**СЕМАНТИЧЕСКИЙ ГЕНЕРАТОР**

**ОТВЕТОВ ДЛЯ ДИАЛОГОВОГО МОДУЛЯ**

**КОМПЬЮТЕРНОЙ ОБУЧАЮЩЕЙ СИСТЕМЫ**

**СТРУКТУРА ВОПРОСА**

Для диалога с базой знаний ограничимся вопросами, имеющих с точки зрения семантики предложения самую простую структуру:

**[вопросное слово] [сказуемое] [подлежащее]**

Например, для предложения в исходном тексте

*«Электрод состоит из металла Mz - (восстановленная форма системы) и ионов Mz+ в растворе (окисленная форма системы)».*

из которого нужно предварительно составить 4 простых предложения и занести в массив базы знаний 4 соответствующие им записи-триады

*«электрод состоит из металла Mz-»*

*«металл является восстановленной формой системы»*

*«электрод состоит из ионов Mz+ в растворе»*

*«ион является окисленной формой системы»*

можно составить 3 простых вопроса:

*«из чего состоит электрод»*

*«чем является металл»*

*«чем является ион»*

При этом вопросное слово (*что, чем, из чего,* ...) в вопросе при поиске ответа в базе знаний будет отбрасываться.

**ПОИСК ОТВЕТА НА ВОПРОС В БАЗЕ ЗНАНИЙ**

В заданном вопросе нужно прежде всего найти ***сказуемое***, так как ключевым моментом при поиске ответа на вопрос в базе знаний будетявляться поиск в *Базе знаний* **сказуемого,** соответствующего **сказуемому** в *вопросе*.

В целом, оптимизация процесса поиска ответа на вопрос основана на том, что типов сказуемых в исходном тексте всегда гораздо меньше, чем типов подлежащих (объектов, т. е. ключевых слов). Поэтому выбранных по совпадению сказуемых из *Базы знаний* записей всегда будет гораздо меньше, чем в исходной *Базе знаний*. На следующем же этапе из отобранных по совпадению сказуемых в вопросе и Базе знаний записях-триадах производится дополнительный отбор по совпадению уже **подлежащего** в *вопросе* с **подлежащим** в записях-триадах, отобранных на предыдущем этапе из *Базы знаний*.

Такой **двойной отбор** в итоге приводит к получению в диалоге, как правило, **точечного ответа** на заданный вопрос, т. е. выборке из *Базы знаний* только одной-единственной триады, из которой и строится **точный ответ на вопрос** в виде отдельного предложения.

**ИСПОЛЬЗОВАНИЕ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ**

В общем случае в **вопросе** *подлежащее* и *сказуемое*, для которых необходимо найти соответствие в *Базе знаний*, могут находится в неподходящих падежах и склонениях, в которых они хранятся в базе знаний.

Поэтому соответствие *сказуемых* и *подлежащих* в **вопросе** и **Базе знаний** определяется **не напрямую**, как их точное соответствие, а по некому **шаблону**, т. н. **регулярному выражению**, которые строятся на основе сказуемого и подлежащего в вопросе.

После получения регулярных выражений для сказуемого и подлежащего именно они используются в цикле перебора записей базы знаний для проверки их соответствия вопросу.

Таким образом, при помощи ***регулярного выражения*** для ***сказуемого*** проверяются ячейки **второго** столбца двумерного массива базы знаний, а при помощи ***регулярного выражения*** для ***подлежащего*** – последовательно ячейки **первого** и **третьего** столбцов базы знаний, т. к. русский язык допускает перестановку слов.

Например, следующие два предложения будут полностью эквивалентны:

***Металлы или полупроводники с электронной проводимостью являются проводниками первого рода***

и

***Проводниками первого рода являются металлы или полупроводники с электронной проводимостью***

На третьем этапе, *если совпадений по* ***подлежащему*** *и* ***сказуемому*** *вместе не найдено*, то выполняется поиск ***только по подлежащему***, чтобы учесть возможность использования в вопросе сказуемого, ***синонимичного*** (совпадающему по смыслу) записанному в базе знаний.

*Однако в этом случае фактически будет выполняться уже* ***поисковый запрос*** *ключевого понятия в базе знаний и, как правило, ответ на такой вопрос будет* ***не точечным****, а состоять из совокупности предложений, соответствующих тем записям в базе знаний, в которых встречается подлежащее из вопроса, но с разными сказуемыми, которые, в свою очередь, могут быть и несиномичными сказуемому в вопросе, что приведет к получению нескольких ответов, среди которых окажется и правильный, но без прямого указания на него.*

**РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ**

Для проверки соответствия текстового выражения в вопросе с текстовыми выражениями, хранящимися в двумерном текстовом массиве БАЗЫ ЗНАНИЙ, и получения в результате требуемого точечного ответа будут использоваться т. н. **регулярные выражения –** объекты, описывающие символьный шаблон.

*Регулярные выражения* используются для поиска и обработки подстрок в тексте, основанный на использовании метасимволов.

Для поиска 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.*

**КЛАССЫ И ФУНКЦИИ JavaScript**

**для работы с регулярными выражениями**

Для разработки программных модулей диалога с БАЗОЙ ЗНАНИЙ будем использовать следующие классы, методы и функции языка программирования JavaScript для работы с регулярными выражениями:

**КЛАССЫ:**

Класс **RegExp** представляет **регулярные выражения** - объекты, описывающие символьный шаблон.

Класс **String** определяет методы, использующие регулярные выражения для выполнения поиска по шаблону и операций поиска в тексте с заменой.

**ФУНКЦИИ:**

**split –** возвращает ***массив элементов***, полученных из исходной строки (если разделитель – пустая строка, т.е. пробел, то возвращается одномерный массив из всех символов строки).

**substring –** возвращает ***подстроку*** исходной строки, начальный и конечный индексы которого указываются параметрами (если параметр один, отбрасывает все символы до указанного индекса).

**slice -** возвращает ***подстроку*** исходной строки, начальный и конечный индексы которой указываются параметрами, за исключением последнего символа.

**МЕТОД:**

**test –** метод, используемый для ***проверки***, соответствует ли строка проверяемому регулярному выражению.

**ВСПОМОГАТЕЛЬНЫЕ ТЕКСТОВЫЕ МАССИВЫ**

**И ПРОГРАММНЫЕ МОДУЛИ**

*В целом, для организации диалога с информационной системой на основе ее БАЗЫ ЗНАНИЙ в виде текстового двумерного массива, состоящего из 3-х столбцов, необходимо* ***предварительно*** *создать:*

* *несколько* ***дополнительных текстовых массивов;***
* *совокупность* ***программных модулей*** *с функциями и методами, обрабатывающих текстовые данные в вопросе и записях-триадах БАЗЫ ЗНАНИЙ.*

**ТАБЛИЦА**

**псевдоокончаний сказуемых**

Соответствие **сказуемых** в БАЗЕ ЗНАНИЙ и **сказуемых**, стоящих в *середине вопроса*, будем определять по совпадению в них наборов последних букв – ***псевдоокончаний,*** поскольку в роли *сказуемых* чаще всего выступают определенные части речи: ***глаголы*** и ***краткие причастия/прилагательные.***

Такая возможность возникает благодаря тому, что в русском языке набор возможных псевдоокончаний сказуемых достаточно ограничен и весь целиком приводится в следующей таблице:

|  |  |  |
| --- | --- | --- |
| **Часть речи** | **Форма** | **Псевдоокончание** |
| глагол I спряжения | 3 лицо, ед. число | -ет |
| глагол I спряжения | 3 лицо, мн. число | -ут |
| глагол I спряжения | 3 лицо, мн. число | -ют |
| глагол II спряжения | 3 лицо, ед. число | -ит |
| глагол II спряжения | 3 лицо, мн. число | -ат |
| глагол II спряжения | 3 лицо, мн. число | -ят |
| глагол I спряжения | 3 лицо, ед. число, возвратный | -ется |
| глагол I спряжения | 3 лицо, мн. число, возвратный | -утся |
| глагол I спряжения | 3 лицо, мн. число, возвратный | -ются |
| глагол II спряжения | 3 лицо, ед. число, возвратный | -ится |
| глагол II спряжения | 3 лицо, мн. число, возвратный | -атся |
| глагол II спряжения | 3 лицо, мн. число, возвратный | -ятся |
| краткое прилагательное | муж. род, ед. число | -ан |
| краткое прилагательное | жен. род, ед. число | -ана |
| краткое прилагательное | ср. род, ед. число | -ано |
| краткое прилагательное | мн. число | -аны |
| краткое прилагательное | муж. род, ед. число | -ен |
| краткое прилагательное | жен. род, ед. число | -ена |
| краткое прилагательное | ср. род, ед. число | -ено |
| краткое прилагательное | мн. число | -ены |
| краткое прилагательное | муж. род, ед. число | -жен |
| краткое прилагательное | жен. род, ед. число | -жна |
| краткое прилагательное | ср. род, ед. число | -жно |
| краткое прилагательное | мн. число | -жны |

**ДОПОЛНИТЕЛЬНЫЕ МАССИВЫ**

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

**Таблицу псевдоокончаний** будем хранить в виде следующего ***текстового двумерного массива***, состоящего из следующих *записей в два столбца*:

**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

];

**ДОПОЛНЕНИЕ**

**в текстовый массив endings**

Для обработки достаточно часто встречающегося в диалоге вопроса:

**«что такое ААААА?»**

где используется «псевдосказуемое» **такое**, в БАЗЕ ЗНАНИЙ должен соответствовать используемый в русском языке оборот речи «псевдосказуемое» **– это**, что приводит к ответу типа:

**«ААААА – это ...»**

Таким образом, в текстовый массив для хранения псевдоокончаний **endings** необходимо добавить и следующую строку:

**var endings =**

[

… ,

["такое"," – это"] ,

]

Например, для ответа на вопрос:

**«что такое электрохимия?»**

в текстовом массиве базе знаний будет выбрана строка:

**var knowledge =**

**[**

**...**

**[ "электрохимия",**

**"- это",**

**"раздел химической науки, исследующий процессы протекании электрического тока в проводниках." ],**

…

**]**

Обработка такого вопроса должна привести к следующему ответу:

**Электрохимия – это раздел химической науки, исследующий процессы протекании электрического тока в проводниках.**

1. **МАССИВ blacklist[]исключений при поиске сказуемого в базе знаний**

В используемой методике обработки вопросов на основе *псевдоокончаний сказуемых* приходится учитывать слова в БАЗЕ ЗНАНИЙ, которые *не являются сказуемыми*, но имеют ***совпадающие со сказуемыми окончания*** и которые будут распознаваться при анализе *как сказуемые по ошибке*.

Для исключения такой ситуации необходимо создать отдельный ***одномерный текстовый массив*** со словами в базе знаний с совпадающими окончаниями и использовать его при анализе содержимого базы знаний для исключения таких слов из рассмотрения при обработке вопросов в режиме диалога.

Например, при обработке псевдоокончаний сказуемых: **-на, -ны, -ут, -ен, -ет** в такой текстовый массив должны быть включены слова, которые имеют те же окончания и встречаются в обрабатываемом текстовом массиве базы знаний (список элементов массива должен быть заполнен ***по всему текстовому массиву*** БАЗЫ ЗНАНИЙ, созданному по исходному тексту):

**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);**

**}**

**СЕМАНТИЧЕСКИЙ ГЕНЕРАТОР**

**ответов на вопросы к базе знаний**

**ГЛАВНАЯ ФУНКЦИЯ ГЕНЕРАТОРА getAnswer()**

**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 из параметра функции **question** 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,** *автоматически формируемом 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 считается 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()** запись 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****.*

*Для поиска сказуемого в БАЗЕ ЗНАНИЙ с учетом возможности использования различных форм сказуемого в вопросе и в исходном тексте, хранящемся в БАЗЕ ЗНАНИЙ, его нужно преобразовать в регулярное выражение путем замены найденного ранее псевдоокончания на набор псевдоокончаний, встречающихся во всех допустимых формах, для чего используется экземпляр класса* ***RegExp()****.*

//создание регулярного выражения для поиска по сказуемому из вопроса

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

*В предложениях, не являющихся вводными, для глаголов, как правило, используются формы третьего лица. Так, окончания глаголов 1-го спряжения -ет, -ут, -ют будут заменены на принятое в регулярных выражениях перечисление возможных вариантов строки «(ет|ут|ют)».*

//для кратких прилагательных нужно захватить следующее за найденным //слово

**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, 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» будет сформировано регулярное выражение «.\*площадь.\*квадрата.\*», которое будет использовано при поиске ответа в базе знаний.*

*Будем 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/>";**

**}**