Синтаксис паттернов

Ассистент на русском распознаёт речь пользователя с помощью используемой системы распознавания. Результатом распознавания является обычный текст, из которого ассистенту необходимо понять, чего хочет пользователь.

Другими словами, необходимо выделить из текста все смысловые части и необходимые данные для правильной диспетчеризации запроса к одному из приложений или модулей самого ассистента.

Для этого каждое приложение должно предоставить по крайней мере одну *грамматику запросов*, в которой будет описано, на какие фразы пользователя нужно реагировать вашему приложению (с помощью *текстовых шаблонов*), какая команда и в каком контексте должна быть выполнена, и какие данные из текста нужно выделить для передачи на вход команде.

Синтаксис шаблонов (или паттернов) достаточно гибок для задания вариативности речи, воспринимаемой вашим приложением. Шаблоны содержатся в грамматике запросов, описываемых в виде xml-файлов. Подробнее о грамматиках читайте в специальном разделе.

Как работают паттерны

Каждый паттерн ставится в соответствие с одной определённой командой и содержит все возможные формулировки запроса на естественном языке, а также именованные placeholder'ы для извлечения параметров. Эти параметры затем передаются на точку входа в приложение (агент) в виде дерева разбора (аналог семантического дерева). Это дерево содержит все данные, определенные в паттерне в виде независимых от языка сущностей (Token).

Как преобразовать Token в данные

Чтобы преобразовать результат разбора фразы (Token) в данные некоторого типа (например, в `int` или экземпляр какого-либо класса) используются конвертеры. Для глобальных паттернов можно пользоваться уже имеющимися в API конвертерами, для собственных паттернов нужно реализовать свои.

Синтаксис

Синтаксис паттернов во многом схож с синтаксисом формальных грамматик систем распознавания речи. Например JSGF. Вложенность паттернов может быть сколь угодно глубокой.

В паттернах используются следующие конструкции:

Альтернативы

Определяет наличие в данной позиции текста одной из перечисленных конструкций или последовательности символов.

```
(первый|второй|первый или второй|$Number)
```

Символ `|` используется для разделения альтернативных конструкций. В данном примере указано, что в тексте должно присутствовать либо слово *первый*, либо слово *второй*, либо фраза целиком *первый или второй*, либо число в любом формате, определённое в паттерне с именем *Number*.

Альтернативы необходимо заключать в круглые скобки.

Опции

Определяет необязательное (опциональное) наличие некоторых конструкций в данной позиции в тексте.

```
[первый|второй|первый или второй|$Number]
```

Опции необходимо заключать в квадратные скобки.

Перестановки

Для задания вариативности речи без необходимости написания множества одинаковых паттернов можно использовать перестановки в тех местах, где последовательность слов может быть любой.

```
{как дела}
```

Перестановки заключаются в фигурные скобки. Символ `|` не используется для разделения конструкций, которые необходимо переставлять. Для этого можно использовать круглые скобки. Например, в случаях, когда переставляются целые фразы, в которых присутствует пробел.

В данном примере паттерн будет обрабатывать сразу две фразы - *как дела* и *дела как*.

Мусор

Означает, что в данной позиции в тексте может быть любое количество любых символов и слов, либо ничего. Для этого используется символ `*`.

```
меня зовут *
```

После обязательного словосочетания *меня зовут* может идти любая последовательность символов или слов, или вообще ничего.

Стемы

Символ `*` может также использоваться для изменения формы используемого слова.

```
красн*
```

Под этот вариант подойдут любые слова, начинающиеся на *красн* (красный, красные и т.д.). Можно указывать `*` как в начале, так и в конце слова. Можно в начале и конце одновременно.

Ссылка на паттерн

На ранее определенные в грамматике или на импортированные паттерны можно ссылаться с помощью указания имени паттерна и символа `$` перед ним.

```
меня зовут $Name
```

После обязательного словосочетания *меня зовут* должна идти обязательная конструкция, определенная в паттерне *Name*.

Синонимы

В паттерне можно сразу задать синоним для ссылки на другой паттерн. Используется два символа двоеточия `::`.

```
$MyNumber::Number
```

В дереве разбора для такого паттерна будет присутствовать токен *MyNumber* в который будет вложен токен *Number*. Такая конструкция удобна в случае использования одних и тех же паттернов в перестановках - для разделения паттернов по именам.

Отображение

Символ `:` обозначает отображение конструкции на некоторое строковое значение

```
(один:1|два:2|три:3)
```

В результатах разбора каждому слову будет поставлено в соответствие заданное значение. Например, если пользователь скажет *три*, то в дереве для этого паттерна будет значение *3*. Это удобно для абстрагирования текстовых данных от речи на каком-либо языке. Далее эти значения можно использовать, например, для конвертации в нужный формат.

Отображаемое значение не должно содержать пробелов!

Если необходимо указать значение для целой конструкции, а не для слова, то необходимо заключить конструкцию в круглые скобки. Например

```
(после завтра):day_after_tomorrow
```

Повторения

Последовательность одного и более паттерна можно указать с помощью конструкции `repeat`

```
repeat($Number)
```

Указывает, что в данной позиции текста должно присутствовать одно или более чисел. Например, подойдёт номер телефона *812 777 12 34*.

Сложные конструкции

Для реализации сложных конструкций с множеством альтернатив, отображений и имён, предлагается декомпозировать паттерн путём вложения одного паттерна в другой. О том, как это делается, читайте в описании синтаксиса грамматик запросов.