**КОНТРОЛЬНЫЕ ВОПРОСЫ**

**ПО ПРОГРАММИРОВАНИЮ ДИАЛОГА С БАЗОЙ ЗНАНИЙ(А БОЛЬШЕ С ГУРИНЫМ НИКТО ДИАЛОГ И НЕ ЗАВОДИТ((( )**

1. Структура массива псевдоокончаний, используемого для поиска сказуемых в вопросе.

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

["такое"," – это"]

];

1. Содержание массива «черный список» слов в вопросе, не являющихся сказуемыми.

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 одно-мерный 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 информационной системы):

|  |
| --- |
| var blacklist = ["замена", "замены", "атрибут", "маршрут", "член", "нет"]; |

1. Структура функции поиска номера слова в вопросе с псевдоокончанием сказуемого.

Для анализа сказуемого на совпадения его окончания с соответствующим ему псевдоокончанием в массиве псевдоокончаний endings[] определим вспомогательную функцию getEnding(word), 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. Структура функций для преобразования первых букв в тексте вопроса и ответа

Функция big()для преобразования первой буквы в тексте ответа в заглавную

function big(str)

{

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

}

1. Процедура учета знаков препинания в предложении вопроса.

НАЧИНАЯ ОТСЮДА И ДО КОНЦА ИДЕТ ОДНА БИГ ФУНКЦИЯ, КТО ЗАХОЧЕТ НАПИСАТЬ ФУЛЛ, ПРОСТО БЕРИТЕ ИЗ ВОПРОСОВ НИЖЕ.

function getAnswer(question) {

let text = question.toLowerCase();

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

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

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

…

1. Процедура разбиения текста вопроса на отдельные слова методом split().

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

var words = text.split(' ');

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

…

1. Процедура поиска сказуемого в вопросе и формирования для него регулярного выражения

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

        var ending = getEnding(words[i]);

        var predicate;

        if (ending >= 0) {

            words[i] = words[i].slice(0, -endings[ending][0].length) + endings[ending][1];

            predicate = new RegExp(words[i]);

…

1. Процедура формирования регулярного выражения для подлежащего в вопросе методами slice() и join()

slice - возвращает подстроку исходной строки, начальный и ко-нечный индексы которой указываются параметрами, за ис-ключением последнего символа.

…

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

…

1. Процедура формирования предложения для ответа на вопрос из базы знаний методом test()

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

join - объединяет все элементы массива в строку с указанным разделителем как параметром.

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

            if (subject\_string.length > 3) {

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

                for (var j = 0; j < knowledgeBase.length; j++) {

                    if (predicate.test(knowledgeBase[j][1].toLowerCase()) &&

                        (subject.test(knowledgeBase[j][0].toLowerCase()) || subject.test(knowledgeBase[j][2].toLowerCase()))) {

                        answer += small(knowledgeBase[j][0]) + " " + knowledgeBase[j][1] + " " + knowledgeBase[j][2] + ". ";

                        result = true;

                    }

                }

            }

        }

    }

1. Процедура и код формирования ответа на вопрос с синомичными сказуемыми

if (!result) {

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

        if (subject\_string.length > 3) {

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

            for (var j = 0; j < knowledgeBase.length; j++) {

                if (subject.test(knowledgeBase[j][0].toLowerCase()) || subject.test(knowledgeBase[j][2].toLowerCase())) {

                    answer += small(knowledgeBase[j][0]) + " " + knowledgeBase[j][1] + " " + knowledgeBase[j][2] + ". ";

                    result = true;

                }

            }

        }

    }

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

    return answer;

}

1. Возможности библиотеки jQuery для организации мультимедийного диалога с Базой знаний симулятора установки на Web-странице.

Диалоговый пользовательский интерфейс с Базой знаний на Web-страницах информационной системы представляет собой использование интерактивных всплывающих диалоговых окон по запросам пользователя на Web-страницах. Для создания интерактивных сайтов чаще всего используется библиотека JavaScript jQuery.

Библиотека jQuery реализует архитектуру на основе концепций, положенных в основу языка разметки HTML и языка каскадных таблиц стилей CSS, со следующими базовыми возможностями:

* может обращаться к любому элементу объектной модели Web-документа DOM и предлагает механизм селекторов;
* может работать с событиями;
* осуществляет различные визуальные эффекты с изменением внешнего вида страницы с использованием CSS;
* имеет различные JavaScript-плагины, предназначенные для создания элементов пользовательских интерфейсов, например, диалоговых окон.

1. Общая структура сайта и разметки диалоговой Web-страницы для создания мультимедийного диалога с Базой знаний симулятора установки.

Для реализации удобного диалога с Базой знаний, содержащей мультимедийную информацию, на диалоговой Web-странице необходимо разработать интерфейс с диалоговыми окнами:

В папку сайта добавить файлы:

* библиотеку jQuery – файл jquery.js,
* файл разработанного в ЛР по Теме 12 программного кода на языке JavaScript для обработки диалога, например, с именем dialog.js с функцией формирования диалогового окна, например, с именем dialog\_window();
* файл внешней таблицы разработанных для диалога CSS-стилей, сохраненного, например, с именем style.css.

В HTML-разметку диалоговой Web-страницы добавить следующие элементы:

* строку с тегом <script> для подключения файла библиотеки jQuery jquery.js;
* строку с тегом <script> для подключения аудио сервиса Yandex;
* строку с тегом <script> для подключения внешнего файла с программным кодом JavaScript dialog.js, в который добавить код, обеспечивающий интерфейс на основе диалогового окна;
* строку с тегом <link> для подключения внешней таблицы CSS-стилей из файла style.css;
* в теге <body> загрузить оператором onLoad() созданную дополнительно функцию организации интерфейса для диалогового окна для Web-страницы, например, с именем dialog\_window().

Таким образом, общая структура Web-страницы с мультимедийным диалогом может выглядеть следующим образом:

<html lang="ru" xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Диалог с базой знаний</title>

<meta charset="utf-8" />

<script src="jquery.js"></script>

<script src="dialog.js"></script>

<link href="style.css" type="text/css" rel="stylesheet"/>

</head>

<body onLoad="dialog\_window()">

…

</body>

</html>

1. Программные модули, добавляемые в файл генератора ответов dialog.js для функционирования мультимедийного диалога с Базой знаний на Web-странице.

Для создания соответствующего интерфейса мультимедийного диалога в программный код файла dialog.js семантического генератора ответов из Базы знаний необходимо добавить код для создания диалогового окна и программных модулей для функционирования его интерфейса – флаговую переменную для активизации диалога и три функции: 1) функцию формирования диалогового окна; 2) функцию выдвижения-скрытия диалогового окна; 3) функцию вывода ответа на заданный вопрос в диалоговом окне.

1. Назначение и общая структура функции dialog\_window() для создания диалогового окна

var dialogOn = false;

function dialog\_window(){

Добавляем на страницу диалоговое окно как новый раздел div с идентификатором dialog, задаем ему CSS-стиль dialog, располагающий этот раздел справа от основного раздела страницы, и затем надвигаем его левое поле на страницу на 25px:

document.body.innerHTML+=

"<div id='dialog' class='dialog'>"

+"<div class='label' onclick='openDialog()'>Нажми, чтобы спросить!</div>"

+"<div class='header'>История:</div>"

+"<div class='history' id='history'></div>"

+"<div class='question'><input id='Qdialog' placeholder='Введите вопрос'/> <br><button onclick='ask(\"Qdialog\")'>Спросить</button></div>"

+"</div>";

1. Структура и стиль оформления общего блока div для диалогового окна.

.dialog{

position:fixed;

top:10%;

left:100%;

width:800px;

height:800px;

margin-left:-30px;

margin-top:-25px;

padding:10px;

border:#000 2px solid;

border-radius:10px;

background:#fff;

color:#f00;

z-index:1500;

font-size:12pt;

}

1. Структура и стиль оформления блока div для полосы с кнопкой «Нажми, чтобы спросить» для запуска функции выдвижения диалогового окна на центр страницы.

.dialog .label{

transform: rotate(270deg);

width:430px;

height:20px;

text-align:center;

color:#f00;

overflow:hidden;

display:inline-block;

margin-left:-26%;

margin-top:50%;

cursor:pointer;

}

1. Структура функции openDialog() плавного выдвижения-закрытия диалогового окна поверх Web-страницы.

function openDialog(){

if(dialogOn){ //анимация закрытия диалогового окна

$("#dialog").animate({"margin-left":"-30px"},1000,function(){});

dialogOn=false;

}

else{ //анимация открытия диалогового окна

$("#dialog").animate({"margin-left":"-1250px"},1000,function() {});

dialogOn=true;

clearInterval(timer);

}

1. Структура и стиль оформления блока div для выдачи ответа на вопрос в окно истории ответов.

.dialog .history .answer{

margin:5px;

width:100%;

border-radius: 5px;

padding:5px;

margin-right:40px;

background:#9ff;

}

.dialog .history .answer img, .dialog .history .answer audio, .dialog .history .answer video {

cursor: pointer;

display: block;

max-width: 650px;

}

1. Структура и стиль оформления блока div для поля формы input и кнопкой «Спросить» для запуска функции получения ответа из Базы знаний и выдачи его в блок ответа диалогового окна.

.dialog input, .dialog button, .dialog .history{

margin:2px 25px;

width:85%;

border:1px #666 solid;

border-radius: 5px;

font-size:20px;

}

.dialog .history .question {

margin:5px;

width:100%;

border-radius: 5px;

padding:5px;

margin-left:40px;

background:#99f;

}

1. Общая структура функция ask() обработки вопроса и выдачи ответа в диалоговое окно.

function ask(questionInput){

//переменная для считывания содержания окна ввода вопроса:

var question=document.getElementById(questionInput).value;

//активизация диалога

dialogOn=true;

//создаем переменную и сохраняем в ней тег <div>

var newDiv=document.createElement("div");

//задаем класс оформления созданного блока

newDiv.className='question';

//заполняем созданный блок текстом вопроса

newDiv.innerHTML=question;

//вставляем созданный блок в блок <history> и закрываем его

document.getElementById("history").appendChild(newDiv);

+"</div>";

//создаем еще один блок <div>

newDiv=document.createElement("div");

//задаем класс оформления созданного блока

newDiv.className='answer';

//наполняем созданный блок ответом, полученным от функции getAnswer()

newDiv.innerHTML=getAnswer(question);

//ОЗВУЧИВАНИЕ ПОЛУЧЕННОГО ОТВЕТА

//добавляем в ответ тег аудио, ссылающийся на звук от синтезатора Yandex

newDiv.innerHTML+="<audio controls='true' autoplay='true'

//обращение к голосовому сервису Yandex

src='http://tts.voicetech.yandex.net/

//в котором указывается формат звука и язык озвучиваемого текста generate?format=wav&lang=ru-RU&

//а также ключ доступа к SpeechKit Cloud API

key=4a4d3a13-d206-45fc-b8fb-e5a562c9f587&

//озвучиваемый текст, который берется из сгенерированного ответа

text="+newDiv.innerText+"'></audio>";

//вставка зввукового блока в блок <history> и закрываем его

document.getElementById("history").appendChild(newDiv);

+"</div>";

//запуск звука

if(newDiv.lastChild.tagName=="AUDIO"){newDiv.lastChild.play();}

//КОНЕЦ ОЗВУЧИВАНИЯ ОТВЕТА

//прокрутка в окне истории ответа в самый низ

document.getElementById("history").scrollTop =

document.getElementById("history").scrollHeight;

//очистка текстового поля для ввода нового вопроса

document.getElementById(questionInput).value="";

}