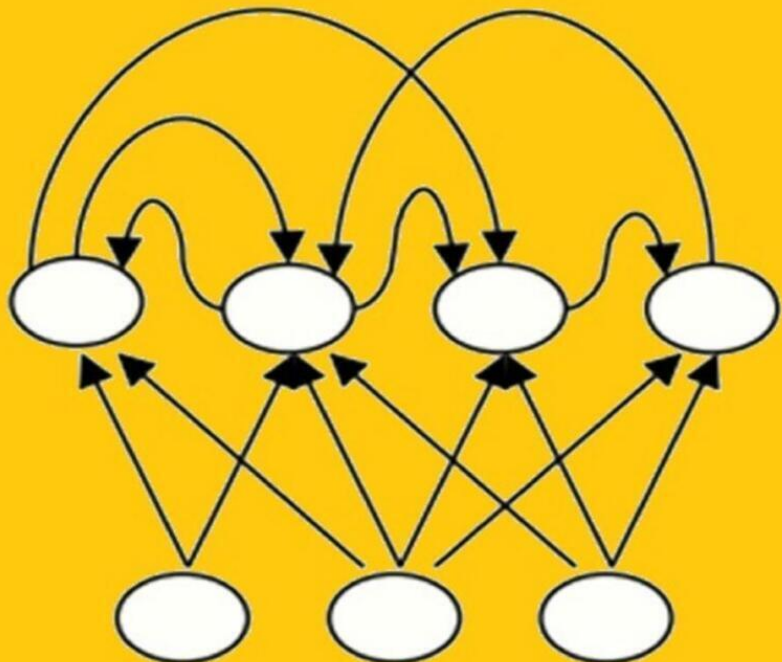
НАУЧИТЕСЬ ИСПОЛЬЗОВАТЬ PYTHON ДЛЯ СОЗДАНИЯ ИИ

МАШИННОЕ ОБУЧЕНИЕ И ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ



ТИМУР МАШНИН

Тимур Машнин

Машинное обучение и Искусственный Интеллект

«Автор»

2022

Машнин Т.

Машинное обучение и Искусственный Интеллект / Т. Машнин — «Автор», 2022

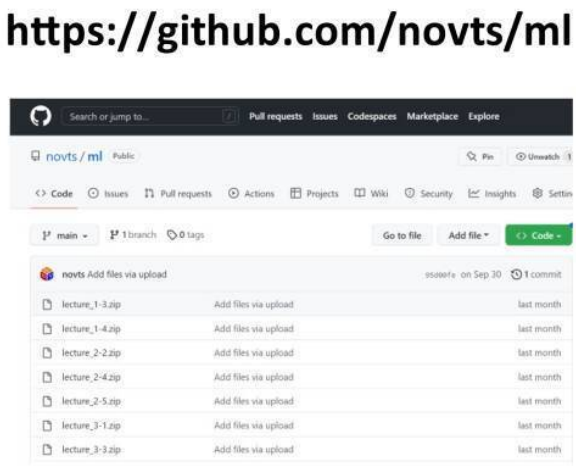
Эта книга будет интересна всем, кто хочет научиться применять Python 3 при разработке в области искусственного интеллекта и машинного обучения.С этой книгой Вы познакомитесь с основными терминами и понятиями ИИ, такими как машинное обучение, глубокое обучение и нейронные сети.Научитесь создавать чат-ботов с использованием IBM Watson на платформе Watson Assistant.Узнаете как можно использовать искусственный интеллект IBM Watson для своих собственных разработок.На реальных примерах познакомитесь с такими алгоритмами машинного обучения, как регрессия, классификация и кластеризация.Познакомитесь с глубоким машинным обучением и научитесь создавать нейронные сети с Keras и TensorFlow.

Содержание

Тимур Машнин
Машинное обучение и Искусственный Интеллект

Исходный код

Исходный код к примерам можно скачать с сайта GitHub.



Введение

ИИ резко меняет нашу жизнь и формирует наше будущее.

Всемирный экономический форум недавно опубликовал отчет, в котором говорится, что в период до 2022 года около 75 миллионов рабочих мест могут быть потеряны из-за машин.

Далее в отчете говорится, что 133 миллиона новых рабочих мест могут появиться из-за быстрого развития машин и ИИ.

Это означает 58 миллионов новых рабочих мест, которые будут созданы в ближайшие несколько лет.

ИИ состоит из 2-х слов Искусственный и интеллект.

Все, что не естественно и создано людьми, является искусственным.

Интеллект означает способность понимать, рассуждать, планировать и т. д.

Таким образом, мы можем сказать, что любой код, технология или алгоритм, которые позволяют машине имитировать, развивать или демонстрировать человеческое познание или поведение, являются ИИ.

И в наши дни много говорят об искусственном интеллекте.

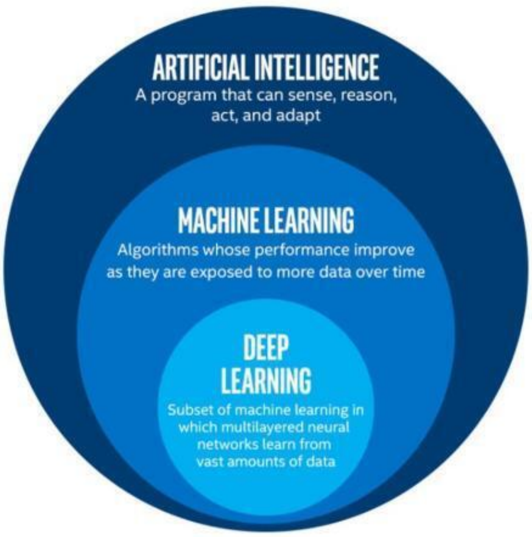
Почему вырос интерес к ИИ?

Причина в том, что раньше у нас было очень мало данных.

Но в наши дни наблюдается огромный рост объема данных, которые генерируются каждую минуту и помогают нам делать более точные прогнозы.

Наряду с огромным объемом данных у нас также появились более совершенные алгоритмы, вычислительные мощности и хранилища данных, которые могут справиться с таким огромным объемом данных.

И искусственный интеллект – это очень широкое понятие, в которое входят машинное обучение и глубокое обучение.



В широком смысле ИИ – это научить машины действовать и думать, как люди.

Это создание программ, которые будут сами действовать без участия человека.

Другое направление ИИ связано с тем, чтобы дать машинам больше когнитивных и сенсорных возможностей.

Это направление связано с анализом изображений и видео, с обработкой и пониманием речи и так далее.

Это создание технологий, которые в некоторых случаях могут заменить то, что делают люди.

И это создание приложений для интеллектуального решения проблем с использованием алгоритмов.

Это разработка программ или алгоритмов таким образом, чтобы они могли учиться и совершенствоваться с течением времени под воздействием новых данных.

Таким образом, искусственный интеллект может быть чем-то, что имитирует человеческий интеллект.

Это в широком смысле.

Более узкое направление – это может быть чисто вычислительный подход и подход для оптимизации, в котором производится манипуляция данными таким образом, чтобы получить неочевидные результаты.

И здесь ИИ – это инструмент, который используется компьютером для автоматического выполнения задач практически без вмешательства человека.

ИИ – это сложная серия алгоритмов, которые что-то делают с поступающей информацией.

Мы будем исходить из того, что искусственный интеллект – это набор технологий, который позволяет извлекать знания из данных.

Таким образом, это любая система, которая изучает или понимает шаблоны или закономерности в этих данных и может идентифицировать их, а затем воспроизводить их на новой информации.

Мы будем исходить из того, что искусственный интеллект – это не симулятор человеческого интеллекта, а ИИ – это машинное обучение.

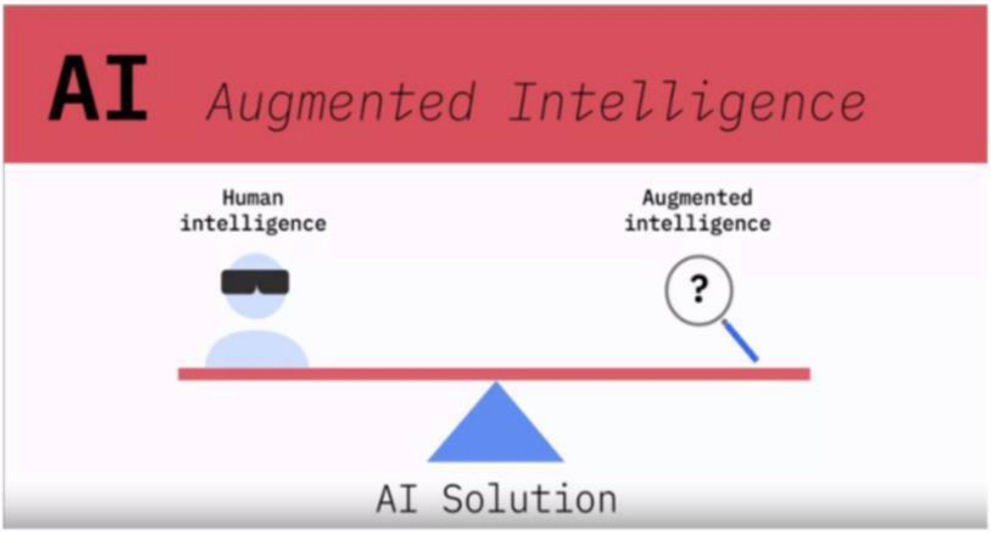
И технология машинного обучения – это использование математики в компьютерах для поиска неких шаблонов или закономерностей в данных.

И эти данные могут быть структурированными или неструктурированными.

Единственное различие между машинным обучением и технологиями, которые были до этого, состоит в том, что раньше нам, людям, приходилось вручную кодировать эти шаблоны.

Но компьютеры могут найти эти шаблоны или закономерности самостоятельно, используя математику.

Таким образом, для нас искусственный интеллект – это набор математических алгоритмов, которые позволяют компьютерам находить шаблоны, о которых мы, возможно, даже не подозреваем, без необходимости кодировать их вручную.



Можно сказать, что ИИ – это все, что заставляет машины действовать более разумно.

Также можно сказать, что ИИ – это дополненный интеллект.

ИИ не обязательно должен пытаться заменить людей, а скорее должен расширять человеческие возможности и выполнять задачи, которые ни люди, ни просто машины не могут выполнить самостоятельно.

Интернет дал нам быстрый доступ к большему количеству информации.

И сейчас у нас есть распределенные вычисления и интернет вещей, которые также связаны с огромным объемом информации.

И также есть социальные сети, которые генерируют огромный объем неструктурированных данных.

И с помощью ИИ, мы можем взять информацию, которая необходима экспертам, в определенной предметной области, и подкрепить ее доказательствами, чтобы эксперты могли принимать более обоснованные решения.

Вот что такое дополненный интеллект.

С ИИ эксперты могут расширить свои возможности и позволить машинам выполнять трудоемкую работу.

Теперь, как ИИ может учится?

Мы предоставляем машинам возможность исследовать примеры и создавать модели машинного обучения на основе входных данных и желаемых результатов.

И мы можем делать это различными способами, такими как контролируемое обучение, неконтролируемое обучение и дополнительное обучение.

На самом деле у людей есть очень распространенное заблуждение о том, что искусственный интеллект – это человеческий разум внутри компьютера, что человеческий интеллект полностью имитируется в компьютере.

Но машинное обучение или ИИ не моделирует человеческий разум, он просто пытается открыть новые возможности для компьютеров.

ИИ пытается позволить компьютерам понимать определенные виды данных, которые они не могли понять раньше.

Так, например, если посмотреть на то, что могут люди, то мы можем понимать язык, мы обладаем такой сложной способностью общаться на языке.

Также наш мозг может воспринимать колебания молекул воздуха и превращать их в мысли, и это действительно удивительно.

Мы также отлично умеем обрабатывать визуальные данные, например, когда вы смотрите на чье-то лицо, вы можете мгновенно узнать его.

Когда вы смотрите на чьи-то глаза, вы можете точно сказать, куда они смотрят, и это действительно удивительная способность.

Это то, что компьютеры не могут сделать, потому что они фундаментально ограничены математикой.

Они могут понимать только числа и математические операции.

Но, используя технологию машинного обучения, вы можете взять эту математику и использовать ее для изучения шаблонов или закономерностей в огромном объеме как структурированных, так и неструктурированных данных.

Вот что такое сейчас практически для нас ИИ.

Общее использование ИИ в настоящее время заключается в получении больших наборов данных и обработки этих данных в режиме реального времени.

В зависимости от типа решаемых задач, ИИ также можно разделить на типы – узкий, общий и супер.

Узкий ИИ – это ИИ, который применяется к определенной области.

Например, языковые репетиторы, виртуальные помощники, автомобили с автопилотом, поиск в Интернете с помощью ИИ, и так далее.

Узкий ИИ – это тип интеллекта, который хорош только для определенной задачи, хотя и узкие ИИ могут выполнять задачи, на выполнение которых уходит обычному человеку годы.

Узкий ИИ может выполнять конкретные задачи, но не изучать новые, принимая решения на основе запрограммированных алгоритмов и обучающих данных.

И узкий ИИ – это то, что мы уже научились создавать.

Общий ИИ – это ИИ, который может работать с широким спектром независимых и не связанных задач.

Он может изучать новые задачи для решения новых проблем, и он делает это, обучая себя новым стратегиям.

Общий интеллект – это сочетание многих стратегий ИИ, которые учатся на опыте и могут действовать на человеческом уровне интеллекта.

Общий ИИ является типом ИИ, наиболее близким человеческому интеллекту. И общий ИИ может выполнять различные задачи, иногда одновременно, также как и мы.

И сегодня мы приближаемся к созданию общего ИИ.

Хотя компьютеры в миллионы раз лучше нас в анализе и обработке данных, им никогда не удавалось мыслить абстрактно или придумывать оригинальные идеи.

Супер-ИИ или сознательный ИИ – это ИИ с человеческим сознанием на уровне самосознания.

И так как мы сами еще не можем адекватно определить, что такое сознание, маловероятно, что мы сможем создать сознательный ИИ в ближайшем будущем.

ИИ – это слияние многих областей исследования.

Информатика и электротехника определяют, как ИИ реализуется в программном и аппаратном обеспечении.

А математика и статистика определяют модели для него.

Создание ИИ основывается на нашем понимании как работает мозг.

И психология, и лингвистика играют важную роль в понимании того, как ИИ может работать, а философия дает рекомендации по интеллекту и этическим соображениям.

И сегодня мы уже видим примеры того, как ИИ участвует в решениях, которые принимают люди.

И ИИ уже доказал свою полезность в различных областях, оказывающих существенное влияние на жизнь людей и нашего общества.

На протяжении веков электричество считалось делом колдунов – магов, которые оставляли зрителей в недоумении о том, откуда электричество берется.

И когда Бенджамин Франклин доказал связь между электричеством и молнией, он с трудом мог представить его практическое использование в 1752 году.

На самом деле, он считал своим самым ценным изобретением – как избежать электричество – громоотвод.

И все новые инновации проходят аналогичную эволюцию: непонимание, избегание, страх и, наконец принятие.

И в течение многих лет после экспериментов Франклина человек постоянно использовал электричество, хотя ему все еще не хватало глубокого понимания происхождения электричества.

И Справочник по электротехнике 1928 года начинается со строк: «Что такое электричество? – Никто не знает».

Но согласно этому руководству для начинающих электриков, понимание происхождения электричества не было важным.

Более важным аспектом было знание того, как электричество можно генерировать и безопасно использовать для света, тепла и энергии.

Сегодня слишком много людей рассматривают искусственный интеллект как еще одну магическую технологию, которую можно использовать для работы, мало понимая, как она работает.

Они считают ИИ чем-то особенным и относятся к экспертам, которые освоили эту технологию, как к магам.

И ИИ приобрел мистический вид с обещаниями величия и вне досягаемости простых смертных.

Но правда, конечно, в том, что в ИИ нет магии.

Термин «искусственный интеллект» впервые появился в 1956 году, и с тех пор технология прогрессировала, разочаровывала и вновь появлялась.

Как и в случае с электричеством, путь к прорывам в ИИ придет с массовыми экспериментами.

Хотя многие из этих экспериментов потерпят неудачу, успешные будут иметь существенное влияние на развитие технологии.

Можно сказать, что ИИ – это новое электричество.

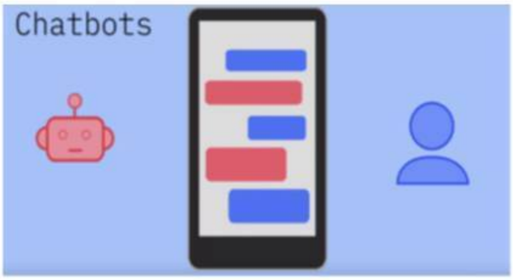
Помимо того, что ИИ становится повсеместным и все более доступным, ИИ расширяет и меняет методы ведения бизнеса по всему миру.

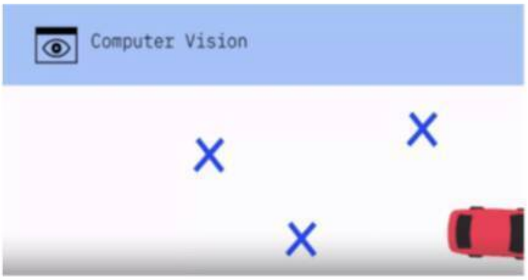
Он позволяет с высокой точностью прогнозировать и автоматизировать бизнес-процессы и принятие решений.

И его воздействие огромно, начиная от повышения качества обслуживания клиентов и заканчивая интеллектуальными продуктами и более эффективными услугами.

И в итоге результатом будет значительное экономическое влияние на компании, страны и общество.

И безусловно, организации, которые проводят массовые эксперименты в области искусственного интеллекта, получат в дальнейшем конкурентные преимущества.





Согласно исследованиям, в период с настоящего времени до 2030 года на основе ИИ прирост валового продукта составит 16 триллионов долларов США.

Это невиданный ранее масштаб экономического воздействия, и он касается не только ИТ-индустрии, он затрагивает практически на все отрасли и аспекты нашей жизни.

Возможности искусственного интеллекта по языковой обработке не только позволяют машинам и людям понимать и взаимодействовать друг с другом, но и создают новые возможности и новые способы ведения бизнеса.

Чат-боты, использующие возможности языковой обработки, используются в здравоохранении для опроса пациентов и выполнения базовых диагнозов, как настоящие врачи.

В сфере образования они предоставляют студентам онлайн-репетиторов.

Чат-боты обслуживания клиентов улучшают качество обслуживания клиентов, решая вопросы и освобождая время персонала.

Достижения искусственного интеллекта в технологии преобразования речи в текст являются причиной того, что компании используют голос на основе ИИ для повышения качества обслуживания клиентов и придают своему бренду уникальный голос.

Именно благодаря достижениям в области ИИ область компьютерного зрения смогла превзойти людей в задачах, связанных с обнаружением и маркировкой объектов.

Компьютерное зрение является одной из причин, по которой автомобили сами могут двигаться по улицам и избегать столкновений с препятствиями.

Алгоритм компьютерного зрения позволяет обнаружение черт лица и изображений и сравнение их с базами данных.

Это то, что позволяет устройствам аутентифицировать личности своих владельцев с помощью распознавания лиц, позволяет приложениям социальных сетей обнаружение и маркировку пользователей, а также правоохранительным органам выявлять преступников в видеопотоках.

ИИ влияет на качество нашей жизни ежедневно.

ИИ не допускает поступление спама в наши почтовые ящики и напоминает нам о важных событиях.

ИИ работает за сценой.

Он отслеживает наши инвестиции, выявляет мошеннические операции с кредитными картами и предотвращает финансовые преступления.

ИИ дает изменяющие жизнь рекомендации относительно здоровья и финансов, сопоставляет данные, которые могут нарушить конфиденциальность, и многое другое.

Идея искусственного интеллекта не нова и восходит к 1950-м годам.

Но ограничения вычислительной мощности препятствовали развитию этой технологии.

В 90-е годы, получив новый импульс, с развитием вычислительных мощностей, многие технологии, такие как нейронные сети, стали частью ИИ.

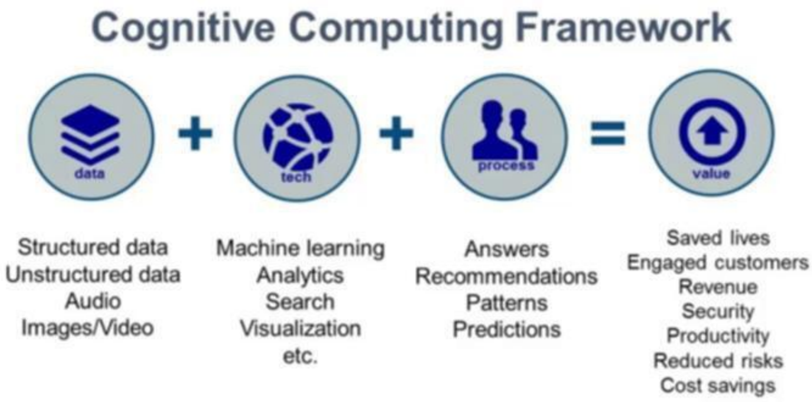
11 мая 1997 года мир был покорен, когда IBM Deep Blue победила Гарри Каспарова, действующего чемпиона мира по шахматам.

Мир исследований ИИ взорвался.

К концу 2000-х годов ИИ добился значительного прогресса в развитии технологий, таких как большие данные, Интернет и облачные вычисления, которые превратились в современные приложения искусственного интеллекта во множестве областей.

И компания IBM стала инициатором такой области ИИ как когнитивные вычисления.

Многие считают, что когнитивные вычисления представляют собой новую эру вычислений.



Когнитивные вычисления – это подобласть ИИ.

Когнитивные вычисления используют те же технологии, что и ИИ, такие как машинное обучение, нейронные сети, обработка языка, и так далее, но, чтобы имитировать процессы решения проблем, которые люди выполняют изо дня в день.

Когнитивные вычисления основаны на самообучающихся системах, которые используют методы машинного обучения для интеллектуального выполнения специфических задач, похожих на человеческие.

Узкий искусственный интеллект не пытается имитировать мыслительные процессы человека.

Вместо этого, узкая система ИИ – это сложные алгоритмы для решения конкретной задачи – в случае беспилотной машины, она просто избегает столкновений, держа курс.

И узкий ИИ не пытается обрабатывать те же данные так же, как человеческий мозг.

Но, с другой стороны, когнитивные вычисления не принимают решения за людей, а скорее дополняют наши собственные решения.

Хотя цель когнитивных вычислений – это понять и воспроизвести суть человеческого интеллекта.

Система когнитивных вычислений имеет способность адаптироваться (как мозг) к любому окружению.

Она является динамичной в сборе данных и понимании целей и требований.

И когнитивная система обладает возможностью легко взаимодействовать с пользователями, чтобы пользователи могли легко определять свои потребности.

Аналогично, она также взаимодействует с другими устройствами и облачными сервисами.

Система когнитивных вычислений обладает способностью понимать, идентифицировать и извлекать контекст, такой как синтаксис, время, местоположение, правила, профили, процессы, задачи и цели.

Она опирается на несколько источников информации, включая как структурированную, так и неструктурированную цифровую информацию.

И мы можем найти множество примеров успешных систем когнитивных вычислений.

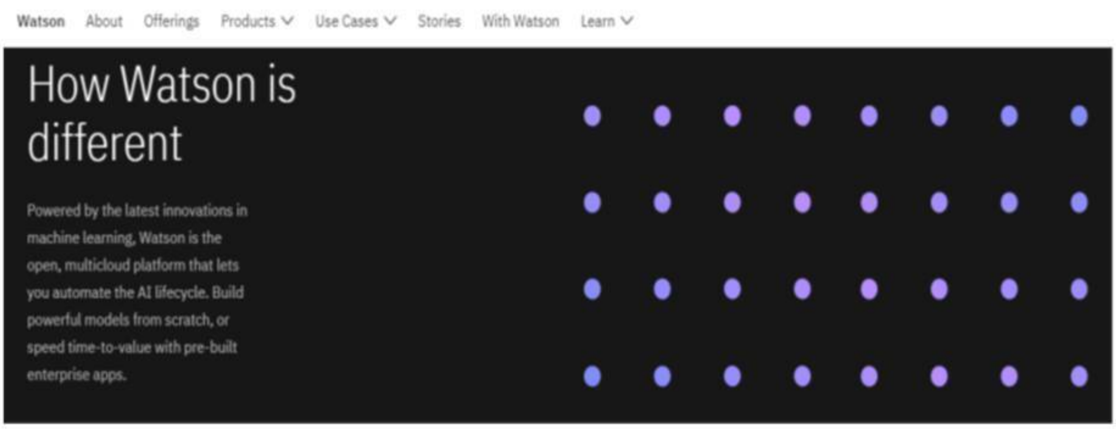
Например, точность технологии распознавания голоса Google выросла с 84 процентов в 2012 году до 98 процентов менее чем за два года.

Технология DeepFace Facebook теперь может распознавать лица с точностью до 97 процентов.

В настоящее время в сфере когнитивных вычислений доминируют такие крупные игроки, как IBM, Microsoft и Google.

IBM, являясь пионером этой технологии, инвестировала 26 миллиардов долларов в большие данные и аналитику и сейчас тратит около трети своего бюджета на исследования и разработки в области когнитивных вычислений.

https://www.ibm.com/watson

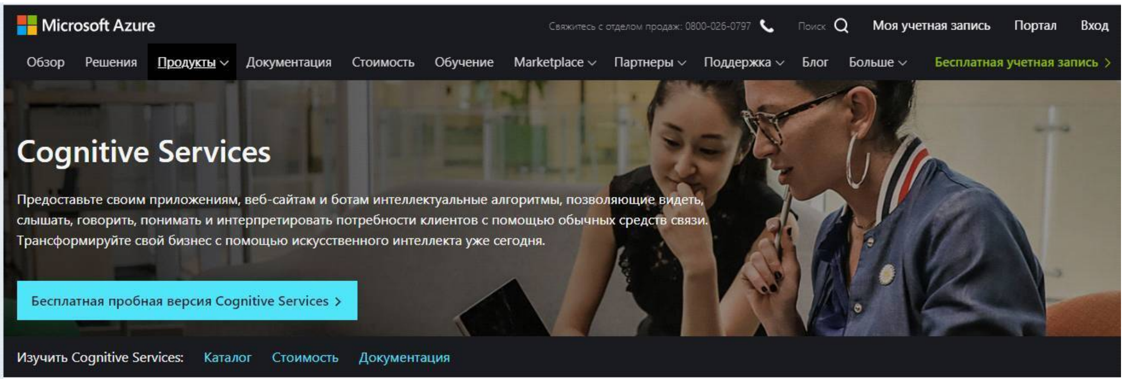


IBM Watson – это суперкомпьютер и платформа когнитивных вычислений IBM.

Основная задача Уотсона – понимать вопросы, сформулированные на естественном языке, и находить на них ответы с помощью ИИ.

IBM Watson использует глубокий анализ контента и обоснование на основе фактических данных.

В сочетании с вероятностными методами обработки, Watson может улучшить процесс принятия решений, сократить расходы и оптимизировать результаты.

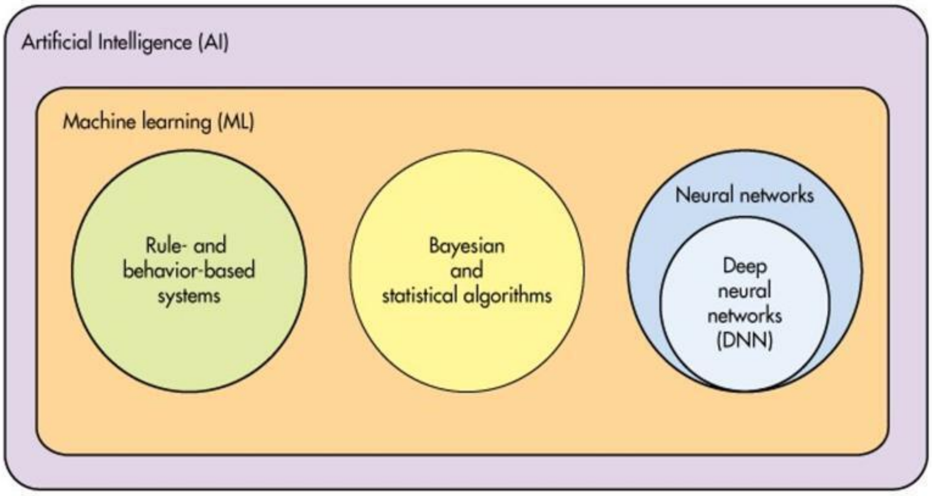


Microsoft Cognitive Services – это набор API, SDK и когнитивных сервисов, которые разработчики могут использовать для повышения интеллектуальности своих приложений.

С помощью таких сервисов разработчики могут легко добавлять интеллектуальные функции в свои приложения – такие как обнаружение эмоций и чувств, распознавание изображений и речи, знание, поиск и понимание языка.

Машинное обучение, глубокое обучение, нейронные сети

Прежде чем мы углубимся в то, как работает ИИ, и его различные варианты использования и приложения, давайте еще раз вернемся к терминам и концепциям ИИ, и разберем понятия искусственного интеллекта, машинного обучения, глубокого обучения и нейронных сетей.



Эти термины иногда используются взаимозаменяемо, но они не относятся к одному и тому же.

Искусственный интеллект – это область информатики, занимающаяся симуляцией интеллектуального поведения.

Системы ИИ, как правило, демонстрируют поведение, связанное с человеческим интеллектом, такое как планирование, обучение, рассуждение, решение задач, представление знаний, восприятие, движение и манипуляция, и в меньшей степени социальный интеллект и креативность.

Машинное обучение – это подмножество ИИ, которое использует компьютерные алгоритмы для анализа данных и принятия разумных решений на основе того, что они узнали, без явного программирования.

Алгоритмы машинного обучения обучаются на больших наборах данных и учатся на примерах.

Они не следуют алгоритмам, основанным на правилах.

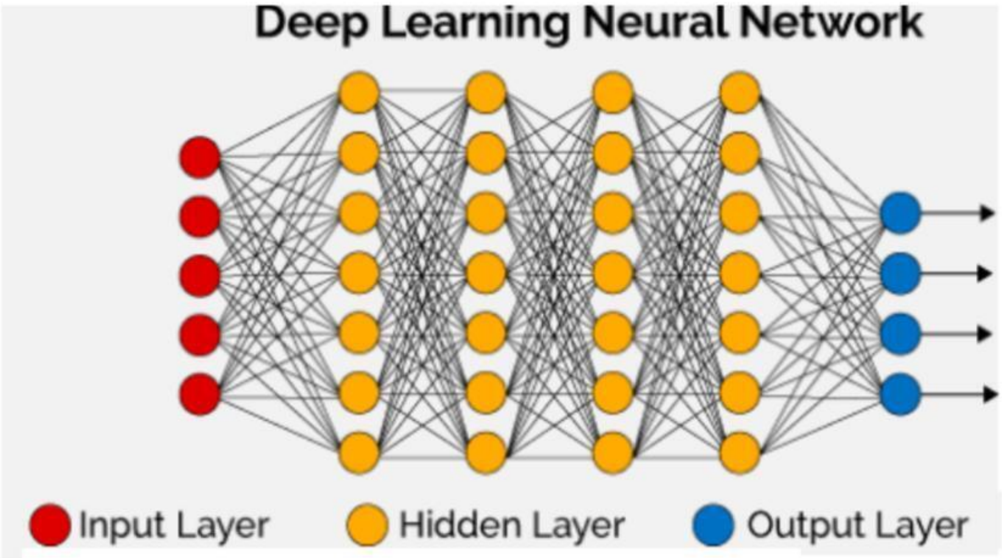
Машинное обучение – это то, что позволяет машинам самостоятельно решать задачи и делать точные прогнозы, используя предоставленные данные.

Глубокое обучение – это специализированный раздел машинного обучения, который использует многоуровневые нейронные сети для имитации принятия человеческих решений.

Алгоритмы глубокого обучения могут маркировать и классифицировать информацию и идентифицировать шаблоны – закономерности.

Это то, что позволяет системам искусственного интеллекта постоянно учиться в процессе работы и повышать качество и точность результатов, определяя правильность принятых решений.

Идея искусственных нейронных сетей основывается на биологических нейронных сетях, хотя они работают совсем по-другому.



Нейронная сеть в ИИ представляет собой набор небольших вычислительных блоков, называемых нейронами, которые принимают входящие данные и учатся принимать решения с течением времени.

Нейронные сети часто являются многоуровневыми и становятся более эффективными по мере увеличения объема наборов данных, в отличие от других алгоритмов машинного обучения.

Теперь, давайте разберем еще одно важное различие, которое важно понять, – это различие между искусственным интеллектом и наукой о данных.

Наука о данных – это процесс и метод извлечения знаний и идей из больших объемов разнородных данных.

Это междисциплинарная область, включающая математику, статистический анализ, визуализацию данных, машинное обучение и многое другое.

Это то, что позволяет нам обрабатывать информацию, видеть закономерности, находить смысл в больших объемах данных и использовать информацию для принятия решений.

И наука о данных, Data Science может использовать многие методы искусственного интеллекта, чтобы получить представление о данных.

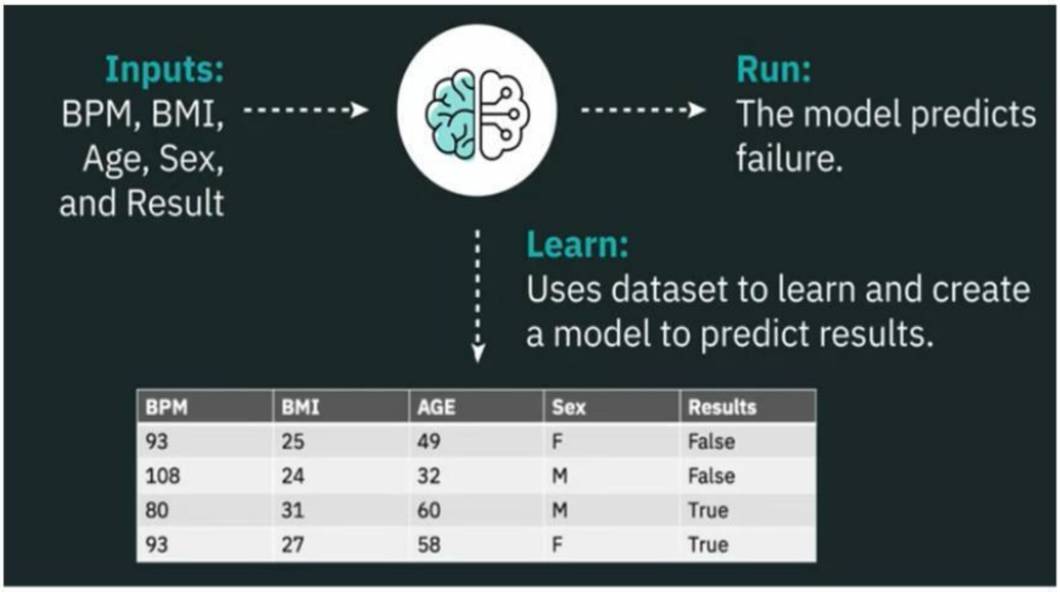
Например, наука о данных может использовать алгоритмы машинного обучения и даже модели глубокого обучения, чтобы извлечь смысл и сделать выводы из данных.

Существует некоторое пересечение между ИИ и наукой о данных, но одно не является подмножеством другого.

Наоборот, наука о данных – это более широкий термин, охватывающий всю методологию обработки данных.

А ИИ включает в себя все, что позволяет компьютерам учиться решать задачи и принимать разумные решения.

И ИИ, и Data Science могут использовать большие данные.



BPM	BMI	AGE	Sex	Results
93	25	49	F	False
108	24	32	M	False
80	31	60	M	True
93	27	58	F	True

Машинное обучение, подмножество искусственного интеллекта, использует компьютерные алгоритмы для анализа данных и принятия разумных решений на основе того, что алгоритмы изучили.

Вместо того, чтобы следовать алгоритмам, основанным на правилах, машинное обучение само строит модели для классификации и прогнозирования на основе данных.

Например, что, если мы хотим определить, может ли возникнуть проблема с нашим сердцем, с помощью машинного обучения?

Можем ли мы это сделать.

И ответ – да.

Допустим, нам даны такие данные, как количество ударов в минуту, вес тела, возраст и пол.

С машинным обучением и этим набором данных, мы можем изучить и создать модель, которая с учетом входных данных будет предсказывать результаты.

Так в чем же разница между этим подходом и просто использования статистического анализа для создания алгоритма?

Алгоритм – это математическая техника.

В традиционном программировании мы берем данные и правила и используем их для разработки алгоритма, который даст нам ответ.

В этом примере, если бы мы использовали традиционный алгоритм, мы бы взяли данные, такие как сердечный ритм, возраст, вес тела и пол и использовали эти данные для создания алгоритма, который определит, будет ли сердце работать нормально или нет.

По сути, это было бы выражение if – else.

Когда мы отправляем входные данные, мы получаем ответы, основанные на том, какой алгоритм мы определили, и этот алгоритм не изменится от данных.

Машинное обучение, с другой стороны, берет данные и ответы и уже потом само создает алгоритм.

Вместо того, чтобы получить ответы в конце, у нас уже есть ответы.

А то, что мы получаем здесь, – это набор правил, определяющих модель машинного обучения.

И эта модель определяет правила и оператор if – else при получении входных данных.

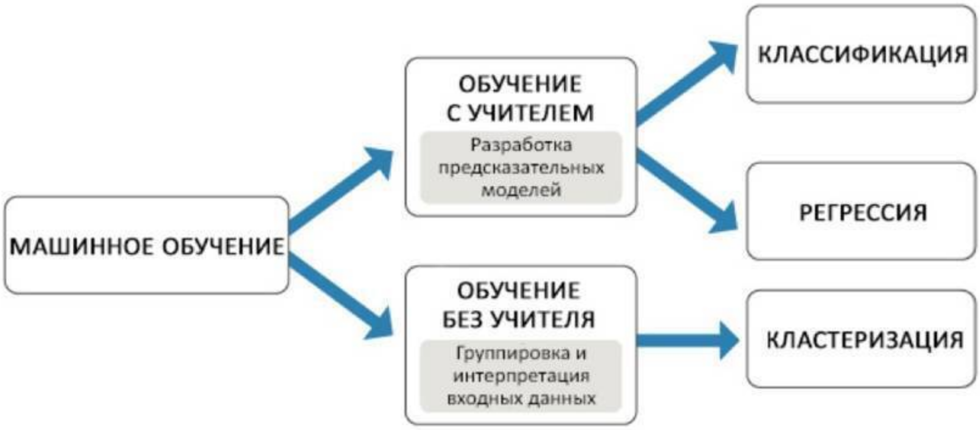
И эта модель, в отличие от традиционного алгоритма, может постоянно обучаться и использоваться в будущем для прогнозирования значений.

Машинное обучение опирается на определение правил путем изучения и сравнения больших наборов данных, чтобы найти общие закономерности.

Например, мы можем создать программу машинного обучения с большим объемом изображений птиц, и обучить модель возвращать название птицы всякий раз, когда мы даем изображение птицы.

Когда для модели отображается изображение птицы, она маркирует изображение с некоторой степенью достоверности.

Этот тип машинного обучения называется контролируемым обучением, где алгоритм обучается на данных, размеченных человеком.



Чем больше примеров мы предоставляем контролируемому алгоритму обучения, тем точнее он производит классификацию новых данных.

Неуправляемое обучение, это еще один тип машинного обучения, которое основывается на предоставлении алгоритму неразмеченных данных и позволяет ему самостоятельно находить шаблоны.

Вы предоставляете просто входные данные, и позволяете машине делать выводы и находить шаблоны.

Этот тип обучения может быть полезен для кластеризации данных, когда данные группируются в соответствии с тем, насколько они похожи на своих соседей и отличаются от всего остального.

Как только данные кластеризованы, можно использовать различные методы для изучения этих данных и поиска шаблонов.

Например, можно создать алгоритм машинного обучения с постоянным потоком сетевого трафика и позволить ему независимо изучать активность в сети – базовый уровень, нормальную сетевую активность, а также выбросы и, возможно, злонамеренное поведение, происходящее в сети.

Третий тип алгоритма машинного обучения, обучение с подкреплением, это алгоритм машинного обучения с набором правил и ограничений и позволяет ему учиться достигать целей.

Вы определяете состояние, желаемую цель, разрешенные действия и ограничения.

И алгоритм выясняет, как достичь цели, пробуя различные комбинации разрешенных действий, и его награждают или наказывают в зависимости от того, было ли решение правильным.

Алгоритм изо всех сил старается максимизировать свои вознаграждения в рамках предусмотренных ограничений.

И вы можете использовать обучение с подкреплением, чтобы научить машину играть в шахматы или преодолеть какие-либо препятствия.

Таким образом, машинное обучение – это широкая область, и мы можем разделить его на три разные категории: контролируемое обучение, неконтролируемое обучение и обучение с подкреплением.

И есть много разных задач, которые мы можем решить с помощью них.

В контролируемом обучении, в наборе данных есть метки, и мы используем их для построения модели классификации данных.

Это означает, что, когда мы получаем данные, у них есть метки, которые говорят о том, что представляют эти данные.

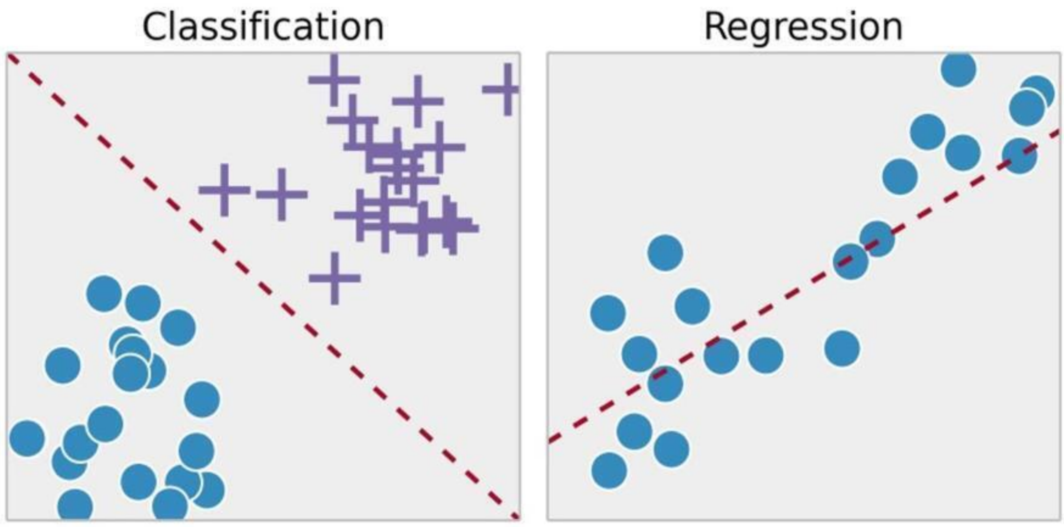
В примере с сердцем, у нас была таблица с метками, это сердечный ритм, возраст, пол и вес.

И каждой такой метке соответствовали значения.

При неконтролируемом обучении у нас нет меток, и мы должны обнаружить эти метки в неструктурированных данных.

И такие вещи обычно делаются с помощью кластеризации.

Обучение с подкреплением – это другое подмножество машинного обучения, и оно использует вознаграждение для наказания за плохие действия или вознаграждение за хорошие действия.



И мы можем разделить контролируемое обучение на три категории: регрессия, классификация и нейронные сети.

Модели регрессии строятся с учетом взаимосвязей между признаками x и результатом y, где y – непрерывная переменная.

По сути, регрессия оценивает непрерывные значения.

Нейронные сети относятся к структурам, которые имитируют структуру человеческого мозга.

Классификация, с другой стороны, фокусируется на дискретных значениях, которые она идентифицирует.

Мы можем назначить дискретные результаты y на основе многих входных признаков x.

В примере с сердцем, учитывая набор признаков x, таких как удары в минуту, вес тела, возраст и пол, алгоритм классифицирует выходные данные y как две категории: истина или ложь, предсказывая, будет ли сердце работать нормально или нет.

В других классификационных моделях мы можем классифицировать результаты по более чем двум категориям.

Например, прогнозирование, является ли данный рецепт рецептом индийского, китайского, японского или тайского блюда.

И с помощью классификации мы можем извлечь особенности из данных.

Особенности в этом примере сердцем, это сердечный ритм или возраст.

Особенности – это отличительные свойства шаблонов ввода, которые помогают определить категории вывода.

BPM	BMI	AGE	Sex	Results
93	25	49	F	False
108	24	32	M	False
80	31	60	M	True
93	27	58	F	True

Здесь каждый столбец является особенностью, а каждая строка – точкой ввода данных.

Классификация – это процесс прогнозирования категории заданных точек данных.

И наш классификатор использует обучающие данные, чтобы понять, как входные переменные относятся к этой категории.

Что именно мы подразумеваем под обучением?

Обучение подразумевает использование определенного алгоритма обучения для определения и разработки параметров модели.

Хотя для этого есть много разных алгоритмов, с точки зрения непрофессионала, если вы тренируете модель, чтобы предсказать, будет ли сердце работать нормально или нет, есть истинные или ложные значения, и вы будете показывать алгоритму некоторые реальные данные, помеченные как истинные, затем снова показывая данные, помеченные как ложные, и вы будете повторять этот процесс с данными, имеющими истинные или ложные значения.

И алгоритм будет изменять свои внутренние параметры до тех пор, пока он не научится распознавать данные, которые указывают на то, что есть сердечная недостаточность или ее нет.

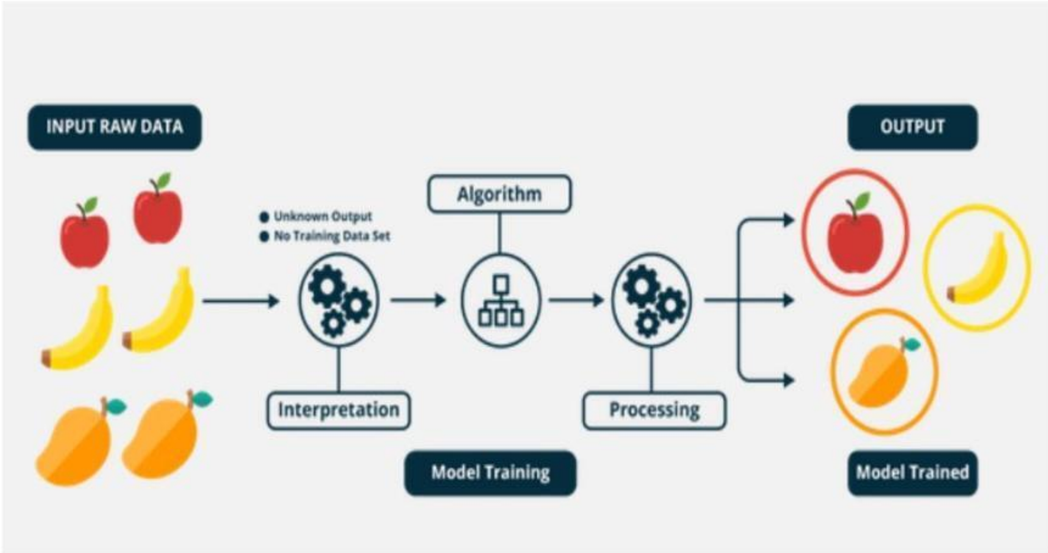
При машинном обучении мы обычно берем набор данных и делим его на три набора: наборы обучения, проверки и тестирования.

Набор обучения – это данные, используемые для обучения алгоритма.

Набор проверки используется для проверки наших результатов и тонкой настройки параметров алгоритмов.

Данные тестирования – это данные, которые модель никогда не видела прежде и которые используются для оценки того, насколько хороша наша модель.

Опять же, чтобы повторить, модель машинного обучения – это алгоритм, используемый для поиска закономерностей в данных без программирования в явном виде.



В то время как машинное обучение является подмножеством искусственного интеллекта, глубокое обучение является специализированным подмножеством машинного обучения.

Глубокое обучение основывается на алгоритмах машинного обучения, которые основываются на структуре и функциях мозга, и эти алгоритмы называются искусственными нейронными сетями.

Эти сети предназначены для непрерывного обучения в процессе работы для повышения качества и точности результатов.

Эти системы могут обучаться на неструктурированных данных, таких как фотографии, видео и аудиофайлы.

Алгоритмы глубокого обучения напрямую не отображают входные данные в выходные.

Вместо этого они полагаются на несколько слоев обработки.

Каждый такой слой передает свой вывод следующему слою, который обрабатывает его и передает его следующему.

Именно поэтому такая система из многочисленных слоев называется глубоким обучением.

При создании алгоритмов глубокого обучения разработчики и инженеры настраивают количество слоев и тип функций, которые соединяют выходы каждого слоя со входами следующего.

Затем они обучают модель, предоставляя множество размеченных примеров.

Например, вы даете алгоритму глубокого изучения тысячи изображений и метки, которые соответствуют содержанию каждого изображения.

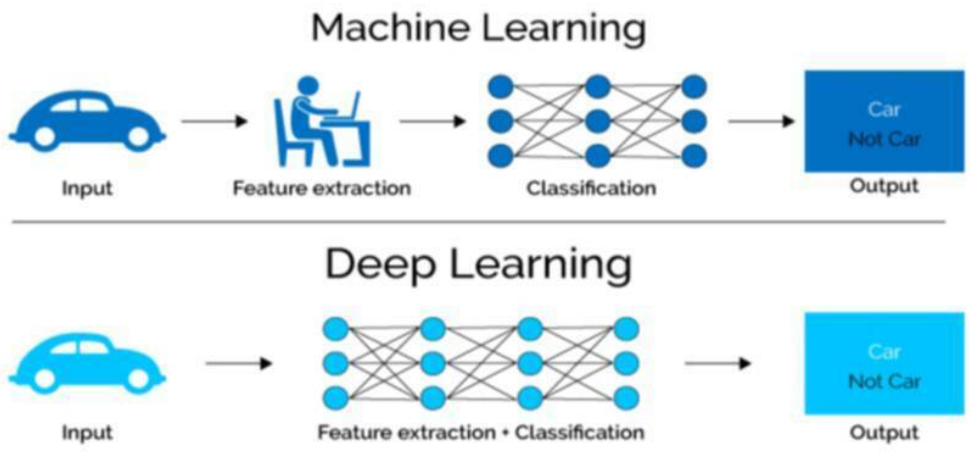
Алгоритм будет запускать эти примеры через свою многоуровневую нейронную сеть и будет подгонять веса переменных в каждом слое нейронной сети, чтобы иметь возможность обнаруживать общие шаблоны, которые определяют изображения с похожими метками.

Глубокое обучение устраняет одну из основных проблем, с которой сталкивались алгоритмы обучения предыдущего поколения.

В то время как эффективность и производительность алгоритмов машинного обучения предыдущего поколения не улучшалась по мере роста наборов данных, алгоритмы глубокого обучения продолжают улучшаться по мере поступления большего количества данных.

Глубокое обучение оказалось очень эффективным при выполнении различных задач, включая распознавание и транскрипцию голоса, распознавание лиц, медицинскую визуализацию и языковой перевод.

Глубокое обучение также является одним из основных компонентов беспилотных автомобилей.



Искусственная нейронная сеть представляет собой совокупность мелких единиц, называемых нейронами, которые представляют собой вычислительные единицы, смоделированные по способу обработки информации человеческим мозгом.

Искусственные нейронные сети заимствуют некоторые идеи из биологической нейронной сети мозга, чтобы приблизить некоторые результаты его обработки.

Эти единицы или нейроны принимают поступающие данные, также как и биологические нейронные сети, и со временем учатся принимать решения.

Нейронные сети учатся через процесс, называемый обратным распространением.

Например, при преобразовании речи в текст, в нейронных сетях вместо кодирования правил вы предоставляете образцы голоса и соответствующий им текст.

И нейронная сеть находит общие шаблоны произношения слов, а затем учится сопоставлять новые голосовые записи с соответствующими им текстами.

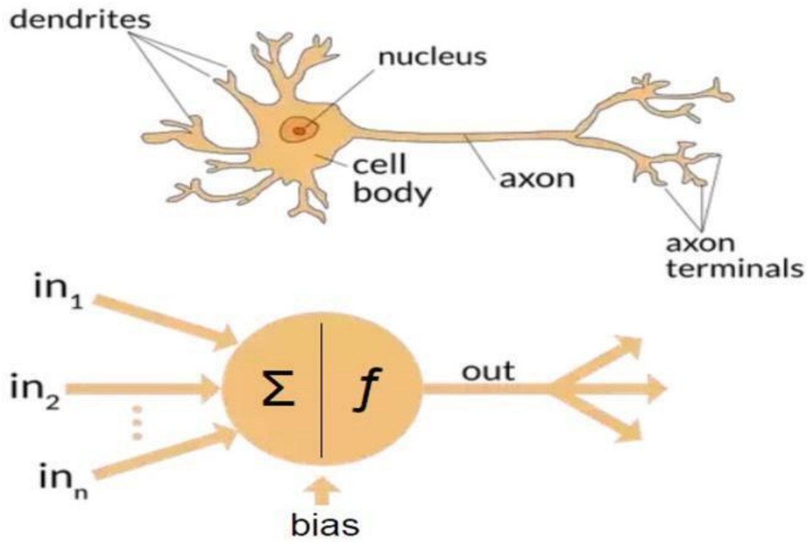
YouTube использует это для автоматического создания субтитров.

Обратное распространение использует набор обучающих данных, которые сопоставляют известные входы с желаемыми выходами.

Сначала входы подключаются к сети и определяются выходы.

Затем функция ошибки определяет, насколько далеко данный выход находится от желаемого выхода.

И наконец, делаются изменения, чтобы уменьшить ошибки.



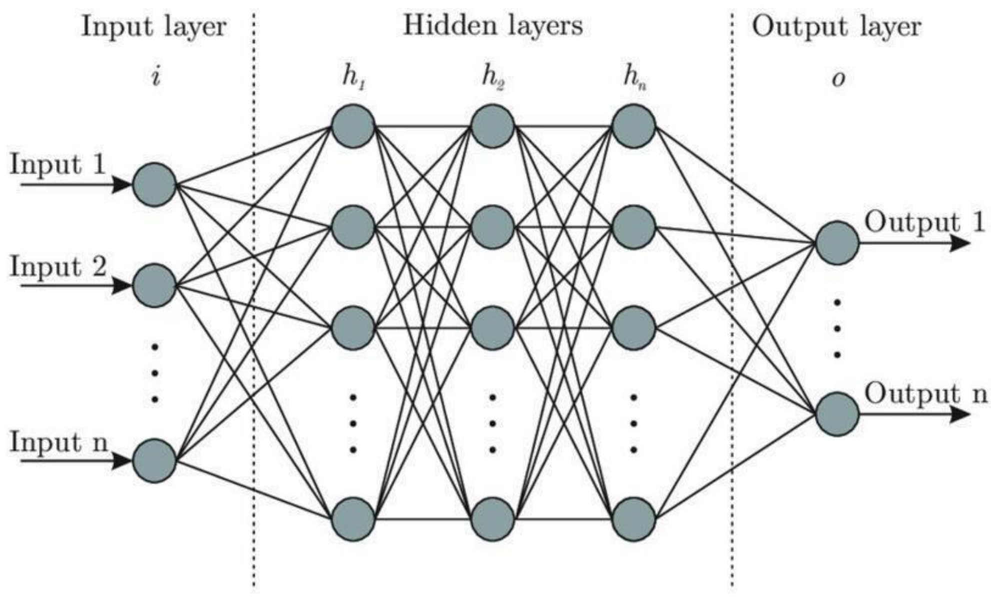
Набор нейронов называется слоем, и слой принимает входные данные и обеспечивает выходные данные.

Любая нейронная сеть будет иметь один входной слой и один выходной слой.

И нейронная сеть также будет иметь один или несколько скрытых слоев, которые имитируют типы деятельности, происходящих в человеческом мозге.

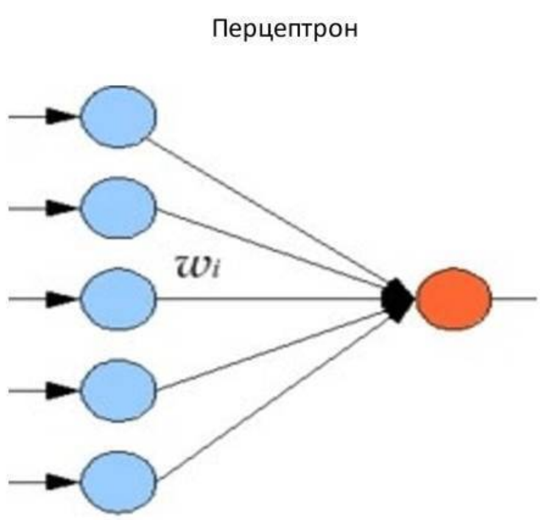
Скрытые слои принимают набор взвешенных входных данных и выдают результат с помощью функции активации.

Нейронная сеть, имеющая более одного скрытого слоя, называется глубокой нейронной сетью.



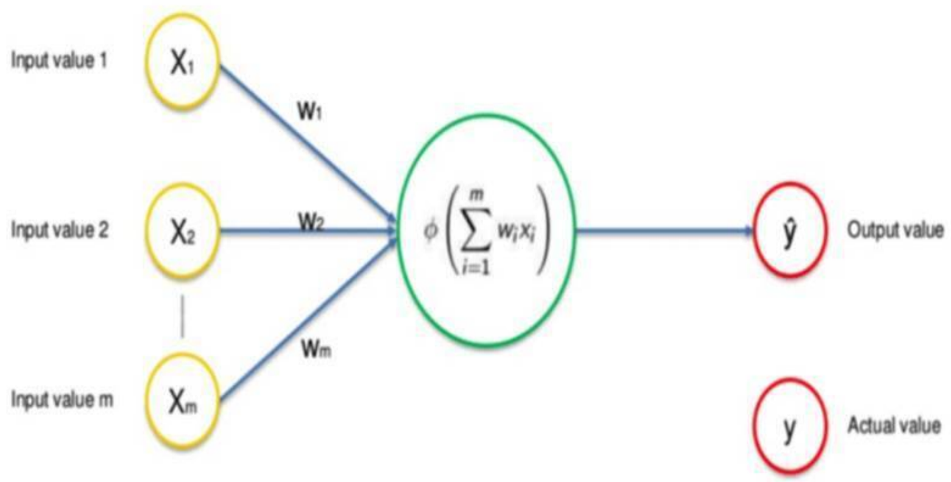
Перцептроны – это самые простые и старые типы нейронных сетей.

Это однослойные нейронные сети, состоящие из входных узлов, подключенных непосредственно к выходному узлу.

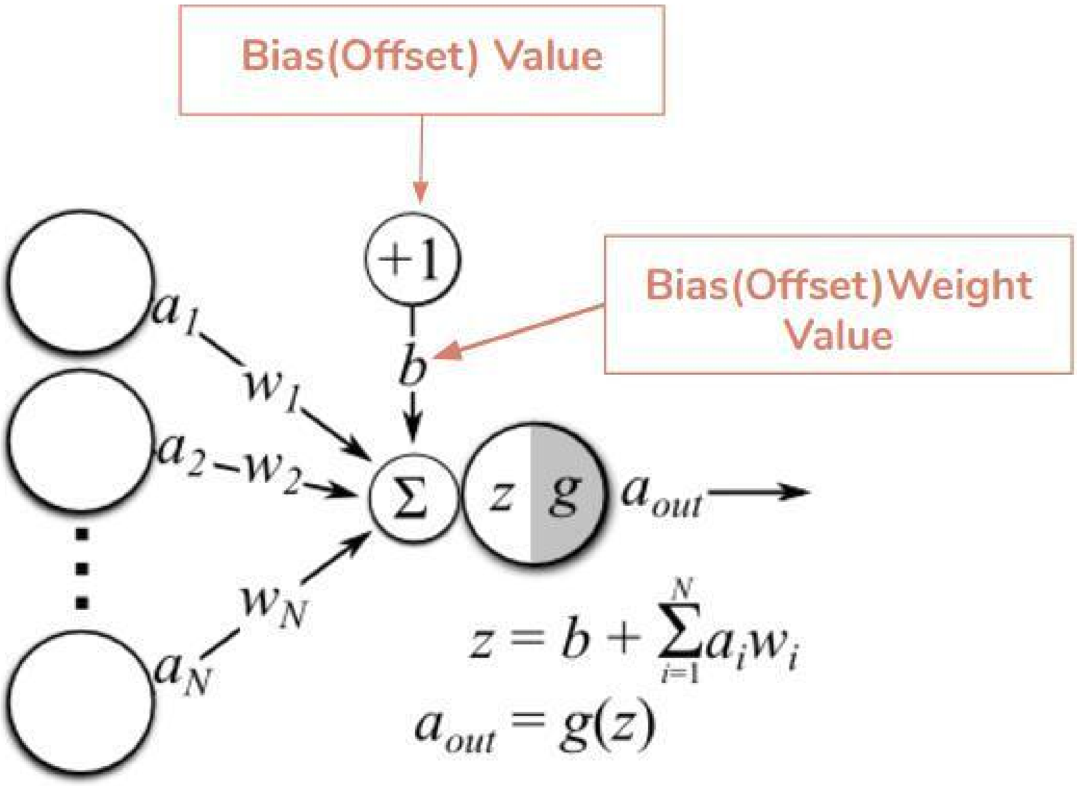


Входные слои передают входные значения следующему слою путем умножения на вес и суммирования результатов.

Скрытые слои получают входные данные от других узлов и направляют свои выходные данные на другие узлы.



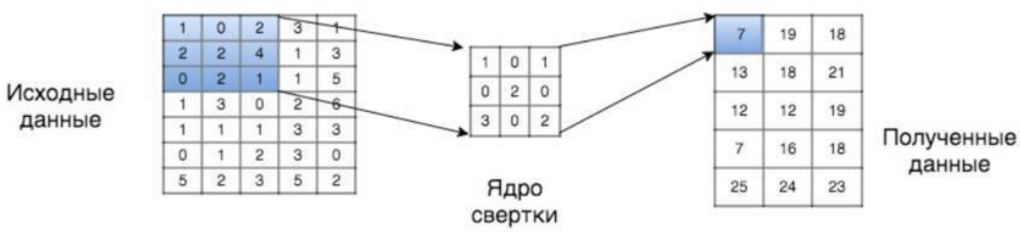
Скрытые и выходные узлы имеют свойство, называемое смещением bias, которое представляет собой особый тип веса, который применяется к узлу после рассмотрения других входных данных.



И наконец, функция активации определяет, как узел реагирует на свои входные данные.

Функция запускается на сумме входов и смещения, а затем результат передается как выходной.

Функции активации могут принимать различные формы, и их выбор является критическим компонентом успеха нейронной сети.



Сверточные нейронные сети или CNN представляют собой многослойные нейронные сети, которые основываются на работе зрительной коры животных.

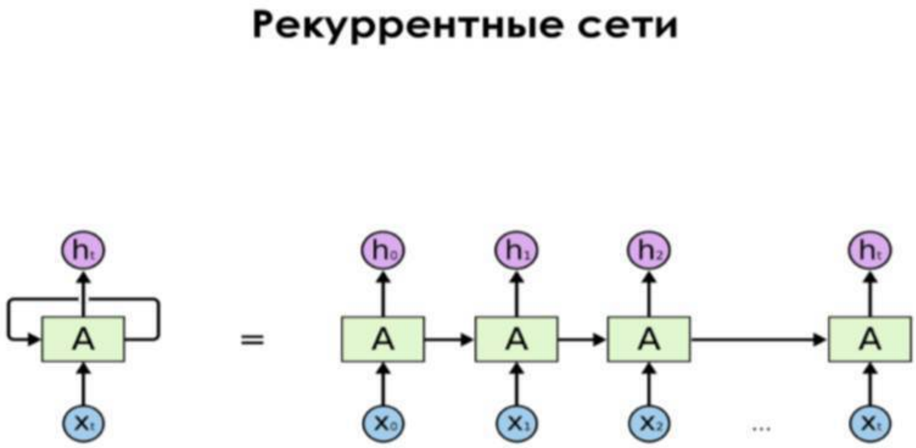
CNN полезны в таких приложениях, как обработка изображений, распознавание видео и обработка языка.

Свертка – это математическая операция, в которой функция применяется к другой функции, а результат представляет собой смесь двух функций.

Свертки хороши при обнаружении простых структур на изображении и объединении этих простых функций для создания более сложных функций.

В сверточной сети этот процесс происходит в последовательности слоев, каждый из которых проводит свертку на выходе предыдущего слоя.

CNN являются экспертами в построении сложных функций из менее сложных.



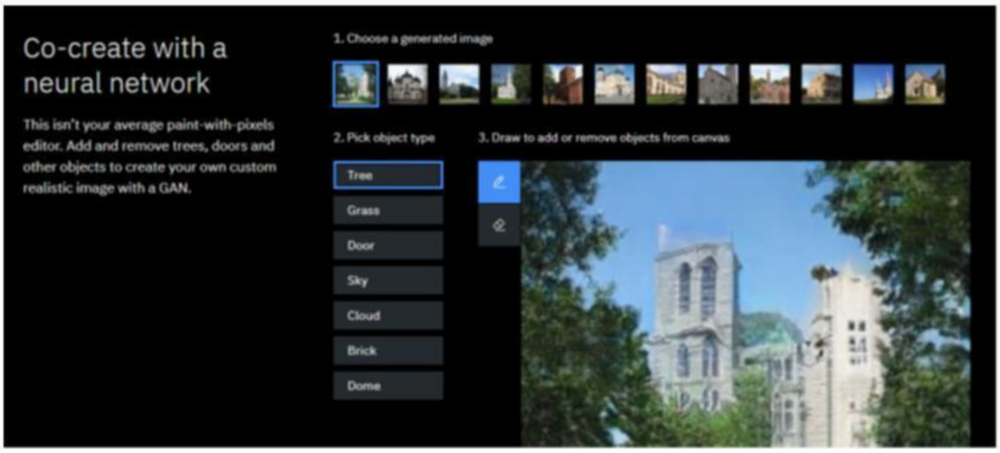
Рекуррентные нейронные сети или RNN являются рекуррентными, потому что они выполняют одну и ту же задачу для каждого элемента последовательности, причем предыдущие выходы питают входы последующих этапов.

В обычной нейронной сети вход обрабатывается через несколько слоев, а выход создается с допущением, что два последовательных входа независимы друг от друга, но это может не выполняться в определенных сценариях.

Например, когда нам нужно учитывать контекст, в котором было произнесено слово, в таких сценариях необходимо учитывать зависимость от предыдущих наблюдений, чтобы получить результат.

И RNN могут использовать информацию в длинных последовательностях, причем каждый уровень сети представляет наблюдение в определенное время.

https://gan-paint-demo.mybluemix.net



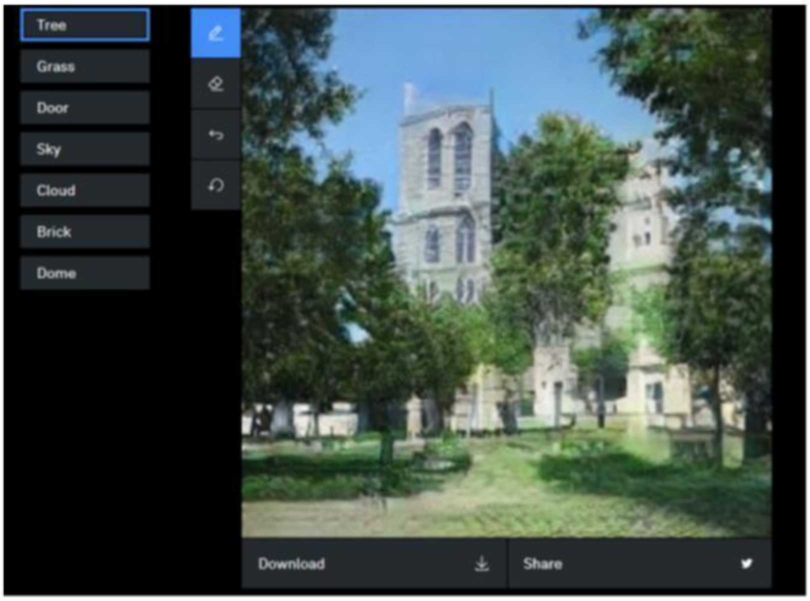
Новый тип нейронной сети, называемый порождающей состязательной сетью (GAN), может использоваться для создания сложных выходных данных, таких как фотореалистичные изображения.

На странице сайта IBM вы можете попробовать создать изображение с помощью GAN.

В разделе «Совместное создание с нейронной сетью» в разделе «Выберите сгенерированное изображение» выберите одно из существующих изображений.

И в списке Pick object type выберите тип объекта, который вы хотите добавить.

Например, нажмите на дерево.



Переместите курсор на изображение.

Нажмите и удерживая кнопку мыши нажатой, наведите курсор на область существующего изображения, в которую вы хотите добавить объект, в данном случае дерево.

Выберите другой тип объекта и добавьте его к изображению.

Поэкспериментируйте: можете ли вы поместить дверь в небо?
И используйте функции отмены и удаления, чтобы удалить объекты.
И нажмите «Загрузить», чтобы сохранить свою работу.

Наука о данных

Наука о данных – это процесс использования данных, чтобы понять различные вещи, понять мир.

Это когда у вас есть модель или гипотеза проблемы, и вы пытаетесь проверить эту гипотезу или модель на данных.

Наука о данных – это искусство раскрытия идей и тенденций, которые скрываются за данными.

Данные реальны, данные имеют реальные свойства, и нам нужно изучить их, если мы собираемся работать с ними.

Это название появилось в 90-х годах, когда некоторые профессора вели учебную программу по статистике, и они подумали, что было бы лучше назвать это наукой о данных.

Но что такое наука о данных?

Если у вас есть данные, и вы работаете с данными, и вы манипулируете ими, вы исследуете их, сам процесс анализа данных, в попытках получить ответы на какие-то вопросы, – это наука о данных.

И наука о данных актуальна именно сегодня, потому что у нас есть огромный объем доступных данных.

Раньше стоял вопрос о нехватке данных.

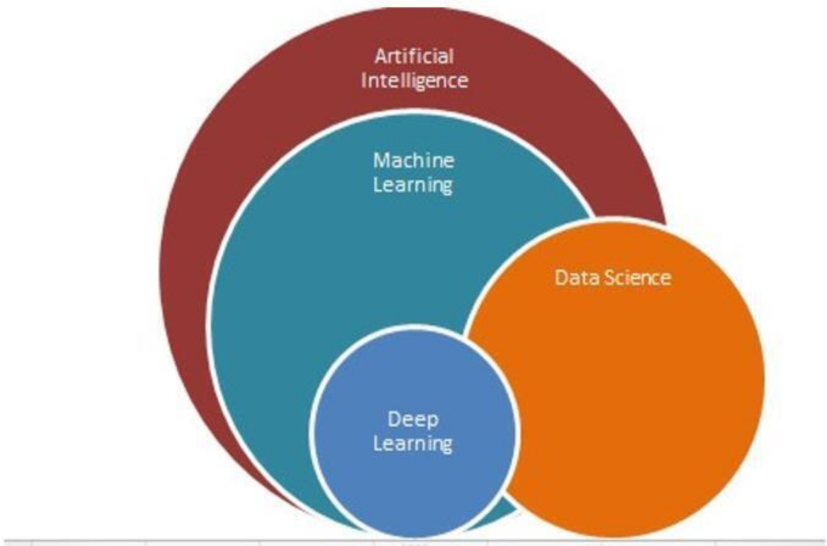
Теперь у нас есть непрерывные потоки данных.

В прошлом у нас не было алгоритмов работы с данными, теперь у нас есть алгоритмы.

Раньше программное обеспечение было дорогим, теперь оно с открытым исходным кодом и бесплатное.

Раньше мы не могли хранить большие объемы данных, теперь за небольшую плату мы можем иметь доступ к большим наборам данных.

Теперь, как соотносятся между собой ИИ, машинное обучение и наука о данных.



Искусственный интеллект – это очень широкий термин для различных применений: от робототехники до анализа текста.

Это все еще развивающаяся технология, и есть вопросы о том, должны ли мы на самом деле стремиться к высокоуровневому ИИ или нет.

Машинное обучение – это подмножество искусственного интеллекта, которое фокусируется на узком диапазоне видов деятельности.

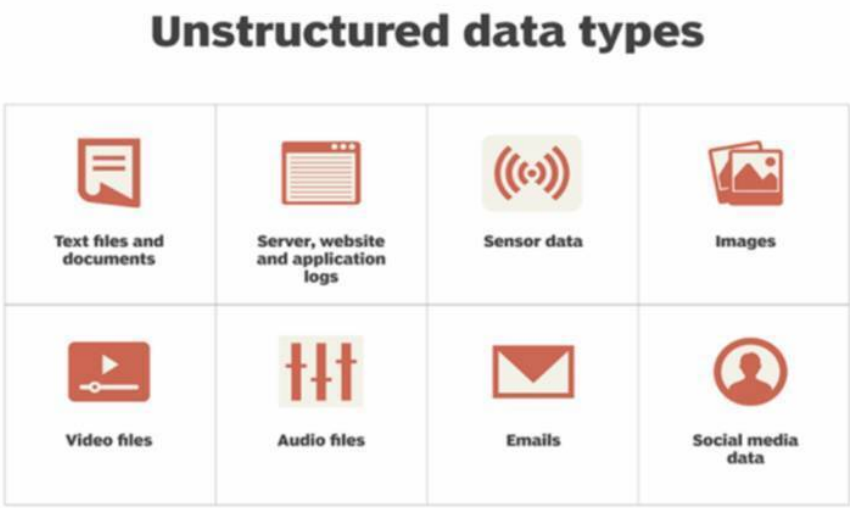
Фактически это единственный вид искусственного интеллекта, который сейчас существует с некоторыми приложениями в реальных задачах.

Наука о данных не является подмножеством машинного обучения, но использует машинное обучение для анализа данных и прогнозирования будущего.

Наука о данных сочетает в себе машинное обучение с другими дисциплинами, такими как анализ больших данных и облачные вычисления.

Наука о данных – это практическое применение машинного обучения с фокусом на решении реальных задач.

Наука о данных в основном сосредоточена на работе с неструктурированными данными.



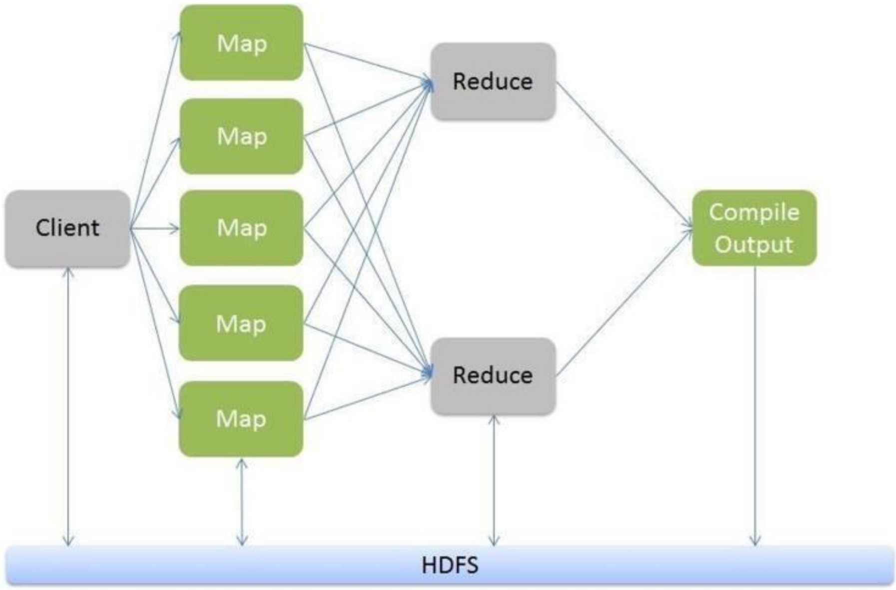
Структурированные данные больше похожи на табличные данные, с которыми мы имеем дело в Microsoft Excel, где у вас есть строки и столбцы, и это называется структурированными данными.

Неструктурированные данные – это данные, поступающие в основном из Интернета, где они не являются табличными, они не в виде строк и столбцов, а в виде текста, иногда это видео и аудио, поэтому вам придется использовать более сложные алгоритмы для обработки этих данных.

Традиционно при вычислении и обработке данных мы переносим данные на компьютер.

Но если данных очень много, они просто могут не поместиться на одном компьютере.

Поэтому Google придумал очень просто: они взяли данные и разбили их на куски, и они отправили эти куски файлов на тысячи компьютеров, сначала это были сотни, а потом тысячи, и теперь десятки тысяч компьютеров.



И они поставили одну и ту же программу на все эти компьютеры в кластере.

И каждый компьютер запускает эту программу на своем маленьком фрагменте файла и отправляет результаты обратно.

Затем результаты сортируются и объединяются.

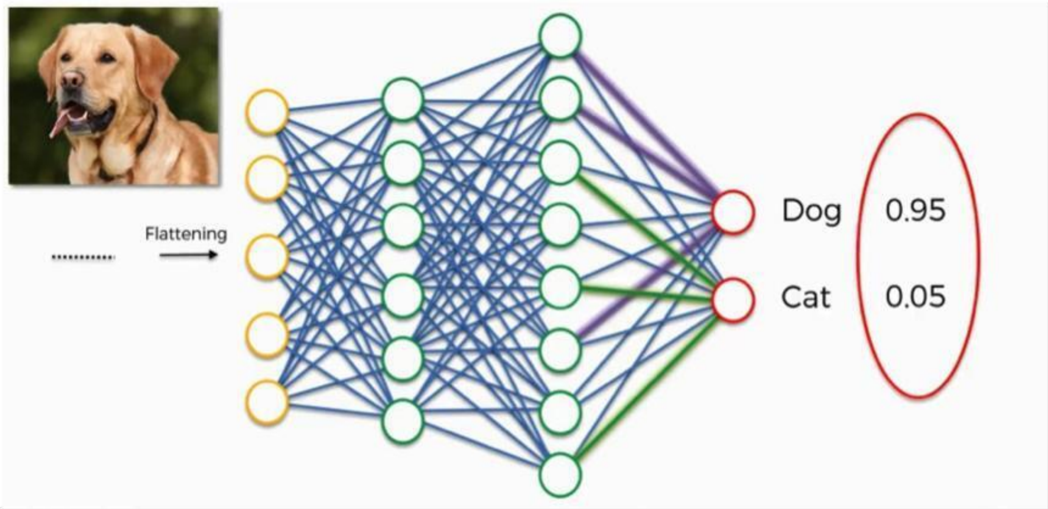
Первый процесс называется процессом Map, а второй – процессом Reduce.

Это довольно простые концепции, но оказалось, что вы можете делать с их помощью много разных видов обработки, выполнять много разных задач и обрабатывать очень большие наборы данных.

И такая архитектура называется Hadoop.

И когда у нас появились вычислительные возможности для обработки данных, у нас появились новые методы, такие как машинное обучение.

С помощью которого мы можем взять большие наборы данных, и вместо того, чтобы брать выборку из этих данных и пытаться проверить какую-то гипотезу, мы можем взять большие наборы данных и искать в них шаблоны – закономерности.



То есть перейти от проверки гипотез к поиску шаблонов, которые, возможно, будут генерировать гипотезы.

Это отличается от традиционной статистики, где у вас должна быть гипотеза, которая не зависит от данных, и затем вы проверяете ее на данных.

В машинном обучении сами данные генерируют гипотезы.

С появлением больших данных и вычислительных возможностей стало актуальным глубокое машинное обучение и использование нейронных сетей.

Jupyter Notebook

Технология нейронных сетей существовала 30 лет назад, но ее развитие сдерживалось нехваткой данных и вычислительных возможностей.

Нейронные сети – это попытка подражать нейронам мозга и тому, как на самом деле функционирует наш мозг.

Нейронная сеть получает некоторые входные данные, которые затем передаются в разные узлы обработки, которые выполняют некоторые преобразования в данных, а затем передают результаты на другой уровень узлов и, наконец, сеть выдает конечный результат.

Таким образом, нейронная сеть представляет собой компьютерную программу, которая имитирует, как наш мозг использует нейроны.

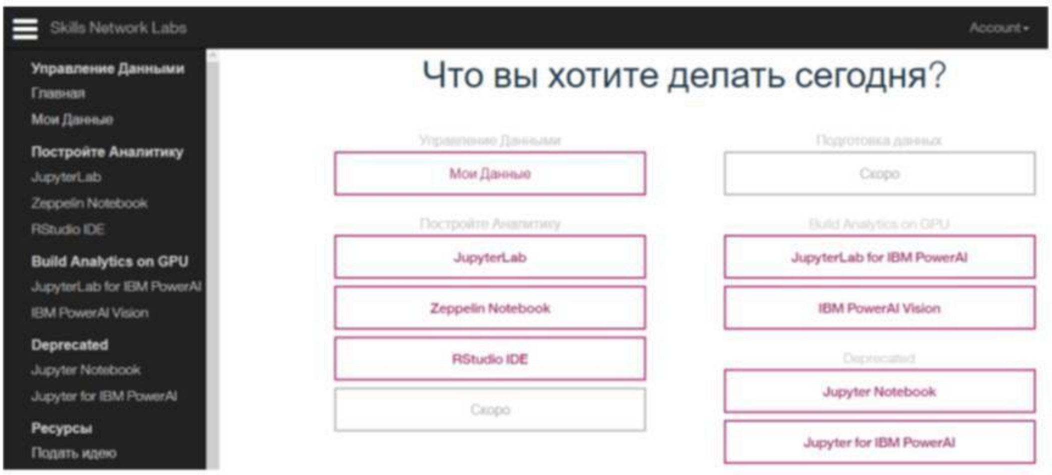
Нейронная сеть содержит входы и выходы, и вы продолжаете вводить данные в эти входы, и смотрите на выходы, и вы продолжаете делать это снова и снова, таким образом, чтобы эта сеть давала нужные результаты, при этом регулируя преобразования внутри сети.

Так вы обучаете нейронную сеть.

И теперь у нас есть нейронные сети и глубокое обучение, которые могут распознавать речь и распознавать людей.

И глубокое обучение требует больших вычислительных мощностей, так что это не то, что вы можете делать на своем ноутбуке, вы можете поиграть с нейронной сетью, но, если вы действительно хотите сделать что-то серьезное, у вас должен быть доступ к специальным вычислительным ресурсам.

https://labs.cognitiveclass.ai



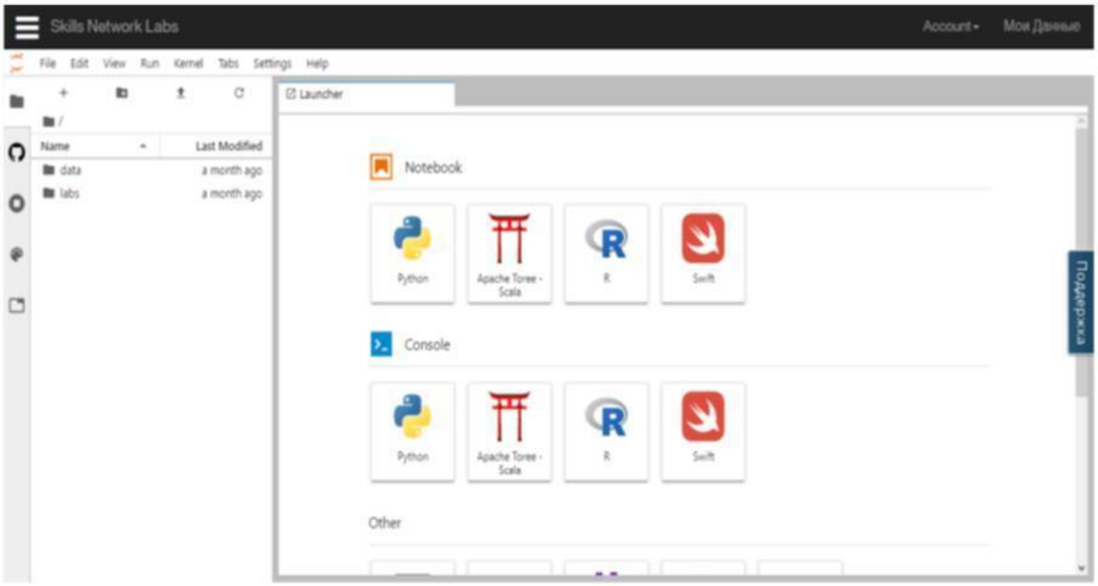
Теперь, для обучения работе с данными существуют бесплатные инструменты, например, Skills Network Labs от компании IBM.

Это бесплатная виртуальная лабораторная среда, которая позволяет практиковаться и изучать науку о данных.

Skills Network Labs содержит такие инструменты, как RStudio, Jupyter и Zeppelin.

И эти инструменты предоставляют интерактивную среду для анализа данных, визуализации данных, машинного обучения и распознавания изображений.

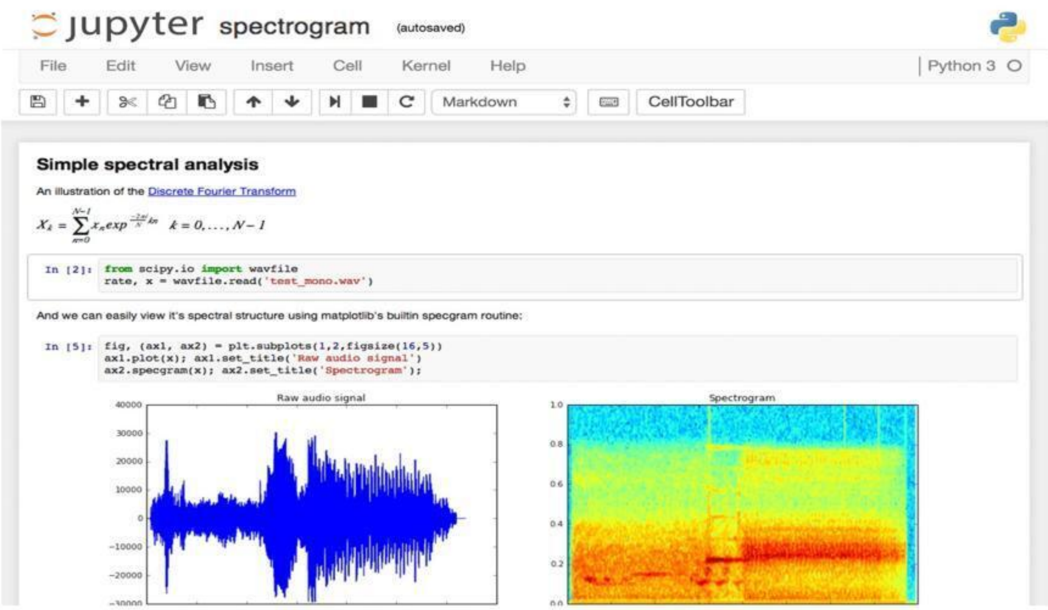
Например, кнопка JupyterLab откроет собой интерактивную среду, которая позволяет запускать или создавать записные книжки notebook, которые запускают коды на Python с помощью Jupyter Notebooks, Scala на Apache Toree и R.



Jupyter Notebook – это веб-приложение, в котором вы можете создавать и обмениваться документами, содержащими живой код, уравнения, визуализации, а также текст.

И Jupyter Notebook является одним из инструментов, помогающих приобрести необходимые навыки в области науки о данных.

Что такое Jupyter Notebook?



В данном случае «записная книжка» notebook означает документ, который содержат как код, так и элементы форматированного текста, такие как рисунки, ссылки, уравнения и так далее.

И из-за такого сочетания кода и текста, эти документы являются идеальным местом собрать воедино описание анализа данных и его результаты, а также возможность выполнить анализ данных в режиме реального времени.

И приложение Jupyter Notebook создает такие документы.

«Jupyter» является аббревиатурой, означающей Julia, Python и R.

Эти языки программирования были первыми языками, которые поддерживал Jupyter, но в настоящее время технология Jupyter также поддерживает другие языки, на которых можно писать код в Jupyter.

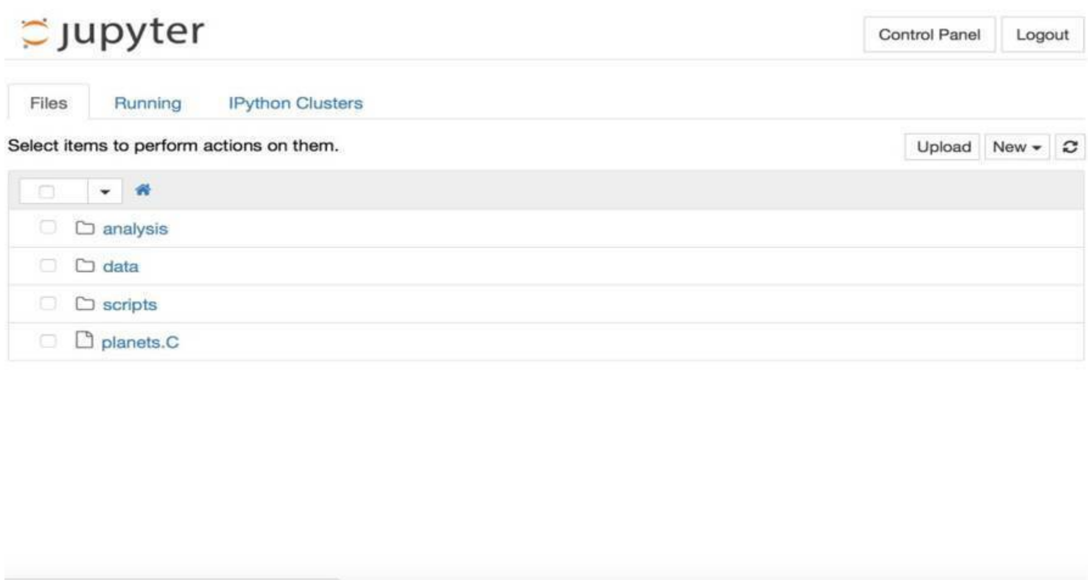
Таким образом, документы notebook – это документы, созданные приложением Jupyter Notebook, которые содержат как компьютерный код (например, python), так и элементы форматированного текста (абзацы, уравнения, рисунки, ссылки и т. д.).

Документы notebook – это читаемые документы, содержащие описание анализа и результаты анализа данных (рисунки, таблицы и т. д.), а также исполняемый код, который можно запустить для анализа данных.

Что такое приложение Jupyter Notebook?

Это клиент-серверное приложение, которое позволяет редактировать и запускать записные книжки notebook через веб-браузер.

Приложение Jupyter Notebook может быть запущено на компьютере без доступа к Интернету или установлено на удаленном сервере, где вы можете получить к нему доступ через Интернет.



Помимо отображения, редактирования и запуска записных книжек, в приложении Jupyter Notebook есть «Панель инструментов» (Notebook Dashboard), отображающая локальные файлы и позволяющая открывать записные книжки и останавливать их ядра.

Ядро – это программа, которая запускает код, написанный в записной книжке.

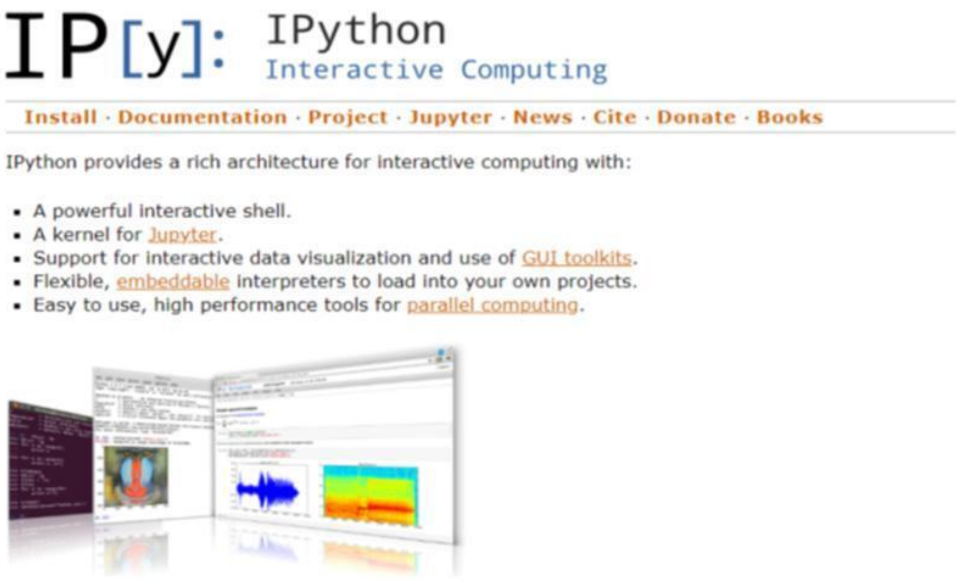
Приложение Jupyter Notebook имеет ядро ipython для кода Python, но также есть ядра, доступные для других языков программирования.

Когда вы открываете документ Notebook, соответствующее ядро запускается автоматически.

И ядро выполняет вычисления и выдает результаты.

Теперь, как появился Jupyter Notebook.

В 2001 году, программист Фернандо Перес начинает разработку IPython – интерактивную оболочку для языка программирования Python.



И в 2005 году и Роберт Керн, и Фернандо Перес попытались создать систему для ноутбуков.

К сожалению, прототип ноутбука так и не стал полностью пригодным для использования.

Но команда IPython продолжала работать, и в 2007 году они сделали еще одну попытку внедрения системы типа ноутбуков.

К октябрю 2010 года появился прототип веб-ноутбука, и в 2011 года этот прототип был реализован.

И наконец, в 2014 году проект Jupyter был запущен как отдельный проект от проекта IPython.

И IPython – теперь одно из ядер Jupyter.

И теперь появилось новое поколение ноутбуков Jupyter, которые называются JupyterLab.

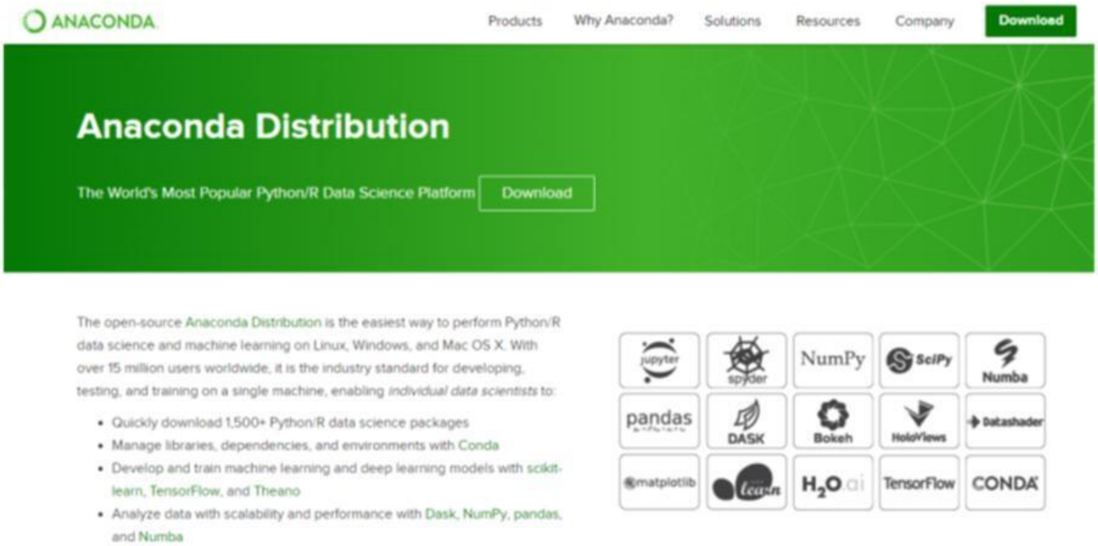
Теперь, как возникла эта идея ноутбуков.

Концепция ноутбука, который содержит обычный текст и расчеты и графику, была не новой.

Фернандо Перес был активным пользователем ноутбуков Mathematica, которые были созданы в качестве графического интерфейса в 1988 году Теодором Греем, основателем компании Wolfram Research – частная компания, занимающаяся производством математического программного обеспечения. Её основным продуктом является среда технических расчётов Mathematica.

И сочетание этой идеи ноутбуков с технологией AJAX веб-приложений, которая не требует от пользователей обновления веб страницы каждый раз, когда изменяются ее данные, дало старт разработке ноутбуков Jupyter.

https://www.anaconda.com/distribution/



Для установки Jupyter на свой компьютер, рекомендуется установить Анаконду, которая установит Python, Jupyter Notebook и другие часто используемые пакеты для научных вычислений и обработки данных.

Anaconda – это бесплатный дистрибутив языков программирования Python и R для научных вычислений с открытым исходным кодом, где пакетами управляет инструмент conda, а не pip.

```
pip3 install --upgrade pip

pip3 install jupyter

jupyter notebook
```

Как альтернатива, вы можете установить только Jupyter с помощью инструмента pip.

И после установки, вы можете запустить Jupyter на своем компьютере с помощью команды jupyter notebook.

Эта команда запустит сервер ноутбука и откроет веб страницу интерфейса Jupyter по адресу http://localhost:8888/.

И здесь вы увидите Панель управления записными книжками.

Закрытие браузера или его вкладки не приведет к закрытию приложения Jupyter Notebook.

Чтобы полностью закрыть приложение, необходимо закрыть соответствующий терминал, где вы запустили Jupyter.

Так как приложение Jupyter Notebook – это сервер, который отображается в вашем браузере по адресу http://localhost:8888.

И закрытие браузера не приведет к выключению сервера.

Вы можете снова открыть адрес, и приложение Jupyter Notebook будет отображено заново.

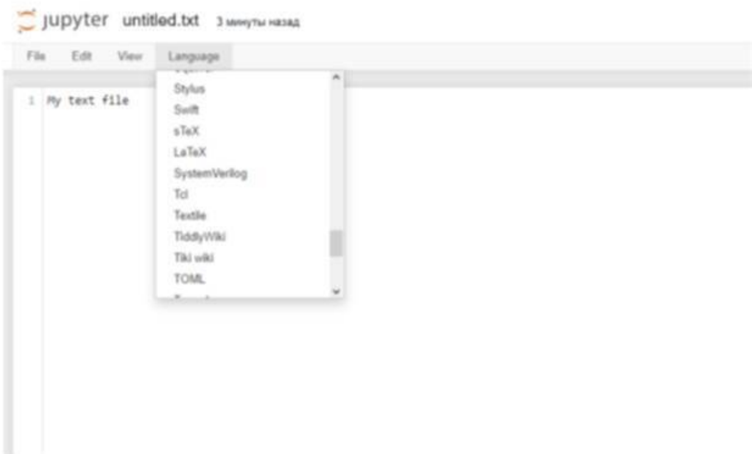
Вы можете запустить много копий приложения Jupyter Notebook, и они будут отображаться по адресу localhost только с разными портами.

И так как с помощью одного приложения Jupyter Notebook вы уже можете открывать множество записных книжек, не рекомендуется запускать несколько копий приложения Jupyter Notebook.

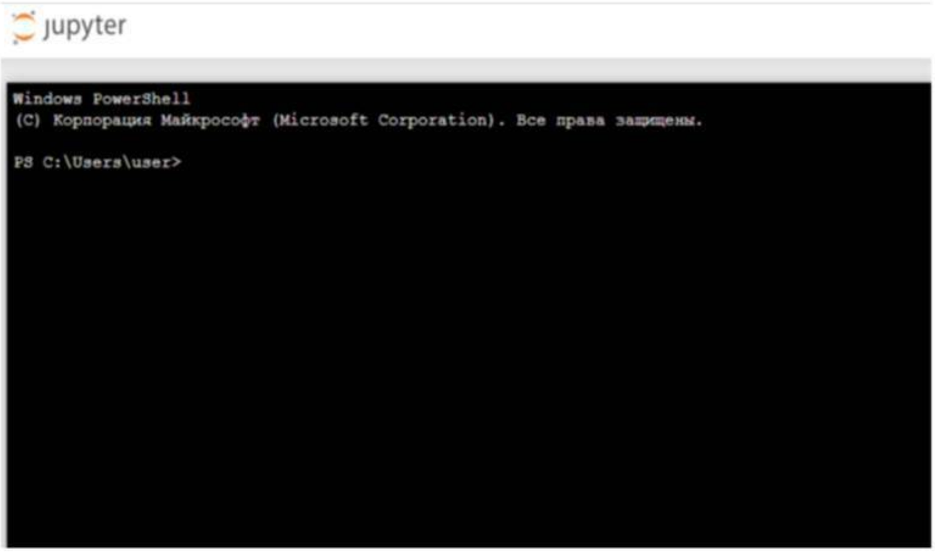
Во вкладке Files веб страницы Jupyter показываются все ваши файлы, во вкладке Running показываются все процессы ноутбуков, а третья вкладка Clusters предоставляется параллельной вычислительной средой IPython.

Чтобы создать новый ноутбук, нужно нажать на кнопку New во вкладке «Файлы».

И здесь вы можете создать обычный текстовый файл, папку и терминал.
Также вы можете создать ноутбук на Python 3.
Начнем сначала с создания обычного текстового файла.

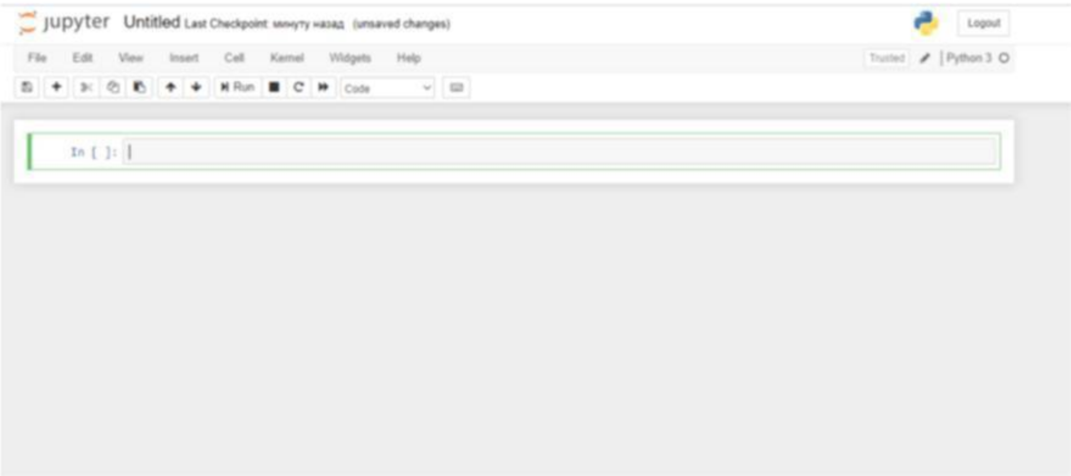


Когда он откроется, вы увидите, что это похоже на любой другой текстовый редактор.
Здесь вы можете указать язык программирования, на котором вы пишете.
Также, вы можете сохранить, переименовать или загрузить файл или создать новый файл.
Вы также можете создавать папки, чтобы организовать ваши документы.
Для этого нажмите Folder в меню New, и у вас появится новая папка.
Далее вы сможете переименовать эту папку.



Что касается терминала, терминал предназначен для поддержки сеансов интерактивного терминала на основе веб браузера.

И этот терминал работает так же, как терминал операционной системы или приложение cmd.



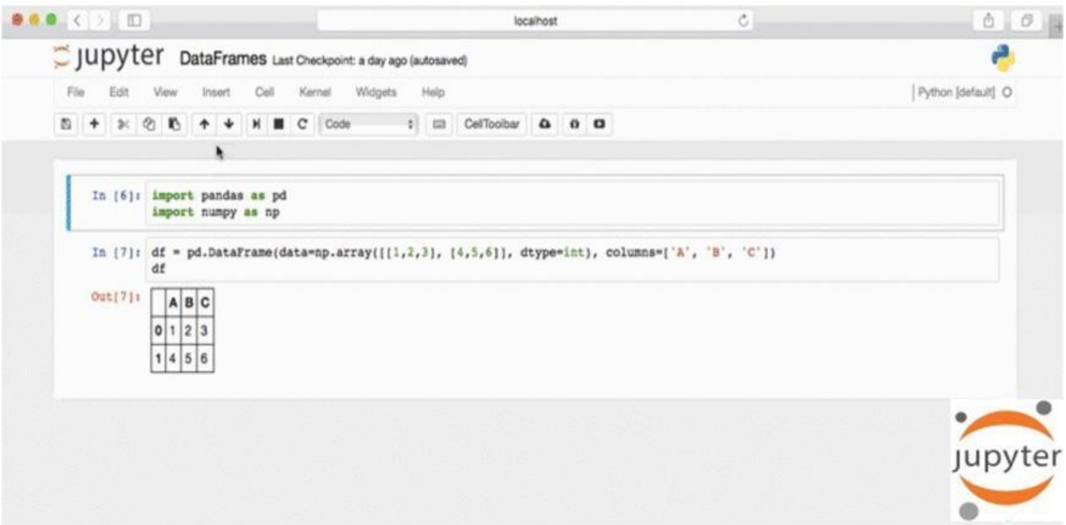
Если же вы хотите запустить свой ноутбук, вернитесь в главное меню и выберите опцию Python 3 в меню New.

Здесь вы увидите строку меню, панель инструментов и пустую ячейку кода.

И в этой ячейке кода вы можете сразу начать с импорта необходимых библиотек для вашего кода.

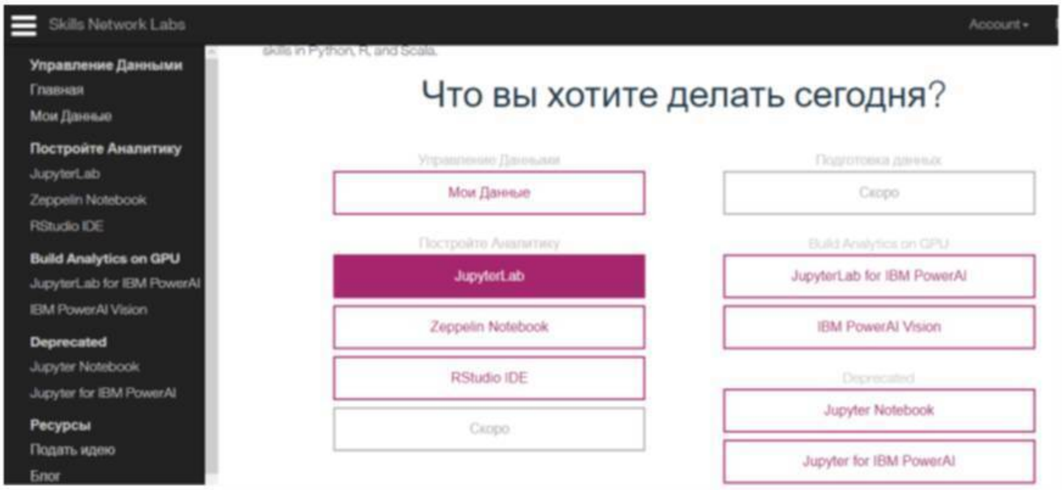
После этого вы можете добавлять, удалять или редактировать ячейки кода, вставляя пояснительный текст и заголовки.

После создания ноутбука, вы можете использовать меню «Файл», чтобы загрузить свою записную книжку в виде HTML, PDF, и так далее.



Вместо настольной версии, вы можете использовать среду Skills Network Labs от IBM.

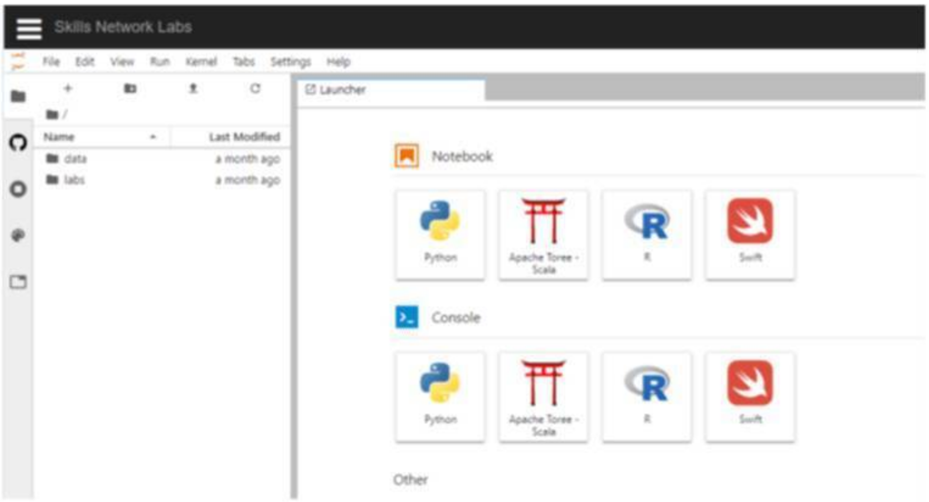
Здесь вы можете открыть JupyterLab – среду, которая позволяет создавать и редактировать блокноты Jupyter.



Когда вы откроете JupyterLab, вы увидите, что с левой стороны у вас есть каталог с файлами, где вы можете отслеживать все ваши файлы, а с правой стороны – экран запуска.

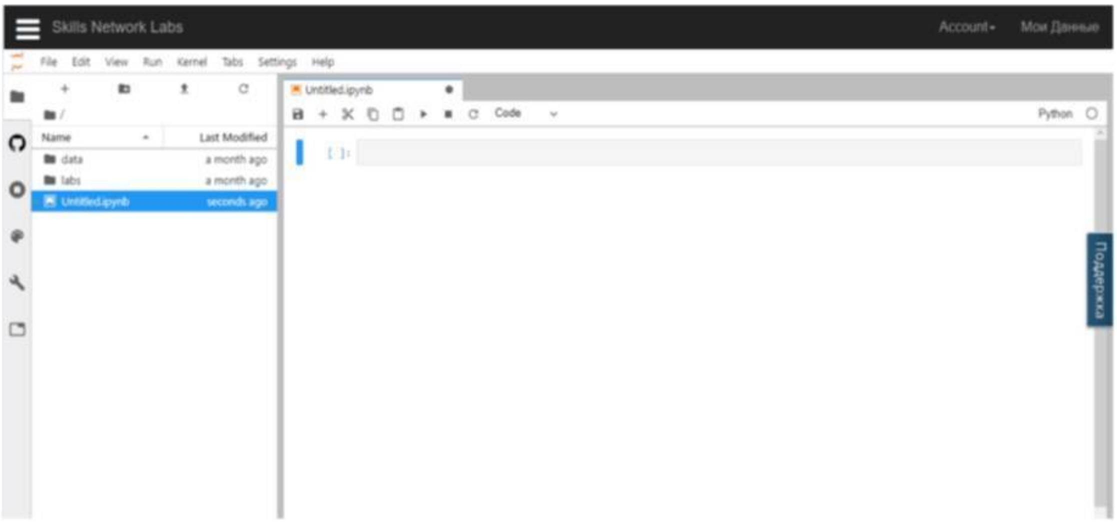
С помощью этого запуска вы можете создавать новые записные книжки Jupyter с Python 3, Scala или R.

Давайте создадим записную книжку с Python 3.

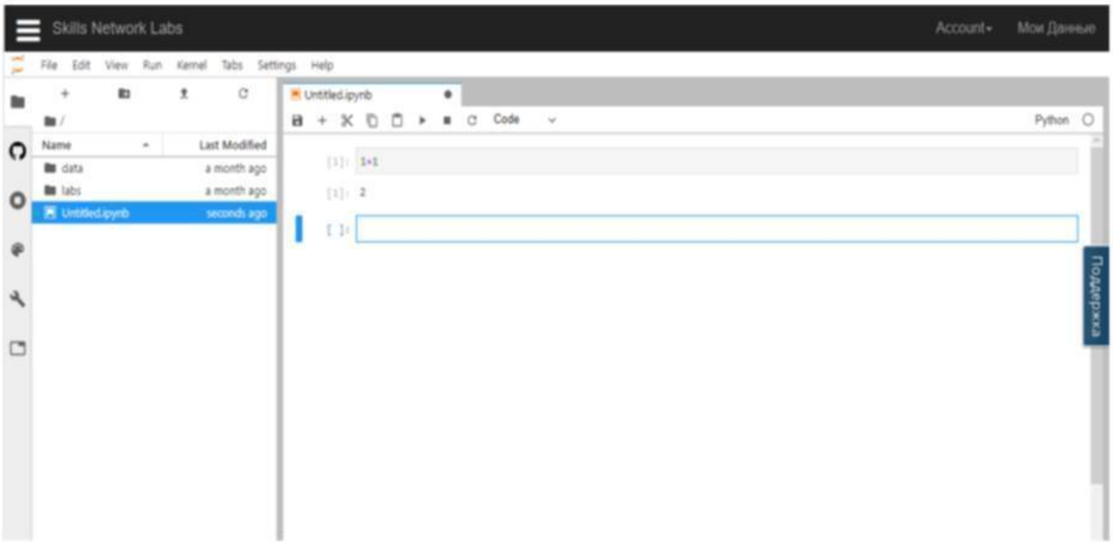


Ноутбуки Jupyter состоят исключительно из ячеек.

Сейчас в этой записной книжке, есть только одна ячейка.

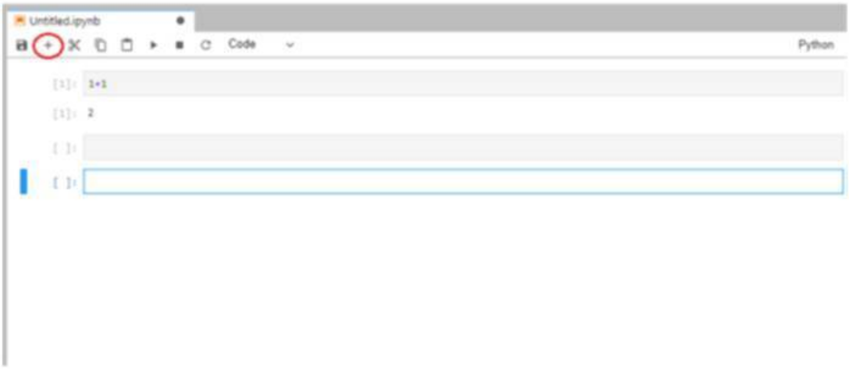


Давайте напишем что-нибудь внутри ячейки, например, 1 + 1, а затем на клавиатуре нажмем Shift + Enter.



В результате выполнится код с использованием Python, который вернет два.

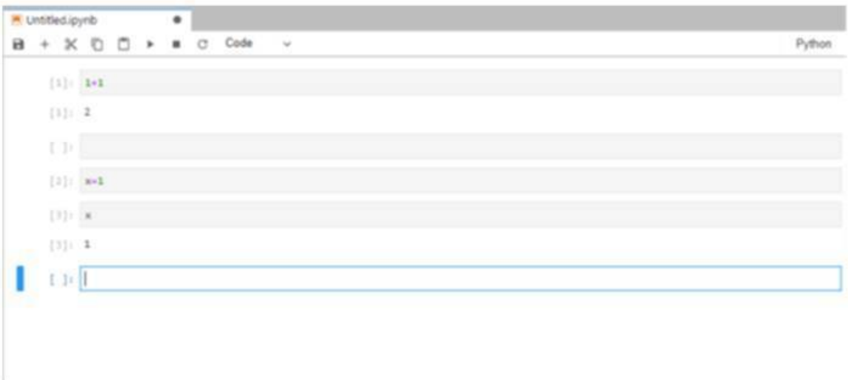
Далее мы можем создать еще больше ячеек, нажав кнопку «Плюс» вверху, и написать больше кода.



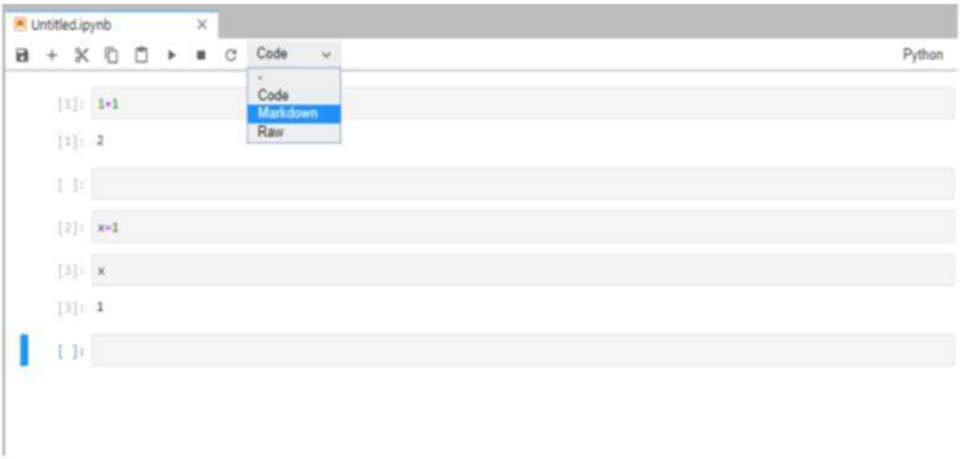
Например, установив X равным целому числу, и распечатать вывод X в ячейке ниже.

И все эти ячейки являются так называемыми ячейками кода, которые позволяют запускать код с использованием интерпретатора, в данном случае Python 3.

Но как мы можем добавить заголовки или текст в нашу записную книжку?



Для этого нужно преобразовать тип ячейки из кода в ячейку markdown, щелкнув на ячейке, и выбрав markdown в меню.



И теперь, если вы введете что-то вроде «один плюс один» в эту ячейку и попытаетесь запустить ее с помощью Shift + Enter, она будет преобразована непосредственно в текст.

И чтобы отредактировать ячейку снова, просто дважды щелкните по ячейке.

markdown – это разметка, с помощью которой вы можете стилизовать ваш текст.

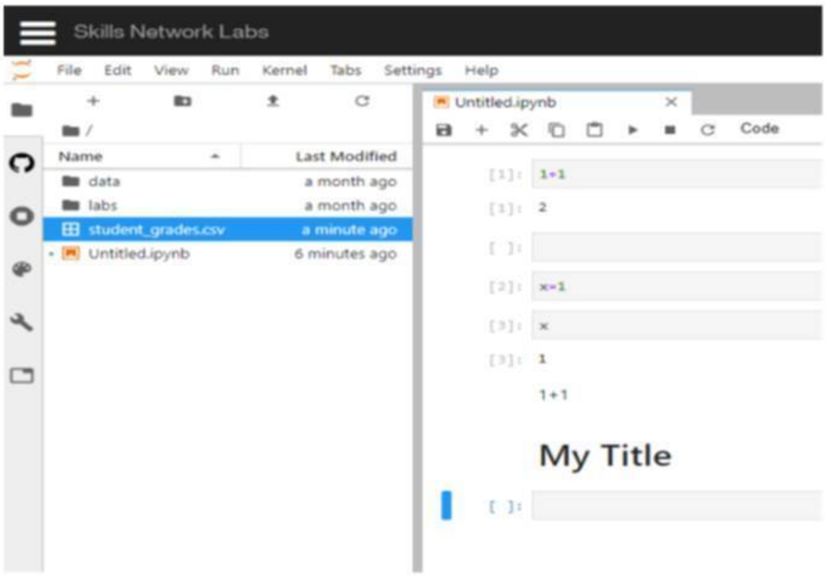
Например, вы можете создавать заголовки, используя символ решетки и пробел, за которым следует некоторый текст, такой как «Мой заголовок».

Есть и другие способы стилизовать ваш текст.

И здесь показаны несколько примеров.

В ячейках markdown вы также можете использовать HTML.

Например, если вы хотите встроить изображение.



В JupyterLab вы можете импортировать данные, например в виде CSV-файла, и использовать их в блокноте Jupyter.

Чтобы импортировать данные, вы можете просто перетащить файл данных прямо в каталог файлов с левой стороны.

После завершения загрузки он будет отображаться в каталоге.

И это не обязательно должен быть файл CSV, это может быть файл любого типа.

Вы также можете создавать различные папки для организации всех ваших файлов.

И вы можете дважды щелкнуть файл, чтобы открыть предварительный просмотр его содержимого.

```
[4]: import pandas as pd

[5]: df=pd.read_csv('student_grades.csv')

[6]: df.head()
```

[6]:

	Student	Grade
0	John Smith	80
1	Jane Smith	75
2	John Doe	65
3	Jane Doe	90

Для обработки данных файла CSV в Python нам нужно использовать функцию чтения CSV библиотеки pandas.

Поэтому сначала импортируем панду.

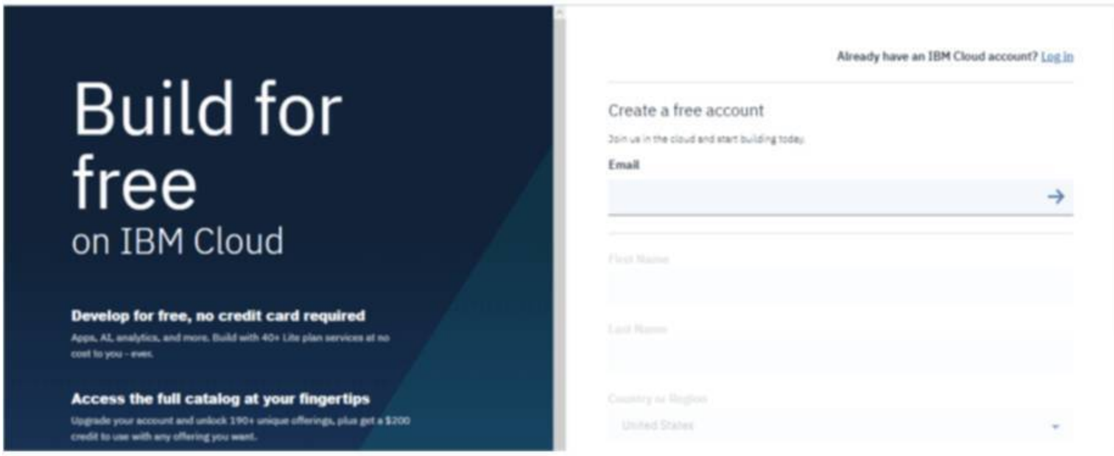
Затем вы можете прочитать файл, используя путь к файлу CSV.

Теперь, вы можете распечатать первые пять строк файла CSV.

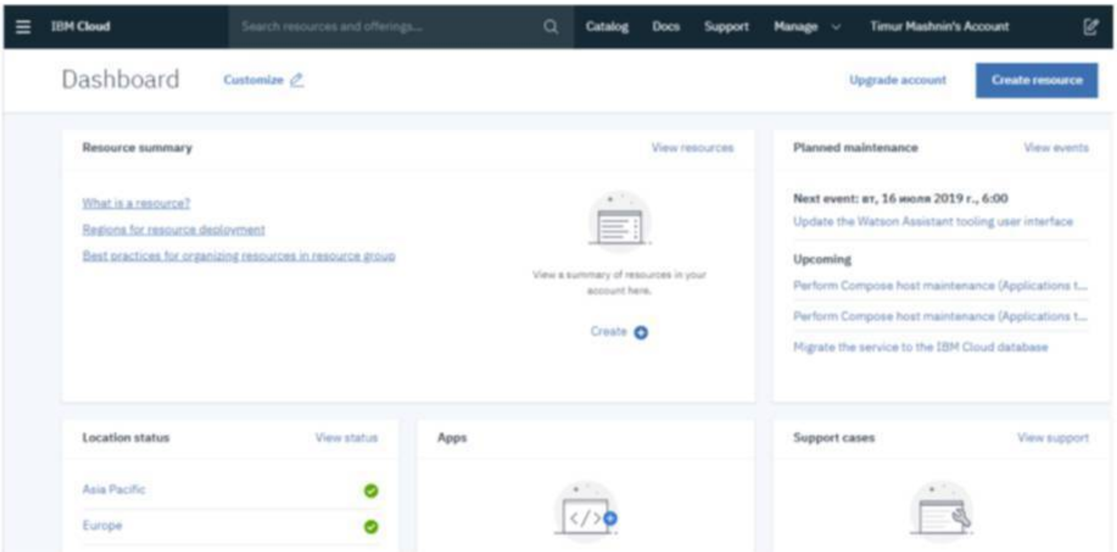
Создание чат-ботов без программирования

Для создания чат-бота, здесь мы будем использовать службу Watson Assistant, размещенную на платформе IBM Cloud.

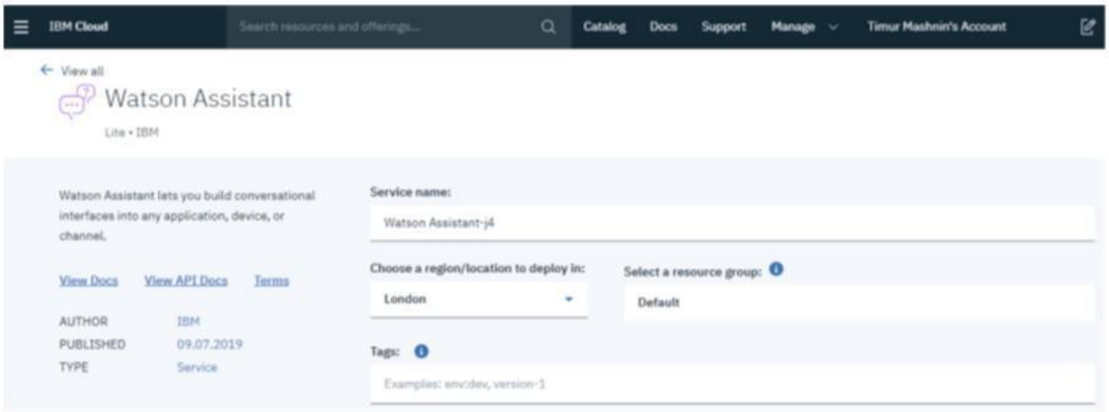
https://cloud.ibm.com/registration



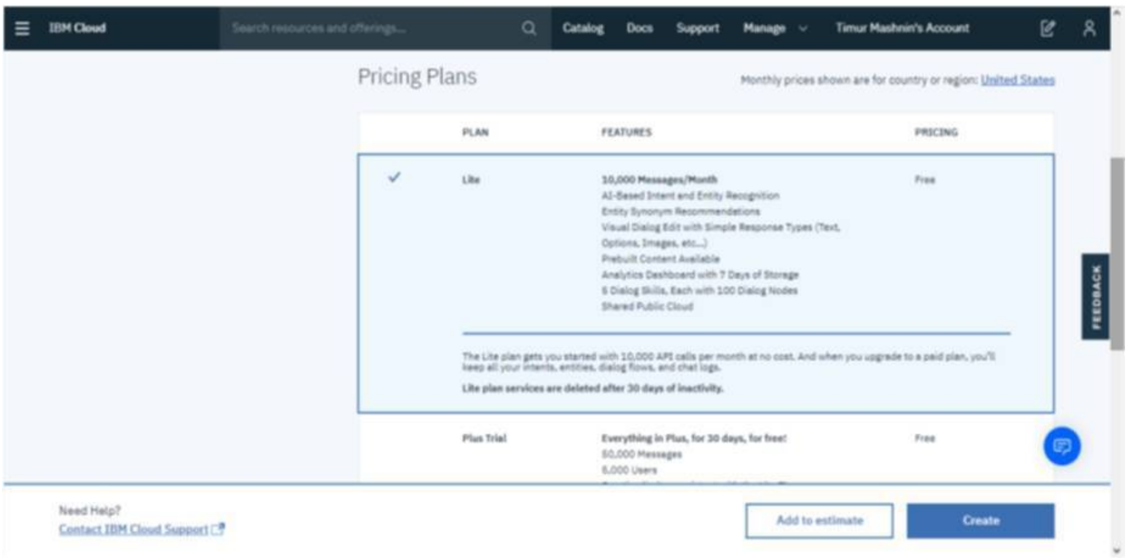
И для начала, вам нужно зарегистрироваться в IBM Cloud.



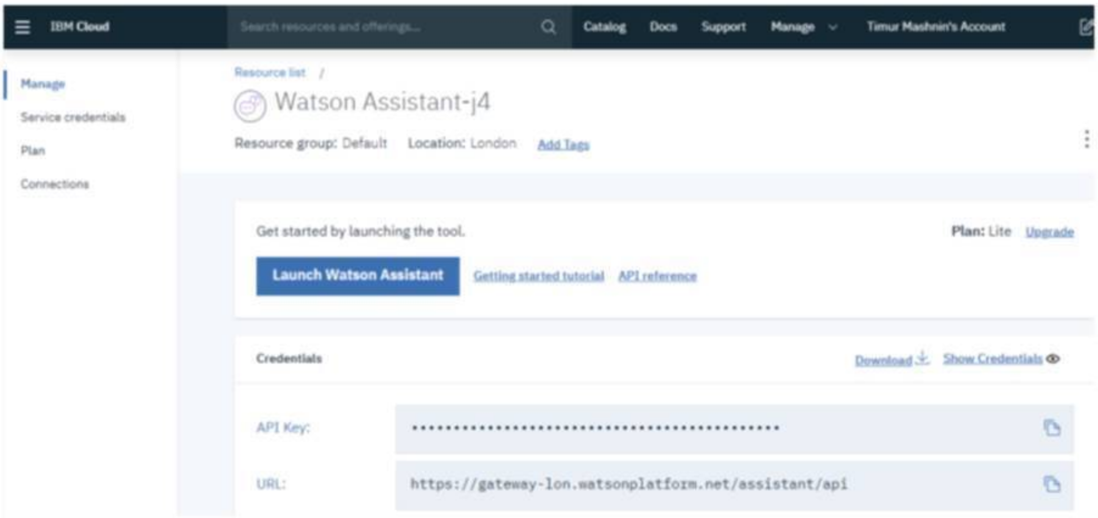
После успешного входа, вы увидите панель управления.
Вверху, в поиске, наберите Watson Assistant и перейдите на страницу Watson Assistant.



Здесь вы можете изменить регион / местоположение для оптимальной производительности.

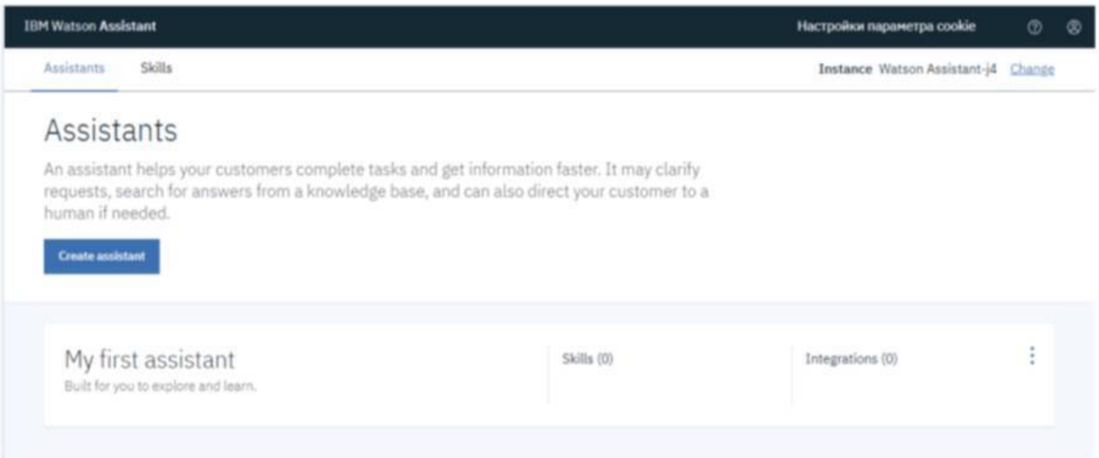


Далее выберете план Lite, и нажмите кнопку «Создать» внизу, чтобы создать свой экземпляр сервиса.



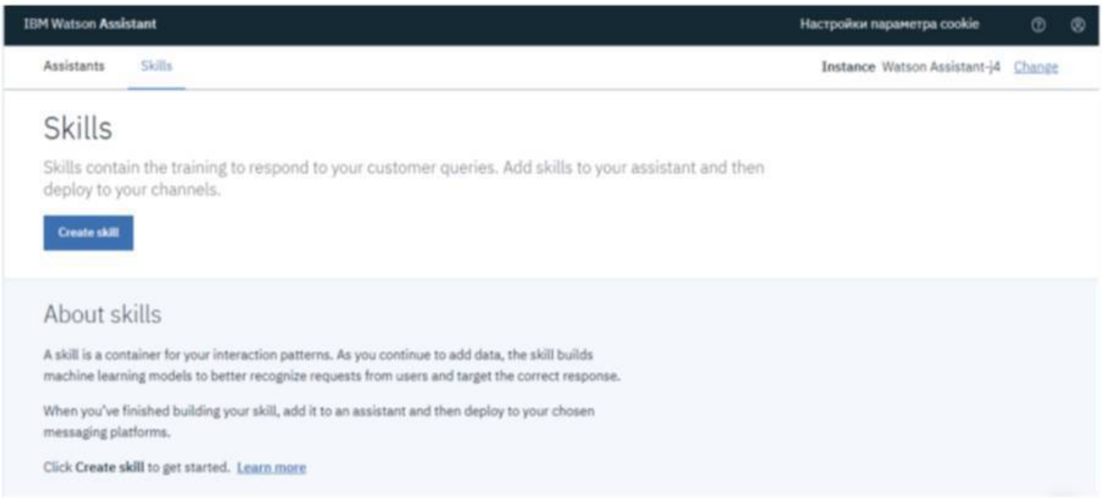
Вы будете перенаправлены на страницу запуска службы, которую вы только что создали.

Нажмите кнопку «Запустить инструмент» для доступа к веб-приложению, которое позволит вам создавать чат-ботов.



Вы попадете на страницу веб-приложения.

Нажмите на меню «Навыки» Skills в верхней части страницы.



Здесь мы будем учить нашего чат-бота общаться.

Для быстрого доступа, добавьте эту страницу в закладки.

Здесь мы будем создавать чат-бота для вымышленной сети цветочных магазинов.

Чатбота, который предоставляет информацию и помощь потенциальным клиентам.

Затем мы развернем его на сайте WordPress.

Теперь, давайте выясним, что такое чат-боты и как они могут помочь нам.

Позвонив в службу поддержки, мы можем услышать сообщение – «Ваш звонок может быть записан».

И как мы увидим, с помощью чат-ботов мы можем использовать разговоры с клиентами, чтобы обучать и делать наших чат-ботов умнее и полезнее.

В этом примере с техподдержкой, как правило мы имеем дело с простыми вопросами, которые требуют простых ответов.

Но сама техподдержка не очень хорошо масштабируется.

Если у вас дела идут хорошо, у вас будет все больше клиентов, которым нужна ваша помощь.

Так что вам придется нанимать все больше и больше людей, а это требует денег.

Вам также нужно будет потратить время и энергию, чтобы правильно их обучить, управлять ими и так далее.

Чатботы не предназначены для полной замены людей.

Но они могут помочь ответить на большое количество простых вопросов клиентов.

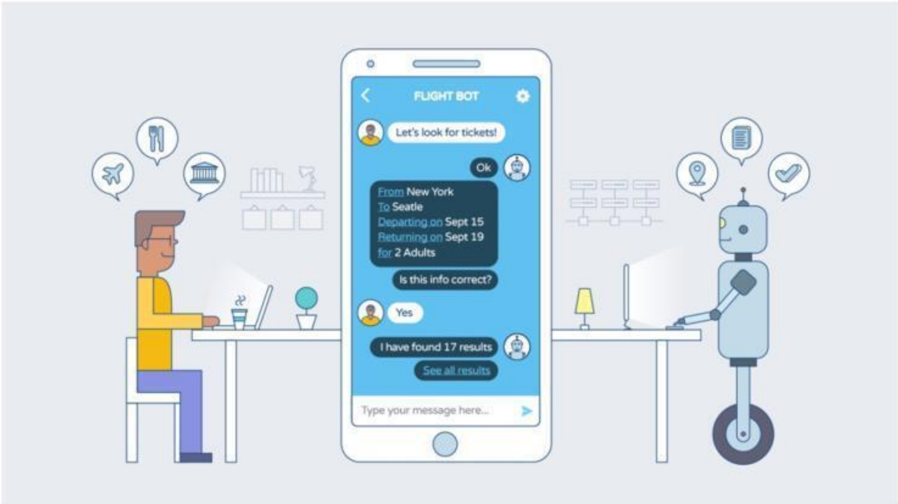
И они могут масштабироваться до бесконечности, в отличие от людей.

И доступны 24/7 дней в году, в отличие от работников.

Например, отель может резко сократить количество звонков, которые получает его стойка регистрации, просто имея чат-бота, который отвечает на самые распространенные вопросы, такие как часы работы ресторана или время оформления заказа, установка будильника или выяснение того, как подключиться к Wi-Fi.

Это оставляет персоналу отеля больше времени и энергии для решения более сложных задач.

Теперь, что такое чатбот?



Это программный агент, способный общаться с пользователями с помощью интерфейса чата.

Как правило, чат-бот приветствует пользователя и предлагает ему выполнить какое-либо действие, например, задать ему вопрос.

Когда пользователь отвечает, чат-бот анализирует ввод и определяет намерение пользователя.

И наконец, он отвечает, предоставляя информацию или запросив дополнительную информацию, прежде чем окончательно ответить на вопрос.

Хорошие чат-боты могут поддерживать это общение естественным образом.

Они заставляют пользователя чувствовать, что его понимают и помогают.

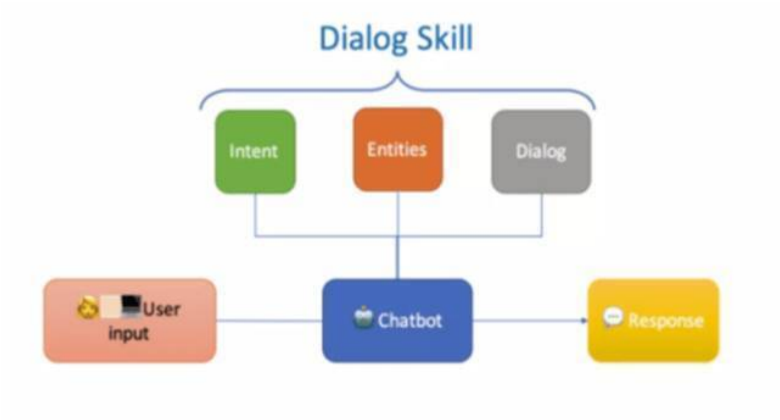
Они создают связь с пользователем, не притворяясь человеком.

И наиболее распространенные чат-боты – это текстовые чат-боты.

Взаимодействие с ними происходит во всплывающем окне чата на веб-сайте или через приложение для обмена сообщениями, такое как Whatsapp.

Тем не менее, вы можете общаться с некоторыми чат-ботами с помощью голоса.

Это виртуальные помощники, такие как Apple Siri.



Чатбот использует три основных компонента, чтобы определить, как интерпретировать вводимые пользователем данные и как на них реагировать.

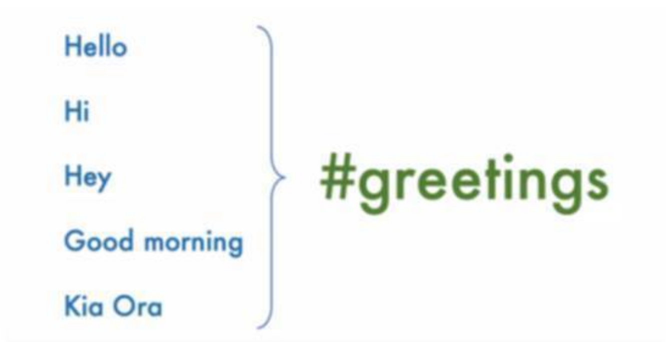
Это intents, намерения, entities, сущности, и, наконец, диалог.

И когда вы создаете чат-бота в Watson Assistant, первое, что вы делаете, это создаете Навык диалога dialog Skill, который будет содержать эти три компонента.

Первое, что вам нужно сделать, это создать навык диалога, который будет содержать эти три компонента.

Намерения являются наиболее важным компонентом, потому что они пытаются определить, что хочет пользователь.

Другими словами, они фиксируют намерение или цель пользователя.



Например, мы могли бы определить намерение #greetings и обучить Уотсона, как выглядит приветствие.

Мы могли бы предоставить Привет, Доброе утро, и так далее.

Хорошей практикой является предоставление как минимум 5 примеров для каждого намерения, чтобы полностью обучить Уотсона этому намерению.

После того, как Уотсон изучит намерения, которые мы определили, он будет изучать вводимые пользователем данные и попытается определить, совпадает ли какое-либо из намерений с запросом пользователя.

Например, если бы пользователь сказал: «Алоха», Уотсон определил бы, что это приветствие, похожее на то, которому мы его обучали.

Именно здесь действительно задействуются возможности искусственного интеллекта Уотсона.

Мы обучаем его нескольким примерам, и Уотсон сможет распознать намерение пользователя, даже если пользователь сформулирует его совершенно иначе, чем данные нами примеры.

Теперь, давайте рассмотрим предметно-ориентированное намерение, а не простое намерение в чате.

Пользователи могут захотеть узнать о часах работы, и мы могли бы определить намерение #hours_info.

И обратите внимание, что имя намерения не может содержать пробелов.

Поэтому мы используем подчеркивание вместо пробела.

И здесь показаны несколько примеров, которые мы могли бы предоставить, чтобы обучить Уотсона работе с информацией о часах.

«До какого времени вы открыты?», «В какие часы вы работаете?», «Вы открыты по субботам?» и так далее.

Все это реальные способы, с помощью которых пользователи могут выражать один и тот же запрос относительно информации о часах работы.

Важно обучить Уотсона реальным примерам, так что вы даже можете вставлять опечатки, которые пользователь может случайно сделать, набирая вопрос.

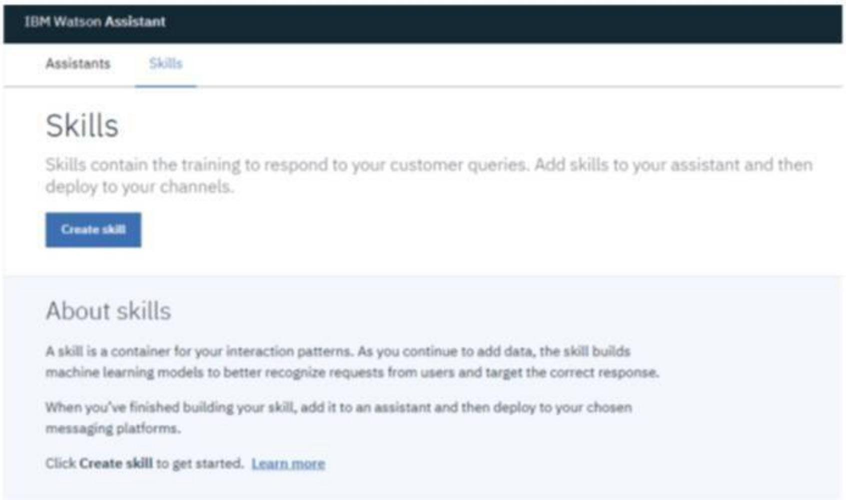
Теперь, когда пользователь спросит: «Когда открыт ваш магазин в Торонто?» Уотсон распознает намерение #hours_info, хотя, опять же, этот конкретный вопрос не был частью наших примеров.

Уотсон достаточно умен, чтобы понять, чего хочет пользователь.

И примеры намерений можно вводить вручную в Watson Assistant, но их также можно импортировать из файлов CSV.

И Watson Assistant также поставляется с Каталогом контента, который предоставляет коллекции намерений, относящихся к различным отраслям, таким как банковское дело, страхование и электронная коммерция.

Это не готовый чат-бот, но вы можете использовать его в качестве отправной точки для дальнейшего развития.

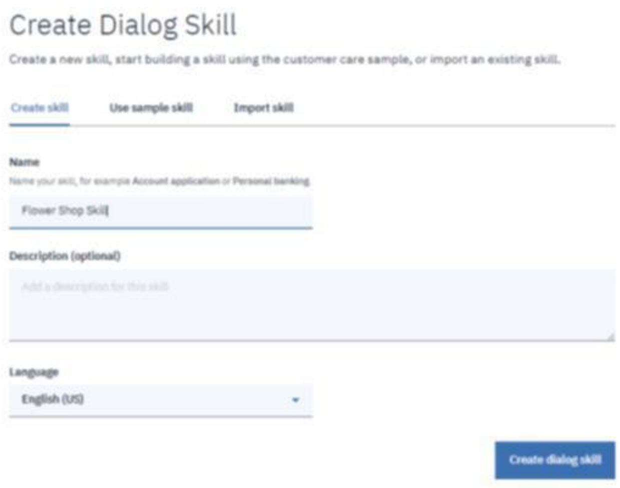


Теперь создадим навык диалога, который будет содержать намерения (наряду с сущностями и самим диалогом).

Хотя возможно создание сложных чат-ботов, использующих несколько навыков, обычно на одного чат-бота обычно приходится только один навык.

Другими словами, сейчас вы можете просто думать о навыке диалога как о чат-боте.

Поэтому нажмем кнопку «Создать навык».



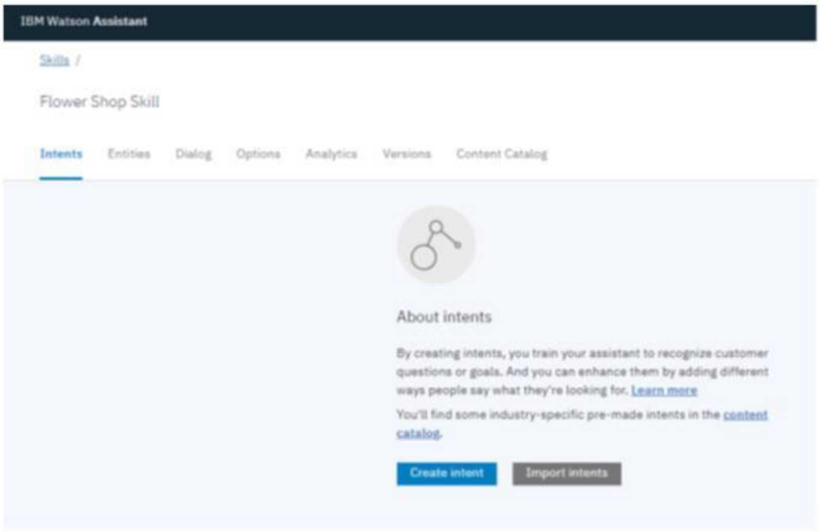
Здесь вы сможете добавить навык диалога.

Вам будет предложено ввести имя и необязательное описание.

Введите навык Цветочный магазин.

Вы также можете увидеть вкладки «Использовать пример навыков» и даже импортировать навыки из файлов JSON во вкладке «Импорт навыков».

Далее нажмите кнопку «Создать», чтобы создать навык для чат-бота.

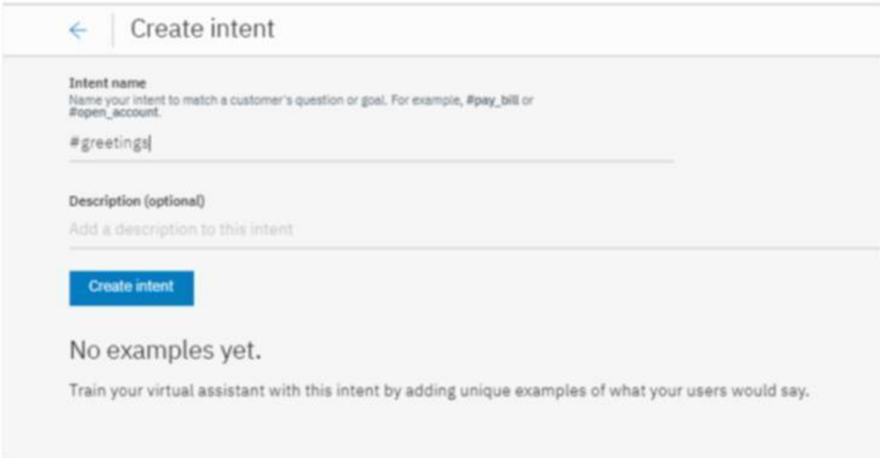


После создания навыка вы окажетесь в разделе намерения этого навыка.

Здесь вы можете добавить намерения несколькими способами.

И мы используем наиболее распространенный способ, а именно добавление намерений вручную.

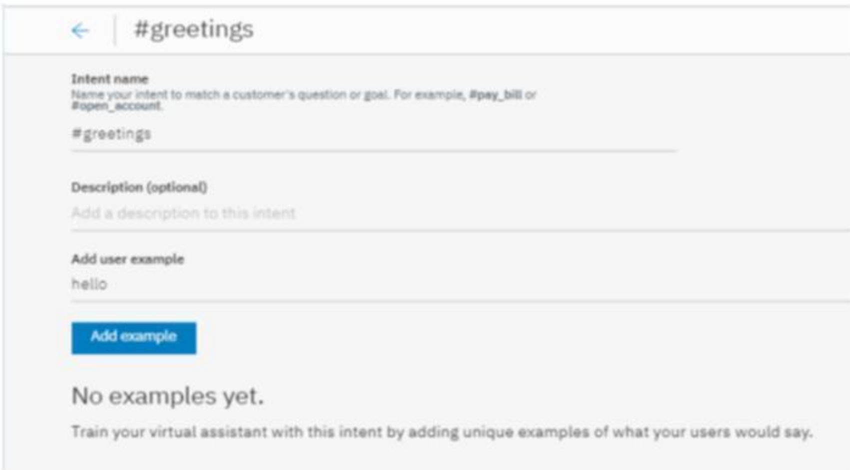
В разделе «Интенты» диалогового навыка нажмем кнопку «Добавить намерение».



Здесь вы сможете определить имя и описание намерения.

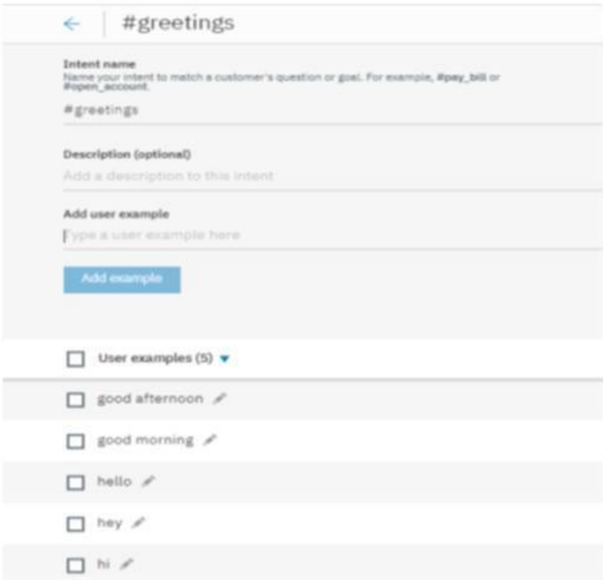
Определим намерение #greetings.

Вы можете оставить описание пустым, а затем нажать кнопку «Создать намерение».



Здесь вам будет предложено создать несколько пользовательских примеров, чтобы обучить Уотсона приветствию.

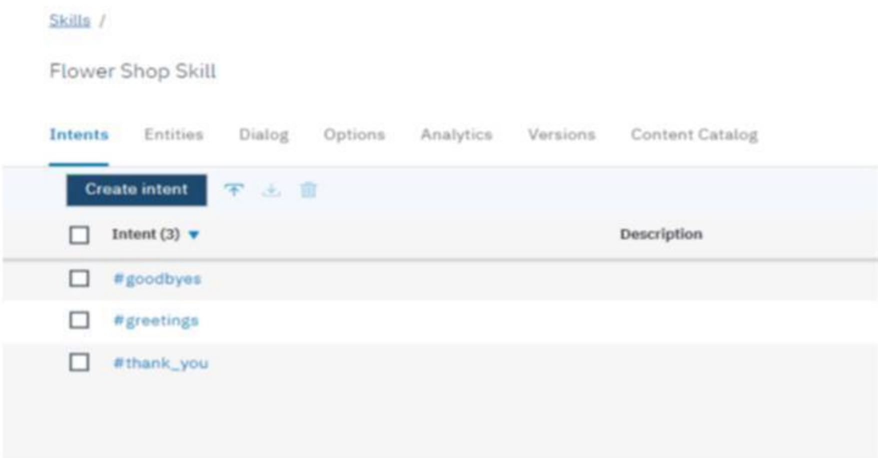
Введите hello, затем нажмите Добавить пример.



И повторите эту процедуру для других примеров приветствия, таких как hi, hey, good morning, good afternoon, и так далее, добавляя по одному пример за раз.

Здесь вы можете редактировать примеры и удалять их.

Нажмите в верхней части на стрелку, чтобы вернуться к списку намерений.

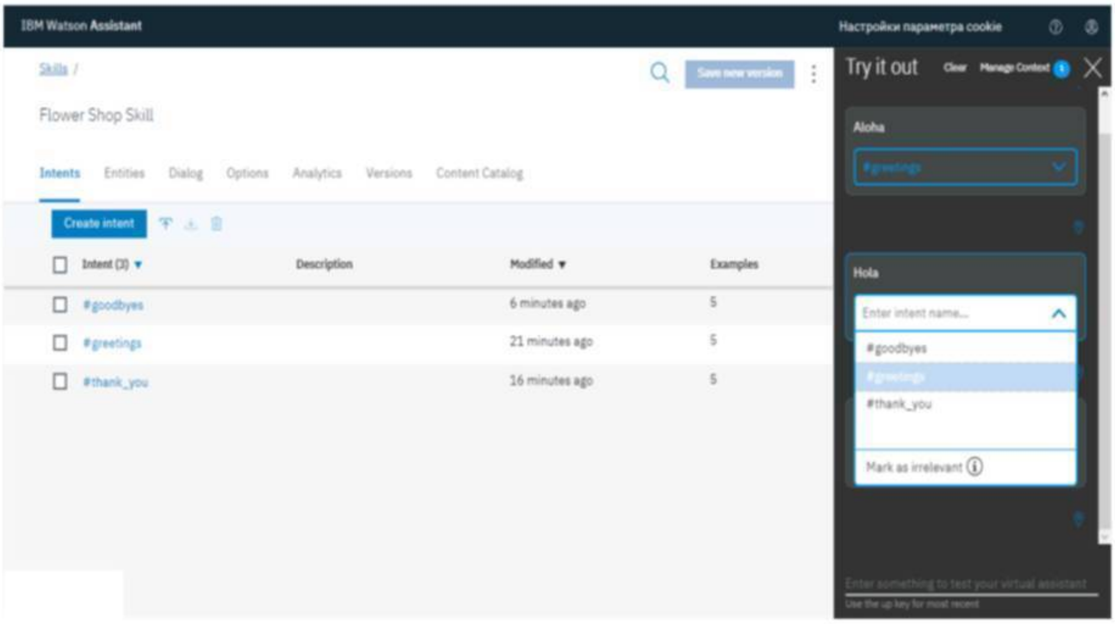


Добавьте намерения #thank_you и #goodbyes по крайней мере с 5 примерами каждое.

Для #thank_you, вы можете использовать такие примеры, как thank you, thanks, thx, cheers, и appreciate it.

Для #goodbyes, вы можете ввести good bye, bye, see you, c ya, и talk to you soon.

И чтобы проверить наши намерения, нажмите кнопку Try it в правом верхнем углу.



Появится панель чата, где вы сможете попробовать ввод пользователя и посмотреть, как Watson анализирует его и как реагирует чат-бот.

Мы еще не предоставили ответы (мы сделаем это в модуле «Диалог»), но мы можем использовать панель для проверки классификации наших намерений.

Попробуйте вводить фразы в панели.

Например, попробуйте Hola и Aloha.

Хотя они специфичны для определенных языков, они достаточно распространены, чтобы быть опознанными Уотсоном в качестве приветствия.

Если Уотсон неправильно классифицировал пример, нажмите на кнопку рядом с введенной фразой, чтобы назначить ей намерение.

Это добавит ваш ввод в качестве примера для намерения, например, #greetings, продолжая обучение Уотсона.

Если вы снова проверите то же самое высказывание, Уотсон на этот раз правильно распознает намерение.

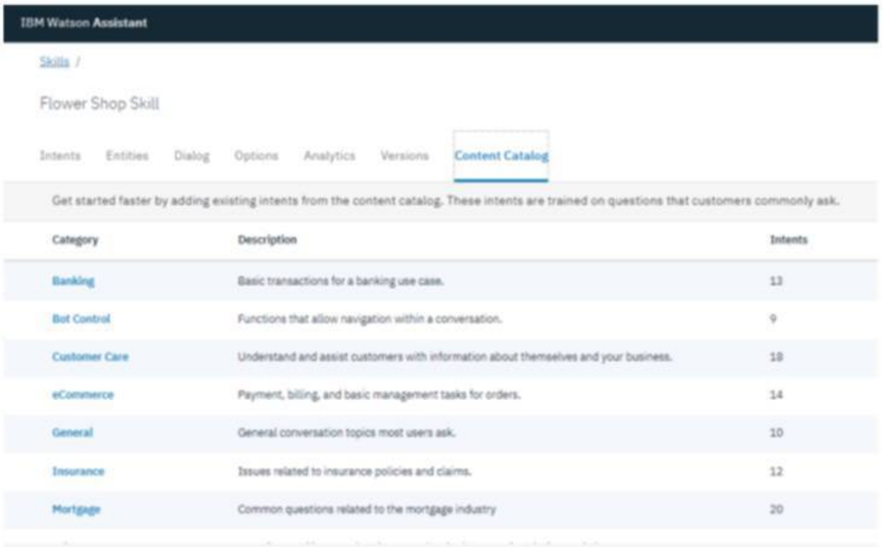
Уотсон всегда будет стараться изо всех сил сопоставить вводимые пользователем данные с существующим намерением, даже если это не идеальное совпадение.

Но если его уровень достоверности в наилучшем подходящем намерении будет очень низок (ниже 20%), Уотсон будет воспринимать входные данные как несущественные, так как они, вероятно, не имеют отношения ни к одному из наших намерений.

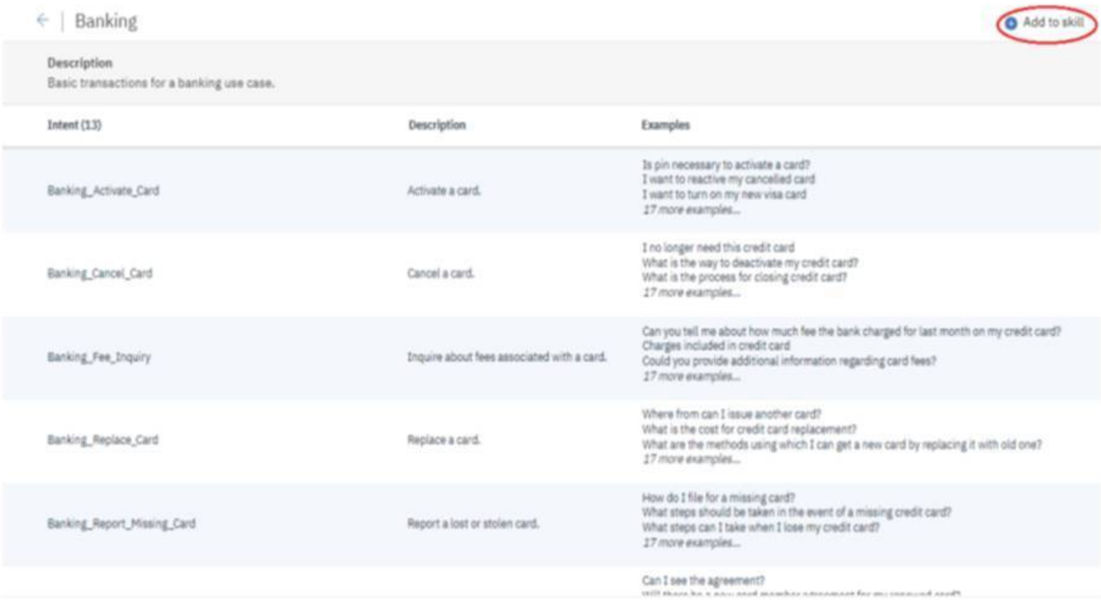
При создании диалога мы узнаем, как справляться с ситуациями, когда пользователь вводит вопрос, который не имеет значения или выходит за рамки нашего чат-бота.

И IBM предоставляет некоторые готовые намерения, которые могут иметь отношение к вашему чат-боту.

Чтобы увидеть их, нажмите Content Catalog.

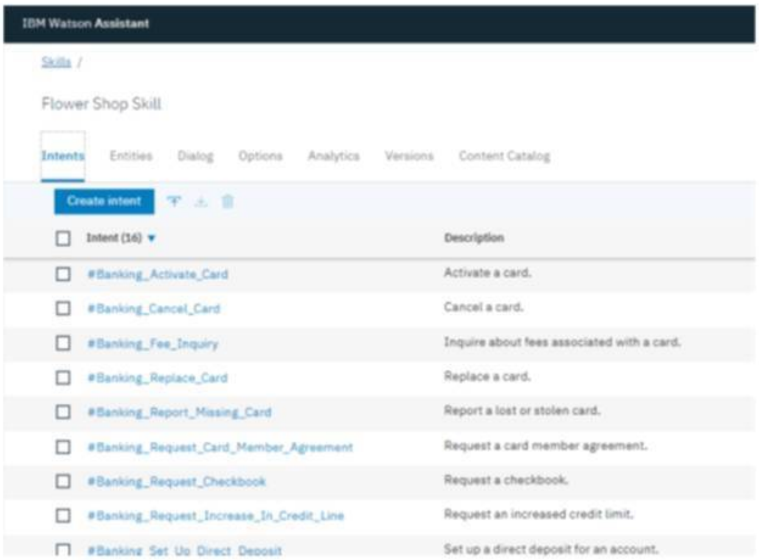


Выберите одну категорию, например, Банковское дело.



А затем нажмите кнопку Добавить к навыку.

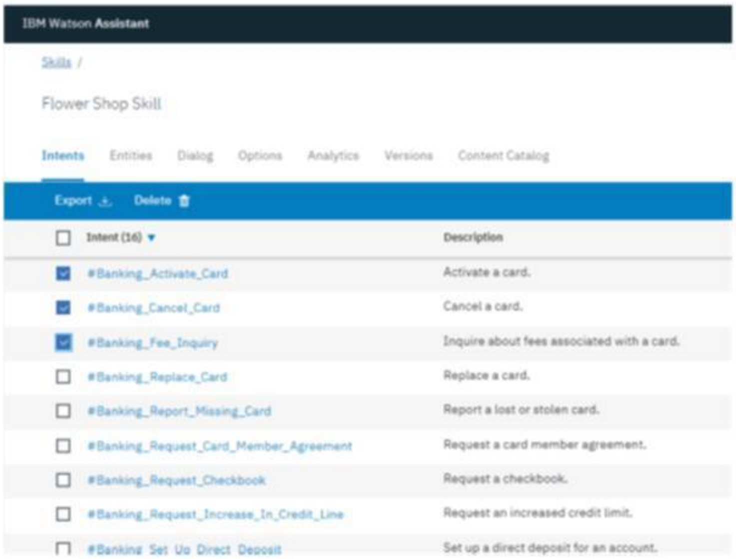
Вернитесь в раздел Intents, и вы увидите новые намерения, относящихся к запросам, которые могут возникнуть у клиентов.



Это не совсем готовый чат-бот, но это хорошее начало, где вы можете редактировать и адаптировать чат-бота.

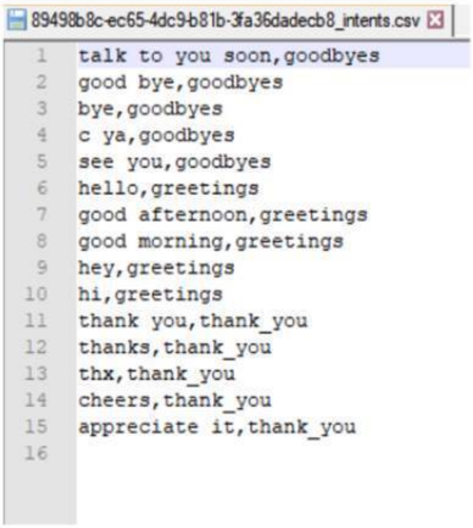
И вы можете пробовать его в панели Try it.

Здесь вы можете пометить намерения и нажать кнопку Удалить, чтобы удалить ненужные намерения.



Здесь также есть возможность экспортировать намерения, что полезно при повторном использовании намерений в разных чат-ботах.

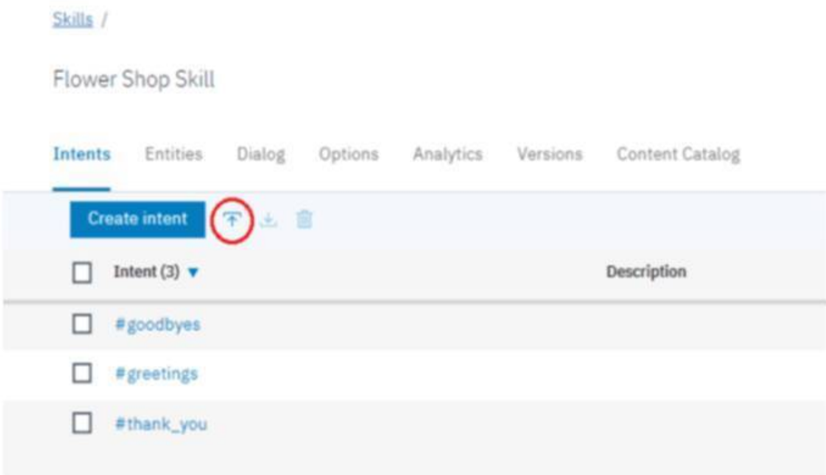
Нажмите кнопку «Экспорт», чтобы загрузить CSV-файл, содержащий наши намерения и примеры.



Откройте этот файл, чтобы посмотреть, как он выглядит.

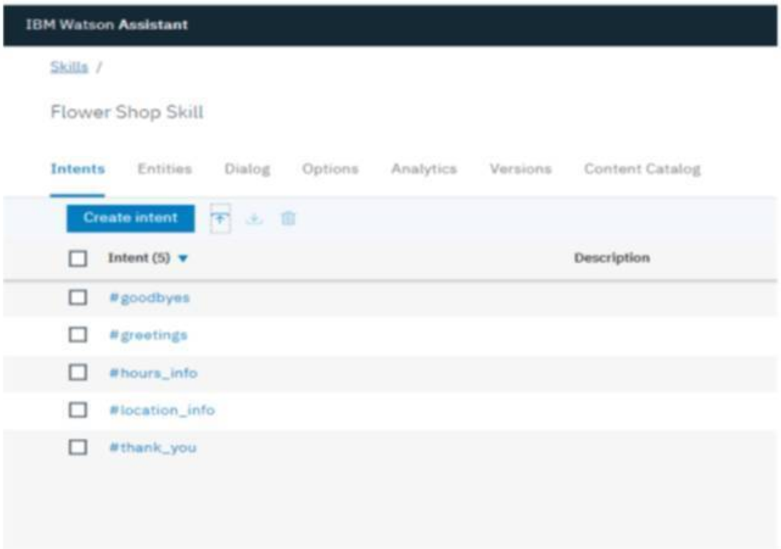
И так же, как мы экспортировали наши намерения в файл CSV, мы можем сделать наоборот и импортировать намерения из файла CSV.

Загрузите файл CSV, который прилагается к лекции.

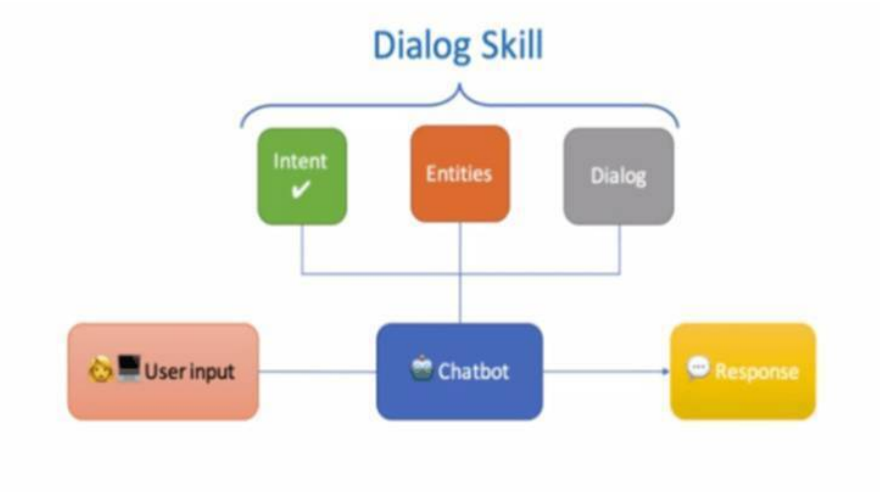


Нажмите значок «Импортировать намерения» рядом с кнопкой «Создать намерение».

Выберите файл в появившемся окне и нажмите на кнопку «Импорт».



В результате вы импортируете два новых намерения и их примеры для обучения Уотсона.



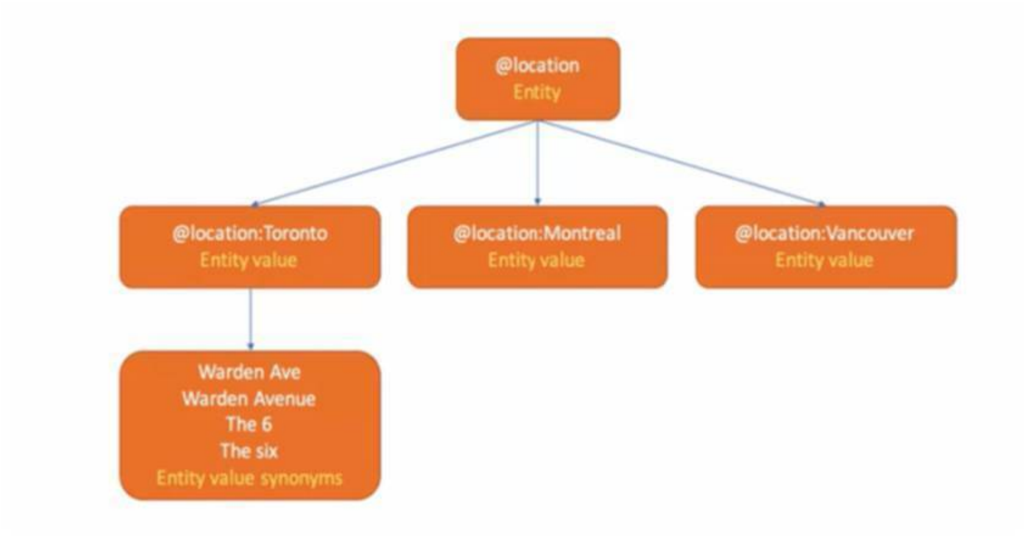
Теперь, давайте рассмотрим сущности, второй ключевой компонент диалогового навыка.

Как мы уже выяснили, намерения описывают цель пользователя.

А сущности позволяют нам фиксировать определенные значения в высказывании пользователя.

Например, без определения сущностей, вопросы «когда открыт ваш магазин в Торонто?» и «Когда открыт ваш магазин в Ванкувере?» – неразличимы для чат-бота.

Это потому, что мы не определили сущность, которая описывает эту конкретную информацию, а именно местоположение магазина.



Поэтому мы можем создать сущность местоположения.

И обратите внимание, что здесь мы используем символ @ вместо символа # для сущностей.

Затем мы можем определить несколько значений для нашей сущности, таких как Торонто, Монреаль, Ванкувер и так далее.

Одно значение для каждого местоположения магазина.

И мы также можем определить синонимы для конкретного значения сущности.

Например, представьте, что наш магазин в Торонто расположен на авеню Уорден.

И клиенты могут спросить о часах работы нашего магазина в Торонто или на авеню Уорден.

Синонимы являются необязательными, но являются полезной функцией и их следует определять, когда это возможно.

Определив сущность местоположения, теперь с помощью этих двух частей информации – намерения и сущности, мы можем предоставить соответствующий и конкретный ответ пользователю.

@location:(New York)

И обратите внимание, если в значении сущности есть пробел, мы заключаем значение в скобки.

Как и намерения, мы также можем импортировать сущности из CSV файла.

@sys-currency	Extracts currency values from user examples including the amount and the unit. (20 cents)
@sys-date	Extracts date mentions (Friday)
@sys-location BETA	The @sys-location system entity extracts place names (country, state/province, city, town, etc.) from the user's input. (Boston)
@sys-number	Extracts numbers mentioned from user examples as digits or written as numbers. (21)
@sys-percentage	Extracts amounts from user examples including the number and the % sign. (15%)
@sys-person BETA	The @sys-person system entity extracts names from the user's input. (Anna)
@sys-time	Extracts time mentions (at 10)

Помимо ввода сущностей вручную и импорта из CSV файла, существует также третий способ добавления сущностей в чат-бот.

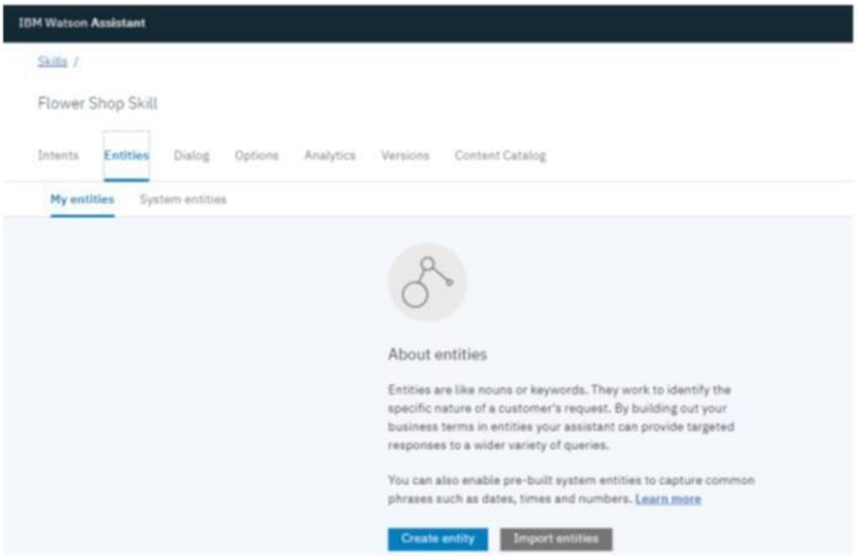
Есть предопределенные системные сущности, которые вы можете добавлять для чатбота.

@sys-currency позволяет нам определять упоминания валют в пользовательском вводе.

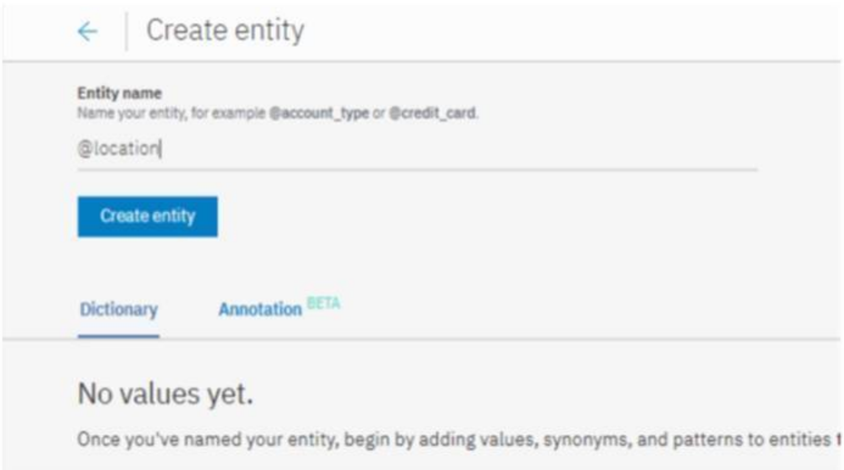
@sys-date определяет даты.

Например, пользователь сказать «следующий понедельник», а @sys-date позволит нам зафиксировать эту часть информации в качестве фактической конкретной даты.

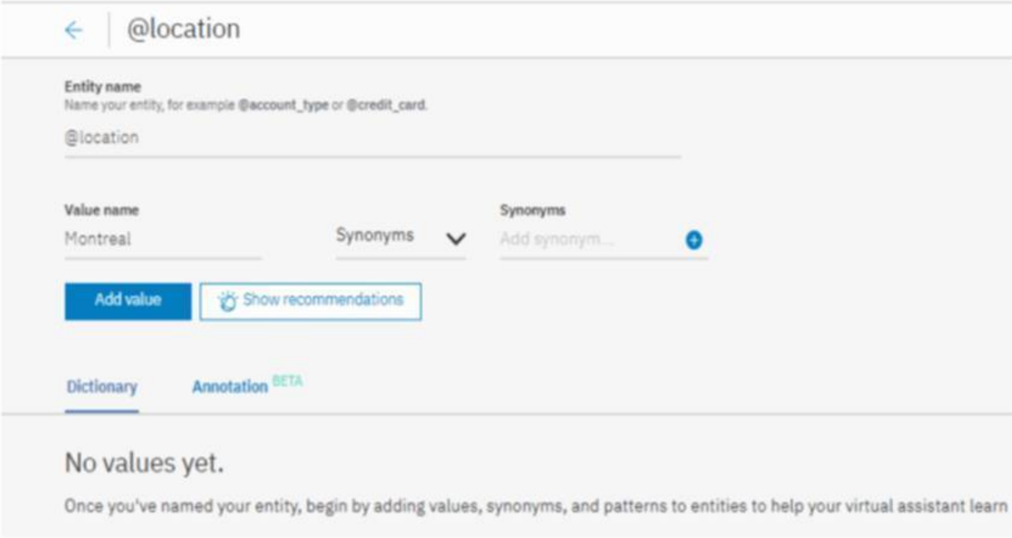
@sys-person позволяет нам обнаруживать имена людей.



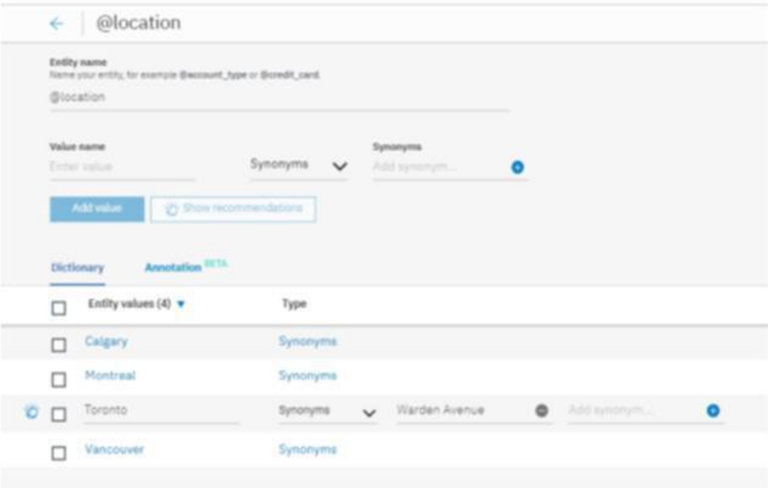
Теперь, давайте создадим сущности @location для нашего чатбота.
Откроем вкладку Entities и нажмем кнопку Create entity.



Введем имя сущности @location.
И нажмем Create entity.

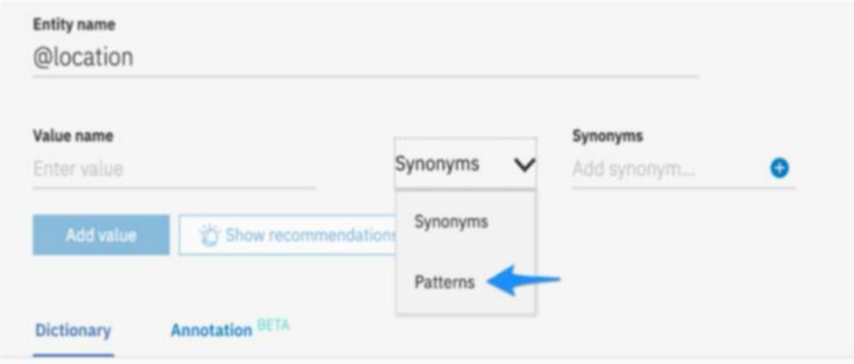


Далее мы будем вводить значения сущностей и возможные синонимы.
А затем нажимать кнопку Add value.



И вы можете добавлять синонимы для городов, если у города есть другие названия или люди ссылаются на местоположение магазина по его улице или району в городе.

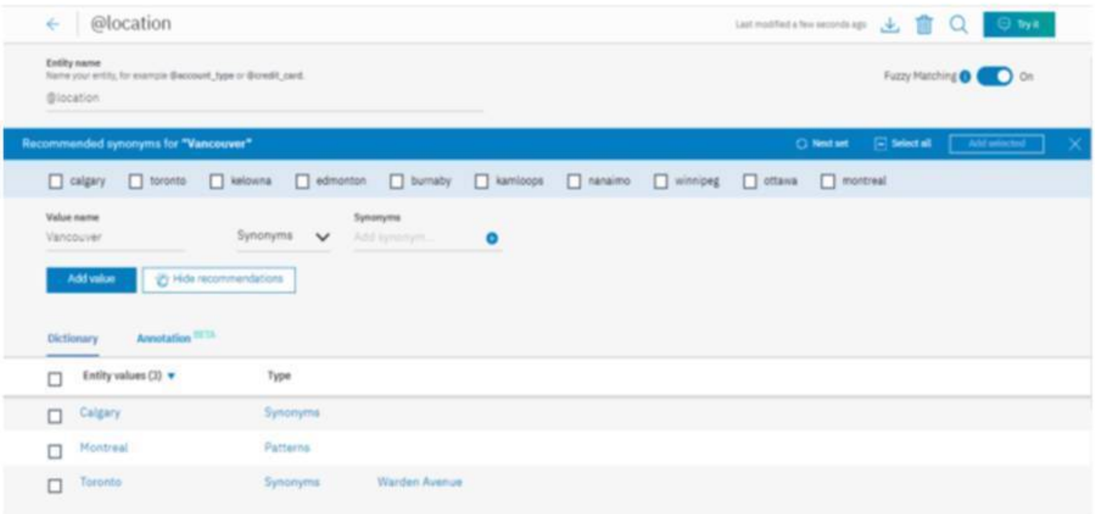
Близлежащие небольшие города также могут выступать в качестве синонимов.



Также, значения сущностей также могут иметь шаблоны, которые добавляются в раскрывающемся списке Синонимы.

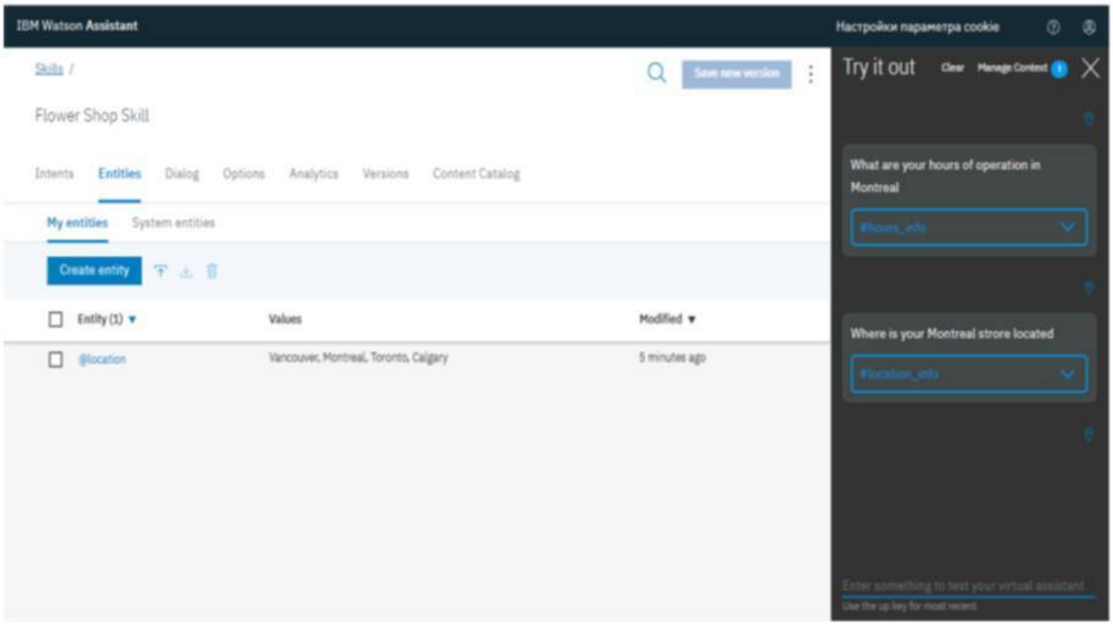
Шаблон – это расширенная функция, которая позволяет определять значение сущности не на основе конкретной строки, а на основе шаблона, такого как отформатированный номер телефона, адрес электронной почты или адрес веб-сайта.

И в любой момент вы можете нажать на значение сущности, чтобы изменить его значение или синоним.



Также вы можете нажать кнопку Показать рекомендации, чтобы выбрать синонимы из списка, предоставленного Watson.

И в конце используйте панель Try it, чтобы проверить эти значения сущностей.



Попробуйте ввести вопросы.

What are your hours of operation in Montreal.

Where is your Montreal strore located

Теперь мы можем распознать цель и города, соответствующие нашим магазинам.

Но что произойдет, если пользователь введет вопрос для Сиэтла или для Мумбаи, где у нас нет магазина?

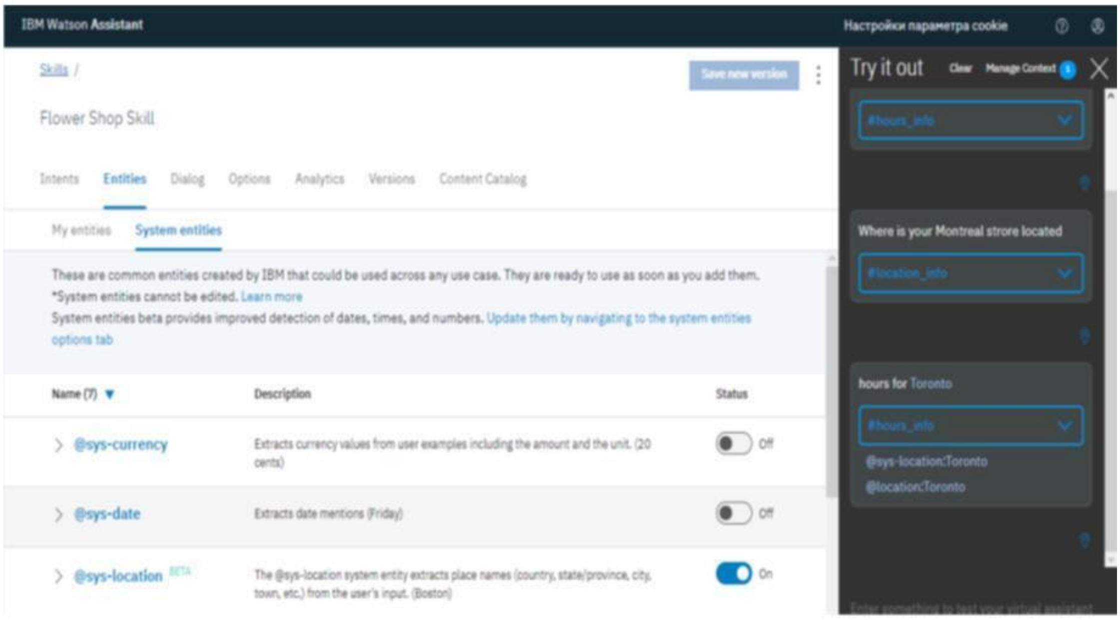
Вы заметите, что, так как у нас нет значения сущности для Сиэтла или Мумбаи, ни один из них не будет выбран в качестве значения сущности.

И мы можем структурировать нашего чат-бота для предоставления общего, информативного ответа, если не указано распознанное местоположение.

Но если мы хотим обнаружить все местоположения, чтобы предоставить более персонализированный ответ (например, «К сожалению, у нас нет магазина в Сиэтле …»), нам потребуется сущность, которая включает в себя список всех крупных городов.

И это можно легко достичь с помощью системных объектов.

Системные объекты позволяют легко обнаруживать общие специфические фрагменты информации, такие как даты, время, числа, валюты и т. д.



И среди них, существует объект @sys-location, который будет определять для нас местоположение и будет обрабатывать любой город (или штат, страну и т. д.).

И теоретически нам даже не нужна наша сущность @location, мы могли бы просто использовать @sys-location.

Хотя здесь есть два ограничения:

Вы не можете определять синонимы для городов, обнаруженных с помощью @sys-location.

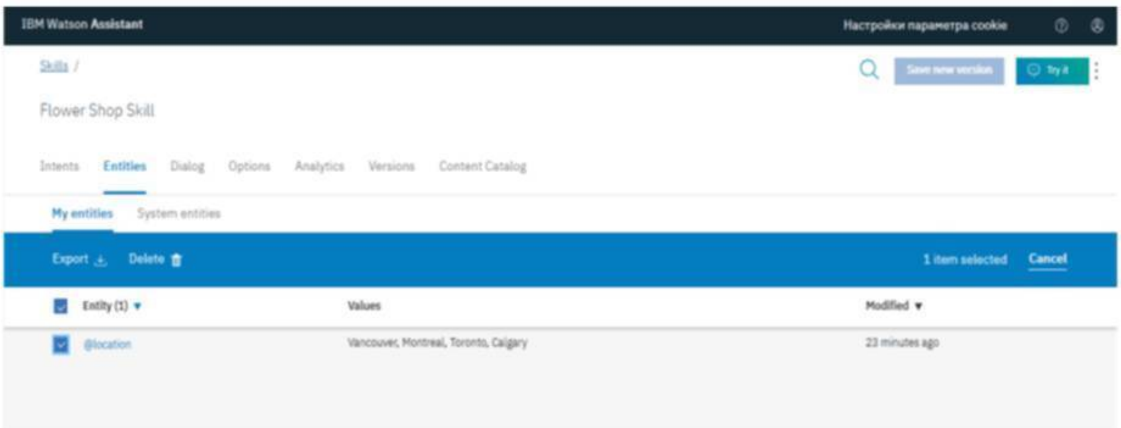
И нечеткое сопоставление в настоящее время недоступно для сущности @sys-location.

Чтобы добавить сущность @sys-location, нажмите «Системные сущности» в разделе «Сущности» вашего навыка.

И включите @sys-location.

Теперь, попробуйте ввести вопрос hours for Toronto.

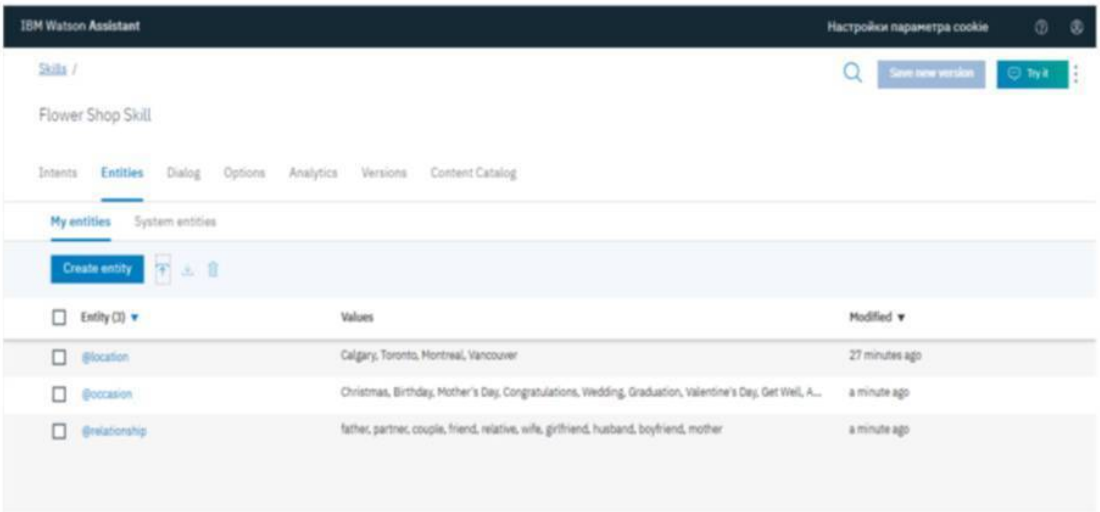
Вы заметите, что обнаружены сущности @sys-location и @location.



Импорт и экспорт сущностей с помощью файлов CSV работает очень похоже на намерения.

Когда вы выбираете одну или несколько сущностей, отметив флажки рядом с ними, вам будет предложено экспортировать их в CSV файл.

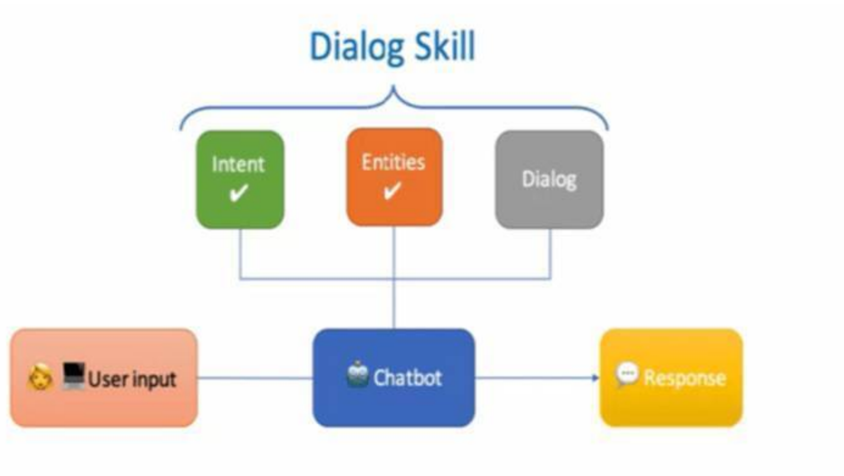
Кроме того, вы можете импортировать сущности, нажав кнопку «Импортировать» рядом с «Create entity».



Загрузите файл CSV с двумя новыми сущностями, который прилагается к лекции.

И импортируйте его.

После успешной загрузки и импорта сущностей вы должны увидеть их в списке.



Теперь, давайте, наконец, рассмотрим третий компонент диалогового навыка.

А именно сам диалог.

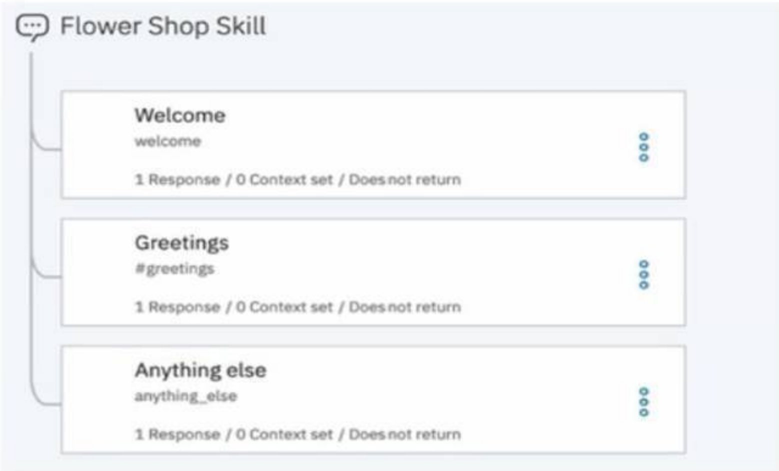
Компонент диалога позволяет нам выдавать ответ пользователю на основе его намерения и специфики его запроса, которую мы определяем с помощью сущности.

Наш чат-бот может обнаруживать и классифицировать вводимые пользователем данные, но он еще пока не может ответить пользователю.

Например, когда пользователь приветствует нас, мы можем захотеть ответить «Привет! Могу я чем-нибудь помочь?".

То же самое верно и для более сложных запросов.

Мы должны использовать точно настроенную классификацию, которую нам дают намерения и сущности, чтобы обеспечить надлежащий и точный ответ пользователю.



Диалог – это дерево узлов, и каждый узел обычно обрабатывает один конкретный сценарий.

Например, здесь у нас есть три узла.

Первый узел – это Welcome, который обрабатывает приглашение чатбота.

Другими словами, наше приветствие при первом появлении пользователя.

Затем у нас есть узел Greetings для ответа на приветствие пользователя.

И, наконец, у нас есть специальный резервный узел, который будет уведомлять пользователя о том, что чатбот не совсем уверен в том, что спрашивает пользователь.

Welcome и Anything else – это два узла по умолчанию, созданные для вас при первом создании диалога для вашего чат-бота.

Любой узел, который вы определите, будет находиться между этими двумя узлами.

Каждому такому узлу дается имя.

Далее у нас есть условие, которое определяет, когда узел должен быть запущен.

В этом примере, узел будет запущен, если в пользовательском вводе обнаружено намерение #greetings.

Условие может быть как простым, так и сложным логическим выражением.

В большинстве случаев это условие будет проверять намерение, сущность или их комбинацию.

Затем у нас есть блок ответа, где мы указываем, что ответить пользователю.

В этом случае просто: «Привет. Чем я могу вам помочь?»

Этот блок ответа имеет много вариантов.

Ответ может быть изображением или некоторыми параметрами, которые может выбрать пользователь.

Мы можем предоставить несколько вариантов ответа, а затем решить, следует ли их выдавать по порядку при каждом запуске этого узла, или случайным образом из списка, который мы предоставляем.

Мы даже можем прикрепить условие к каждому отдельному ответу в пределах одного и того же узла.

Наконец, мы можем указать, что происходит после того, как мы выдали ответ.

Как правило, мы просто ждем, когда пользователь скажет что-то еще, и это действие по умолчанию.

Однако доступны и другие параметры, в том числе возможность перехода к другим узлам в диалоге.

Запуск узлов выполняется сверху вниз в диалоге.

Поэтому, когда пользователь отправляет свое высказывание, будет оцениваться на выполнение первый узел.

Если условие не выполнено, мы перейдем ко второму узлу.

Если и этот узел не соответствует критериям, мы рассмотрим третий узел и так далее.

И мы остановимся на первом узле, условие которого соответствует пользовательскому вводу.

Вот почему так важно, чтобы резервный узел Anything else находился в самом низу дерева узлов.

Этот узел имеет специальное условие, которое всегда выполняется, даже если все узлы над ним не выполняются.

Это гарантирует, что у нас всегда есть ответ для пользователя, например: «Я не понимаю, не могли бы вы перефразировать?».

По этой же причине у нас есть узел Welcome в самом вверху, приветствующий пользователя.

У этого узла есть специальное условие приветствия, которое выполняется только в начале разговора с пользователем.

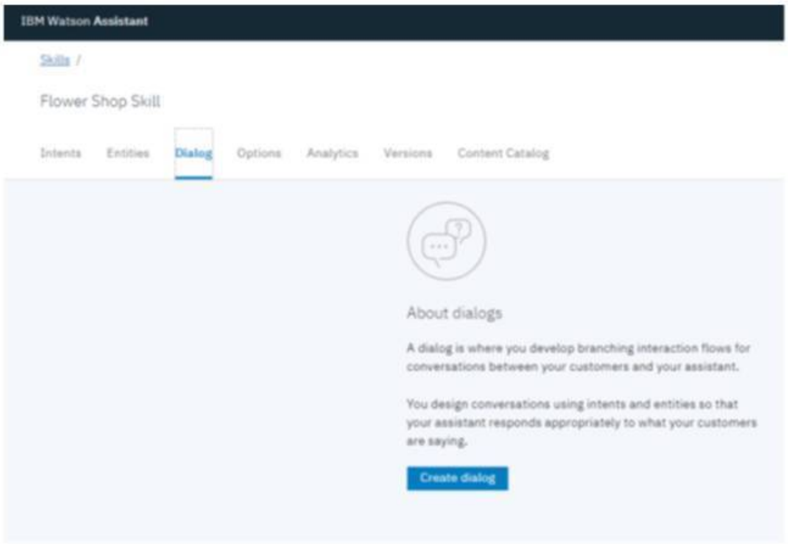
Размещая его наверху, мы гарантируем, что что этот узел выполняется до любого другого узла.

И тот факт, что условие выполняется только один раз, гарантирует, что мы не будем приветствовать пользователя каждый раз, при его вводе.

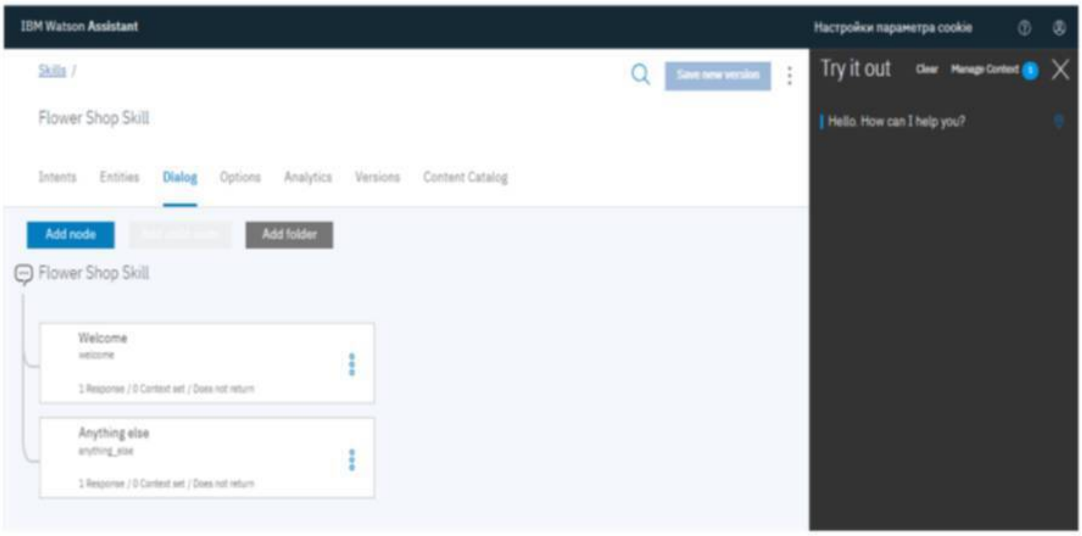
Узлы также могут иметь дочерние узлы, и в сложных чат-ботах бывают ситуации, когда использование этой функции позволяет нам создать более детализированный чат-бот.

Дочерние узлы рассматриваются для выполнения только после того, как был выполнен родительский узел, если какой-либо другой узел в диалоге явно не перешел на данный дочерний узел.

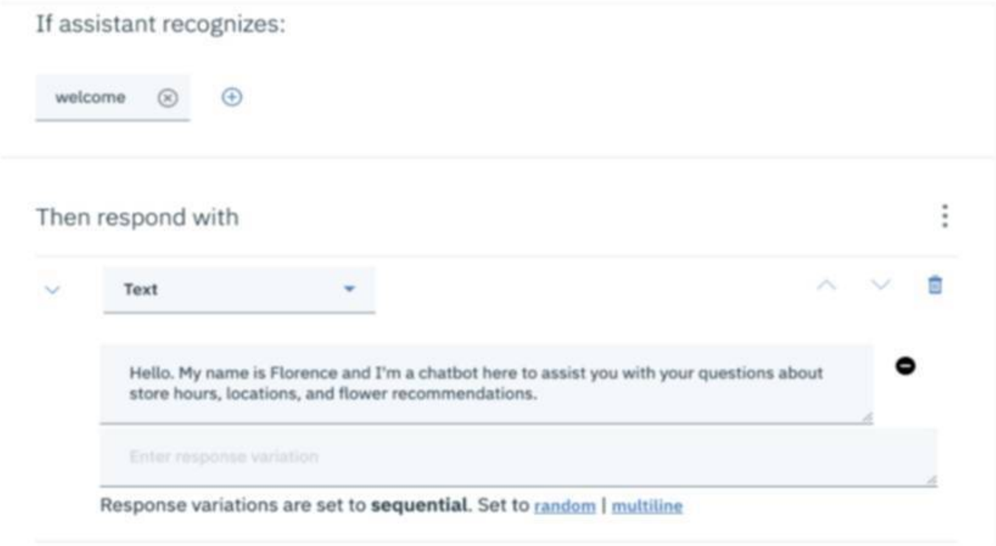
И дочерние узлы также выполняются сверху вниз, и мы останавливаемся на выполнении первого дочернего узла, который соответствует условию.



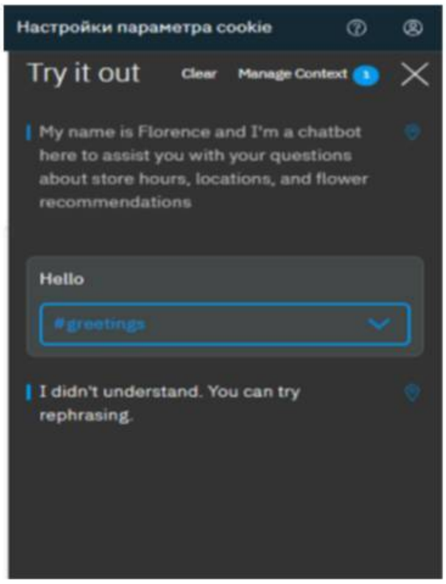
Теперь, давайте создадим диалог для нашего чат-бота.
Откроем вкладку Диалог навыка.
И нажмем кнопку Create Dialog.



И здесь у нас уже есть два узла Welcome и Anything else.
Открыв панель Try it мы сразу увидим приветствие.



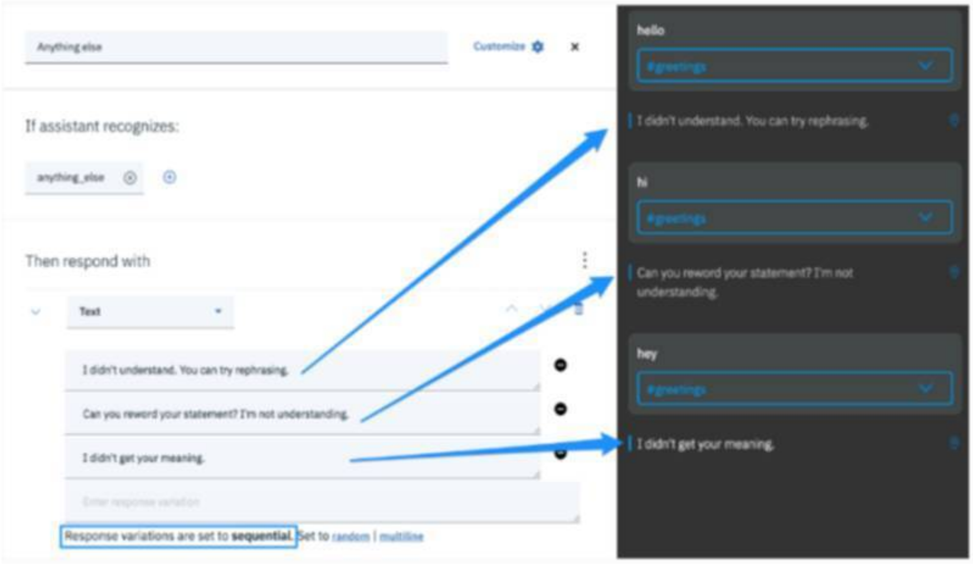
Открыв узел Welcome, мы можем изменить это приветствие.



Теперь, если мы ответим «Привет», Уотсон правильно распознал намерение #greetings, но у него нет нужного узла для обработки приветствий, поэтому выполнился резервный узел Anything else.

И стоит отметить, что, если вы будете вводить приветствие или что-либо еще в несколько раз, вы каждый раз будете получать разные ответы.

Причина этого заключается в том, что узел Anything else по умолчанию имеет три варианта ответа.

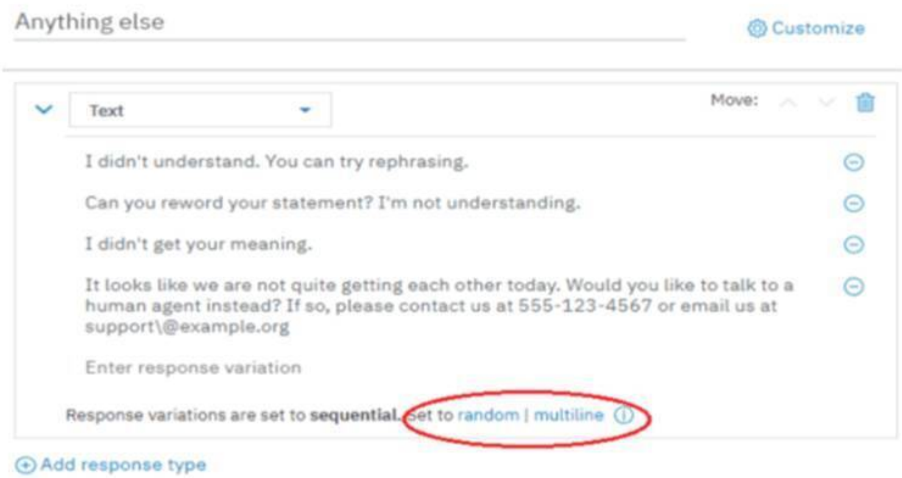


Это сделано, чтобы чатбот выглядел умнее.

Для узлов, на которые вряд ли удастся попасть несколько раз в течение разговора, можно иметь один ответ без изменений.

В любом другом случае наличие нескольких вариаций делает чатбота умнее.

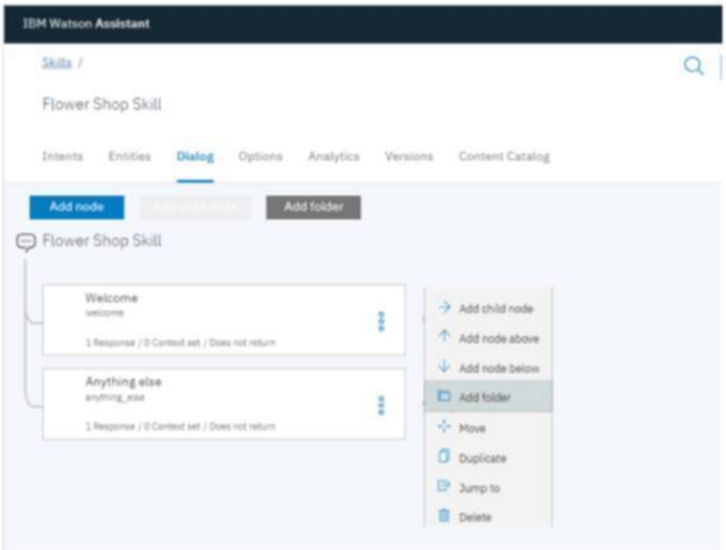
Давайте добавим четвертый вариант к узлу Anything else со следующим текстом: «Похоже, мы сегодня друг друга не понимаем». Хотели бы вы вместо этого поговорить с человеческом? Если это так, пожалуйста, свяжитесь с нами по телефону или напишите нам по адресу support\@example.org.



Знак \ перед @ необходим для отображения специального символа собака.

И так как у нас этот набор ответов в последовательном режиме, мы гарантируем, что этот ответ будет только после того, как мы не смогли понять пользователя четыре раза.

Если бы этот режим был случайным, мы могли бы обострить ситуацию уже в первый раз.



У нас есть три намерения для чата: #greetings, #thank_you и #goodbyes.

Теперь нам нужно иметь узлы, которые будут выполнены при обнаружении этих намерений.

И здесь у нас есть несколько возможных стратегий.

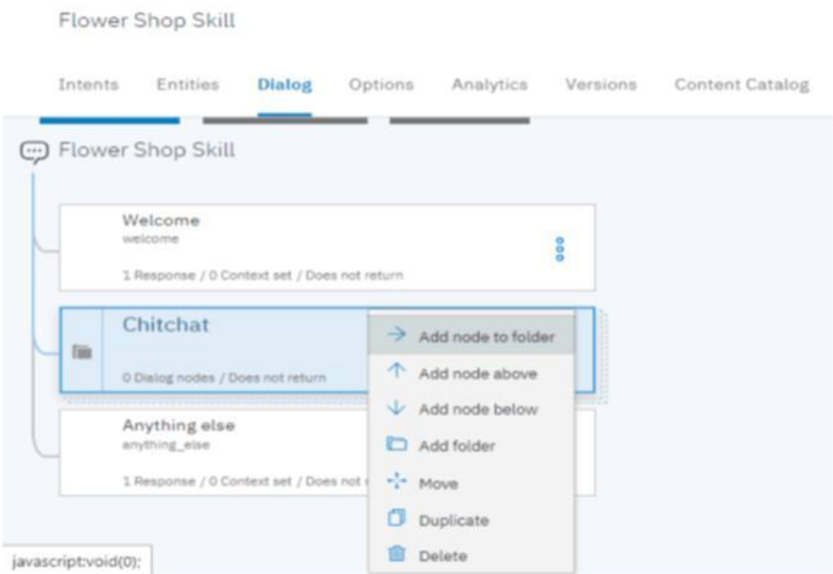
Мы могли бы создать три узла, по одному для каждого из этих намерений.

Это самый распространенный и простой подход.

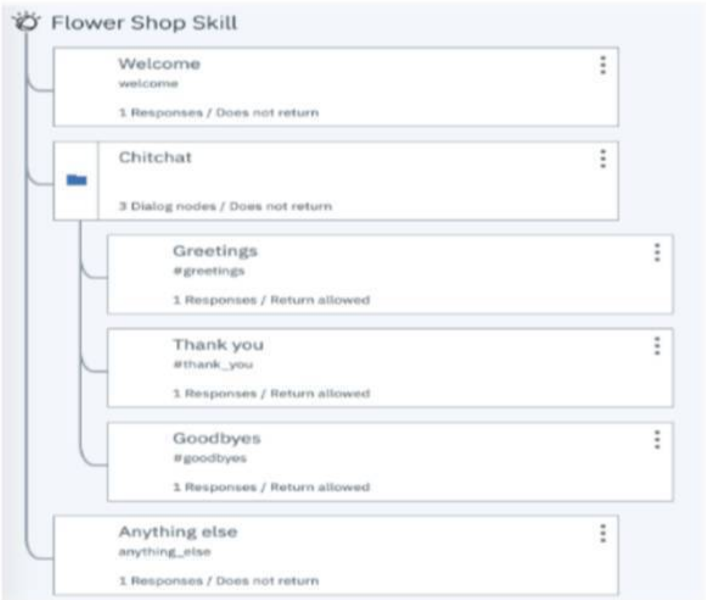
Другим вариантом будет создание одного узла, который использует несколько ответов, прикрепляя условие к каждому ответу.

Мы будем использовать простой подход.

И мы создадим папку для чата, нажав на три вертикальные точки на узле Welcom, и выбрав команду Add folder.



Далее нажмем на три вертикальные точки на созданной папке и выберем команду Add node to folder.

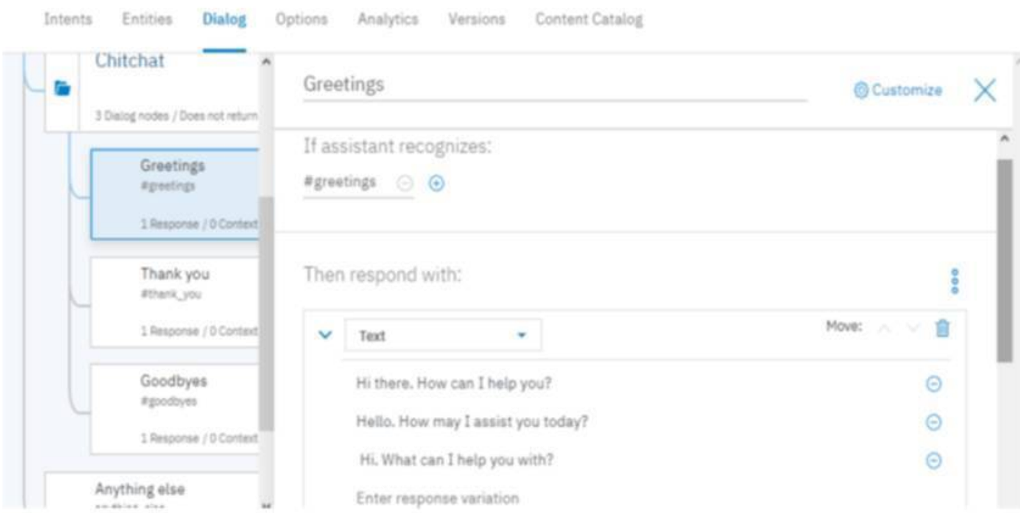


Таким образом мы создадим в папке три узла, для каждого из них определяя соответствующее намерение.

Стоит отметить, что вы можете сделать условие узла сложным.

Например, вы можете использовать оператор OR или AND, чтобы составить сложное условие из нескольких намерений.

Например, #greetings OR #goodbyes.



И для каждого узла, мы введем несколько ответов пользователю.

Например, «Привет. Чем я могу вам помочь?», «Здравствуйте. Как я могу помочь вам?», и так далее.

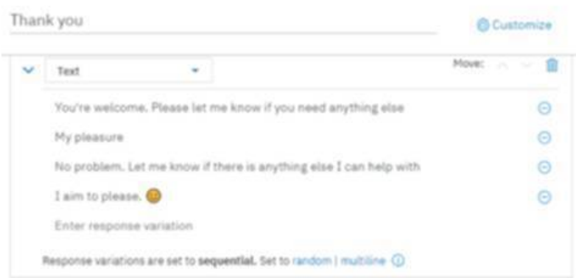
И вы можете оставить режим ответа последовательным или установить его случайным, если захотите.

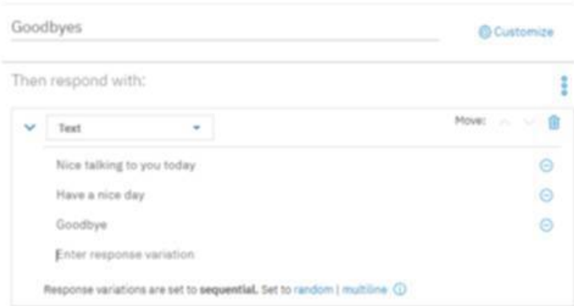
Третий вариант, многострочный, здесь не применим, так как он обеспечивает ответ несколькими строками, используя каждый введенный ответ, как отдельную строку.

И раздел And finally в нижней части узла определяет, что происходит после того, как этот узел был выполнен и пользователю был дан ответ.

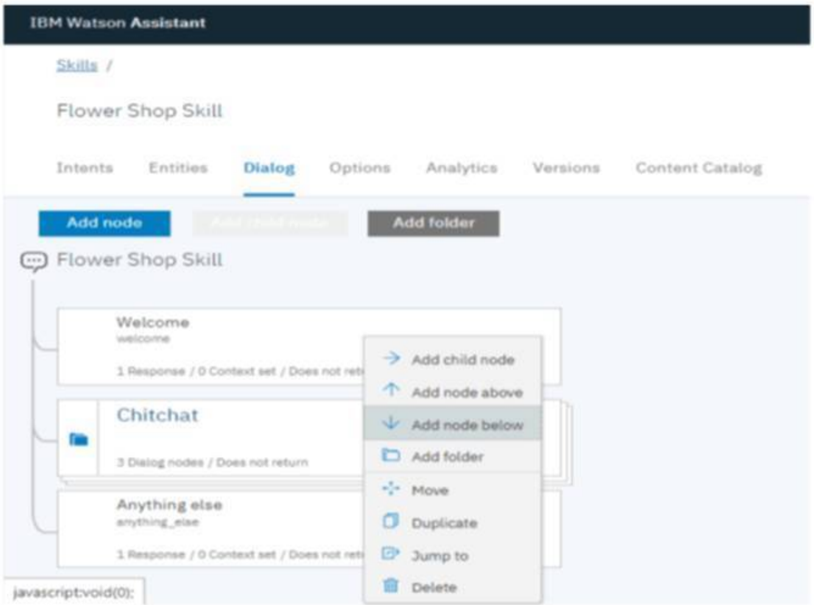
По умолчанию, после того как мы ответили пользователю, мы ожидаем, что он введет еще вопрос.

Теперь, вы можете попробовать панель Try it.





Тоже самое сделайте для других узлов.
Для узла #thank_you. введите что-то вроде:
Пожалуйста, дайте мне знать, если вам нужно что-нибудь еще.
С удовольствием.
Нет проблем. Дайте мне знать, если есть что-то еще, с чем я могу помочь.
Для узла #goodbyes.
Приятно с вами поговорить.
Хорошего дня.
Прощай.

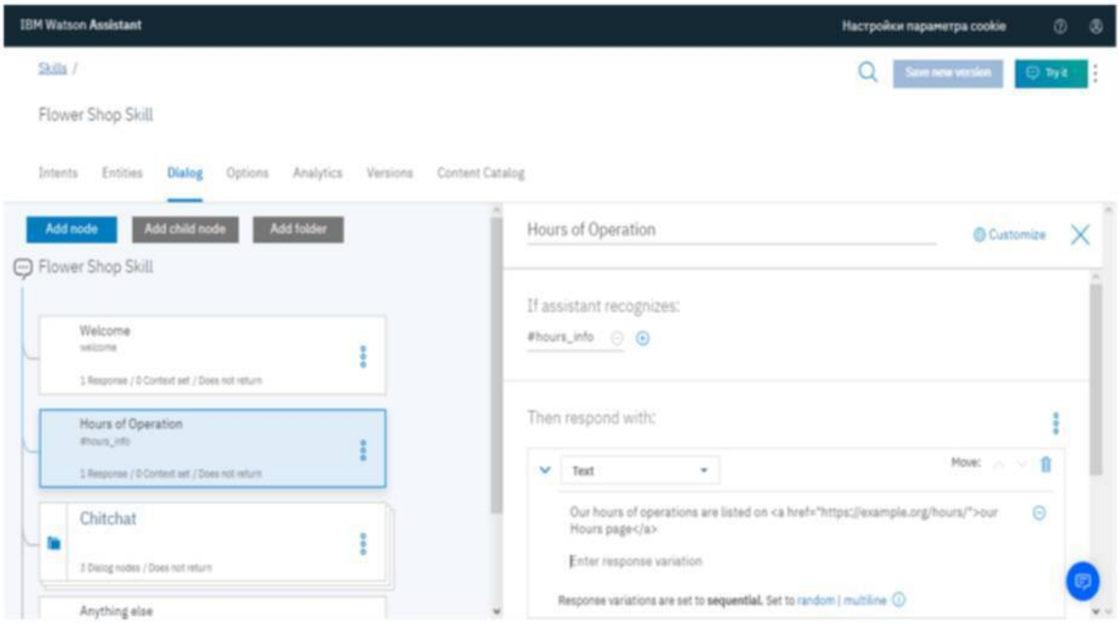


Теперь, ранее мы определили намерение #hours_info для людей, спрашивающих о часах работы и адресах нашей сети магазинов, и даже создали сущность @location, чтобы иметь возможность предоставлять ответы, специфичные для конкретного места.

Теперь, нам нужно создать узлы диалога для ответа клиенту.

И мы начнем с создания узла для запроса часов работы.

На узле Welcom нажмем на три вертикальные точки и выберем команду Add node below, которая создаст пустой узел ниже первого узла в диалоге.



Здесь введем имя узла и укажем намерение #hours_info в качестве условия узла.

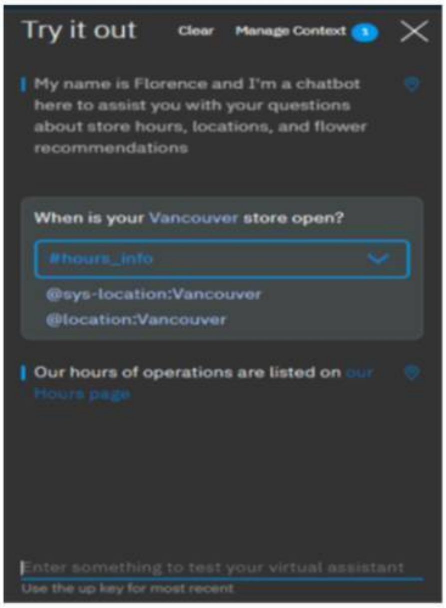
Это обеспечит выполнение этого узла, когда пользователь запросит часы работы магазина.

В качестве ответа введите:

Наши часы работы указаны на нашей странице.

И обратите внимание, что здесь используется HTML-код в ответе.

Затем перейдите в панель Try it и проверьте, работает ли узел, спросив:



When is your Vancouver store open?

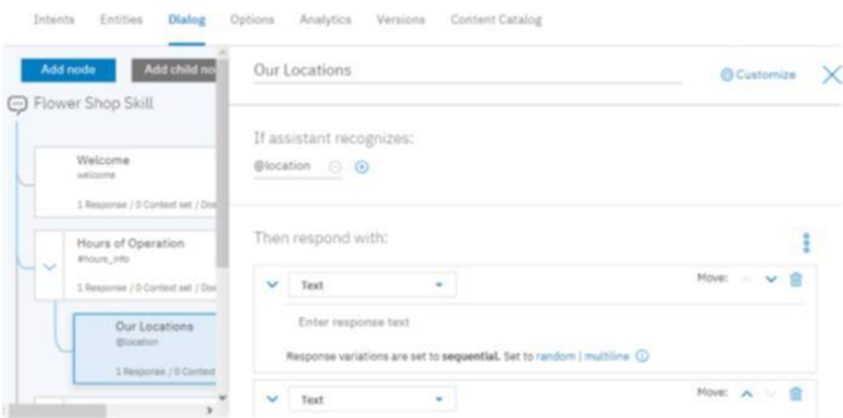
Когда ваш магазин в Ванкувере открыт?

Теперь, мы можем улучшить этот ответ, рассмотрев три возможных сценария.

Один, когда указано местоположение, второй, когда указан город, но у нас нет местоположения, и третий, когда пользователь просто запрашивает часы работы в целом, не указывая город.

Это классический вариант использования дочерних узлов.

Мы будем использовать наш текущий узел для захвата запроса часов работы, а затем перейдем к дочерним узлам, чтобы решить, как обрабатывать запрос на основе конкретной информации о местоположении, которая была предоставлена.



Поэтому удалим ответ из нашего узла «Часы работы», щелкнув значок удалить.

Мы это сделаем, потому что не хотим, чтобы этот родительский узел давал ответ.

Так как мы позволим дочерним узлам решать, каков правильный ответ.

Далее на узле «Часы работы», нажмем «Добавить дочерний узел».

Это создает первый дочерний узел.

Мы будем использовать его для случая, когда пользователь предоставит нам город, в котором у нас есть цветочный магазин.

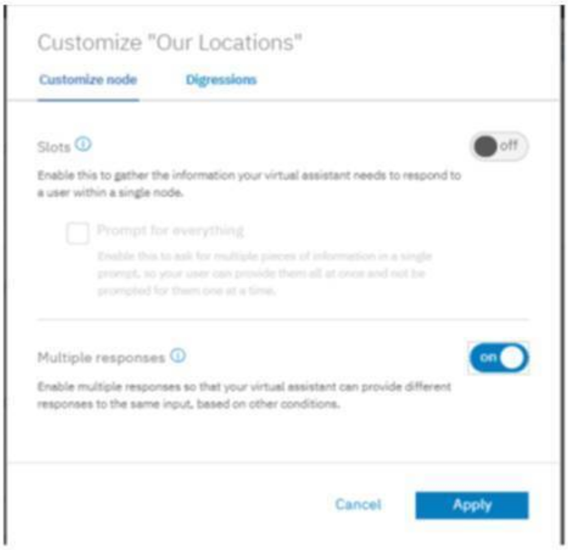
Мы назовем этот узел Наши места.

И мы установим условие @location с оператором any для выполнения этого узла.

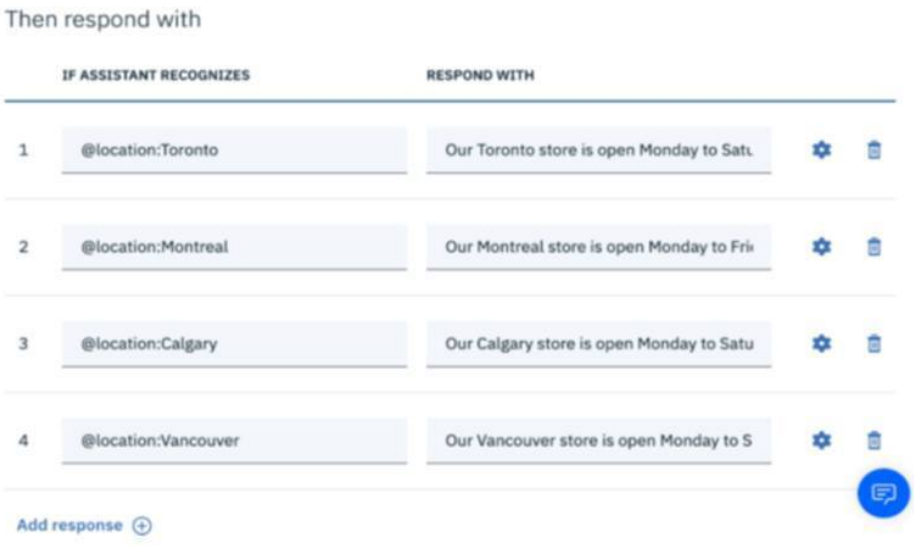
Это означает, что для пользовательского ввода будут выполняться два условия: намерение #hours_info и ввод будет содержать сущность @location.

И нам нужен способ предложить разные ответы для разных городов, поэтому мы включим опцию Multiple conditioned responses.

Для этого мы нажмем вверху Customize.



И
включим
Multiple responses
и
нажмем
Apply.



Теперь у нас есть возможность прикрепить условие к каждому ответу.

Здесь мы создадим серию ответов, по одному для каждого города.

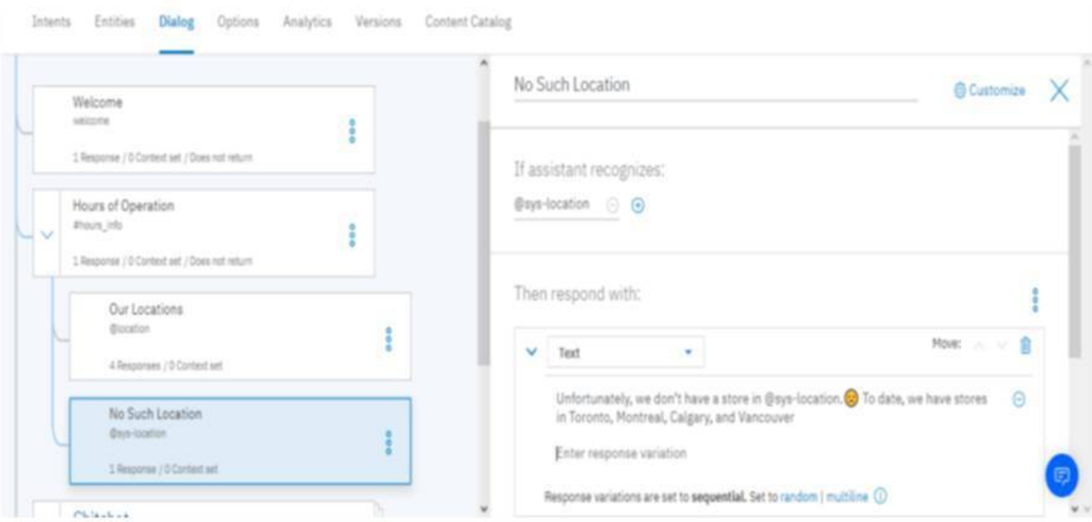
Например, для условия @location: Toronto наш ответ будет -

Наш магазин в Торонто открыт с понедельника по субботу с 9 утра до 18.00, кроме выходных.

Стоит отметить, что, если часы работы одинаковые для всех местоположений, мы могли бы просто включить @location в наш ответ.

Например, наш магазин @location открыт с понедельника по субботу с 9 до 18 часов, кроме выходных.

Это автоматически выведет обнаруженное значение сущности обратно пользователю в ответе.



Теперь, создадим дочерний узел «Нет такого местоположения».

Это случай, когда пользователь указывает город, но у нас нет магазина в этом городе.

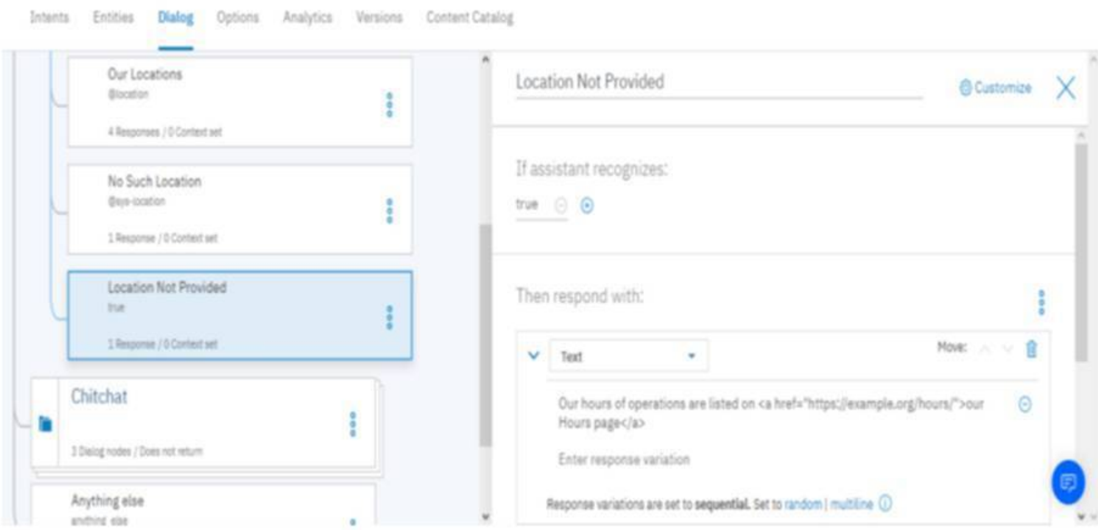
Здесь укажем сущность @sys-location с оператором any в качестве условия для этого узла.

В качестве ответа введем -

К сожалению, у нас нет магазина в @sys-location. На сегодняшний день у нас есть магазины в Торонто, Монреале, Калгари и Ванкувере.

И обратите внимание, что @sys-location будет вставлять местоположения, однако этот узел никогда не будет выполняться для наших собственных местоположений, так как он идет после узла Our Locations.

При организации вашего диалога всегда размещайте узлы с определенными условиями вверху и узлы с более общими условиями внизу.



Теперь, создадим дочерний узел «Расположение не указано» для обработки случая, когда пользователь не указал местоположение.

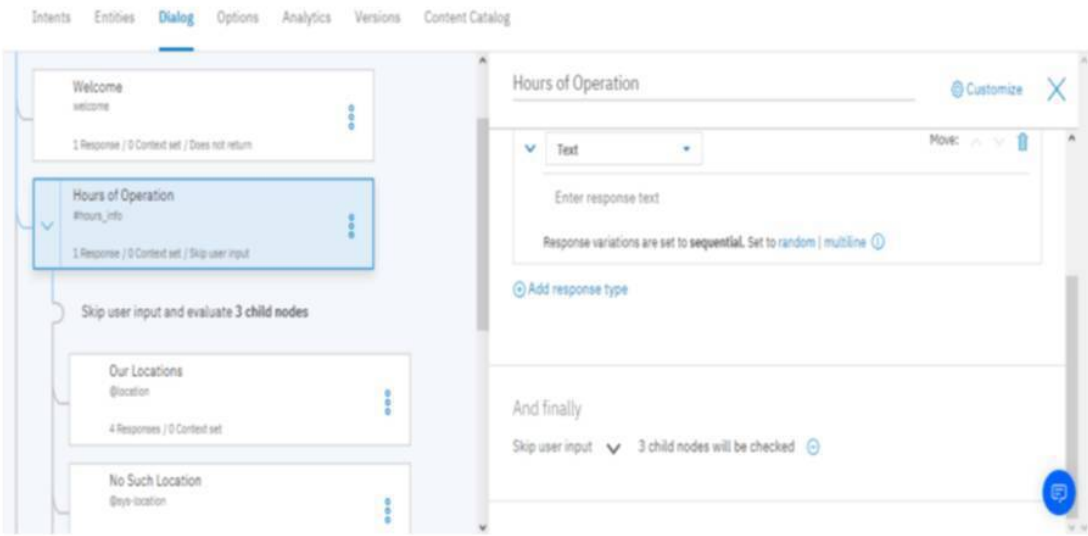
И установим условие для этого узла true.

И так как условие для этого узла имеет значение true, этот узел будет выполнен автоматически если не будут выполнены два других узла.

Если бы мы оставили условие пустым, мы получили бы ошибку, потому что ни один дочерний узел не мог бы соответствовать запросу пользователя без местоположения.

И здесь нам нужен общий ответ, поэтому наш ответ будет -

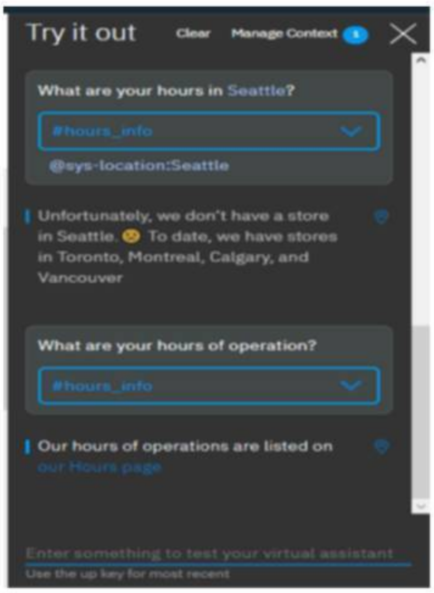
Наши часы работы перечислены на нашей странице.



Прежде чем мы сможем все это протестировать, нам необходимо убедиться, что родительский узел передает управление дочерним узлам.

Для этого откроем родительский узел, и в разделе And finally изменим Ожидание ввода пользователя на пропустить ввод пользователя.

Это передаст выполнение дочерним узлам, которые мы только что создали.



Теперь, откроем панель Try it и попробуем вводы:

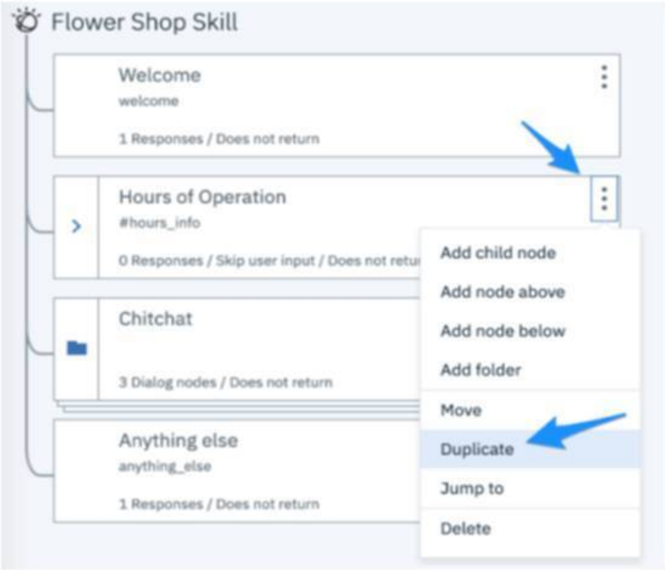
Какие у вас часы работы в Торонто?

Какие часы работы в Калгари?

Какие часы работы в Сиэтле?

Какие у вас часы работы?

И вы должны увидеть правильный ответ для каждого из этих вводов.



Теперь нам нужно обработать запросы адреса местоположения.

Для этого мы продублируем часы работы и в новых родительских и дочерних узлах для обработки адреса запроса потребуется изменить условие на #location_info вместо #hours_info и изменить ответы с часов работы на адреса.

Например, Our Toronto store is located at 123 Warden Avenue.

Our store locations are listed on our site on the stores page.

Попробуйте ввести в панели Try it -

hello

where are you stores located?

what are your hours of operations in Montreal?

thank you

bye

Технически говоря, нам не нужны дочерние узлы для обработки сценариев, которые мы реализовали.

Мы могли бы просто добавить несколько условных ответов в родительские узлы для каждого из городов, все в одном узле.

Однако здесь мы увидели, как работать с дочерними узлами.

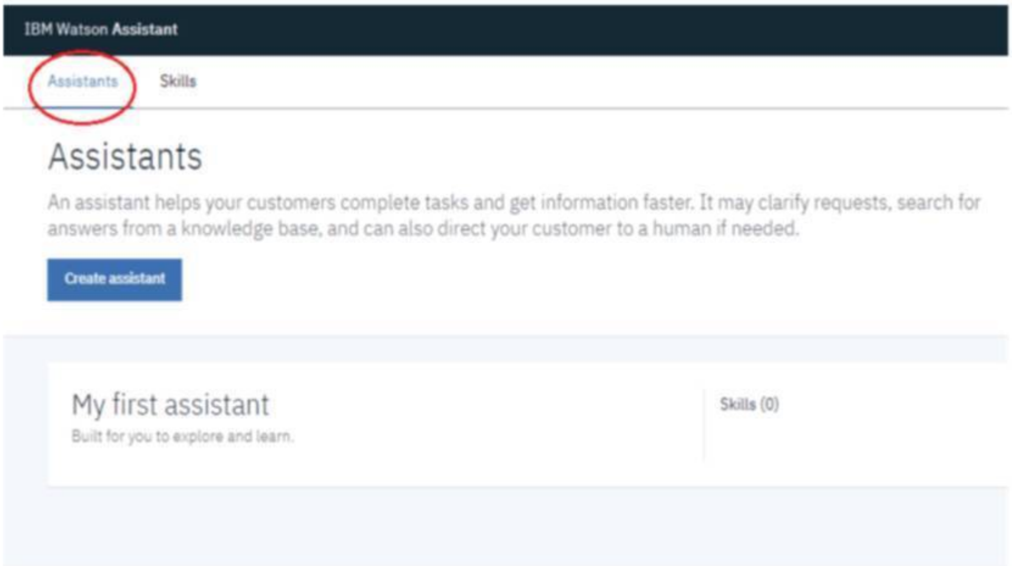
И в любом случае неплохо иметь выделенный дочерний узел для обработки запроса пользователя.

В некоторых сложных чат-ботах у вас могут быть дочерние узлы, которые имеют свои собственные дочерние узлы.

Таким образом, мы создали простой, но работающий чатбот.

Проблема в том, что в настоящее время он доступен только в панели Try it после входа в систему.

И нам нужно развернуть нашего чат-бота где-нибудь на сайте.



В панели IBM Watson Assistant есть вкладка помощников Assistants.

И на самом деле, помощник – это и есть чатбот, который может иметь один или несколько навыков Skill.

И мы разработали навык, который позволяет чат-боту понимать и отвечать пользователю.

Теперь нам нужно создать помощника и связать с ним наш навык диалога.

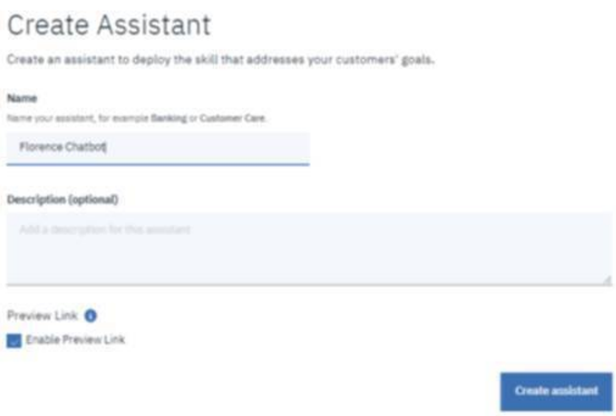
Это позволит развернуть чат-бот по различным каналам, включая сайты WordPress, Facebook Messenger, и так далее.

И при создании помощника вам предоставляется возможность включить ссылку предварительного просмотра, которая дает вам страницу, на которой можно опробовать чат-бот.

Виджет на этой странице работает, отправляя пользовательский ввод в созданную вами службу Watson Assistant, получая ответ от вашего навыка и представляя его пользователю.

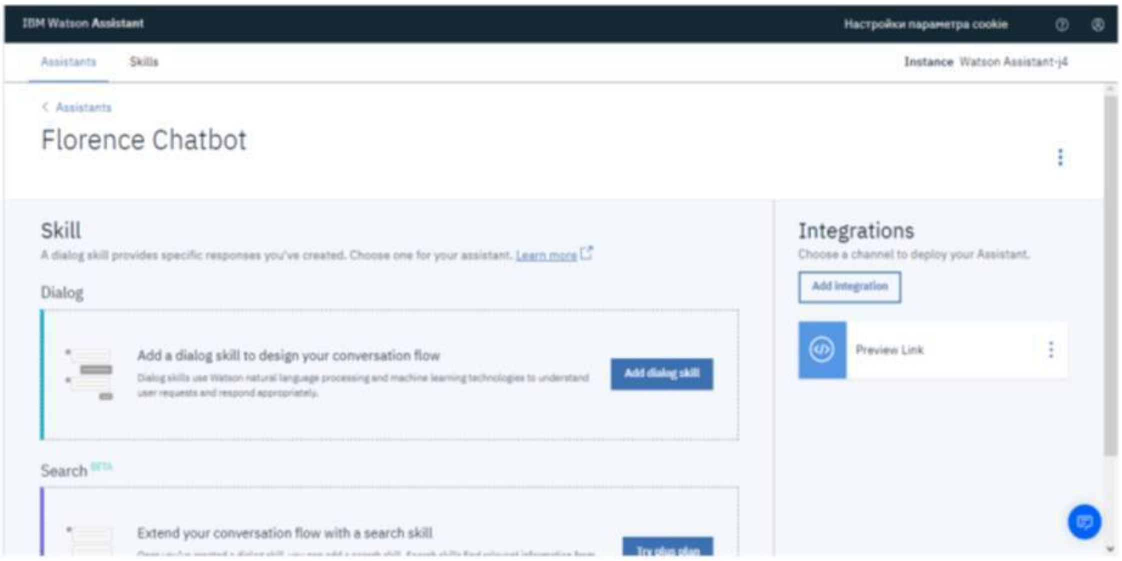
Но для начала мы создадим помощника и свяжем его с навыком.

Для этого нажмем кнопку «Create assistant».



Здесь вы введете имя (например, Florence Chatbot), необязательное описание и укажите, хотите ли вы ссылку для предварительного просмотра.

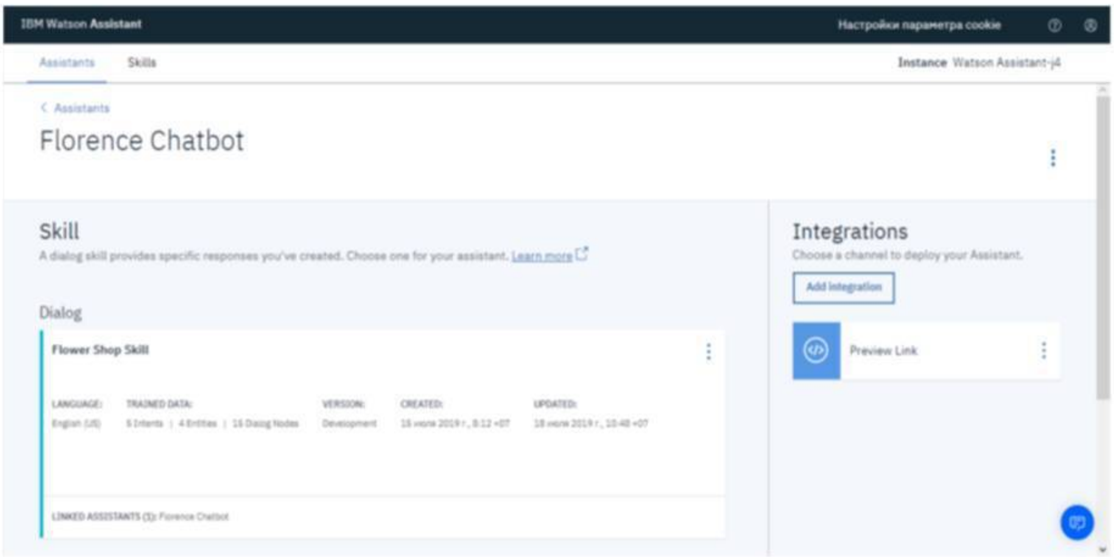
И нажмите кнопку Create assistant, чтобы создать помощника.



Здесь вы добавите навык диалога к помощнику, которого вы только что создали.

Нажмем «Добавить навык диалога» и затем выберем «Добавить существующий навык».

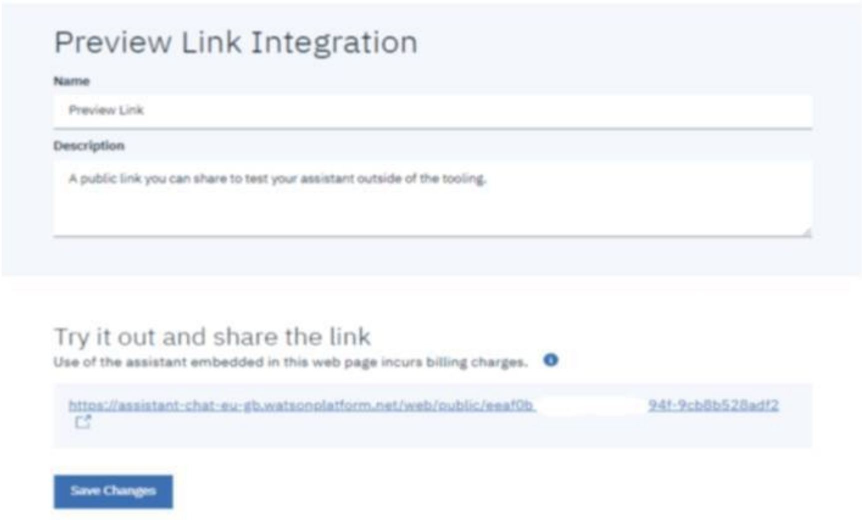
Затем выберем навык «Цветочный магазин», нажав на него.



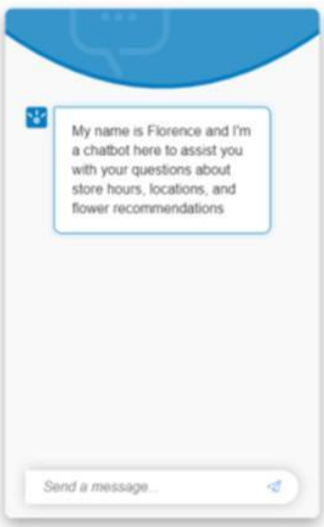
Теперь, помощник Florence Chatbot связан с навыком цветочного магазина.

И ваш помощник чат-бота теперь готов к развертыванию.

Вы можете открыть предварительный просмотр, нажав кнопку «Просмотр ссылки» в разделе «Интеграции».



Откроется страница, содержащая ссылку.
И нажмите на эту ссылку, чтобы увидеть своего чат-бота в действии.



С помощью этой ссылки можно опробовать ваш чат-бот.

И имейте в виду, что каждый раз, когда кто-то отправляет сообщение в чат-бот, выполняется один вызов API, и он учитывается в бесплатной квоте 10 000 вызовов API в месяц.

Конец ознакомительного фрагмента.