгоритмах извлечения семантики из текстов, хотелось бы подчеркнуть, что в компьютерной лингвистике сложилась ситуация, когда алгоритмизация некой методологии неизбежно идет по одной из трех возможных ветвей:

*Алгоритмы, основанные на правилах (шаблонные алгоритмы).*В этом случае подход предполагает использование заранее подготовленных словарей (в виде структурированных наборов данных – чаще всего, таблиц), которые описывают все возможные варианты использования той или иной лексической единицы в тексте;

*Алгоритмы, основанные на статистике* (с применением машинного обучения). Эти подходы предполагают статистический анализ текста – и на основе вероятности появления той или иной лексической единицы в том или ином контексте подсказывают, куда «определить» ее в конкретном случае.

*Смешанные алгоритмы***.** Подходы, в которых используются как статистические методологии, так и словари. Считается, что такие алгоритмы лучше всего работают в случае четко определенной предметной области данного конкретного набора данных (то есть, текста).

Очевидно, что задача извлечения семантики из текстов получила в свое распоряжение все те же три вида алгоритмов. Правда, различия в них проявляются только на определенном шаге – предлагаем рассмотреть общую последовательность семантического анализа текста:

* Полученный на входе текст сегментируется (разделяется) на предложения. Каждое из предложений затем проходит процедуру синтаксического анализа – определяются члены предложения и части речи, после чего начинается извлечение семантики из каждого слова. Здесь алгоритм может сверяться с правилами, либо применять машинное обучение – а возможно, появятся признаки и того, и другого подхода; в общем и целом, результатом первого шага будет служить некий вывод синтаксического процессора по поводу того, какое значение каждое слово может принимать в данном случае. Например, для смешанного (и считающегося наиболее совершенным) подхода вывод будет сделан в форме так называемой семантической базы данных – со всеми возможными значениями слова и вероятностями его появления в данном контексте.
* По результатам предыдущего анализа семантика извлекается уже из всего предложения. При этом, если после первого шага семантика того или иного слова остается неопределенной (значение не ясно из контекста всего одного предложения), на данном шаге неопределенность исключается.
* На третьем этапе происходит представление результата и вывода – текст приобретает форму структурированного набора данных, с которым можно работать. Конечная цель семантического анализа достигнута.

## Обзор систем с ЕЯ-интерфейсом

* + 1. ***История развития***

Первые прототипы NLP систем появились еще в в конеце 60-х, прошлого столетия, но, несмотря на малые вычислительные мощности, они уже показывали значительные результаты. Рассмотрим самые значимые из них.

*ELIZA -* компьютерная программа и ранний пример примитивной обработки естественного языка. ELIZA работал, обрабатывая ответы пользователей на подлинники, самым известным из которых был ДОКТОР, моделирование психотерапевта Rogerian. Используя почти никакую информацию о мысли человека или эмоциях, ДОКТОР иногда обеспечивал поразительно подобное человеку взаимодействие. ELIZA был написан в MIT Йозефом Вайценбаумом между 1964 и 1966 [21].

*SHRDLU* был ранней компьютерной программой понимания естественного языка, развитой Терри Виногрэдом в MIT в 1968–1970. В нем пользователь вводил в терминал команды, а виртуальная модель робота принимала решение какое действие необходимо совершить. Модель робота находилось в виртуальном пространстве «мира игрушечных блоков», который представлял из себя комнату, наполненную блоками.

Робот мог перемещать блоки и сообщать информацию об окружающей среде, такую как «что находится в коробке» или «блок какого цвета ты держишь» [6].

Во второй половине 20-го века BBN Laboratories проводили исследования и разработку в области технологий интерфейса на естественном языке со следующими аспектами:

* продемонстрировать новейшие технологии в программах для стратегических вычислений, собирая данные об эффективности продемонстрированной эвристики;
* провести исследование технологии интерфейса на естественном языке JANUS;
* интегрировать технологии от других поставщиков интерфейса на естественном языке, включая USC / Институт информационных наук, Университет Пенсильвании и Массачусетский университет.

Отправной точкой для разработки системы интерфейса на естественном языке стала система *IRUS*, которая разрабатывалась в BBN для Командного центра Тихоокеанского флота. Она представляла собой каскад, состоящий из последовательности трансляторных модулей:

* парсер ATN, который создает синтаксическое дерево;
* семантический интерпретатор, который вырабатывает формулу смыслового представления языка MRL;
* постпроцессор для разрешения анафоры и эллипса;
* трансляторный модуль, который создает формулу языка реляционных баз данных ERL («Расширенный реляционный язык»);
* трансляторный модуль, который производит последовательность команд для базовой системы доступа к базе данных.

Новая версия этой системы, *IRUS-86*, была установлена для демонстрации в испытательном полигоне FCCBMP в командном центре Тихоокеанского флота. Особенно стоит упомянуть два из этих расширений:

* IRUS-86 использовал систему NIKL для представления своей доменной модели, то есть отношений между предикатами и отношениями смыслового языка представления MRL. Модель домена NIKL поддерживает системную обработку семантической аномалии, анафоры и номинальных соединений;
* IRUS-86 содержала новый модуль, который использует эту модель домена NIKL для упрощения выражений MRL; это позволяет переводить сложные выражения MRL в константы ERL, что позволяет существенно расхождения между исходным английским языком и структурой базовой базы данных.

Дальнейшие фундаментальные исследования по проблемам взаимодействия с естественными языками продолжаются, и результаты этого и будущих исследований будут включены в систему взаимодействия на естественном языке следующего поколения под названием *JANUS*, которая будет доставлена в командный центр Тихоокеанского флота позднее. JANUS поделится большей частью доменно-зависимых данных с IRUS-86, а также с другими модулями; Таким образом, IRUS-86 сможет постепенно эволюционировать в финальную версию JANUS. Он использовал такие методы, как:

* Упрощение преобразований для выявления абсурдных интерпретаций, сводя ее к ИСТИНЕ или ЛЖИ или пустого множества.
* Контекст диалога и модель целей пользователя накладывают ограничения на ожидаемые предложения.

Велись разработки для решения проблемы в области синтаксиса. Для этого писалась декларативная грамматика, которая в конечном итоге будет охватывать английскую грамматику, написанную в формализме без побочных эффектов (грамматика без переменных с контекстом). Первый такой алгоритм был реализован в мае 1986 года на BBN's Butterfly.

Разработчики не гарантировали, что JANUS всегда сможет правильно оценить, какое толкование предложения является намеченным. Главной задачей было перефразирование ввода пользователя, что подразумевает интерпретацию команд. Это может быть сделано либо явно, либо как часть ответа. Чтобы иметь возможность развивать такие возможности, необходима работа над созданием собственного ограниченного естественного языка.

В USC / ISI реализуется проект под руководством Уильяма Манна и Нормана Сондхаймера по разработке системы интерпретации *PENMAN* с использованием системной грамматики NIGEL. PENMAN будет интегрирован, чтобы стать компонентом генерации JANUS. Сам PENMAN состоит из нескольких подкомпонентов. Некоторые из них, в частности компонент «планирование текста», будут разработаны в результате совместной работы USC / ISI и Дэвида Макдональда в Массачусетском университете на основе опыта фермера с системой *MUMBLE*.

Под руководством Аравинда Джоши и Бонни Уэббер из Университета Пенсильвании было проведено несколько целенаправленных исследований для изучения различных аспектов поведения кооперативной системы и выяснения взаимодействий. В рамках усилий по разработке стратегических вычислений для естественных языков UPenn в конечном итоге превратит это в модуль, который интегрируется в JANUS для дальнейшего расширения его возможностей.

Рассмотрим две связанные работы : (Бадлер и др., 1993)[4] и (Бадлер и др., 1999)[5]. Бадлер с соавторами собирались разработать программного робота, анимированного в виртуальном пространстве и понимающего естественный язык. Роботу предоставлялась база данных с инструкциями с необходимыми параметрами, по которым он должен был выполнять действия. Параметры в себе содержали в себе много информации, такую как лингвистическая, пространственно-временная и т. д., однако авторы не учитывали анафорические выражения, которые часто встречаются в разговорном языке.

Каcсэл и др. [6] в 1999 обратили внимание на то, что не вербальные выражения в разговоре очень важны во взаимодействиях человек-машина. Такие выражения включают мимику лица, положение рук и звучание голоса. Они разработали систему, названную «Rea», которая представляла собой диалогового агента, и в ней были фактически реализованы социальные, лингвистические и психологические условия разговоров. Помимо вербальной речи она могла отвечать жестами рук, мимикой лица и пристальным взглядом.

В 2000 г. Шиньяма с соавторами разработали систему «Kairai» [7]. Система не только могла интерпретировать естественную речь в команды, но и обрабатывать анафорические выражения. Их основной задачей было разработать мультиагентную систему, где бы программные роботы могли работать и взаимодействовать друг с другом независимо. С первого взгляда может показаться, что им это удалось, однако фактически все роботы соединялись одним не визуальным роботом, который прослушивал команды и перенаправлял их тому роботу, которому она предназначалась по контексту.

В 2004 Гаврилов А.В. предложил архитектуру гибридной системы управления мобильным роботом [13]. Предложенная ГИСУР должна решать следующие задачи:

1. восприятие и распознавание информации, поступающей из внешнего мира от датчиков;
2. общение с человеком;
3. создание и корректировка модели мира путем обучения в процессе общения с человеком, восприятия сигналов с датчиков и выполнения действий;
4. планирование и перепланирование поведения;
5. управление выполнением действий; 6) общение с другими роботами.

Далее он представляет 2 модели архитектуры, первая из которых основана на построена на основе нейронных сетей и экспертной системы в среде оболочки ESWin (Рисунок 3).

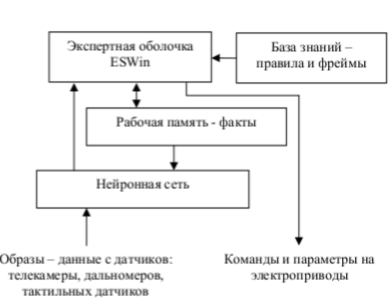


Рисунок 3. Архитектура системы управления робота, ориентированная на программирование поведения робота в виде правил.

Вторая модель предназначена для исследования архитектуры на основе нейронных сетей и семантической сети (Рисунок 4), и ориентирована на естественно-языковое взаимодействие с роботом в процессе его обучения и выполнения перемещений по словесной команде. В этой архитектуре нейронная сеть может выполнять функции классификации и кластеризации образов или ситуаций, памяти ассоциаций между словами (фразами) естественного языка и образами, между ситуациями и реакциями (действиями) на них. Последняя функция полезна при выработке автоматизма при планировании перемещений среди объектов-препятствий. Эта архитектура была предложена и планируется к использованию в системе управления роботизированного инвалидного кресла.

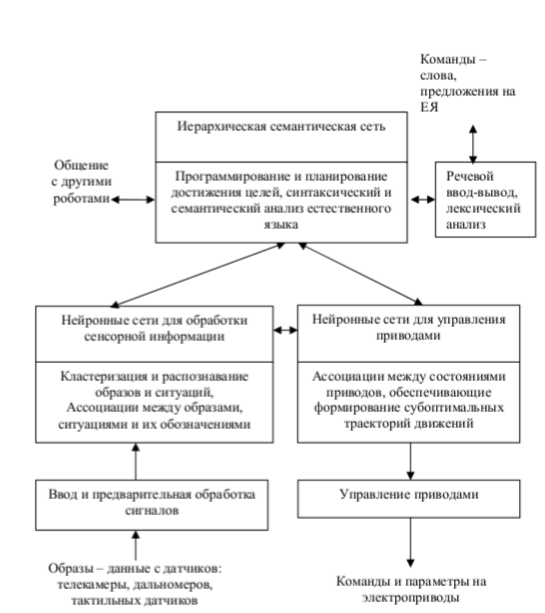


Рисунок 4. Архитектура системы управления робота, ориентированная на естественно-языковое общение

В статье 2014 г. [15] Гаврилов А.В. развивает эту идею, где архитектура системы имеет вид, представленный на Рисунке 5. В процессе обучающего диалога пользователь стремится описывать элементарные поведения и условия для их запуска. Чтобы запустить какое-либо поведение, система распознает соответствующее понятие (т.н. «releaser» в реактивной парадигме системы управления роботом).

Ассоциативная память должна удовлетворять следующим требованиям:

1. позволять использование аналоговых и бинарных входов/выходов,
2. обеспечить продолжающееся (incremental) обучение,
3. обеспечить хранение цепочек понятий (как поведений или сценариев).

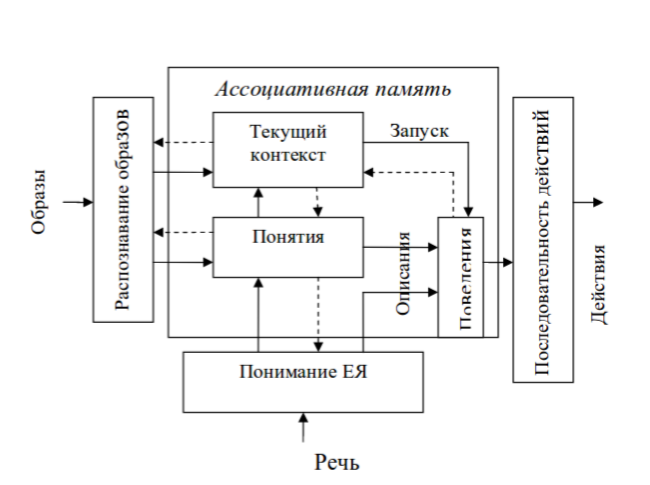


Рисунок 5. Контекстно-ориентированная архитектура системы управления мобильного робота

Элементарное поведение подобно подпрограмме и содержит последовательность действий адаптируемых к переменным контекста, которые могут рассматриваться как аналог параметров подпрограммы. Для описания сценариев поведения робота требуется высокоуровневый язык, который бы отражал семантическую структуру команд. Автор предлагает использовать язык БАЯР[14](позднее язык CBLR в [3,4]).

Примерные предложения на естественном языке для обучения мобильного робота с использованием последней версии CBLR:

1. Идти – {#act}. 2)
2. Объект – {#object}.
3. Яблоко, стол – объекты.

4) Идти к объекту – {#object; #act}.

5) Стол это мебель.

6) Мебель: стол, стул, книжный шкаф, посудный шкаф.

7) Какой объект прямо – {#direction= Forward; #say}.

8) Найди объект – {#action=Find; #act}.

В общем случае, когда мы имеем дело со сложным роботом (например, гуманоидным), который должен обеспечивать сложное поведение, необходимо разбить контекст на иерархию контекстов для каждого вида действия, как показано на рисунке 6.



Рисунок 6. Иерархия контекстов

Обработка команды роботу в виде предложения на ЕЯ состоит из следующих этапов:

1. распознавание слов с присваиванием им роли (глагол, существительное, прилагательное или ничего),
2. распознавание словосочетаний с использованием иерархической семантической сети (в первую очередь с использованием ролей слов),
3. Поиск в семантической сети по ассоциативным связям примитивов или групп примитивов. При этом происходит присваивание значений контекстным переменным.
4. Запуск подпрограмм (выполнение притивов типа #act, #say и т.п.).
   * 1. ***Анализ актуальных прототипов***

В настоящее время ЕЯ-интерфейсы приобретают большое распространение. Они используются в таких областях, как:

* *Диктовка*, наиболее популярный способ использования систем автоматизированного распознавания речи (ASR) сегодня. Это включает медицинскую транскрипцию, юридическую и деловую диктовку и обработку обычного слова. В некоторых случаях специальные словари используются, чтобы увеличить точность системы.
* *Обработка команд*, системы ASR, которые разработаны, чтобы выполнить функции и действия на системе, определены как системы командования и управления. Произнесение как «Открытый Netscape» и «Начало новый xterm» сделает просто это.
* *Телефония*, некоторые Почтовые системы PBX/Voice позволяют посетителям говорить команды вместо неотложных кнопок, чтобы послать определенные тоны.
* *Wearables*, потому что входы ограничены для пригодных устройств, разговор, является естественной возможностью.
* *Медицина*, много людей испытывают затруднения при печати из-за физических ограничений, таких как хронические растяжения сухожилий (RSI), мышечная дистрофия и многие другие. Например, люди со слушанием трудности могли использовать систему, связанную с их телефоном, чтобы преобразовать речь посетителя в тексте.
* *Вложенные заявления*, некоторые новые сотовые телефоны включают C&C распознавание речи, которые позволяют произнесение, такое как «требование домой». Это может быть основным фактором в будущем автоматического распознавания речи и [Linux](http://ru.knowledgr.com/02745100/Linux).

Далее рассмотрим некоторые актуальные прототипы таких систем.

***WolframAlpha***

Проект, запущенный в 2009-м году, который представляет из себя базу знаний и набор алгоритмов для упрощения поиска ответов на естественно-языковые вопросы в интернете. Основателем проекта выступил Стивен Вольфрам, заявив, что сможет перевести естественно-языковые вопросы в формат, понятный для компьютеров, что позволит производить вычисления и поиск через триллионы единиц «Кураторов данных» с использованием миллионов строк алгоритмов для предоставления пользователю ответов.

Wolfram|Alpha не возвращает перечень ссылок, основанный на результатах запроса, а вычисляет ответ, основываясь на собственной базе знаний, которая содержит данные о [математике](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0), [физике](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D0%BA%D0%B0), [астрономии](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%82%D1%80%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D1%8F), [химии](https://ru.wikipedia.org/wiki/%D0%A5%D0%B8%D0%BC%D0%B8%D1%8F), [биологии](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F), [медицине](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B4%D0%B8%D1%86%D0%B8%D0%BD%D0%B0), [истории](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%82%D0%BE%D1%80%D0%B8%D1%8F), [географии](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F), [политике](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D1%82%D0%B8%D0%BA%D0%B0), [музыке](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%B7%D1%8B%D0%BA%D0%B0), [кинематографии](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BD%D0%B5%D0%BC%D0%B0%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84), а также информацию об известных людях и интернет-сайтах. Он способен переводить данные между различными [единицами измерения](https://ru.wikipedia.org/wiki/%D0%95%D0%B4%D0%B8%D0%BD%D0%B8%D1%86%D1%8B_%D1%84%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D1%85_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD), [системами счисления](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F), подбирать общую формулу [последовательности](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C), находить возможные замкнутые формы для приближенных дробных чисел, выполнять простые вычислительные операции с числами и матрицами, определять свойства чисел и геометрических фигур.

Однако, система не лишена сушественных недостатков, вытекающих из использования собственной базы. Недостатки вытекают в уязвимость для ошибок данных. Например, на момент открытия, запрос «*president of russia 1999»* выдавал имя Аслана Масхадова (в настоящее время эта ошибка уже исправлена). Это говорит о том, что система заполнения базы знаний плохо автоматизирована и требует ручного ввода информации и ее корректирование. Немало важным аспектом является то, что система работает только с английским языком.

***Apple Siri***

В 2011 году Apple представила Siri, задав моду на голосовых помощников. На тот момент это было что-то действительно новое Однако время шло, и на рынке появились первые конкуренты - Google Assistant, Amazon Alexa, и, на российском рынке, Яндекс Алиса. Их развитие было очень стремительным, тогда как Siri, на протяжении многих лет, практически не совершенствовалась.

Основные технические области Siri сосредотачиваются на диалоговом интерфейсе, распознавании контекста и сервисной делегации. [Распознавание речи](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D0%BE%D0%B7%D0%BD%D0%B0%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1%80%D0%B5%D1%87%D0%B8) Siri основано на разработках голосовых технологий компании Nuance Communications.

Siri обладает собственным API, который могут использовать заинтересованные разработчики, однако имеются некоторые недостатки:

* Как и большая часть программного обеспечения Apple, исходный код Siri закрыт, а API обладает очень ограниченным функционалом.
* Русский синтезатор речи слабый, по сравнению с прототипами от конкурирующих компаний, а распознавание речи работает не точно

***Google Assistant***

Наибольшую популярность среди обычных пользователей ЕЯ-интерфейсы приобрели благодаря их использованию в голосовых помощниках на мобильных устройствах.

На конференции Google I/O в мае 2016 года был представлен голосовой помощник для смартфонов Pixel и Pixel XL. Его можно рассматривать как апгрейд или расширение для Google Now. Он разрабатывался для того, чтобы стать полноценным голосовым ассистентом под операционную [систему Android](http://ipkey.com.ua/faq/912-android.html) в дополнение к привычному всем “Окей, Гугл”.

Google Assistant становится для пользователя не просто помощником, но персональным Google.

Актуальные возможности помощника:

* предоставление информации в Google, проверка погоды или статуса самолётного рейса;
* напоминания о событиях;
* программирование девайсов, входящих в ваш персональный [интернет вещей](http://ipkey.com.ua/faq/908-internet-of-things.html);
* безопасная передача данных и персональный контроль информации, которой вы обмениваетесь;
* совместимость со многими популярными сервисами, такими как Netflix, Uber, [Philips](http://ipkey.com.ua/teg-philips.html) Hue и другими

Сервис, так же, представляет возможность использовать API Ассистента в общем доступе.

Архитектура приложений системы, представленной на рисунке 7, реализованных на стандартном технологическом стеке Google:

* *Actions on Google* — платформа для создания приложений для Google Ассистента.
* *Dialogflow*— NLU-движок (Natural Language Understanding), отвечающий за обработку естественных языков и дизайн диалогов.
* *Cloud Functions for Firebase*— облачные функции для обработки сложной логики взаимодействия с пользователем и для работы со сторонними сервисами. Firebase Functions и Dialogflow взаимодействуют через webhook, поэтому технически можно использовать любое другое серверное решение.

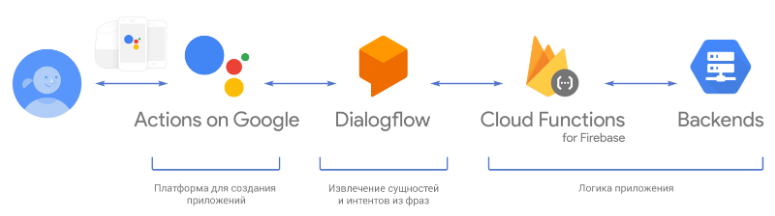


Рисунок 7 - Архитектура приложения для Ассистента

На сегодняшний момент, система является одной из наиболее перспективных для использования в веб-сервисах, а главное обладает открытым кодом.

***Яндекс Алиса***

*Алиса* — голосовой помощник, запущенный Яндексом 10 октября 2017 года. Разработчики из Яндекс уверяют, что для генерации ответов, данная система совмещает использование шаблонных сценариев и deep learning (глубоких нейронных сетей), что очень похоже на использование *гибридного подхода* к решению общей задачи.

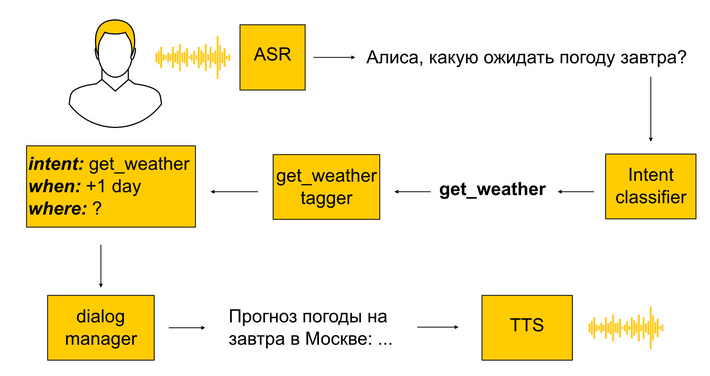


Рисунок 8. Алиса, вид сверху

Когда пользователь произносит команду, система записывает звуковую дорожку и отправляет ее в сервер распознавания речи (ASR), который переводит ее в текст. Затем текст отправляется в подсистему классификатора интентов (Intent classifier), который определяет какое действие нужно совершить: выдать текстовый ответ или, например, вывести информацию о погоде. Все это выводится в виде некоего структурированного представления, которое называется фреймом. Все результаты взаимодействия с пользователем обрабатывает dialog manager, который хранит информацию об этом пользователе, такую как данные об его устройстве, геолокации и т.д., а так же контекст диалога.

А теперь рассмотрим подробнее каждый элемент.

*Dialog manger.* В этом модуле отсутствую какие-либо методы машинного обучения, только шаблонные сценарии. В основе концепции лежит form-filling. Пользователь своими репликами заполняет некую виртуальную форму, и когда все необходимые поля будут заполнены, он получит ответ. Логика диалога конструируется благодаря движку event-driven, который запускает события в ответ на пользовательские реплики.

*Язык шаблонов.* В этом модуле используются лингвистические средства, например возможности склонять слова или согласовывать числительные и существительные, чтобы легко писать текст с логико-семантическими связями, а так же рандомизировать кусочки текста, для увеличения вариативности речи Алисы.

*Классификатор интентов.* В этом методе использовались множество разных моделей нейронных сетей, начиная от логистической регрессии и заканчивая градиентным бустингом, рекуррентными сетями. В итоге остановились на классификаторе, который основан на ближайших соседях. Такая модель определяет сематически близкие фразы и не требует полного переобучения после обновления тренировочной выборки.

*Сематический теггер.*  В основе лежат обычные двунаправленные нейронные сети для задачи тегирования. Так же используются N-best гипотезы, при котором генерируется не самая вероятная гипотеза, среди всех имеющихся, а только те, которые прошли процедуру перебалансеровки весов, в зависимости от текущего диалога.

*Анафоры и Элипсис*. Cистема разрешения анафор, основанна на form filling, она работает примерно так: если в предыдущем интенте в форме был геообъект, и в текущем есть слот для геообъекта, и он не заполнен, и еще мы в текущий интент попали по фразе с местоимением «туда», то наверное, можно предыдущий геообъект импортировать из формы и подставить сюда. Это простая эвристика, но производит неплохое впечатление. В части интентов работает одна система, а в части обе. Разработчики смотрят, где работает, где не работает, гибко это настраивают.

Эллиптические фразы или фразы-уточнения — это отдельные интенты. Если есть интент get\_weather, для которого в трейне фразы типа «расскажи погоду», «какая сегодня погода», то у него будет парный интент get\_weather\_ellipsis, в котором всевозможные уточнения погоды: «а на завтра», «а на выходные», «а что там в Сочи» и так далее. И эти эллиптические интенты в классификаторе интентов на равных конкурируют со своими родителями. Если вы скажете «а в Москве?», классификатор интентов, например, скажет, что с вероятностью 0,5 это уточнение в интенте погода, и с вероятностью 0,5 уточнение в интенте поиска организаций, например. И затем диалоговый движок перевзвешивается scores, которые назначил классификатор интентов, который назначил их с учетом текущего диалога, потому что он, например, знает, что до этого шел разговор о погоде, и вряд ли это было уточнение про поиск организаций, скорее это про погоду. Такой подход позволяет обучаться и определять эллипсисы без контекста.

*Болталка.* Болталка — DSSM-подобная нейронная сеть, где есть две башни энкодера. Один энкодер кодирует текущий контекст диалога, другой — ответ-кандидат. У вас получается два embedding-вектора для ответа и контекста, и сеть обучается так, чтобы косинусное расстояние между ними было тем больше, чем уместнее данный ответ в контексте и чем неуместнее.

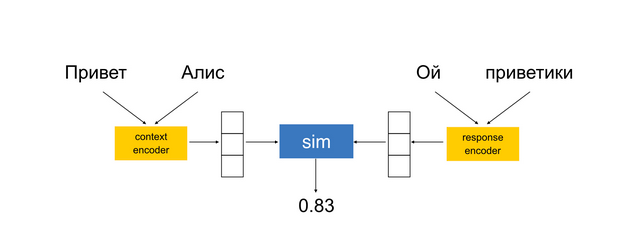


Рисунок 9. Архитектура болталки

Так же система имеет открытое API, доступное для всех разработчиков. На техническом уровне это значит, что вы можете подключить сервис, работающий на вашем сервере, к платформе Алисы. Когда пользователь активирует такой навык, Алиса транслирует реплики пользователя на ваш сервер, и отвечает переданным вами текстом. Распознавание речи в текст и обратно платформа берёт на себя. Для пользователя разговор с использованием навыка выглядит как обычный диалог с голосовым помощником.

В выводе можно сказать, что на данный момент, Яндекс Алиса, является самым лучшим сервисом разработки собственных голосовых помощников в российском интернете. Но существуют также и недостатки, такие как:

* Нельзя отвязаться от головных серверов Яндекса. Компания жестко контролирует использование их API, а чтобы добавлять новые навыки своему ассистенту придется проходить долгую и нелогичную модерацию, из-за которой модерация может быть не пройдена из за слишком общего или неоригинального названия.
* Не всегда документация соответсвует действительности. Сама платформа развивается быстро, документация не поспевает, трактуется пользователями и представителями яндекса иногда по-разному.
* Платформа пока предоставляет лишь базовые средства для дебага навыков.При добавлении навыка в панель разработчика ваша система оповещений уже должна отвечать правильно, или просто произойдет ошибка валидации.

Существуют еще многие сервисы, которые занимаются интерпретацией естественного языка в систему команд, но большинство из них либо ориентированы на использование английского языка, либо лишены открытого API, которым бы мог свободно воспользоваться сторонний разработчик. Рассмотрим далее сервисы, которые обработку естественного языка выдвигают на первый план, в то время как посторонние сервисы не обязательны.

* + 1. ***Сравнительный анализ платформ обработки естественного языка***

Разработка любой крупной компьютерной системы или прог раммного пакета – это достаточно трудоемкий и длительный процесс. Затраченное время на такие проекты варьируется от нескольких месяцев до года (и более), также влияет на процесс создания и величина финансовых вложений. Инструментарием, который значительно облегчает влияние этих факторов на разработку, является использование вспомогательных компонентов, которые обычно включают различные наборы библиотек, фреймворков и пакетов. Инструментарий обработки ЕЯ не исключение, он также представляет собой различные библиотеки и фреймворки для символического и статистического анализа естественного языка и обработки речи, а также готовые платформы, с помощью которых сразу можно работать с данными. В сравнительном анализе были рассмотрены как платформы, так и отдельные компоненты, в виде наборов библиотек.

Среди разработчиков, представленных в табл. 1 и 2, как и в любой другой коммерческой среде, существует определенная конкуренция. Ее наличие следует из двух логичных фактов: во-первых, разработчиков несколько; во-вторых, некоторые из них владеют платным вариантом собственного «решения». В табл. 1 приведены популярные готовые платформы, а в табл. 2 актуальные подключаемые компоненты обработки ЕЯ (наборы библиотек). В качестве критериев сравнительного анализа (оцениваемости), были выбраны следующие: вид лицензии (модель распространения), разработчик или автор. Также в сравнение включен вид предоставляемого решения, то есть, является ли компонент готовой платформой, с помощью которой сразу можно заниматься обработкой данных или это подключаемая библиотека, содержащая только алгоритмы обработки ЕЯ. В критерии включены: поддерживаемый язык программирования (если это библиотека) и наличие дополнительных сервисов, которые разработчик предоставляет клиенту.

Таблица 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Платформа | Разработчик/ автор | Лицензия | Вид предоставляемого решения | Язык програм- мирования (для наборов  библиотек | Дополнительные сервисы |
| Alchemy API | Orchestr8 | бесплатная демо-версия/ коммерческая | платформа | - | Обширная поддержка по продукту. Готовые решения для: «облачных» сервисов, обработки файлов документов и e-mail, а также веб-сайтов. |
| Carabao | Digital Sonata Pty Ltd | бесплатная демо-версия/ коммерческая | платформа | - | Автоматизированный многоязычный поиск, компоненты для мониторинга веб-ресурсов. |
| Ellogon | Georgios Petasis | свободное ПО (LGPL) | платформа | - | Открытое внедрение собственных словарей, инструменты для обработки и визуализации текстов HTML/XML и наличие базового графического интерфейса. |

Продолжение таблицы 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Платформа | Разработчик/ автор | Лицензия | Вид предоставляемого решения | Язык програм- мирования (для наборов  библиотек | Дополнительные сервисы |
| Free Ling | Политехнический университет Каталонии | свободное ПО (GNU) | набор библиотек | С++ | - |
| Ling Pipe | Alias-i | бесплатная демо-версия/ коммерческая | набор библиотек | Java | - |
| Lingua Stream | Университет Кан (Франция) | академи- ческая | набор библиотек | Java | поддержка и помощь разработчикам, использующим это решение, в виде вебсообщества на соответствующем форуме |
| modular Audio Recognition Framework | The MARF Research and Development Group | свободное ПО (BSD) | набор библиотек | Java | расширенный инстру- ментарий по обработке звуковой информации |
| UIMA | Apache | коммер- ческая | набор библиотек | Java, C++ | качественная коммерческая техническая поддержка |

## Результаты и выводы по главе

В первой главе были рассмотрены методы разработки систем управления, обеспечивающих понимание ЕЯ. Это требует создание механизмов, отличных от традиционных способов представлений ЕЯ, а системы с естественно-языковыми интерфейсами применяются только в узких предметных областях.

Технологии речевого распознавания нашли свое применение в различных областях. Однако в области систем управления множество проблем все еще остаются не решенными, многие идеи требуют дальнейшего развития. Была произведена попытка рассмотреть проблемы, которые лежат перед разработчиками систем искусственного интеллекта при попытках моделирования понимания. Часть проблем уже решена на хорошем уровне, где-то системы ИИ работают с большей эффективностью, чем человек, где-то едва делаются первые шаги. Проблема машинного понимания, на данный момент, все еще остается очень сложной и связана с крайне широким спектром проблем. Значительный успех в этой области приобрели нейронные сети, но они все еще способны выполнять только узкие задачи и обрабатывать только ограниченный ЕЯ. Однако этого полностью достаточно для создания полнофункциональных ЕЯ-интерфейсов, не требующих глубокого анализа семантики и грамматических особенностей ЕЯ.

Проведен сравнительный анализ современных прототипов систем с ЕЯ-интерфейсами и ПО для обработки ЕЯ. Многие программные инструменты находятся в свободном доступе с открытым исходным кодом, что позволяет использовать их в собственных проектах без юридических ограничений, однако ни один из них не удовлетворяет требованиям ГИСУМР (Гибридная Система Управления Мобильным Роботом). Все сводится к тому, что полностью отвязаться от серверов разработчиков не получится, API обладают ограниченным функционалом, и привязать их к своей модели робота очень проблематично. Для реализации ГИСУМР принято решение разрабатывать собственный интерпретатор ограниченого ЕЯ, который бы полностью подходил для решения поставленных перед системой задач.

# Разработка интерпретатора естественного языка

## Эмбединг слов

*Метод Word2Vec* подразумевает кодирование словаря таким образом, чтобы каждому слову соответствовал вещественный вектор, элемент евклидова пространства Rd для некоторого d (обычно несколько сотен). Эти векторы далее входят в последующие модели и предполагается, что геометрическое соотношения в пространстве Rd будут соответствовать семантическим отношениям между словами, благодаря чему синонимы или другие тесно связанные слова оказываются ближайшими соседями []. Соотношение какой либо сущности, в данном случае слова, какому либо вектору называется эмбеддинг( от анг. Embedding. Русского эквивалента эта слово не имеет).

Это распределение основывается на предположениях, которые основываются на дистрибутивной гипотезе (distributional hypothesis)[], по которому слова с похожим смыслом будут встречаться в похожих контекстах. Эта гипотеза не вполне практична, но ее количественное выражение крайне полезно для автоматической обработки естественного языка.

Идея современной модели word2vec базируется на работе Йошуа Бенджи и др. [22], которые, основываясь на задаче построения языковой модели, предложили идею распределенного представления слов, которая обучалась таким образом:

* Каждому i-му слову () сопоставляется вектор представления слова  размерности , при ;
* Выражается вероятность появления i-го слова в локальном контексте c1, … , cn как функцию от этих векторов представлений слов:

,

где  - это векторы слов из контекста, а *f —* некоторая функция с параметрами Θ, которая на вход получает векторы слов;

* далее, имея в качестве данных большой корпус текстов, можно провести обучение параметров функции *f ,* одновременно с векторами *w,* максимизируя логарифм общего правдоподобия корпуса:

,

где *t* пробегает все доступные в корпусе окна слов от 1 до *T*, а *R(W,Θ) —* регуляризатор.

Этот метод предназначен, прежде всего, для трансляции текста (перевода с одного языка на другой), путем интерпретации одного набора слов в другой. При этом вовсе не обязательно, чтобы метод пользовался двумя словарями разных языков.

На базе архитектуры этой системы из работы [26] предлагается разработать интерпретатор, который на выходе генерировал не текст, а классы действий, который необходимо совершить управляемому роботу. Для усиления работы этого метода предлагается протестировать его работу в двух вариациях:

* Команды представляются в виде классов слов, дополненных классом контекста. В этом случае, при любых внесенных изменениях, интерпретатор необходимо переобучать полностью;
* Команды представляются в виде классов символов в команде, дополненных классом контекста. Подразумевается, что символы обрабатываются как отдельные слова. Это позволит минимизировать объем словаря, и полное переобучение интерпретатора будет происходить значительно реже.

Все команды считываются системой через микрофон, или вводятся через текстовый терминал, и разбиваются на слова / символы, которые проходят Word Embedding, после чего слова представляются в виде one-hot векторов (рисунок 1).



Рисунок 10- One-Hot кодирование слов

## Архитектура используемой нейронной сети

Основой интерпретатора должна служить нейронная сеть, которая преобразует вектор чисел в другой вектор чисел (seq2seq – от анг. Sequence to sequence ). При этом элементы вектора должны иметь связи по смыслу (внутренний контекст вектора фразы).

*Вектор фразы*  - это вектор целых чисел, каждая ордината которого обозначает индекс объекта из словаря интерпретатора. Объектами словаря являются символы/слова команды и классы выполняемых действий.

*Внутренний контекст* обозначает связь между соседними ординатами вектора фразы. Далее будем их называть *токенами* (от анг. Token - знак, лексема). В случае, если за токены приняты слова команды, то в фразе «на столе стоит стакан», токен «стоит», должен учитывать высокую вероятность появления токена «стол» слева и токена «стакан» справа. Это позволит сохранить корректность последовательности действий на выходе из интерпретатора.

Нейроны обычной нейронной сети (рисунок 2) имеют слишком примитивную математическую модель, чтобы справляться с «сохранением» какой либо информации, которая указывает на влияние одного слоя нейронов на другой. Решением этой проблемы служит использование рекуррентных нейронных сетей, в частности LSTM. Усложнение математической модели таких сетей заключается в том, что каждый нейрон обладает индивидуальной ячейкой, которая может хранить необходимую информацию о предыдущих ее состояниях. Так как основные задачи интерпретатора ГИСУМР схожи с текстовыми трансляторами, наиболее подходящей моделью нейронной сети является LSTM, которая, на данный момент, наиболее преуспела в генерации текста с внутренним контекстом.

*LSTM* (от анг. Long Short-Term Memory – долгая кратковременная память). Базовая модель ячейки LSTM изображена на рисунке 3. Нейрон обладает, так называемыми, переключателями (гейтами), которые сигнализируют о том, какую информацию необходимо сохранить, а какую удалить. Таким образом ячейка будет «помнить» о ее предыдущих состояниях и не накапливать ненужную информацию.

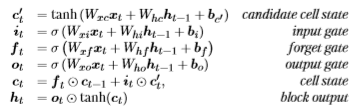
|  |  |
| --- | --- |
| ÐÐ°ÑÑÐ¸Ð½ÐºÐ¸ Ð¿Ð¾ Ð·Ð°Ð¿ÑÐ¾ÑÑ "Ð¼Ð°ÑÐµÐ¼Ð°ÑÐ¸ÑÐµÑÐºÐ¸Ð¹ Ð½ÐµÐ¹ÑÐ¾Ð½""  Рисунок 11 – Ячейка нейрона обычной нейронной сети | Рисунок 12 - LSTM ячейка |

Есть 3 основных вида переключателей:

* Input gate – входной ;
* Foget gate – забывающий;
* Output gate – выходной;

и узел рекуррентной ячейки со скрытым состоянием (cand. Cell state) — вектор, который выполняет функцию памяти.

На вход ячейки подается 2 вектора: вектор входных данных, обозначенный через xt,, и вектор скрытого слоя, обозначенный через ht. Через Wi (с разными вторыми индексами) обозначается матрицы весов, применяющиеся ко входу, через Wh — матрицы весов в рекуррентных соединениях, а через b—векторы свободных членов, мы получим следующее формальное определение того, как работает LSTM: на очередном входе xt, имея скрытое состояние из предыдущего шага ht−1 и собственно состояние ячейки ct−1, мы последовательно вычисляем:



В общем виде модель нейронной сети будет выглядеть так, как изображено на рисунке 4,

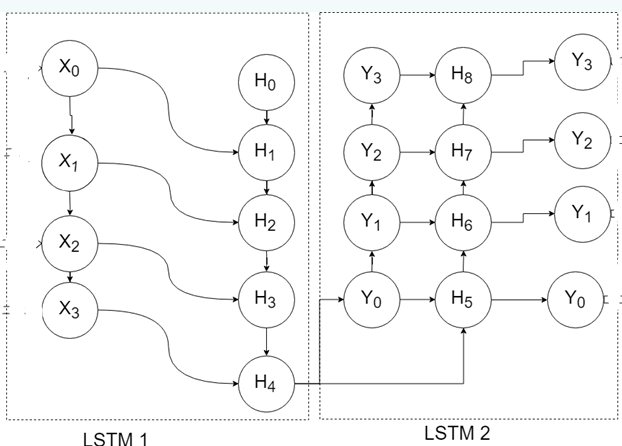


Рисунок 13 – 2 последовательно соединенных модели нейронной сети LSTM

Где:

* Xi – вектор идентефикаторов входных токенов;
* Hi – вектор скрытого слоя
* Yi – вектор выходных токенов.

## Архитектура ГИСУМР и гибридный подход к решению задач

В работе [] автором была представлена успешная реализация диалоговой системы (чат-бота с исполняемыми сценариями), где использовался один словарь русского языка, а в задачи метода входило генерация текстовых ответов на вопросы пользователя. Интерпретатор использовал два последовательно соединенных рекуррентных слоя нейронных сети (LSTM), и распределение слов Word2Vec. Сама по себе система являлась диалоговым агентом, который поддерживал беседу с клиентами и выполнял запросы к базам данных института. Система распознавала принадлежность к группе клиентов (гость, студент, преподаватель, администратор), помнила контекст предыдущего диалога и, опираясь на него, принимала решение какое действие необходимо совершить. Основными исполняемыми действиями являлись вывод/ввод/изменение информации в базах данных института, а также выдача текстового ответа.

*Гибридный подход* к решению задач подразумевает комбинирование методов машинного обучения и шаблонных алгоритмов, а также символьного и численного представления понятий и образов. В работе [26] применялся подобный подход для разделения подсистемы команд и «поддержания беседы» с пользователем. Для улучшения ее архитектуры, изображенной на рисунке 14 слева, и ее адаптации под систему управления мобильным роботом, требуется изменить последовательность узлов интерпретатора.

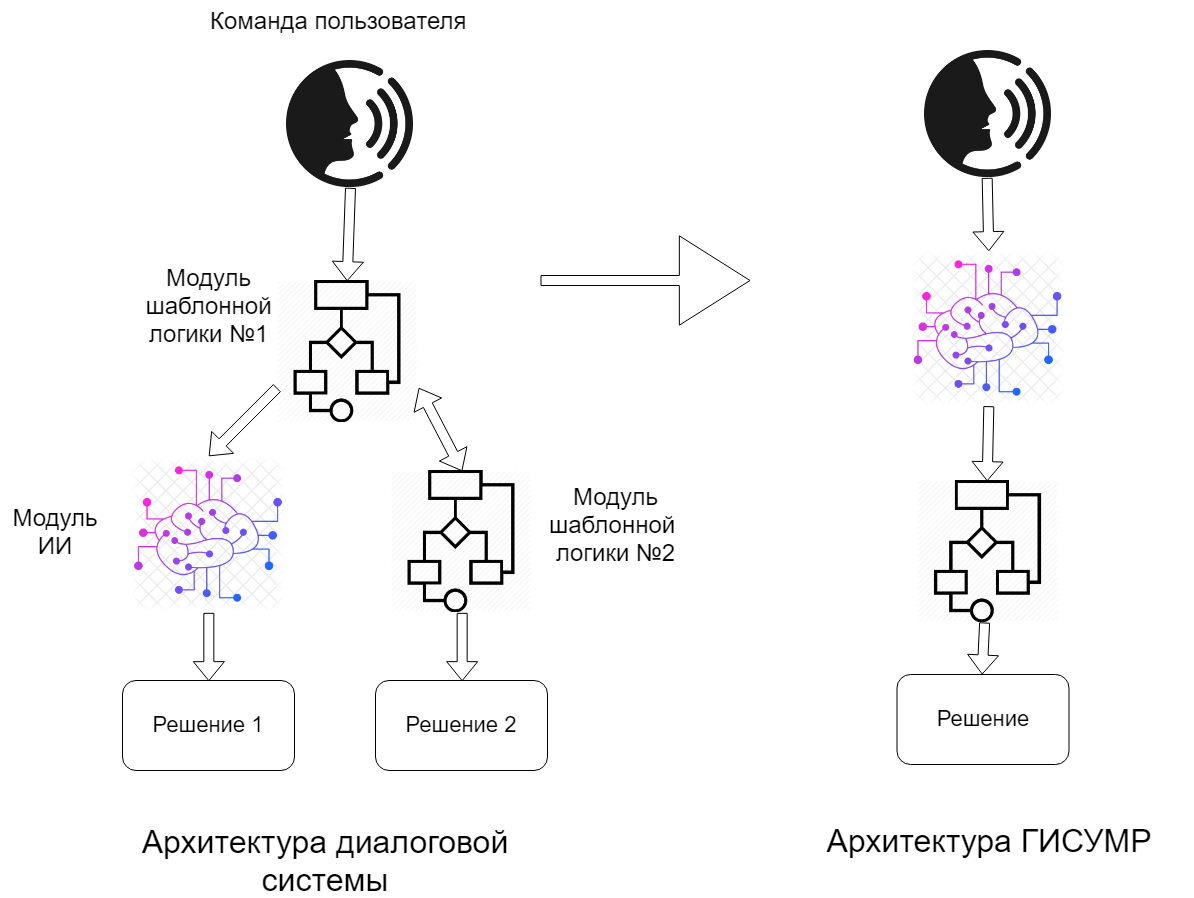


Рисунок 14 - адаптация архитектуры диалоговой системы под ГИСУМР

В диалоговой системе использовалась параллельная последовательность модулей интерпретатора, что объясняется сложностью интерпретации текстовых данных на выходе из «Модуля ИИ», если системе необходимо совершить какое-то дополнительное действие, кроме выдачи текстового сообщения. Для решения этой проблемы все входные текстовые сообщения обрабатывались, в первую очередь, «Модулем шаблонной логики 1», который искал в сообщении известные системе команды, определял уровень доступа пользователя к этой команде. В случае если доступные команды не были выявлены, сообщение отправлялось в «Модуль ИИ», который, с помощью нейронной сети типа LSTM, генерировал текстовый ответ. Второй «Модуль шаблонной логики» служил для выполнения тех команд, которые «одобрил» первый.

В основные задачи ГИСУМР не входит «поддержание беседы» с пользователем, следовательно, для обеспечения более корректной и эффективной, с точки зрения затрат вычислительных ресурсов, работы интерпретатора, его архитектуру следует привести к виду, изображенному справа на рисунке 5.

Таким образом, все сигналы, поступающие на вход системы, а именно данные сенсоров робота и команды пользователя, будут обрабатываться нейронной сетью, а ее выходную информацию будет декодировать модуль шаблонной логики, основываясь на контексте команды и сенсоров, анализировать адекватность принятия решения и выстраивать последовательность действий, которые должен выполнить робот. В общем виде архитектура модуля шаблонной логики (или модуля принятия решения) выглядит подобно предложенной Гавриловым А.В. в статье [3] (рисунок 15).

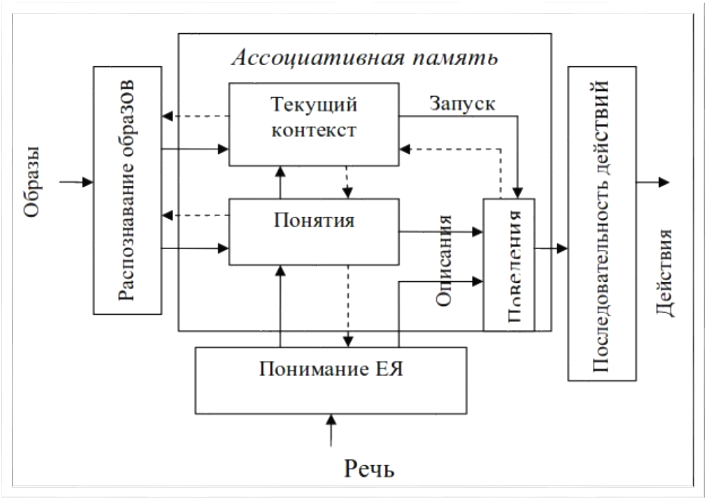


Рисунок 15 - архитектура интерпретатора ГИСУМР

В общем виде архитектуру интерпретатора системы можно представить так, как изображено на рисунке 7.

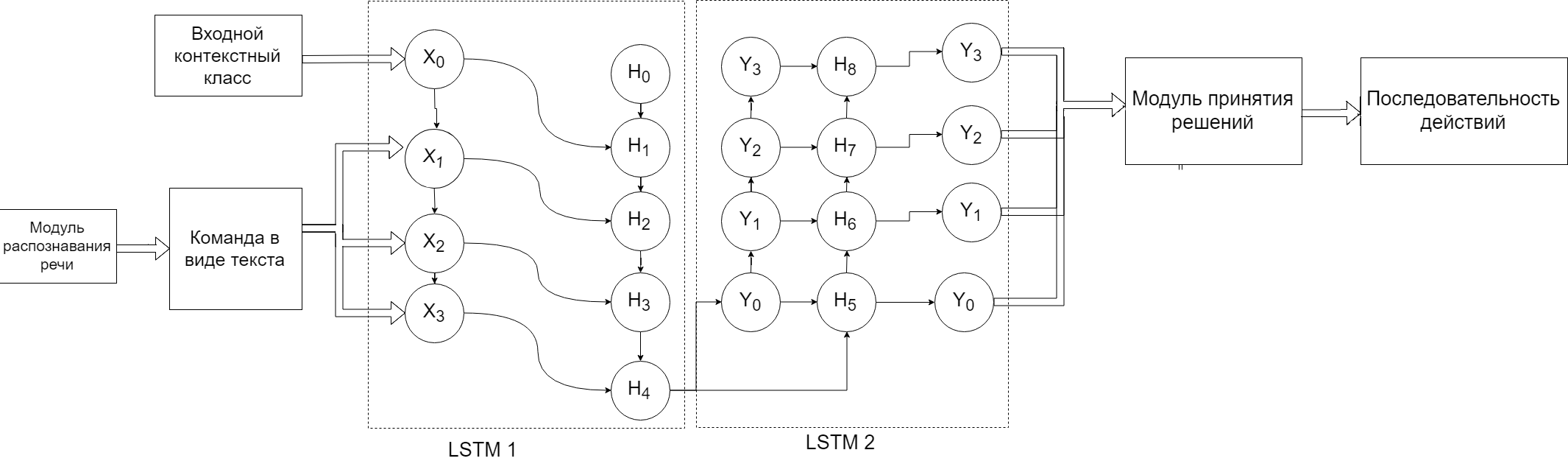


Рисунок 16 - общая архитектура интерпретатора ЕЯ

Использование LSTM позволяет учитывать только внутренний контекст фразы (расположение токенов относительно друг-друга), но это не решает проблему распознавания внешнего контекста диалога, а только корректность построения последовательности действий в рамках текущей задачи, без учета внешних факторов.

*Внешний контекст команды* – это информация о предыдущих обработанных командах, данные сенсоров робота и последние выполненные действия. Внешний контекст фразы жестко привязывается к заданному контексту и снизит количество шума при ее обработке.

## Результаты и выводы по главе

Во второй главе были описаны основные методы, использованные в разработке интерпретатора ограниченного ЕЯ для ГИСУМР. Метод кодирования Word2Vec в паре с моделью нейронной сети LSTM хорошо зарекомендовали себя в области машинного перевода, что является подобным задаче интерпретации команд.

Была разработана архитектура интерпретатора, опираясь на существующие работы, с учетом их недостатков.

# Разработка программной системы, реализующей ГИСУМР

## Общая информация о Unreal Engine 4

Unreal Engine – один из самых популярных графических пакетов для моделирования трехмерных изображений, видеороликов, и разработки видеоигр для различных платформ. Свою популярность пакет завоевал благодаря своей открытости для независимых разработчиков и богатым функциональным возможностям, позволяющим создавать фотореалистичное изображение.

Unreal широко используется и вне игр, в таких областях, как архитектура, дизайн и кинопроизводство - Джон Фавро использовал Unreal Engine 4 для предварительной визуализации и производственных потоков в сериале «Star Wars. Mandalorian» и ремейка Диснеевского мультфильма «The Lion King».

Unreal Engine выпущен в 1998 году. В актуальной на данный момент четвертой версии пакета, выпущенной в 2014 году, UE4 отказался от своего прежнего собственного языка сценариев в пользу C++ и заменил свою систему визуального скриптинга Blueprints на новую.

Система визуального скриптинга Blueprints в Unreal Engine представляет собой законченную систему сценариев, основанную на концепции использования интерфейса на основе блок-схем для создания элементов игрового процесса из редактора Unreal. Как и во многих распространенных языках сценариев, он используется для определения объектно-ориентированных (ОО) классов или объектов в движке.

Эта система чрезвычайно гибкая и мощная, поскольку она позволяет дизайнерам использовать практически весь спектр концепций и инструментов, обычно доступных только программистам. Кроме того, специфичная для Blueprint разметка, доступная в реализации C ++ в Unreal Engine, позволяет программистам создавать базовые системы, которые могут быть расширены дизайнерами.

Гибкость UE4 является одной из основных причин его популярности - независимо от сложности проекта, он может справиться с задачей.

Доступ ко всему исходному коду C ++ для движка является одной из причин гибкости UE4. Код можно скачать с GitHub , вы можете настроить его, и вам не нужно за него платить.

В дополнение к гибкости и свободе, UE4 упрощает быструю итерацию, что полезно при запуске нового проекта.

Программный пакет имеет существенные недостатки только при разработке проектов для разных платформ, например смартфоны, игровые консоли и т.д., при которой необходимо учитывать индивидуальные настройки и вычислительные возможности устройства, на котором будет осуществляться запуск. Но для персональных компьютеров UE4 является самым удобным инструментом для моделирования интерактивной трехмерной анимации. Именно поэтому он был выбран для моделирования результатов работы ГИСУМР.

## Построение тренировочной выборки для нейронной сети интерпретатора

Главной задачей нейронной сети служит интерпретацией текстовой команды в вектор индексов последовательности действий, которые необходимо совершить роботу.

Для построения фразы (команды, последовательности действий), все вектора приводятся к одной размерности (максимальной размерности фразы), при этом недостающие элементы дополняются ключевыми словами-заполнителями. Кроме того, так же как в работе [] входной вектор принимает реверсивный вид, то есть зеркально переворачивается так, что его последний элемент становится первым, а первый последним. Необходимость такого формата входных данных обоснована в упомянутой статье.

Это может выглядеть так, как представлено в таблицах 1-2:

Таблица 2 – общий вид входных данных

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Вектор фразы | | | | | | | | | | | |
|  | Контекст | Команда | | | | | | | | | | |
| Исходный вид | передвижение | И | Д | И |  | В | П | Е | Р | Е | Д | - |
| Кодированный вид | 27 | 8 | 5 | 8 | 1 | 4 | 14 | 6 | 15 | 6 | 5 | 29 |
| Реверсивный вид | 29 | 5 | 6 | 15 | 6 | 14 | 4 | 1 | 8 | 5 | 8 | 27 |

Таблица 3 - общий вид выходных данных

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Вектор фразы | | | | | | | | | | | |
|  | Субьект  выполнения | Действия | | | | | | | | | | |
| Исходный вид | Система | Закончить\_все | Идти\_вперед | Конец\_выолнения | - | - | - | - | - | - | - | - |
| Кодированный вид | 32 | 45 | 48 | 50 | 29 | 29 | 29 | 29 | 29 | 29 | 29 | 29 |

В таблицах приведен вид входных и выходных данных в случае, когда за токены команды приняты символы. Интерпретатор имеет один словарь, куда входят классы символов команды, контекстов, действий, субьектов взаимодействия. Все классы проиндексированы по признаку строковой сортировки словаря.

Исходные данные для обработки собираются системой в виде, представленном в таблице 3.

Таблица – пример первичного набора необработанных данных

|  |  |  |
| --- | --- | --- |
| Команда | Ответ | Контекст |
| Иди вперед | система -> идти\_вперед | любой |
| Иди вперед | система -> идти\_вперед | передвижение |
| Двигай вперед | система -> идти\_вперед | передвижение |
| Двигай вперед | система -> идти\_вперед | любой |
| Вперед | система -> идти\_вперед | передвижение |
| Вперед | система -> идти\_вперед | любой |
| Поверни налево | система -> повернуть\_налево | передвижение |
| Поверни налево | система -> повернуть\_налево | любой |
| налево | система -> повернуть\_налево | передвижение |
| налево | система -> повернуть\_налево | любой |
| Сверни налево | система -> повернуть\_налево | передвижение |
| Сверни налево | система -> повернуть\_налево | любой |
| Поверни направо | система -> повернуть\_направо | любой |
| направо | система -> повернуть\_направо | передвижение |
| направо | система -> повернуть\_направо | любой |
| сверни направо | система -> повернуть\_направо | любой |
| сверни направо | система -> повернуть\_направо | передвижение |
| Иди назад | система -> закончить\_все -> вернуться\_в\_предыдущую\_локацию | любой |
| Иди назад | система -> закончить\_все -> вернуться\_в\_предыдущую\_локацию | передвижение |
| Иди задом | система-> закончить\_все -> идти\_задом | любой |
| Иди задом | система -> закончить\_все -> идти\_задом | передвижение |
| Вернись назад | система -> закончить\_все -> вернуться\_в\_предыдущую\_локацию | передвижение |
| Вернись назад | система -> закончить\_все -> вернуться\_в\_предыдущую\_локацию | любой |
| Иди задом | система -> идти\_задом | любой |
| Стой | система -> закончить\_все | любой |
| Стой | система -> закончить\_все | передвижение |
| Стоп | система -> закончить\_все | любой |
| Стоп | система -> закончить\_все | передвижение |
| Прекрати | система -> закончить\_все | любой |
| Прекрати | система -> закончить\_все | передвижение |

Продолжение Таблицы

|  |  |  |
| --- | --- | --- |
| Команда | Ответ | Контекст |
| Стоп | система -> закончить\_все | любой |
| Прекрати | система -> закончить\_все | любой |
| Выведи необработанные данные | система -> вывод\_цсв\_предобработки | любой |
| Обучение модели | система -> закончить\_все -> обучение | любой |
| Выведи список команд | система -> вывод\_списка\_команд | любой |
| Доложить о состоянии системы | система -> вывод\_статуса | любой |
| Доложи | система -> вывод\_статуса | любой |
| Докладывай | система -> вывод\_статуса | любой |

## Анализ обучения моделей LSTM в различных индексациях словаря

Для обучения основы интерпретатора ограниченного ЕЯ были использованы 2 одинаковые модели нейронной сети LSTM, размером 256 юнитов, но с разной индексацией входных данных. В первом случае команда рассматривалась как набор слов, во втором как набор символов, а выходные данные у обеих моделей одинаковы.

Результаты обучения представлены в таблице 4.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вид модели | Количество эпох обучения | Затраченное время | Средняя погрешность | Скорость обучения эпохи | Размер словаря |
| LSTM 1 (Слова) | 13 500 | 2.5 ч | 0,4960 | 0,56 | 41 |
| LSTM 2 (Символы) | 18 500 | 3 ч | 0,4755 | 0,6 | 54 |

Наилучшую погрешность показала вторая модель (0,47). Учитывая ничтожно малый размер тренировочной выборки и малое количество выходных классов, по сравнению с моделью, описанной в работе [5], это достаточно хороший результат. Диалоговая система, имея в расположении более 2000 диалогов и словарь, состоящий из 5451 слова, имела погрешность равную 0,35. Вполне вероятно, что после увеличения размера тренировочной выборки, результаты обучения интерпретатора ограниченного ЕЯ улучшатся, и даже превзойдут, те результаты, что показала диалоговая система, благодаря использованию символов, вместо слов.

Обе модели прекрасно показывают себя в диалогах, на которых они были обучены, но стоит изменить последовательность слов, или их вариации, результаты принятия решения будут совершенно иные, и не всегда правильные. Эту проблему можно решить одним из следующих способов:

* Вводить в тренировочную выборку системы известные слова в различных вариациях, что достаточно трудоемко для пользователя.
* Обрабатывать эти результаты простыми алгоритмами и отсеивать нежелательные (например, движение вперед, когда впереди находится препятствие).

На выходе из нейронной сети строится первичная последовательность действий, которая отправляется не на выполнение, а в «Модуль принятия решений», который обрабатывает эти результаты и, по средствам простых шаблонных алгоритмов, анализирует насколько это решение адекватно, в контексте предыдущего диалога и данных сенсоров.

## Итеграция ГИСУМР в Unreal Engine 4

Модель нейронной сети, реализованная на языке Python, импортирована в UE4 в виде Blueprint объекта, с помощью которого система управляет сценариями, прописанными на Python. Главной задачей этих сценариев является обработка и индексация словаря, перевод текста в векторы индексов, и обучение модели нейронной сети.

При первом запуске системы с обученным интерпретатором на текстовых командах, необходим еще один этап обучения, чтобы исследовать окружающую среду робота. Во время этой операции система запоминает окружение, включая интерактивные объекты и название комнат в доме, что может быть использовано в таких командах, как например «иди в зал». Все объекты на локации помечены особым тегом, которые считывает робот. Физическим аналогом тегов могут служить наклейки QR-кодами. Робот оборудован сенсорами, изображенных на рисунке 17, и когда объект с тегом впервые попадает в его поле зрения, данные о его местоположении заносятся в память.

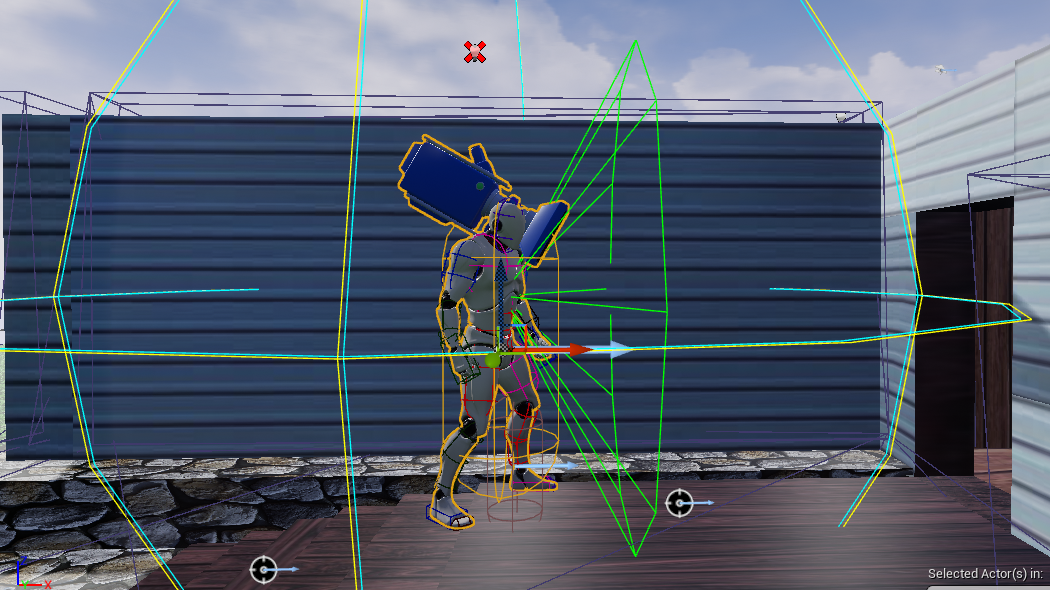


Рисунок - Сенсоры робота

Для построения контекста, робот, так же, находятся массивы данных об интерактивных объектах, которые задают правила взаимодействия с ними. Кроме того, для построения контекста используются набор флагов использования приводов, например есть ли свободная рука для выполнения новой команды, а так же буферный вектор, в котором находятся данные о предыдущей команде. Буферный вектор используется если команда была прервана и необходимо продолжить выполнение незавершенных действий.

После исследования локации робот будет готов к выполнению команд.

С помощью терминала пользователя, текст команды отправляется в обученную нейронную сеть, возвращающую вектор индексов действий, и система принимает решение какие действия робот должен выполнять, дополняет эту информацию условиями выполнения, такими как продолжительность, количество и правила остановки и т.д., после чего модель робота начинает выполнение.

Выполнение заключается в последовательном переборе элементов вектора индексов действий, декодированием индекса в класс действия и его запуске (рисунок 18).

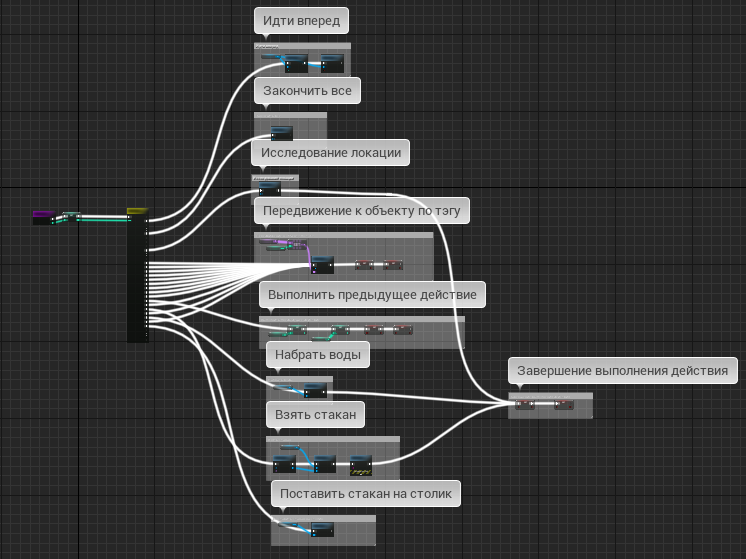


Рисунок - Blueprint сценарий выполнения вектора действий

В контроллере искусственного интеллекта робота каждый кадр выполняется последовательность действий, изображенная в сценарии на рисунке 19 сверху вниз. В этом сценарии так же проходит проверка особых действий, которые требуют дополнительных условий. Если команда недопустима к выполнению, например в на траектории движения робота препятствие, или системе не известна такая команда, то об этом будет сообщено пользователю. Если дана такая абстрактная команда, как «принеси воды», то система должна определить, что для этого необходимо найти сосуд «стакан», источник воды и место, куда нужно принести наполненный стакан «журнальный столик». Если же робот не закончил выполнение этой команды, а уже дана следующая команда «нет, сначала лекарство», то возникает решаемая анафорическая неточность, при которой выполняются не уточненные команда «остановить текущее действие», «запомнить незавершенные действия», выполнить команду «принеси» установив явно указанный объект.

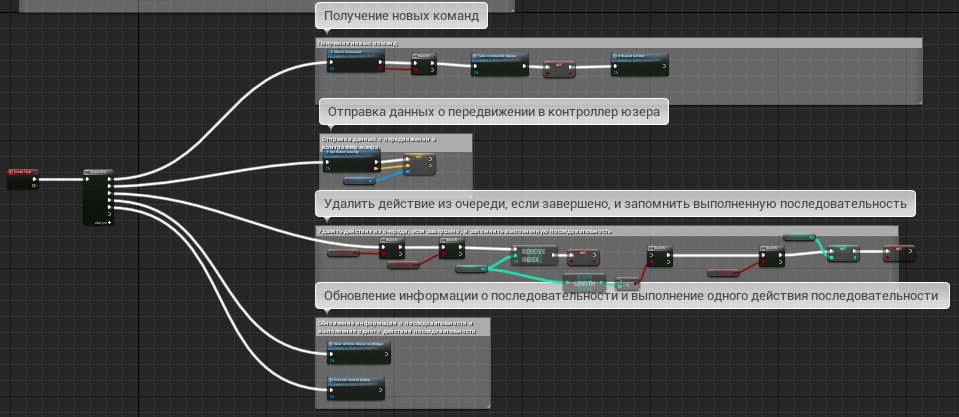


Рисунок - Сценарий Blueprint постоянно выполняющихся действий

В случае, когда робот выполняет команду неправильно, пользователь может дать команду «ошибка», при которой система завершит выполнения действий и отложит данные о них в лог ошибок, которые можно исправить позднее.

Исправление ошибок осуществляется уточнением системе какую последовательность действий нужно выполнить. Эта информация будет отправлена в лог обучения, которая будет использована в процессе переобучения интерпретатора и обновления списка допустимых команд.

К контроллекру искусственного интеллекта робота привязана управляемая модель робота, которая возвращает информацию об его окружении и состоянии.

Модель робота имеет сенсоры, с помощью которых он ориентируется в пространстве. Радиусы действий сенсоров показаны на рисунке 19.

## Результаты и выводы по главе

В третьей главе приведена общая информация о среде моделирования Unreal Engine 4.

Описаны методы форматирования первичных данных для обучения нейронной сети.

Проанализирована часть интерпретатора ограниченного ЕЯ с двумя моделями нейронной сети LSTM: первая модель (LSTM 1) представляет команду как набор слов, а вторая (LSTM 2) как набор символов.

Проведен сравнительный анализ результатов работы обеих моделей нейронных сетей, в результате которого было принято решение использовать модель LSTM 2. Вторая модель показала результаты лишь немного выше, чем представление команды, как набор слов в LSTM 1, но предполагается, что качество работы LSTM 2 улучшится с увеличением тренировочной выборки. Кроме того, использование LSTM 2, позволит реже пополнять словарь интерпретатора, а, следовательно, и реже производить его полное переобучение.

Описана интеграция интерпретатора в среду Unreal Engine 4.

# Проведение экспериментов в среде моделирования Unreal Engine 4

После запуска среды моделирования пользователь берет под управление свободную камеру с пользовательским терминалом, а робот начинает исследование локации (рисунок 20).

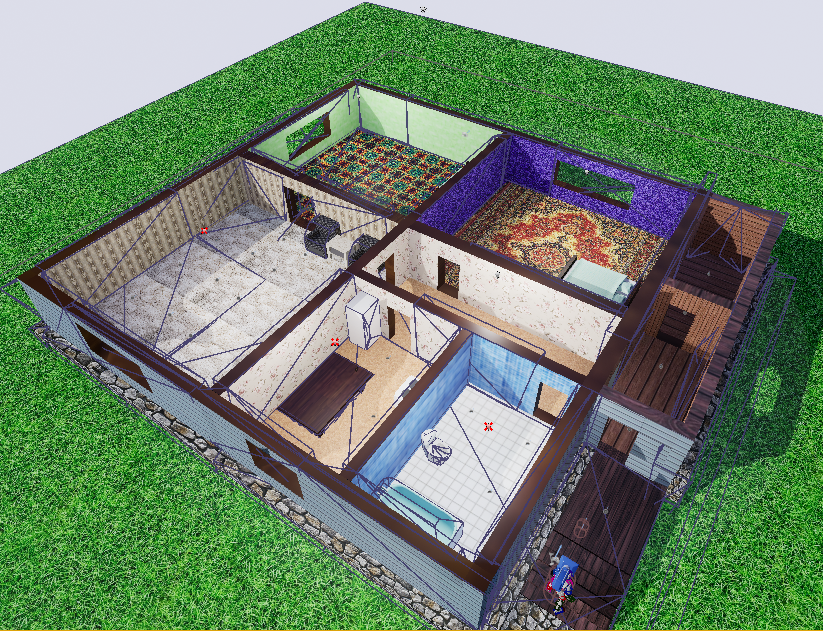


Рисунок – Локация среды моделирования



Рисунок - Интерфейс пользователя среды моделирования

Интерфейс пользователя представлен на рисунке 21, где:

1. Терминал пользователя для ввода команд;
2. камера, отображающая то, что видит робот;
3. список комнат, которые исследовал робот;
4. список исследованных предметов, с которыми робот может взаимодействовать;
5. очередь действий, которые робот выполняет в данный момент.

Проведем эксперименты после исследования роботом локации и занесем результаты в таблицы.

## Эксперимент 1

В первом эксперименте от робота требуется только принести стакан воды. Предполагается, что команда была подана из зала, первичные данные изображены на рисунке 22.



Рисунок - первичные данные для эксперимента 1

Здесь:

1. Предполагаемый источник команды (пользователь)
2. Начальное положение робота
3. Пустой стакан
4. Кухонная раковина
5. Журнальный столик, на который необходимо принести стакан с водой

Поэтапное выполнение команды представлено в таблице 6.

Таблица - Эксперимент 1 (подношение стакана воды)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Команда | Скриншот | Описание |
| 1 | Принеси воды |  | Робот перемещается к стакану, расположенному в кухне, и берет его в свободную руку. |
| 2 |  | Затем робот отправляется к ближайшему источнику воды, в данном случае к раковине в кухне, и набирает воду в стакан. |
| 3 |  | После чего, идет в зал к журнальному столику и ставит стакан на него. |

## Эксперимент 2

Во втором эксперименте проверим, как система решит анафорическую неясность во время выполнения другой команды. Сначала будет дана команда «принеси воды», а потом сразу же вторая команда «нет, сначала лекарство». Начальные данные приведены на рисунке 23.



Рисунок - первичные данные для эксперимента 2

Здесь все данные аналогичны рисунку 22, с добавлением точки 6, где располагается лекарство.

Поэтапное выполнение команды приведено в таблице 7.

Таблица Эксперимент 2 (Анафорическая неясность)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Команда | Скриншот | Описание |
| 1 | Принеси воды |  | Робот перемещается к стакану, и берет его в свободную руку. |

Продолжение таблицы 7

|  |  |  |  |
| --- | --- | --- | --- |
| № | Команда | Скриншот | Описание |
| 2 |  |  | Затем робот набирает воду в стакан. |
| 3 | Нет, сначала лекарство |  | Система принимает новую команду, прекращает выполнение текущее команды, но запоминает не выполненные действия. Робот начинает выполнять следующую команду и, держа стакан с водой в правой руке, немедленно перемещается к лекарству. |
| 4 |  |  | Робот берет лекарство в свободную (правую) руку и отправляется к журнальному столику. |

Продолжение таблицы 7

|  |  |  |  |
| --- | --- | --- | --- |
| № | Команда | Скриншот | Описание |
| 5 |  |  | Робот ставит лекарство на столик. |
| 6 |  |  | Затем возвразается к предыдущей незаконченной команде и ставит на столик стакан. |

## Результаты и выводы по главе

В четвертой главе были проведены эксперименты, доказывающие работоспособность представленных методов. После предварительного обучения нейронной сети, на базовых командах, и исследования локации системой, робот способен выполнять простые команды, с учетом контекста диалога.

# Заключение

В ходе работы были рассмотрены методы разработки систем управления, обеспечивающих понимание ЕЯ. Это требует создание механизмов, отличных от традиционных способов представлений ЕЯ, а системы с естественно-языковыми интерфейсами применяются только в узких предметных областях.

Технологии речевого распознавания нашли свое применение в различных областях. Однако в области систем управления множество проблем все еще остаются не решенными, многие идеи требуют дальнейшего развития. Была произведена попытка рассмотреть проблемы, которые лежат перед разработчиками систем искусственного интеллекта при попытках моделирования понимания. Часть проблем уже решена на хорошем уровне, где-то системы ИИ работают с большей эффективностью, чем человек, где-то едва делаются первые шаги. Проблема машинного понимания, на данный момент, все еще остается очень сложной и связана с крайне широким спектром проблем. Значительный успех в этой области приобрели нейронные сети, но они все еще способны выполнять только узкие задачи и обрабатывать только ограниченный ЕЯ. Однако этого полностью достаточно для создания полнофункциональных ЕЯ-интерфейсов, не требующих глубокого анализа семантики и грамматических особенностей ЕЯ.

Проведен сравнительный анализ современных прототипов систем с ЕЯ-интерфейсами и ПО для обработки ЕЯ. Многие программные инструменты находятся в свободном доступе с открытым исходным кодом, что позволяет использовать их в собственных проектах без юридических ограничений, однако ни один из них не удовлетворяет требованиям ГИСУМР (Гибридная Система Управления Мобильным Роботом). Все сводится к тому, что полностью отвязаться от серверов разработчиков не получится, API обладают ограниченным функционалом, и привязать их к своей модели робота очень проблематично. Для реализации ГИСУМР принято решение разрабатывать собственный интерпретатор ограниченого ЕЯ, который бы полностью подходил для решения поставленных перед системой задач.

Были описаны основные методы, использованные в разработке интерпретатора ограниченного ЕЯ для ГИСУМР. Метод кодирования Word2Vec в паре с моделью нейронной сети LSTM хорошо зарекомендовали себя в области машинного перевода, что является подобным задаче интерпретации команд.

Была разработана архитектура интерпретатора, опираясь на существующие работы, с учетом их недостатков.

Разработана часть интерпретатора ограниченного ЕЯ с двумя моделями нейронной сети LSTM: первая модель (LSTM 1) представляет команду как набор слов, а вторая (LSTM 2) как набор символов.

Проведен сравнительный анализ результатов работы обеих моделей нейронных сетей, в результате которого было принято решение использовать модель LSTM 2. Вторая модель показала результаты лишь немного выше, чем представление команды, как набор слов в LSTM 1, но предполагается, что качество работы LSTM 2 улучшится с увеличением тренировочной выборки. Кроме того, использование LSTM 2, позволит реже пополнять словарь интерпретатора, а, следовательно, и реже производить его полное переобучение.

# Список литературы

1. Andrey V.Gavrilov. Context and Learning based Approach to Programming of Intelligent Equipment // Proceedings of the 8 th Int. Conf. on Intelligent Systems Design and Applications ISDA’08, November 26-28, 2008 Kaohsiung City, Taiwan, 2008 – Pp. 578-582.
2. Andrey V.Gavrilov. New Paradigm of Context based Programming-Learning of Intelligent Agent // Proc. of 1 st Workshop on Networked embedded and control system technologies. In conjunction with 6 th International Conference on Informatics in Control, Automation and Robotics ICINCO-2009, 2-5 July, 2009 - Milan, Italy, 2009 – Pp. 94-99.
3. Badler I., Phillips C. B., and Webber B. L. . 1993. Simulating Humans Computer Graphics Animation and Control. Oxford University Press. N.
4. Badler, I. Palmer M. S., and Bindinganavale R..1999. Animation Control for Real-Time Visual Humans. Comm. of the ACM, pages 65–73.
5. Cassel, T. Bickmore, L. Billinghurst, L. Campbell, K. Chang, H. Vilhjalmsson, and H. Yan. 1999. Embodiment in Conversational Interfaces: Rea.
6. Shinyama, Yusuke, Takenobu Tokunaga, and Hozumi Tanaka. 2000. “Kairai”—software robots understanding natural language. In Proceedings of the Third International Workshop on Human–Computer Conversation , pages 158–163, Bellagio
7. SHRDLU [Электронный источник] : URL : <http://ru.knowledgr.com/00067177/SHRDLU> (дата обращения 27.11.19)
8. Weischedel R., Scha R, etc. «Research and Development in Natural Language Processing at BBN Laboratories in the Strategic Computing Program» , BBN Laboratories, Inc/ Cambrige, NA 02238
9. [Wolfram|Alpha FAQ](http://www.wolframalpha.com/faqs.html). [Электронный источник] : URL : <http://www.wolframalpha.com/faqs.html> (дата обращения 28.11.19)
10. Xuedong, H. Spoken Language Processing: A Guide to Theory, Algorithm and System Development / Huang Xuedong. – New Jersey : Prentice Hall PTR, 2001. – 1008 p. 5.
11. Алимурадов А.К. «Параметры и классификация систем распознавания речи», УДК 621.391; 519.21 Пензенский государственный университет, 2014г.
12. Гаврилов А.В. Гибридные системы управления мобильных роботов. Материалы Межд.конф. "Интеллектуальные системы", г.Дивноморское, 2004.
13. Гаврилов А.В. Диалоговая система подготовки программ для роботов // Glivice, Poland - Automatyka – 1988 - Vol. 99 - Pp. 173-180. 4.
14. Гаврилов А.В. Контекстно-ориентированная гибридная архитектура системы управления интеллектуального робота. Материалы VI Всероссийской научно-технической конференции с международным участием. - Красноярск: ЦНИ "Монография", 2014, с.74-79.
15. Гаврилов А.В., Губарев В.В «Архитектура гибридной системы управления мобильного робота» - 2004
16. Как устроена Алиса. Лекция Яндекса. [Электронный источник] : URL : <https://habr.com/ru/company/yandex/blog/349372/> (дата обращения 25.11.19)
17. Обработка естественного языка [Электронный источник] : URL : <https://cybernetics.wikia.org/ru/wiki/Обработка_естественного_языка> (дата обращения 25.11.19)
18. Пользовательский интерфейс естественного языка. [Электронный источник] : URL : <http://ru.knowledgr.com/06688487/ПользовательскийИнтерфейсЕстественногоЯзыка> (дата обращения 27.11.19)
19. Разработка навыков для Алисы. Опыт работы с голосовыми интерфейсами, советы начинающим [Электронный источник] : URL : <https://habr.com/ru/post/434194/> (дата обращения 28.11.19)
20. Чесебиев И. А. Компьютерное распознавание и порождение речи – М. : Спорт и культура, 2008. – 128 с.
21. Элиза [Электронный источник] : URL : <https://habr.com/ru/post/130877/> (дата обращения 27.11.19)
22. Bengio Y., Ducharme R., Vincent P. A Neural Probabilistic Language Model // Journal of Machine Learning Research, 2003, vol. 3. — P. 1137–1155.
23. Sahlgren M*.* [The Distributional Hypothesis. From context to meaning](http://linguistica.sns.it/RdL/20.1/Sahlgren.pdf) (англ.) // Distributional models of the lexicon in linguistics and cognitive science (Special issue of the Italian Journal of Linguistics), Rivista di Linguistica: журнал. — 2008. — Vol. 20, no. 1. — P. 33-53.
24. Гаврилов А. В. Контекстно-ориентированная гибридная архитектура системы управления интеллектуального робота = Context-oriented hybrid architecture of control system of intelligent robot / А. В. Гаврилов // Робототехника и искусственный интеллект: материалы 6 Всерос. науч.-техн. конф. с междунар. участием, Железногорск, 13 дек. 2014 г. – Красноярск: ЦНИ «Монография», 2014. – С. 74-79. - 200 экз. - ISBN 978-5-905284-45-8.
25. Николенко С.И., Кадурин А.А., Архангельская Е.О. «Глубокое обучение. Погружение в мир нейронных сетей». Санк-Петербург 2017, 480с.
26. Шпаков В.К. «Использование искусственных нейронных сетей для разработки интеллектуальной диалоговой системы» // Сборник статей XX Всероссийской научно–технической конференции «Проблемы социального и научно–технического развития в современном мире»с международным участием. -2018 г. – С. 54.
27. Unreal Engine 4 Documentation [Электронный источник] [Руководство по использованию пакета UE4] : URL : <https://docs.unrealengine.com/en-US/index.html> (дата обращения 01.05.2019)