**Доклад**

**(пришлось сократить свой доклад, потому что много желающих выступить, а времени мало)**

На начальном этапе развития ИИ языков и систем, ориентированных на создание прикладных систем, основанных на знаниях, не существовало. С одной стороны, в то время еще не было самого подхода, 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.

**1 слайд**

**Lisp**

Автором Лиспа является Джон Маккарти, на период создания языка, работавший в Массачусетском технологическом институте (MIT). 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. IPL был языком обработки списков и предназначался для реализации проекта «Логик-теоретик» — системы искусственного интеллекта, предназначенной для автоматического вывода теорем математической логики. 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. В результате Маккарти пришёл к мысли о необходимости создания нового языка программирования.

*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).*

*К 1962 году была готова следующая версия оригинальной лисп-системы «Lisp 1.5», в которой были устранены обнаруженные за время эксплуатации недостатки первой версии.*

**2 слайд**

**Prolog**

Начало истории языка относится к 1970-м годам.

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.

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.

**3 слайд**

**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, 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*

**4 слайд**

**Инструментальные средства разработки экспертных систем**

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.*

**5 слайд (читать слайд)**

**Языки программирования (LISP, JAVA, Prolog, C++, Python, Matlab, Haskell)**

***LISP***

*Первый компьютерный язык, применяемый для создания искусственного интеллекта. 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.*

***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.*

***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.*

***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.*

***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.*

***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.*

***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 слайд**

**Фреймворки**

**TensorFlow**

TensorFlow — это комплексная платформа для машинного обучения с открытым исходным кодом.

Плюсы:

* 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.

*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.*

*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.*

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.*

**7 слайд**

**PyTorch**

PyTorch — это среда машинного обучения на языке Python с открытым исходным кодом, обеспечивающая тензорные вычисления с GPU-ускорением. Фреймворк подходит для быстрого прототипирования в исследованиях, а также для любителей и небольших проектов.

*Она была разработана компанией Facebook и представлена в октябре 2016 года, а открыта для сторонних разработчиков — в январе 2017 года.*

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*.

**8 слайд**

**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.

*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.*

**9 слайд**

**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 помучиться 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.*

**10 слайд**

**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.

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

**11 слайд**

**Заключение**

На сегодняшний день тема «Искусственный интеллект» очень популярна. 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. По этому ИИ имеет большое будущее также, как и разработчики в этой сфере и языки, которые используются для разработки этих систем.