Молдавский государственный университет

Факультет математики и информатики

Департамент информатики

**Кадомцев Алексей**

**Логические методы в Искусственном Интеллекте**

**0613.5 InformaticaAplicată**

Реферат

Директор департамента: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Унгуряну Валериу, др. конф.

Руководитель: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Г .Капацина, ст. препод.

Автор: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Кадомцев Алексей

Кишинев – 2022

**Оглавление**

**Вступление 2**

**1.Логические методы в ИИ 5**

**1.1** Представление знаний в системах искусственного интеллекта 5

**1.2** Методы искусственного интеллекта 8

**1.3** Наиболее распространенные алгоритмы машинного обучения.................................20

**1.4** Язык Python в системах искусственного интеллекта 25

**2.Применение логических методов 23**

**2.1** Часть 1 . Заполнение данными файла , для обучения бота…………………………..27

**2.2** Часть 2 . Установка библиотеки PyTorch и создание модели……………………….28

### 2.3 Часть 3 . Предварительная обработка данных , использование NLTK……………..30

### 2.4 Часть 4 . Создание файла “тренировки”……………………………………………….32

### 2.5 Часть 5 . Создание файла “chat”………………………………………………………..35

### 2.6 Часть 6 . Cоздание графического интерфейса с помощью tkinter ………………….36

**Выводы 39**

**Библиография 40**

**Приложения……………………………………………………………………………………..41**

**Вступление**

**Цель работы**-разобрать методы и правила ИИ , определить роль ИИ в современном мире , применить методы ИИ в решении задачи.

Понятие искусственного интеллекта, как и интеллекта, довольно расплывчато. Обобщая все, что было сказано за последние тридцать лет, получается, что человек хочет только создать свое собственное, как в той или иной форме, хочет, чтобы некоторые действия выполнялись более рационально, с меньшими затратами времени и энергии.

Начиная с конца 1940-х годов, ученые во все большем количестве университетских и промышленных исследовательских лабораторий стремились к достижению амбициозной цели: Построить компьютеры, которые будут работать таким образом, чтобы они были неотличимы от человеческого разума в результатах своей работы. В последнее время растет интерес к искусственному интеллекту, вызванный растущими требованиями к информационным системам.

Искусственный интеллект является горячей точкой исследования. Именно здесь сосредоточены наибольшие усилия кибернетистов, лингвистов, психологов, философов, математиков и инженеров. Именно здесь решаются многие фундаментальные вопросы о пути развития научной мысли, о влиянии достижений компьютерной науки и робототехники, о влиянии на будущие поколения.

Оказалось, что для того, чтобы создать машины, имитирующие человеческий мозг, необходимо понять, как работают миллиарды его взаимосвязанных нейронов. И тогда многие исследователи пришли к выводу, что, пожалуй, самая трудная проблема, стоящая перед современной наукой, заключается в том, чтобы понять, как работает человеческий разум, а не просто подражать его работе. Что непосредственно повлияло на фундаментальные теоретические проблемы психологической науки.

На самом деле, ученым трудно даже прийти к единой точке зрения относительно самого объекта своих исследований — интеллекта. Одни считают, что интеллект — это способность решать сложные проблемы; другие рассматривают его как способность учиться, обобщать и аналогизировать; третьи рассматривают его как способность взаимодействовать с внешним миром через общение, восприятие и осознание того, что воспринимается.

Тем не менее, многие исследователи ИИ склонны принять тест машинного интеллекта, предложенный в начале 1950-х годов известным английским математиком и ученым в области вычислительной техники Аланом Тьюрингом. «Компьютер можно считать умным, — утверждал Тьюринг, — если он может заставить нас поверить, что мы имеем дело не с машиной, а с человеком».

Начало современного этапа развития систем искусственного интеллекта (ИИ) может быть датировано серединой 1950-х годов. Он был разработан А. Ньюэллом для доказательства теорем в пропозиционном исчислении и назывался Логическо-Теретическим. Некоторые авторы называют эту систему экспертной и прослеживают определение ее назначения до анализа ее возможностей, проведенного Клодом Шенноном и Марвином Минским.Эта работа положила начало исследованиям в области ИИ, которые касаются разработки программ, решающих проблемы путем применения различных эвристических методов и правил.

**Представление знаний**является важнейшей областью исследований в области искусственного интеллекта, основой всех других дисциплин. Знания принимают форму описания объектов, взаимосвязей и процессов. Наличие адекватных знаний и способность их эффективно использовать означает «навык».

**Создание общей теории**или метода представления знаний является стратегической задачей. Такая теория открыла бы возможность накопления знаний, которые необходимы каждый день для решения все новых и новых задач. Однако для достижения этой цели необходимо найти способ выразить общие закономерности предметных областей (SD), что является ядром проблемы представления знаний.

**Решение задач**— это нахождение пути от начальной точки к цели. Люди делают это очень эффективно посредством дедуктивного рассуждения (аргументации), процедурного анализа, аналогии и индукции. Люди также могут учиться на собственном опыте. Компьютеры, как правило, решают проблемы только с помощью дедуктивных рассуждений и процедурного анализа.

**Тип задачи**определяет метод, наиболее подходящий для ее решения. Задачи, сводимые к процедурному анализу, как правило, лучше всего решать на компьютере. Проблемы учета и анализа являются примерами процедурных проблем, которые могут быть решены компьютерами быстрее и надежнее, чем людьми. Напротив, задачи, требующие использования аналогии или индукции, более эффективно решаются человеком. Задачи, требующие дедуктивного и индуктивного рассуждения, скорее всего, будут решаться с помощью экспертных систем (систем, основанных на знаниях).

Одним из важнейших признаков интеллекта является способность учиться. Например, в 1961 году один из ведущих специалистов Англии по искусственному интеллекту, профессор Мичи, описал механизм, состоящий из 300 спичечных коробок, которые могли научиться играть «тик-так-нолик». Однако явно недостаточно сделать вывод об интеллекте, не говоря уже об искусственном интеллекте, на основе одного лишь атрибута.

Искусственный интеллект сегодня используется повсюду в мире. Компьютеры и другие вычислительные машины уже давно являются верными помощниками человека, всегда готовыми прийти на помощь в решении различного рода задач. Во многих случаях ЭВМ обеспечивает более высокую скорость решения задач и более высокую вычислительную точность, чем если бы человек решал эти задачи самостоятельно.

Сегодня искусственный интеллект является неотъемлемой частью цивилизации и используется во всех сферах человеческой деятельности. Примеры варьируются от использования микрокалькуляторов до использования роботов в качестве врачей, нянь, уборщиков (все чаще используется в Японии).

Что не может сделать компьютер в современном (цифровом) смысле, и, вероятно, никогда не сможет? Ответ очевиден: выполнить решение полностью аналитически. Задача состоит в том, чтобы заменить аналитическое решение числовым алгоритмом, который рекурсивно выполняет операции, шаг за шагом приближающиеся к решению. С увеличением количества таких операций увеличивается и время выполнения, и, возможно, потребление других ресурсов (например, ограниченной памяти машины) имеет тенденцию к бесконечности. С помощью алгоритмов их решения задачи, резко повышающие потребление ресурсов, не могут быть решены на цифровых компьютерах, так как ресурсы всегда ограничены.

В настоящее время наличие сверхвысокопроизводительных микропроцессоров и дешевизна электронных компонентов позволяют значительно продвинуться в алгоритмическом моделировании искусственного интеллекта. Такой подход дает некоторые результаты на цифровых компьютерах общего назначения и заключается в моделировании процессов жизни и мышления с использованием численных алгоритмов, реализующих искусственный интеллект.

Главное ограничение ИИ заключается в том, что обучение возможно только на основе данных, другими способами - невозможно. Это означает, что любые неточности в данных отразятся на результатах. А новые уровни прогнозирования или анализа необходимо добавлять отдельно.

В большинстве случаев реализации ИИ — от компьютерных шахматистов до беспилотных автомобилей — крайне важна возможность глубокого обучения и обработки естественного языка. Благодаря этим технологиям компьютеры можно «научить» выполнению определенных задач с помощью обработки большого объема данных и выявления в них закономерностей.

**Глава №1:Логические методы в ИИ**

**Представление знаний в системах искусственного интеллекта**

Теория представления знаний — фундаментальнейший раздел искусственного интеллекта. Ее назначение — найти такие способы описания и представления фактов, общих сведений, закономерностей, правил и предписаний об окружающем мире, которые позволят использовать все эти знания с помощью некоторых универсальных и формальных процедур анализа, рассуждения и синтеза, доступных для программной реализации на ЭВМ. Для реализации этих универсальных процессов разработаны специальные логико-алгоритмические языки, позволяющие объединить вычислительные, комбинаторные и логические шаги в обработке слож-ноорганизованной информации (например, язык "Лисп").

Традиционно выделяются две группы методов представлений знаний — декларативные и процедурные. В декларативных методах знания — это данные, так или иначе структурированные. В процедурных методах знания также представляются в ЭВМ структурами данных, но при этом с элементами структур ассоциируются некоторые специализированные процедуры. В группе декларативных методов представления знаний выделяются логические и сетевые (основанные на аппарате семантических сетей).

Основная идея логического подхода к представлению знаний состоит в том, чтобы рассматривать всю систему знаний, необходимую для решения каких-то задач как совокупность фактов (утверждений). Факты представляются как формулы в некоторой логике (первого или высшего порядков, модальной, многозначной, нечеткой или какой-либо другой). Система знаний отображается совокупностью таких формул. Будучи представленной в компьютере, она образует базу знаний. Формулы неделимы и при модификации базы знаний могут лишь добавляться и удаляться.

Логические методы представления знаний обеспечивают простую и ясную систему для записи фактов, обладающую четко определенной семантикой (по крайней мере, для методов, основанных на традиционной логике первого порядка). Каждый факт представляется в базе знаний только один раз, независимо от того, как он будет использоваться в дальнейшем.

Логические методы предоставляют также и развитый аппарат вывода новых фактов из тех, которые представлены в базе знаний явно. Основным инструментом манипуляции знаниями является операция логического вывода.

При этом в системах прямой дедукции новые знания получают, применяя выводы к фактам и правилам. Алгоритм завершает работу при получении некоторого знания, эквивалентного цели (или непосредственно влекущего ее). Систему прямой дедукции можно толковать как систему, основанную на теореме о прямой дедукции: формула G является логическим следствием формул тогда и только тогда, когда формула G является тождественно ложной.

В системах обратной дедукции выводы применяют к цели и к правилам, чтобы построить новые частичные цели. Алгоритм завершает работу, когда все частичные цели соответствуют фактам. Такую систему с логической точки зрения можно толковать как систему, в которой применяется теорема об обратной дедукции: формула G является логическим следствием формул тогда и только тогда, когда формула  G является тавтологией (тождественно истинна).

Инструмент логического вывода определяет интенсивное использование логических методов при создании так называемых экспертных систем и всевозможных решателей задач. Имеется и другое применение этого аппарата, важное для любых систем искусственного интеллекта. Это — возможность контроля логической целостности базы знаний, т.е. ее непротиворечивости и соответствия предустановленным правилам (ограничениям целостности).

Основная идея подхода к представлению знаний, основывающемуся на аппарате семантических сетей, состоит в том, чтобы рассматривать проблемную среду как совокупность объектов (сущностей) и связей (отношений) между ними. Объекты представляются при этом поименованными вершинами, а отношения — направленными поименованными ребрами. Система знаний отображается сетью (семантическая сеть) — ориентированным графом, составленным из поименованных вершин и ребер, или совокупностью таких сетей. Но было показано, что и этот метод представления знаний можно переписать и переинтерпретировать с помощью подходящего логического формализма.

Итак, мы видим, что роль математической логики во всех системах, связанных с приобретением, хранением и использованием знаний, исключительно велика. По существу, логика неизбежна в этих исследованиях. Она доставляет средство, хорошо подходящее для представления знаний и рассуждений. Она может рассматриваться как формализм для ссылок. Она может рассматриваться как метод подтверждения рассуждений и семантического анализа представленных знаний.

Поскольку конечная цель — представление знаний, основным критерием адекватности используемого логического языка является его выразительность. Системы искусственного интеллекта чаще всего ограничиваются применением языков логики высказываний и логики предикатов. Логика предикатов достаточно выразительна для решения многих проблем представления знаний в искусственном интеллекте и служит своего рода эталоном выразительности. (Тем не менее некоторые знания формализуются лишь в логических языках более высоких порядков.) Некоторые проблемы представления знаний и рассуждений решаемы лишь с помощью логических языков и ассоциированных с ними дедуктивных (аксиоматических) систем. В частности, благодаря точному определению принципов применения логических операторов и связок (отрицания, конъюнкции, дизъюнкции, равенства, кванторов общности и существования) логика позволяет выразить некоторые часто полезные парадигмы рассуждений. Например, логика предикатов с равенством дает возможность:

а) выразить, что нечто обладает определенным свойством, не указывая, что именно (роль квантора существования);

б) выразить, что каждый элемент некоего множества обладает определенным свойством, без указания того, что представляет собой каждый такой элемент (роль квантора общности);

в) выразить, что хотя бы одно из двух утверждений ложно, не говоря, какое именно (роль конъюнкции);

г) выразить, что хотя бы одно из двух утверждений истинно, не говоря, какое именно (роль дизъюнкции);

д) явно сказать, что нечто ложно (роль отрицания); е) утверждать или оставлять неустановленным тот факт, что два различных выражения означают один и тот же объект (роль равенства).

Формальная логика связана с формализацией и обоснованием корректных рассуждений, которые также называются общезначимыми: их правильность несомненна при всех интерпретациях. Дедуктивные системы логики специально приспособлены для формализации этого класса рассуждений. Но рассуждения, которые желательно моделировать в приложениях искусственного интеллекта, не всегда общезначимы. Часто они приблизительны и неопределенны по сути или от неполноты, или неопределенности предпосылок. Выведенные из неопределенных рассуждений заключения должны допускать возможность отказа от них, если предпосылки, приведшие к принятию предположения о возможности этих заключений, больше не подтверждаются или если новая информация блокировала эту дедукцию. Дедуктивные системы классической логики не позволяют прямо формализовать такие рассуждения. Для этих целей разработаны различные неклассические логические системы. Среди них — немонотонные логики Мак-Дермотта, логики умолчаний Рейтера, автоэпистемические логики Столнекера и Мура.

В заключение отметим, что математическая логика представляет собой важнейшее и мощнейшее средство анализа знаний и рассуждений. Она выполняет роль блюстителя логических принципов и правил в подавляющем большинстве систем искусственного интеллекта. Она может прямо использоваться для представления знаний и рассуждений. Она может пригодиться для ссылок и как эталон выразительности, модель компетенции, гарант элементарных логических принципов. Она определяет принципы и законы, незаменимые при решении многих проблем. Она позволяет анализировать смысл некоего представления знаний и обоснованность выводов. Она является преимущественно средством анализа знаний и рассуждений как таковых.

**Методы искусственного интеллекта**

**Что такое метод искусственного интеллекта?**

У термина «метод» есть множество определений. Так как сфера искусственного интеллекта, в основном, подразумевает знания в области математики, программирования и информационных технологий, то метод, в нашем случае, — путь познания или способ познания какой-либо предметной области, способ достижения цели. А метод искусственного интеллекта — это способ, а фактически, — алгоритм решения какой-либо задачи.

Существует большое количество направлений развития искусственного интеллекта. В рамках этих направлений есть различные методы, которые могут применяться по отдельности или в группах для решения задач, стоящих перед наукой, промышленностью, экономикой, медициной и другими областями.

Классификация методов искусственного интеллекта (ИИ)

Есть разные мнения о том, как классифицировать методы ИИ. Мы предлагаем следующую классификацию, которая состоит из пяти пунктов:

1.Искусственные нейронные сети

2.Нечеткая логика (нечеткие множества и мягкие вычисления)

3.Системы, основанные на знаниях (экспертные системы)

4.Эволюционное моделирование (генетические алгоритмы, многоагентные системы)

5.Machine Learning (Data Mining и анализ данныхи, поиск закономерностей в хранилищах данных)

**Искусственные нейронные сети**

Искусственная нейронная сеть — это преимущественно математический аппарат, хотя иногда в различных парадигмах нейросетей встречаются элементы логики.

Нейронная сеть — математическая модель, прототипом которой служит центральная нервная система человека или животного.

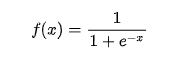
Данный метод ИИ применяется в задачах распознавания образов, прогнозирования, классификации, кластеризации и оптимизации .Принцип действия искусственной нейронной сети состоит в формировании связей между множеством различных обрабатывающих элементов, каждый из которых служит аналогом одного нейрона в головном мозге биологического существа. Нейроны могут быть воспроизведены физически или смоделированы с помощью цифрового компьютера. Каждый нейрон получает множество входных сигналов, а затем, с учетом внутренней системы весовых коэффициентов, порождает один выходной сигнал, который, как правило, служит входным для другого нейрона.

Нейроны тесно взаимосвязаны друг с другом и организованы в несколько различных уровней. Входной уровень получает входные данные, а выходной — порождает конечный результат. Обычно между этими двумя уровнями находятся один или несколько скрытых уровней. В такой структуре невозможно предсказать или точно узнать, как именно передаются данные.

Нейронные сети состоят из нейронов связанных между собой, поэтому нейрон является главной частью нейронной сети. Нейроны выполняют только 2 действия: перемножают входные данные на веса и суммируют их и прибавляют смещение, и второе действие это активация.

Теперь подробнее про входные данные, веса, смещения и функцию активации. Входные данные это данные которые получает нейрон из предыдущих нейронов или от пользователя. Веса присваиваются каждому входу нейрона, изначально им присвоены рандомные числа. При обучении нейронной сети изменяется значение нейронов и смещений. На веса перемножаются входные данные которые поданы на нейрон. Смещения присваиваются каждому нейрону, как и веса изначально смещения, это рандомные числа. Смещения облегчают и ускоряют обучение нейронной сети.

Функция активации которую мы будем использовать в нашей нейронной сети называется сигмоида. Формула сигмоиды имеет вид:



Эта функция заворачивает любое число от -∞ до +∞ в диапазон от 0 до 1.Если расписать математически, задачи, которые выполняет нейрон то, получится 2 формулы: Обозначения:

f(x) - функция активации

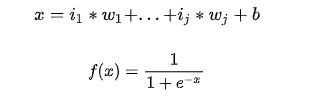
x - сумма произведения входных данных с весами, и смещения

i - массив входных данных

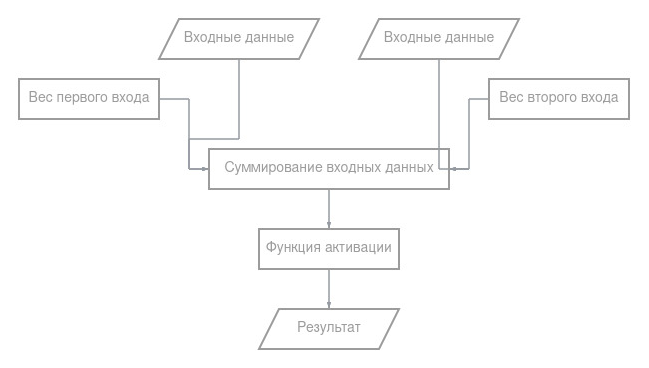
w - массив весов

b - смещение

j - количество входных данных и весов



Алгоритм работы нейрона будет выглядеть так:



**Нечёткая логика, нечёткие множества и мягкие вычисления**

Нечёткая логика, теория нечётких множеств, нечёткие рассуждения, мягкие вычисления — всё это близкие или тесно связанные между собой понятия, относящиеся к более высокому уровню работы центральной нервной системы, нежели искусственные нейронные сети. Методы нечеткой логики используются в экпертных системах, системах управления объектом.

Нечёткая логика в большей степени связана с качественной оценкой анализируемых процессов и явлений и принятием решений на основе этой качественной оценки. **Нечеткие нейронные сети**

Нечеткие нейронные сети (fuzzy-neural networks) осуществляют выводы на основе аппарата нечеткой логики, однако параметры функций принадлежности настраиваются с использованием алгоритмов обучения НС. Поэтому для подбора параметров таких сетей применим метод обратного распространения ошибки, изначально предложенный для обучения многослойного персептрона. Для этого модуль нечеткого управления представляется в форме многослойной сети. Нечеткая нейронная сеть, как правило, состоит из четырех слоев: слоя фазификации входных переменных, слоя агрегирования значений активации условия, слоя агрегирования нечетких правил и выходного слоя.

Наибольшее распространение в настоящее время получили архитектуры нечеткой НС вида ANFIS и TSK. Доказано, что такие сети являются универсальными аппроксиматорами.Быстрые алгоритмы обучения и интерпретируемость накопленных знаний — эти факторы сделали сегодня нечеткие нейронные сети одним из самых перспективных и эффективных инструментов мягких вычислений.

**Адаптивные нечеткие системы**

Классические нечеткие системы обладают тем недостатком, что для формулирования правил и функций принадлежности необходимо привлекать экспертов той или иной предметной области, что не всегда удается обеспечить. Адаптивные нечеткие системы (adaptive fuzzy systems) решают эту проблему. В таких системах подбор параметров нечеткой системы производится в процессе обучения на экспериментальных данных. Алгоритмы обучения адаптивных нечетких систем относительно трудоемки и сложны по сравнению с алгоритмами обучения нейронных сетей, и, как правило, состоят из двух стадий:

1.Генерация лингвистических правил;

2.Корректировка функций принадлежности.

Первая задача относится к задаче переборного типа, вторая — к оптимизации в непрерывных пространствах. При этом возникает определенное противоречие: для генерации нечетких правил необходимы функции принадлежности, а для проведения нечеткого вывода — правила. Кроме того, при автоматической генерации нечетких правил необходимо обеспечить их полноту и непротиворечивость.

Значительная часть методов обучения нечетких систем использует генетические алгоритмы. В англоязычной литературе этому соответствует специальный термин — Genetic Fuzzy Systems.

Значительный вклад в развитие теории и практики нечетких систем с эволюционной адаптацией внесла группа испанских исследователей во главе с Ф. Херрера (F. Herrera).

**Нечеткие запросы**

Нечеткие запросы к базам данных (fuzzy queries) — перспективное направление в современных системах обработки информации. Данный инструмент дает возможность формулировать запросы на естественном языке, например: «Вывести список недорогих предложений о съеме жилья близко к центру города», что невозможно при использовании стандартного механизма запросов. Для этой цели разработана нечеткая реляционная алгебра и специальные расширения языков SQL для нечетких запросов. Большая часть исследований в этой области принадлежит западноевропейским ученым Д. Дюбуа и Г. Праде.

**Нечеткие ассоциативные правила**

Нечеткие ассоциативные правила (fuzzy associative rules) — инструмент для извлечения из баз данных закономерностей, которые формулируются в виде лингвистических высказываний. Здесь введены специальные понятия нечеткой транзакции, поддержки и достоверности нечеткого ассоциативного правила.

**Нечеткая кластеризация**

Нечеткие методы кластеризации, в отличие от четких методов (например, нейронные сети Кохонена), позволяют одному и тому же объекту принадлежать одновременно нескольким кластерам, но с различной степенью. Нечеткая кластеризация во многих ситуациях более «естественна», чем четкая, например, для объектов, расположенных на границе кластеров. Наиболее распространены: алгоритм нечеткой самоорганизации c-means и его обобщение в виде алгоритма Густафсона-Кесселя.

Список можно продолжить и дальше: нечеткие деревья решений, нечеткие сети Петри, нечеткая ассоциативная память, нечеткие самоорганизующиеся карты и другие гибридные методы.

**Эволюционное или многоагентное моделирование**

Эволюционное моделирование целесообразно применять тогда, когда пространство поиска решения настолько большое и сложно устроенное, что традиционные и более простые методы просто неспособны выполнить глобальный поиск решения или способны, но на это потребуется неприемлемо много времени.

Основное преимущество**эволюционного моделирования** заключается в возможности решения задач, имеющих много локальных оптимумов за счет комбинирования элементов случайности и направленности аналогично тому, как это происходит в природе. Другим важным фактором эволюционного моделирования является моделирование процессов селекции, размножения и наследования. При этом получаемые по определенным правилам альтернативные решения могут порождать новые решения, которые будут наследовать лучшие характеристики предыдущих. Суть стратегии эволюционного моделирования состоит в реализации целенаправленного процесса размножения-гибели, при котором размножению соответствует появление новых объектов, а гибели - удаление объектов в соответствии с определенными критериями естественного и искусственного отбора.

Эволюционное моделирование это уже достаточно сложившаяся область, в которой можно выделить:

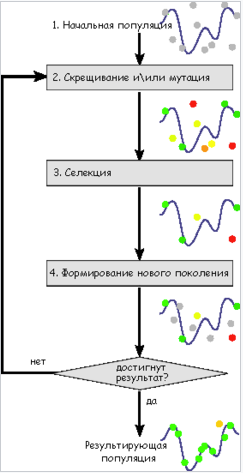
1.модели возникновения молекулярно-генетических информационных систем;

2.Моделирование общих закономерностей эволюции (Эволюционные алгоритмы). Это системы, которые используют только эволюционные принципы. Они успешно использовались для задач типа функциональной оптимизации и могут легко быть описаны на математическом языке. К ним относятся эволюционные алгоритмы, такие как Эволюционное программирование, Генетические алгоритмы, Эволюционные стратегии, Генетическое программирование;

3.Эволюционные модели. Это системы, которые являются биологически более реалистичными, чем эволюционные алгоритмы, но которые не оказались полезными в прикладном смысле. Они больше похожи на биологические системы и менее направлены на решение технических задач. Они обладают сложным и интересным поведением, и, видимо, вскоре получат практическое применение. К этим системам относят так называемую искусственную жизнь.

4.Прикладное эволюционное моделирование.

**Схема работы генетического алгоритма**



**Экспертные системы. Поддержка принятия решений**

Экспертная система — это искусственный аналог лица, принимающего решения, или, как минимум, эксперта-консультанта предметной области.Структура и логико-математический аппарат экспертной системы определяются, в первую очередь, её назначением и предметной областью. Сами решения, предлагаемые системой, могут вырабатываться с использованием различных механизмов вывода. Наиболее близкий аналог человеческому механизму вывода — это аппарат нечёткой логики и теории нечётких множеств.

Экспертная система поддержки принятия решений является составной частью СППР, которая позволяет:

1) извлекать данные из разнородных источников, включая неструктурированную информацию;

2) осуществлять многомерный анализ данных и обработку статистических материалов;

3) моделировать правила стратегии деловой активности;

4) представлять результаты анализа в графическом виде;

5) проводить анализ вариантов по критерию «что если?»;

6) использовать искусственный интеллект.

В отличие от традиционных программ, предназначенных для решения математически определенных задач по строгим алгоритмам, с помощью экспертных систем решаются задачи, относящиеся к классу неформализованных или слабо формализованных (не структурированных задач). Алгоритмические решения таких задач не существуют в силу неполноты, неопределенности, неточности и некорректности рассматриваемых ситуаций или же такие решения неприемлемы практически в силу сложности реализуемых алгоритмов. Поэтому экспертные системы используют логические выводы и эвристический поиск решения.

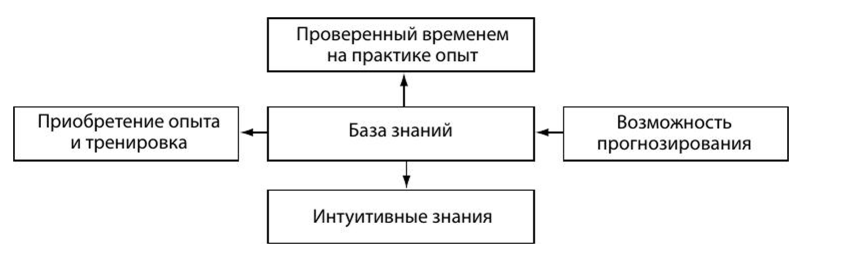
От других систем экспертные системы СППР отличаются тем, что первые опираются на математические методы и модели, а вторые в основном базируются на эвристических, эмпирических знаниях, оценках, методах, которые получены от экспертов, способных анализировать и объяснять пользователю свои действия и знания.

Идея построения экспертных систем сформировалась в ходе исследований в области искусственного интеллекта. Экспертные системы с точки зрения решаемых задач принято разделять на два больших класса.

Экспертные системы должны заменить высококвалифицированных специалистов, способных дать полезный совет, сообщить необходимые сведения человеку, находящемуся в затруднительном положении. Потребителем такой информации может оказаться неопытный специалист, перед которым возникла нестандартная, ранее ему не знакомая ситуация.

Экспертная система обычно имеет доступ или хранит сведения, полученные из самых различных источников (книг, журнальных публикаций, устных сообщений специалистов и т. п.) и может их использовать для консультации и при необходимости пояснить специалисту полученные ею выводы.

Одним из самых сложных процессов при создании экспертных систем является построение *базы знаний.* Эта сложность в основном связана с необходимостью структурирования знаний, которые определяются решаемой проблемой. Эксперт, знания которого вводятся в систему, может быть не знаком с особенностями программного продукта и технических характеристик ЭВМ, реализующих экспертную систему. Поэтому ему в качестве помощника необходим специалист (инженер по знаниям), знающий одновременно область возможного применения экспертной системы, структуру и детали программы, а также особенности ее реализации. Такой специалист должен подобрать рациональный вариант структурирования вводимых знаний, соответствующих возможностям системы.

**Структура экспертной системы**

**Machine Learning, Data Mining, Data Science**

Machine Learning (машинное обучение) — это целый класс методов искусственного интеллекта. Все они подразумевают решение задач не напрямую, а путем предварительного обучения как до, так и в процессе принятия решения.Data mining. Данный термин введён Григорием Пятецким-Шапиро в 1989 году. По сути — это собирательное название, которое применяется для обозначения целой группы методов обнаружения определенных закономерностей в общем объеме данных, которые могут получены в различных сферах человеческой деятельности. Например, методы Data Mining могут быть использованы для больших данных (Big Data), накопленных в розничных продажах, для подтверждения каких-либо гипотез и принятия управленческих решений.

**Что такое машинное обучение?**

Причина, по которой первые исследователи столкнулись с тем, что некоторые задачи на самом деле оказались намного сложнее, чем представлялось ранее, состоит в следующем: эти задачи просто не поддаются решению ранними методами, используемыми в ИИ. Жестко закодированные алгоритмы или фиксированные, основанные на правилах системы не очень хорошо работают с такими вещами, как распознавание изображений или понимание текста.Решением оказалось не просто подражание человеческому поведению, а имитация того, как люди учатся.

Подумайте о том, как вы освоили чтение. Вы ведь не садились изучать орфографию и грамматику, прежде чем прочесть свою первую книгу. Вы сначала читаете простые книги, которые со временем становятся все более сложными. Вы действительно изучили правила (и исключения) орфографии и грамматики – но в процессе чтения. Иными словами, вы обработали много данных и научились на них.

Именно в этом и состоит идея машинного обучения. Дать алгоритму (в отличие от мозга) огромное множество данных и позволить ему сделать заключения. Снабдите алгоритм большим количеством данных о финансовых транзакциях, укажите, какие из них являются мошенническими, и дайте ему понять, что именно говорит о мошенничестве, чтобы он мог прогнозировать такие злоупотребления. Или дайте информацию о своей клиентской базе и позвольте ему сделать вывод, как лучше ее сегментировать. Больше узнать о методах машинного обучения можно [здесь](https://blogs.oracle.com/bigdata/machine-learning-techniques).

По мере совершенствования этих алгоритмов они могли бы решить многие задачи. Но некоторые вещи, которые людям казались довольно легкими (например, распознавание речи или рукописного ввода), все еще были трудными для машин. Однако, если машинное обучение - это подражание тому, как люди учатся, почему бы не пройти весь путь и не попытаться имитировать человеческий мозг? Эта идея – в основе [нейронных сетей](https://blogs.oracle.com/bigdata/3-more-machine-learning-techniques-to-know).

Идея использования искусственных нейронов (а нейроны, связанные посредством синапсов, являются основными элементами нашего мозга) была известна довольно давно, и нейронные сети, смоделированные в программном обеспечении, стали использовать для решения определенных задач. Подход оказался многообещающим и позволил решить некоторые сложные задачи, с которыми не справлялись другие алгоритмы.

Но машинное обучение все еще ограничивалось тем, с чем легко справлялись младшие школьники: сколько собак на этой картине или не волки ли это на самом деле? Идите туда и принесите мне спелый банан. Что заставило героя этой книги так много плакать?

Оказалось, что проблема – вовсе не в самой концепции машинного обучения. И даже не в идее подражать работе человеческого мозга. Простые нейронные сети с сотнями или даже тысячами нейронов, связанные относительно простым способом, не могли дублировать то, на что способен человеческий мозг. Если подумать, то это вовсе не удивительно: в человеческом мозгу насчитывается около 86 миллиардов нейронов и имеют место очень сложные взаимосвязи.

**Data mining**

Технология Data mining – интеллектуальная обработка данных с использованием методов машинного обучения, математической статистики и теории баз данных.

Фундаментально data mining основывается на 3-х понятиях:

1.Математическая статистика – является основой большинства технологий, используемых для data mining, например, кластерный анализ, регрессионный анализ, дискриминирующий анализ и пр.;

2.Искусственный интеллект – воспроизведение нейронной сети мышления человека в цифровом виде;

3.Машинное обучение – совокупность статистики и искусственного интеллекта, способствующая пониманию компьютерами данных, которые они обрабатывают для выбора наиболее подходящего метода или методов анализа.

В data mining используются следующие основные классы задач:·

1.обнаружение отклонений – выявление данных, отличающихся по каким-либо параметрам из общей массы;

2.обучение ассоциациям – поиск взаимосвязей между событиями;

3.кластеризация – группирование наборов данных, без заранее известных шаблонов;

4.классификация – обобщение известного шаблона для применения к новым данным;

5.регрессия – поиск функции, отображающей набор данных с наименьшим отклонением;

6.подведение итогов – отображение в сжатом виде исходной информации, включая предоставление отчетов и визуализацию.

Знания, обнаруженные в процессе Data Mining, должны быть нетривиальными и ранее неизвестными. Нетривиальность предполагает, что такие знания не могут быть обнаружены путем простого визуального анализа. Они должны описывать связи между свойствами бизнес-объектов, предсказывать значения одних признаков на основе других и т.д. Найденные знания должны быть применимы и к новым объектам.

Практическая полезность знаний обусловлена возможностью их использования в процессе поддержки принятия управленческих решений и совершенствовании деятельности компании.

Знания должны быть представлены в виде, понятном для пользователей, которые не имеют специальной математической подготовки. Например, проще всего воспринимаются человеком логические конструкции «если, то». Более того, такие правила могут быть использованы в различных СУБД в качестве [SQL](https://wiki.loginom.ru/articles/sql.html)-запросов. В случае, когда извлеченные знания непрозрачны для пользователя, должны существовать методы постобработки, позволяющие привести их к интерпретируемому виду.

**Что такое Data Science?**

Data Science — это наука о методах анализа данных и извлечения из них ценной информации, знаний. Она тесно пересекается с такими областями как машинное обучение (Machine Learning) и науку о мышлении (Cognitive Science и, конечно же, технологиями для работы с большими данными (Big Data). За время массового распространения технологий человек нагенерировал огромное количество данных. Такое, которое он не способен обработать и визуализировать. Данные о наших звонках и перемещениях, поведении в интернете, предпочтениях в магазинах, антропогенных изменениях в ландшафте, климатических процессах и многих других вещах. Это все — большие данные, Big Data. И из них при правильной обработке можно извлечь большую пользу.

Во все времена ранее компьютеры получали новые возможности посредством программирования – человек создавал для машины понятные алгоритмы работы, которые приводили к ожидаемому результату. Этот подход устарел.

Для эффективной работы с большими данными нужен другой, им стало машинное обучение. В этом случае человек только дает компьютеру какие-то вводные, но результаты работы такого алгоритма не детерменированы человеком. Человек определяет способ обучения машины, но машина учится сама; сама приходит к тем или иным ответам и анализирует информацию. Это похоже на то, как учимся мы с вами. Машинное обучение — это не только искусственный интеллект. К этой сфере относятся генетические и эволюционные алгоритмы, и более простые задачи, связанные с кластерным анализом, например.

Наконец, Cognitive Science. Это междисциплинарная наука, изучающая, механизмы познания и мышления. Результаты таких исследований в первую очередь ложатся в основу разработки различных подходов к созданию искусственного интеллекта.

**Причем тут нейросети?**

Нейросети уже можно применять практически везде. В области права – для поиска прецедентов (что особенно популярно в американской судебной системе). В финтехе — для анализа сделок, программы лояльности, надежности клиента и так далее. В логистике — для прогнозирования потребности в тех или иных продуктах. В медицине нейросети, обрабатывая огромные массивы данных, могут находить неожиданные факторы, влияющие на здоровье пациента, и точно диагностировать даже самые сложные заболевания.

Нейросети могут проявлять себя и в творческих задачах. Но знаменитые фильтры на изображениях – это побочный продукт, который появился в результате того, что разработчики пытались разобраться, как же работает алгоритм обучения нейросети. Он стал широкоизвестным среди массовой аудитории. Незаслуженно меньше говорится о том, что нейросети успели хорошо себя зарекомендовать в вопросах перевода, распознавания или анализа данных. Между тем, эти технологии уже можно использовать в проектах, подключая соответствующие функции через API – эту возможность предоставляют гиганты технологической индустрии.

**Почему Data Science – это важно?**

Data Science и технологии искусственного интеллекта позволяет больше узнать о том, что предпочитает человек (собирая и анализируя данные), стать ближе к нему, создавая более персонализированные интерфейсы (например, отбирая предложения в соответствии с тем, что ранее было интересно пользователю, отправляя персонализированные рассылки) и т.д. Для IT-отрасли возможность работы с данными представляет собой такой большой качественный скачок, что новые стартапы нельзя представить себе без применения этой технологии – это все равно что продолжать использовать лошадей для перевозки в эпоху расцвета автомобилей. А ведь сам термин IT-стартап подразумевает инновационность.

**Наиболее распространенные алгоритмы машинного обучения**

**Линейная регрессия**

Это один из самых известных алгоритмов в статистике и машинном обучении.

Основная концепция – в основном линейная регрессия – это линейная модель, которая предполагает линейную зависимость между входными переменными, скажем, х, и единственной выходной переменной, скажем, у. Другими словами, мы можем сказать, что у можно рассчитать из линейной комбинации входных переменных х. Связь между переменными может быть установлена ​​путем подбора наилучшей линии.

**Типы линейной регрессии**

Линейная регрессия бывает следующих двух типов:

**Простая линейная регрессия** . Алгоритм линейной регрессии называется простой линейной регрессией, если он имеет только одну независимую переменную.

**Множественная линейная регрессия** . Алгоритм линейной регрессии называется множественной линейной регрессией, если он имеет более одной независимой переменной.

Линейная регрессия в основном используется для оценки реальных значений на основе непрерывных переменных. Например, общая продажа магазина за день, основанная на реальных значениях, может быть оценена с помощью линейной регрессии.

**Логистическая регрессия**

Это алгоритм классификации, также известный как регрессия логита .

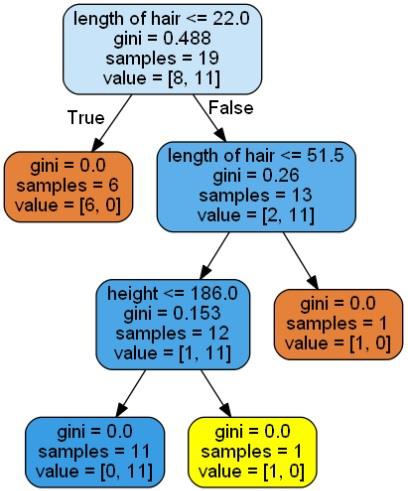
В основном логистическая регрессия – это алгоритм классификации, который используется для оценки дискретных значений, таких как 0 или 1, истина или ложь, да или нет на основе заданного набора независимых переменных. По сути, он предсказывает вероятность, следовательно, его вывод лежит в диапазоне от 0 до 1.

**Древо решений**

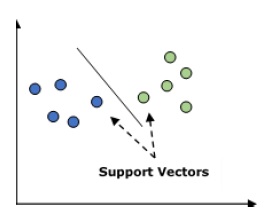
Дерево решений – это контролируемый алгоритм обучения, который в основном используется для задач классификации.

По сути, это классификатор, выраженный как рекурсивное разбиение на основе независимых переменных. Дерево решений имеет узлы, которые образуют корневое дерево. Корневое дерево – это направленное дерево с узлом под названием «корень». Корень не имеет входящих ребер, а все остальные узлы имеют одно входящее ребро. Эти узлы называются узлами или узлами решения. Например, рассмотрите следующее дерево решений, чтобы увидеть, подходит ли человек или нет.

Древо решений



**Машина опорных векторов (SVM)**

Он используется для задач классификации и регрессии. Но в основном это используется для задач классификации. Основная концепция SVM заключается в построении каждого элемента данных в виде точки в n-мерном пространстве, причем значением каждого объекта является значение определенной координаты. Здесь n будет теми функциями, которые у нас были бы. Ниже приводится простое графическое представление, чтобы понять концепцию SVM –

**Машина опорных векторов**

На приведенной выше диаграмме у нас есть две особенности, поэтому нам сначала нужно построить эти две переменные в двухмерном пространстве, где каждая точка имеет две координаты, называемые опорными векторами. Линия разбивает данные на две разные классифицированные группы. Эта строка будет классификатором.

**Наивный Байес**

Это также метод классификации. Логика этой методики классификации заключается в использовании теоремы Байеса для построения классификаторов. Предполагается, что предикторы независимы. Проще говоря, это предполагает, что наличие определенной функции в классе не связано с наличием любой другой функции. Ниже приведено уравнение для теоремы Байеса –

P left( fracAB right)= fracP left( fracBA right)P left(A right)P left(B right)

Наивная байесовская модель проста в построении и особенно полезна для больших наборов данных.

**K-Ближайшие соседи (KNN)**

Он используется для классификации и регрессии проблем. Он широко используется для решения задач классификации. Основная концепция этого алгоритма заключается в том, что он используется для хранения всех доступных случаев и классифицирует новые случаи большинством голосов своих k соседей. Затем случай присваивается классу, который является наиболее распространенным среди его K-ближайших соседей, что измеряется функцией расстояния. Функция расстояния может быть Евклидовым, Минковским и расстоянием Хэмминга. Рассмотрим следующее, чтобы использовать KNN – В вычислительном отношении KNN дороже, чем другие алгоритмы, используемые для задач классификации.Необходима нормализация переменных, в противном случае переменные более высокого диапазона могут смещать ее.

В KNN нам нужно работать на стадии предварительной обработки, такой как удаление шума.В вычислительном отношении KNN дороже, чем другие алгоритмы, используемые для задач классификации.

Необходима нормализация переменных, в противном случае переменные более высокого диапазона могут смещать ее.В KNN нам нужно работать на стадии предварительной обработки, такой как удаление шума.

**K-средства кластеризации**

Как следует из названия, он используется для решения проблем кластеризации. Это в основном тип обучения без присмотра. Основная логика алгоритма кластеризации K-Means заключается в классификации набора данных по ряду кластеров. Выполните следующие шаги, чтобы сформировать кластеры с помощью K-средних –

K-means выбирает k количество точек для каждого кластера, известного как центроиды.Теперь каждая точка данных образует кластер с ближайшими центроидами, т.е. k кластерами.Теперь он найдет центроиды каждого кластера на основе существующих членов кластера.Нам нужно повторять эти шаги, пока не произойдет сближение.K-means выбирает k количество точек для каждого кластера, известного как центроиды.Теперь каждая точка данных образует кластер с ближайшими центроидами, т.е. k кластерами.Теперь он найдет центроиды каждого кластера на основе существующих членов кластера.Нам нужно повторять эти шаги, пока не произойдет сближение.

**Случайный Лес**

Это контролируемый алгоритм классификации. Преимущество алгоритма случайного леса состоит в том, что его можно использовать как для задач классификации, так и для задач регрессии. В основном это коллекция деревьев решений (т. Е. Лес) или, можно сказать, ансамбль деревьев решений. Основная концепция случайного леса состоит в том, что каждое дерево дает классификацию, и лес выбирает из них лучшие классификации. Ниже приведены преимущества алгоритма Random Forest –Классификатор случайных лесов может использоваться как для задач классификации, так и для задач регрессии.Они могут обрабатывать пропущенные значения.Это не будет соответствовать модели, даже если у нас будет больше деревьев в лесу.

Классификатор случайных лесов может использоваться как для задач классификации, так и для задач регрессии.Они могут обрабатывать пропущенные значения.Это не будет соответствовать модели, даже если у нас будет больше деревьев в лесу.

**Язык Python в системах искусственного интеллекта**

## **Что делает Python лучшим языком программирования в сфере машинного обучения и ИИ?**

Разработка приложений с использованием ИИ отличается от обычной разработки. Для работы с ИИ нужен особый стек технологий и особые навыки. Кроме того, создание приложений на основе ИИ требует глубоких исследований.Для реализации ваших идей, связанных с применением ИИ, вам понадобится надежный, гибкий язык программирования с богатым инструментарием. Python — именно такой язык, и поэтому сегодня на нем разрабатывается множество ИИ-проектов.

Python помогает разработчикам работать продуктивно и уверенно, причем на всех стадиях проекта, от разработки до поддержки. Этот язык обладает определенными характеристиками, которые делают его наилучшим выбором для ML- и ИИ-проектов: он простой и логичный, гибкий и мультиплатформенный, имеет отличные библиотеки и фреймворки для машинного обучения и работы с ИИ, а еще за ним стоит многочисленное сообщество разработчиков. Благодаря всему этому Python является одним из самых популярных языков программирования в мире, причем не только в сфере ML и ИИ.

### ****Простота и логичность****

В Python легко писать лаконичный и читаемый код. Несмотря на то, что за машинным обучением и искусственным интеллектом стоят сложные алгоритмы и процессы, простота Python позволяет создавать надежные системы. Разработчики могут полностью сосредоточиться на задачах, которые они пытаются решить при помощи ML, не отвлекаясь на технические нюансы языка.

Кроме того, Python прост в изучении, чем и привлекает многих разработчиков. Написанный на нем код легко понятен человеку, что упрощает создание моделей для машинного обучения.Кто-то считает, что Python куда интуитивнее других языков программирования. Кто-то указывает на наличие множества фреймворков, библиотек и расширений, упрощающих реализацию задуманных функций. И все соглашаются, что Python хорошо подходит для командной разработки.

Поскольку Python — язык общего назначения, с его помощью можно решать многие сложные задачи машинного обучения и быстро создавать прототипы для последующей их отладки.

### ****Богатый выбор библиотек и фреймворков****

Создание алгоритмов ИИ и машинного обучения — это сложная задача, требующая много времени. И чтобы было легче находить оптимальные пути решения задач, программистам нужна хорошо структурированная и надежная среда разработки.

Многочисленные фреймворки и библиотеки Python помогают существенно уменьшить количество времени, необходимого для разработки приложений.

Программная библиотека — это предварительно написанный код, который разработчики используют для решения распространенных задач. Python, с его богатым стеком технологий, имеет обширный набор библиотек для искусственного интеллекта и машинного обучения. Вот некоторые из них:

* Keras, TensorFlow, и Scikit-learn — для машинного обучения
* NumPy — для высокопроизводительных научных вычислений и анализа данных
* SciPy — для продвинутых вычислений
* Pandas — для общего анализа данных
* Seaborn — для визуализации данных

Scikit-learn предлагает различные алгоритмы классификации, регрессии и кластеризации, включая вспомогательные векторные машины, случайные леса, повышение градиента, k-средних и DBSCAN. Эта библиотека предназначена для работы с числовыми и научными библиотеками Python NumPy и SciPy.С помощью этих готовых решений вы сможете создавать свои продукты гораздо быстрее. Вашей команде разработчиков не придется заново изобретать велосипед, т. к. для реализации необходимых функций они смогут использовать существующие библиотеки.

Вот таблица с распространенными вариантами использования ИИ и технологиями, которые мы можем для них порекомендовать:

|  |  |
| --- | --- |
| Машинное обучение | TensorFlow, Keras, Scikit-learn |
| Компьютерное зрение | OpenCV |
| Обработка естественного языка | NLTK, spaCy |

### ****Мультиплатформенность****

Мультиплатформенность (в нашем случае) — это свойство языка программирования или фреймворка, позволяющее разработчикам переносить ПО на разные машины с минимальными изменениями, либо без изменений вовсе.Одной из причин популярности Python является то, что этот язык от платформ не зависит, так как поддерживается многими из них, включая Linux, Windows и macOS.

Код Python может использоваться для создания программ для большинства операционных систем, а это означает, что программное обеспечение Python легко распространять и использовать в этих системах без специальных интерпретаторов.

Обычно для своих вычислительных нужд разработчики используют такие сервисы, как Google или Amazon. Но они также могут использовать свои собственные машины с мощными графическими процессорами (GPU) для обучения своих моделей машинного обучения. А тот факт, что Python независим от платформы, делает это обучение намного проще — и дешевле.

Python настолько надежен, что Google использует его для сканирования веб-страниц, Pixar — при создании фильмов, а Spotify — для рекомендации песен.

**Глава №2Применение логических методов**

**Часть 1 . Заполнение данными файла , для обучения бота**

Первым делом необходимо создать файл с данными которым бот будет обучатся . Проанализировав источники приведенные в разделе библиография , был создан файл intents ,куда были записаны запросы которые бот может получить , а также ответы которыми бот может ответить пользователю.

{  
 "intents": [  
 {  
 "tag": "greeting",  
 "patterns": [  
 "Hi",  
 "Hey",  
 "How are you",  
 "Is anyone there?",  
 "Hello",  
 "Good day"  
 ],  
 "responses": [  
 "Hey :-)",  
 "Hello, thanks for visiting",  
 "Hi there, what can I do for you?",  
 "Hi there, how can I help?"  
 ]  
 },  
 {  
 "tag": "goodbye",  
 "patterns": ["Bye", "See you later", "Goodbye"],  
 "responses": [  
 "See you later, thanks for visiting",  
 "Have a nice day",  
 "Bye! Come back again soon."  
 ]  
 },  
 {  
 "tag": "thanks",  
 "patterns": ["Thanks", "Thank you", "That's helpful", "Thank's a lot!"],  
 "responses": ["Happy to help!", "Any time!", "My pleasure"]  
 },  
 {  
 "tag": "items",  
 "patterns": [  
 "Which items do you have?",  
 "What kinds of items are there?",  
 "What do you sell?"  
 ],  
 "responses": [  
 "We sell coffee and tea",  
 "We have coffee and tea"  
 ]  
 },  
 {  
 "tag": "payments",  
 "patterns": [  
 "Do you take credit cards?",  
 "Do you accept Mastercard?",  
 "Can I pay with Paypal?",  
 "Are you cash only?"  
 ],  
 "responses": [  
 "We accept VISA, Mastercard and Paypal",  
 "We accept most major credit cards, and Paypal"  
 ]  
 },  
 {  
 "tag": "delivery",  
 "patterns": [  
 "How long does delivery take?",  
 "How long does shipping take?",  
 "When do I get my delivery?"  
 ],  
 "responses": [  
 "Delivery takes 2-4 days",  
 "Shipping takes 2-4 days"  
 ]  
 },  
 {  
 "tag": "funny",  
 "patterns": [  
 "Tell me a joke!",  
 "Tell me something funny!",  
 "Do you know a joke?"  
 ],  
 "responses": [  
 "Why did the hipster burn his mouth? He drank the coffee before it was cool.",  
 "What did the buffalo say when his son left for college? Bison."  
 ]  
 }  
 ]  
}

**Часть 2 . Установка библиотеки PyTorch и создание модели.**

PyTorch это библиотека машинного обучения с открытым исходным кодом, используемая для разработки и обучения нейронная сеть основанная на модели глубокого обучения. Библиотека в основном разработана исследовательской группой Фейсбука. PyTorch может использоваться как с Python, так и с C ++. Естественно, интерфейс Python более отточен. Pytorch (при поддержке таких крупных компаний, как Facebook, Microsoft, SalesForce, Uber) пользуется огромной популярностью в исследовательских лабораториях. Пока еще не на многих производственных серверах, управляемых такими системами, как TensorFlow (при поддержке Google), Pytorch набирает обороты.

В отличие от большинства других популярных систем глубокого обучения, таких как[TensorFlow](https://thewiz.net/eULN), которые используют статические графы вычислений, PyTorch использует динамические вычисления, что обеспечивает большую гибкость при построении сложных архитектур. Pytorch использует основные понятия Python, такие как классы, структуры и условные циклы, которые нам знакомы, а значит, более интуитивны для понимания. Это делает его намного проще, чем другие фреймворки, такие как TensorFlow, которые привносят свой собственный стиль программирования.

Для установки данного framework , необходимо ввести в терминал команду

Pip install torch

Среда разработки установит все необходимые для этого файлы и связи.

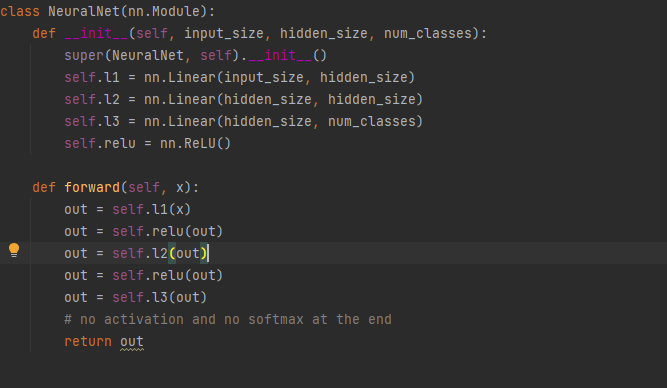
Идиома определения модели в PyTorch включает определение класса, расширяющего класс модуля.

Конструктор вашего класса определяет слои модели, а функция forward() является переопределением, которое определяет, как пересылать входные данные через определенные слои модели.

Доступны многие слои, такие как Linear для полносвязных слоев, Conv2d для сверточных слоев и MaxPool2d для объединения слоев.

Функции активации также могут быть определены как слои, такие как ReLU, Softmax и Sigmoid.

Ниже приведен пример модели.



**Часть 3 . Предварительная обработка данных , использование NLTK.**

NLTK – это библиотека Python, которую можно использовать в любом приложении для обработки естественного языка. От преобразования текстовых данных до создания приложения на основе NLP, такого как анализатор настроений, распознавание именованных сущностей и т. д. Все это можно сделать с помощью библиотеки NLTK в Python.

Предварительная обработка необходима для преобразования текстовых данных в чистый набор данных, который в конечном итоге будет передан в наши модели чат-ботов Python.Необходимо создать метод класса process\_data(), который будет выполнять различные операции с нашими текстовыми данными.

а. Токенизация: токенизация — это процесс разделения потока текстов, таких как предложения, на более мелкие фрагменты (токены), такие как слова.

В данном случае необходимо использовать модуль обработки естественного языка с именем «nltk», который содержит функцию «word\_tokenize()» для процесса токенизации. В этом методе перебираются шаблоны и токенизировать каждое предложение. Для итерации необходимо использовать встроенную функцию карты Python.

Функция Map используется для связывания «Функций» с каждым элементом «Итерируемых объектов» и возврата генератора. Этот генератор может быть позже преобразован в списки или любой другой тип данных. Давайте посмотрим, что такое аргументы картографических функций:

1. Функция: вы можете создать свою функцию, используя ключевое слово «def» и поместить имя функции в карту в качестве первого параметра, или вы можете использовать выражение лямбда-функции. Для получения более подробной информации о лямбда-функции вы можете посетить Python Lambda Expression.

2. Iterables: Iterables — это не что иное, как объект Python, который может возвращать свой член по одному. Примерами могут быть строки, списки, наборы, кортежи и т. д. Если вы хотите, вы можете применить больше итераций, разделенных запятыми, в функции карты.

import numpy as np  
import nltk  
# nltk.download('punkt')  
from nltk.stem.porter import PorterStemmer  
stemmer = PorterStemmer()  
  
def tokenize(sentence):  
 *"""  
 split sentence into array of words/tokens  
 a token can be a word or punctuation character, or number  
 """* return nltk.word\_tokenize(sentence)  
  
  
def stem(word):  
 *"""  
 stemming = find the root form of the word  
 examples:  
 words = ["organize", "organizes", "organizing"]  
 words = [stem(w) for w in words]  
 -> ["organ", "organ", "organ"]  
 """* return stemmer.stem(word.lower())  
  
  
def bag\_of\_words(tokenized\_sentence, words):  
 *"""  
 return bag of words array:  
 1 for each known word that exists in the sentence, 0 otherwise  
 example:  
 sentence = ["hello", "how", "are", "you"]  
 words = ["hi", "hello", "I", "you", "bye", "thank", "cool"]  
 bog = [ 0 , 1 , 0 , 1 , 0 , 0 , 0]  
 """* # stem each word  
 sentence\_words = [stem(word) for word in tokenized\_sentence]  
 # initialize bag with 0 for each word  
 bag = np.zeros(len(words), dtype=np.float32)  
 for idx, w in enumerate(words):  
 if w in sentence\_words:   
 bag[idx] = 1  
  
 return bag

После каждой итерации будут получены слова или фрагменты из этих предложений, которые необходимо хранить в переменной «words». После этого была создана переменная «документ», которая будет иметь объект кортежа, содержащий два данных. Первыми будут токенизированные слова, а вторыми будут соответствующие теги или имена классов.

После отделения слов от шаблонов следующим шагом будет знание значения слов. Для этого необходимо использовать:

а. Стемминг

Стемминг - это процесс сведения слов к их основе слова, то есть к корню слов. Если запрос пользователя содержит слово с ошибкой, такое как «сумасшедший», определение корня уменьшит его до «сумасшедшего», которое является корнем слова. Основная проблема с определением корней заключается в том, что это может привести к неправильному значению. Например, если слово ошибки «орд», оно будет происходить от корневого слова «орд». Мы не можем сделать вывод, что пользователь ищет слово «орд» или «испытание», основываясь на этом.

б. Лемматизация

Лемматизация — это процесс сведения слов к их корневым словам, который гарантирует, что слово имеет смысл или находится в словарной форме. В лемматизации корневое слово называется леммой. Лемматизация способна поддерживать контекст шаблона запроса пользователя, что помогает получить значимые слова.

В этом проекте будет использован метод стемминга, также можно попробовать метод лемматизации. После лемматизации слов необходимо сохранить их в переменной «words», а также удалить все символы, если они присутствуют в токенизированных словах, и все слова в нижнем регистре.

**Часть 4 . Создание файла “тренировки”**

В данном файле будут обьеденены все выше сказанные методы , для построения модели бота , а также логики ответов.

Был создан класс с названием «Обучение». В конструкторе этого класса инициализируемы все переменные, которые будут доступны через все методы класса. Итак, сначала необходимо прочитать файл «intents.json» и проанализировать файл намерений в формате JSON с помощью модуля json в переменную «намерения», используя json. Переменная «игнорировать слова» будет содержать все символы, которые мы собираемся удалить из шаблонов.

Эта модель будет иметь 3 слоя: входной слой, скрытый слой и выходной слой. Мы будем использовать функцию Keras TensorFlow для создания модели.Будет использованна Dropout, чтобы предотвратить переобучение между слоями. Кроме того, наш входной и скрытый слои будут иметь функцию активации «relu», а выходной плотный слой будет иметь функцию активации «softmax». Мы собираемся использовать оптимизатор SGD со скоростью обучения 0,01 и обучать нашу модель в течение 200 милисеккунд эпох в массиве размером 10.

import numpy as np  
import random  
import json  
  
import torch  
import torch.nn as nn  
from torch.utils.data import Dataset, DataLoader  
  
from nltk\_utils import bag\_of\_words, tokenize, stem  
from model import NeuralNet  
  
with open('intents.json', 'r') as f:  
 intents = json.load(f)  
  
all\_words = []  
tags = []  
xy = []  
# loop through each sentence in our intents patterns  
for intent in intents['intents']:  
 tag = intent['tag']  
 # add to tag list  
 tags.append(tag)  
 for pattern in intent['patterns']:  
 # tokenize each word in the sentence  
 w = tokenize(pattern)  
 # add to our words list  
 all\_words.extend(w)  
 # add to xy pair  
 xy.append((w, tag))  
  
# stem and lower each word  
ignore\_words = ['?', '.', '!']  
all\_words = [stem(w) for w in all\_words if w not in ignore\_words]  
# remove duplicates and sort  
all\_words = sorted(set(all\_words))  
tags = sorted(set(tags))  
  
print(len(xy), "patterns")  
print(len(tags), "tags:", tags)  
print(len(all\_words), "unique stemmed words:", all\_words)  
  
# create training data  
X\_train = []  
y\_train = []  
for (pattern\_sentence, tag) in xy:  
 # X: bag of words for each pattern\_sentence  
 bag = bag\_of\_words(pattern\_sentence, all\_words)  
 X\_train.append(bag)  
 # y: PyTorch CrossEntropyLoss needs only class labels, not one-hot  
 label = tags.index(tag)  
 y\_train.append(label)  
  
X\_train = np.array(X\_train)  
y\_train = np.array(y\_train)  
  
# Hyper-parameters   
num\_epochs = 1000  
batch\_size = 8  
learning\_rate = 0.001  
input\_size = len(X\_train[0])  
hidden\_size = 8  
output\_size = len(tags)  
print(input\_size, output\_size)  
  
class ChatDataset(Dataset):  
  
 def \_\_init\_\_(self):  
 self.n\_samples = len(X\_train)  
 self.x\_data = X\_train  
 self.y\_data = y\_train  
  
 # support indexing such that dataset[i] can be used to get i-th sample  
 def \_\_getitem\_\_(self, index):  
 return self.x\_data[index], self.y\_data[index]  
  
 # we can call len(dataset) to return the size  
 def \_\_len\_\_(self):  
 return self.n\_samples  
  
dataset = ChatDataset()  
train\_loader = DataLoader(dataset=dataset,  
 batch\_size=batch\_size,  
 shuffle=True,  
 num\_workers=0)  
  
device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')  
  
model = NeuralNet(input\_size, hidden\_size, output\_size).to(device)  
  
# Loss and optimizer  
criterion = nn.CrossEntropyLoss()  
optimizer = torch.optim.Adam(model.parameters(), lr=learning\_rate)  
  
# Train the model  
for epoch in range(num\_epochs):  
 for (words, labels) in train\_loader:  
 words = words.to(device)  
 labels = labels.to(dtype=torch.long).to(device)  
   
 # Forward pass  
 outputs = model(words)  
 # if y would be one-hot, we must apply  
 # labels = torch.max(labels, 1)[1]  
 loss = criterion(outputs, labels)  
   
 # Backward and optimize  
 optimizer.zero\_grad()  
 loss.backward()  
 optimizer.step()  
   
 if (epoch+1) % 100 == 0:  
 print (f'Epoch [{epoch+1}/{num\_epochs}], Loss: {loss.item():.4f}')  
  
  
print(f'final loss: {loss.item():.4f}')  
  
data = {  
"model\_state": model.state\_dict(),  
"input\_size": input\_size,  
"hidden\_size": hidden\_size,  
"output\_size": output\_size,  
"all\_words": all\_words,  
"tags": tags  
}  
  
FILE = "data.pth"  
torch.save(data, FILE)  
  
print(f'training complete. file saved to {FILE}')

**Часть 5 . Создание файла “chat”**

В данном файле используется вся выше сказанная логика . Данный файл являеться предпоследним связующим лицом между ботом и пользователем . Здесь необходимо написать размер блоков с сообщениями в зависимости от длины слова , также получая разные запросы на одну и туже тему , необходимо чтобы бот также отвечал по разному , для этого используеться библиотека random.Также в данном файлы мы задаем имя боту . После чего создаем и определяем метод получения запросов , суть в том , что каждый запрос токенизируется , после чего сравниваеться уже с созданным файлом data , в котором содержится весь “мозг ” бота .После обработки данного запроса бот либо же отвечает , либо говорит , что не понимает.

import random  
import json  
  
import torch  
  
from model import NeuralNet  
from nltk\_utils import bag\_of\_words, tokenize  
  
device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')  
  
with open('intents.json', 'r') as json\_data:  
 intents = json.load(json\_data)  
  
FILE = "data.pth"  
data = torch.load(FILE)  
  
input\_size = data["input\_size"]  
hidden\_size = data["hidden\_size"]  
output\_size = data["output\_size"]  
all\_words = data['all\_words']  
tags = data['tags']  
model\_state = data["model\_state"]  
  
model = NeuralNet(input\_size, hidden\_size, output\_size).to(device)  
model.load\_state\_dict(model\_state)  
model.eval()  
  
bot\_name = "Sam"  
  
def get\_response(msg):  
 sentence = tokenize(msg)  
 X = bag\_of\_words(sentence, all\_words)  
 X = X.reshape(1, X.shape[0])  
 X = torch.from\_numpy(X).to(device)  
  
 output = model(X)  
 \_, predicted = torch.max(output, dim=1)  
  
 tag = tags[predicted.item()]  
  
 probs = torch.softmax(output, dim=1)  
 prob = probs[0][predicted.item()]  
 if prob.item() > 0.75:  
 for intent in intents['intents']:  
 if tag == intent["tag"]:  
 return random.choice(intent['responses'])  
   
 return "I do not understand..."

**Часть 6 . Cоздание графического интерфейса с помощью tkinter.**

Tkinter – это пакет для Python, предназначенный для работы с библиотекой Tk. Библиотека Tk содержит компоненты графического интерфейса пользователя (graphical user interface – GUI). Эта библиотека написана на языке программирования Tcl.

Под графическим интерфейсом пользователя (GUI) подразумеваются все те окна, кнопки, текстовые поля для ввода, скроллеры, списки, радиокнопки, флажки и др., которые вы видите на экране, открывая то или иное приложение. Через них вы взаимодействуете с программой и управляете ею. Все эти элементы интерфейса будем называть виджетами (widgets).

В настоящее время почти все приложения, которые создаются для конечного пользователя, имеют GUI. Редкие программы, подразумевающие взаимодействие с человеком, остаются консольными. В предыдущих двух курсах мы писали только консольные программы.

from tkinter import \*  
from chat import get\_response, bot\_name  
  
BG\_GRAY = "#ABB2B9"  
BG\_COLOR = "#17202A"  
TEXT\_COLOR = "#EAECEE"  
  
FONT = "Helvetica 14"  
FONT\_BOLD = "Helvetica 13 bold"  
  
class ChatApplication:  
   
 def \_\_init\_\_(self):  
 self.window = Tk()  
 self.\_setup\_main\_window()  
   
 def run(self):  
 self.window.mainloop()  
   
 def \_setup\_main\_window(self):  
 self.window.title("Chat")  
 self.window.resizable(width=False, height=False)  
 self.window.configure(width=470, height=550, bg=BG\_COLOR)  
   
 # head label  
 head\_label = Label(self.window, bg=BG\_COLOR, fg=TEXT\_COLOR,  
 text="Welcome", font=FONT\_BOLD, pady=10)  
 head\_label.place(relwidth=1)  
   
 # tiny divider  
 line = Label(self.window, width=450, bg=BG\_GRAY)  
 line.place(relwidth=1, rely=0.07, relheight=0.012)  
   
 # text widget  
 self.text\_widget = Text(self.window, width=20, height=2, bg=BG\_COLOR, fg=TEXT\_COLOR,  
 font=FONT, padx=5, pady=5)  
 self.text\_widget.place(relheight=0.745, relwidth=1, rely=0.08)  
 self.text\_widget.configure(cursor="arrow", state=DISABLED)  
   
 # scroll bar  
 scrollbar = Scrollbar(self.text\_widget)  
 scrollbar.place(relheight=1, relx=0.974)  
 scrollbar.configure(command=self.text\_widget.yview)  
   
 # bottom label  
 bottom\_label = Label(self.window, bg=BG\_GRAY, height=80)  
 bottom\_label.place(relwidth=1, rely=0.825)  
   
 # message entry box  
 self.msg\_entry = Entry(bottom\_label, bg="#2C3E50", fg=TEXT\_COLOR, font=FONT)  
 self.msg\_entry.place(relwidth=0.74, relheight=0.06, rely=0.008, relx=0.011)  
 self.msg\_entry.focus()  
 self.msg\_entry.bind("<Return>", self.\_on\_enter\_pressed)  
   
 # send button  
 send\_button = Button(bottom\_label, text="Send", font=FONT\_BOLD, width=20, bg=BG\_GRAY,  
 command=lambda: self.\_on\_enter\_pressed(None))  
 send\_button.place(relx=0.77, rely=0.008, relheight=0.06, relwidth=0.22)  
   
 def \_on\_enter\_pressed(self, event):  
 msg = self.msg\_entry.get()  
 self.\_insert\_message(msg, "You")  
   
 def \_insert\_message(self, msg, sender):  
 if not msg:  
 return  
   
 self.msg\_entry.delete(0, END)  
 msg1 = f"{sender}: {msg}\n\n"  
 self.text\_widget.configure(state=NORMAL)  
 self.text\_widget.insert(END, msg1)  
 self.text\_widget.configure(state=DISABLED)  
   
 msg2 = f"{bot\_name}: {get\_response(msg)}\n\n"  
 self.text\_widget.configure(state=NORMAL)  
 self.text\_widget.insert(END, msg2)  
 self.text\_widget.configure(state=DISABLED)  
   
 self.text\_widget.see(END)  
   
   
if \_\_name\_\_ == "\_\_main\_\_":  
 app = ChatApplication()  
 app.run()

**Выводы.**

Компьютерный или искусственный интеллект оказывает глубокое влияние на мир, в котором мы живем, и число новых приложений постоянно растет. Все больше отличных программистов выбирают Python для решения различных задач искусственного интеллекта и глубокого обучения.Широкий выбор разнообразных AI-библиотек сильно упрощает разработку и снижает затраченное на нее время. Простая и понятная базовая грамматика Python помогает тестировать приложения и делает язык доступным для начинающих разработчиков.Это также снижает психологическую нагрузку c разработчиков, давая им возможность сосредоточиться на главной цели. И наконец, простой синтаксис упрощает командную работу, снимая «барьеры» между программистами.

Вдобавок, вокруг Python расцвело большое сообщество разработчиков, которые всегда готовы прийти на выручку в трудную минуту. При решении сложных задач это может оказаться важным.Хотя в AI-проектах также могут быть использованы и другие языки программирования, нельзя отрицать того, что Python находится на «передовой». Поэтому при возникновении задач, связанных с искусственным интеллектом, стоит в первую очередь подумать о применении Python.

**Библиография**

**1 https://www.calltouch.ru/glossary/iskusstvennyy-intellekt/**

**2 https://www.sas.com/ru\_ru/insights/articles/analytics/what-is-artificial-intelligence.html**

**3https://www.studmed.ru/ashinyanc-r-a-logicheskie-metody-v-iskusstvennom-intellekte\_24ae7469b2c.html**

**4 https://aisimple.ru/6-methody-ai.html**

**5https://lfirmal.com/kursovaya-rabota-na-temu-iskusstvennyy intellekt/#%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5\_%D0%B2\_%D0%B8%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9\_%D0%B8%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82**

**6 https://www.ngpedia.ru/id159039p1.html**

**7 https://www.anti-malware.ru/practice/methods/data-mining**

**8 https://basegroup.ru/community/articles/data-mining-loyality**

**9 https://cyberleninka.ru/article/n/ekspertnaya-sistema-podderzhki-prinyatiya-resheniy**

**10**[**https://www.it.ua/ru/knowledge-base/technology-innovation/iskusstvennye-nejronnye-seti-ins**](https://www.it.ua/ru/knowledge-base/technology-innovation/iskusstvennye-nejronnye-seti-ins)

**11 https://younglinux.info/tkinter/tkinter**

**12 https://machinelearningmastery.com/pytorch-tutorial-develop-deep-learning-models/**

**13 https://discuss.pytorch.org/t/inference-with-a-custom-model/115675**

**14 https://www.machinelearningmastery.ru/introduction-to-py-torch-13189fb30cb3/**

**15 https://coderlessons.com/tutorials/python-technologies/uznaite-pytorch/pytorch-kratkoe-rukovodstvo**

**Приложения**

