VK Assistant

VK Assistant

Общая информация и основные понятия

Оглавление

Основные понятия

- **Скилсет (skillset)** — набор умений, объединенных по общим бизнес признакам. Состоит из следующих элементов:
  - **NLP инструменты**:
    - **NER-сервисы (системные и кастомные экстракторы)** — это инструмент, предназначенный для извлечения информации из текстов пользователя с последующим использованием этой информации в рамках сессии.
    - **NLP модели** — математический модуль, который использует датасет для классификации текста.
  - **Датасеты** — хранилище данных для классификации, наборы данных в формате класс (ключ) / текст (значение).
  - **Тесты** — сохраненные эталонные сессии, которые облегчают работу разработчикам и бизнес-аналитикам при проверке функционирования персептронов и скилсетов.
  - **FAQs** — блок ответов на часто задаваемые вопросы, которые вводит пользователь через usertext.
  - **Факт (fact)** — минимальная единица информации, которой оперирует система. Набор значений фактов — это контекст диалога (сессии).
  - **Agent (агент)** — внешняя или внутренняя программа, которая запускается при возникновении события. Есть три типа агентов:
    - **Researcher** — агент типа «исследователь». Результатом работы является найденный/не найденный ответ из базы знаний.
    - **Solver** — агент, который инициирует выполнение внешних задач, исполняет действия в целевых системах и возвращает результаты работы в виде кода и варианта.
    - **Internal** — это внутренние агенты, для использования которых не требуется подключение к внешней системе.
  - **Персептрон (perceptron)** — базовый элемент логики системы, простейшая нейронная сеть. Состоит из следующих элементов:
    - **Sensor (сенсор)** — это реагирующий нейрон, рецептор, который активируется от фактов.
    - **Association (ассоциация)** — логический нейрон персептрона.
    - **Abstraction (абстракция)** — вложенный персептрон.
    - **Response (респонс)** — реагирующий нейрон персептрона, реакция на логические выводы от ассоциации.
    - **Knowledge (элемент базы знаний, ЭБЗ, КБ)** — знание о необходимых действиях, желаемый результат диалога с пользователем.
- **Системный скилсет** - такой же скилсет, только в нем находятся сущности, отвечающие за работоспособность самой системы, которые также доступны для использования в кастомных скилсетах.
  - **Системные агенты** — это агенты типа Internal, которые отвечают за процессы ведения сессии.

    - **Системные факты** — это факты, поставляемые, с системой, которые обеспечивают работоспособность процесса общения пользователя с системой VK Assistant.
    - **Системные экстракторы** — это комплекс сложных экстракторов для продвинутого извлечения информации из текстов пользователей, например, извлечение адресов, стран назначения, номеров документов и т.д.
- **Сессия** — один диалог между пользователем и цифровым помощником, ведущий к определенной цели.
- **Событие** — отчет о любом изменении состояния сессии.
- **Классификатор** — инструмент классификации.
- **Классификация** — номер или наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту.
- **Лемматизация** — метод морфологического анализа, который сводится к приведению словоформы к ее первоначальной словарной форме (лемме).
- **Опечаточник** — система коррекции орфографических ошибок в запросах пользователей, использующаяся для улучшения работы классификатора.
- **Препроцессинг** — предварительная обработка данных.
- **Регулярное выражение** — формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов.
- **Токен** — единица текстовой информации, используемая для классификации текста пользователя.
- **Репозиторий** — хранилище пакетов скилсетов с тегами (документация для ознакомления — https://docs.npmjs.com/adding-dist-tags-to-packages) реализованный при помощи npm (документация для ознакомления — https://docs.npmjs.com/packages-and-modules/introduction-to-packages-and-modules) для транспортировки скилсетов с одного ландшафта на другой.

Скилсет (skillset)

Skillset (скилсет) — продукт решения определенной задачи, поставленной бизнесом, или результат улучшений для оптимизации работы системы.

С технической точки зрения скилсет — это функциональная единица VK Assistant, содержащая в себе средства решения конкретной диалоговой задачи или группы задач.

Каждый скилсет имеет уникальный идентификатор (ID), название, версию, флаг активности. Скилсету присвоен список ролей пользователей, для которых должны быть доступны сценарии диалогов, реализованных посредством данного скилсета.

Скилсет содержит следующие элементы:

- Персептроны — базовые элементы логики VK Assistant, содержащие алгоритм обработки событий и фактов в рамках диалога. Построены по принципу простейшей нейронной сети и состоят из элементов нескольких типов;
- Агенты.
- Факты.
- Датасеты.
- Тесты.
- FAQs.
- NLP инструменты — в частности, отвечающие за классификацию текстов, например, текстов запросов (пользовательских обращений):
    - NER-сервисы (системные и кастомные экстракторы) — это инструмент, предназначенный для извлечения информации из текстов пользователя с последующим использованием этой информации в рамках сессии.
    - NLP модели — математический модуль, который использует датасет для классификации текста.

Стартовый скилсет

Прежде чем решить вопрос пользователя, система использует стартовый скилсет, отвечающий за выбор темы обращения.

Стартовый персептрон прописывается в параметре конфигурации farewell_perceptron.

В настоящий момент логика определения темы обращения пользователя для каждого сервера разнится и выбирается индивидуально под проект. Порядок разработки:

1. Создание и утверждение архитектуры сценариев с учетом всех доступных методов платформы.
2. Создание скилсетов с использованием всех доступных методов платформы.

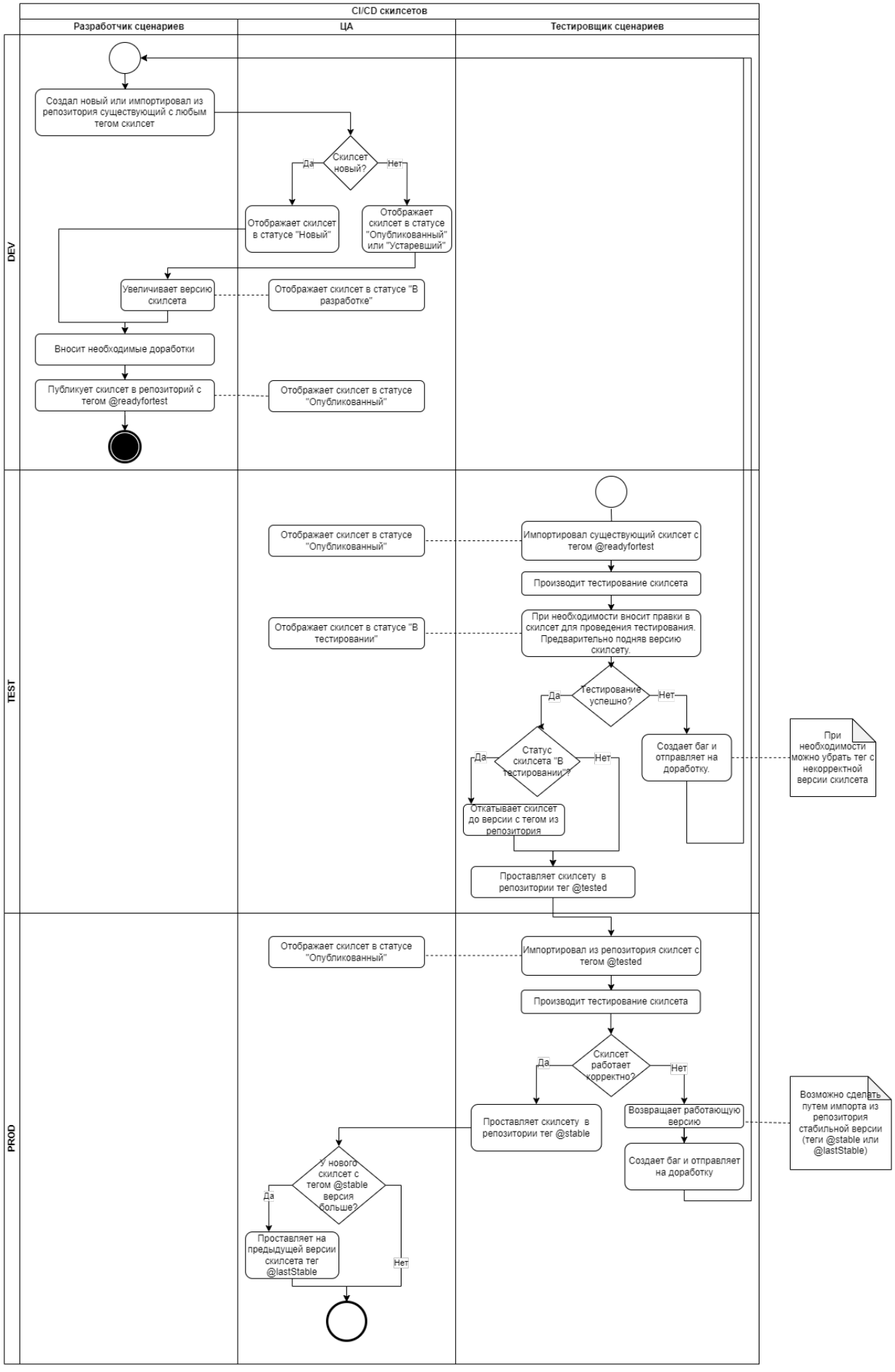
CI/CD скилсета

CI/CD — процесс разработки, тестирования и передачи скилсетов между инстансами системы.

Теги — метки на скилсетах в npm репозитории, позволяющие определить, какие сейчас производятся действия над скилсетом, на каком стенде скилсет доступен и прочее. Виды тегов:

- @latest — тег на последней публикации скилсета (данный тег не выбирается оператором, он ставится автоматически на версии, опубликованной последней);
- @readyForTest — скилсет, доступный для тестирования (проставляется на инстансе DEV);
- @tested — скилсет, прошедший тестирование (проставляется на инстансе TEST);
- @stable — стабильный скилсет, доступный для наката на продуктивную зону (проставляется на инстансе PROD);
- @lastStable — предыдущая стабильная версия, тег проставляется автоматически, когда ставиться тег @stable на новую стабильную версию прода, при условии, что новая версия будет больше старой.

Работа с репозиторием происходит следующим образом (нажмите на изображение, чтобы открыть его в новом окне браузера):



Особенности реализации

Фильтрация по ролям, статусам скилсетов и инстансе осуществляется по следующему алгоритму:

- Проверяется наличие подключенного репозитория.
- Если репозиторий подключен, то определяется инстанс и вычисляется статус скилсета.
- В зависимости от статуса скилсета и типа инстанса определяются доступные команды для скилсета (взять в работу, опубликовать и т.п.).
- Проверяется доступ к ресурсам по ролевой модели.

При отображении доступного обновления и расчете статусов скилсетов используется следующая логика:

- На DEV отображаются и учитываются в расчете статусов все версии скилсетов из репозитория.
- На TEST отображаются и учитываются в расчете статусов только те версии скилсетов из репозитория, которые имеют тег @readyForTest, @tested, @stable и @lastStable.
- На PROD отображаются и учитываются в расчете статусов только те версии скилсетов из репозитория, которые имеют тег @tested, @stable и @lastStable.
- Скилсет "system" нельзя публиковать в репозиторий на стендах DEV, TEST и PROD и нельзя редактировать на стендах TEST и PROD (для внесений изменений можно использовать импорт из файла измененного скилсета на стенде DEV).

Маппинги и соответствия для CI/CD скилсета

Ограничения по статусам скилсета не действуют на роль ADMIN.

Ниже представлен маппинг определения статуса скилсета в зависимости от комбинации его версий и подключенного репозитория (конкретные версии приведены для наглядности).

> **Важно**
>
> При отсутствии репозитория колонка статусов скилсетов не отображается на форме перечня скилсетов.

Английское наименование статуса	Описание	Локальная версия (Л)	Базовая версия (Б)	Версия в репозитории (Р)	Инстанс	Комментарий
	Отсутствует статус	-	-	-	DEV, TEST, PROD	Репозиторий не подключен
Conflict	Конфликт	не пустая	любая	любая	DEV, TEST, PROD	Репозиторий подключен Отсутствует или не

Английское наименование статуса	Описание	Локальная версия (Л)	Базовая версия (Б)	Версия в репозитории (Р)	Инстанс	Комментарий
						заполнен параметр repoId у скилсета
New	Новый	не пустая	-	-	DEV	Р = null и репозиторий подключен
Latest	Опубликованный	1.0.0	Не важна	1.0.0	DEV, TEST, PROD	Л = Р
Obsolete	Устаревший	1.0.0	Не важна	1.0.1	DEV, TEST, PROD	Л < Р
In dev	В разработке	1.0.1	Не важна	1.0.0	DEV	Л > Р
In test	В тестировании	1.0.1	Не важна	1.0.0	TEST, PROD	Л > Р

Ниже представлена таблица соответствия производимых действий для определенного процесса.

№	Процесс	Базовая версия	Локальная	Версия в репозитории	Обязательные действия
1	Создать скилсет	-	= указанный номер (дефолтно 1.0.0)	-	• Указать идентификатор • Указать название
2	Начать редактирование		= базовая + N		Поднять локальную версию (больше версии репо)

№	Процесс	Базовая версия	Локальная	Версия в репозитории	Обязательные действия
3	Опубликовать новую	= локальная		= локальная	• Выбор репозиторий для публикации • Проставление тега (при необходимости) • Добавление комментария
4	Опубликовать изменение	= локальная		= локальная	• Проставление тега (при необходимости) • Добавление комментария • Репозиторий подставится автоматически (перевыбрать при необходимости)
5	Импорт нового	= из репозитория	= из репозитория		• Выбор репозитория • Выбор скилсета • Выбор версии доступной в репозитории
6	Обновление до версии в репозитории	= из репозитория	= из репозитория		Подтвердить обновление
7	Вернуть предыдущую	= из репозитория	= из репозитория		• Репозиторий подставится автоматически (перевыбрать при необходимости) • Наименование скилсета подставится автоматически

№	Процесс	Базовая версия	Локальная	Версия в репозитории	Обязательные действия
					(перевыбрать при необходимости) • Выбрать версию на экране
8	Удалить скилсет				Подтвердить удаление
9	Создать снапшот				Вписать наименование снапшота

Ниже представлен маппинг отображения действий в зависимости от сред/инстанса.

Статус скил сета	Редакти ровать мета данные	Удалять	Открыть список слепков	Сделать слепок	Скачать в файл (Экспорт)	Взять в работу	Опубли ковать в репози торий	Откат версии	Работа с тегами	Обнов ление
DEV										
-	+	+	+	+	+	-	-	-	-	-
Conflict	- (просмотр доступен)	+	-	-	+	-	-	-	-	+
Obsolete	- (просмотр доступен)	+	-	-	+	-	-	+	-	+
New	+	+	+	+	+	-	+	-	-	-
Latest	- (просмотр доступен)	+	+	+	+	+	-	+	-	-
TEST										

Статус скил сета	Редактировать мета данные	Удалять	Открыть список слепков	Сделать слепок	Скачать в файл (Экспорт)	Взять в работу	Опубликовать в репозиторий	Откат версии	Работа с тегами	Обновление
-	+	+	+	+	+	-	-	-	-	-
Conflict	- (просмотр доступен)	+	-	-	+	-	-	-	-	+
Obsolete	- (просмотр доступен)	+	-	-	+	-	-	+	+	+
Latest	- (просмотр доступен)	+	-	-	+	-	-	+	+	-
PROD										
-	+	+	-	-	+	-	-	-	-	-
Conflict	- (просмотр доступен)	+	-	-	+	-	-	-	-	+
Obsolete	- (просмотр доступен)	+	-	-	+	-	-	+	+	+
Latest	- (просмотр доступен)	+	-	-	+	-	-	+	+	-

NLP инструменты

В ходе каждой диалоговой сессии может использоваться множество NLP инструментов. Базовые операции на сенсорах персептрона, классификаторы на сенсорах, классификаторы в агентах и т.д.

Уникальной особенностью VK Assistant является возможность управлять моделями непосредственно в графическом интерфейсе административного приложения. NLP сервисы основаны на SyntaxNet (TensorFlow based NLU toolkit) и др.

При разворачивании VK Assistant внутри корпоративного ландшафта NLP сервисы также являются внутренними, то есть не требуется никаких обращений к внешним сервисам.

К NLP инструментам относятся:

- NLP модели;
- NER-сервисы (экстракторы):
    - Системные экстракторы;
    - Кастомные экстракторы.

NLP модели (Natural Language Processing)

NLP модель — математический модуль, который использует датасет для классификации текста.

Решаемые задачи:

- **Классификация текстов**

  Классификация текстов используется для определения намерения пользователя в диалоге при помощи определения класса (темы) запроса. Для решения задачи классификации текстов на основе размеченных датасетов создаются NLP-модели. Для этого используются методы предобработки текстов и методы машинного обучения. Модель интегрируется в диалог. Работа с датасетами подробно описана здесь.

- **Распознавание речи (speech recognition)**

  Распознавание речи используется для обработки звучащего голоса пользователя. Запись пользовательского запроса сохраняется в файле и преобразуется в текст. Дальше текст обрабатывается методом классификации текстов.

- **Морфологический анализ**

  Морфологический анализ текста используется для распознавания лексических единиц в структуре предложения. Морфологический анализ необходим для создания иерархической структуры слов в предложении, которая используется для выявления ключевых слов. Ключевые слова – это лексемы, несущие основной смысл сообщения. Морфологический анализ используется в середине диалога, чтобы понимать намерения пользователя.

- **Распознавание именованных сущностей**

  Распознавание именованных сущностей (NER) используется для извлечения типовой информации из usertext. Извлеченную информацию можно устанавливать как факт в сессию или передавать моделям для последующей обработки.

NER-сервисы (экстракторы)

NER-сервисы (от англ. named entity recognition - распознавание именованных сущностей) — это инструмент, предназначенный для извлечения информации из текстов пользователя (особенно из usertext) с последующим использованием этой информации в рамках сессии (установка значений фактов). В рамках данной документации ner-сервисы также называются экстракторами (от англ. extract - извлекать). Экстракторы NER-сервисов созданы на базе библиотек Natasha и Yargy а также при помощи regexp.

NER-сервисы делятся на системные и кастомные экстракторы.

- Системные экстракторы — это готовые инструменты для извлечения типовой информации.
- Кастомные экстракторы — это набор инструментов в административной панели, предназначенный для создания новых экстракторов, а также созданные экстракторы.

Вызов экстрактора (системного или кастомного) осуществляется в js-агенте при помощи метода **await helper.extract:**

```
await helper.extract(fact.value, 'extractorID', 'RU')
```

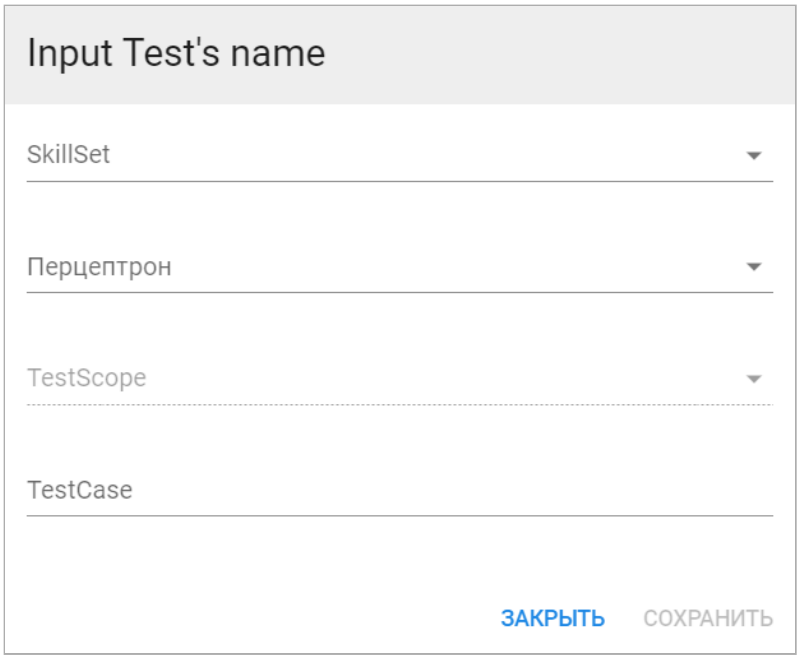
Тесты

Тесты — это сохраненные эталонные сессии, которые облегчают работу разработчикам и бизнес-аналитикам при проверке функционирования персептронов и скилсетов. Создание тестов является завершающим этапом разработки сценариев перед передачей его в пользование. **Созданные тесты считаются результатом работы разработчика сценариев.**

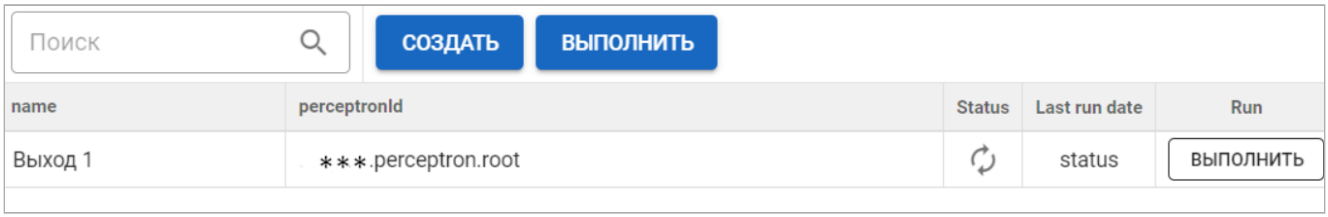
При переносе диалога из одного ландшафта в другой тесты должны быть созданы как завершение переноса. Каждый ландшафт должен содержать тесты ко всем сценариям. При любых изменениях в диалоге (сценарии) тесты должны быть актуализированы. **Поддержание актуальности теста является задачей разработчика сценариев.**

Чтобы создать тест:

1. В своем скилсете во вкладке добавьте папку для сохранения тестов.
2. В течение или после сессии включите debug-режим и нажмите на значок
3. Выберите id скилсета, id персептрона, папку, введите название теста и сохраните его:



4. Вернитесь в папку и поменяйте id персептрона на "root":



5. Сэмулируйте работу сессии при помощи кнопки **Выполнить**.

Подробная инструкция по составлению, редактированию и использованию тестов представлена в инструкции к административной панели.

FAQs

FAQ — это блок ответов на часто задаваемые вопросы, которые вводит пользователь через usertext.

В VK Assistant есть два типа FAQ:

- Глобальный FAQ — включает в себя все FAQ с разных областей знаний (скилсетов).
- Локальный FAQ — включает в себя вопросы и ответы на определенную область знаний.

Факт (Fact)

Fact (факт) — это минимальная единица информации, которой оперирует система. Логика будущего диалога (сессии) должна начинаться с перечня фактов, которые нужно собрать.

Факты бывают двух типов:

- Системные — поставляются вместе с системой и их можно использовать в любом скилсете.
- Кастомные — создаются в скилсете под нужды сценарной логики.

Факт можно получить одним из способов:

- спросить его у пользователя при помощи Prompt;
- задать значение при помощи действия Set Parameter в респонсе;
- установить значение с помощью агента.

Первый Fact в сессии — это usertext, т.е. ответ пользователя на приветствие системы. На основе usertext система делает выводы и спрашивает следующие факты (задает вопросы).

В старой парадигме «факт» назывался «параметром». Это отразилось на названиях некоторых функций API: setParameter, getParameter и т.д.

Набор значений фактов — это контекст диалога (сессии). Логические выводы производятся только в контексте сессии, т.е. на основе набора значений фактов. После того, как факт получен, он влияет на сенсор.

Переход от факта к сенсору неявный, для него используется операция Stimulate.

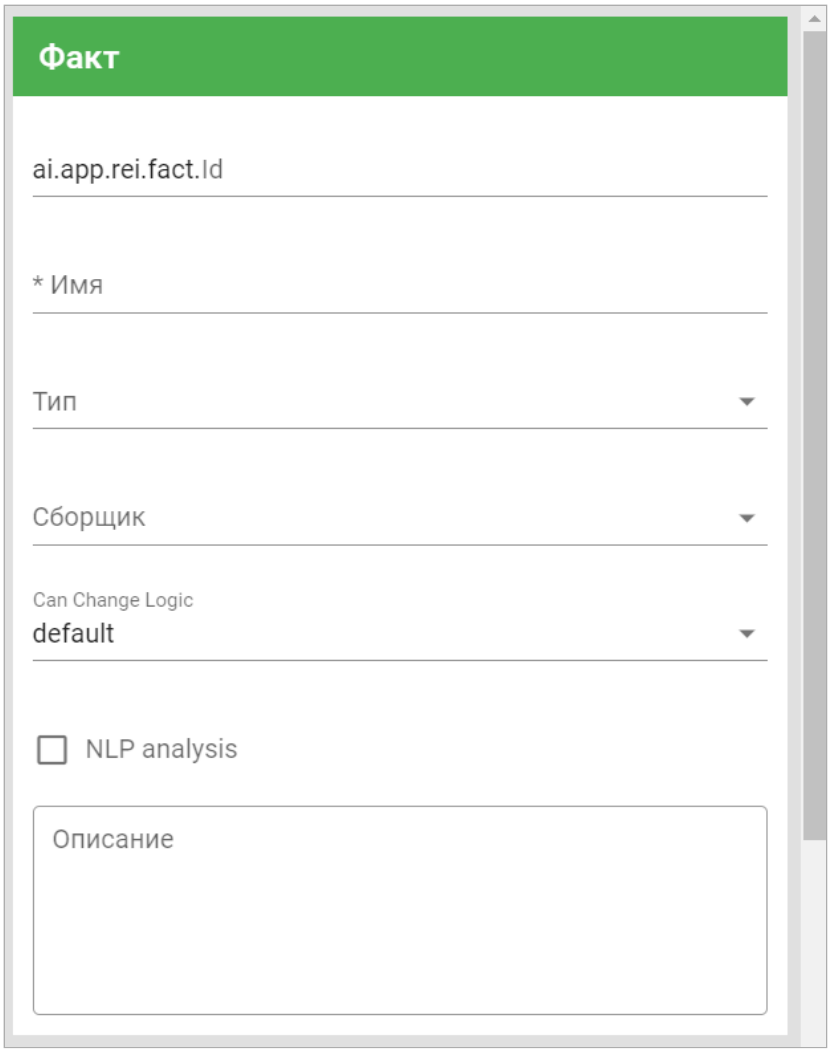
Параметры и свойства факта

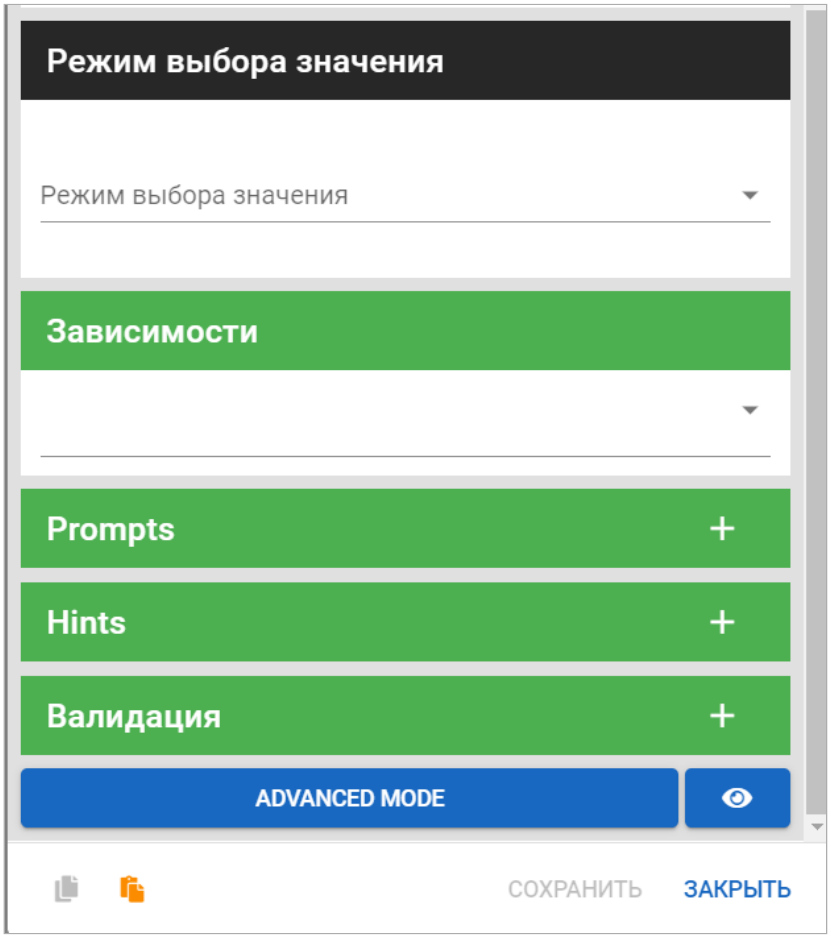
Свойства факта — это описание единичного «знания», которое позволяет системе понять: как обращаться с фактом, как спрашивать его у пользователя, как проверять на валидность.

Некоторые свойства являются системными. По умолчанию формулировка в чат: «введите $id факта$».

В expandParameter добавлена возможность доставать по ключу значение из JSON. Можно записать JSON с ключами (например, id и name) в факт, а потом достать значение по ключу. Например из факта test можно будет достать значения ${test}.id или ${test}.name

Форма создания факта


Факт
ai.app.rei.fact.Id
* Имя
Тип
Сборщик
Can Change Logic
default
NLP analysis
Описание


Режим выбора значения
Режим выбора значения
Зависимости
Prompts
Hints
Валидация
ADVANCED MODE
СОХРАНИТЬ
ЗАКРЫТЬ

Элементы на форме:

- id — обратная доменная нотация для уникальности в формате ".fact.<уникальный id факта в skillset>" (тип: String).
- Имя — наименование (тип: String).
- Тип — не используемый параметр.
- Can Change Logic — реализация перехода в другой скилсет в рамках диалога.
- NLP analysis — необходимость запуска NLP-анализатора.
- Режим выбора значения — настроить выбор значения:
  - Классификатор или choice from classify — для выбора значения по классификации введенного текста пользователя по заданным вариантам.
  - Кнопки — для выбора значения по точному выбору из предлагаемого списка.
  - Дата — для выбора значения из календаря.
  - Агент — используется для запросов в целевые системы через встроенное веб приложение (создавалось для использования miniapp).
- Зависимости — задать родительский факт из выпадающего списка.
- Prompts — вопрос, который задает система, чтобы выяснить факт.
- Hints — help prompt, пояснение факта для пользователю.
- Валидация — проверка факта, список валидаторов (тип: Array).
- Advanced mode — открыть блок кода для описания дополнительных свойств факта.

Немного подробнее про свойство NLP analysis

Факт с активным признаком "NLP analysis" при answer делаем разбор и сохраняем в факте дополнительные признаки (keywords и результат синтаксического разбора).

Например сообщение: «ну убираю снег с крыши» в usertext добавит в факт свойство nlpParsing типа:

```
{ "nlpParsing": { "syntax": {}, "keywords": [ "убирать", "снег", "крыша", "ну", "убираю",
"с", "крыши" ], "language": "RU" }
```

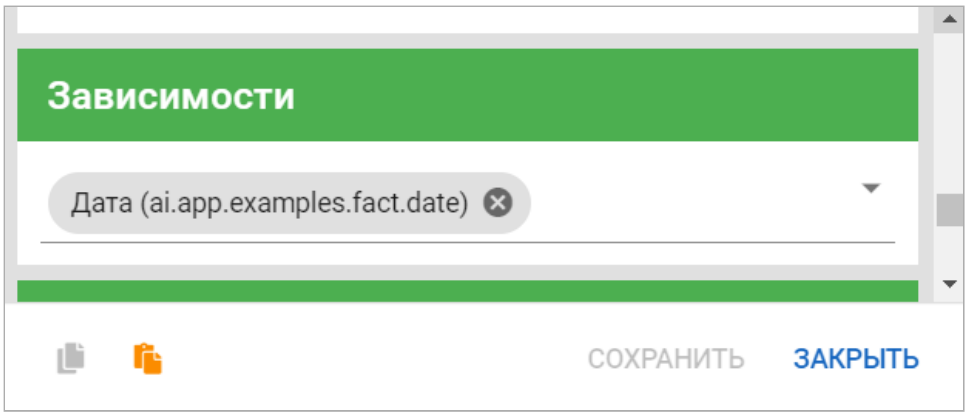
Дополнительные свойства факта:

Свойство	Тип	Описание
properties	Object	Дополнительные свойства факта
properties.nlp	Boolean	TRUE значит, что при получении значения факта запускаются NLP анализаторы
properties.prompts	Complex	PROMPT
properties.hint	Text	пояснение о назначении факта для пользователя

Свойство	Тип	Описание
properties.type_details	Object	Описание способа запрашивания факта у пользователя. Например, для "choice" - список вариантов

Зависимости (Depends)

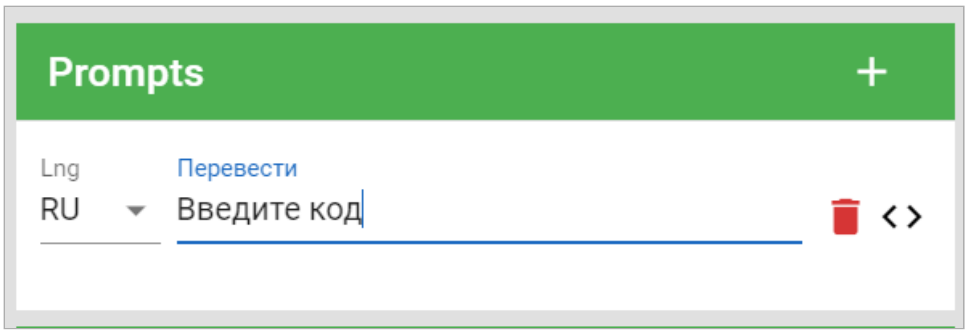
Depends реализует зависимость фактов и используется для построения дерева зависимостей от общего к частному. Depends позволяет назначить факт-родитель. Пока факт-родитель не будет установлен (пользователем или агентом), невозможно спросить зависимый факт.



Запрос (Prompts)

Prompts — это текст (вопрос или рекомендация), который система покажет пользователю, чтобы выяснить факт. По умолчанию формулировка в чат: “Введите $id факта$”.

Prompts представлен в виде JSON-объекта (ключ: целевой язык). Значение Prompts представлено в виде одной строки или массива строк. В случае наличия вариабельности (при значении "массив") вопрос выбирается случайным образом во избежание повторения одинаковых вопросов. Prompts могут использовать любой заданный язык.



В Prompts есть возможность реализации подстановок информации из уже известных фактов с помощью символа **$**, например: $имя пользователя$

Способы задания вопросов (text, choice, etc)

Способ запроса факта зависит от его типа:

- text (по умолчанию) — факт спрашивается в виде текста, ответ сохраняется в виде строки;
- choice — система предлагает пользователю несколько вариантов ответа на выбор.

Список готовых вариантов можно задать в Advanced mode.

При помощи Advanced mode можно:

- добавить список готовых вариантов для Choice в Prompt.

```
{
    "nlp": false,
    "type": "choice",
    "type_details": {
        "RU": {
            "list": [{
                "title": "Первый сценарий",
                "value": "1"
            }, {
                "title": "Второй сценарий",
                "value": "2"
            }, {
                "title": "Третий сценарий",
                "value": "3"
            }],
            "disableNumbering": true
        }
    },
    "prompts": {
        "RU": "Выберите сценарий",
        "EN": "Please, choose the topic"
    },
    "canChangeLogic": "default"
}
```

Параметр disableNumbering отвечает за наличие нумерации элементов списка (true-нумерации нет, false или отсутствие параметра - нумерация есть).

Hints

Для поддержания user friendly интерфейса существует Help prompt, который прописывается в Hints. Он используется для того, чтобы дать пользователю подробную информацию о запрашиваемом факте: какой именно факт, зачем он нужен системе и где пользователь может найти этот факт. Таким образом Help prompt переспрашивает факт. Для вызова Help prompt пользователю нужно ввести знак вопроса.

Пример диалога:

— Введите номер расчетного счета

— ?

— Мы спрашиваем номер расчетного счета, потому что ... . Узнать номер расчетного счета можно ...

Валидация фактов (Validators)

Если у факта есть точные характеристики, то для него можно задать параметры валидации. В таком случае валидатор проверит факт в Целевой Системе (ЦС) и даст ответ TRUE/FALSE. Факт не устанавливается, пока валидатор не даст ответ TRUE.

Существуют стандартные валидаторы и валидаторы-агенты. Для создания агента-валидатора можно использовать https://regexr.com/. Валидацию нужно начинать с символа **^** и заканчивать **$**, например:

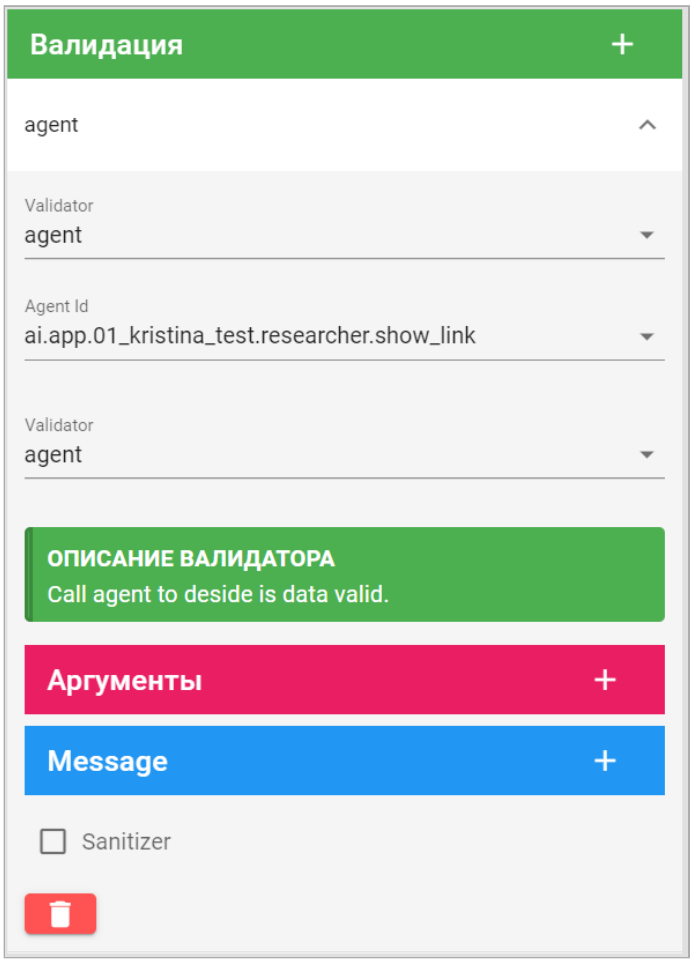
^([7]+[1]+\d{8}|[S]+[R]+\d{8})$

, где:

- ^ — значит начало
- $ — конец строки (т.е. после символа $ нельзя продолжать строку)

Также можно выбрать один из системных валидаторов:

- contains — содержит;
- equals — идентично;
- isAlphanumeric — содержит буквы и цифры;
- isCurrency — является валютой;
- isEmail — является адресом email.



Укажите в поле **Message** сообщение, которое система покажет пользователю в случае несоблюдения валидации, например:

usertext: 222

system: Введите семизначный номер

Флаг **Sanitizer** необходим для случая **valid**, когда валидатор может вернуть и своё какое-то значение. Обработанное, более правильное и т.д.

Например, есть какой-то условный валидатор ОС, который каким-то своим особым образом обрабатывает ответ юзера на какой-то факт. Юзер пишет "робот", ответ обрабатывается, валидатор возвращает: {"code":"valid"}. В факт записывается "робот" — это ситуация, когда валидатор ничего не нашел.

В аналогичном же случае валидатор может вернуть и свой value: {"code":"valid", "value":"android"}. В таком случае, если у валидатора проставлено свойство **sanitizer**, хоть юзер и написал "дроид", в факт по итогу попадёт "android".

Агенты

Агент — это внутренняя или внешняя программа, которая запускается при возникновении события, на которое он подписан. Агент получает на вход состояние сессии, включая значения фактов, и возвращает статус и одно из действий. Если агент вернул действие "askParameter", то он автоматически подписывается на событие change этого факта и вызывается один раз при ответе.

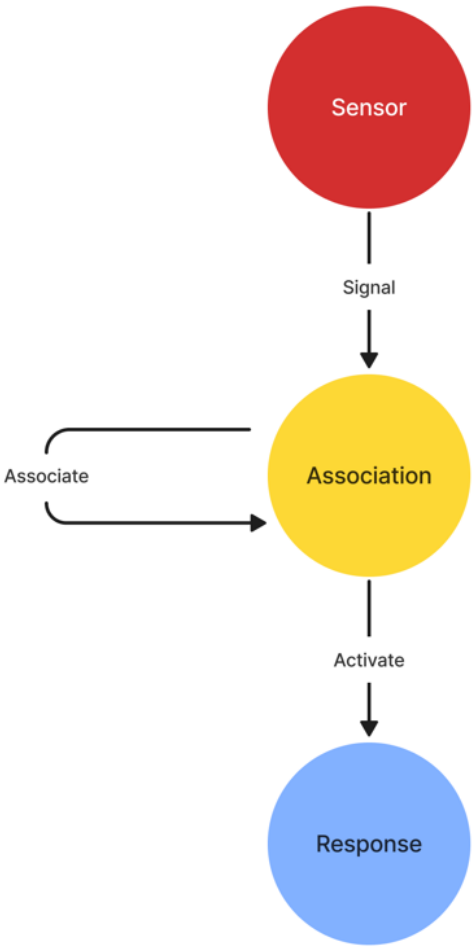
В рамках системы используется 2 типа агентов:

- системные (отвечающие за session flow).
- функциональные (для выполнения специальных функций сценария):
    - Researcher (исследователь) — получает на вход состояние сессии (все собранные факты) и путь к целевой системе. Агент может вернуть Knowledge или выполнить определенные действия: установить/спросить факт, показать пользователю информацию и т.д.
    - Collector - researcher агент, который запускается по событию изменения факта. т.е. при появлении какой-то информации в сессии добыть на основании этого что-то ещё дополнительно. Может быть агентом типа JS, REST, Remout и т.д. (чаще всего JS).
    - Validator - researcher агент, который указывается в настройках факта и валидирует ответ пользователя на аск. Может быть агентом типа JS, REST, Remout и т.д. (чаще всего JS).
    - Solver — исполняет действия в целевых системах и возвращает результаты работы в виде кода и варианта. Solver имеет возможность устанавливать ЭБЗ и факты (не прямое назначение).

Персептрон (Perceptron)

Классический простейший персептрон — базовый элемент логики системы. Это простая нейронная сеть, состоящая из трех видов элементов:

- сенсор (S), или рецептор, принимающий нейрон персептрона;
- ассоциация (A), или логический вывод, ассоциативный нейрон персептрона;
- респонс (R), или реакция на логический вывод, реагирующий нейрон персептрон.



Персептрон получает на входе значения фактов и выдает на выходе перечень действий в рамках одной сессии. Сигнал движется только по направлению стрелок: S → A, A → A, A → R.

Принцип действия персептрона:

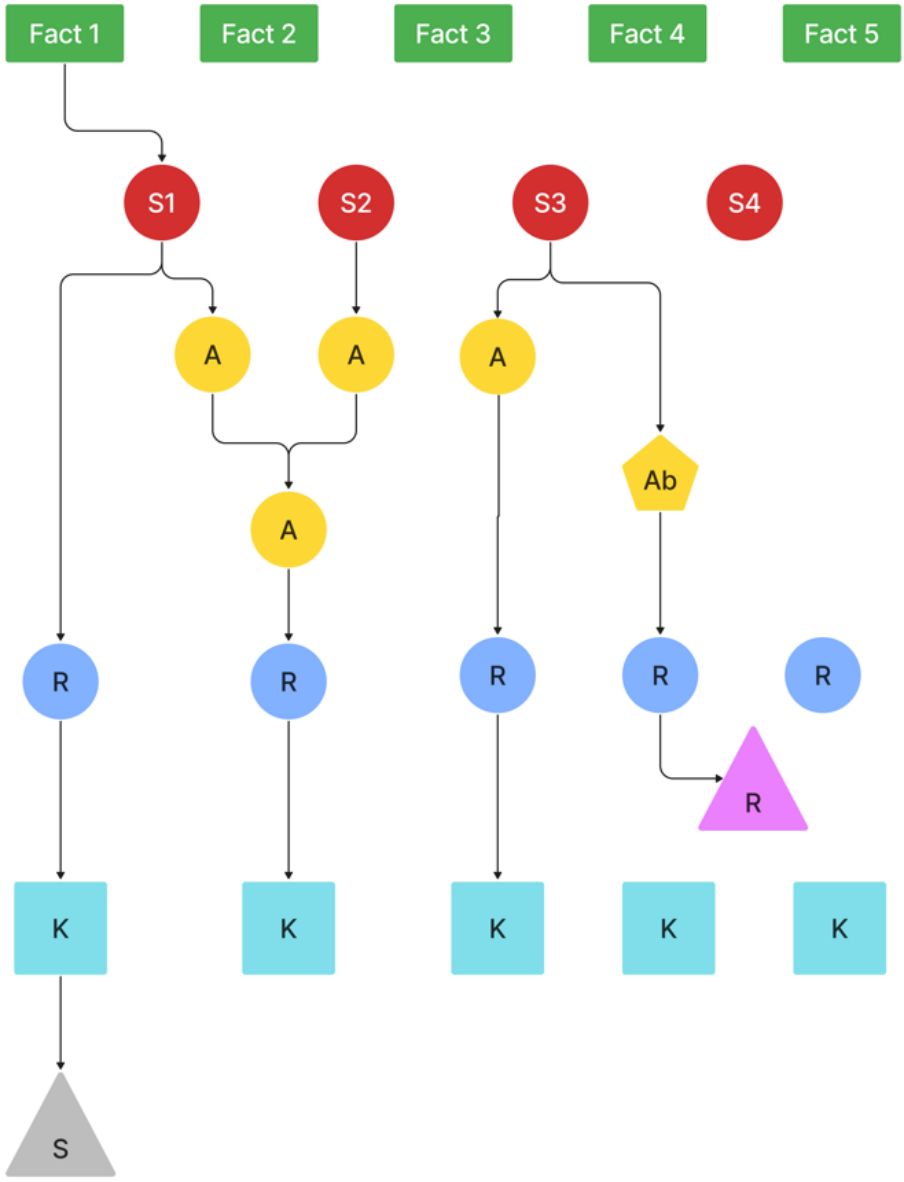
- факты активируют сенсор посредством операции Stimulate;
- сигналы между нейронами имеют целочисленный вес (от -100 до 100);
- уровень активации нейрона - это сумма входящих сигналов (stimulate);
- исходящие сигналы имеют пороговый уровень:
  - если уровень активации нейрона выше (или равен) пороговому уровню, то сигнал идет дальше;
  - если уровень активации нейрона ниже порогового уровня, то сигнал не активирует дальнейшую ветку;

- в том случае, если сигнал достиг реагирующего нейрона (Response), система получает руководство к действию и запускает КБ или агента.

Расчет персептрона

Стартом расчета персептрона занимается агент Dispatcher. Пересчет персептрона осуществляется после каждой смены состояния факта или фактов. Результатом расчета персептрона считается перечень активированных респонсов, где уровень активации больше 0:

- рассчитываются все связи при проходе по графу (S - A - R)



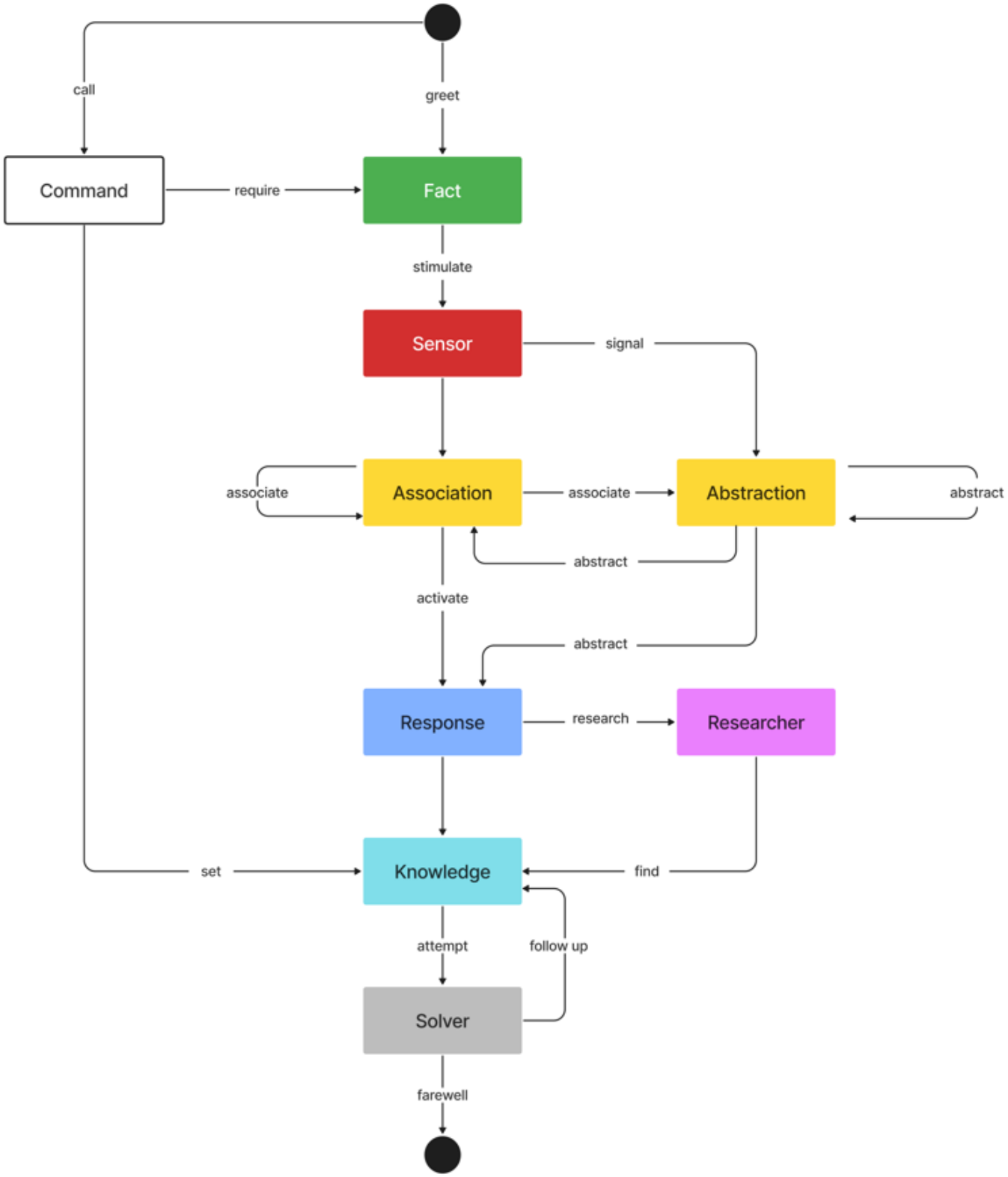
- респонсы сортируются по силе входящего сигнала;
- запускается реакция в соответствии с очередностью.

Возможен другой способ реализации персептрона, при котором компилируется функция на JS. Выходами являются агенты типа Solver и Researcher:

- агент Solver выводит Knowledge;
- агент Researcher выводит actions.

При создании персептронов можно использовать факты, агенты, knowledge из Root skillset.

Жизненный цикл персептрона

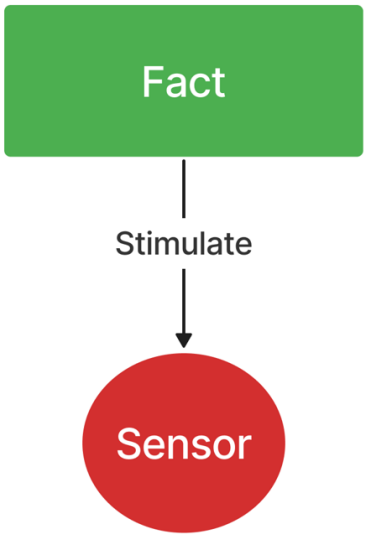


- **Greet** — приветствие.
- **Fact** — единица знания.
- **Stimulate** — правило, по которому факт активирует сенсор.
- **Sensor** — реагирующий нейрон персептрона, рецептор.

- **Association** — ассоциативный нейрон - логический вывод.
- **Signal** — связь между сенсором и ассоциацией.
- **Associate** — связь между ассоциациями.
- **Abstraction** — вложенный персептрон.
- **Abstract** — связь между результирующим респонсом вложенного персептрона и нейронами типа ассоциация, абстракция, респонс.
- **Activate** — активация - связь между логическим выводом и реагирующим нейроном.
- **Response** — реагирующий нейрон персептрона, реакция на логические выводы.
- **Researcher** — агент-исследователь.
- **Research** — процесс запуска агента.
- **Find** — нахождение знания (Knowledge) в процессе исследования.
- **Guess** — догадка о знании (Knowledge) в результате реакции персептрона.
- **Knowledge** — знание (KB в старой версии).
- **Attempt** — попытка выполнить действие, старт агента Solver.
- **Solver** — агент-решатель проблемы.
- **FollowUp** — настраиваемая ассоциация на продолжение подобных решений .
- **Farewell** — прощание.
- **Command** — краткая команда, которая должна выполняться без размышлений.
- **Require** — для выполнения команда требует факты (параметры).
- **Set** — установка знания (Knowledge) в результате выполнения команды.

Сенсор (Sensor)

Sensor (сенсор) — это реагирующий нейрон, рецептор, который активируется от фактов.

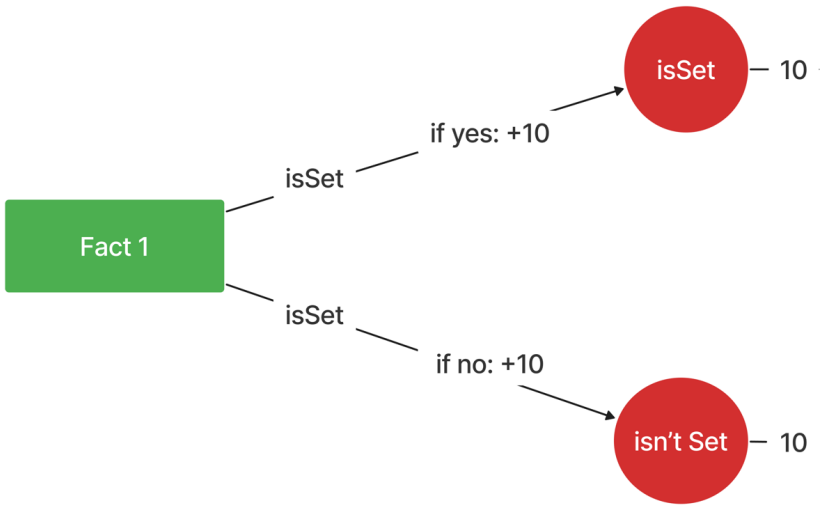


- **Факт (Fact)** — параметр, единица знания.
- **Операция (Stimulate)** — операция, правило, в соответствии с которым факт активирует сенсор.

- **Сенсор (Sensor)** — реагирующий нейрон персептрона, рецептор. Это реакция на отработанную функцию. Имеет входящие и исходящие сигналы. Входящими сигналами являются только операции над фактами — сенсор получает «количественный коэффициент» операции. Каждый факт ведет к определенному сенсору. Если сумма входящих сигналов больше 0, то нейрон активируется и передает сигналы дальше. Каждый исходящий сигнал имеет собственный порог. Сигналы сенсора суммируются в ассоциациях.

Операции

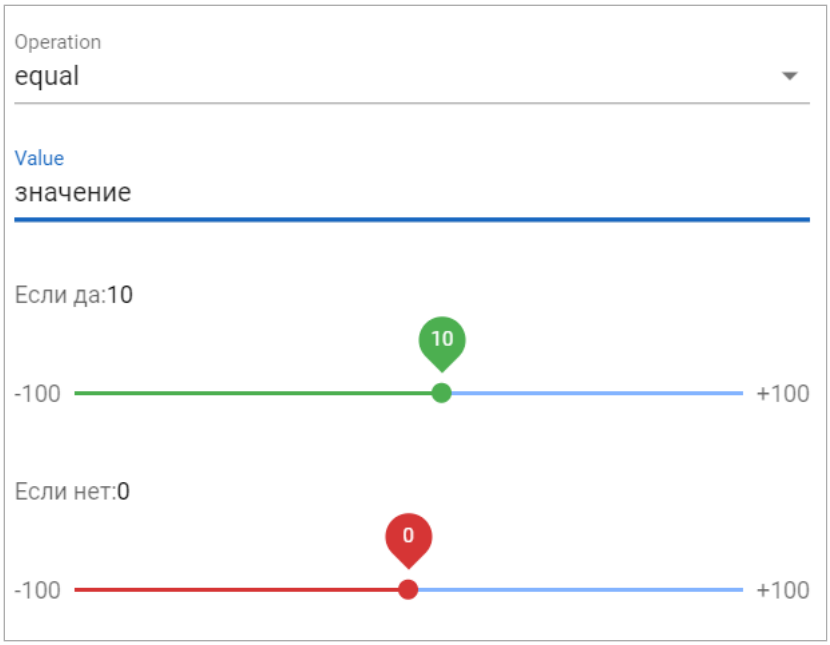
Операция — это функция, которая на вход берет значение привязанного факта и аргументы самой операции и на выходе возвращает TRUE или FALSE. Если факт установлен, то на выходе операция возвращает TRUE, активируя следующий нейрон.



Каждая операция имеет id, имя, тип и value.

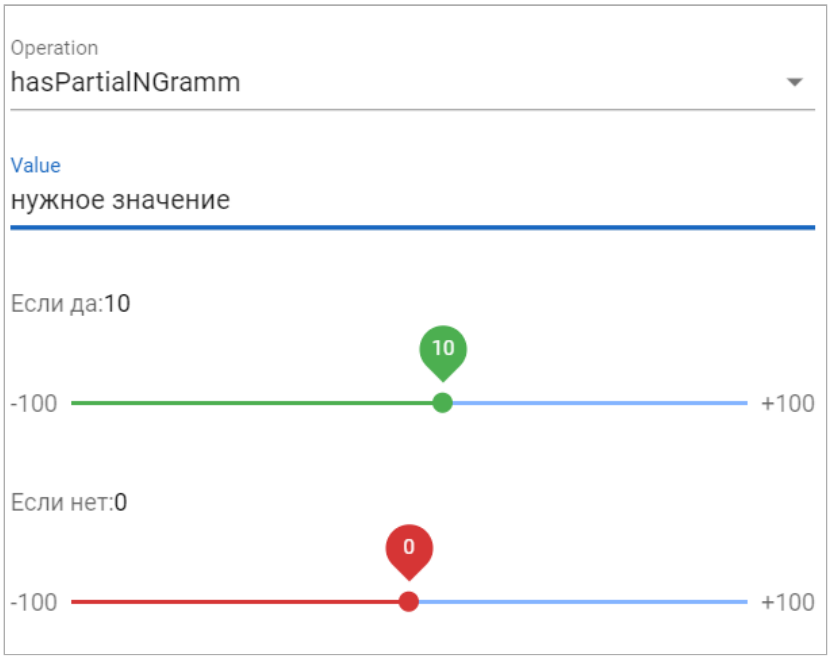
По типу различают встроенные (internal) и кастомные (custom) операции. Встроенные операции включены в ядро. Если для реализуемой логики требуется новая операция, то можно написать кастомную операцию. Кастомные операции написаны на языке JS в административной панели.

Пример встроенной операции equal, которая сравнивает факт со значением, которое указано в value:



Кастомные операции могут быть достаточно сложными и включать в себя агенты. Без необходимости вызовов агентов следует избегать: вызовы агентов асинхронны, а операция должна быть выполнена здесь и сейчас

Пример кастомной операции hasPartialNGramm, value которой представляет собой фразу, состоящую из корней слов:



Часто используемые операции:

Встроенные:

№	id	name	Назначение
1	equal	equal	Возвращает TRUE, когда факт равен указанному значению

№	id	name	Назначение
2	hasBigram	hasBigram	Возвращает TRUE, когда введенный usertext имеет лексемы, указанные в value
3	hasKeyword	hasKeyword	Должна стоять галочка NLP analysis. Возвращает TRUE, когда в предложении существует указанное слово (полное) Пример: указано: статус предложение: Нужен статус заявки (вернет TRUE) предложение: Нужен статус заявки (вернет FALSE)
4	hasKeywords	hasKeywords	Должна стоять галочка NLP analysis. Возвращает TRUE, когда в предложении существует указанное словосочетание. Возможна запись нескольких слов в словосочетании через запятую, а несколько словосочетаний через точку с запятой.
5	hasMultipleSentences	hasMultipleSentences	Возвращает TRUE, когда введенный usertext состоит из нескольких предложений
6	isChanged	isChanged	Возвращает TRUE, когда факт изменяется в сессии
7	isChangedBtwnCalc	isChangedBtwnCalc	Возвращает TRUE, когда значение факта изменилось в процессе расчета персептрона
8	isInFaqWithProbability	isInFaqWithProbability	Возвращает TRUE, когда введенный usertext является вопросом из FAQs
9	isPresent	Parameter is present in session	Возвращает TRUE, когда факт уже установлен в сессии

№	id	name	Назначение
10	isSet	isSet	Возвращает TRUE, когда факт известен
11	isUserInPlatform	isUserInPlatform	Вести операцию надо от факта system.fact.session.user_platforms Возвращает TRUE, когда Platform пользователя совпадает с указанной (например, SAP_S4)
12	isUserInPlatformType	isUserInPlatformType	Вести операцию надо от факта system.fact.session.user_platforms Возвращает TRUE, когда PlatformType пользователя совпадает с указанным (например, SAP)
13	isUserInRole	isUserInRole	Возвращает TRUE, когда Role пользователя совпадает с указанной (например, ADMIN)
14	nlpClassify	nlpClassify	Возвращает TRUE, когда введенный usertext с указанной вероятностью классифицирован моделью
15	nlpIsInFaqWithProbability	nlpIsInFaqWithProbability	Возвращает TRUE, когда введенный usertext с определенной вероятностью является вопросом из FAQs
16	receivedSignalFrom	receivedSignalFrom	Возвращает TRUE, когда сигнал получен от сущности, указанной в value
17	receivedSignalFrom Abstraction	receivedSignalFrom Abstraction	Возвращает TRUE, когда сигнал получен от абстракции, указанной в value
18	sentenceType	sentenceType	Возвращает TRUE, когда тип предложения в факте совпадает с нужным типом предложения. Пример: нужно : assertive предложение: Обычный текст

№	id	name	Назначение
			или нужно : exclamatory предложение : Обычный текст!

Кастомные:

№	id	name	Назначение
1	greaterThen	greaterThan	Возвращает TRUE, когда значение usertext больше указанного
2	isMonth	isMonth	Возвращает TRUE, когда usertext является месяцем
3	neq	NotEqual	Возвращает TRUE, когда usertext не равен значению, указанному в value
4	partialNGramm	hasPartialNGramm	Возвращает TRUE, когда usertext содержит корень, указанный в value
5	partialNGramms	hasPartialNGramms	Возвращает TRUE, когда usertext содержит корни, указанные в value

Пример использования операции isUserInRole в JS агенте

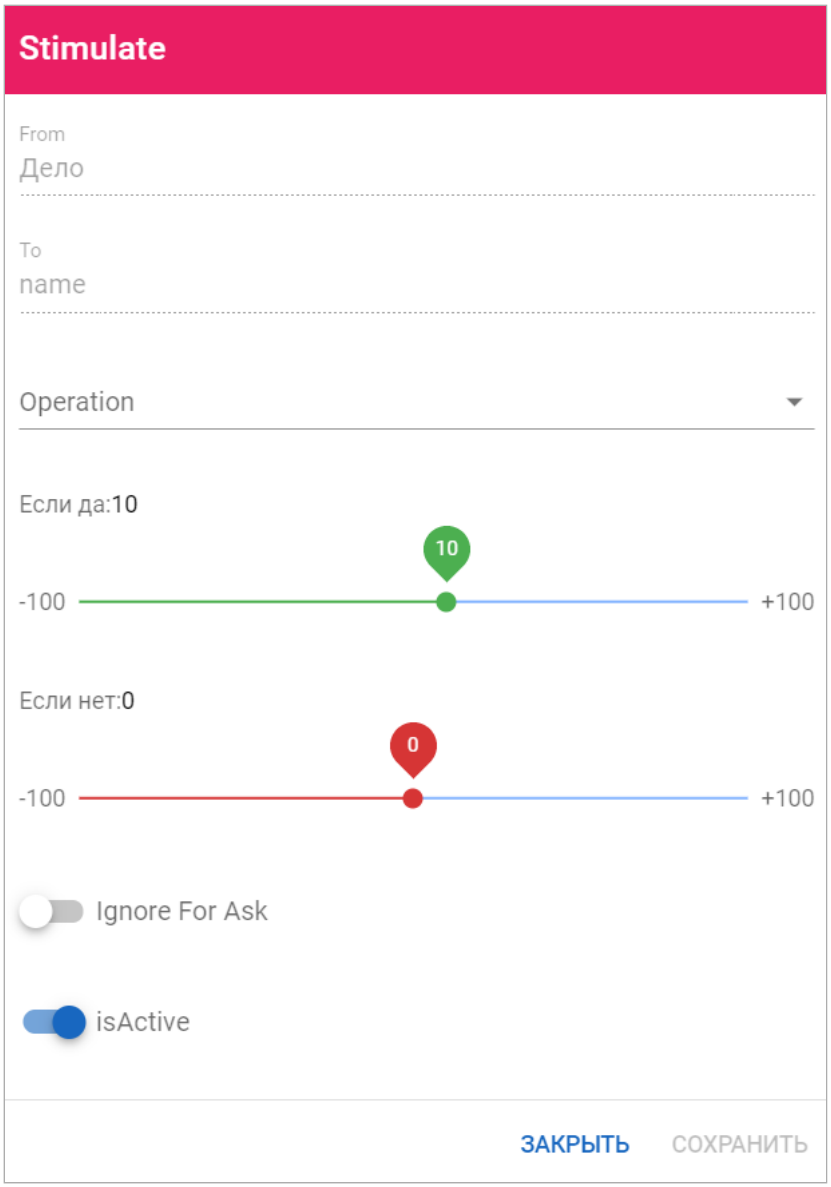
Ниже представлен пример использования операции isUserInRole посредством JS агента:

```
(() => { // Do not remove this line!

const user = _.find(sessionParameters, ["parameterId", "[gp]system.fact.quboUser"]);
return (async () => {
    try {
        if (user && user.value) {
            let userJson = JSON.parse(user.value);
            let isGuest = _.filter(userJson.roles, ["name", "GUEST"]);
                if (isGuest.length) {
                    <Какое-то действие>
                    }
            }
} catch (err) {
    await logger.write('generateIdeal text err', JSON.stringify(err));
    }
    return {}
})();

}); // Do not remove this line!
```

Операция Stimulate

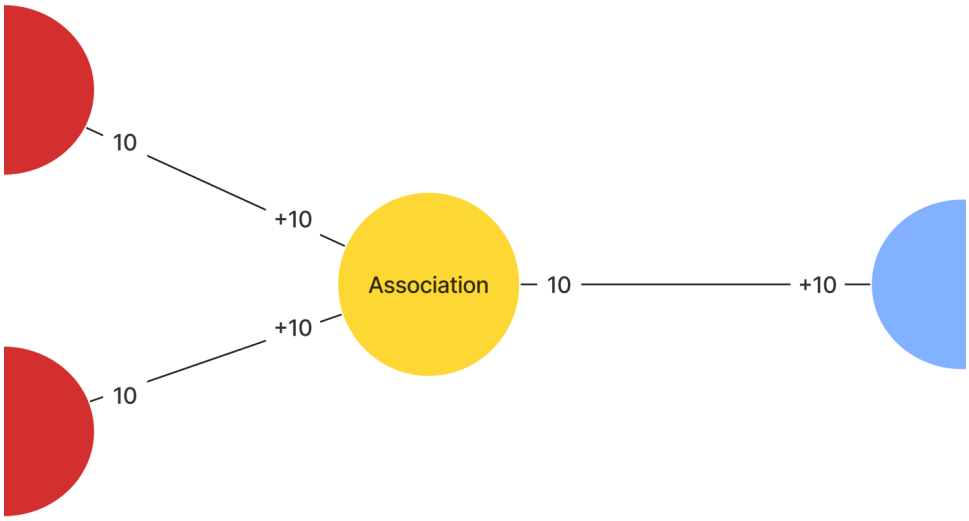
Операция Stimulate определяет порог входящего сигнала (Input signal level), т.е. достаточен ли сигнал, полученный от факта, для перехода нейрона в активное состояние. Если сигнал достаточен, т.е. сумма входящих сигналов больше 0, то нейрон активируется

Параметры настройки:



Ассоциация (Association)

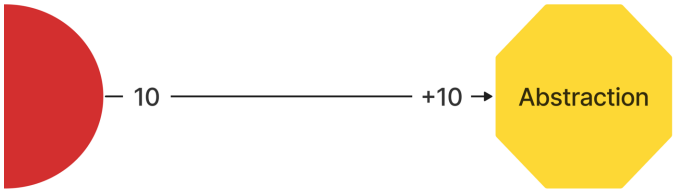
Association (Ассоциация) — логический нейрон персептрона. Принимает на вход сумму исходящих сигналов от сенсора или других ассоциаций и абстракций и активируется при сумме входящих сигналов больше 0. Ассоциация используется для того, чтобы продолжить цепочку действий, если накопленная сумма сигналов равна или выше порога выхода. Если накопленная сумма сигналов ниже порога выхода, ассоциация останавливает действие.



Каждая ассоциация имеет логическое объяснение. Настоятельно рекомендуется всегда присваивать ассоциациям простые, понятные и короткие названия на русском языке.

Абстракция

Абстракция — это ассоциация более высокого уровня, которая содержит в себе вложенный персептрон. Является вложенным персептроном. Принцип работы абстракции схож с капсульной нейронной сетью.



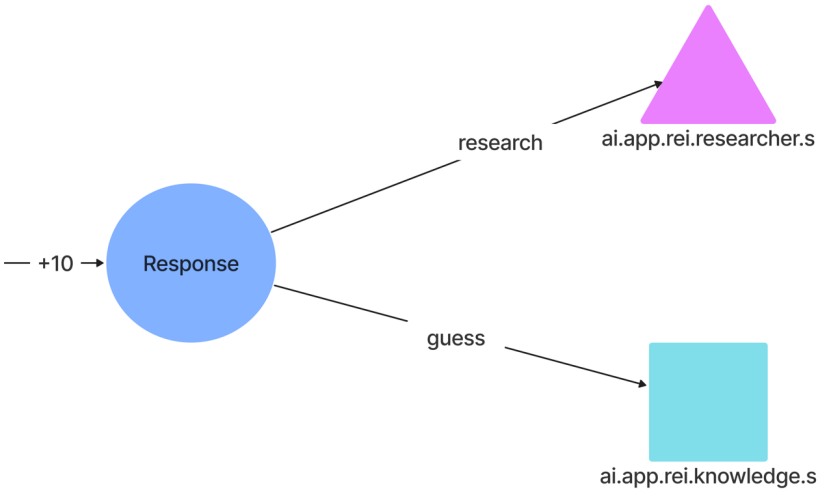
Система рассчитывает вложенный персептрон в том случае, когда уровень активации абстракции равен или выше установленного. Исходящий из абстракции сигнал связан напрямую с реагирующими нейронами внутри вложенного персептрона, т.е. можно узнать, сработала ли абстракция, только тогда, когда сработал респонс вложенного персептрона. Абстракция передаст сигнал дальше, если внутри встроенного персептрона реагирующий нейрон (указанный в сигнале) получит уровень активации больше 0.

В правильно построенном персептроне у абстракции нет выхода в КБ, только респонс, т.к. абстракция — это только логический вывод. Внутри абстракции можно создать несколько респонсов и добавить внутри персептрона несколько действий для каждого из них.

На данный момент чаще используется переход в другой персептрон через респонс при помощи action setPerceptron.

Респонс (Response)

Response (Респонс) — это реагирующий нейрон перцептрона, реакция на логический вывод от ассоциации. Респонс активируется при сумме входящих сигналов больше 0. Входящие сигналы поступают от ассоциаций, абстракций и сенсоров. Не рекомендуется использовать непосредственную связь сенсор-респонс (такую логику сложно модифицировать).



Реакции, связанные с Response:

- Reseacher — явный запуск агента исследователя.
- Knowledge — явная установка KB.
- Actions — переход к действиям.

Если после респонса выполняется любой агент или KB, этому агенту должна быть присвоена соответствующая платформа.

Если КБ не предполагает последующий вызов агента, устанавливается платформа internal.

Actions

Список действий:

№	id	name	Назначение
1	'askParameter'	parameterId	ID факта
2	Discover Aux Intents		какой-то текст
3	Extract, Ectractor	`"data": { "source": "${system.fact.ui.usertext}", "method": "extract", "rule": { "fact": "001.examples.fact.`	Объект

№	id	name	Назначение
		1", "mode": "all", "allow_repeat": false, "extractor": "001.examples.RU.aux_intents", "field": "entity_name" } },	

4	Extract, Model	"data": {
		"source": "$
		{system.fact.ui.usertext}",
		"method": "model",
		"rule": {
		"fact": "ai.fact.a",
		"mode": "first",
		"model": "FAQ-skill-set",
		"threshold": 0.25
		}
		},
			Объект

5	Extract, Preset	"data": {
		"source": "$
		{system.fact.ui.usertext}",
		"method": "preset",
		"rule": {
		"fact": "ai.fact.a",
		"mode": "single",
		"allow_repeat": true,
		"preset": {
		"name": "datetime",
		"options": {
		"extrapolate_to":
		"future"{
		}
		}
		}
		},
			Объект

6	'setParameter'	parameterId	ID факта
3	'setParameter PlatformId'	platformId	ID платформы
4	'setPerceptron'	perceptronId	ID персептрона

№	id	name	Назначение
5	'clearParameter'	parameterId	ID факта
6	showInfo		какой-то текст ${ID факта} еще какой-то текст
7	type: "text"	data: "Hello world!"	строка

8	type: "table"	data: {
		title: "Name",
		table: [{
		"numbers": numbs,
		"date": dates,
		"true symbols": symbols,
		"work logic": logic,
		"results of smth": res
		}]
		}
			объект

Первичный вид:

```
return {
    actions: [{
                // используя choice
                action: "askParameter",
                parameterId: "[gp].payment_status.fact.num_contract",
                options: {
                    prompt: "Выберите интересующий договор по компании № ${[gp].payment_status.fact.compcode}",
                    type: "choice",
                    list: choice // массив элементов a = ['1', '2', '3', '4'];
                }
            }, {
                action: "setParameter",
                parameterId: '[gp].payment_status.fact.znp_number',
                value: number // значение let number = 1;
            }, {
                action: "setPerceptron",
                perceptronId:[gp].purchases.qualification.perceptron.faqRegisteredUsers'
            }, {
                action: "showInfo",
                type: "text",
                data: "Hello world!"
            }, {
                action: "clearParameter",
                parameterId: "[gp].payment_status.fact.num_contract"
            }, {
                action: "showInfo",
                type: "table",
```

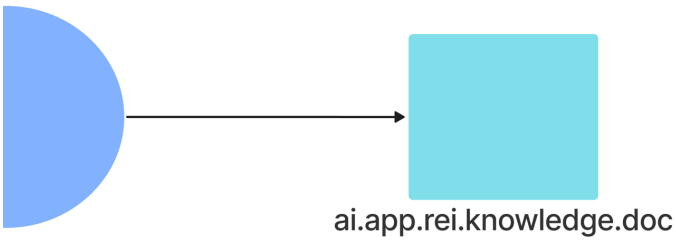
```
                data: {
                        title: "Name",
                        table: [{
                            "numbers": numbs,
                            "date": dates,
                            "true symbols": symbols,
                            "work logic": logic,
                            "results of smth": res
                        }]
                    }
            }];
}
```

, где `[gp]` — нотация обратного доменного имени.

Элемент базы знаний (Knowledge)

Knowledge (элемент базы знаний) — знание о необходимых действиях, желаемый результат диалога с пользователем.

Цель сессии в классическом SessionFlow]— переход в Knowledge, т.е. система должна понять, что нужно пользователю, и подобрать подходящий Knowledge. Персептрон пересчитывается до тех пор, пока не появится выход в Knowledge. Если после расчета персептрона появляются новые факты и Knowledge, то преимущество отдается Knowledge.



Если агент Dispatcher закончил работу выходом в Knowledge, то запускается агент SolverFinder. Агент SolverFinder запускает агент типа Solver.

Типы решения (solutionType)

- url — перенаправить пользователя на нужную страницу;
- none (он же agent) —запуск агента типа Solver;
- text —показать текст пользователю;
- html —показать пользователю HTML размеченный текст;
- document —дать пользователю ссылку на документ (см. надстройка Документы). Ссылка на документ может формироваться двумя способами:
    - для работы с файлом в текущей вкладке;
    - для работы с файлом вне текущей вкладки (другое окно или вкладка).

В трех последних случаях стартует специальный агент типа Solver DocumentFinder.

Follow Up

Follow Up — предложение выполнить действие по связанным темам.

К одному элементу базы знаний (Knowledge) можно привязать до 10 follow up. В зависимости от результата работы агента Solver цепочки follow up могут быть бесконечными.

Системный скилсет (System skillset)

Системный скилсет содержит:

- Системные факты, которые могут быть использованы в любом скилсете текущего инстанса.
- Основные экстракторы.
- Основные агенты.

> **Важно**
>
> Импортировать системный скилсет не рекомендуется.

Системные факты (System facts)

Список фактов, их ключи и значения:

№	ID	Название	Назначение	Триггерит событие system.change.event
1	system.fact.active_kb	ACTIVE_KB	ЭБЗ для солвера **Пример возможного значения:** ai.test.sage.test.case.knowledge.end	-
2	system.fact.calc_snapshot	PERCEPTRON_CALC_PARAMS	JSON фактов со значениями перед расчетом персептрона. Содержит список всех фактов, участвующих в расчете персептрона	-

№	ID	Название	Назначение	Триггерит событие system.change.event
			Пример возможного значения `{ "parameterId": "system.fact.session.user_obj", "value":"{ \"username\": \"a.grebenik\", \"email\": \"a.grebenik@mail.ru\", \"id\": \"6001b8215e6e710011c1c7fa\", \"isActive\":true, \"name\":\"Anna\", \"language\":\"RU\", \"lastPlatformsCheck\": \"03-02-2021 13:24:59\", \"roles\":[{ \"id\": \"5ff8257adc60400018c88d8e\", \"name\": \"OPERATOR\", \"created\": \"2021-01-08T09:27:22.652Z\", \"modified\": \"2021-01-08T09:27:22.652Z\"}, { \"id\": \"5ff8257adc60400018c88d8f\", \"name\":\"ADMIN\", \"created\": \"2021-01-08T09:27:22.653Z\", \"modified\": \"2021-01-08T09:27:22.653Z\"}], \"lastSession\": \"2021-01-18T08:47:57.858Z\"}" }, { "parameterId": "system.fact.ui.usertext", "value": "создать СО заказа" }`	
3	system.fact.custom_	CUSTOM_FAREWELLER_INVOKED	Устанавливается, если был настроен кастомный fareweller и он запустился вместо основного	-

№	ID	Название	Назначение	Триггерит событие system.change.event
	fareweller_invoked			
4	system.fact.endless_session	ENDLESS_SESSION	Фаревеллер не завершает сессию, а рестартует если есть такой факт	-
5	system.fact.faq_skillset	FAQ_SKILLSET	FAQ SPECIFIC **Пример возможного значения:** https://sandbox.dakub.ru/#/session/601c0de2055a4b001159a208	+
6	system.fact.fareweller_done	FAREWELLER_DONE	При установке в true указывает, что сессия завершена, и фаревеллер не работает больше **Пример возможного значения:** true	-
7	system.fact.is_faq_state	IS_FAQ_STATE	Устанавливается в fareweller влияет на поведение сессии в режиме FAQ **Пример возможного значения:** https://sandbox.dakub.ru/#/session/601c0de2055a4b001159a208	+
8	system.fact.md5hash	MD5_FACT_HASH	Хеш фактов при расчете перцептрона диспетчером **Пример возможного значения:** закодированный контекст сессии	-
9	system.fact.need_login	SOLVER_NEED_LOGIN	Используется в solver finder для запроса логина при старте сессии	+
10	system.fact.operator.mode	OPERATOR_MODE	При работе с оператором принимает два значения ask или waterfall **Пример возможного значения:** ask	+

№	ID	Название	Назначение	Триггерит событие system. change. event
			или waterfall (статус режима сообщений в чате с оператором)	
11	system.fact. operator.text	UI_OPERATOR	Факт, через который ведется разговор с пользователем в режиме waterfall **Пример возможного значения:** текст, который напишет оператор	+
12	system.fact. session.channel	SESSION_CHANNEL	Канал сессии (копия из свойства сессии uiChannel) **Пример возможного значения:** web-chat<	-
13	system.fact. session.context	CONTEXT	Хранит информацию о контексте сессии	+
14	system.fact.sessi on .context.last	CONTEXT_LAST	CONTEXT COLLECTOR AGENT **Пример возможного значения:** факты, участвующие в сессии в текущий момент	-
15	system.fact.sessi on. dialog.agent	SESSION_DIALOG _AGENT	Агент, с которым сейчас ведется диалог - ask от других агентов игнорируются **Пример возможного значения:** system.agent.fareweller	-
16	system.fact.sessi on. domain	SESSION_DOMAIN	Основная тема сессии (например skillsetId) **Пример возможного значения:** 001.test-examples	+
17		LOCAL_NAME	Имя пользователя для обращения	-

№	ID	Название	Назначение	Триггерит событие system.change.event
	system.fact.session.full_name		**Пример возможного значения:** user.name или user.username	
18	system.fact.session.id	SESSION_ID	id сессии (копия из свойства сессии id) **Пример возможного значения:** 01be2b1055a4b0011599c03	-
19	system.fact.session.language	SESSION_LANGUAGE	Язык сессии (копия из свойства сессии language) **Пример возможного значения:** RU	-
20	system.fact.session.platform_id	PLATFORM_ID	Текущая целевая платформа. по умолчанию internal копия свойства сессии platformId **Пример возможного значения:** internal	+
21	system.fact.session.platform_username	PLATFORM_USER_NAME	Проставляется после логина в платформу, если в свойствах UserPlatform есть username **Пример возможного значения:** username	+
22	system.fact.session.scope	SESSION_SCOPE	Подтема сессии **Пример возможного значения:** например faqTitle	+
23	system.fact.session.sub.finish	SESSION_SUB_FINISH	• установка в finish - создает и сохраняет новый инцидент если system.fact.session.sub.start не пустой	+

№	ID	Название	Назначение	Триггерит событие system.change.event
			• установка в cancel - очищает system.fact.session.sub.start сбрасывая инцидент **Пример возможного значения:** finish или cancel	
24	system.fact.session.sub.result	SESSION_SUB_RESULT	Хранит список инцидентов в виде JSON массива (по умолчанию): • skillSetId • domain • scope • duration **Пример возможного значения** `[{ "duration": 10, "skillSetId": "ai.app.root", "domain": "faq name", "scope": "faq title" }]`	+
25	system.fact.session.sub.start	SESSION_SUB_START	Хранит время начала инцидента, если установить в start - начинает новый инцидент (ставит текущее время) **Пример возможного значения:** start или время начала инцидента	+
26	system.fact.session.user_obj	USER_OBJ	json stringify(user)	-

№	ID	Название	Назначение	Триггерит событие system.change.event

			Пример возможного значения
			
			{
			"username": "a.grebenik",
			"email": "a.grebenik@mail.ru",
			"id": "6001b8215e6e710011c1c7fa",
			"isActive": true,
			"name": "Anna",
			"language": "RU",
			"lastPlatformsCheck": "дата последней проверки платформы",
			"roles": [список доступных ролей],
			"lastSession": "дата последней сессии через текущий канал"
			}	

№	ID	Название	Назначение	Триггерит событие system.change.event
27	system.fact.session.user_platforms	USER_PLATFORMS	Массив пользовательских платформ, список типов платформ, платформ и ролей, доступных пользователю **Пример возможного значения** `{ "platforms": [ { "value": "internal", "title": "Internal" } ], "platformTypes": [ { "value": "internal", "title": "Internal" } ], "roles": [ "OPERATOR", "ADMIN", "NONSTUDY", "_ALL" ] }`	-
28	system.fact.session.username	USER_NAME	Значение user.username имя пользователя **Пример возможного значения:** имя пользователя	-
29	system.fact.skillset	SKILLSET	Выбранный скиллсет **Пример возможного значения:** ai.test.sage.test.case — текущий скилсет, который определил рутовый скилсет	+
30	system.fact.solver.code	SOLVER_CODE	Результаты работы солвера	-

№	ID	Название	Назначение	Триггерит событие system.change.event
			Пример возможного значения: solved	
31	system.fact.solver.started	SOLVER_STARTED	Признак что солвер начал работу **Пример возможного значения:** system.agent.document_solver	-
32	system.fact.solver.variant	SOLVER_VARIANT	Результаты работы солвера **Пример возможного значения:** default	-
33	system.fact.ui.confirmstart	UI_CONFIRMSTART	Если у солвера есть признак confirm_start, то этот факт запрашивается у пользователя	+
34	system.fact.ui.fact.text	UI_FACT_TEXT	Продолжение вопроса по FAQ - через endless в fareweller	-
35	system.fact.ui.problem_solved	UI_PROBLEM_SOLVED	Факт, запрашиваемый у юзера в виде "Ваша проблема решена?"	+
36	system.fact.ui.usertext	UI_USERTEXT	Первый вопрос к пользователю при старте сессии **Пример возможного значения: текст пользователя**	+
37	system.fact.validation.error_count	VALIDATION_ERROR_COUNT	Количество ошибок валидации на факте **Пример возможного значения:** 2	-
38	system.fact.validation.processing	VALIDATION_ERROR_PROCESSING	Поведение при провале валидации	-

Системные агенты (System agents)

Перечень системных агентов:

№	ID	Тип	Назначение
1	system.agent.command_finder	internal	Разбирает команды /add_budget <счет> <сумма>
2	system.agent.dispatcher	internal	Считает перцептроны и ведет сессию
3	system.agent.document_solver	internal	Обрабатывает KB - показывает документы или тексты
4	system.agent.fareweller	internal	Прощается и завершает или перезапускает сессию
5	system.agent.greeter	internal	Здоровается и начинает сессию
6	system.agent.solver_finder	internal	Обрабатывает KB если найдены
7	system.agent.session.sub.finish	internal	Подписки: • system.event.session.sub.finish:* сохраняет новый инцидент • system.event.fact.change:system.fact.solver.code - при отработке солвера это событие позволяет поймать код солвера если solved - сохраняем новый инцидент иначе отменяем текущий инцидент • system.event.fact.change:system.fact.session.sub.finish если значение факта == finish - сохраняет новый инцидент если значение факта == cancel - отменяет текущий инцидент
8	system.agent.session.sub.start	internal	Подписки: • system.event.session.sub.start - при старте события начинает отсчет инцидента • system.event.fact.change:system.fact.ui.usertext - при старте события начинает отсчет инцидента • system.event.fact.change:system.fact.session.sub.start - если значение факта == start - начинает отсчет времени инцидента
9	system.agent.task_dispatcher	internal	

№	ID	Тип	Назначение
			Координирует межсессионные взаимодействия и отложенные задачи.
10	system.agent.test.executor	internal	Прогоняет тесты - делает сессию, мониторит события, делает answer и тп
11	system.agent.timeouter	internal	По умолчанию запускается один раз в минуту, проверяет время и статус каждой сессии и делает вывод о закрытии сессии. Если в течение определенного времени внутри сессии не происходят события, сессия закрывается по time out

Системные экстракторы

Системные экстракторы (NER-сервисы) — это комплекс сложных экстракторов для продвинутого извлечения информации из текстов пользователей, например, извлечение адресов, стран назначения, номеров документов и т.д. Экстракторы NER-сервисов созданы на базе библиотек Natasha и Yargy или при помощи regexp.

Все системные экстракторы находятся в системном скилсете. Протестировать их можно как из системного скилсета, путем выбора экстрактора из списка, либо в любом другом скилсете, путем ввода идентификатора на форме тестирования.

Описание экстракторов

№	Идентификатор	Описание и назначение
1	address	Извлекает адреса.

<table>
<tr><th>№</th><th>Идентификатор</th><th>Описание и назначение</th></tr>
<tr><td></td><td></td><td>

Пример ответа при успешном извлечении данных

Текст для экстракции: улица Ленина 5

<pre>
{
  "result": [
    {
      "extractor": "address",
      "entity_name": "улица",
      "entity_value": "Ленина",
      "__span": [
        0,
        14
      ]
    },
    {
      "extractor": "address",
      "entity_name": "name",
      "entity_value": "5",
      "__span": [
        0,
        14
      ]
    }
  ]
}
</pre>

</td></tr>
<tr><td>2</td><td>credit_card</td><td>

Извлекает номер кредитной карты. Примечание: извлекает 16 и 19 символьный номер карты.

Пример ответа при успешном извлечении данных

Текст для экстракции: 1234 5678 1234 5678

<pre>
{
  "result": [
    {
      "entity_value": "1234 5678 1234 5678",
      "__span": [
        0,
        19
      ],
      "extractor": "credit_card",
      "entity_name": "credit_card"
    }
  ]
}
</pre>

</td></tr>
<tr><td>3</td><td>country</td><td>Извлекает название страны.</td></tr>
</table>

№	Идентификатор	Описание и назначение
		> **Пример ответа при успешном извлечении данных** Текст для экстракции: Россия `{ "result": [ { "extractor": "country", "entity_name": "country", "entity_value": "Россия", "__span": [ 0, 6 ] } ] }`
4	country_from	Извлекает название страны по предлогам "из", "с". **Пример ответа при успешном извлечении данных** Текст для экстракции: из Москвы `{ "result": [ { "extractor": "country_from", "entity_name": "country_from", "entity_value": "Россия", "__span": [ 0, 9 ] } ] }`
5	country_in	Извлекает название страны по предлогам "в", "на".

№	Идентификатор	Описание и назначение
		Пример ответа при успешном извлечении данных Текст для экстракции: в Москву `{ "result": [ { "extractor": "country_in", "entity_name": "country_in", "entity_value": "Россия", "__span": [ 0, 8 ] } ] }`
6	dates	Извлекает даты из следующих сущностей: • Числа • 1-31го; 1-31 го; 1-31-го • 1-31ого; 1-31 ого; 1-31-ого • 1-31е; 1-31 е; 1-31-е • первого - тридцать первого числа • первое - тридцать первое число • 01-31 числа; 1-31 числа • Число + Месяц (комбинации должны распознаваться со следующими разделителями: " " "." "-") • 01-31.01-12; 1-31.1-12 • Любая комбинация из "Числа" + январь - декабрь (с правильным падежом) • 01-31 + январь - декабрь (с правильным падежом); 1-31 + январь - декабрь (с правильным падежом) • первого - тридцать + январь - декабрь (с правильным падежом) • первое - тридцать + январь - декабрь (с правильным падежом) • Число + Месяц + Год (комбинации должны распознаваться со следующими разделителями: " " "." "-") • 01-31.01.12.2022; 1-31.1-12.2022 - четырехзначный формат года • Любая комбинация из "Числа" + январь-декабрь + Любая комбинация из "Год"

<table>
<tr><th>№</th><th>Идентификатор</th><th>Описание и назначение</th></tr>
<tr><td></td><td></td><td>

1-31 + январь - декабрь + Любая комбинация из "Год"
01-31.01.12.22; 1-31.1-12.22 - двухзначный формат года

Год (если извлечен год от 0 до текущий год + 6, то добавляется в начало 20, если распознался текущий год + 7 до 99, то добавляется в начало 19)

1- 3000 год или г
первого - трех тысячного года или г

Пример ответа при успешном извлечении данных
Текст для экстракции: 31 января 2021
<pre>
{
  "result": [
    {
      "extractor": "dates",
      "entity_name": "year",
      "entity_value": 2021,
      "__span": [
        0,
        14
      ]
    },
    {
      "extractor": "dates",
      "entity_name": "month",
      "entity_value": 1,
      "__span": [
        0,
        14
      ]
    },
    {
      "extractor": "dates",
      "entity_name": "day",
      "entity_value": 31,
      "__span": [
        0,
        14
      ]
    }
  ]
}
</pre>
</td></tr>
<tr><td>7</td><td>digits</td><td>Извлекает число (дробная часть числа распознается как отдельное число).</td></tr>
</table>

№	Идентификатор	Описание и назначение

		Пример ответа при успешном извлечении данных
		Текст для экстракции: 10 раз к вам обращался
		<pre>{
		  "result": [
		    {
		      "entity_value": "10",
		      "__span": [
		        0,
		        2
		      ],
		      "extractor": "digits",
		      "entity_name": "digits"
		    }
		  ]
		}</pre>

8	foreign_passport	Извлекает номер иностранного паспорта.
		Примечание: От 6 до 12 символов (в номере могут быть буквы).
		Пример ответа при успешном извлечении данных
		Текст для экстракции: 123456
		<pre>{
		  "result": [
		    {
		      "entity_value": "123456",
		      "__span": [
		        0,
		        6
		      ],
		      "extractor": "foreign_passport",
		      "entity_name": "foreign_passport"
		    }
		  ]
		}</pre>

9	iccid	Извлекает уникальный серийный номер SIM-карты.

<table>
<tr><th>№</th><th>Идентификатор</th><th>Описание и назначение</th></tr>
<tr><td></td><td></td><td>

Пример ответа при успешном извлечении данных

Текст для экстракции: номер 8901260232714958936

<pre>{
  "result": [
    {
      "entity_value": "8901260232714958936",
      "__span": [
        6,
        25
      ],
      "extractor": "iccid",
      "entity_name": "iccid"
    }
  ]
}</pre>

</td></tr>
<tr><td>10</td><td>location</td><td>

Встроенный экстрактор в Natasha для извлечения объектов вроде "Швейцария" или "Академгородок".

Пример ответа при успешном извлечении данных

Текст для экстракции: я щас в академгородке

<pre>{
  "result": [
    {
      "extractor": "location",
      "entity_name": "name",
      "entity_value": "академгородок",
      "__span": [
        0,
        13
      ]
    }
  ]
}</pre>

</td></tr>
<tr><td>11</td><td>military_id</td><td>Извлекает номер военного билета.</td></tr>
</table>

№	Идентификатор	Описание и назначение
		Пример ответа при успешном извлечении данных Текст для экстракции: `{ "result": [ { "entity_value": "0999999", "__span": [ 0, 7 ], "extractor": "military_id", "entity_name": "military_id" } ] }`
12	money_range	Извлекает разброс стоимости (вместе с валютой).

№	Идентификатор	Описание и назначение

		Пример ответа при успешном извлечении данных
		Текст для экстракции: от 100 до 200 рублей
		{
		"result": [
		{
		"extractor": "money_range",
		"entity_name": "min_integer",
		"entity_value": "100",
		"__span": [
		0,
		20
		]
		},
		{
		"extractor": "money_range",
		"entity_name": "max_integer",
		"entity_value": "200",
		"__span": [
		0,
		20
		]
		},
		{
		"extractor": "money_range",
		"entity_name": "max_currency",
		"entity_value": "RUB",
		"__span": [
		0,
		20
		]
		}
		]
		}

13	money_rate	Извлекает сумму в период.

№	Идентификатор	Описание и назначение

		Пример ответа при успешном извлечении данных
		Текст для экстракции: 1000р в день
		{
		"result": [
		{
		"extractor": "money_rate",
		"entity_name": "money_integer",
		"entity_value": "1000",
		"__span": [
		0,
		12
		]
		},
		{
		"extractor": "money_rate",
		"entity_name": "money_currency",
		"entity_value": "RUB",
		"__span": [
		0,
		12
		]
		},
		{
		"extractor": "money_rate",
		"entity_name": "period",
		"entity_value": "DAY",
		"__span": [
		0,
		12
		]
		}
		]
		}

14	money	Извлекает количество денег и валюту (рубли, доллары и евро при условии, что валюта написана словами).

<table>
<tr><th>№</th><th>Идентификатор</th><th>Описание и назначение</th></tr>
<tr><td></td><td></td><td>

Пример ответа при успешном извлечении данных

Текст для экстракции: 100 долларов

<pre>
{
  "result": [
    {
      "extractor": "money",
      "entity_name": "integer",
      "entity_value": "100",
      "__span": [
        0,
        12
      ]
    },
    {
      "extractor": "money",
      "entity_name": "currency",
      "entity_value": "USD",
      "__span": [
        0,
        12
      ]
    }
  ]
}
</pre>

</td></tr>
<tr><td>15</td><td>msisdn</td><td>

Извлекает номер сотового телефона.
Примечания:

- Извлекает номера телефона, которые начинаются на +7, 7 и 8
- Извлекает номера телефона, которые написаны через " " и "-"

Пример ответа при успешном извлечении данных

Текст для экстракции: 7 926 426 0224

<pre>
{
  "result": [
    {
      "entity_value": "7 926 426 0224",
      "__span": [
        0,
        14
      ],
      "extractor": "msisdn",
      "entity_name": "msisdn"
    }
  ]
}
</pre>

</td></tr>
</table>

№	Идентификатор	Описание и назначение

16	names	Извлекает имена и фамилии.
		Примечания:
		• Экстрактор Извлекает двойную фамилию, если она написана через "-". Через проблем берется только первое значение.
		• Фамилии и Имена ищутся в словаре. Если значения нет или оно ошибочно находится в словаре, то необходимо корректировать словарь.
		Пример ответа при успешном извлечении данных
		Текст для экстракции: Иванов-Петров Андрей
		{
		"result": [
		{
		"extractor": "names",
		"entity_name": "last",
		"entity_value": "иванов-петров",
		"__span": [
		0,
		20
		]
		},
		{
		"extractor": "names",
		"entity_name": "first",
		"entity_value": "андрей",
		"__span": [
		0,
		20
		]
		}
		]
		}

17	numbers	Извлекает числа.

№	Идентификатор	Описание и назначение
		Пример ответа при успешном извлечении данных Текст для экстракции: 1 10 или 15 3 `{ "result": [ { "extractor": "numbers", "entity_name": "numbers", "entity_value": "1 10", "__span": [ 0, 4 ] }, { "extractor": "numbers", "entity_name": "numbers", "entity_value": "15 3", "__span": [ 9, 13 ] } ] }`
18	order	Извлекает значения "по убыванию" или "по возрастанию".

№	Идентификатор	Описание и назначение

		Пример ответа при успешном извлечении данных
		Текст для экстракции: расположить 1 4 7 2 по убыванию и возрастанию
		{
		"result": [
		{
		"extractor": "order",
		"entity_name": "order",
		"entity_value": "по убыванию",
		"__span": [
		20,
		31
		]
		},
		{
		"extractor": "order",
		"entity_name": "order",
		"entity_value": "по возрастанию",
		"__span": [
		34,
		45
		]
		}
		]
		}

19	organisation	Извлекает названия организаций (аббревиатуры и полные слова).
		Пример ответа при успешном извлечении данных
		Текст для экстракции: МГУ
		{
		"result": [
		{
		"extractor": "organisation",
		"entity_name": "name",
		"entity_value": "МГУ",
		"__span": [
		0,
		3
		]
		}
		]
		}

20	period	Извлекает значение, через которое будет что-то выполнено.

<table>
<tr><th>№</th><th>Идентификатор</th><th>Описание и назначение</th></tr>
<tr><td></td><td></td><td>

Пример ответа при успешном извлечении данных

Текст для экстракции: с 1 по 10 буду в отпуске

<pre>
{
  "result": [
    {
      "extractor": "period",
      "entity_name": "period",
      "entity_value": "10",
      "__span": [
        7,
        9
      ]
    }
  ]
}
</pre>

</td></tr>
<tr><td>21</td><td>person</td><td>

Извлекает должность и имя (нуждается в доработке).

Пример ответа при успешном извлечении данных

Текст для экстракции: Президент России Путин

<pre>
{
  "result": [
    {
      "extractor": "person",
      "entity_name": "position",
      "entity_value": "Президент России",
      "__span": [
        0,
        22
      ]
    },
    {
      "extractor": "person",
      "entity_name": "name_last",
      "entity_value": "Путин",
      "__span": [
        0,
        22
      ]
    }
  ]
}
</pre>

</td></tr>
<tr><td>22</td><td>persons</td><td></td></tr>
</table>

№	Идентификатор	Описание и назначение
		Комбинация из экстрактора "names" и нейросетевого экстрактора от natasha. Используются следующие этапы экстракции: • допустим, на входе имеем text = "обошлось Юлия" • если names находит в тексте имена, он делат слова данного имени с большой буквы • names отработал так себе и теперь имеем text = "Обошлось Юлия" • полученный текст попадает в нейронку natasha. Даже если слово написано с большой буквы, natasha скорее всего поймет что это все-таки не имя • получаем результат: [] (пустой массив сущностей, сущности не обнаружены) **Пример ответа при успешном извлечении данных** Текст для экстракции: ИваноВ пЕтр `{ "result": [ { "extractor": "nn_person", "entity_name": "person", "entity_value": "ИваноВ пЕтр", "__span": [ 0, 11 ] } ] }`
23	region	Извлекает название региона (только РФ).

№	Идентификатор	Описание и назначение
		Пример ответа при успешном извлечении данных Текст для экстракции: Тверская область `{ "result": [ { "extractor": "region", "entity_name": "region", "entity_value": "Тверская область", "__span": [ 0, 16 ] } ] }`
24	region_from	Извлекает название региона (только РФ) по предлогам "из", "с". **Пример ответа при успешном извлечении данных** Текст для экстракции: из Москвы `{ "result": [ { "extractor": "region_from", "entity_name": "region_from", "entity_value": "Москва", "__span": [ 0, 9 ] } ] }`
25	region_in	Извлекает название региона (только РФ) по предлогам "в", "на".

<table>
<tr><th>№</th><th>Идентификатор</th><th>Описание и назначение</th></tr>
<tr><td></td><td></td><td>

Пример ответа при успешном извлечении данных

Текст для экстракции: в Питер

<pre>
{
  "result": [
    {
      "extractor": "region_in",
      "entity_name": "region_in",
      "entity_value": "Санкт-Петербург",
      "__span": [
        0,
        7
      ]
    }
  ]
}
</pre>

</td></tr>
<tr><td>26</td><td>residence</td><td>Извлекает номер вида на жительство. От 7 цифр до 12.

Пример ответа при успешном извлечении данных

Текст для экстракции: серия 82 номер 1234567

<pre>
{
  "result": [
    {
      "entity_value": "1234567",
      "__span": [
        15,
        22
      ],
      "extractor": "residence",
      "entity_name": "residence"
    }
  ]
}
</pre>

</td></tr>
<tr><td>27</td><td>russian_passport</td><td>Извлекает серию и номер российского паспорта.</td></tr>
</table>

№	Идентификатор	Описание и назначение
		Пример ответа при успешном извлечении данных Текст для экстракции: 4615 123123 `{ "result": [ { "entity_value": "4615 123123", "__span": [ 0, 11 ], "extractor": "ai.app.extractors.extractor.RU.russian_passport", "entity_name": "ai.app.extractors.extractor.RU.russian_passport" } ] }`
28	seafarer_id	Извлекает номер удостоверения личности моряка. **Пример ответа при успешном извлечении данных** Текст для экстракции: 0999999 `{ "result": [ { "entity_value": "0999999", "__span": [ 2, 9 ], "extractor": "seafarer_id", "entity_name": "seafarer_id" } ] }`
29	units	Извлекает единицу измерения.

№	Идентификатор	Описание и назначение
		Пример ответа при успешном извлечении данных Текст для экстракции: 105 дм `{ "result": [ { "extractor": "units", "entity_name": "standard_unit", "entity_value": "дециметр", "__span": [ 4, 6 ] } ] }`
30	words	Извлекает отдельные слова из текста.

№	Идентификатор	Описание и назначение

		Пример ответа при успешном извлечении данных
		Текст для экстракции: Здесь три слова
		{
		"result": [
		{
		"entity_value": "Здесь",
		"__span": [
		0,
		5
		],
		"extractor": "words",
		"entity_name": "words"
		},
		{
		"entity_value": "три",
		"__span": [
		6,
		9
		],
		"extractor": "words",
		"entity_name": "words"
		},
		{
		"entity_value": "слова",
		"__span": [
		10,
		15
		],
		"extractor": "words",
		"entity_name": "words"
		}
		]
		}

Экстракторы NER-сервисов подключаются к сценарию аналогично простым экстракторам, т.е. вызываются методом agentHelper в коде JavaScript-агента:

await helper.extract(fact.value, 'extractorID', 'RU')

```
(() => { // Do not remove this line!

const country = _.find(sessionParameters, ["parameterId", "ai.app.country.fact.country"]);

return (async () => {
    try {
        let country_ext = await helper.extract(country.value, 'country', "RU") // вызов
экстрактора country
        let country_from = await helper.extract(country.value, 'country_from', "RU") // вызов
экстрактора country_from
        let country_in = await helper.extract(country.value, 'country_in', "RU") // вызов
экстрактора country_in

        await logger.write('country_ext', `${JSON.stringify(country_ext)}`) // для просмотра в
```

```
логах

        return {
            actions: [{
                action: "setParameter",
                parameterId: "ai.app.country.fact.result",
                value: "Ваш регион - " + country_ext
            }]
        }
    } catch (err) {
        await logger.write('error', JSON.stringify(err))
    }
})();

}); // Do not remove this line!
```

Вызов больше одного экстрактора

Для того, чтобы прогнать текст пользователя через несколько экстракторов сразу, используется используется метод run_many.

Сессия

Сессия рассматривается как один диалог между пользователем и цифровым помощником, ведущий к определенной цели. Если цель достигнута — диалог считается успешно завершенным, если нет — неудачно завершенным. Контекстом диалога в любой момент является набор значений фактов, собранных в одной сессии.

Каждая сессия имеет:

- userId — уникальное имя пользователя, запустившего сессию;
- uiChannel — канал, который пользователь использует для обращения к системе (веб, чат, VK Teams, Skype и т.д.);
- platformId — платформа/ы, которую в данный момент использует сессия;
- language — язык сессии;
- perceptronId — уникальное имя активного персептрона;
- debugLevel — уровень прохождения сессии (по умолчанию 0);
- ttl — таймаут сессии (значение конфига сервера);
- state — состояние сессии (описаны ниже);
- title — автоматически присваивается первый пользовательский текст (usertext);
- operatorId — id назначенного оператора (если есть);
- id сессии — уникальный номер сессии.
- summary — это объект свободной формы, в который можно в ходе сессии что-то складывать в зависимости от требований и нужд проекта. Для этого можно использовать доступные из js методы helper.getSessionSummary и helper.setSessionSummary. Агент просто использует это как временный буфер для своей внутренней логики. При желании summary также можно пользоваться для разных целей в рамках проекта.

Состояния сессии

- asking parameter — система запрашивает факт у пользователя;
- closed — сессия закрыта;
- opened — сессия активна;
- operator ask — оператор ждет ответа пользователя;
- waiting — сессия ждет назначения оператору.

Этапы сессии

- Приветствие и первичный вопрос.
- Определение намерения пользователя: одному намерению соответствует один набор умений (скилсет).
- Сбор фактов, необходимых для реализации цели в рамках определенного набора умений.

- Исполнение цели пользователя, т.е. получение информации или выполнение действия.
- follow up — предложение выполнить действие по связанным темам (опционально).
- Завершение сессии (оценка работы и прощание).

Завершение сессии

Для завершения сессии существует агент timeouter. Агент timeouter по умолчанию запускается один раз в минуту, проверяет время и статус каждой сессии и делает вывод о закрытии сессии. Если в течение определенного времени внутри сессии не происходят события, сессия закрывается по time out.

Endless сессия Сессия не всегда прямолинейна и имеет возможности возврата к любому моменту сессии с полным или частичным сохранением контекста диалога. Для этого в сессии устанавливается факт endless (setParameter - endless).

Сессия хранится в БД и содержит подчиненные списки сущностей:

- sessionParameter — значения фактов сессии;
- sessionEvent — события в сессии;
- sessionAgent — вызовы агентов;
- sessionListener — подписка на события;
- sessionResult — найденные KB;
- sessionAskedParameter — очередь вопросов (запросов фактов);
- sessionUI — UI история чата сессии.

Сбор метрик сессии (инциденты)

Это возможность отслеживать количество решенных вопросов (инцидентов) в рамках сессии.

Детали реализации

Новые факты:

- system.fact.session.sub.start — хранит время начала инцидента, если установить в start - начинает новый инцидент (ставит текущее время);
- system.fact.session.sub.finish — отвечает за очистку старта инцидента и создание нового;
- system.fact.session.sub.result — хранит список инцидентов в виде JSON массива (по умолчанию);
- system.fact.session.domain — основная тема сессии (например skillsetId);
- system.fact.session.scope - под тема сессии (например faqTitle).

При формировании записи system.fact.session.sub.result берем значения фактов из конфига sessionSubConfig (см. ниже) + duration это разница между временем в system.fact.session.sub.start и текущим временем в секундах.

Пример:

```
[
{ "duration": 10, "skillSetId": "ai.app.root", "domain": "faq name", "scope":  "faq title"}
]
```

Новые конфиги:

- statisticIncludeSubs
- sessionSubConfig

Новые события:

- system.event.session.sub.start
- system.event.session.sub.finish

Новые системные агенты:

- system.agent.session.sub.start
- system.agent.session.sub.finish

Новые методы AgentHelper:

- sessionSubStart()
- sessionSubFinish()
- sessionSubCancel()

Процессы:

1. Варианты начать новый инцидент:
   - system.event.fact.change usertext — при изменении usertext;
   - создать событие system.event.session.sub.start (await helper.triggerEvent или из дебага);
   - записывает в факт system.fact.session.sub.start значение «start» (можно setParameter на Response);
   - await helper.sessionSubStart().
2. Варианты завершить начатый инцидент:
   - создать событие system.event.session.sub.finish (await helper.triggerEvent или из дебага)
   - отработал solver и вернул result.code == solved (в том числе documentSolver);
   - записывает в факт system.fact.session.sub.finish значение «finish» (можно setParameter на Response);
   - await helper.sessionSubFinish().
3. Варианты отменить начатый инцидент:
   - записать в факт system.fact.session.sub.finish значение «cancel» (можно setParameter на Response);
   - await helper.sessionSubCancel();
   - clearParameter system.fact.session.sub.start (можно clearParameter на Response).

4. Варианты посмотреть инциденты:

   • Новое поле в статистике.

   • При экспорте статистики (новая закладка Subs в Excel).

Событие (Event)

Все процессы в рамках сессии управляются событиями. Любое изменение состояния сессии сопровождается регистрацией события.

При возникновении события в сессии:

1. Событие записывается в сессионный флоу (sessionEvent);
2. Запускаются агенты, подписанные на событие (sessionListener);
3. Обрабатываются результаты работы агентов.

Подписать агента на событие можно через форму в агентах или в подписках агентов.

При подписке можно указать следующие свойства:

- agentId — идентификатор агента;
- eventId — идентификатор события (например: **system.fact.change**)
- eventSuffix — конкретика по событию, при возникновении которого срабатывает агент;
    - подписка на **system.fact.change:*** запускает подписанного агента при изменении любого факта;
    - подписка на **system.fact.change:fact1** запускает подписанного агента при изменении факта fact1;
- params.sync — агент реагирует каждый раз при возникновении события. Возможные значения : true/false;
- params.once — агент реагирует на данное событие, возникшее впервые. Возможные значения : true/false;
- params.keepOnChangeContext — агент реагирует на смену контента. Возможные значения : true/false;
- onlyForUiChannels — добавляются UI каналы, на которые необходимо подписываться. Если указан не валидный канал или канал не указан, то подписка не сработает.