ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Дисциплина «Интеллектуальные системы»

**ОТЧЕТ**

по лабораторной работе №7

на тему

«**Программные среды разработки искусственного интеллекта**»

Выполнил:

студент группы 3540901/02001

Бараев Д. Р.

«\_\_» \_\_\_\_\_\_\_\_\_\_ 2020г., \_\_\_\_\_\_\_\_\_\_

(подпись)

(подпись)

Проверил:

Е. Н. Бендерская

«\_\_» \_\_\_\_\_\_\_\_\_\_ 2020г., \_\_\_\_\_\_\_\_\_\_

Санкт-Петербург 2020

**Содержание**

[Введение 3](#_Toc56175409)

[1. История развития программных средств для разработки ИИ 4](#_Toc56175410)

[1.1 Lisp 4](#_Toc56175411)

[1.2 Prolog 6](#_Toc56175412)

[2. Средства создания систем ИИ 8](#_Toc56175413)

[2.1 Инструментальные средства разработки экспертных систем 9](#_Toc56175414)

[2.2 Языки программирования ИИ 14](#_Toc56175415)

[2.3 Фреймворки и библиотеки для разработки ИИ 16](#_Toc56175416)

[2.4 Дополнительные инструменты, участвующие в разработке ИИ 21](#_Toc56175417)

[Заключение 22](#_Toc56175418)

[Список источников 23](#_Toc56175419)

## Введение

В последние десятилетия искусственный интеллект вторгается во все сферы деятельности, становится средством интеграции наук. Программные средства, базирующиеся на технологии и методах искусственного интеллекта, получили значительное распространение в мире. Интенсивные исследования по созданию единого информационного пространства, создающего условия для совместной дистанционной работы на основе баз знаний, сегодня начали проводить все развитые в экономическом отношении страны.

Так же на сегодняшний день существует множество инструментов, предназначенных для разработки искусственного интеллекта. Языки программирования искусственного интеллекта и интеллектуальных систем появились достаточно давно. Например, prolog был создан в 1970 году, а lisp -1958 год, один из самых первых языков высокого уровня программирования. Язык программирования Пролог, основанный на идеях и методах математической логики, изначально создавался для разработки приложений искусственного интеллекта. В среде Visual Prolog эффективно программируются такие приложения, как базы знаний, экспертные системы, естественно-языковые интерфейсы и интеллектуальные системы управления информацией. Высокий уровень абстракции, возможность представления сложных структур данных и моделирования логических отношений между объектами позволяют решать задачи различных предметных областей. В настоящие время написать ИИ или какую-либо интеллектуальную систему можно на множестве языков программирования используя специальные Фреймворки и библиотеки для конструирования ИС. Так же существует большое количество сервисов помогающие разрабатывать ИС: базы знаний, облачные вычисления, хранилища на которых можно развернуть систему, базы знаний и т.д.

## История развития программных средств для разработки ИИ

На начальном этапе развития ИИ языков и систем, ориентированных специально на создание прикладных систем, основанных на знаниях, не существовало. С одной стороны, в то время еще не оформился сам подход, в котором центральное место отводилось бы изложению теории в форме программ, а с другой — сама область ИИ только зарождалась как научное направление. Немаловажным было и то, что появившиеся к тому времени универсальные языки программирования высокого уровня казались адекватным инструментом для создания любых, в том числе и интеллектуальных, систем. Однако сложность и трудоемкость разработки здесь настолько велики, что практически полезные интеллектуальные системы становятся недоступными для реализации. Для этого были разработаны языки и системы обработки символьной информации, которые на несколько десятилетий стали основным инструментом программирования интеллектуальных систем.

До недавнего времени наиболее популярным базовым языком реализации систем ИИ вообще и ЭС, в частности, был LISP [1]

## 1.1 Lisp

Автором Лиспа является Джон Маккарти, на период создания языка, работавший в Массачусетском технологическом институте (MIT) в должности профессора по связи. Вместе с Марвином Мински он занимался работами по искусственному интеллекту, в связи с чем и возникла потребность в создании языка программирования, адекватного задачам, решаемым в этой области. Работа по созданию языка была проделана Маккарти в MIT в период с 1958 по 1963 год, после чего он перешёл в Стенфордский университет в Калифорнии, где получил должность «профессор по искусственному интеллекту».

Основой для Лиспа послужил ранний язык IPL, разработанный Ньюэллом, Шоу и Саймоном. IPL был языком обработки списков и предназначался для реализации проекта «Логик-теоретик» — системы искусственного интеллекта, предназначенной для автоматического вывода теорем математической логики. IPL был довольно низкоуровневым языком, но в нём уже были реализованы такие базовые идеи, как единый механизм хранения программ и данных в виде списков — иерархических структур элементов, связанных ссылками (сама идея списочного представления знаний была позаимствована из исследований по психологии и ассоциативной памяти), а также идея динамического распределения памяти. После ознакомления в 1956 году с IPL у Маккарти появилась идея реализовать обработку IPL-списков в Фортране, который как раз в это время проектировался в IBM (причём под ту же систему IBM 704, с которой Маккарти работал в MIT), но эта идея так и не была реализована. Позже Маккарти принял участие в работе «комитета по языку высокого уровня», разрабатывавшего Алгол, но и там его предложения были встречены холодно. В результате Маккарти пришёл к мысли о необходимости создания нового языка программирования.

Первоначально Маккарти сформулировал списочный формализм для описания данных (S-выражения) и основанный на нём же механизм описания лямбда-выражений, что позволило записывать программы в виде наборов функций, представленных в списочной форме. Как писал позже Маккарти, изначально он планировал применять для записи программ отдельный формализм, отличающийся от S-выражений, но это оказалось излишним. Когда с помощью своей списочной записи Маккарти описал алгоритм функционирования интерпретатора нового языка (формализм, который впоследствии стал известен как «Лисп на Лиспе»), Стив Рассел заметил, что теперь для создания реального работающего интерпретатора достаточно просто перевести эту запись в машинный код. Маккарти отнёсся к этой идее скептически, но Рассел действительно проделал данную работу и получил первый интерпретатор Лиспа для компьютера IBM 704. В дальнейшем идея написания транслятора языка на нём самом многократно использовалась, и не только в функциональных и логических языках, но и в императивных.

Исторически первой реализацией Лиспа, включающей все современные базовые элементы языка, был интерпретатор, работавший на IBM 704, появившийся в октябре 1958 года. Это, кстати, позволяет говорить о Лиспе как об одном из двух старейших языков высокого уровня, которые находятся в употреблении с момента создания до настоящего времени (первый — Фортран). Помимо этого, Лисп сохранил первенство ещё в одном отношении. Дело в том, что активная работа с динамическими списками сделала невозможным ручное управление памятью, которое в императивных языках отчасти сохраняется по сей день. Создание новых списочных ячеек и списков и выход из использования имеющихся при работе лисп-программы происходят настолько активно, что практически невозможно обойтись без системы автоматического управления памятью, которая контролировала бы использование ранее созданных в памяти объектов и периодически удаляла те из них, использование которых прекратилось, то есть системы сборки мусора. Маккарти пришлось реализовать эту систему, благодаря чему Лисп, помимо прочего, является ещё и самым старым из применяемых сегодня языков программирования с автоматическим управлением памятью и сборкой мусора.

Позднее были созданы реализации для IBM 7090, в дальнейшем — для серий IBM 360 и 370. Компьютеры IBM оказались неудобны для работы в интерактивном режиме, вследствие чего в конце 1950-х годов небольшая группа разработчиков, в том числе работавших ранее в IBM, выделилась в самостоятельную компанию Digital Equipment Corporation (DEC). Первым её изделием стал компьютер PDP-1, изначально ориентированный на интерактивный режим работы. На этой машине в 1960 году была реализована интерактивная система «Lisp 1», включающая в себя интегрированные интерпретатор, редактор исходного кода и отладчик, позволявшая выполнять весь цикл работ над программой непосредственно в системе. По сути, это была первая «среда программирования» в том смысле, который вкладывается в это понятие сейчас. Тогда же в журнале «Communications of ACM» вышла статья Маккарти «Recursive Functions of Symbolic Expressions and their Computation by Machine.», в которой Лисп был описан в виде алгебраического формализма на самом Лиспе. Статья стала классической, а формализм типа «Лисп на Лиспе» с тех пор стал одним из наиболее употребимых в литературе по теории программирования. Ещё одним технологическим новшеством, появившимся в связи с реализацией системы «Lisp 1» был изобретённый Маккарти механизм, позволявший запускать интерпретатор Лиспа одновременно с выполнением обычных вычислительных работ в пакетном режиме (то, что сейчас известно как «система разделения времени»).

К 1962 году была готова следующая версия оригинальной лисп-системы «Lisp 1.5», в которой были устранены обнаруженные за время эксплуатации недостатки первой версии. Её описание было выпущено в издательстве «MIT Press» в виде отдельной книги. Поскольку руководство включало описание реализации системы, оно стало основой для создания лисп-систем для множества других компьютеров как в США, так и за её пределами. [2]

## 1.2 Prolog

Начало истории языка относится к 1970-м годам. Будучи декларативным языком программирования, Пролог воспринимает в качестве программы некоторое описание задачи или баз знаний и сам производит логический вывод, а также поиск решения задач, пользуясь механизмом поиска с возвратом и унификацией.

Интерес к Прологу поднимался и затихал несколько раз, энтузиазм сменялся жёстким неприятием. Наиболее высоко был поднят интерес к языку Пролог, как к языку будущего, во время разработок японской национальной программы компьютеры пятого поколения в 1980-х годах, когда разработчики надеялись, что с помощью Пролога можно будет сформулировать новые принципы, которые приведут к созданию компьютеров более высокого уровня интеллекта.

Язык Пролог в 1980-х годах был включён в ряд советских вузовских и школьных учебников информатики для изучения элементов математической логики, принципов логического программирования и проектирования баз знаний и моделей экспертных систем. С этой целью на IBM PC и ряде советских школьных компьютеров были реализованы учебные русскоязычные интерпретаторы Пролога.

В языке Пролог факты описываются в форме логических предикатов с конкретными значениями. Правила вывода описываются логическими предикатами с определением правил логического вывода в виде списка предикатов над базами знаний и процедурами обработки информации.

В настоящее время Пролог, несмотря на неоднократные пессимистические прогнозы, продолжает развиваться в разных странах и вбирает в себя новые технологии и концепции, а также парадигмы императивного программирования. В частности, одно из направлений развития языка (в том числе и в России) реализует концепцию интеллектуальных агентов. [3]

## Средства создания систем ИИ

Общие требования к современным инструментальным средствам создания систем искусственного интеллекта, следующие:

1. Специализация.

Переход от разработки инструментальных средств общего назначения к проблемно/предметно специализированным средствам, что обеспечивает сокращение сроков разработки приложений, увеличивает эффективность использования инструментария, упрощает и ускоряет работу эксперта, позволяет повторно использовать информационное и программное обеспечение (объекты, классы, правила, процедуры).

1. Использование языков традиционного программирования и рабочих станций.

Переход от систем, основанных на языках искусственного интеллекта (Lisp, Prolog и т.п.), к языкам традиционного программирования (С, С++ и т.п.) упростил "интегрированность" и снизил требования приложений к быстродействию и емкости памяти. Использование рабочих станций вместо ПК резко увеличило круг возможных приложений методов искусственного интеллекта.

1. Интегрированность.

Разработаны инструментальные средства искусственного интеллекта, легко интегрирующиеся с другими информационными технологиями и средствами (с CASE, СУБД, контроллерами, концентраторами данных и т.п.).

1. Открытость и переносимость.

Разработки ведутся с соблюдением стандартов, обеспечивающих данные характеристики.

1. Архитектура клиент/сервер.

Разработка распределенной информационной системы в данной архитектуре позволяет снизить стоимость оборудования, используемого в приложении, децентрализовать приложения, повысить надежность и общую производительность, поскольку сокращается объем информации, пересылаемой между ЭВМ, и каждый модуль приложения выполняется на адекватном оборудовании.

Среди специализированных систем, основанных на знаниях, наиболее значимы экспертные системы реального времени, или динамические экспертные системы. Экспертные системы реального времени — одно из главных направлений искусственного интеллекта.

Классы задач, решаемых экспертными системами реального времени, таковы: мониторинг в реальном масштабе времени, системы управления верхнего уровня, системы обнаружения неисправностей, диагностика, составление расписаний, планирование, оптимизация, системы - советчики оператора, системы проектирования.[4]

## 2.1 Инструментальные средства разработки экспертных систем

Экспертные системы (ЭС) - это системы искусственного интеллекта (интеллектуальные системы), предназначенные для решения плохоформализованных и слабоструктурированных задач в определенных проблемных областях, на основе заложенных в них знаний специалистов-экспертов. В настоящее время ЭС внедряются в различные виды человеческой деятельности, где использование точных математических методов и моделей затруднительно или вообще невозможно. К ним относятся: медицина, обучение, поддержка принятия решений и управление в сложных ситуациях, деловые различные приложения и т. д.

Основными компонентами ЭС являются базы данных (БД) и знаний (БД), блоки поиска решения, объяснения, извлечения и накопления знаний, обучения и организации взаимодействия с пользователем. БД, БЗ и блок поиска решений образуют ядро ЭС.

Для конструирования ЭС используются различные инструментальные средства: универсальные языки программирования, языки искусственного интеллекта, инструментальные системы и среды и системы-оболочки. Системы-оболочки являются наиболее простым средством формализации (автоформализации) экспертных знаний, практически не требующие участия посредников в лице инженера по знаниям или программиста при их использовании. Инженер по знаниям только помогает эксперту выбрать наиболее подходящую для его проблемной области оболочку.

Известны три основные разновидности исполнения экспертных систем:

* Экспертные системы, выполненные в виде отдельных программ, на некотором алгоритмическом языке, база знаний которых является непосредственно частью этой программы. Как правило, такие системы предназначены для решения задач в одной фиксированной предметной области. При построении таких систем применяются как традиционные процедурные языки PASCAL, C и др., так и специализированные языки искусственного интеллекта LISP, PROLOG.
* Оболочки экспертных систем - программный продукт, обладающий средствами представления знаний для определенных предметных областей. Задача пользователя заключается не в непосредственном программировании, а в формализации и вводе знаний с использованием предоставленных оболочкой возможностей. Недостатком этих систем можно считать невозможность охвата одной системой всех существующих предметных областей. Примером могут служить ИНТЕРЭКСПЕРТ, РС+, VP-Expert.
* Генераторы экспертных систем - мощные программные продукты, предназначенные для получения оболочек, ориентированных на то или иное представление знаний в зависимости от рассматриваемой предметной области. Примеры этой разновидности - системы KEE, ART и др.

Системы EXSYS и GURU относятся к системам дедуктивного продукционного типа, причем система GURU (в ее современной версии, ориентированной на рабочие станции) по сути является инструментальной средой, поддерживающей различные режимы конструирования прикладных ЭС и обладающей достаточно развитыми средствами обработки фактора неопределенности. Эта система ориентирована на различные классы пользователей в зависимости от их подготовки в области искусственного интеллекта и программирования, имеет развитый интерфейс с современными СУБД и электронными таблицами, средства сбора статистики и т. д. Система может работать на различных вычислительных платформах под управлением различных операционных систем, а также имеет поддержку сетевой конфигурации.

Экспертная система Exsys представляет собой интеллектуальную систему, которая может быть использована для разработки базы знаний в любой предметной области. При этом знания представляются в виде продукционных правил. В систему включены средства отладки и тестирования программы, редактирования для модификации знаний и данных.

В списке наиболее распространенных в настоящее время за рубежом экспертных систем и их оболочек можно выделить следующие наименования: INSIGT, LOGIAN, NEXPERT, RULE MASTER, KDS, PICON, KNOWLEDGE CRAFT, KESII, S1, TIMM и др.

В качестве критериев, по которым можно судить о возможности создания экспертной системы следует отметить следующие:

1. Необходимость символьных рассуждений, очевидно, нет смысла разрабатывать экспертную систему для численных расчетов, например, для преобразований Фурье, интегрирования, решения систем алгебраических уравнений и др.
2. Наличие экспертов, компетентных в избранном круге вопросов, которые согласны сотрудничать при создании ЭС.
3. Поставленная проблема должна быть достаточно важной и актуальной. Это могут быть проблемы, требующие высокого уровня экспертизы, либо простые, но трудоемкие многократно повторяющиеся проверки. Нет смысла тратить время на решение проблем, которые возникают редко и могут быть разрешены человеком с обычной квалификацией.
4. Необходимо четко ограничивать круг решаемых задач, т.е. предметная область выбирается достаточно "узкой", чтобы избежать "комбинаторного взрыва" объема информации необходимой для компетентного решения поставленной задачи.
5. Необходима согласованность мнений экспертов о том, как следует решать поставленные задачи, какие факты необходимо использовать и каковы общие правила вынесения суждений. В противном случае невозможно расширить базу знаний за пределы опыта одного человека и осуществить сплав экспертных знаний из нескольких областей.
6. Должно быть достаточно исходных данных для проверки работоспособности экспертной системы в выбранной предметной
7. Области, чтобы разработчики смогли убедиться в достижимости некоторого заданного уровня ее функционирования.
8. Должна обеспечиваться возможность постепенного наращивания системы. База знаний должна легко расширяться и корректироваться, так как правила часто меняются с появлением новых фактов.

Ценность использования ЭС проявляется в следующих аспектах:

1. В сборе, оперативном уточнении, кодировании и распространении экспертных знаний.
2. В эффективном решении проблем, сложность которых превышает человеческие возможности и для которых требуются экспертные знания нескольких областей.
3. В сохранении наиболее уязвимой ценности коллектива - коллективной памяти.

Создание баз знаний открывает широкие возможности, которые обусловлены безошибочностью и тщательностью, присущими ЭВМ и синтезом знаний экспертов. Если база знаний объединяет информацию по нескольким дисциплинам, то такой "сплав" знаний приобретает дополнительную ценность.

Экспертная система позволяет решить проблему сохранения экспертных знаний, связанную с утратой наиболее квалифицированных экспертов в результате их продвижения по службе, смерти, перехода на другую работу или выхода на пенсию, а также позволит сделать знания легко доступными для тех, кто займет места ушедших экспертов.

**MatLab**

Зарождение системы MATLAB относится к концу 70-х годов, когда первая версия этой системы была использована в Университете Нью Мехико и Станфордском университете для преподавания курсов теории матриц, линейной алгебры и численного анализа. В это время активно разрабатывались пакеты прикладных программ по линейной алгебре LINPACK и EISPACK на языке FORTRAN, и авторы системы MATLAB искали способы использовать эти пакеты, не программируя на языке FORTRAN.

Сейчас возможности системы значительно превосходят возможности первоначальной версии матричной лаборатории Matrix Laboratory. Нынешний MATLAB — это высокоэффективный язык инженерных и научных вычислений. Он поддерживает математические вычисления, визуализацию научной графики и программирование с использованием легко осваиваемого операционного окружения, когда задачи и их решения могут быть представлены в нотации, близкой к математической.

Наиболее известные области применения системы MATLAB:

* математика и вычисления;
* разработка алгоритмов;
* вычислительный эксперимент, имитационное моделирование, макетирование;
* анализ данных, исследование и визуализация результатов;
* научная и инженерная графика;
* разработка приложений, включая графический интерфейс пользователя.

MATLAB — это интерактивная система, основным объектом которой является массив, для которого не требуется указывать размерность явно. Это позволяет решать многие вычислительные задачи, связанные с векторно-матричными формулировками, существенно сокращая время, которое понадобилось бы для программирования на скалярных языках типа C или FORTRAN.

Одно из назначений математики - служить языком общения между учеными и инженерами. Матрицы, дифференциальные уравнения, массивы данных, графики — это общие объекты и конструкции, используемые как в прикладной математике, так и в системе MATLAB. Именно эта фундаментальная основа обеспечивает системе MATLAB непревзойденную мощь и доступность. Стоит прислушаться к следующему афористичному мнению: "Причина, по которой MATLAB столь полезен для обработки сигналов, состоит в том, что он не проектировался специально для этой цели, а создавался для математиков".

Система MATLAB — это одновременно и операционная среда, и язык программирования. Одна из наиболее сильных сторон системы состоит в том, что на языке MATLAB могут быть написаны программы для многократного использования. Пользователь может сам написать специализированные функции и программы, которые оформляются в виде М-файлов. По мере увеличения количества созданных программ возникают проблемы их классификации и тогда можно попытаться собрать родственные функции в специальные папки. Это приводит к концепции пакетов прикладных программ (ППП), которые представляют собой коллекции М-файлов для решения определенной задачи или проблемы.

В действительности ППП — это нечто большее, чем просто набор полезных функций. Часто это результат работы многих исследователей по всему миру, которые объединяются в зависимости от области применения - теория управления, обработка сигналов, идентификация и т. п. Именно поэтому пакеты прикладных программ - MATLAB Application Toolboxes, входящие в состав семейства продуктов MATLAB, позволяют находиться на уровне самых современных мировых достижений.

**PROLOG**

Пролог (Prolog) — язык логического программирования, основанный на логике дизъюнктов Хорна, представляющей собой подмножество логики предикатов первого порядка.

Разработка языка Prolog началась в 1970 г. Аланом Кулмероэ и Филиппом Русселом. Будучи декларативным языком программирования, Пролог воспринимает в качестве программы некоторое описание задачи, и сам производит поиск решения, пользуясь механизмом бэктрекинга и унификацией. Целью разработки языка Prolog было предоставить возможность задания спецификаций решения и позволить компьютеру вывести из них последовательность выполнения для этого решения, а не задание алгоритма решения задачи, как в большинстве языков.

Интерес к Прологу поднимался и затихал несколько раз, энтузиазм сменялся жёстким неприятием. Наиболее высоко был поднят интерес к языку Пролог как к языку будущего во время разработок японской национальной программы компьютеры пятого поколения в 1980-х годах, когда разработчики надеялись, что с помощью Пролога можно будет сформулировать новые принципы, которые приведут к созданию компьютеров более высокого уровня интеллекта. Неправильная оценка этой перспективы явилась одной из причин неудачи проекта.

В настоящее время Пролог, несмотря на неоднократные пессимистические прогнозы, продолжает развиваться в разных странах и вбирает в себя новые технологии и концепции, а также парадигмы императивного программирования. В частности, одно из направлений развития языка (в том числе и в России) реализует концепцию интеллектуальных агентов.

Пролог реализован практически для всех известных операционных систем и платформ (в том числе для Java и .NET). В число операционных систем входят OS для мэйнфреймов, всё семейство Unix, Windows, OS для мобильных платформ.

Многие современные реализации языка имеют внутреннее расширение за счет ООП-архитектуры. Кроме проприетарных решений также существуют реализации Пролог на условиях open source. Важно, что для языка существует стандарт ISO принятый как ISO/IEC JTC1/SC22/WG17.

Базовым принципом языка является равнозначность представления программы и данных (декларативность), отчего утверждения языка одновременно являются и записями, подобными записям в базе данных, и правилами, несущими в себе способы их обработки. Сочетание этих качеств приводит к тому, что по мере работы системы Пролога знания (и данные и правила) накапливаются. Поэтому Пролог-системы считают естественной средой для накопления базы знаний.

Пролог критикуется в первую очередь за свою недостаточную гибкость, отчего решения на обычных языках программирования (типа C++, Java) в сочетании с базами данных оказываются более технологичными, чем аналогичные решения на Прологе. Негибкость заключается в трудности изучения языка, более высоких требований к квалификации программиста на Прологе, трудности отладки программы, неразвитости технологии программирования, плохой контролируемости промежуточных результатов.[4]

## 2.2 Языки программирования ИИ

С огромным увеличением степени внедрения искусственного интеллекта (ИИ) и машинного обучения растет число программных инструментов, доступных разработчикам для использования в этой сфере.

**LISP**

Первый компьютерный язык, применяемый для создания искусственного интеллекта — ЛИСП. Этот язык является довольно-таки гибким и расширяемым. Такие особенности, как быстрое прототипирование и макросы очень полезны в создании ИИ. LISP — это язык, который превращает сложные задачи в простые. Мощная система объектно-ориентированности делает LISP одним из самых популярных языков программирования для искусственного интеллекта. [6]

**Java**

Java, чрезвычайно популярный язык программирования, также может рассматриваться как хороший выбор для программирования ИИ, поскольку он обеспечивает алгоритмы поиска и нейронные сети. Это простой для понимания язык, который предлагает графическое представление, отладку и масштабируемость. Его портативность делает его предпочтительной реализацией для различных приложений на основе наличия различных встроенных типов. [5]

**Prolog**

Это интерактивный символический язык программирования популярен для проектов, которые требуют логики. Имея мощную и гибкую основу, она широко применяется для non-численного программирования, доказательства теорем, обработки естественного языка, создания экспертных систем и искусственного интеллекта в целом.

Пролог — это декларативный язык с формальной логикой. Разработчики искусственного интеллекта ценят его за высокий уровень абстракции, встроенный механизм поиска, недетерминизм и т.д. [5]

**Python**

Python – широко применяемый язык программирования и может быть использован для реализации ИИ из-за простой и бесшовной структуры, которую он предлагает. Синтаксис Python позволяет легко реализовать различные алгоритмы ИИ, что также позволяет сократить время разработки по сравнению с другими доступными языками программирования. Применение Python позволяет пользователям создавать нейронные сети с набором полезных библиотек, которые могут использоваться для разработки ИИ. Другие функции включают возможность тестирования алгоритмов без необходимости их реализации. Он также поддерживает объектно-ориентированные, функциональные и процедурно-ориентированные стили программирования. [5]

**C++**

C++ - был разработан с учетом производительности, эффективности и гибкости, что делает его идеальным выбором для многих проектов программирования ИИ, которым необходима скорость. По сравнению с другими языками программирования, C++ имеет более быстрое выполнение и более низкую задержку, что делает его полезным для поиска решений сложных проблем ИИ. Он также позволяет широко использовать алгоритмы и является эффективным средством написания статистических методов ИИ, таких как нейронные сети. [6]

**MATLAB**

MATLAB (Matrix Laboratory) является патентованным языком программирования, разработанным MathWorks. Он широко используется многими разработчиками программного обеспечения и разработчиками для анализа краевых систем и проектов ИИ. Это простой в использовании язык со встроенной графикой, которая позволяет разработчикам визуализировать данные и получать от них значимую информацию. MATLAB – хороший выбор для машинного обучения и проектов ИИ при задачах визуализации и выполнения матриц. [6]

**Haskell**

Haskell – стандартизированный, универсальный язык программирования, разработанный с нестрогой семантикой и сильной статической типизацией. Первоначально разработанный в 1990 году, Haskell в основном используется в академических кругах, хотя есть и некоторые примеры его использования в промышленности и коммерции для проектов в AT&T, Facebook, Google и других. Haskell основан на семантике языка программирования Miranda и позволяет эффективным библиотекам реализовывать алгоритмы ИИ.[6]

## 2.3 Фреймворки и библиотеки для разработки ИИ

**TensorFlow**

TensorFlow — это комплексная платформа для машинного обучения с открытым исходным кодом. Она была разработана командой Google Brain как продолжение закрытой системы машинного обучения DistBelief, однако в ноябре 2015 года компания передумала и открыла фреймворк для свободного доступа.

Как и большинство фреймворков глубокого обучения, TensorFlow имеет API на Python поверх механизма C и C ++, что ускоряет его работу.

TensorFlow имеет гибкую экосистему инструментов, библиотек и ресурсов сообщества. Это позволяет исследователям использовать самые современные МО-технологии, а разработчикам — создавать и развёртывать приложения на базе машинного обучения.

Платформа предоставляет интуитивно понятные высокоуровневые API-интерфейсы, например Keras, с быстрым выполнением, что обеспечивает немедленную итерацию модели и простую отладку. За счёт мультиплатформенности решение позволяет обучать и разворачивать модели в облаке и локально, независимо от используемого пользователем языка.

Стоит отметить, что фреймворк постоянно развивается за счёт открытого исходного кода и огромного сообщества энтузиастов. Также за счёт его популярности есть множество уже решённых задач, что существенно упрощает жизнь новоиспечённым разработчикам.

Плюсы:

* Отличный фреймворк для создания нейронных сетей, которые будут работать в продакшене.
* Берёт на себя оптимизацию ресурсов для вычислений.
* Огромное комьюнити.
* За счёт популярности выше вероятность, что проблему, подобную вашей, уже решили.

Минусы:

* Сложен в использовании и освоении.
* Недружелюбный.
* Необходимо постоянно контролировать используемую видеопамять.
* Имеет свои стандарты.
* Плохая документация.

Проекты, которые используют фреймворк TensorFlow:

* DeepSpeech — система распознавания речи.
* Mask R-CNN — модель, которая генерирует ограничительные рамки и маски сегментации для каждого объекта на изображении.
* BERT — предобученная нейронная сеть, используемая для решения задач обработки естественного языка.

**PyTorch**

PyTorch — это среда машинного обучения на языке Python с открытым исходным кодом, обеспечивающая тензорные вычисления с GPU-ускорением. Она была разработана компанией Facebook и представлена в октябре 2016 года, а открыта для сторонних разработчиков — в январе 2017 года. Фреймворк подходит для быстрого прототипирования в исследованиях, а также для любителей и небольших проектов.

Фреймворк предлагает динамические графы вычислений, которые позволяют обрабатывать ввод и вывод переменной длины, что полезно, например, при работе с рекуррентными нейронными сетями. Если коротко, то за счёт этого инженеры и исследователи могут менять поведение сети «налету».

За счёт глубокой интеграции фреймворка с кодом C++ разработчики могут программировать на C и C++ с помощью API-расширения на основе FFI для Python.

В отличие от TensorFlow, PyTorch менее гибок в поддержке различных платформ. Также в нём нет родных инструментов для визуализации данных, но есть сторонний аналог, называемый tensorboardX.

Однако, снова же, в отличие от TensorFlow, если при работе с PyTorch вылетает ошибка, то это конкретная недоработка в коде и система выделит вам именно ту строчку, которая её спровоцировала.

Также при развёртке сетей на GPU PyTorch самостоятельно займёт только необходимую видеопамять.

Плюсы:

* Имеет множество модульных элементов, которые легко комбинировать.
* Легко писать собственные типы слоев и работать на GPU.
* Имеет широкий выбор предварительно обученных моделей.

Минусы:

* Вам придётся самостоятельно писать тренировочный код.
* Плохая документация, то и дело будут попадаться функции и методы, документация которых существует исключительно на форумах сообщества и получена эмпирическим путём.

Проекты, которые используют фреймворк PyTorch:

* PyText — библиотека для обработки устной и письменной речи.
* vid2vid — генеративная нейросеть для подмены деталей и свойств видеопотока.
* pix2pix — алгоритм, который превращает пользовательские наброски в фотографии.

**Keras**

Keras — открытая среда глубокого обучения, написанная на Python. Она была разработана инженером из Google Франсуа Шолле и представлена в марте 2015 года.

Фреймворк нацелен на оперативную работу с нейросетями и является компактным, модульным и расширяемым. Подходит для небольших проектов, так как создать что-то масштабное на нём сложно, и он явно будет проигрывать в производительности нейросетей тому же TensorFlow.

Keras работает поверх TensorFlow, CNTK и Theano и предоставляет интуитивно понятный API, который, по мнению наших инженеров, пока что является лучшим в своём роде.

Фреймворк содержит многочисленные реализации широко применяемых строительных блоков нейронных сетей, таких как слои, целевые и передаточные функции, оптимизаторы, а также множество инструментов для упрощения работы с изображениями и текстом.

Deeplearning4j использует Keras в качестве своего Python API и позволяет импортировать модели из Keras, а также через Keras из Theano и TensorFlow.

Плюсы:

* Удобен в использовании.
* Лёгок в освоении.
* Быстроразвивающийся фреймворк.
* Хорошая документация.
* Встроен в TF.

Минусы:

* Не подходит для больших проектов.

Проекты, которые используют фреймворк Keras:

* Mask R-CNN — модель, которая генерирует ограничительные рамки и маски сегментации для каждого объекта на изображении
* face\_classification — алгоритм для распознавания лиц в режиме реального времени и классификации эмоций и пола.
* YOLOv3 — нейронная сеть для обнаружения объектов в режиме реального времени.

**Darknet**

Darknet — это фреймворк с открытым исходным кодом, написанный на языке C с использованием программно-аппаратной архитектуры параллельных вычислений CUDA. Он быстрый, лёгкий и удобный в использовании. Также Darknet поддерживает вычисления на базе CPU и GPU.

Обученные веса Darknet хранит в формате, который может быть распознан с помощью разных методов на различных платформах. Однако это может стать проблемой, если вы решите натренировать модель на одном сверхмощном оборудовании, а затем использовать её на другом.

Так как фреймворк написан на C и не имеет другого API, то в случае, когда требования платформы или собственные предпочтения заставят обратиться к другому языку программирования, вам придётся дополнительно заморочиться над его интеграцией. К тому же он распространяется только в формате исходного кода, и процесс компиляции на некоторых платформах может быть несколько проблематичным.

Фреймворк не рекомендуется использовать для сложных проектов, разве что вам необходимо создать сверхбыстрый детектор объектов.

Плюсы:

* Простой.
* Быстрый.
* Удобный.

Минусы:

* Кроме задач с обнаружением больше нигде не используется.
* Не рекомендуется для больших проектов.
* Плохая документация.

Проекты, которые используют фреймворк Darknet:

* YOLOv3 — нейронная сеть для обнаружения объектов в режиме реального времени.
* Tiny-YOLO 3 — компактная нейронная сеть для обнаружения объектов. [7]

**Accord.Net**

Accord.Net — это фреймворк, основанный на C#, занимающийся развитием нейросетей, используемых для обработки аудио и изображений.

Компании могут использовать данный фреймворк в коммерческих целях, например, выпуская приложения с «компьютерным зрением», приложения для обработки сигналов, а также приложения для статистики.

Особенности:

* Проверенный годами, хорошо протестированный исходный код
* Большой набор датасетов для быстрого запуска вашего продукта

Плюсы:

* Постоянно обновляется и поддерживается активной командой разработчиков
* Стабильная система, которая эффективно обрабатывает огромное количество вычислений и визуализирует их
* Крайне удобная работа с алгоритмами и обработкой сигналов
* Легко справляется с числовой оптимизацией и искусственными нейросетями

Минусы:

* Малоизвестный, по сравнению с другими фреймворками
* Производительность гораздо ниже, в сравнении с другими фреймворками [8]

## 2.4 Дополнительные инструменты, участвующие в разработке ИИ

**IBM Watson**

IBM является крупным игроком в области ИИ, а его платформа Watson оснащена множеством инструментов, предназначенных как для разработчиков, так и для бизнес-пользователей. Благодаря доступным ресурсам в виде набора открытых API пользователи Watson будут иметь доступ к большому количеству шаблонов кода, стартовых наборов и могут создавать когнитивные поисковые системы и виртуальные агенты. Watson также имеет платформу для создания чатботов, ориентированную на новичков, что требует небольших навыков машинного обучения.

**Ai-one**

Ai-one позволяет разработчикам создавать интеллектуальных помощников в большинстве программных приложений. Ai-one Analyst Toolbox предоставляет документированную библиотеку, агенты построения и API для разработчиков. Ai-one может, по существу, превращать данные в обобщенные наборы правил, реализуя множество глубоких структур ИИ и машинного обучения.

**PredictionIO**

Apache PredictionIO – это сервер машинного обучения, созданный поверх стека с открытым исходным кодом для разработчиков и аналитиков данных что бы создавать интеллектуальные движки для любой задачи машинного обучения. Он состоит из трех основных компонентов. Во-первых, это платформа PredictionIO, это стек с открытым исходным кодом для построения, оценки и развертывания движков с алгоритмами машинного обучения. Во-вторых, Event Server – слой аналитики для обучения и объединения событий с нескольких платформ. В-третьих, Template Gallery – место загрузки шаблонов движков для различных типов приложений машинного обучения.

**Azure Machine Learning Studio**

Позволяет разработчикам, не имеющим глубокого опыта в сфере машинного обучения, перетаскивать (DnD) datasets и выгружать предиктивную аналитику.

Microsoft также предлагает Cortana Intelligence – инструмент, позволяющий в полной мере управлять большими данными и аналитикой и соответственно трансформировать данные в значимую информацию и последующие действия. [9]

## Заключение

На сегодняшний день тема «Искусственный интеллект» очень популярна. Она является частым предметом разговоров в сфере IT. С развитием интеллектуальных систем развивается и софт для его создания. Благодаря множествам проектов с открытым исходным кодом очень часто появляются разные фреймворки и библиотеки, как и от крупных компаний так и от разработчиков-энтузиастов, которые решают область конкретных задач. Так как тема весьма глубока в изучение, развитие искусственного интеллекта еще находится в начальной стадии. ИИ имеет очень большие перспективы так как уже на сегодняшний день решает весьма сложные задачи. С развитием ИИ будут и развиваться языки программирования для его написания. Будут разрабатываться большое количество библиотек и фреймворков, а также различные вспомогательные инструменты. Во многих IT-компаниях формируются специальные команды и лаборатория по изучению ИИ и разработки разных систем по решению конкретных задач. Поэтому ИИ имеет большое будущее также, как и разработчики в этой сфере и языки, которые используются для разработки этих систем.

## Список источников

1. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский — СПб: Питер, 2000. — 384 с.
2. Лисп [Электронный ресурс] / Режим доступа: https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%81%D0%BF (дата обращения: 10.11.2020).
3. Пролог (язык программирования) [Электронный ресурс] /Режим доступа https://ru.wikipedia.org/wiki/Пролог\_(язык\_программирования) (дата обращения: 10.11.2020).
4. Искусственный интеллект [Электронный ресурс] / Режим доступа: https://works.doklad.ru/view/6Bu4Twb21L8/all.html (дата обращения: 10.11.2020).
5. На каком языке программирования разрабатывать искусственный интеллект? [Электронный ресурс] / под. ред. Д. Н. Ушакова // 1940. – Режим доступа: https://neuronus.com/stat/1298-na-kakom-yazyke-programmirovaniya-razrabatyvat-iskusstvennyj-intellekt.html (дата обращения: 10.11.2020).
6. Языки программирования для искусственного интеллекта [Электронный ресурс] / Режим доступа: http://digitrode.ru/articles/1643-yazyki-programmirovaniya-dlya-iskusstvennogo-intellekta.html#sel= (дата обращения: 10.11.2020).
7. Топ-10 фреймворков для искусственного интеллекта [Электронный ресурс] / Режим доступа: https://vc.ru/ml/80391-top-10-freymvorkov-dlya-iskusstvennogo-intellekta-chast-pervaya (дата обращения: 10.11.2020).
8. Топ — 9 фреймворков в мире искусственного интеллекта. [Электронный ресурс] / Режим доступа: https://medium.com/nuances-of-programmingтоп-9-фреймворков-в-мире-искусственного-интеллекта-часть-2-3bd87aa6ea60 (дата обращения: 10.11.2020).
9. Инструменты для разработки искусственного интеллекта и машинного обучения [Электронный ресурс] / Режим доступа: http://digitrode.ru/articles/1644-instrumenty-dlya-razrabotki-iskusstvennogo-intellekta-i-mashinnogo-obucheniya.html (дата обращения: 10.11.2020).