|  |
| --- |
| Оглавление  [1. Структура массива псевдоокончаний, используемого для поиска сказуемых в вопросе. 2](#_Toc185782066)  [2. Содержание массива «черный список» слов в вопросе, не являющихся сказуемыми. 2](#_Toc185782067)  [3. Структура функции поиска номера слова в вопросе с псевдоокончанием сказуемого. 2](#_Toc185782068)  [4. Структура функций для преобразования первых букв в тексте вопроса и ответа 3](#_Toc185782069)  [5. Процедура учета знаков препинания в предложении вопроса. 3](#_Toc185782070)  [6. Процедура разбиения текста вопроса на отдельные слова методом split(). 3](#_Toc185782071)  [7. Процедура поиска сказуемого в вопросе и формирования для него регулярного выражения 4](#_Toc185782072)  [8. Процедура формирования регулярного выражения для подлежащего в вопросе методами slice() и join() 4](#_Toc185782073)  [9. Процедура формирования предложения для ответа на вопрос из базы знаний методом test() 5](#_Toc185782074)  [10. Процедура и код формирования ответа на вопрос с синомичными сказуемыми 5](#_Toc185782075)  [11. Возможности библиотеки jQuery для организации мультимедийного диалога с Базой знаний симулятора установки на Web-странице. 6](#_Toc185782076)  [12. Общая структура сайта и разметки диалоговой Web-страницы для создания мультимедийного диалога с Базой знаний симулятора установки. 6](#_Toc185782077)  [13. Программные модули, добавляемые в файл генератора ответов dialog.js для функционирования мультимедийного диалога с Базой знаний на Web-странице. 7](#_Toc185782078)  [14. Назначение и общая структура функции dialog\_window() для создания диалогового окна 7](#_Toc185782079)  [15. Структура и стиль оформления общего блока div для диалогового окна 7](#_Toc185782080)  [16. Структура и стиль оформления блока div для полосы с кнопкой «Нажми, чтобы спросить» для запуска функции выдвижения диалогового окна на центр страницы 9](#_Toc185782081)  [17. Структура функции openDialog() плавного выдвижения-закрытия диалогового окна поверх Web-страницы 10](#_Toc185782082)  [18. Структура и стиль оформления блока div для выдачи ответа на вопрос в окно истории ответов 10](#_Toc185782083)  [19. 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 10](#_Toc185782084)  [20. Общая структура функция ask() обработки вопроса и выдачи ответа в диалоговое окно 11](#_Toc185782085) |

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

В русском языке окончания сказуемых часто зависят от спряжения, возвратности, рода и числа. Массив псевдоокончаний endings[] помогает обрабатывать эти особенности через регулярные выражения.

|  |
| --- |
| var endings = [  ["ет", "(ет|ут|ют)"], // Окончание для 3-го лица ед. числа глаголов I спряжения  ["ут", "(ет|ут|ют)"], // Окончание для 3-го лица мн. числа глаголов I спряжения  ["ится", "(ит|ат|ят)ся"], // Возвратное окончание II спряжения  ["ано", "ано"], // Краткое прилагательное среднего рода  ["такое", "– это"] // Исключение для "что такое ...?"  ]; |

Первая колонка: фактическое окончание, встречающееся в вопросе. Вторая колонка: регулярное выражение, которое используется для проверки совпадений.

Для конструкций типа «что такое ...?», где используется "псевдосказуемое" (например, такое), добавляется строка:

|  |
| --- |
| ["такое", "– это"] |

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

Массив «чёрный список» (blacklist[]) служит для исключения из обработки слов, которые имеют окончания, схожие со сказуемыми, но на самом деле сказуемыми не являются. Это помогает избежать ложного срабатывания при поиске сказуемых в вопросах и записи базы знаний.

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

Во время обработки текста каждое слово проверяется не только на соответствие псевдоокончаниям, но и на присутствие в массиве blacklist[]. Если слово находится в этом списке, оно пропускается, даже если формально его окончание подходит под шаблон сказуемого.

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

Функция getEnding(word) проверяет, является ли слово сказуемым:

|  |
| --- |
| function getEnding(word) {  if (blacklist.indexOf(word) !== -1) return -1; // исключение из обработки  for (var j = 0; j < endings.length; j++) {  if (word.endsWith(endings[j][0])) return j; // возвращает индекс окончания  }  return -1; // не найдено  } |

Для анализа сказуемого на совпадения его окончания с соответствующим ему псевдоокончанием в массиве псевдоокончаний endings[] определяется вспомогательная функция 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

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

# Структура функций для преобразования первых букв в тексте вопроса и ответа

Функция small()для преобразования первой буквы в тексте ответа в прописную:

|  |
| --- |
| function small(str) {  return str[0].toLowerCase() + str.slice(1);  } |

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

|  |
| --- |
| function big(str) {  return str[0].toUpperCase() + str.slice(1);  } |

# Процедура учета знаков препинания в предложении вопроса.

Знаки препинания отделяются пробелами.

Создается массив separators, содержащий все символы пунктуации, которые нужно обработать. Каждому символу из separators присваивается пробел перед ним.

Это выполняется для того, чтобы знаки препинания, прикрепленные к словам (например, "металл," или "что?"), не сливались со словами и корректно разделялись при дальнейшей обработке:

|  |
| --- |
| var separators = "'\",.!?()[]\\/";  for (var i = 0; i < separators.length; i++) {  txt = txt.replace(separators[i], " " + separators[i]);  } |

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

После учета знаков препинания текст вопроса хранится в переменной txt. Например(Придумай свой текст) (Ну и пробел перед ‘?’ нужен)):

|  |
| --- |
| var txt = "Что такое металл ?"; |

Метод split() применяется с разделителем ' ' (пробел). Это позволяет разделить текст на массив слов, где каждое слово или знак препинания становится отдельным элементом массива:

|  |
| --- |
| var words = txt.split(' '); |

Как итог, в массив words добавляются все элементы строки, разделенные пробелами, включая знаки препинания.

Для примера текста "Что такое металл ?" массив будет выглядеть так:

|  |
| --- |
| words = ["Что", "такое", "металл", "?"]; |

# Процедура поиска сказуемого в вопросе и формирования для него регулярного выражения

Цель этой процедуры — найти сказуемое в вопросе и преобразовать его в шаблон регулярного выражения, чтобы сопоставлять с данными из БЗ.

1. **Перебор слов в вопросе:**

После разбиения текста на массив слов (words) начинается цикл по каждому слову:

|  |
| --- |
| for (var i = 0; i < words.length; i++) {  var ending = getEnding(words[i]); // Проверка слова на псевдоокончание  } |

1. **Проверка псевдоокончаний:**

Функция getEnding(word) проверяет, соответствует ли слово сказуемому, используя массив псевдоокончаний (endings). Если найдено соответствие, возвращается индекс псевдоокончания.

1. **Создание шаблона для сказуемого**:

Если getEnding() возвращает индекс (то есть слово распознано как сказуемое), это слово преобразуется: у псевдоокончания в слове отсекается его часть. И добавляется шаблон из второго столбца массива endings[]

|  |
| --- |
| if (ending >= 0) {  words[i] = words[i].slice(0, -endings[ending][0].length) + endings[ending][1];  var predicate = new RegExp(words[i]); // Формируется регулярное выражение  } |

1. **Дополнительная обработка кратких прилагательных**:

Если сказуемое является кратким прил (например, "явен"), то для шаблона также захватывается следующее слово, так как прил часто употребляется в связке с сущ:

|  |
| --- |
| if (endings[ending][0] == endings[ending][1]) {  predicate = new RegExp(words[i] + " " + words[i + 1]);  i++; // Пропускаем следующее слово  } |

1. **Сохранение шаблона**:

После создания регулярное выражение для сказуемого сохраняется в переменной predicate и будет использоваться для поиска в базе знаний.

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

1. **Выделение слов, относящихся к подлежащему**:

После нахождения сказуемого в вопросе (на этапе поиска), подлежащее — это все слова, идущие за сказуемым. Эти слова выделяются с помощью метода slice() для массива слов words:

|  |
| --- |
| var subject\_words = words.slice(i + 1); |

1. **Объединение слов в строку с использованием шаблона**:

Для формирования регулярного выражения пробелы между словами заменяются на шаблон .\*, который означает произвольную последовательность символов (любая строка):

|  |
| --- |
| var subject\_string = subject\_words.join(".\*"); |

1. **Добавление шаблонов в начале и конце строки**:

К строке добавляются .\* в начале и в конце, чтобы учитывать случаи, когда подлежащее может быть частью более длинного текста:

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

1. **Проверка длины строки подлежащего**:

Регулярное выражение создается только если длина подлежащего превышает 3 символа, чтобы избежать ложных совпадений с короткими словами:

|  |
| --- |
| if (subject\_string.length > 3) {  var subject = new RegExp(".\*" + subject\_string + ".\*");  } |

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

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

Регулярное выражение subject, полученное для подлежащего, используется для проверки на соответсвие ячеек как первого, так и третьего столбцов в двумерном массиве базы знаний, поскольку одинаковые по смыслу предложения в исходном тексте могут быть сформированы с переставленными в предложении субъектной и объектными частями.

|  |
| --- |
| //поиск совпадений с шаблонами среди связей семантической сети  for (var j = 0; j < knowledge.length; j++)  {  //проверяемое условие на совпадение регулярных выражений для  //сказуемого и подлежащего в вопросе с записями в массиве БЗ  if (predicate.test(knowledge[j][1]) && (subject.test(knowledge[j][0]) ||  subject.test(knowledge[j][2])))  //создание предложения для ответа из найденной по заданному  //вопросу строки из массива БЗ  {  answer+=big(knowledge[j][0] + " " + knowledge[j][1] + " " + knowledge[j][2] + ". ");  result = true;} } |

# Процедура и код формирования ответа на вопрос с синомичными сказуемыми

Поскольку в вопросе может быть использовано сказуемое, синонимичное (совпадающее по смыслу, но не совпадающее по написанию) сказуемому использованного в тексте, предусмотрен повторный проход по строкам базы знаний без проверки сказуемого.

|  |
| --- |
| //если совпадений с двумя шаблонами нет  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 < knowledge.length; j++) {  if (subject.test(knowledge[j][0]) || subject.test(knowledge[j][2])) {  //создание простого предложения из найденной строки в массиве БЗ  answer += big(knowledge[j][0] + " " + knowledge[j][1] + " " + knowledge[j][2] + ". ");  result = true;  }}}} |

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

* Может обращаться к любому элементу объектной модели Web-документа DOM и предлагает механизм селекторов
* Управление 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

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

1. **В папку сайта добавить файлы:**

* библиотеку jQuery – файл jquery.js,
* файл для диалога, разработанный на JS, с именем dialog.js с функцией формирования диалогового окна, например, с именем dialog\_window();
* файл таблицы стилей, например, с именем style.css.

1. **В HTML-разметку диалоговой Web-страницы добавить следующие элементы:**

|  |
| --- |
| <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> //подключение файла библ. jQuery  <script src="dialog.js"></script> //обеспечивает интерфейс на основе диалогового окна  <script src="https://webasr.yandex.net/jsapi/v1/webspeechkit.js" type="text/javascript"></script> // подкл аудио сервис яндекс  <link href="style.css" type="text/css" rel="stylesheet"/> //подключение внешней таблицы стилей  </head>  <body onLoad="dialog\_window()"> //загрузка доп функции организации интерфейса для диалогового кона  …  </body>  </html> |

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

**1. Флаговая переменная dialogOn** для отслеживания активности диалогового окна.

**2. Функции:**

* dialog\_window() – создание диалогового окна.
* openDialog() – плавное открытие/закрытие окна.
* ask() – обработка вопроса и выдача ответа​.

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

Функция предназначена для создания диалогового окна с разделами для текста, истории, и формы ввода

Код добавляет HTML-разметку окна и предполагает добавление стилей через CSS:

|  |
| --- |
| <div id="dialog" class="dialog">  <!-- Полоса с кнопкой "Нажми, чтобы спросить" -->  <div class="label" onclick="openDialog()">Нажми, чтобы спросить!</div>  <!-- Заголовок "История" -->  <div class="header">История:</div>  <!-- Область для отображения истории вопросов и ответов -->  <div class="history" id="history"></div>  <!-- Поле ввода вопроса и кнопка "Спросить" -->  <div class="question">  <input id="Qdialog" placeholder="Введите вопрос" />  <br />  <button onclick="ask('Qdialog')">Спросить</button>  </div>  </div> |

# 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;  } |

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

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

|  |
| --- |
| .dialog .label {  transform: rotate(270deg);  width: 430px;  text-align: center;  color:#f00;  overflow:hidden;  display:inline-block;  margin-left:-26%;  margin-top:50%;  cursor: pointer;  } |

# 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);  dialogOn = false;  } else { //анимация открытия диалогового окна  $("#dialog").animate({"margin-left":"-1250px"}, 1000);  dialogOn = true;  }  } |

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

Блок div для выдачи ответа на вопрос:

|  |
| --- |
| <div class=’history’ id=’history’></div> |

Стиль:

|  |
| --- |
| .dialog .history {  height: 730px;  overflow-y: scroll;} |

# 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

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

|  |
| --- |
| .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;  } |

# 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="";  } |