**СЕМАНТИЧЕСКИЙ ГЕНЕРАТОР**

**ОТВЕТОВ**

**ДЛЯ ДИАЛОГОВОГО МОДУЛЯ**

**РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ**

Одной из проблем, решаемых в процессе поиска ответа на заданный вопрос в Базе знаний является то, что

***подлежащее и сказуемое в вопросе и в массиве триад Базы знаний могут находиться соответственно в разных падежах и склонениях.***

Поэтому для проверки соответствия текстового выражения в вопросе с текстовыми выражениями, хранящимися в текстовом массиве Базы знаний, и получения в результате требуемого точечного ответа будут использоваться т. н. ***регулярные выражения* –** объекты, описывающие символьный шаблон.

Для построения *регулярного выражения* используется ***строка-образец – шаблон или маска, состоящая из символов и метасимволов и задающая правило поиска***. Для обработки текста задаётся строка замены, которая может содержать в себе дополнительно специальные символы.

Результатом работы с регулярным выражением может быть:

* ***проверка наличия искомого образца в заданном тексте;***
* ***определение подстроки текста, которая сопоставляется образцу;***
* ***определение групп символов, соответствующих отдельным частям образца.***

**ВСПОМОГАТЕЛЬНЫЕ ТЕКСТОВЫЕ МАССИВЫ**

1. **endings[]для хранения псевдоокончаний сказуемых**

**var endings =**

[

["ет", "(ет|ут|ют)"],

["ут", "(ет|ут|ют)"],

["ют", "(ет|ут|ют)"], //1 спряжение

["ит", "(ит|ат|ят)"],

["ат", "(ит|ат|ят)"],

["ят", "(ит|ат|ят)"], //2 спряжение

["ется","(ет|ут|ют)ся"],

["утся","(ет|ут|ют)ся"],

["ются","(ет|ут|ют)ся"], //1 спряжение, возвратные

["ится","(ит|ат|ят)ся"],

["атся","(ит|ат|ят)ся"],

["ятся","(ит|ат|ят)ся"], //2 спряжение, возвратные

["ен", "ен"],

["ена", "ена"],

["ено", "ено"],

["ены","ены"],

["ан", "ан"],

["ана", "ана"],

["ано", "ано"],

["аны","аны"],

["жен","жен"],

["жна","жна"],

["жно","жно"],

["жны","жны"] //краткие прилагательные

];

1. **blacklist[ ] – «черный» список подлежащих**

**var blacklist =**

["замена", "замены", "атрибут", "маршрут", "член", "нет"];

**ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ**

1. **Функция getEnding()определения сказуемых в вопросе по псевдоокончаниям**

*Для анализа сказуемого на совпадения его окончания с соответствующим ему псевдоокончанием в массиве псевдоокончаний* ***endings[]*** *определим вспомогательную функцию* ***getEnding(word)****, в которой вначале сразу же проверяется, не находится ли рассматриваемое слово* ***word*** *в списке исключений массива* ***blacklist[]*** *и поэтому его нужно исключить из дальнейшего рассмотрения:*

**function getEnding(word)**

**{**

//проверка слова на совпадение по черному спискув массиве **blacklist**

**if (blacklist.indexOf(word)!==-1) return -1;**

*Затем в цикле для всех записей в первом столбце массива endings производится проверка, не имеет ли это слово одно из псевдоокончаний, характерных для сказуемого:*

//перебор псевдоокончаний в массиве **endings**

**for (var j = 0; j < endings.length; j++)**

**{**

//проверка, оканчивается ли слово **word** на j-ое псевдоокончание

**if(word.substring(word.length-endings[j][0].length)==endings[j][0])**

//возврат номера найденного псевдоокончания для сказуемого

**return j;**

**}**

//если совпадений нет, то возврат -1

**return -1;**

**}**

1. **Функции small1() и big1()преобразования первых букв в тексте вопроса и ответа**

*Определим вспомогательные функции преобразования заглавных букв в строчные и наоборот, так как написание первых букв в слове учитывается при сравнении слов в вопросе и выборке слов из Базы знаний.*

*Функция, которая делает первую букву в тексте маленькой:*

**function small1(str)**

**{**

**return str.substring(0, 1).toLowerCase() + str.substring(1);**

**}**

*Функция, которая делает первую букву в тексте большой:*

**function big1(str)**

**{**

**return str.substring(0, 1).toUpperCase() + str.substring(1);**

**}**

**ФУНКЦИЯ СЕМАНТИЧЕСКОГО ГЕНЕРАТОРА getAnswer()**

Головная функция семантического генератора ответов на задаваемые к Базе знаний вопросы строится следующим образом.

**function getAnswer(question){**

*Параметром* ***question*** *для этой функции является текст вопроса, который вводится в текстовое окно вопроса диалогового окна разработанного в пользовательском интерфейсе на Web-странице сайта информационной системы.*

1. *В начале функции* ***getAnswer()*** *задаются переменные для фиксации успеха в поиске ответа на вопрос к БЗ и записи ответа на вопрос, который должен быть получен в результате выполнения функции:*

//флаг, найден ли ответ на вопрос

**var result = false;**

//формируемый функцией ответ на вопрос – вначале пустой

**var answer="";**

1. *Затем текст вопроса в параметре* ***question*** *готовится к обработке путем уменьшения первой буквы до прописной и отделения знаков препинания от слов вставкой между ними пробелов:*

//преобразование текста в параметре **question** вспомогательной функцией //**small1()**чтобы сделать первую букву в тексте вопроса прописной

**var txt = small1(question);**

//знаки препинания

**var separators = "'\",.!?()[]\\/";**

//добавление пробелов перед знаками препинания

**for (var i = 0; i < separators.length; i++)**

**txt = txt.replace(separators[i], " " + separators[i]);**

1. *После этого текст вопроса методом* **split()** *разбивается на отдельные слова, ориентируясь на все пробелы в тексте вопроса для его дальнейшего анализа, которые сохраняются в массиве* **words,** *который автоматически формируется из текста методом* **split()***.*

//массив слов и знаков препинания, отделенных пробелами

**var words = txt.split(' ');**

*Поскольку вопрос состоит из вопросного слова, сказуемого и подлежащего, то сказуемое оказывается в центре вопроса, а если сказуемое в вопросе будет найдено, то искомое подлежащее будет следовать за ним.*

1. *Теперь будем решать* з*адачу поиска сказуемого в вопросе, для чего выполним цикл, перебирающем все слова в предложении вопроса, записанные в массив* **words***. При этом текущее слово в массиве будет считаться сказуемым, если обладает характерным для сказуемых* ***псевдоокончанием.***

//перебор слов в массиве слов из вопроса

**for (var i = 0; i < words.length; i++)**

**{**

//поиск номера псевдоокончания с использованием вспомогательной //функции **getEnding()** сзаписью его в переменную **ending**

**var ending = getEnding(words[i]);**

*Если псевдоокончание будет найдено, а это эквивалентно возвращаемому значению функции* ***getEnding()*** *в виде номера в массиве, отличного от -1, то это сказуемое в вопросе, а подлежащее в вопросе будет следовать сразу после него:*

**if (ending >= 0)**

**{**

//замена псевдоокончания на набор возможных окончаний, хранящихся

//во втором столбце массива

**words[i] =**

**words[i].substring(0, words[i].length - endings[ending][0].length)**

**+ endings[ending][1];**

**}**

*Таким образом, найденное в массиве* ***words[i]*** *сказуемое в вопросе заменяется выражением* ***words[i],*** *но уже с набором соответствующих ему псевдоокончаний из второго столбца массива* ***endings[]****, что в конечном итоге превращает найденное в вопросе сказуемое в соответствующий ему шаблон – регулярное выражение для этого сказуемого.*

*Так например, окончания в сказуемых для глаголов 1-го спряжения -ет, -ут, -ют будут заменены на принятое в регулярных выражениях перечисление возможных вариантов строки «(ет|ут|ют)».*

*Наконец, для поиска сказуемого в записях БАЗЫ ЗНАНИЙ, соответствующего шаблону сказуемого в вопросе, необходимо сохранить полученный шаблон в переменную из класса* ***RegExp()****.*

//создание регулярного выражения для поиска по сказуемому из вопроса

**var predicate = new RegExp(words[i]);**

*В случае, если в роли сказуемых выступают не глаголы, а кратные прилагательные, имещие соответственно окончания типа –ен, -жен, -ена и т. п., которые в шаблоне в соответствии с таблицей псевдоокончаний сказуемых будут просто совпадать со словами с такими окончаниями, то в качестве сохраняемого значения для переменной* **predicate** *для обрабатываемого регулярного выражения необходимо захватить следующее слово за найденным в вопросе слова с окончанием, характерным для кратких прилагательных:*

**if (endings[ending][0] == endings[ending][1])**

**{**

**predicate = new RegExp(words[i] + " " + words[i + 1]);}**

**i++;**

**}**

*Когда сказуемое в вопросе найдено, слова, стоящие за сказуемым и образующие* ***подлежащее*** *с относящимися к нему дополнительными членами предложения, также превращаются в регулярное выражение, что позволяет не повторять дословно термины, используемые в исходном тексте.*

//создание регулярного выражения для поиска по подлежащему из вопроса

**var subject\_string = words.slice(i + 1).join(".\*");**

*При формировании регулярного выражения в подлежащем выполняется замена пробелов на принятое в регулярных выражениях обозначение произвольной последовательности символов «.\*».*

*Данное обозначение еще добавляется в начале и в конце регулярного выражения. В результате для вопроса:*

***«Как рассчитывается площадь квадрата»***

*после сказуемого будет сформировано регулярное выражение для подлежащего:*

***«.\*площадь.\*квадрата.\*»,***

*которое будет использовано при поиске ответа в базе знаний.*

*Будем кроме того считать в дальнейшем, что выражение для подлежащего в вопросе должно содержать больше трех символов:*

//только если в подлежащем больше трех символов

**if (subject\_string.length>3)**

**{**

**var subject = new RegExp(".\*" +subject\_string +".\*");**

*Полученные регулярные выражения используются при проходе по всем записям массива в базе знаний.*

*Ячейки* ***второго столбца*** *проверяются на соответствие регулярному выражению* ***predicate****, полученному из* ***сказуемого****.*

*Регулярное выражение* ***subject****, полученное из* ***подлежащего****, используется для проверки ячеек как* ***первого****, так и* ***третьего*** *столбцов в двумерном массиве базы знаний, поскольку предложения в вопросе и исходном тексте могут быть сформированы с противоположными по смыслу сказуемыми (“состоять из” <–> “входить в состав”).*

//поиск совпадений с шаблонами среди связей семантической сети

**for (var j = 0; j < knowledge.length; j++)**

**{**

**if (predicate.test(knowledge[j][1]) &&**

**(subject.test(knowledge[j][0]) || subject.test(knowledge[j][2])))**

**{**

//создание простого предложения из семантической связи

**answer+=big1(knowledge[j][0] + " " +**

**knowledge[j][1] + " " + knowledge[j][2] + ". ");**

**result = true;**

**}**

**}**

*Поскольку в вопросе может быть использовано сказуемое, синонимичное используемому в тексте, предусмотрен повторный проход по строкам базы знаний без проверки сказуемого.*

//если совпадений с двумя шаблонами нет,

**if (result == false){**

//поиск совпадений только с шаблоном подлежащего

**for (var j = 0; j < knowledge.length; j++)**

**{**

**if (subject.test(knowledge[j][0]) || subject.test(knowledge[j][2]))**

**{**

//создание простого предложения из семантической связи

**answer+=big1(knowledge[j][0] + " " + knowledge[j][1] + " " + knowledge[j][2] + ". ");**

**result = true;**

**}}}}}}**

//если ответа нет

**if(!result)answer = "Ответ не найден. <br/>";**

**}**