**Курсовая работа**

**Построитель вопросительных предложений**

**Содержание**

Введение

1. Теоретическая часть

1.1 Генерация учебно-тренировочных задач на основе текста учебного материала

1.2 Постановка вопросов к членам предложения

2. Описание алгоритмов решения задачи

2.1 Выделение отдельных членов предложения

2.2 Построение дерева синтаксического подчинения

2.3 Генерация вопросов

2.3.1 Задание вопроса к слову

2.3.2 Построение вопросительного предложения

Заключение

Список литературы

Приложение 1. Листинг программы разбиения предложения на отдельные слова и поиска вопросительных слов к ним

Приложение 2. Список сокращений

**Введение**

Информационные технологии в образовании играют все более существенную роль. Современный учебный процесс трудно представить без использования компьютерных учебников, задачников, тестирующих и контролирующих систем. Одной из главных задач в таких системах является автоматическая генерация вопросительных предложений к различным частям лекционных материалов.

Цель нашей работы заключается в обеспечении автоматизации процесса составления различных тестов, проверок и задач. Для этого требуется рассмотреть несколько способов решения подобных задач и на их основе составить основной алгоритм решения поставленной перед нами задачи.

Задача генерации вопросительных предложений разбивается на несколько более простых подзадач:

1. разбор предложения (выделение отдельных членов предложения, построение дерева синтаксического подчинения);
2. построение вопросительных предложений на основе полученных результатов;
3. вывод результатов.

В нашей работе наиболее подробно будет рассмотрен только второй пункт. Так же будут затронуты проблемы разбора предложения.

**1. Теоретическая часть**

**1.1 Генерация учебно-тренировочных задач на основе текста учебного материала**

Рис. 1

Значительную долю учебного материала КУ и КОС составляет текст. Поэтому идея генерации УТЗ на его основе представляется весьма логичной. Схема, иллюстрирующая её воплощение изображена на рис. 1. В неё входят две процедуры. Первая применяется в процессе разработки КУ (КОС), вторая – при его эксплуатации. Очевидно, вторая процедура должна быть автоматической. "Идеальная" реализация рассматриваемой схемы имеет место, когда первая процедура является автоматической, т.е анализ текста и формирование моделей УТЗ для генерации производиться авторской системой без участия человека. [1]

Методы анализа текстов на естественном языке и построения на их основе моделей представления ПО находятся в фокусе исследований в области прикладной лингвистики и искусственного интеллекта. К числу ключевых проблем, связанных с развитием этих методов, относятся:

* сложность моделирования семантики;
* наличие так называемых "не-факторов", присущих человеческим представлениям и отражаемых в текстах (неточности, неполноты, несогласованности и др.)
* неоднозначность соответствия между естественно - языковыми и формализованным представлениям (одна и таже мысль может быть выражена по-разному; дополнительные трудности анализа вызывают синонимия и омонимия);
* необходимость учета контекста.

Особо отметим последнюю проблему. Человек понимает текст благодаря тому, что он обладает знаниями о ПО, к который данный текст относится. Эти знания составляют контекст, позволяющий прояснить смысл положений, содержащихся в тексте, а также извлечь из него мысли, которые имелись в виду, но не были выражены явно.

Таким образом, исходный фрагмент текста, поступающий на вход анализатора, недостачею для построения его адекватной семантической модели. Наряду с ним необходимо использовать базу знаний, отражающую как общие, так и специфичные для данной ПО представления. Создание такой базы знаний – непростая задача, на сегодняшний день в полной мере нерешенная.

Исходя из сказанного, мы считаем не рациональным стремиться во что бы то ни стало добиться автоматической Процедуры 1.

Распределение ролей между компьютерной системой и человеком в рамках интерактивной процедуры формирований УТЗ может быть разным. Чем выше степень автоматизации, тем эффективнее инструментарий и тем сложнее его реализация.

**1.2 Постановка вопросов к членам предложения**

Идея методов генерации вопросов к членам предложения заключается в следующем. Из текста выбирается предложение. Для него составляется грамматическая спецификация, служащая ядром модели задачи. С помощью нее в модели описываются члены предложения, к которым имеет смысл ставить вопросы. Для каждого из них определяется семантический класс, содержащий естественно-языковые выражения, представляющие некорректные альтернативные варианты ответа или его неисключающие компоненты.

По сути формулировка вопроса к какому-то члену предложения строится на основе исходного предложения путем удаления из него данного члена и подчиненных ему синтаксических единиц, добавления вопросительного слова или словосочетания, а также изменения порядка оставшихся членов.

Основные этапы интерактивной процедуры формирования модели УТЗ приведены в таблице 1.

Таблица 1.

|  |  |
| --- | --- |
| Этапы, выполняемые автором | Этапы, выполняемые системой |
| 1. Составление запроса на выборку предложений |  |
|  | 2. Анализ текста учебного материала и формирование множества предложений, релевантных запросу |
| 3. Выбор предложения из сформированного множества |
| 4. Обработка предложения с целью упрощения его структуры |  |
| 5. Составление грамматической спецификации предложения. При необходимости возврат на предыдущий этап для корректировки предложения |
| 6. Выбор членов предложения, к которым имеет смысл ставить вопросы |  |
| 7. Для каждого выбранного члена указание вопросительного слова или словосочетания. Продолжение работы в рамках процедуры либо выход из нее |

Отметим, что не все предложения подходят для генерации вопросов. К потенциально применимым относятся предложения, являющиеся повествовательными, синтаксически членимыми и полными в плане грамматической структуры. Предпочтение при выборе стоит отдавать простым двусоставным предложениям с прямым порядком слов. Осложненные и сложные предложения также могут использоваться, однако сложность их синтаксической структуры необходимо ограничить.

Выбор предложения на третьем этапе производится автором либо реализуется автоматически. Во втором случае критерием служит максимум степени релевантности предложения поисковому запросу.

Обработка предложения на четвертом этапе может включать:

* разбиение сложного предложения на части, соответствующие простым предложениям;
* замена местоимений наименованиями обозначаемых имя сущностей;
* преобразование неполного предложения к эквивалентному ему полному;
* удаление несущественных в смысловом отношении фрагментов;
* раскрытие неоднозначностей, интерпретация которых зависит от контекста.

Развитые программные средства анализа текста, автоматически выполняющие грамматический разбор предложений, могут использоваться на пятом этапе. Вариант разбора, предложенный системой, корректируется и дополняется автором.

Наличие в инструментарии представительного тезауруса позволяет частично автоматизировать решение задач седьмого этапа. Вопросительное слово или словосочетание выбирается в зависимости от типа предложений, грамматической роли члена, к которому ставиться вопрос, типа ассоциируемой с ним предикативной единицы, а также связей между этим членом и другими синтаксическими элементами предложения.

**2. Описание алгоритмов решения задачи**

**2.1 Выделение отдельных членов предложения**

Выделение отдельных слов в предложении является простой задачей по поиску лексем в строке. За этот этап отвечает следующий алгоритм:

1. ввод исходной строки;
2. если рассматриваемый символ в строке пробел, то пропускаем все пробелы, пока не встретим букву;
3. если рассматриваемый символ буква, то добавляем его и каждый следующий символ, если он буква, к слову, пока не встретим какой-либо знак препинания;
4. увеличиваем счетчик слов на 1 и добавляем полученное слово в массив;
5. если исходная строка не закончилась, переходим к шагу 2, иначе заканчиваем работу.

Данному алгоритму соответствует нижеприведенная программа:

char res[20]; // Буфер для хранения слова

char \*tmp; // Ссылка на исходную строку

int words(){

char \*r;

memset(res,0,20); // Обнуление буфера слова

r=res;

if(\*tmp==0)return 0; // Если исходная строка пуста или закончилась, выход из подпрограммы

while(\*tmp==' '||\*tmp=='\t')tmp++; // Пропуск пробелов

if(islower(\*tmp)||isupper(\*tmp)) // Если рассматриваемый символ буква…

{

 while(islower(\*tmp)||isupper(\*tmp))\*r++=\*tmp++; // Добавляем к буферу слова очередной символ, если он буква

 tmp++;

}

return 1;

}

Функция words() возвращает в качестве результата два значения: 1 и 0. 1 возвращается в том случае, когда найдено слово. Если же просмотр строки закончился или строка изначально была пуста, возвращается 0. Поэтому данная функция должна вызываться в цикле вида :

while(words()!=0){

 arr[i]=(word \*)calloc(1,sizeof(word)); // Добавление новой записи в массив

 memset(arr[i]->wrd,0,20);

 strcpy(arr[i]->wrd,res); // Запись результата работы функции в массив

 i++; // Увеличение количества найденых слов

}

Так как мы рассматриваем только простые предложения и предполагаем, что числительные и порядковые пишутся словами, то функция words() не учитывает знаки препинания и цифры.

После выделения слов в предложении идет этап анализа полученных результатов. То есть, для каждого слова создается список его параметров (часть речи, падеж, род, число и т.д.), которые хранятся в структуре вида:

struct word{

 char wrd[20]; //слово

 char qwrd[40]; //вопрос к слову

 char params[6]; // параметры слова

}.

Если же какой-то параметр не учитывается или у данного слова его нет, то в масиве params он обозначается как число 255 (или FF в шестнадцатиричной системе). Остальные значения могуь лежать в интервале от 0 до 254.

**2.2 Построение дерева синтаксического подчинения**

Один из самых трудоемких этапов в разборе предложений это построение дерева подчинения. Данная задача не входит в рамки нашего исследования, т. к. является проблемой из области прикладной лингвистики.

Однако хотелось бы сказать, что на данный момент нам известен один способ решения: с помощью семантической нейронной сети[2].

**2.3 Генерация вопросов**

По сути, формулировка вопроса к какому-то члену предложения строится на онове исходного предложения путем удаления из него данного члена и подчененных ему синтаксических единиц, добавления вопросительного слова или словосочетания, а также изменения порядка оставшихся членов.

Отметим, что не все предложения подходят для генерации вопросов. К потенциально применимым относятся предложения, являющиеся повествовательными, синтаксически членимыми и полными в плане граматической структуры. Предпочтение при выборе стоит отдавать простым двусоставным предложениям с прямым порядком слов. Осложненные и сложные предложения также могут использоваться, однако сложность их синтаксической структуры необходимо ограничить.

**2.3.1 Задание вопроса к слову**

На наш взгляд, самым простым способом задания вопроса к слову будет выбор подходящего вопросительного слова из заранее подготовленной таблицы по некоторым критериям. Такой подход можно наблюдать в базах данных, когда по некоторым исходным параметрам выбираются подходящие результаты. Например:

Пусть задана таблица с вопросительными словами или словосочетаниями QTABLE.

Для того, что бы задать вопрос к слову «раму», надо сделать следующий запрос:

SELECT \* FROM QTABLE WHERE params=wparams,

где wparams – список атрибутов слова, к которому задается вопрос (в этом примере это «сущ.,ед.,нар.,неод.»).

Результатом работы данного запроса будет вопросительное слово «что».

В нашей работе за этот этап отвечает функция getqword(). Алгоритм ее работы заключается в следующем:

Пусть у нас есть файл, в котором хранятся структуры word, в которых заданы только параметры и вопросительные слова. Функция принимает в качестве параметра структуру word. По заданным параметрам из базы данных (в нашем случае это файл qwrds.dat) выбирается вопросительное слово или словосочетание и записывается в соответствующее поле во входной структуре. Если же вопросительное слово не найдено, то функция возвращает 0.

Код функции getqword():

int getqword(word \*wrd){

 FILE \*f;

 int res=0;

 word tmp;

 f=fopen("qwrds.dat","r"); // Открытие базы данных

 while(fread(&tmp,sizeof(word),1,f)>0){ // Пока есть записи...

 if((wrd->params[0]&tmp.params[0])&& \

 (wrd->params[1]&tmp.params[5])&& \

 (wrd->params[2]&tmp.params[5])&& \

 (wrd->params[3]&tmp.params[5])&& \

 (wrd->params[4]&tmp.params[5])&& \

 (wrd->params[5]&tmp.params[5])){

 strcpy(wrd->qwrd,tmp.qwrd); // Запись результата во входную структуру

 res=1;

 break;

 }

 }

 fclose(f);

 return res; // выход из функции

}

**2.3.2 Построение вопросительного предложения**

Опишем правила построения вопросов. Для того чтобы сформировать вопрос к главному члену, необходимо представить структуру предложения в виде тройки:

(pq, Ps, Pn), (1)

где pq – член, к которому ставится вопрос (подлежащее или сказуемое);

Ps – множество членов, транзитивно подчиненных pq;

Pn – множество прочих членов.

Выражение (1) определяет аспекты синтаксической структуры, существенные при постановке вопроса к pq. Позиции, занимаемые составляющими (1), не отражают порядка слов в предложении. Например, pq может располагаться в его середине, а компоненты, соответствующие Pn, в начале и конце.

Структура вопроса к подлежащему двусоставного предложения описывает кортеж:

(wq, pop, Pn`), (2)

где wq – вопросительное слово или словосочетание;

pop – сказуемое (pop ∈ Pn);

Pn` - Pn\{pop}.

Из (2) видно, что из предложения удаляются pq и Ps. На первое место wq, после него указывается pop, за которым приводятся оставшиеся члены из Pn`. Выделение pop из Pn фиксирует изменение порядка слов, относящихся к Pn, по сравнению с исходным предложением. Таким образом, (2) отражает порядок слов в вопросе на уровне его декомпозиции на wq, pop и Pn`. Порядок слов в рамках pop и Pn` соответствует исходному предложению.

Следующий кортеж определяет структуру вопроса к сказуемому двусоставного предложения:

(wq, Pn, Psq), (3)

где Psq ⊆ Ps.

В контексте (3) Pn состоит из подлежащего и транзитивно подчиненных ему членов. Другое отличие (3) от (2) заключается в том, что при формировании вопроса из предложения могут быть удалены не все члены, транзитивно подчиненные pq. К числу обязательно удаляемых относятся определения, дополнения, придаточные изъяснительные, определительные, образа действия, меры и степени, а так же все транзитивно подчиненные им члены. Обстоятельства и другие виды придаточных могут быть как удалены, так и оставлены в вопросе.

Для выражения правила построения вопросов к второстепенным членам перепишем (1), выделив еще два компонента в структуре предложения:

(Pmq, Pmn, pq, Ps, Pn), (4)

где Pmq = (pmq.1, pmq.2, …, pmq.i) – цепочка второстепенных членов, транзитивно подчиняющих pq;

Pmn – множество, включающее: главные члены, однородный по отношению к главному члену, которому подчинен pq; второстепенные члены, однородные по отношению к элементам Pmq; второстепенные члены, транзитивно подчиненные этим главным и второстепенным членам;

Pn – множество прочих членов не входящих в Pmq, Pmn и Ps.

Место, которое занимают элементы из Pmq и Pmn в структуре предложения, иллюстрирует ориентированный граф, изображенный на рис. 2. Его вершины соответствуют главным («Г») и второстепенным («В») членам предложения, а ребра – отношениям подчинения. Множества однородных членов охватывают прямоугольники. Гиперребро в виде овала обозначает ПЕ. Вершина, ассоциируемая с членом pq, к которому задается вопрос, выделена жирной рамкой.

Рис. 2

Из рис. 2 видно, что цепочка Pmq задает путь, ведущий из вершины pq к вершине подчиняющего ее главного члена, образованный транзитивными вершинами второстепенных членов. Вершины, представляющие элементы Pmn, помечены крестиками. При построении вопроса соответствующие члены удаляются из предложения.

Структуру вопроса к второстепенному члену определяет кортеж:

(pmq.i, pmq.i-1, …, pmq.2, wq, pmq.1, Pp, Pn`), (5)

где Pp – множество, состоящее из главного члена, непосредственно подчиняющего pmq.i или pq, и транзитивно подчиненных ему членов, исключая pq и элементы из Pmq и Pmn;

Pn` = Pn\Pp.

При единичной длине Pmq вместо (5) используем кортеж:

(wq, pmq.1, Pp, Pn`). (6)

Если pq непосредственно подчинен главному члену, то Pmq не задается, а структура вопроса имеет вид:

(wq, Pp, Pn`). (7)

При обработке предложения в рамках интерактивной процедуры формирования модели УТЗ следует стремиться к тому, чтобы Pmq содержала не больше трех элементов.

Примеры вопросов разных типов приведены в таблице 2.

Таблица 2.

|  |  |
| --- | --- |
| Тип вопроса | Примеры |
| Вопрос к подлежащему | М. Планк впервые предположил, что энергия осциллятора принимает дискретные значения, пропорциональные частоте колебаний. |
| Кто | предположил | впервые, что энергия осциллятора принимает дискретные значения, пропорциональные частоте колебаний? |
| wq | pop | Pn` |
| В трансформаторе тока первичная обмотка включается в цепь измеряемого тока, а вторичная обмотка включается в цепь измерительных приборов и реле защиты. |
| Что | включается | в трансформаторе тока последовательно в цепь измеряемого тока? |
| wq | pop | Pn` |
| Что | включается | в трансформаторе тока в цепь измерительных приборов и реле защиты? |
| wq | pop | Pn` |
| Вопрос к сказуемому | Сальник герметизирует зазор между подвижной и неподвижной частями машины. |
| 1-й вариант вопроса: |
| Что делает | сальник | между подвижной и неподвижной частями машины? |
| wq | Pn | Psq |
| 2-й вариант вопроса: |
| Что делает | сальник? |
| wq | Pn |
| Вопрос к определению | Кварцевые пески широко применяются в производстве стекла, фарфора, динаса и силикатного кирпича. |
| Какие | пески | широко применяются в производстве стекла, фарфора, динаса и силикатного кирпича? |
| wq | pmq.1 | Pp |
| В производстве | какого  | кирпича | широко применяются кварцевые пески? |
| pmq.2 | wq | pmq.1 | Pp |
| В какую  | цепь | в трансформаторе тока включается последовательно | первичная обмотка? |
| wq | pmq.1 | Pp | Pn` |
| В цепь | какого | реле | В трансформаторе тока включается | Вторичная обмотка? |
| pmq.2 | wq | pmq.1 | Pp | Pn` |
| Какие  | значения | принимает | энергия осциллятора? |
| wq | pmq.1 | Pp | Pn` |
| Вопрос к дополнению | Что | широко применяется в производстве стекла, фарфора, динаса и силикатного кирпича? |
| wq | Pp |
| Куда | в трансформаторе тока включается последовательно | первичная обмотка? |
| wq | Pp | Pn` |
| Вопрос к обстоятельству | Как | в трансформаторе тока включается в цепь измеряемого тока | первичная обмотка? |
| wq | Pp | Pn` |
| Где | включается последовательно в цепь измеряемого тока | первичная обмотка? |
| включается в цепь измерительных приборов и реле защиты | вторичная обмотка? |
| wq | Pp | Pn` |
| Вопрос ко всему придаточному | Что | впервые предположил | М.Планк? |
| wq | Pp | Pn` |

**Заключение**

В процессе работы над темой «Построитель вопросительных предложений» были рассмотрены алгоритмы построения вопросительных предложений и разбора исходного предложения на составные части.

При рассмотрении основных алгоритмов часть из них была реализована на языке Си. Это алгоритмы разбиение предложения на состовляющие его слова и генирация вопроса к слову.

В ходе наших исследований были рассмотренны различные электронные ресурсы, книги и научные работы, посвященные данной тематике.

**Список литературы**

1. Башмаков А.И., Башмаков И.А. Разработка компьютерных учебников и обучающих систем. М.: Филинъ, 2003
2. Семантическая нейронная сеть, как формальный язык описания и обработки смысла текстов на естественном языке [Электронный ресурс] http://www.shuklin.com/ai/ht/ru/ai00001f.aspx –Загл. С экрана. Яз. Рус.
3. Греков В.Ф., Крючков С.Е., Чешко Л.А Пособие для занятий по русскому языку. М.: Просвящение, 1990
4. Гладкий А.В. Формальные грамматики и языки. М.,: Наука, 1973.
5. Розенталь Д.Э. Управление в русском языке. Словарь-справочник. М.: Книга, 1986.
6. Апресян Ю.Д., Богуславский И.М., Иомдин Л.Л., Лазурский А.В., Перцов Н.В., Санников В.З., Цинман Л.Л. Лингвистическое обеспечение системы ЭТАП-2. М.: Наука, 1989
7. Розенталь Д.Э., Голуб И.Б., Теленкова М.А. Современный русский язык: Учебное пособие. М.: Международные отношения, 1994. 560 с.
8. Синтаксический разбор в системах статистического анализа текста [Электронный ресурс] http://www.metric.ru/publications.asp?ob\_no=306 –Загл. С экрана. Яз. Рус.
9. Автоматическая генерация позитивных и негативных тестов для тестирования фазы синтаксического анализа [Электронный ресурс] http://citforum.univ.kiev.ua/SE/testing/generation/ –Загл. С экрана. Яз. Рус.

**Приложение 1**

Листинг программы разбиения предложения на отдельные слова и поиска вопросительных слов к ним

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <alloc.h>

typedef struct{

 char wrd[20];

 char qwrd[40];

 char params[6];

} word;

char res[20];

char \*tmp;

int words(){

 char \*r;

 memset(res,0,20);

 r=res;

 if(\*tmp==0)return 0;

 while(!(islower(\*tmp)||isupper(\*tmp)))tmp++;

 if(islower(\*tmp)||isupper(\*tmp))

 while(islower(\*tmp)||isupper(\*tmp))\*r++=\*tmp++;

 tmp++;

 return 1;

}

int getqword(word \*wrd){

 FILE \*f;

 int res=0;

 word tmp;

 f=fopen("qwrds.dat","r");

 while(fread(&tmp,sizeof(word),1,f)>0){

 if((wrd->params[0]&tmp.params[0])&& \

 (wrd->params[1]&tmp.params[5])&& \

 (wrd->params[2]&tmp.params[5])&& \

 (wrd->params[3]&tmp.params[5])&& \

 (wrd->params[4]&tmp.params[5])&& \

 (wrd->params[5]&tmp.params[5])){

 strcpy(wrd->qwrd,tmp.qwrd);

 res=1;

 break;

 }

 }

 fclose(f);

 return res;

}

int main(int argc, char \*argv[])

{

 char str[1000];

 word \*arr[100];

 int i=0,j;

 memset(str,0,1000);

 printf("Input string:");

 gets(str);

 tmp=str;

 while(words()!=0){

 arr[i]=(word \*)calloc(1,sizeof(word));

 memset(arr[i]->wrd,0,20);

 strcpy(arr[i]->wrd,res);

 i++;

 }

 for(j=0;j<i;j++)

 if(getqword(arr[i])==0)

printf(“Вопрос к слову не найден!!!\n”);

 return 0;

}

**Приложение 2**

Список сокращений

КУ – компьютерный учебник

КОС – компьютерная обучающая система

УТЗ – учебно-тренировочная задача

ПО – предметная область

СНС – синтаксически неразложимое словосочетание

ПЕ – предикативная единица