Глава 8
АЛГОРИТМ ПОСТРОЕНИЯ МАТРИЧНОГО СЕМАНТИКО-СИНТАКСИЧЕСКОГО ПРЕДСТАВЛЕНИЯ ЕСТЕСТВЕННО-ЯЗЫКОВОГО ТЕКСТА

8.1. Постановка задачи разработки алгоритма семантико-синтаксического анализа текстов

8.1.1. Назначение и крупноблочная структура алгоритма

Цель данной главы и главы 9 заключается в разработке алгоритма семантико-синтаксического анализа текстов из подъязыков русского языка (РЯ), реализующего предложенный в главе 7 новый метод выполнения преобразования “ЕЯ-текст – Семантическое представление (СП) текста”. При этом предложенное в главе 6 формальное понятие лингвистического базиса интерпретируется как описание структуры лингвистической базы данных (ЛБД), используемой алгоритмом. Рассматриваемые тексты могут выражать высказывания (сообщения), команды, специальные вопросы (т.е. вопросы с вопросительными словами), общие вопросы (т.е. вопросы с ответом «Да»/ «Нет»). Тексты могут, в частности, включать причастные обороты и придаточные определительные предложения.

Для реализации предложенного в главе 7 нового метода выполнения преобразования “ЕЯ-текст –> СП текста” ставится задача разработки алгоритма SemSyn, являющегося композицией некоторых алгоритмов BuildMatr и BuildSem, удовлетворяющих следующим условиям:

(1) BuildMatr – алгоритм преобразования текстов из некоторых практически интересных подъязыков русского языка в их матричные семантико-синтаксические представления (МССП);

(2) BuildSem – алгоритм сборки семантического представления ЕЯ-текста по его МССП, причем построенное СП текста является выражением

некоторого стандартного К-языка (т.е. является К-представлением входного текста).

Отправной точкой для разработки алгоритма являлся анализ поверхностной и смысловой структуры текстов из следующих подъязыков русского языка и английского языка, представляющих практический интерес:

- вопросы и сообщения на естественном языке к поисковой Интернет-системе нового поколения, касающиеся научных публикаций и участия специалистов в различных научных конференциях;
- команды и вопросы транспортно-погрузочному интеллектуальному роботу, в частности, роботу, действующему на автоматизированном складе, и роботу, действующему в аэропорту;
- вопросы и сообщения для базы данных, касающиеся выпуска, экспорта и импорта продукции различными предприятиями, фирмами;
- вопросы, с которыми оператор автоматизированного склада обращается к интеллектуальной базе данных;
- вопросы потенциальных покупателей к интеллектуальной базе данных Интернет-магазина.

8.1.2. Набор текстов, рассматривавшийся в качестве ориентира при разработке алгоритма

Познакомимся с примерами входных текстов, которые могут анализироваться алгоритмом, разработанным в данной главе. Тексты в примерах будут рассматриваться в качестве типичных представителей определенных подклассов входных текстов.

1. Сообщения (описания фактов)

1. Профессор Игорь Петрович Сомов опубликовал в 1996 - 2001 годах 8 статей в зарубежных научных журналах.
2. Академик Иван Петрович Павлов разработал теорию условных рефлексов.
3. Фирма “GlaxoWelcome” выпускает лекарства для больных астмой.
4. АО “Парус” с 1999 года экспортирует продукцию в Болгарию.

2. Команды

1. Отправить контейнеры фабрике “Заря” до 16:30.
2. Подтащи металлическую балку к самосвалу.

3. Погрузи 4 контейнера на рейс компании «Люфтганза».

3. Частноутвердительные (или общие) вопросы

1. Экспортирует ли продукцию в Болгарию АО “Парус”?

4. Вопросы с вопросительно-относительным местоимением “какой”

1. Какой контейнер предназначен для АО “Радуга”?
2. Каким рейсом улетел профессор Семенов?
3. Какие препараты, выпускаемые фирмой “GlaxoWelcome”, предназначены для больных астмой?
4. Какую монографию опубликовал в прошлом году профессор Семенов?
5. В каком университете работет профессор Игорь Сергеевич Сомов, о котором писала газета “Поиск” в ноябре 2002-го года?
6. В какие страны экспортируют продукцию предприятия, расположенные в Саратовской области?
7. Какие контейнеры с индийской керамикой, поступившие в пятницу, предназначены для АО “Парус“?
8. В каких странах выпускается препарат бекатит?
9. Какие европейские страны выпускают препарат серетид?
10. В какие страны экспортирует станки объединение “Радуга”?
11. Из каких стран получает оборудование завод “Старт”?
12. Какие статьи опубликованы профессором Семеновым в 2001 г.?

5. Вопросы частноинформативного актуально-синтаксического типа

В работе (Воробьева, Панюшева, Толстой 1975) к этому классу вопросов относят такие вопросы, когда спрашивающий не знает часть информации о ситуации и просит собеседника восполнить неизвестный ему аспект интересующего его факта.

Вопросы этого класса можно было бы назвать ролевыми, поскольку они начинаются с одного или нескольких вопросительных слов или словосочетаний, каждое из которых связано с определенной тематической ролью. Примеры таких вопросов приведены ниже:

1. Откуда и для кого поступили 3 двухтонных контейнера с индийской керамикой ?

2. Где выступал в 2003-м году профессор Новосельцев из МВТУ?
3. Где работает профессор Сомов?
4. Кто разработал теорию условных рефлексов ?
5. Кто выпускает препарат серетид ?
6. Где находится центральный офис фирмы “GlaxoWelcome”?
7. Для кого предназначены два контейнера с итальянской обувью ?

6. Вопросы относительно количества предметов

1. Сколько статей, опубликованных Игорем Сомовым с 1996 года, относятся к искусственному интеллекту?

2. Сколько трехтонных контейнеров, поступивших в пятницу из Ростова, предназначены для АО «Парус»?
3. Откуда и для кого поступили 3 двухтонных контейнера с индийской керамикой ?

7. Вопросы относительно количества событий

1. Сколько раз в этом году запрашивался учебник Коробова?
2. Сколько раз в прошлом году профессор Коробов участвовал в международных научных конференциях?

8.2. Формализация исходных предположений о рассматриваемых подъязыках естественного (русского) языка

Представим в формальном виде основную часть исходных предположений о структуре текстов из рассматриваемых подъзыков ЕЯ. Хотя анализ будет

касаться подъязыков русского языка, тот же метод применим и для описания поверхностной структуры текстов на английском, немецком, французском и многих других языках. Теоретической основой анализа являются понятие лингвистического базиса, введенное в предыдущей главе, и понятие бесконтекстной грамматики – одно из центральных понятий теории формальных грамматик, языков и трансляторов.

Будем использовать аппарат бесконтекстных грамматик для описания расширенного входного языка лингвистического анализатора. Построим бесконтекстную грамматику G, которая будет описывать слова тех частей речи, которые могут встречаться во входном языке, и способы комбинирования слов, относящихся к различным частям речи, чисел и числовых значений параметров.

Как известно, бесконтекстной грамматикой (или контекстно-свободной грамматикой, КС-грамматикой) называется упорядоченная четверка G вида (N, T, P, s_0), где N, T - конечные непересекающиеся множества символов, P - конечное множество выражений вида $s \rightarrow u$, где $s \in N$, u – цепочка (возможно, пустая) в алфавите $N \cup T$, $s_0 \in N$.

Элементы множеств N и T называются нетерминальными и терминальными символами, элементы множества P называются продукциями, s_0 называется начальным символом.

Будем использовать компактную форму записи $s \rightarrow u_1 \mid u_2 \mid u_3 \mid \ldots \mid u_n$ для нескольких продукций вида $s \rightarrow u_1$, $s \rightarrow u_2$, …. , $s \rightarrow u_n$. Кроме того, будем использовать в продукциях символ ::= вместо символа → .

Таким образом, из соображений компактности мы будем рассматривать ниже бесконтекстные грамматики в форме Бэкуса-Наура (Johnsonbaugh 2001).

Построим некоторую бесконтекстную грамматику в форме Бэкуса-Наура. Нетерминальными символами (или нетерминалами) этой грамматики будут служить выражения вида $< s_1, \ldots, s_n >$, где $n \geq 1$, для $k = 1, \ldots, n$ s_k - буква русского алфавита, цифра или знак – (тире). Такие выражения будут интерпретироваться как метки определенных фрагментов текста либо метки частей речи. Например, нетерминалами строящейся грамматики будут являться выражения <вопрос> , <команда> , <сообщение> , <вопрос-на-перечисление>, <да-нет-вопрос>, <зависимое-выражение>, <описание-участников-события> ,

<числовая-часть> . Начальным символом грамматики будет являться нетерминал <текст>.

Терминальными символами (или терминалами) этой грамматики будут служить элементы “?”, “,” , “.” (т.е. вопросительный знак, запятая и точка), “Сколько” , “ Сколько раз” , “ли” и элементы (какой), [сущ] , [прилаг] , [глаг] , [прич], [предлог], [колич-числит] , [число], [наречие] , [конструкт] , [имя], {сущ-нарицат} , {сущ-собств} , {глаг-в-неопред-форме}, {глаг-в-изъявит-накл} , {глаг-в-повелит-накл} , {вопр-относит-местоим}, {местоим-наречие}.

Рассмотрим следующую систему продукций:

<текст > ::= <вопрос> | <команда> | <сообщение> ;

<вопрос> :: = <да-нет-вопрос> | <вопрос-о-тематич-ролях> |

<колич-вопрос1> | <колич-вопрос2> | <вопрос-на-перечисление> ;

<да-нет-вопрос>: : = {глаг-в-изъявит-накл} “ли” <зависимое-выражение> “?” ;

<зависимое-выражение>:: = <описание-сущности> | [конструкт] | <зависимое-выражение><зависимое-выражение>;

<описание-сущности> : := <возм-предлог ><числовая-часть> <атрибутная-часть> <ядро-описания-сущности> ;

<возм-предлог > : := | <предлог > ; (5.2.1)

<числовая-часть> :: = | [число] | [колич-числит];

<атрибутная-часть > : := | [прилаг] [прилаг] <атрибутная-часть> ;

<ядро-описания-сущности> : : = [сущ] | {сущ-нарицат} [имя] | {сущ-нарицат} <послед-сущ-собств > | <послед-сущ-собств > | [сущ] <причастн-оборот> |

[сущ] <придаточное-опред-предложение> ;

<послед-сущ-собств > : := {сущ-собств} | {сущ-собств} <послед-сущ-собств > | {сущ-нарицат} <послед-сущ-собств > ;

<вопрос-о-тематич-ролях> ::= < ролевое-вопр-сочетание> <сообщение> “?” ;

< ролевое-вопр-сочетание> ::= < местоим-сочетание> | {местоим-наречие} | < ролевое-вопр-сочетание> <связка> < ролевое-вопр-сочетание> ;

< местоим -сочетание> ::= <возм-предлог > {вопр-относит-местоим} ;

<связка> ::= “ , “| “и “ ;

<колич-вопрос1> ::= “Сколько” <описание-сущности> <неполное-сообщение> “?” ;

<колич-вопрос2> :: =

“Сколько раз” <описание-сущности> <неполное-сообщение> “?” ;

<вопрос-на-перечисление> :: = <возм-предлог > (какой) <сообщение> “?” ;

<команда> : : = <действие> <описание-участников-события>;

<описание-участников-события> :: = <зависимое-выражение> |

<зависимое-выражение> <описание-участников-события> ;

<действие> : : = {глаг-в-повелит-накл} | {глаг-в-неопред-форме};

<причастн-оборот> : : = <возм-запятая> [прич] <простое-описание-участников-события> ;

<простое-описание-участников-события> ::= <простое-зависимое-выражение> |

<простое-зависимое-выражение>< простое-зависимое-выражение> ;

<придаточн-опред-предложение>::=<присоединяющая-часть><простое-сообщение>;

<присоединяющая-часть> ::= <возм-запятая> <возм-предлог> <вопр-относит-местоим> ;

<возм-запятая> ::= | “,” ;

<вопр-относит-местоим> ::= (который)(какой);

<простое-сообщение>::= <простое-описание-участников-события>{глаг-в-изъявит-накл}<возм-точка> | <простое-описание-участников-события>{глаг-в-изъявит-накл}<возм-точка><простое-описание-участников-события> ;

<возм-точка>. ::= | “.” ;

<простое-зависимое-выражение> ::= <простое-описание-сущности> | [конструкт] ;

<простое-описание-сущности> ::= <возм-предлог > <числ-часть><атрибутная-часть>< простое-ядро-описания-сущности> ;

<простое-ядро-описания-сущности> ::= [сущ] | {сущ-нарицат} [имя] | {сущ-нарицат} <послед-сущ-собств > | <послед-сущ-собств > ;

<расширенное-ядро-описания-сущности> ::= <простое-ядро-описания-сущности> > | [сущ] <причастн-оборот> |

[сущ] <придаточн-опред-предложение> ;

<сообщение> ::= <описание-участников-события> {глаг-в-изъявит-накл} <правая-часть-сообщения> <возм-точка> ;

<правая-часть-сообщения> :: = | <описание-участников-события> ;

<неполн-сообщение> :: = {глаг-в-изъявит-накл} <описание-участников-события>.

Пример. Пусть В1 = “В каком петербургском издательстве в 2000-м году вышла книга “Базы знаний интеллектуальных систем”? ”. Тогда в грамматике с системой продукций (5.2.1) можно выполнить следующую систему замен нетерминалов на правые части продукций, приводящую к выводу цепочки, дающей обобщенное описание структуры вопроса В1:

<текст > = > <вопрос> ,

<вопрос> = > <вопрос-на-перечисление> ;

<вопрос-на-перечисление> = > <возм-предлог > (какой) <сообщение> “?” ;

<возм-предлог > (какой) <сообщение> “?” = >

[предлог] (какой) <сообщение> “?” ;

<сообщение> => <описание-участников-события> {глаг-в-изъявит-накл} <правая-часть-сообщения> <возм-точка>

=> <описание-участников-события><описание-участников-события> {глаг-в-изъявит-накл} <описание-участников-события> =>

[прилаг] [сущ] [предлог] [конструкт] {глаг-в-изъявит-накл} [сущ-нарицат] [имя] .

Поэтому, очевидно, <текст > => Expr1 , где Expr1 – цепочка вида

[предлог] (какой) [прилаг] [сущ] [предлог] [конструкт]

{глаг-в-изъявит-накл} [сущ-нарицат] [имя] “?” . (8.2.2)

Определение. Пусть *G = (N, T, P, s)* – бесконтекстная грамматика с множеством нетерминальных символов (нетерминалов) *N* , множеством терминальных символов (терминалов) *T,* множеством продукций *P* и начальным символом *s*. Тогда через *L (G)* обозначим множество всех цепочек в алфавите *T* , выводимых из *s* с помощью продукций из *P* .

Определение. Пусть *Lingb* – лингвистический базис вида (8.8.2), *Qmk* – разметка вопросов вида (6.2.1), *G = (N, T, <текст>, P)* – произвольная бесконтекстная грамматика с нетерминалами вида <a > и терминалами видов *(b) ,“c”, [d], {h}* , где *a* – некоторое выражение в русском алфавите,

обогащенным символом ‘-‘ ; $b \in Lecs$, *где* $Lecs$ – множество лексем текстообразующей системы, являющейся компонентом $Lingb$; с $\in W$, где W – множество словоформ морфологического базиса, являющегося компонентом $Lingb$; $d \in Classes(Lingb)$, $h \in Subclasses(Lingb)$, и <*текст*> – начальный символ G .

Тогда через $Linp(G, Lingb)$ обозначим множество всех цепочек, каждая из которых может быть получена из некоторой цепочки x из языка $L(G)$ выполнением одного или нескольких из следующих преобразований: (1) терминал вида (b) заменяется на произвольную словоформу $u \in W(Morphbs(Tform))$, такую, что $lcs(u) = b$ (т.е. b является лексемой словоформы u) ; (2) терминал вида *“c”* заменяется на c ; (3) терминал вида $[d]$ заменяется на произвольный такой элемент $u \in Textunit$s $(Tform)$, что $tclass(u) = d$; (4) терминал вида $\{h\}$ заменяется на произвольный такой элемент z $\in Textunits(Tform)$, что $subclass(z) = h$.

Пример. Нетрудно определить такой лингвистический базис $Lingb$, что язык $Linp(G, Lingb)$ включает вопрос B1 из предыдущего примера, и B1 может быть получен из цепочки Expr1 вида (8.2.2) применением перечисленных выше преобразований.

Таким образом, выше предложен новый метод формального описания предположений о структуре входных текстов лингвистического процессора на основе комбинированного использования аппарата бесконтекстных грамматик и понятия лингвистического базиса, введенного в 6-й главе.

8.3. Начальные этапы разработки алгоритма построения матричного семантико-синтаксического представления входного текста лингвистического процессора

8.3.1. Назначение алгоритма

Разрабатываемый в данной главе алгоритм BuildMatr предназначен для преобразования входных текстов из подъязыков русского языка (РЯ) в их матричные семантико-семантические представления (МССП). Операции, осуществляемые по входному тексту, зависят от выбранного лингвистического

базиса (л.б.) *Lingb*, интерпретируемого как формальная модель лингвистической базы данных (ЛБД).

При этом входной текст T должен являться выражением языка *Linp (G, Lingb),* где *G* – бесконтекстная грамматика с системой продукций вида (8.2.1), построенная в предыдущем параграфе. Главными выходными данными алгоритма должны являться строка *kindtext*, задающая вид входного текста (т.е. классифицирующая этот текст), и строково-числовая матрица *Matr* – МССП входного текста.

Известно, что проблема семантико-синтаксического анализа (ССА) компьютером ЕЯ-текстов включает много аспектов, и разные аспекты этой проблемы проработаны в разной степени. Начальный этап ССА текстов на РЯ включает нахождение базовых форм словоформ из входного текста, нахождение возможных значений морфологических признаков (число, падеж, лицо, время и т.д.) для этих словоформ и разбиение текста на такие сегменты, которые соответствуют определенным элементарным единицам смыслового уровня. К таким сегментам относятся, например, выражения “были отправлены”, “будет подготовлен”, “Олимпийские игры”, “840 км” , “1999-й год” , “2 часа”, “пять градусов“.

Вопросы автоматизации морфологического анализа текстов на русском языке исследованы во многих публикациях.

Вопросы, касающиеся автоматического выделения таких коротких сегментов текста, которые обозначают элементарные единицы смыслового уровня, оказываются не слишком сложными с логической точки зрения и могут решаться непосредственно на уровне программной реализации алгоритма.

Анализ литературы и собственный опыт автора говорят о том. что основные трудности логического характера при автоматизации ССА ЕЯ-текстов касаются поиска смысловых отношений между компонентами входного текста.

В связи с этим при разработке алгоритма BuildMatr основной акцент делается на формализации и алгоритмизации процесса поиска смысловых отношений между компонентами входного текста. Реализация этого акцента достигается следующим образом:

1. Во многих случаях входной текст T алгоритма является некоторой абстракцией по сравнению с реальным входным текстом лингвистического процессора. Дело в том, что единицами входного текста алгоритма могут быть не только словоформы, но и короткие словосочетания (“были отправлены”, “будет подготовлен”, “Олимпийские игры”, “840 км” , “1999-й год” , “2 часа”), являющиеся обозначениями элементарных единиц смыслового (семантического, концептуального) уровня.
2. Постулируется существование отображений, ставящих в соответствие словоформам их базовые формы (лексемы) и наборы значений морфологических признаков, а также связываюшие с каждым конструктом (числовым значением какого-то параметра) некоторую семантическую (или концептуальную) единицу; с этой целью в главе 4 было введено понятие текстообразующей системы.

Известны два основных подхода к разработке алгоритмов: проектирование сверху вниз (нисходящее проектирование) и проектирование снизу вверх (восходящее проектирование). Представляется целесообразным при разработке сильно структурированного алгоритма *Semsyn* сочетать оба этих метода. Это сочетание методов нисходящего и восходящего проектирования алгоритмов нашло отражение в описании алгоритма BuildMatr в последующих параграфах данной главы и в описании алгоритма BuildSem в главе 9.

В тех случаях, когда вспомогательный алгоритм является весьма простым (непосредственно программируемым) либо его разработка не представляет теоретических трудностей с учетом имеющихся научных публикаций, в структурированных описаниях алгоритмов BuildMatr и BuildSem указывается только внешняя спецификация такого алгоритма, т.е. описание назначения, входных и выходных данных алгоритма.

Для описания алгоритмов в данной главе используется один из возможных вариантов языка разработки алгоритмов, или псевдокода. Служебные слова *нач*, *кон*, *кесли, кцикл, квыбор* интерпретируются следующим образом: *нач = начало*, *кон = конец*, *кесли = конец-если, кцикл = конец-цикла*, *квыбор= конец оператора выбора.*

8.3.2. Внешняя спецификация алгоритма BuildMatr

Входные данные:

Lingb – лингвистический базис (см. параграф 6.8);

T – текст из языка *Linp(G, Lingb)* , где *G* – бесконтекстная грамматика вида (8.2.1).

Основные выходные данные:

nt – целое, количество единиц текста;

Rc – классифицирующее представление входного текста T (см. подраздел 7.1.1);

Rm – морфологическое представление входного текста (см. подраздел 7.1.1);

kindtext – строковая переменная, значение которой позволяет отнести входной текст к одному из подклассов текстов;

Arls – двумерный массив – проекция лексико-семантического словаря **Lsdic** на входной текст **T** (см. подраздел 7.1.2);

Arvfr – двумерный массив – проекция словаря глагольно-предложных фреймов **Vfr** на входной текст **T** (см. подраздел 7.1.2);

Arfrp – двумерный массив - проекция словаря предложных семантико-синтаксических фреймов **Frp** на входной текст **T** (см. подраздел 7.1.2);

Matr - матричное семантико-синтаксическое представление (МССП) входного текста (см. параграф 7.2).

8.3.3. Разработка плана алгоритма BuildMatr

План алгоритма BuildMatr

Нач Построение-компон-морфол-представления (T, Rc, nt, Rm)
Построение-проекции-лексико-семантич-словаря (Lsdic, nt, Rc, Rm, Arls)
Построение-проекции-словаря-глаг-фреймов (Arls, Vfr, nt, Rc, Rm, Arvfr)
Постр-проекции- словаря-предложных-фреймов (Arls, Frp, nt, Rc, Rm, Arfrp)
Формирование-начальных-значений-данных
Выявление-вида-текста (nt, Rc, Rm, leftprep, mainpos, kindtext, pos)
Цикл-до pos := pos + 1
Class := Rc[pos, tclass]
выбор class **из**

предлог: Обработка-предлога

прилаг: Обработка-прилаг

колич-числит: Обработка-колич-числит

сущ: Обработка-сущ

местом: Обработка-местоим

наречие: Обработка- наречия

глаг, прич: Обработка-глаг-формы

союз: Пустой оператор

констр: Обработка-конструкта

имя: Обработка-названий

маркер: если Rc[pos, unit] = ',' {запятая }

то Обработка-запятой

кесли

квыбор

выход-при *(pos = nt)*

конец

Алгоритм “Формирование-начальных-значений-данных”

Нач Обнулить все целочисленные переменные.

Присвоить значение nil (пустой элемент) всем строковым переменным.

Обнулить все числовые столбцы и заполнить цепочкой nil все строковые столбцы используемых одномерных и двумерных массивов.

Для каждой такой строки с номером k классифицирующего представления Rc, что эта строка соответствует словоформе из входного текста,

Matr[k, locunit] := наименьший номер строки из массива Arls (проекции лексико-семантического словаря Lsdic на входной текст) с информацией, соответствующей данной словоформе;

Matr[k, nval] := количество строк из Arls, соответствующих этой словоформе

кон

Последующие параграфы данной главы посвящены детализации этого плана, т.е. разработке первой части алгоритма семантико-синтаксического анализа текстов из представляющих практический интерес подъязыков естественного (русского) языка.

8.4. Описание алгоритма выявления вида входного текста

8.4.1. Назначение алгоритма

Алгоритм “Выявление-вида-текста” предназначен для отнесения текста к определенному классу; вид класса является значением выходной переменной kindtext. Спектр значений, которые может принимать переменная kindtext, представлен в следующей таблице:

Высказывания (сообщения)	АО “Парус” с 1999 года экспортирует продукцию в Болгарию.	stat
Команды	Отправить контейнеры фабрике “Заря”до 16:30.	imp
Частноутвердительные (общие) вопросы	Экспортирует ли продукцию в Болгарию АО “Парус”?	genqs
Вопросы о количестве предметов	Сколько статей по органической химии опубликовано профессором Игорем Сомовым в прошлом году?	specqs-quant1
Вопросы о количестве событий	Сколько раз в этом году запрашивался учебник Коробова?	specqs-quant2
Ролевые вопросы	Откуда поступили трехтонные контейнеры?	specqs-rol
Вопросы с вопросительно-относительным местоимением “какой” в единственном числе	В каком институте работает профессор Игорь Павлович Сомов?	specqs-relat1
Вопросы с вопросительно-относительным местоимением “какой” во множественном числе	Из каких стран импортирует комплектующие АО “Радуга”?	specqs-relat2

Табл. 8.1. Примеры входных текстов разных видов

Значение переменной kindtext используется в алгоритме построения семантического представления (СП) текста по его матричному семантико-синтаксическому представлению (параграф 5.10). Например, пусть В1 = “Откуда поступили трехтонные контейнеры?”, В2 = “Сколько раз в этом году запрашивался учебник Коробова?”. Для вопроса В1 kindtext = specqs-rol, и СП вопроса В1 может являться К-формулой *Вопрос (x1, (Ситуация(e1, поступление2 * (Место1, x1)(Время, x2)(Объект1, нек множество *(Качеств-состав, контейнер * (Вес, 3/ тонна)) : S1)) ∧ Раньше (x1, #сейчас#)))*.

В случае вопроса В2 kindtext = specqs-quant2, и СП вопроса В2 может являться К-формулой

*Вопрос (x1, (x1 ≡ Колич-элем(все запрос1 *(Время, текущий-год) (Предмет-запроса,*

*нек учебник * (Автор, нек человек *(Фамилия, “Коробов”) : x2)))))* .

Кроме переменной kindtext, к выходным данным алгоритма “Выявление-вида-текста” относятся переменные mainpos и pos. Значением переменной mainpos является номер позиции, занимаемой в тексте вопросительным словом. Например, для вопросов В3 = “Из каких стран импортирует комплектующие АО “Радуга”? и В1 = “ Откуда поступили трехтонные контейнеры?” переменная mainpos принимает соответственно значения 2 и 1. Для команд, сообщений и вопросов с ответом “Да/Нет” переменная mainpos принимает значение 1.

8.4.2. Внешняя спецификация алгоритма “Выявление-вида-текста”

Вход: Lingb – лингвистический базис, Rc и Rm – классифицирующее и морфологическое представления входного текста T (см. подраздел 7.1.1.).

Выход: kindtext, leftprep – строковые переменные для обозначения соответственно вида входного текста и предлога в начале текста; mainpos, pos - целочисленные переменные для обозначения соответственно позиции вопросительного слова и позиции, после которой следует продолжить обработку текста.

Внешние спецификации вспомогательных алгоритмов

Спецификация алгоритма-функции “Число”

Вход: d – элемент морфологического пространства Spmorph(Tform(Lingb)), соответствующий словоформе, которая может иметь число.

Выход: number1 – значение 1 для единственного числа, 2 для множественного числа, 3 в тех случаях, когда словоформа может быть отнесена как к единственному, так и множественному числу (пример: “реки”).

Спецификация алгоритма “Форма-глагола”

Вход: d – элемент морфологического пространства Spmorph(Tform(Lingb)), соответствующий глаголу.

Выход: form1– строка со значением ‘неопр’, если d соответствует глаголу в неопределенной форме; значением ‘изъявит’ для представления глагола в изъявительном наклонении (форме, с которой связано грамматическое время); значением 'повелит’, если d соответствует глаголу в повелительном наклонении.

5.4.3. Алгоритм “Выявление-вида-текста”

```
Нач      leftprep := nil, kindtext := nil
     Если     kindtext = nil
     То       log1 := (Rc[1,subclass] = вопр-относ-местоим);
          log2:=((Rc[1,tclass] = предлог) И (Rc[2,subclass] = вопр-относ-местоим));
               если (log1 = Ист) то mainpos := 1 кесли;
               если (log2 = Ист) то mainpos := 2 кесли;
{(log1 = Ист) – признак вопроса Класса 1А. Пример:
Какие препараты фирмы  GlaxoWelcome выпускаются в Польше?}
{(log2 = Ист) – признак вопроса Класса 1Б. Пример:
“Из каких стран импортирует комплектующие АО “Радуга”?}
               если (log1 = Ист) или (log2 = Ист)
               то нач m1 := Rc[mainpos,mcoord]; baseform:= Rm[base, m1];
                    number1:= Число (Rm[m1, morph]);
                 если baseform = ‘какой‘
                 то если number1= 1 то kindtext := specqs-relat1
```

иначе kindtext := specqs-relat2 кесли

если (log2 = Ист) то leftprep := Rc[1,unit] кесли;

если number1= 1 то var1 := ‘x1’ {признак вопроса о единственном объекте} иначе var1:= ‘S1’{признак вопроса о множестве объектов} кесли; pos := mainpos кесли кон кесли;

Если kindtext = nil

То log3 := ((Rc[1,subclass] = вопр-относ-местоим) ИЛИ ((Rc[1,subclass] = местоим-наречие) И (Rc[1, unit] – одно из слов (местоименных наречий)”когда”, “куда”, “откуда”, “где”)));

log 4 := ((Rc[1,tclass] = предлог) И (Rc[2,subclass] = вопр-относ-местоим));

если (log3 = Ист) ИЛИ (log4 = Ист)

то если (log3 = Ист) то mainpos := 1 кесли;

если (log4 = Ист) то mainpos := 2 кесли;

если Rc[mainpos, subclass] = вопр-относ-местоим

то m1 := Rc[mainpos,mcoord]; baseform:= Rm[base, m1];

если baseform не является лексемой ‘какой‘

то kindtext := specqs-rol;

если (log4 = Ист) то leftprep := Rc[1,unit] кесли

var1 := ‘x1’; pos := mainpos – 1;

{(log3 = Ист) – признак вопроса Класса 2А. Пример:

“Откуда поступили трехтонные контейнеры?”}

{(log4 = Ист) – признак вопроса Класса 2Б. Пример:

“Для кого предназначены контейнеры с керамикой”?} кесли кесли кесли

Если kindtext = nil

То если ((Rc[1,tclass] =глаг) И (Rc[2,unit] = “ли”))

то kindtext := genqs кесли

{ Признак вопроса Класса 3 (вопроса с ответом “Да”/”Нет”).

Пример: “Экспортирует ли продукцию в Болгарию АО “Парус”?}

Если kindtext = nil

То log5 := (Rc[1,unit] = ‘сколько’); log6 := ((Rc[2,tclass] = сущ) ;

log7 := (Rc[2,unit] = ‘раз’);

если ((log5 = Ист) И (log6 = Ист))

то var1 := ‘S1’ {признак вопроса о множестве объектов}

если (log7 = Ложь) то kindtext := specqs-quant1, mainpos := 1

{признак вопроса класса 4. Пример: “Сколько статей по органической химии опубликовано профессором И.П. Сомовым в прошлом году?”}

иначе {т.е. в случае log7 = Ист}

kindtext := specqs-quant2, mainpos := 2 кесли

{признак вопроса класса 5. Пример: “Сколько раз в этом году запрашивался роман Сергея Коробова?”} pos := mainpos кесли

Если kindtext = nil

То log8 := (Rc[1, tclass] = ‘глаг’);

если (log8 = Ист)

то нач pos := mainpos; m1 := Rc[mainpos,mcoord];

form1:= Форма-глаг (Rm[m1, morph]);

если ((form1 = неопр) ИЛИ (form1 = повелит))

то kindtext := imp {признак команды. Пример : “Отправить контейнеры фабрике “Заря”до 16:30”} кесли кон кесли кесли

Если (kintext = nil) то kindtext := stat {признак сообщения. Пример :

“ АО “Парус” с 1999 года экспортирует продукцию в Болгарию. ”} кесли конец

8.5. Принципы обработки ролевых вопросительных словосочетаний

Условимся говорить, что предложение начинается с вопросительного слова wd, если wd является первым словом данного предложения или первым словом предложения является некоторый предлог, за которым следует wd. Например, будем считать, что каждый из вопросов В1=”Кого командировали в Пензу?” и В2=”Для кого поступили три двухтонных контейнера?” начинается с вопросительного слова “кого”.

Разобьем на две группы все вопросительные слова, с которых могут начинаться рассматриваемые вопросы. К первой группе относятся вопросительно-относительные местоимения “какой”, “какие” и их формы, соответствующие различным грамматическим падежам, а также слово “сколько”. Во вторую группу включим вопросительно-относительные

местоимения “кто”, “кому”, “кого”, “чем” и т.п., а также местоименные наречия “где”, “когда”, “куда”, “откуда”. Каждое слово из этой группы вместе с определенным предлогом предназначены для выражения определенной тематической роли, т.е. смыслового отношения между значением глагола и значением выражения, зависящим в предложении от данного глагола. Если вопросительное слово wd не требует предлога для выражения определенной тематической роли, то будем говорить, что это слово используется вместе с пустым предлогом nil. Например, пара (nil, кого) в вопросе В1 используется для выражения тематической роли “Объект действия”, а пара (для, кого) в предложении В2 позволяет выразить тематическую роль “Адресат”. Учитывая сказанное, слова из второй группы будем называть *ролевыми вопросительными словами.* Начальная обработка вопросительных местоимений из первой группы осуществляется алгоритмом “Выявление-вида-текста”. Позиция вопросительного слова является значением выходной переменной mainpos. Значение переменной kindtext указывает на подкласс, к которому относится данное вопросительное слово.

Ролевые вопросительные слова (т.е. слова из второй группы) могут вместе с предлогами образовывать последовательности, являющиеся левыми сегментами вопросов. Примером может послужить вопрос В3 = “Когда, для кого откуда поступили три алюминиевых контейнера?”. В связи с этим необходим специальный алгоритм “Обработка-ролевых-вопрос-слов”. Этот алгоритм использует множество наборов Rqs, являющееся частью лингвистического базиса (ЛБД). Наборы из Rqs имеют следующую структуру:

nb (номер)	prep (предлог)	qswd (вопросительное слово)	relq (тематическая роль)
1	nil	откуда	Место1
2	для	кого	Адресат

Табл. 8.2. Структура набора из множества Rqs

Алгоритм “Обработка-ролевых-вопрос-слов” вызывается для обработки ролевых вопросительных слов, располагающихся в начале многих вопросов. Такие слова являются вопросительно-относительными местоимениями (“кто”, “кому”, “чем” и т.д.) или местоименными наречиями (“когда”, “куда”, “откуда”).

Внешняя спецификация алгоритма “Обработка-ролевых-вопрос-слов”

Вход: nt – цел - количество единиц текста; **Rc** – классифицирующее представление входного текста T (см. параграф 7.1); **pos** – цел – позиция вопросительного слова в Rc**; Rqs** – словарь вопросительных словосочетаний (одна из составляющих Lingb);
Matr - матричное семантико-синтаксическое представление (МССП) входного текста (см. параграф 7.2); **leftprep** - строка – значение предлога слева; **numbent** – цел – количество объектов, упомянутых в тексте; **numbqswd** – цел – количество уже найденных вопросительных слов в тексте; **posqswd** – одномерный массив длины nt, где для k≥ 1 posqswd [k] – либо позиция в Rc k-го вопросительного слова, либо 0.
Выход: Matr, numbent, numbqswd, posqswd, leftprep.

Алгоритм “Обработка-ролевых-вопрос-слов”

нач {Условие вызова:(subclass = вопр-относ-местоим) ИЛИ (subclass = местоим-наречие)}

numbqswd:=numbqswd+1; { количество вопросительных слов в тексте } posqswd[numbqswd]:=pos; word1:=Rc[pos,unit]; {запоминается позиция вопросительного слова для последующего связывания с глаголом}

если (subclass = вопр-относ-местоим)

то Найти в множестве Rqs - одной из составляющих Lingb – набор с таким порядковым номером k1, что

Rqs[k1,prep] = leftprep, Rqs[k1, qswd]:= word1

иначе Найти в множестве Rqs набор с таким порядковым номером k1, что Rqs [k1, qswd] = word1 кесли

Затем role:=Rqs[k1, relq], Matr[pos, reldir, 1] := role

{ для кого=> leftprep=’для’, word1=’кого’, role:=’Адресат’}

{ кто=> leftprep=nil, word1=’кто’, role:=’Агент’}

numbent := numbent+1; var1:= var(‘x’, numbent);

Matr[pos, mark] := var1; leftprep:=nil кесли кон

8.6. Принципы и методы обработки причастных оборотов и придаточных определительных предложений

8.6.1. Принципы обработки

Каждому причастному обороту и придаточному определительному предложению ставится в соответствие определенный номер уровня вложения в главное предложение; этот номер является значением переменной depth (“глубина” в переводе с английского). Пусть pos – целочисленная переменная, значение которой является порядковым номером единицы текста, анализируемой в данный момент вычисления. Тогда, если pos указывает какую-либо текстовую единицу в главном предложении, то depth = 1. При переходе из главного предложения в причастный оборот или придаточное определительное предложение уровень глубины увеличивается на единицу. Увеличение уровня глубины на единицу происходит и в том случае, когда осуществляется переход из фрагмента текста с уровнем глубины depth > 1 в причастный оборот или придаточное определительное предложение.

Позиция глагольной формы (глагола или причастия) во фрагменте с глубиной *depth* анализируемого предложения задается элементом verbposmag[depth] анализируемого массива verbposmag длины 4. Размер массива определяется предположением о том, что в реальных фразах максимальный уровень глубины

depth равен 4, причем даже уровень глубины 3 достигается редко. Перед началом анализа фразы элементы массива verbposmag обнуляются.

Пример. Пусть T1 = “Профессор Игорь Сергеевич Сомов, о котором пишет газета “Поиск” в номере, поступившем в субботу, работает в МИФИ.”, E1 – фрагмент “Профессор Игорь Сергеевич Сомов,”, E2 – фрагмент “о котором пишет газета “Поиск” в номере,”, E3 – фрагмент “поступившем в субботу,“, E4 – фрагмент “работает в МИФИ.”. Тогда, если переменная *pos* указывает какую-либо текстовую единицу во фрагментах E1, E2, E3, E4, то переменная depth принимает соответственно значения 1, 2, 3, 1.

Если pos = 4 (позиция слова “Сомов”), то depth = 1 и verbposmag[depth] = 0, поскольку глагол из главного предложения еще не найден. Пусть pos = 9 (позиция слова “газета”), тогда depth = 2 и verbposmag[depth] = 8 (позиция глагола “пишет”), при этом в позициях 1, 3, 4 массива verbposmag расположен 0.

Матричное семантико-синтаксическое представление (МССП) Matr и вспомогательный целочисленный массив posconnectword используются для отображения информации о взаимосвязях главного предложения и либо причастного оборота, либо придаточного определительного предложения. Количество строк в Matr и длина массива posconnectword равны nt - количеству строк в классифицируюшем представлении Rc входного текста, т.е. количеству элементарных значащих единиц текста.

Напомним, что если pos – позиция причастия, с которого начинается причастный оборот, и это причастие “прикреплено” к существительному в позиции m, то Matr[pos, contr] = m.

Пример. Пусть T2 = “Сколько контейнеров с индийской керамикой, поступивших из Новороссийска, были отправлены АО “Радуга”?”. Рассмотрим размеченное представление текста T2 (проставим после каждой элементарной значащей единицы текста ее номер): “Сколько (1) контейнеров (2) с (3) индийской (4) керамикой (5) , (6) поступивших (7) из (8) Новороссийска (9) , (10) были отправлены (11) АО (12) “Радуга” (13) ? (14)”. Тогда Matr[7, contr] = 2, где 7 и 2 – позиции слов “поступивших” и “контейнеров”.

Ненулевые элементы массива posconnectword предназначены для установления взаимосвязи между позицией pos глагола в придаточном определительном предложении и союзным словом, “прикрепляющим” это придаточное предложение к главному.

Пример. Пусть Т3 – размеченный текст “Сколько (1) контейнеров (2) с (3) индийской (4) керамикой (5) , (6) которые (7) в (8) пятницу (9) поступили (10) из (11) Новороссийска (12) , (13) были отправлены (14) АО (15) “Радуга” (16) ? (17)”. Тогда posconnectrword[10] = 7, где 10 и 7- позиции слов “поступили” и “которые”. В остальных позициях массива posconnectrword расположен 0.

Если pos – позиция глагола в придаточном определительном предложении с уровнем глубины depth > 1, k - позиция союзного слова в том же придаточном предложении, m – позиция существительного, на которое дается ссылка союзным словом, то Matr[pos, contr] = Matr[k, contr] = m.

Пример. Рассмотрим размеченное представление текста Т1:

“Профессор (1) Игорь (2) Сергеевич (3) Сомов (4) , (5) о (6) котором (7) пишет (8) газета (9) “Поиск” (10) в (11) номере (12) , (13) поступившем (14) в (15) субботу (16) , (17) работает (18) в (19) МИФИ(20) . (21)”

Тогда *Matr[8, contr] = Matr[7, contr] =1* (позиция слова “профессор”) ,

posconnectword [8] = 7 (позиция слова “ котором ”) ,

Matr[14, contr] =12 (позиция слова “ номере ”) .

Назначение двумерного целочисленного массива pos-free-dep[1: 4, 1: nt] с количеством строк 4 и количеством строк nt (равным количеству значащих строк Rc - классифицирующего представления входного текста) заключается в следующем. Пусть $1 \leq \text{depth} \leq 4$. Тогда строка с номером depth массива pos-free-dep содержит позиции таких единиц фрагмента текста с уровнем глубины depth, для которых в данный момент анализа текста еще не найдена смысловая связь с глаголом на том же уровне глубины.

Пример. Рассмотрим размеченное представление текста Т4 = “С (1) 2001-го года (2) АО (3) ”Радуга” (4) экспортирует (5) станки (6) в (7) Болгарию (8) . (9)”. Тогда в момент рассмотрения единицы ”Радуга” в позиции 4 массив pos-free-dep должен иметь следующую конфигурацию:

2	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Табл. 8.3. Структура массива pos-free-dep

Значением элемента numb-free-dep[depth] одномерного массива numb-free-dep длины 4 является количество существительных и конструктов в анализируемом фрагменте с уровнем глубины depth рассматриваемого предложения, для которых в данный момент обработки предложения еще не найдена смысловая связь с глагольной формой из того же фрагмента с уровнем глубины depth.

Пример. Для сформулированного выше вопроса Т3 после обработки словоформы “пятницу” в позиции 9 numb-free-dep[1] = 2, поскольку в главном предложении (уровень глубины 1) встретились существительные “контейнеров” и “керамикой”, и numb-free-dep[2] = 2, так как в придаточном предложении встретилось союзное слово “которые” и существительное “пятницу”.

Одномерный массив nmasters[1 : nt] предназначен для отображения количества семантических отношений между элементарной значащей единицей текста с произвольным порядковым номером k, $1 \leq k \leq nt$, и другой элементарной значащей единицей текста.

Пример. В классифицирующем представлении Rc входного текста (вопроса) Т3 слову “контейнеров” соответствует строка с номером k = 2. Между значением этого слова и значениями слов “поступили”, “ были отправлены” (порядковые номера 10 и 14 в Rc) существуют семантические отношения R1, R2. Поэтому nmasters[2] = 2.

8.6.2. Описание алгоритма “Обработка запятой”

Внешняя спецификация алгоритма “Обработка запятой”

Вход: Rc – классифицирующее представление входного текста T (см. параграф 7.1);

p – цел – позиция запятой; kindtext – строка – обозначение вида текста; depth – цел - значение уровня глубины к моменту рассмотрения запятой в позиции p; verbposmag[1 : 4], pos-free-dep[1: nt, 1:4], numb-free-dep[1 : 4] - целочисленные массивы (см. выше описание принципов использования этих массивов).

Выход: p, pos - целочисленные переменные; verbposmag[1 : 4], pos-free-dep[1: nt, 1:4], numb- free-dep[1 : 4] - целочисленные массивы.

Алгоритм «Обработка запятой»

Нач если ((kindtext = specqs-rol) И (depth = 1) И (verbposmag[1] = 0)
И ((Rc[p + 1, subclass] = вопр-относ-местоим)
ИЛИ (Rc[p + 1, subclass] = местоим-наречие)
ИЛИ ((Rc[p + 1, tclass] = class) = предлог)
И (Rc[p + 2, subclass] = вопр-относ-местоим)))
То Пустой оператор {в этом случае запятая разделяет вопросительные слова в начале вопроса}
иначе ind:=0;

Если Rc[p+1, tclass] = ’причаст’ то ind:=1 кесли
Если Rc[p+1, subclass] = ’вопр-относ-местоим’
То ind:=2 кесли
Если ((Rc[p+1, tclass] =’предлог’) И (Rc[p+2, subclass] = ’вопр-относ-местоим’)) то ind := 3 кесли

Выбор ind из
0: {Возвращение на предыдущий уровень глубины}
verbposmag[depth] : = 0; numb-free-dep[depth] := 0;
Обнулить элементы строки с номером depth массива pos-free-dep ;
Depth := depth-1 ;
1, 2, 3 : depth := depth + 1{переход на следующий уровень глубины}
Квыбор кесли конец

8.6.3. Описание алгоритма «Обработка-местоимения»

Условие вызова алгоритма: Rc[p, tclass] = ’местоим’, причем либо местоимение в позиции p входит в вопросительное ролевое словосочетание в начале вопросительного предложения, либо местоимение играет роль союзного слова, соединяющего придаточное определительное предложение с главным предложением.

Описание вспомогательного алгоритма «Поиск-существительного»

Условие вызова алгоритма: в позиции pos массива Rc располагается причастие или словоформа с лексемой «который» или «какой». При этом в главном предложении слева от союзного слова есть по крайней мере одно существительное. Слева от позиции pos ищется такое существительное, к которому в первом случае «прикреплено” причастие, а во втором рассматриваемом случае это существительное является референтом союзного слова.

Внешняя спецификация алгоритма

Вход: Rc – классифицирующее представление текста, p – цел – позиция причастия или словоформы с лексемой «который» или «какой»; массив verbposmag[1:4]; Matr – МССП текста.

Выход: poscontr – цел – позиция существительного, к которому “прикреплены” причастный оборот или придаточное определительное предложение.

Алгоритм “Поиск-существительного”

```
Нач   log:=ложь,    k1:= pos , posvb1:= verbposmag[depth-1]
      цикл-до       k1:= k1-1;
                    part1:= Rc[k1, tclass];
                    если (part1=’сущ’) и
                     ((Matr[k1, posdir, nmasters[k1]]= posvb1) ИЛИ
                     (Matr[k1, posdir, nmasters[k1]]= 0))
                    то      log:=  Ист;    poscontr := k1
      выход при (log = Ист)       кцикл
кон
```

Пример: Пусть T = Сколько контейнеров с индийской керамикой, которые в пятницу поступили из Новороссийска, предназначены для АО «Парус»?". В про-цессе разбора T слева направо найдем зависимость [контейнеров] --->[керамикой]. Так как у элемента 'контейнеров' пока не найдено управляющего слова, то этот элемент является референтом слова «который».

Внешняя спецификация алгоритма «Обработка-местоимения»

Вход: Rc и Rm– классифицирующее и морфологическое представления входного текста T (см. параграф 4.9); Rqs – словарь вопросительных словосочетаний; subclass – строка – значение подкласса местоимения; p – цел – позиция местоимения; leftprep – строка – значение предлога слева (включая пустой предлог nil); depth – цел - значение уровня глубины к моменту рассмотрения местоимения; verbposmag[1 : 4], pos-free-dep[1: nt, 1:4], numb-free-dep[1 : 4] - целочисленные массивы (см. выше описание принципов их использования).
Выход: p - целочисленная переменная; verbposmag[1 : 4], pos-free-dep[1: nt, 1:4], numb- free-dep[1 : 4] - целочисленные массивы.

Алгоритм «Обработка-местоимения»

Нач если ((subclass] = вопр-относ-местоим) И (depth = 1)
И (verbposmag[depth] = 0))
то Обработка-ролевых-вопрос-слов (Rc, Rm, Rqs, leftprep)
иначе нач если (((Rc[p – 1, unit] ≠ ',') И (Rc[p-1, tclass] ≠ предлог))
ИЛИ ((Rc[p-1, tclass] = предлог) И (Rc[p – 2, unit] ≠ ',')))
то {во входном тексте пропущена запятая перед придаточным определительным предложением} depth := depth + 1 кесли
Поиск-существительного (p, poscontr)
Matr [p, contr] := poscontr; numb-free-dep[depth] := 1;
{в придаточном определительном предложении с уровнем глубины depth найдено первое слово, для которого пока отсутствует управляющая стрелка (с меткой семантического отношения) из глагольной формы в позиции verbposmag[depth] (пока в этой позиции расположен 0)}
pos-free-dep[depth, 1] := p кон
кон

8.6.4. Описание алгоритма “Обработка наречия”

Внешняя спецификация алгоритма

Вход: Rc – классифицирующее представление входного текста T (см. параграф 7.1);

p – цел – позиция наречия; subclass – строка – значение подкласса наречия; depth – цел - значение уровня глубины к моменту рассмотрения наречия в позиции p; numbqswd – цел – количество вопросительных слов; posqswd [1 : nt] – массив для представления позиций вопросительных слов; verbposmag[1 : 4], pos-free-dep[1: nt, 1:4], numb-free-dep[1 : 4] - целочисленные массивы (см. выше описание принципов использования этих массивов).

Выход: p, numbqswd - целочисленные переменные; posqswd [1 : nt], verbposmag[1 : 4], pos-free-dep[1: nt, 1:4], numb- free-dep[1 : 4] - целочисленные массивы.

Алгоритм “Обработка наречия”

Нач если (subclass = местоим-наречие) И (depth = 1)

To leftprep := nil;

Обработка-вопрос-слов (p, Rc, Rm, Rqs, leftprep, numbqswd, posqswd, Matr)

кесли кон

8.7. Разработка алгоритма поиска возможных смысловых связей между глагольной формой и значением зависящей от нее группы слов

8.7.1. Основные идеи формализации необходимых условий существования смысловой связи между значением глагольной формы и значением зависящей от нее группы слов.

Субстантивными выражениями будем называть существительные, а также существительные с зависимыми словами, обозначающие понятия, предметы и множества предметов (сочетание noun substantive означает в английском языке имя существительное). Например, пусть T1=“Откуда и для кого поступили два алюминиевых контейнера с керамической плиткой?”, T2=“Когда поступила статья профессора А.П.Сомова?” и T3 =“Поставь синюю коробку на зеленый

ящик”. Тогда сочетания “два алюминиевых контейнера”, “статья профессора А.П.Сомова”, “синюю коробку” являются субстантивными выражениями.

Под глагольной формой будем понимать глагол в личной или неопределенной форме либо причастие. Установление возможных смысловых отношений между глагольной формой и словосочетанием, включающем существительное или вопросительно-относительное местоимение, играет важную роль в процессе осуществления семантико-синтаксического анализа ЕЯ-текста.

Будем полагать, что posvb - это позиция в представлении Rc глагольной формы, posdepword - позиция в представлении Rc существительного или вопросительно-относительного местоимения..

Входными данными алгоритма “Найти-множ-тематич-ролей” являются натуральные числа posvb, posdepword, а также двумерные массивы Arls, Arvfr, где Arls – проекция лексико-семантического словаря Lsdic на входной текст, Arvfr – проекция словаря глагольно-предложных фреймов Vfr на входной текст.

Назначение алгоритма “Найти-множ-тематич-ролей” заключается, во-первых в нахождении целого числа nrelvbdep – количества возможных смысловых отношений между значениями единиц текста с номарами p1 и p2 в представлении Rc.

Во-вторых, этот алгоритм должен строить вспомогательный двумерный массив arrelvbdep, хранящий информацию о возможных смысловых связях между единицами Rc с номерами p1 и p2. Строки этого массива представляют информацию о комбинациях значений глагольной формы и зависимой группы слов (или одного слова). Структура каждой строки представлена на Рис. 8.1.

Arrelvbdep

Linenoun	Linevb	trole	example

Рис. 8.1. Структура строки вспомогательного массива Arrelvbdep

Для k-й заполненной строки массива Arrelvbdep (k>=1) linenoun - номер строки из массива Arls, соответствующего слову в позиции p1; linevb – номер набора из массива Arls, соответствующего глагольной форме в позиции p2; trole – обозначение смыслового отношения (тематической роли), связывающего

глагольную форму в позиции p2 и зависимое слово в позиции p1; example – пример выражения на ЕЯ, в котором реализуется та же самая тематическая роль.

Поиск возможных смысловых отношений между значением глагольной формы (ГФ) и значением зависимой группы слов (ЗГС) осуществляется с помощью проекции на входной текст словаря глагольно-предложных фреймов (с.г.п.ф.) Arvfr. В этом словаре ищется такой шаблон (или шаблоны), который был бы совместим с некоторыми семантико-синтаксическими характеристиками ГФ в позиции posvb и ЗГС, имеющей номер posdepword в Rc.

К таким характеристикам, во-первых, относится множество кодов грамматических падежей Grcases, ассоциированных с текстообразующей единицей, порядковым номером, которым в Rc является величина posdepwd.

Предположим, что Rc[posvb, tclass] = глаг, Rc[posdepword, tclass] = сущ

или Rc[posdepword, subclass]= вопрос-относ-местоим.

Тогда Grcases – это множество грамматических падежей, соответствующих существительному или вопросительно-относительному местоимению в позиции posdepword.

Пример. Пусть B1=”Какие(1) лекарственные(2) препараты(3) выпускаются(4) на(5) фабрике(6) “Рассвет”(7) ?(8)” и B2=”Где(1) работает(2) профессор(3) И.П.(4) Семенов(5) ,(6) о (7) котором(8) пишет(9) газета(10) «Поиск»(11) в(12) последнем(13) номере(14) ?(15)”

Пусть для B1 posvb=4, posdepword=6 (позиция слов «выпускаются» и «фабрике»), для B2 posvb=9, posdepword=8 (позиции слов «пишет» и «котором»). Тогда в первом случае Grcases:={3,6}, т.к. словоформа «фабрике» может находиться как в дательном падеже (код 3), так и в предложном падже (код 6). Для второго случая Grcases:={6}, поскольку словоформа «котором» относится к предложному падежу.

При поиске возможных смысловых отношений в сочетаниях «Существительное+Причастие» («препарат, выпускаемый», «сотрудники, работающие») используются шаблоны из множества Arvfr, связанные со значениями глаголов, от которых образованы причастия.

Например, при поиске возможных смысловых отношений между причастием и существительным в сочетании «препарат, выпускаемый» используется

семантико-синтаксический шаблон, позволяющий найти тематическую роль (роли) в сочетании «препарат был выпущен».

Чтобы найти смысловое отношение (отношения) в сочетании «сотрудников, работающих», будем фактически искать тематическую роль (роли) в сочетании «сотрудники работают».

Учитывая сказанное выше, процесс поиска в предложении тематической роли, связывающей глагольную форму в позиции posvb и слово (существительное или относительное местоимение) в позиции posdepword, к которому относится предлог prep (возможно, пустой предлог nil) можно пояснить следующим образом:

В тройном цикле по параметрам i1, i2, k1, где i1, i2 - номера наборов строк из множества Arls, соответствующие существительному или вопросительно-относительному местоимению в позиции posdepword и глагольной форме в позиции posvb, k1 – номер набора из проекции словаря глагольно-предложных фреймов Arvfr, ищется такое сочетание значений параметров i1, i2, k1, что выполняются следующие условия:

Если sem1 - значение поля sem для набора с номером i1 из Arls, sem2 – значение поля sem для набора с номером i2 из Arls, и *semsit1, trole1, sprep1, grc1* – значения полей semsit, trole1, sprep, grcase набора с номером k1 из Arvfr, такие, что

Semsit1=sem2, sprep1=prep, grc1 ∈ Grcases, то выполняется соотношение (см. параграф 4.6)

(T, posdepword, sem1, prep, grc1, posvb, sem2, relat1) ∈ Смысл-связь1.

8.7.2. Описание алгоритма поиска возможных смысловых связей между глагольной формой и субстантивным выражением

Назначение алгоритма “Найти-множ-отнош-глаг-сущ”

Установить тематическую роль, связывающую глагольную форму в позиции **posvb** и слова (существительного или союзного слова) в позиции **posdepword** с учетом возможного предлога перед этим словом. Как следствие, выбрать одно из нескольких возможных значений глагольной формы и одно из нескольких возможных значений слова в позиции **posdepword**. Для этого потребуются три

вложенных цикла: (1) по возможным значениям слова в позиции **posdepword**, (2) по возможным значениям глагольной формы; (3) по глагольно-предложным фреймам, связанным с данной глагольной формой.

Внешняя спецификация алгоритма алгоритма “Найти-множ-отнош-глаг-сущ”

Вход **Rc** - классифицирующее представление, **nt** – цел - количество единиц текста в классифицирующем представлении Rc, т.е. количество строк в Rc,
Rm – морфологическое представление лексических единиц, входящих в Rc,
posvb – цел – позиция глагольной формы (глагола в личной или неопределенной форме, причастия), **posdepword** – цел – позиция существительного или вопросительно-относительного местоимения (“котором” в сочетании “о котором”, являющемся началом придаточного определительного предложения и т.д.), **depth** – цел - значение уровня глубины вложенности для слова в позиции **posdepword** ,
Matr – начальное значение МССП текста; **Arls** – массив – проекция лексико-семантического словаря (ЛСС) **Lsdic** на входной текст **T**; **Arvfr** – массив – проекция словаря глагольно-предложных фреймов **Vfr** на входной текст **T**.
Выход

arrelvbdep – одномерный массив, предназначенный для представления информации о значении зависимого слова, значении глагольной формы и о смысловом отношении между глагольной формой в позиции posvb и зависимым словом в позиции posdepword;
nrelvbdep – цел - количество значащих строк в массиве arrelvbdep.

Внешние спецификации вспомогательных алгоритмов

Спецификация алгоритма “Признаки-глаг-формы”

Вход: p1 – номер строки из Rc, соответствующей глаголу или причастию.
Выход:.form1, refl1, voice1 – строки, значения которых определяются следующим образом. Если p1 – позиция глагола, то form1 может иметь одно из следующих значений: *изъявит* (признак изъявительного наклонения), *неопр* (признак неопределенной формы глагола), *повелит* (признак повелительного

наклонения). Если p1 – позиция причастия, то form1 := *изъявит.* Строка refl1 получает значение *действит* (признак действительного залога) или *страд* (признак страдательного залога). Значения параметров form1, refl1, voice1 вычисляются по набору числовых кодов значений морфологических признаков, связанных с текстовой единицей с порядковым номером p1.

Спецификация алгоритма “Спектр-сорта”

Вход: z - сорт, т.е. элемент множества St(B(Cb(Lingb))), где Lingb – лингвистический базис.

Выход: spectrum - множество всех сортов, являюшихся обобщениями сорта z, включая сорт z..

Алгоритм “Найти-множ-отнош-глаг-сущ”

Нач Признаки-глаг-формы (posvb, form1, refl1, voice1)

nrelvbdep := 0

{Далее вычисляется предлог}

если ((Rc[posvb, tclass] = прич) И (posdepword = Matr[posvb, contr])

то prep := nil иначе prep := leftprep кесли

{ Вычисление posn1 – позиции существительного, которое определяет множество сортов для текстовой единицы в позиции posdepword }

если (Rc[posdepword, subclass] = вопрос-относ-местоим)

и (Rc[posdepword, unit] – слово с лексемой “который” или “какой»)

то posn1 := Matr[posdepword, contr]

иначе posn1 := posdepword кесли

{ Далее вычисляется множество грамматических падежей Grcases, которое будет связано со словом в позиции posdepword для нахожднгтя множества смысловых отношений между словами в позициях posvb и posdepword }

t1 := Rc[posvb, tclass]; t2 := Rc[posvb, subclass];

если t1 = прич то если posdepword = Matr[posvb, contr]

то Grcases := {1} кесли

иначе {в случае posdepword ≠ poscontrword [posvb]} переход к L1 кесли

иначе {т.е. в случае t1 = глаг} переход к L1 кесли

L1: p1 := Rc[posdepword, mcoord];

Grcases := Падежи (Rm[p1, morph])

кесли

line1 := Matr[posn1, locunit]; numb1 := Matr[posn1, nval]

{количество строк в Arls со значениями существительного}

цикл для i1 от line1 до line1 + numb1 – 1 {цикл по строкам массива Arls, соответствующих существительному в позиции posn1}

Set1 := пустое множество

цикл для j от 1 до m {m – семантическая размерность сортовой системы S(B(Cb(Lingb))), т.е. наибольшее количество несравнимых сортов, которые могут характеризовать одну сущность}

current-sort := Arls[i1, st_j];

если current-sort ≠ nil

то Спектр-сорта (current-sort, spectrum);

Set1 := Объединение множеств Set1 и spectrum кесли

{для произвольного сорта z значением spectrum является множество всех сортов, являюшихся обобщениями сорта z, включая сорт z} кцикл {по j}

{Далее следует цикл по значениям глагольной формы}

line2 := Matr[posvb,locunit]

numb2 := Matr[posvb,nval]

{количество строк в Arls со значениями глагольной формы}

цикл для i2 от line2 до line2 + numb2 - 1

{цикл по строкам массива Arls соотв. глаг. в позиции posvb}

current-pred := Arls[i2, sem]

<u>цикл</u> для k1=1 до narvfr

<u>если</u> Arvfr[k1, semsit] = current-pred

<u>то</u> <u>нач</u>

s1 := Arvfr[k1, str]

<u>если</u> ((prep=Arvfr[k1, sprep] и (s1∈ Set1) и (form1 =Arvfr[k1, form]) и и (refl1 =Arvfr[k1, refl]) и (voice1 =Arvfr[k1, voice]))

<u>то</u> grc := arvfr[k1, grcase]

<u>если</u> (grc ∈ Grcases)

<u>то</u> {отношение существует}

nrelvbdep:=nrelvbdep+1; arrelvbdep[nrelvbdep, linevb] := i2 ;

arrelvbdep[nrelvbdep, linenoun] := i1 ; arrelvbdep[nrelvbdep, gr] := grc ;

arrelvbdep[nrelvbdep+1, role] := arvfr[k1, trole]

кесли

кесли

конец

кесли

кесли

кесли

конец

Комментарий к алгоритму "Найти-множ-отнош-глаг-сущ"

Найдено количество nrelvbdep смысловых отношений между глагольной формой и зависимым от нее существительным. Рассматривается такой подъязык русского языка, что в вопросах всегда после глагола находится хотя бы одно существительное. Информация о таких комбинациях значений глагола V и существительного N1, которое даёт хотя бы одно смысловое отношение между V и N1, отображена во вспомогательном массиве arrelvbdep:

linevb	linenoun	role	example
c1	c2		Поступила цистерна
…			

Рис. 8.2. Структура строки вспомогательного массива arrelvbdep.

В столбце linevb помещается c1 – номер строки массива Arls, для которой Arls[c1, numb] = posvb, т.е. строка c1 указывает какое-то одно значение глагола V в позиции posvb. Например, для B1 = "Откуда и для кого поступили три алюминиевых контейнера с керамикой?" в столбце linevb ставится c1 – номер строки массива Arls, такой, что Arls[c1,sem] = поступление2.

В столбце linenoun ставится c2 – номер строки массива Arls, такой что Arls[c2, numb] = posn1 (позиция существительного n1). Например, для B1 Arls[c2,

sem]=контейнер. Солбец role предназначен для отображения возможных отношений между глаголом V и существительным N1.

Если nrelvbdep = 0, то не найдено смысловых отношений. Будем предполагать, что это невозможно для рассматриваемого входного языка. Если nrelvbdep = 1, то однозначно определены значение глагола V (по строке c1), значение существительного N1 (по строке c2) и значение смыслового отношения arrelvbdep [nrelvbdep, role]. Например, для вопроса B1 выполняются соотношения V=”посту-пили”, N1=”контейнера”, nrelvbdep = 1, arrelvbdep [nrelvbdep, role] = Объект1.

Если nrelvbdep > 1 то необходимо вызвать процедуру, которая задает уточняющие вопросы пользователю, и сформировать эти вопросы на основе примеров в столбце example.

8.7.3. Описание алгоритма обработки конструктов

Назначение алгоритма “Найти-множ-отнош-глаг-конструкт”: установить тематическую роль, связывающую глагольную форму в позиции **posvb** и конструкт в позиции **posdep** с учетом возможного предлога перед этим конструктом. Как следствие, выбрать одно из нескольких возможных значений глагольной формы. Для этого потребуются два вложенных цикла: (1) по возможным значениям глагольной формы; (2) по глагольно-предложным фреймам, связанным с данной глагольной формой.

Внешняя спецификация алгоритма

Вход: posvb – цел – позиция глагольной формы (глагола в личной или неопределенной форме, причастия), posdep (сокращение от “position of dependent word”) – цел – позиция конструкта (выражения, обозначающего числовое значение параметра), subclass1 – строка – обозначение сорта конструкта, Matr – МССП текста; Arls – проекция лексико-семантического словаря на входной текст; Arvfr - проекция словаря глагольно-предложных фреймов на входной текст;

prep1 – строка – предлог, относящийся к конструкту, или пустой предлог nil.

Выход:

arrelvbdep – двумерный массив, предназначенный для представления информации о значении глагольной формы и смысловом отношении между глагольной формой в позиции posvb и конструктом в позиции posdep;

nrelvbdep – цел – количество значащих строк в массиве arrelvbdep.

Алгоритм “Найти-множ-отнош-глаг-конструкт”

```
Нач     startline  := Matr[posvb, locunit] – 1 ;
        Line1 := startline;    numbvalvb := Matr[posvb, nval]
{количество возможных значений глагольной формы}
цикл-до
        Line1:= Line1 + 1;   Current-pred := Arls[line1, sem]
        K1:=0; log1:=false
        Цикл-до
                k1:=k1+1
                если (Arvfr[k1, semsit] =current-pred)
                то     если (Arvfr[k1, str] = subclass1) и (Arvfr[k1, sprep] = prep1)
                         то {отношение существует}
                     nrelvbdep := 1; arrelvbdep[1, linevb] := line1
                arrelvbdep[1, role] := Arvfr[k1, trole];   Log1:=true
        Выход-при (log1=true)      Кцикл
Выход-при ((log1=true) или (line1 = startline + numbvalvb))          кцикл
          leftprep := nil       кон
```

8.7.4. Описание алгоритма “Найти-множ-тематических-ролей”

Назначение алгоритма: установить множество тематических ролей, связывающих глагольную форму в позиции **posvb** и слово (существительного, союзного слова, конструкт) в позиции **posdep** с учетом возможного предлога перед этим словом. Для этого потребуются три вложенных цикла: (1) по возможным значениям слова в позиции **posdep**, (2) по возможным значениям глагольной формы; (3) по глагольно-предложным фреймам, связанным с данной глагольной формой.

Внешняя спецификация

Вход

Rc - классифицирующее представление текста, nt – цел - количество единиц текста в Rc, т.е. количество строк в Rc, Rm – морфологическое представление лексических единиц, входящих в Rc, posvb – цел – позиция глагольной формы (глагола в личной или неопределенной форме, причастия),

posdep (сокращение от “position of dependent word”) – цел – позиция зависимого слова (существительного, вопросительно-относительного местоимения, конструкта - выражения, обозначающего числовое значение параметра),

Matr – строково-числовая матрица – исходное МССП текста,

depth – цел - значение уровня глубины вложенности для слова в позиции **posdep**,

Arls – массив – проекция лексико-семантического словаря (ЛСС) Lsdic на входной текст T; Arvfr – массив – проекция словаря глагольно-предложных фреймов Vfr на входной текст T,

nmasters [1:nt] – массив для отображения количества управляющих слов для каждой единицы текста.

Выход:

class1 – строка – обозначение класса текстовой единицы в позиции posdep,

subclass1 - обозначение подкласса текстовой единицы в позиции posdep,

arrelvbdep – двумерный массив, предназначенный для представления информации о значении зависимого слова, значении глагольной формы и о смысловом отношении между глагольной формой в позиции posvb и зависимым словом в позиции posdep,

nrelvbdep – цел - количество значащих элементов в массиве arrelvbdep.

Алгоритм “Найти-множ-тематических-ролей”

Нач Заполнить числом 0 все числовые позиции массива arrelvbdep и заполнить пустым элементом *nil* все строковые позиции массива arrelvbdep

class1 := Rc[posdep,tclass]

subclass1 := Rc[posdep, subclass]; prep1 := Matr [posdep, prep]

если (class1 = сущ) ИЛИ ((class1 = местоим) И (subclass1 = вопросит-относит-местоим))

то Найти-множ-отнош-глаг-сущ (Rc, Rm, posvb, posdep, prep1, depth, Arls, Arvfr, Matr, nmasters, nrelvbdep, arrelvbdep) **кесли**

если (class1 = констр) **то** Найти-множ-отнош-глаг-конструкт (posvb, posdep, prep1, subclass1, depth, Arls, Arvfr, Matr, nmasters, nrelvbdep, arrelvbdep) **кесли конец**

8.7.5. Описание алгоритма поиска смысловой связи между глагольной формой и зависимым выражением

Назначение алгоритма "Смысл-связь-глаг-формы": установить тематическую роль, связывающую глагольную форму в позиции **posvb** и выражение (существительное, союзное слово, конструкт) в позиции **posdep** с учетом возможного предлога перед этим выражением. Как следствие, выбрать одно из нескольких возможных значений глагольной формы и одно из нескольких возможных значений слова в позиции **posdep**.

Занести полученную информацию о значении глагольной формы, значении зависимой единицы текста и о смысловом отношении (т.е. о тематической роли) в МССП Matr.

Внешняя спецификация алгоритма

Вход: **Rc** - классифицирующее представление, **nt** – цел - количество единиц текста в Rc, т.е. количество строк в Rc,

Rm – морфологическое представление лексических единиц, входящих в Rc,

posvb – цел – позиция глагольной формы (глагола в личной или неопределенной форме, причастия), **posdep** – цел – позиция существительного или относительного местоимения ("котором" в сочетании "о котором", являющемся началом придаточного определительного предложения и т.д.)

depth – цел - значение уровня глубины вложенности для слова в позиции **posdep**,

Arls – массив – проекция лексико-семантического словаря (ЛСС) **Lsdic** на входной текст **T**; **Arvfr** – массив – проекция словаря глагольно-предложных фреймов **Vfr** на входной текст **T**;

Matr – начальное значение МССП текста; **nmasters [1:nt]** – массив для отображения количества управляющих слов для каждой единицы текста.

Выход: **Matr –** строково-числовая матрица – преобразованное значение исходной матрицы **Matr**.

Внешняя спецификация алгоритма “Выбор-тематич-роли”

Вход: posvb –цел – позиция глагольной формы; posdep - цел – позиция зависимой единицы текста (существительного или конструкта); arrelvbdep – двумерный массив, представляющий информацию о возможных комбинациях значения глагольной формы в позиции posvb, значения зависимой единицы в позиции posdep и тематической роли rel, реализующейся в таком сочетании (см. описание массива arrelvbdep в подпараграфе 5.7.2); nrelvbdep – цел – количество значащих строк в массиве arrelvbdep, т.е. количество возможных смысловых отношений между рассматриваемыми глагольной формой и зависимой единицей.

Выход: m1 **–** цел – номер некоторой значащей строки массива arrelvbdep. Параметр m1 приобретает ненулевое значение в результате обработки ответа пользователя на уточняющий вопрос ЛП. Пользователю предлагается указать, какое из нескольких смысловых отношений реализуется в сочетании “Глагольная форма в позиции posvb + Зависимая единица в позиции posdep”. Для этого пользователю с помощью столбца example даются примеры сочетаний, в которых реализуется такое же смысловое отношение, как и потенциально возможное отношение между единицами текста в позициях posvb и posdep.

Алгоритм “Смысл-связь-глаг-формы”

нач Найти-множ-тематических-ролей (Rc, Rm. posvb, posdep, depth, Arls, Arvfr, Matr, nmasters, nrelvbdep, arrelvbdep)

{Найти количество элементов массива nrelvbdep и массив arrelvbdep, описывающий возможные смысловые отношения между глагольной формой и зависимой единицей текста}

если nrelvbdep = 1 то m1 := 1

иначе Выбор-тематич-роли (posvb, posdep, nrelvbdep, arrelvbdep, m1)

{m1 — номер строки в массиве arrelvbdep, дающей реализуемое в Т сочетание значения глагольной формы, значения зависимой единицы текста, и смыслового отношения между глагольной формой в позиции posvb и единицей текста в позиции posdep с учетом предлога, который может относиться к позиции posdep}

кесли

rel1 := arrelvbdep[m1,role]

locvb := arrelvbdep [m1,linevb] {строка из Arls}

если (class1 = сущ) то locnoun := arrelvbdep [m1,linenoun] {строка из Arls} кесли

⇒ {Внесение информации в Matr (см. описание Matr в главе 4)}

Matr[posvb].posdir := 0, Matr[posvb,locunit] := locvb, Matr[posvb,nval] := 1

если (class1 = сущ) то Matr[posdep,locunit] := locnoun кесли

Matr[posdep,nval] := 1, Matr[posvb].ndep := Matr[posvb].ndep + 1

Matr[posdep1].posdir := posvb, Matr[posdep].reldir := rel1

Конец

Комментарий к алгоритму "Смысл-связь-глаг-формы". Если nrelvbdep >1 то необходимо задать уточняющие вопросы пользователю, используя поле example массива arrelvbdep, и найти: locnoun – строку из Arls, указывающую значение единицы текста в позиции posdep, если эта единица является существительным; locvb – строку из Arls, указывающую значение глагольной формой в позиции posvb; rel1 – смысловое отношение (тематическую роль), между единицами текста в позициях posvb и posdep.

Если nrelvbdep =1, то m1 := 1, поэтому locvb= arrelvbn [1,linevb];

Locnoun = arrelvbn [1,linenoun]; rel1= arrelvbn [1,role].

Далее полученную информация запоминается в матрице Matr*:*

Matr[posdep,locunit]:=locnoun; Matr[posdep].posdir:=posvb;

Matr[posndep].reldir:=rel1.

В результате проведенного анализа однозначно определяются значения как глагольной формы в позиции posvb, так и существительного в позиции posdep. Поэтому Matr[posvb,nval] := 1 , Matr[posdep,nval] := 1 .

8.7.6 Заключительная часть описания алгоритма обработки глагольных форм

Внешняя спецификация алгоритма «Обработка-глаг-формы”

Вход: Rc, Rm**,** МССП Matr, Arls, Arvfr, одномерные массивы verbposmag[1:4], posqswd[1:nt], двумерный массив pos-free-dep[1: nt, 1:4], одномерный массив numb-free-dep[1: 4], целочисленные переменные pos, nsit, numbqswd, переменная depth (номер уровня глубины вложения рассматриваемого фрагмента текста), class – строка, обозначающая часть речи глагольной формы.

Выход: преобразованное значение МССП Matr..

Алгоритм “Обработка-глаг-формы”

Нач nsit := nsit + 1

(nsit – количество уже упомянутых в тексте ситуаций}

Matr[pos].mark := Var (‘e’, nsit)

verbposmag [depth] := pos,

Если ((class = прич)

То если Rc[pos, unit] ≠ ‘,’ то depth := depth+1 кесли

{Учтена возможность отсутствия запятой перед причастием, с которого начинается причастный оборот}

Поиск-существительного(pos, poscontr, Rc, Matr)

{см. описание алгоритма Поиск-существительного в подразделе 8.6.3}

Matr [pos, contr] := poscontr

{Пример. Пусть T1 = “Сколько предприятий, расположенных в Саратовской области, экспортируют продукцию в Болгарию?, и pos = 4. Тогда Matr [4, contr] := 2 (позиция слова “предприятий”)}

posvb := pos; posdepword := poscontr;

{Далее находится смысловое отношение между причастием в позиции posvb и управляюшим существительным в позиции poscontr}

class1 := сущ ; subclass := Rc[posdep, subclass];

nmasters[posdep] := nmasters[posdep] + 1;

Смысл-связь-глаг-формы (Rc, nt. Rm. posvb, posdep, class, subclass, class1, subclass1, depth, Arls, Arvfr, nmasters, Matr)

Кесли {завершение начальной части обработки причастия}

Если ((class = глаг) И (depth = 1) И (numbqswd > 0))

То {от позиции глагола в главном предложении проводятся управляющие стрелки с метками смысловых отношений (тематических ролей) к позициям вопросительных слов}

Цикл для k1 от 1 до numbqswd

P1 := posqswd [k1]; Matr[p1, posdir, 1] := pos

Кцикл

numbqswd := 0;

Кесли

Если numb-free-dep [depth] > 0 {на том же уровне глубины вложения существуют свободные единицы текста, т.е. такие единицы, для которых пока не найдено семантико-синтаксическое управление от другой единицы}

То Цикл для m1 от 1 до numb-free-dep [depth]

Смысл-связь-глаг-формы (Rc, nt, Rm, pos, pos-free-dep [depth, m1], depth, Arls, Arvfr, Matr, nmasters)

Кцикл конец

Пример. Пусть B1 – следующее размеченное представление вопроса: “Когда (1) и (2) где (3) будет проходить (4) очередная (5) международная (6) научная (7) конференция (8) “COLING” (9) ? (10)”. Тогда, если pos = 4, то будут проведены помеченные стрелки от позиции 4 к позициям 1 и 3 (в цикле по параметру k1 со значениями от 1 до numbqswd).

8.8. Обработка прилагательных , предлогов, количественных числительных, названий и существительных

8.8.1. Обработка прилагательных

Описание алгоритма “Обработка-прилаг”

Внешняя спецификация

Вход : pos – целое – порядковый номер единицы текста, являющейся прилагательным; Matr –МССП текста; Arls - массив – проекция лексико-семантического словаря (ЛСС) Lsdic на входной текст **T**.

Выход : nattr – целое – количество подряд идущих прилагательных; Attributes массив, имеющий следующую структуру:

Attributes

place	prop
позиция в Rc	семантическая единица для очередного прилагательного

Рис. 8.3. Структура строки вспомогательного массива Attributes

Пример. Пусть B1 = “Откуда (1) поступили (2) 2 (3) зеленых (4) алюминиевых (5) контейнера (6) ? (7)”. Тогда nattr:=2, а массив Attributes имеет следующий вид:

place	prop
4	Цвет (z1, зел)
5	Материал (z1 , алюм)
0	пустая строка

Рис. 8.4. Пример вспомогательного массива Attributes

Замечание. В конце алгоритма Обработка-сущ выполняются, в частности, следующие действия: nattr:=0; столбец place обнуляется, столбец prop заполняется цепочкой nil – обозначением пустой строки.

Алгоритм “Обработка-прилаг”

Нач nattr:=nattr+1, k1:= Matr[pos, locunit],
semprop:=Arls[k1, sem] ;
Attributes [nattr, place]:= pos,
Attributes [nattr, prop]:= semprop кон

8.8.2. Обработка предлогов, количественных числительных и названий

Алгоритмы “Обработка-предлога” и “Обработка-колич-числит” очень просты. Первый из них предназначен для запоминания предлога в рассматриваемой позиции pos с помощью переменной leftprep (“предлог слева”). Второй алгоритм преобразует лексическую единицу, относящуюся к классу количественных числительных, в число, обозначаемое данной лексической единицей. Например, слову “трех” и сочетанию “двадцать три“ соответствуют числа 3 и 23. Для запоминания числа предназначена переменная leftnumber (“число слева”). Входными параметрами этих алгоритмов являются классифицирующее представление текста Rc и переменная pos (номер строки в Rc).

Алгоритм “Обработка-предлога”

Нач Leftprep := Rc[pos, unit] кон

Алгоритм “Обработка-колич-числит”

Нач Leftnumber := Число (Rc[pos, unit]) кон

Описание алгоритма “Обработка-названий”

Внешняя спецификация алгоритма

Вход: Rc – классифицирующее представление текста, pos – позиция выражения в кавычках или апострофах, Matr – МССП текста.

Выход: преобразованное значение Matr.

Алгоритм

Нач Matr[pos, posdir, 1] := pos – 1; Matr[pos, reldir, 1] := ‘Название’

{Смысл операций: проведена управляющая стрелка с меткой ‘Название’ от выражения в кавычках или апострофах к существительному, стоящему слева от него} кон

8.8.3. Описание алгоритма поиска возможных смысловых связей между двумя существительными с учетом предлога

Назначение алгоритма “Найти-множ-отношений-сущ1-сущ2”

Алгоритм “Найти-множ-отношений-сущ1-сущ2” (“Найти-множество-смысловых-отношений-между-Существительным-1-и-Существительным-2”) позволяет установить смысловые отношения, которые могут существовать между существительным в позиции posn1 (в дальнейшем обозначается выражением Сущ1) и существительным в позиции posn2 (в дальнейшем обозначается выражением Сущ2) при условии, что ко второму существительному относится некоторый предлог, расположенный в позиции между posn1 и posn2.

Для этого потребуются три цикла: (1) по возможным значениям слова в позиции posn1, (2) по возможным значениям слова в позиции posn2, (3) по предложным фреймам, связанным с рассматриваемым предлогом..

Внешняя спецификация алгоритма алгоритма “Найти-множ-отношений-сущ1-сущ2”

Вход **Rc** - классифицирующее представление, **nt** – цел - – количество единиц текста в классифицирующем представлении R1, т.е. количество наборов в R1,

Rm – морфологическое представление лексических единиц, входящих в R1,

Posn1 – цел – позиция первого существительного, **Posn2** – цел – позиция второго существительного, **Matr** –МССП текста;

Arls – массив – проекция лексико-семантического словаря (ЛСС) **Lsdic** на входной текст **T**; **Arfrp** – массив – проекция словаря предложных фреймов **Frp** на входной текст **T**.

Выход **arrelvbdep** – двумерный массив, предназначенный для представления информации о значении первого существительного, значении второго существительного и о смысловом отношении между словом в позиции posn1 и зависимым словом в позиции posn2,

nreln1n2 – цел - количество значащих строк в массиве arrelvbdep.

Алгоритм "Найти-множ-отношений-сущ1-сущ2"

Нач nreln1n2 := 0

{Вычисление предлога} prep1 := Matr[posn2,prep]

{Вычисление множества грамматических падежей}

p1 := Rc[posn2,mcoord]; Grcases := Падежи (Rm[p1].morph)

line1 := Matr[posn1, locunit], numb1 := Matr[posn1, nval]

{количество строк в Arls со значениями существительного}

цикл для n1 от line1 до line1 + numb1 – 1 {цикл по строкам массива Arls, соответствующих существительному в позиции posn1}

Set1 := пустое множество

цикл для j от 1 до m {m – семантическая размерность сортовой системы S(B(Cb(Lingb))), т.е. наибольшее количество несравнимых сортов, которые могут характеризовать одну сущность}

current-sort := Arls[n1, st_j];

если current-sort ≠ nil то Спектр-сорта(current-sort, spectrum);

Set1 := Объединение множеств Set1 и spectrum кесли

{для произвольного сорта z spectrum (z) – это множество всех сортов, являюшихся обобщениями сорта z, включая сорт z} кцикл {по j}

{Пример Если u = дин.физ.об, то

spectrum (u) = { дин.физ.об, физ.об, простр.об }}

{цикл по значениям Сущ2}

line2 := Matr[posn2, locunit], numb2 := Matr[posn2, nval]

{количество строк в Arls со значениями Сущ2}

цикл для n2 от line2 до line2 + numb2 – 1 {цикл по строкам массива Arls, соответствующих существительному в позиции posn2}

Set2 := пустое множество

цикл для q от 1 до m {m – семантическая размерность сортовой системы S(B(Cb(Lingb))), т.е. наибольшее количество несравнимых сортов, которые могут характеризовать одну сущность}

current-sort := Arls[n2, st_q];

если current-sort ≠ nil то Спектр-сорта(current-sort, spectrum);

Set2 := Объединение множеств Set2 и spectrum кесли кцикл {по q}

цикл для k1=1 до narfrp {количество строк в массиве Arfrp – проекции словаря предложных фреймов Frp на входной текст}

если Arfrp[k1, prep] = prep1 {найден нужный предлог}

то нач s1 := Arfrp [k1, sr1]; s2 := Arfrp [k1, sr2];

если (s1∈ Set1) И (s2∈ Set2)

то если grc ∈ Grcases

то {отношение существует}

nreln1n2 := nreln1n2 + 1

arreln1n2 [nreln1n2, locn1] := n1; arreln1n2 [nreln1n2, locn2] := n2

arreln1n2 [nreln1n2, relname] := arfrp [k1, rel]

кесли

кесли

конец

кесли

кесли

кесли

конец

Комментарий к алгоритму “Найти-множ-отношений-сущ1-сущ2”

Найдено количество nreln1n2 смысловых отношений между существительными в позициях posn1 и posn2. Информация о таких комбинациях значений первого и второго существительных, которые дают хотя бы одно смысловое отношение между элементами в позициях posn1 и posn2, отображена во вспомогательном массиве arreln1n2:

Locn1	Locn2	relname	example
n1	n2	Против2	лекарство от астмы
…			

Рис. 8.5. Структура строки вспомогательного массива Attributes

- В столбце locn1 помещается n1 – номер строки массива Arls, задающей возможное значение существительного в позиции posn1.

- В столбце locn2 находится n2 – номер строки массива Arls, задающей возможное значение существительного в позиции posn2.

- Столбец relname предназначен для отображения возможных отношений между существительными в позициях posn1 и posn2.

Если nreln1n2 = 0, то не найдено смысловых отношений. Будем предполагать, что это невозможно для рассматриваемого входного языка.

Если nreln1n2 = 1, то однозначно определены значение существительного в позиции posn1 (по строке n1), значение существительного в позиции posn2 (по строке n2) и значение смыслового отношения arreln1n2 [nreln1n2, relname].

Если nreln1n2 > 1 то необходимо вызвать процедуру, которая задает уточняющие вопросы пользователю, и сформировать эти вопросы на основе примеров в столбце example.

План алгоритма “Обработка-сущ”

Нач

Занесение в Matr информации о стоящих (возможно) слева числе (или количественном числительном) и прилагательных посредством вызова алгоритма «Запись-атрибутов»

Генерация метки элемента и типа метки (вызов алгоритма «Вычисление-метки»)

Если Rc[pos +1, tclass] = сущ-собств то Обработка- сущ-собств кесли

Поиск смысловой зависимости от ближайшего слева существительного, управляемого глаголом в позиции verbposmag[depth]

Если такой зависимости нет

То в случае verbposmag[depth] ≠ 0 поиск смысловой зависимости от глагольной формы в позиции verbposmag[depth]

иначе (т.е. в случае verbposmag[depth] = 0) номер позиции pos заносится в массив свободных единиц текста pos-free-dep в строку depth, где depth – уровень глубины вложенности рассматриваемого фрагмента текста, включающего единицу в позиции pos кон

Внешняя спецификация алгоритма “Обработка-сущ”

Вход **Rc** - классифицирующее представление, **nt** – цел - – количество единиц текста в классифицирующем представлении Rc, т.е. количество наборов в Rc,

Rm – морфологическое представление лексических единиц, входящих в Rc,

pos – цел – позиция существительного, **depth** – цел - значение уровня глубины вложенности для слова в позиции **pos** ,

Matr – начальное значение МССП текста;

Arls – массив – проекция лексико-семантического словаря (ЛСС) **Lsdic** на входной текст **T**; **Arvfr** – массив – проекция словаря глагольно-предложных фреймов **Vfr** на входной текст **T**;

Arfrp – массив – проекция словаря предложных фреймов **Frp** на входной текст **T**.

Выход **pos** – цел - позиция единицы текста; **Matr** - преобразованное значение исходной матрицы **Matr**.

Внешние спецификации вспомогательных алгоритмов

Спецификация алгоритма “Найти-сущ-слева”

Вход: pos –цел – позиция существительного.

Выход: posleftnoun – цел – позиция ближайшего слева к позиции pos существительного, которое может оказаться управляющим словом для существительного в позиции pos (см. ниже подраздел “Описания вспомогательных алгоритмов”).

Спецификация алгоритма “Обработка-сущ-собств”

Вход: pos –цел – позиция существительного нарицательного или собственного, после которое следует хотя бы одно существительное собственное; Arls – проекция лексико-семантического словаря Lsdic на входной текст; Matr – исходное значение МССП текста.

Выход: Matr – преобразованное значение МССП текста (см. ниже подраздел “Описания вспомогательных алгоритмов”).

Спецификация алгоритма “Обработка-названий”

Вход: pos - позиция существительного нарицательного, после которого следует выражение в кавычках или апострофах; Matr – исходное значение МССП текста.

Выход: Matr – преобразованное значение МССП текста (см. подпараграф 8.8.2).

Спецификация алгоритма “Найти-множ-тематич-ролей”

Спецификация этого алгоритма и алгоритм приведены в подпараграфе 8.7.4.

Спецификация алгоритма “Смысл-связь-глаг-формы”

Спецификация этого алгоритма и алгоритм приведены в подпараграфе 8.7.5.

Спецификация алгоритма “Обработка-названий”

Спецификация этого алгоритма и алгоритм приведены в подпараграфе 8.8.2.

Спецификация алгоритма “Найти-множ-отношений-сущ1-сущ2“

Спецификация этого алгоритма и алгоритм приведены в подпараграфе 5.8.3.

Спецификация алгоритма “Выбор-управления-глаг-сущ”

Вход: **pos** – цел - позиция единицы текста; posvb – цел – позиция глагольной формы; posleftnoun – цел – позиция существительного слева; prep – строка – значение предлога, относящегося к позиции pos.

Выход: **res** – строка – получает значение 1 или 2 в результате уточняющего диалога с пользователем; если существительное в позиции pos непосредственно зависит от глагольной формы в позиции posvb, то res := 1; если существительное в позиции pos (с учетом предлога) непосредственно зависит от стоящего слева существительного в позиции posleftnoun, то res := 2.

Спецификация алгоритма “Выбор-отнош-между-сущ”

Вход: posleftnoun –цел – позиция существительного 1; pos - цел – позиция существительного 2, стоящего правее существительного 1; prep – строка – значение предлога (возможно, пустого предлога nil), относящегося к

существительному 2; arreln1n2 – двумерный массив, представляющий информацию о возможных комбинациях значения существительного 1, существительного 2 и смыслового отношения между ними с учетом предлога prep (см. описание массива arreln1n2 в подпараграфе 8.8.3); nreln1n2 – цел – количество значащих строк в массиве arreln1n2, т.е. количество возможных смысловых отношений между рассматриваемыми существительными.

Выход: m2 **–** цел – номер некоторой значащей строки массива arreln1n2. Параметр m2 приобретает ненулевое значение в результате обработки ответа пользователя на уточняющий вопрос ЛП. Пользователю предлагается указать, какое из нескольких

смысловых отношений реализуется в сочетании “Существительное 1 в позиции posleftnoun + зависимое Существительное 2 в позиции pos” с учетом предлога prep . Для этого пользователю с помощью столбца example даются примеры сочетаний, в которых реализуется такое же смысловое отношение, как и потенциально возможное отношение между единицами текста в позициях posleftnoun и pos.

Спецификация алгоритма “Выбор-тематич-роли”

Внешняя спецификация этого алгоритма приведена в подпараграфе 8.7.5.

Алгоритм “Обработка-сущ”

Нач если leftnumber > 0 то Matr[pos, qt] := leftnumber кесли

Если nattr > 0 то цикл для m от 1 до nattr

p1 := Attributes[m, place]; Matr[p1, posdir, 1] := pos;

Semprop:=Attributes[m, prop]; Matr[p1,.reldir, 1] := semprop кесли

leftnumber := 0 ; nattr := 0; Matr[pos, prep] := leftprep; leftprep := nil

Linenoun := Matr[pos, locunit] {номер набора из Arls, содержащего начальное значение существительного}

Sort1 := Arls[linenoun, st1]

Если Sort1 ≠ сит {ситуация}

То numbent := numbent + 1 {количество сущностей, упомянутых в просмотренной части текста} кесли

gramnumber := Число (Rc[pos, mcoord])

если gramnumber = 1 то Var1 := Varstring (‘x’, numbent) кесли

если gramnumber - число 2 или 3 то Var1 := Varstring (‘S’, numbent) кесли

Matr[pos, mark] := var1

Найти-сущ-слева (pos, posleftnoun)

Если posleftnoun = 0 {слева от позиции pos нет существительных, которые, возможно, управляют существительным в позиции pos}

То если verbposmag [depth] = 0

То numb-free-dep[depth] := numb-free-dep[depth] + 1

K1 := numb-free-dep[depth]; pos-free-dep[depth, k1] := pos

иначе posvb := verbposmag [depth]

Смысл-связь-глаг-формы (posvb, pos, Matr)

Иначе {в случае posleftnoun >0}

Найти-множ-отношений-сущ1-сущ2 (posleftnoun, leftprep, pos, Matr, nreln1n2, arreln1n2) {находятся возможные смысловые связи (и их количество) между рассматриваемым существительным в позиции pos и ближайшим слева существительным в позиции postleftnoun}

если (nreln1n2 = 0) {нет семантико-синтаксического управления от предыдущего существительного}

то posvb := verbposmag [depth

если posvb > 0

то Смысл-связь-глаг-формы (posvb, pos, Matr)

иначе { в случае posvb = 0}

numb-free-dep[depth] := numb-free-dep[depth] + 1

K1 := numb-free-dep[depth] ; pos-free-dep[depth, k1] := pos

кесли

кесли {случай nreln1n2 = 0 рассмотрен }

если (nreln1n2 > 0) {существует возможность семантико-синтаксического управления от предыдущего существительного}

то posvb := verbposmag [depth]

если posvb > 0

то Найти-множ-тематических-ролей (posvb, pos, class1, subclass1, Matr, nrelvbdep, arrelvbdep)

если (nrelvbdep = 0) {нет смысловой связи с глагольной формой}

то если (nreln1n2 = 1)

то m2 := 1 {m2 — номер эл-та массива arreln1n2, откуда берется инф-ция для Matr о связи между posn1 и posn2}

иначе Выбор-отнош-между-сущ (posn1, prep, posn2, nreln1n2, arreln1n2, m2) кесли

Добавление в Matr информации о связи между единицами текста в позициях posn1 и posn2, эта информация берется из позиции m2 массива arreln1n2;

кесли {случай nrelvbn2 = 0}

если (nrelvbdep > 0) {возможна связь с глаголом}

то если (nreln1n2>0) {возможна связь и с предыдущим существительным}

то Выбор-управления-глаг-сущ (posvb, prep, posn1, n2, res)

{res=1 ⇒ связь с глаголом; res=2 ⇒ связь с сущ. в позиции posn1}

если (res=1)

то если (nrelvbdep = 1) то m1:=1

иначе Выбор-тематич-роли (posvb, prep, posn2, nrelvbdep, arrelvbdep, m1)

{запись в Matr информации о связи между глагольной формой в позиции posvb и существительным в позиции pos, которая берется из строки m1 массива arrelvbdep}

nmasters[pos] := nmasters[pos] + 1;

{найдена новая управляющая стрелка, ведущая в позицию pos}

d := nmasters[pos]; Matr[pos, posdir, d] := posvb;

Matr[pos, reldir, d] := arrelvbdep [m1, role] ;

Matr[posvb, locunit] := arrelvbdep [m1, linevb]; Matr[posvb, nval] := 1;

Matr[pos, locunit] := arrelvbdep [m1, linenoun]; Matr[pos, nval] := 1

кесли кесли

если (res=2) {нет связи с глагольной формой, но есть связь с существительным. в позиции posleftnoun} то если (nreln1n2=1) то m2:=1

иначе

Выбор-отнош-между-сущ (posleftnoun, prep, pos, nreln1n2, arreln1n2, m2)

Кесли

{запись в Matr информации о смысловой связи между существительными в позициях posleftnoun и pos с учетом предлога prep (возможно, prep – это пустой предлог nil), которая берется из строки m2 массива arrelvbdep}

Matr[posleftnoun, locunit] := arreln1n2 [m2, locn1]; Matr[posleftnoun, nval] := 1;

Matr[pos, locunit] := arreln1n2 [m2, locn2]; Matr[pos, nval] := 1;

Matr[pos, posdir, 1] := posleftnoun;

Matr[pos, reldir, 1] := arreln1n2 [m2, role] Кесли

Если Rc[pos + 1, subclass] = сущ-собств

То logname := (слова в позициях pos и pos + 1 могут быть связаны с одним и тем же грамматическим падежом) И (семантические единицы, соответствующие этим словам в массиве Arls, имеют один и тот же набор сортов в Arls)

Если logname = Истина То Обработка-сущ-собств (pos) кесли кесли

Если Rc[pos + 1, subclass] = имя то Обработка-названий (pos) кесли

Leftprep := nil; leftnumber := 0; nattr := 0; обнулить столбец place массива Attributes; обнуляется, заполнить цепочкой nil – обозначением пустой строки - столбец prop массива Attributes.

Конец {алгоритма “Обработка-сущ”}

Описания вспомогательных алгоритмов

Описание алгоритма “Найти-сущ-слева”

Внешняя спецификация (см. выше)

Алгоритм

Нач posleftnoun := 0; p1 := pos

Цикл-до p1 := p1 – 1; classleft := Rc[p1, tclass]

Если classleft = сущ то posleftnoun := p1 кесли

Выход-при (p1 = 1) ИЛИ (posleftnoun > 0)

ИЛИ nlassleft ∈ {глаг, прич, наречие, местоим, констр, маркер}

кцикл кон

Пример. Пусть В1 = “Сколько контейнеров с индийской керамикой поступило из Новороссийска?”. Преобразуем вопрос В1 в следующее размеченное представление: “Сколько (1) контейнеров (2) с (3) индийской (4) керамикой (5) поступило (6) из (7) Новороссийска (8) ? (9)”. Пусть pos = 5 (позиция словоформы “керамикой”). Тогда после завершения работы алгоритма posleftnoun = 2 (позиция словоформы “контейнеров”).

Описание алгоритма «Обработка-сущ-собств»

Внешняя спецификация (см. выше)

Алгоритм

Нач k1 := pos + 1

Пока Rc[k1, tclass]= сущ-собств цикл

m1:= Matr [k1, locunit] {Найдена первая и единственная строка массива Arls с информацией о единице Rc[k1, unit]}

Matr[k1, posdir, 1]:= pos {Проведена управляющая стрелка от элемента в позиции pos к элементу в позиции k1}

sem1:=Arls[m1, sem]; Matr[k1, reldir, 1] :=sem1; k1:=k1+1 кцикл

pos : = k1 – 1 кон

Пример. Пусть В2 = “Сколько статей профессор Игорь Петрович Сомов опубли-ковал в 2003-м году?”. Тогда в результате вызова алгоритма Обработка-сущ-собств с параметром pos = 3 (позиция слова ”профессор”) будут как бы проведены управ-ляющие стрелки (посредством преобразования МССП Matr) от позиции pos к позициям pos + 1, pos + 2, pos + 3, соответствующим фрагменту “Игорь Петрович Сомов”.

8.9. Завершение разработки алгоритма построения матричного семантико-синтаксического представления входного текста

8.9.1. Описание головного модуля алгоритма

Для облегчения понимания головного модуля алгоритма построения МССП входного текста ниже приводится его внешняя спецификация (разработанная в параграфе 8.3).

Внешняя спецификация алгоритма SemSyn

Входные данные:

Lingb – лингвистический базис (л.б.);

T – текст из языка *Linp(G, Lingb)* , где - *G* – бесконтекстная грамматика вида (8.2.1).

Выходные данные:

nt – целое, количество единиц текста; **Rc** – классифицирующее представление входного текста T (см. параграф 7.1);

Rm – морфологическое представление входного текста (см. параграф 7.1);

Arls – множество упорядоченных наборов – проекция лексико-семантического словаря (ЛСС) Lsdic на входной текст T;

Arvfr – множество упорядоченных наборов – проекция словаря глагольно-предложных фреймов Vfr на входной текст T;

Arfrp – множество упорядоченных наборов - проекция словаря предложных семантико-синтаксических фреймов **Frp** на входной текст **T**;

Matr - матричное семантико-синтаксическое представление (МССП) входного текста (см. параграф 7.2).

numbqswd – переменная, отображающая количество вопросительных слов в предложении; одномерные массивы **posvbmag, numb-free-dep, posconnectword, nmasters**, двумерный массив **pos-free-dep** (структура и принципы использования этих массивов описаны в параграфе 5.6).

8.9.2. Внешние спецификации вспомогательных алгоритмов

Спецификация алгоритма “Построение-компон-морфол-представления”

Вход: **Lingb** – лингвистический базис; **T** – текст из Linp(G, Lingb), где G – бесконтекстная грамматика вида (8.2.1).

Выход: **Rc** - классифицирующее представление текста T; **nt** – цел – количество единиц текста в классифицирующем представлении Rc, т.е. количество значащих строк в Rc; **Rm** – морфологическое представление текста T**.**

Спецификация алгоритма “Построение-проекции-лексико-семантич-словаря”

Вход: **Rc**, **nt, Rm; Lsdic** - лексико-семантический словарь (см. параграф 4.4).

Выход: **Arls –** двумерный массив – проекция словаря Lsdic на входной текст T.

Спецификация алгоритма "Построение-проекции-словаря-глаг-фреймов"

Вход: **Rc**, **nt, Rm, Arls; Vfr** – словарь глагольно-предложных семантико-синтаксических фреймов (см. параграф 6.5).

Выход: **Arvfr –** двумерный массив – проекция словаря глагольно-предложных фреймов Vfr на входной текст T.

Спецификация алгоритма "Построение- проекции-словаря-предложных-фреймов"

Вход: **Rc**, **nt, Rm, Arls; Frp** – словарь предложных семантико-синтаксических фреймов (см. параграф 4.7).

Выход: **Arfrp –** двумерный массив – проекция словаря предложных фреймов Frp на входной текст T.

8.9.3. Алгоритм построения МССП входного текста

Алгоритм BuildMatr

Нач Построение-компон-морфол-представления (T, Rc, nt, Rm)
Построение-проекции-лексико-семантич-словаря (Rc, nt, Rm, Lsdic, Arls)
Построение-проекции-словаря-глаг-фреймов (Rc, nt, Rm, Arls, Vfr, Arvfr)
Построение-проекции-словаря-предложных-фреймов
(Rc, nt, Rm, Arls, Frp, Arfp)
Формирование-начальных-значений-данных
Выявление-вида-текста (nt, Rc, Rm, leftprep, mainpos, kindtext, pos)
Цикл-до pos := pos + 1
Class := Rc[pos, tclass]
выбор class **из**
предлог: Обработка-предлога (Rc, pos, leftprep);
прилаг: Обработка-прилаг (Rc, pos, nattr, Attributes)
колич-числит: Обработка-колич-числит (Rc, pos, numb);
сущ: Обработка-сущ (Rc, Rm, pos, Arls, Arfrp, Matr, leftprep, numb, nattr, Attributes)

местом: Обработка-местоим (Rc, Rm, pos, Arls, Rqs, Arfrp, Matr, leftprep)

наречие: Обработка- наречия (Rc, Rm, pos, Arls, Rqs, Matr)

глаг, прич: Обработка-глаг-формы (Rc, Rm, pos, Arls, Rqs, Arvfr, Matr, leftprep)

союз: Пустой оператор

констр: Обработка-конструкта

имя: Обработка-названий

маркер: если Rc[pos, unit] = ',' {запятая }

то Обработка-запятой (Rc, Rm, pos, Arls, Matr) кесли

квыбор

выход-при (pos = nt)

кон

Таким образом, в этом и предыдущих параграфах данной главы разработан алгоритм BuildMatr, находящий: (а) смысловые отношения между единицами ЕЯ-текста, (б) конкретные значения глагольных форм и существительных из текста. Эта информация отражена в строково-числовой матрице Matr.

Обрабатываемые алгоритмом тексты могут выражать сообщения (факты), вопросы и команды и могут включать глаголы (в неопределенной форме, изъявительном и повелительном наклонениях), причастия, существительные, прилагательные, числовые значения параметров (конструкты), количественные числительные и цифровые представления чисел, вопросительные слова (являющиеся вопросительно-относительными местоимениями и местоименными наречиями), союзные слова, являющиеся вопросительно-относительными местоимениями с лексемой “какой”. Входные тексты могут включать придаточные определительные предложения, составные описания множеств.

Построенный алгоритм BuldMatr является оригинальным и обладает рядом преимуществ по сравнению с известными подходами к алгоритмизации поиска смысловых отношений в ЕЯ-текстах. Эти преимущества и особенности алгоритма обсуждаются в заключительной части главы 9. Следует отметить, что алгоритм BuldMatr позволяет реализовать семантико-синтаксический анализ текстов из представляющих практический интерес подъязыков естественного (русского) языка.

Глава 9

АЛГОРИТМ СБОРКИ СЕМАНТИЧЕСКОГО ПРЕДСТАВЛЕНИЯ ТЕКСТА ПО ЕГО МАТРИЧНОМУ СЕМАНТИКО-СИНТАКСИЧЕСКОМУ ПРЕДСТАВЛЕНИЮ

9.1. Начальный шаг построения семантических представлений входных текстов

Алгоритм, преобразующий матричное семантико-синтаксическое представление (МССП) *Matr* в некоторое формальное выражение *Semrepr* $\in$ *Ls(B*) , где *B* – концептуальный базис, являющийся первым компонентом используемого размеченного концептуального базиса (р.к.б.) *Cb*, *Ls(B)* – СК-язык в базисе *B*, в параграфе 7.3 был назван *алгоритмом семантической сборки.*

Рассмотрим алгоритм “Подготовка-к-постр-СемП”, являющийся начальной частью разрабатываемого в данной главе алгоритма семантической сборки. Алгоритм “Начало-постр-СемП” строит начальное значение семантического представления (СП) входного текста, являющееся начальным значением строки *Semrepr* (“Semantic representation”) и зависящее от вида входного текста, т.е. от значения переменной kindtext, формируемого алгоритмом BuildMatr.

Выбор формы семантического представления входного текста в зависимости от значения переменной kindtext осуществляется на основе анализа, проведенного в параграфе 7.3. Некоторые примеры из этого параграфа используются ниже в алгоритме в качестве комментариев, показывающих контекст построения начального значения переменной. Для упрощения формы СП входного текста кванторы существования (когда они должны быть в соответствии с подходом, изложенным в главе 4) явно не указываются, а только подразумеваются.

Описание алгоритма “ Подготовка-к-постр-СемП”

Внешняя спецификация

Вход : Rc – массив – классифицируюшее представление входного текста; Rm – массив – морфологическое представление входного текста; kindtext – строка, характеризующая вид входного текста (возможными значениями этой строки

являются Stat, Imp, Genqs, Specqs-relat, Specqs-rol, Specqs-quant1, Specqs-quant2 (см. параграф 8.4); mainpos – целое число – позиция вопросительного слова в начале текста; Matr – МССП текста.

Выход : Semrepr – строка - начальное значение семантического представления входного текста.

Алгоритм “Подготовка-к-постр-СемП”

Нач Выбор kindtext из

Stat: Semrepr := пустая строка ;

{Пример. Пусть Т1 = “Профессор Игорь Новиков преподает в Томске”.

Тогда сначала Semrepr := пустая строка.

После завершения работы алгоритма BuildSem

Semrepr = Ситуация(e1, преподавание * (Время, #сейчас#)(Агент1, нек чел * (Квалиф, профессор)(Имя, ‘Игорь‘)(Фамилия, 'Новиков’) : x2)(Место1, нек город * (Название, 'Томск’) : x3)). }

Imp: Semrepr = *(Команда(#Оператор#, #Исполнитель#, #сейчас#, e1)*

{Пример. Пусть Т2 = “"Доставь ящик с деталями на склад № 3.".

Тогда сначала Semrepr := *(Команда(#Оператор#, #Исполнитель#, #сейчас#, e1)*

.

После завершения работы алгоритма BuildSem

Semrepr = *(Команда(#Оператор#, #Исполнитель#, #Сейчас#, e1) ∧ Цель (e1, доставка1*(Объект1, нек ящик * (Содерж1, нек множ * (Кач-состав, деталь)) : x1)(Место2, нек склад * (Номер, 3) : x2)))* }

Genqs: Semrepr := *Вопрос(x1≡Ист-знач (*

{Пример. Пусть Т3 = “Проходила ли в Азии международная научная конференция “COLING”?”. Тогда сначала

Semrepr := *Вопрос(x1, (x1≡Ист-знач (* .

После завершения работы алгоритма BuildSem

Semrepr = *Вопрос(x1, (x1≡ Ист-знач (Ситуация (e1, прохождение2* (Время, нек мом * (Раньше ,#сейчас#) : t1)(Событие, нет конф* (Вид1, междун) (Вид2,*

научная) (Название, ‘COLING’) : x2) (Место, нек континент (Название, ‘Азия’) :x3))))).* }

Specqs-relat1, Specqs-relat2:

начало k1 : = R1 [mainpos, mcoord];

numb := Число (R2 [k1, morph]) {Значением переменной numb является код грамматического числа, соответствующего вопросительному слову с лексемой “какой” из начального сегмента входного вопроса; 1 - код единственного числа, 2 - код множественного числа }

если kindtext = Specqs-relat1 то Semrepr := ‘*Вопрос(x1,’

иначе Semrepr := ‘*Вопрос (S1, (Кач-состав (S1,'* конец

{Пример 1. Пусть T4 = “Какое издательство опубликовало роман «Ветры Африки»?”. Тогда сначала Semrepr := ‘*Вопрос(x1,’* . После завершения работы алгоритма BuildSem *Semrepr = Вопрос(x1, Ситуация(e1, опубликование * (Время, нек мом * (Раньше, #сейчас#) : t1) (Агент2, нек издательство: x1) (Объект3, нек роман1* (Название, ‘Ветры Африки’) :x3))) . }*

{Пример 2. Пусть T5 = “ С какими зарубежными издательствами сотрудничает писатель Игорь Сомов?”. Тогда сначала Semrepr := *Вопрос (S1, (Кач-состав (S1, .*

После завершения работы алгоритма BuildSem

Semrepr = *Вопрос (S1, (Кач-состав (S1, издательство * (Вид-географич, зарубежное)) ∧ Описание (произв издательство* (Элем, S1) : y1, Ситуация(e1, сотрудничество * (Время, #сейчас#)(Агент1, нек чел* (Профессия, писатель)(Имя, ‘Игорь’)(Фамилия, ‘Сомов’): x1)(Организация1, y1))))) .* }

Specqs-rol: Semrepr := '*Вопрос (‘*

{Пример 1. Пусть T6 = “Кем выпускается препарат “Зиннат”?”.

Тогда сначала Semrepr := *Вопрос (.*

После завершения работы алгоритма BuildSem

Semrepr = *Вопрос (x1, Ситуация (e1, выпуск1 * (Время, #сейчас#) (Агент1, x1)(Продукция1, нек препарат1 * (Название, ‘Зиннат’) : x2)))*

Пример 2. Пусть T7 = “Откуда и для кого поступил трехтонный алюминиевый контейнер?”. Тогда сначала Semrepr := *Вопрос (.*

После завершения работы алгоритма BuildSem

Semrepr = *Вопрос ((x1 ∧ x2) , Ситуация (e1, поступление2 * (Время, нек мом * (Раньше, #сейчас#) : t1) (Место1, x1) (Адресат, x2) (Объект1, нек контейнер * (Вес, 3/тонна)(Материал, алюминий) : x3))) .* }

Specqs-quant1: Semrepr := '*Вопрос(x1, ((x1≡ Колич(’ ;*

{Пример. Пусть T8 = “Сколько человек участвовало в создании статистического сборника?”. Тогда сначала Semrepr := 'Вопрос(x1, ((x1≡ Колич(’ .

После завершения работы алгоритма BuildSem

Semrepr = Вопрос(x1, ((x1≡ Колич(S1)) ∧ Кач-состав (S1, чел) ∧ Описание(произв чел (Элемент, S1) : y1, Ситуация(e1, участие1* (Время, нек мом * (Раньше, #сейчас#) : t1) (Агент1, y1)(Вид-деятельности, создание1* (Продукт1, нек сборник1 * (Область1, статистика) : x2))))) .*

Specqs-quant2:

sortsit := выделенный сорт *сит (* “ситуация“) используемого концептуального базиса;

Semrepr := *Вопрос(x1, ((x1≡ Колич(S1)) ∧ Кач-состав (S1,+ sortsit + ') ∧ Описание(произв’ + sortsit + ‘* (Элемент, S1) : e1,‘*

{Пример. Пусть T9 = “Сколько раз Иван Михайлович Семёнов летал в Мексику?”.

Тогда сначала Semrepr := *Вопрос(x1, ((x1≡ Колич(S1)) ∧ Кач-состав (S1, сит) ∧ Описание(произв сит * (Элемент, S1) : e1, .*

После завершения работы алгоритма BuildSem

Semrepr = *Вопрос(x1, ((x1≡ Колич(S1)) ∧ Кач-состав (S1, сит) ∧ Описание(произв сит * (Элемент, S1) : e1, Ситуация (e1, полёт * (Время, нек мом * (Раньше, #сейчас#) : t1)(Агент1, нек чел* (Имя, ‘Иван’)(Отчество, ‘Михайлович’)(Фамилия, ‘Семёнов’): x2)(Место2, нек страна* (Название, ‘Мексика’):x3)))))’ .* } квыбор кон

9.2. Построение семантических представлений коротких фрагментов входного текста с помощью алгоритма “Начало-постр-СемП”

9.2.1. Основные используемые структуры данных

В параграфе 7.3 были рассмотрены главные структуры данных, позволяющие по матричному семантико-синтаксическому представлению (МССП) входного текста построить его семантическое представление (СП), являющееся К-представлением, т.е. выражением стандартного К-языка в используемом концептуальном базисе. Такими структурами являются одномерные массивы Sembase (“Семантическая основа”), Semdes (“Семантическое описание”), Performers (“Исполнители ролей в ситуациях, упоминаемых во входном тексте ”) и двумерный массив Sitdescr (“Описание ситуаций”).

В данном параграфе разрабатывается алгоритм “Начало-постр-СемП”, предназначенный для формирования массивов Sembase, Semdes, Performers и начальной конфигурации массива Sitdescr.

9.2.2. Вспомогательные алгоритмы

Расмотрим алгоритмы, взаимодействие которых позволяет сформировать массивы Sembase, Semdes, Performers и начальную конфигурацию массива Sitdescr.

Описание алгоритма “Вычисление-вида-случая“

Внешняя спецификация

Вход : Rc – массив – классифицируюшее представление входного текста; k1 – номер строки классифицируюшего представления входного текста, т.е. порядковый номер единицы текста; Arls – двумерный массив – проекция лексико-семантического словаря (ЛСС) Lsdic на входной текст T; Matr – МССП текста; class1 – строка, задающая класс единицы текста; sem1 – семантическая единица, соответствующая k1–й единице текста.

Выход : casemark – строка, принимающая значения case1 – case7 в зависимости от вида обрабатываемого фрагмента классифицируюшего представления текста.

Алгоритм

Нач если class1 = прилаг то casemark := ‘Case1’ кесли

если class1 = констр то casemark := ‘Case2’ кесли

если class1 = сущ

то если Rc[k1 + 1, tclass] = имя то casemark := ‘Case3’

иначе numb1 := Matr[k1, qt]

{число, относящееся к существительному в позиции k1}

ref := нек **{**квантор референтности}

beg1 := sem1[1] {первый символ цепочки sem1, если считать каждый элемент первичного информационного универсума X(B(Cb(Lingb))) и каждую переменную из V(B) одним символом}

setind1 := 0 {признак обозначения индивида, а не множества индивидов}

len1 := Длина (sem1)

если (len1 ≥ 2) И (sem1[2] = ' множ ')

то setind1 := 1 кесли

{ sem1[2] – 2-й символ структурированной семантической единицы sem1 , если интерпретировать как символы элементы первичного информационного универсума *X(B(Cb))*, где *Cb* – используемый размеченный концептуальный базис (р.к.б.)}

если ((numb1 = 0) ИЛИ (numb1 = 1)) И

(beg1 = ref) И (setind1 = 0)

{т.е. RC[k1, unit] – обозначение индивида, а не множества }

то casemark := ‘Case4’ {Пример: ‘Бельгия’} кесли

если (numb1 = 0) И (beg1 ≠ ref) И (sem1 не является обозначением функции из F(B(Cb)), где Cb – используемый размеченный концептуальный базис) то нач loc1 := Rc[k1, mcoord], md1 := Rm[loc1, morph];

если (Число(md1) = 1) то casemark := ‘Case5’ {Пример: ‘конференция’}

иначе { т.е. в случае Число(md1) = 2) casemark := ‘Case6’

{Примеры: ‘5 статей’ , ‘3 международные конференции’} кесли кон кесли

если (numb1 = 0) И (setind1 = 1) {Пример: ‘с индийской керамикой’}

то casemark := ‘Case7’ кесли

конец

Описание алгоритмов Buildsemdes1 – Buildsemdes7

Внешняя спецификация каждого из алгоритмов Buildsemdes1 – Buildsemdes7

Вход : Rc – массив – классифицирующее представление входного текста; k1 – цел – номер строки из Rc; Arls –двумерный массив – проекция лексико-семантического словаря (л.с.с.) Lsdic на входной текст T; Matr – МССП текста; sem1 – строка – семантическая единица, соответствующая единице текста с номером k1; casemark – строка, принимающая значения case1 – case7 в зависимости от вида обработанного фрагмента классифицирующего представления входного текста; массивы Sembase, Semdes, Performers.

Выход: массивы Sembase, Semdes, Performers (эти массивы были описаны в параграфе 7.3), хранящие блоки для образования финального значения переменной Semrepr – семантического представления входного текста.

Описание алгоритма Buildsemdes1

Описание вспомогательных алгоритмов

Функция Transform1

Аргументы: s – строка вида r(z, b), где r – обозначение бинарного отношения, b – второй атрибут отношения, или вида (f(z) ≡ b) , где f - имя одноместной функции, b – строка, обозначающая значение функции, z – буква ‘z', интерпретируемая как переменная.

Значение: строка t вида (r, b) в первом случае и вида (f, b) во втором случае.

Пример. Пусть T1 = “Сколько двухтонных алюминиевых контейнеров поступило из Пензы?”. Тогда лингвистический базис может быть определен так, что для k1 = 2 sem1 := (Вес (z) ≡ 2/тонна) , Transform1(sem1)= (Вес ,

2/тонна) , для k1 = 3 sem1 := Материал (z , алюминий) , Transform1(sem1)= (Материал, алюминий).

Алгоритм Buildsemdes1

Нач {Отображение семантики прилагательных в массиве sembase}

Если Matr[k1 – 1, nattr] = 0

{непосредственно слева от позиции k1 нет прилагательных, т.е. в позиции k1 расположено первое прилагательное из группы идущих подряд прилагательных}

то Sembase[k1] := Transform1(sem1)

иначе {непосредственно слева от позиции k1 есть прилагательное}

Sembase[k1] := Sembase[k1 - 1] + Transform1(sem1)

{здесь знак + обозначает операцию конкатенации, т.е. операцию приписывания строки справа} кесли кон

Пример. В процессе использования алгоритма Buildsemdes1 для обработки вопроса Т1 = “Сколько двухтонных алюминиевых контейнеров поступило из Пензы?” будут выполнены операторы Sembase[2] := (Вес , 2/тонна) , Sembase[3] := (Вес , 2/тонна) (Материал, алюминий) .

Алгоритм Buildsemdes2

Нач {Обработка конструкта }

Sembase[k1] := sem1; Performers[k1] := Rc[k1, unit]

{Пример. Performers[k1] := ‘720/км’ } кон

Описание алгоритма Buildsemdes3 (“Обработка названий”)

Назначение: построение семантического представления (СП) фрагмента текста Т, являющегося сочетанием вида “Существительное + Выражение в кавычках или апострофах ”.

Условие вызова: в позиции k1 расположено существительное, в позиции k1 + 1 расположено выражение в кавычках или апострофах.

Пример. Пусть Т2 = «Кем выпускается препарат “Зиннат”?. Тогда в результате применения этого алгоритма будет выполнено присваивание

Performers[k1] := *нек препарат1 * (Назаниев, ‘Зиннат’)* .

Алгоритм

Нач name := RcT[k1 + 1, unit] ;

Если (Performers [k1] не включает символ *)

То Performers[k1] := Performers [k1] + ‘* (Название,’ + name + ‘)’

Иначе Performers[k1] := Performers [k1] + ‘ (Название,’ + name + ‘)’ кон

Алгоритм Buildsemdes4

Нач {Обработка существительных собственных}

{Пример контекста – опубликовал в Бельгии }

Sembase[k1] := sem1; Semdes[k1] := Sembase[k1]

Var1 := Matr[k1, mark] ; Performers[k1] := Semdes[k1] + ‘ : ‘ + var1

{Пример. Performers[k1] := ‘нек страна * (Название, ‘Бельгия) : x2’ } Кон

Алгоритм Buildsemdes5

Нач {Обработка нарицательных существительных }

{Пример контекста – опубликовал монографию}

если Matr[k1, nattr] ≥ 1 {слева есть прилагательные}

то Sembase[k1] := sem1 + ‘*’ + sembase[k1 – 1]

иначе Sembase[k1] := sem1 кесли

Ref := ‘нек‘ ; Semdes[k1] := ref sem1

Var1 := Matr[k1, mark] ; Performers[k1] := Semdes[k1] + ‘ : ‘ + var1

{Пример 1. Performers[k1] := ‘нек монография : x3’ }

{Пример 2. Performers[k1] := ‘нек принтер * (Вид, струйный) : x4’ } кон

Алгоритм Buildsemdes6

Нач {Обработка сочетаний с существительными, обозначающих множества объектов. Пример контекста – “Поступили 5 трехтонных контейнеров”}

numb1 := Matr[k1, qt] ; Sembase[k1] := sem1

Если numb1 > 0 то Semdes[k1] := *‘нек множ * (Колич,’ + numb1 + ‘)(Кач-состав,’ + sembase[k1] + ‘)’*

иначе Semdes[k1] := *‘нек множ * (Кач-состав,’ + sembase[k1] + ‘)’*
кесли

beg1 := sem1[1] {первый символ цепочки sem1, если считать символами элементы первичного информационного универсума X(B(Cb(Lingb))) и переменные}

Var1 := Matr[k1, mark] ; Var2 := Varsetmember(var1);

{Переменная var2 обозначает произвольный элемент множества с меткой var1. Пример. Если var1 = S2, то var2 = y2 }

Performers[k1] := ‘произвольн’ + beg1 + ‘* Элем(' + Semdes[k1] +‘ : ‘ + var1 + ‘): ‘ + var2 { Пример. Performers[k1] := ‘произвольн контейнер1 * (Элем, нек множ * (Колич, 5)(Кач-состав, контейнер1 * (Вес, 3/тонна)): S1) : y1’} кон

Описание алгоритма Buildsemdes7 (“Обработка собирательных существительных”)

Назначение: Построение фрагмента семантического представления (СП) текста T, являющегося сочетанием, включающим собирательное существительное (“индийская керамика”, “итальянская обувь” и т.п.).

Условие вызова: в позиции k1 расположено собирательное существительное.

Пример. Пусть ТЗ = “Откуда поступили три контейнера с индийской керамикой? “. Словоформа “керамикой“ в вопросе ТЗ имеет порядковый номер 7. Лингвистический базис может быть определен так, что в результате применения алгоритма Buildsemdes7 будут выполнены операторы

Semdes[7] := *нек множ * (Кач-состав, керамич-изделие * (Географич-локализация, нек страна * (Назв, ‘Индия‘))) ,*

Performers [7] := *нек множ * (Кач-состав, керамич-изделие * (Географич-локализация, нек страна * (Назв, ‘Индия‘))) : S1 .*

Описание вспомогательных алгоритмов

Функция Transform2

Аргументы: s – строка, отображающая семантику прилагательного или последовательности прилагательных; например, s может отображать семантику прилагательного “индийская “ и являться строкой *(Географич-локализация, нек*

*страна * (Назв, ‘Индия‘))* ; t – строка, являющаяся структурированной семантической единицей, соответствующей собирательному существительному и включающая подцепочку *(Кач-состав*, (например, t может соответствовать существительному “керамика“ и являться строкой *нек множ * (Кач-состав, керамич-изделие)*).

Значение: строка u , формируемая следующим образом. Пусть pos1 – позиция первой левой скобки (в подстроке *(Кач-состав*, строки s, и пусть pos2 – позиция правой скобки) , закрывающей скобку в позиции pos1. Пусть h – подстрока строки t, лежащая между подстрокой *(Кач-состав*, и правой скобкой в позиции pos2. Тогда u получается из строки t заменой подстроки h на строку h * s .

Пример. В контексте вопроса ТЗ = “Откуда поступили три контейнера с индийской керамикой? “ пусть *s = (Географич-локализация, нек страна * (Назв, ‘Индия‘)) , t = нек множ * (Кач-состав, керамич-изделие)* .
Тогда *h = керамич-изделие, u = Transform2 (s, t) = нек множ * (Кач-состав, керамич-изделие * (*Географич-локализация, *нек страна * (Назание, ‘Индия‘)))* .

Алгоритм Buildsemdes7

Нач если Rc[k1 – 1, tclass] ≠ прилаг то semdes[k1] := sem1

иначе prop1 := sembase [k1 – 1]; Semdes[k1] := Transfrom2 (prop1, sem1)

{Пример. Semdes[k1] := *нек множ * (Кач-состав, керамич-изделие * (Место-производства, нек страна * (Назв, ‘Индия‘)))* }

Описание алгоритма ProcessSit

Алгоритм ProcessSit предназначен для представления в массиве Sitdescr структурированных единиц концептуального уровня (другими словами, семантических единиц), соответствующих тем ситуациям, которые упоминаются во входном тексте с помощью глаголов или причастий.

Внешняя спецификация

Вход : Rc – массив – классифицируюшее представление входного текста T; k1 – номер строки классифицируюшего представления текста T, т.е. порядковый

номер единицы текста, являющейся глагольной формой; Rm – массив – морфологическое представление текста T; kindtext – строка – обозначение вида текста T; Arls – проекция лексико-семантического словаря Lsdic на входной текст T; Matr – МССП текста; Sitdescr – исходная конфигурация массива описания ситуаций, упоминаемых в тексте; timevarnumb – максимальный номер переменной, обознача-ющей момент времени.

Выход: Sitdescr - преобразованная конфигурация массива для описания упоминаемых в тексте ситуаций .

Описание вспомогательных алгоритмов

Функция Numb

Аргумент: v – строка вида *R S* , где *R* - буква латинского алфавита, *S* – строка, представляюшая натуральное число. **Значение:** *N* - натуральное число, ассоции-рованное со строкой *S* .

Пример. Для строки *e3 Numb(e3)* – это число 3.

Функция Stringvar

Аргументы: *R* - буква латинского алфавита, *N* - натуральное число.

Значение: строка вида *R S* , где *S* – строка, представляюшая натуральное число *N*.

Пример. Если *R* - буква ‘t’, *N* – число 2, то *Stringvar(R, N)* – это строка *t2*.

Функция Time

Аргументы: M- набор морфологических признаков, связанный с произвольной глагольной формой vbform (глаголом или причастием)

Значение: цифра ‘1’, если форме vbform соответствует прошедшее время; цифра ‘2’, если форме vbform соответствует настоящее время; цифра ‘3’, если форме vbform соответствует будущее время.

Алгоритм ProcessSit

Нач pos1 := Rc[k1, mcoord]

armorph := Rm[pos1, morph] {набор морфологических признаков, связанный с глагольной формой в позиции k1}

timevarnumb := timevarnumb + 1

Vartime := Stringvar (‘t’ , timevarnumb)

time1 := Time(armorph)

Выбор Time1 из

‘1’: timesit := ‘(Время, нек мом * (Раньше, #сейчас#) : ‘ + vartime + ‘)’

‘2’: timesit := ‘(Время, #сейчас#) ‘

‘3’: timesit := ‘(Время, нек мом * (Позже, #сейчас#) : ‘ + vartime + ‘)’

Квыбор

linesit := Matr[k1, locunit] ; concsit := Arls[linesit, sem]

var1 := Matr[k1, mark] ; numbsit := Numb (var1)

если (kindtext = Imp) И (numbsit = 1)

то Sitdescr [numbsit, expr] := ‘Цель (‘ + var1 + ‘,’ + concsit + ‘*’

иначе Sitdescr [numbsit, expr] := ‘Ситуация (‘ + var1 + ‘,’ + concsit + ‘*’ + timesit

кесли

{Пример 1. Sitdescr [1, expr] := *‘Ситуация (e1, выпуск1 *‘(Время, нек мом * (Раньше, #сейчас#) : t1)‘* }

{Пример 2. Sitdescr [1, expr] *:= 'Цель (e1, доставка1*(Объект1, нек контейнер : x1)(Место2, нек склад * (Номер, 4) : x2))'* } кон

Описание алгоритма “Начало-постр-СемП”

Внешняя спецификация

Вход : Rc – массив – классифицирующее представление входного текста T; Rm – массив – морфологическое представление текста T; Arls –проекция лексико-семантического словаря (ЛСС) Lsdic на текст T; kindtext – строка, характеризующая вид входного текста T; mainpos – целое число – позиция вопросительного слова в начале текста; Matr – МССП текста.

Выход : Semrepr – строка - начальное значение семантического представления входного текста; Performers – одномерный массив, содержащий семантические представления коротких фрагментов входного текста.

Алгоритм

Нач Подготовка-к-постр-СемП (Rc, Rm, Matr, kindtext, mainpos, Semrepr)

{Пример: Если kindtext = genqs (общий вопрос, т.е. вопрос с ответом “Да/Нет”), то Semrepr := ‘Вопрос (x1, (x1 ≡ Ист-знач (‘ }

цикл для k1 от 1 до nt

{формирование массивов Sembase, Semdes, Performers и начальной конфигурации массива Sitdescr}

class1 := Rc[k1, tclass]

если (class1 ≠ конструкт) И (class1 ≠ имя) И (class1 ≠ маркер)

то loc1 := Matr[k1, locunit] ; sem1 := Arls[loc1, sem] кесли

если (class1 = глаг) ИЛИ (class1 = прич)

то ProcessSit (k1, Rc, Rm, k1, Arls, Matr , Sitdescr, timevarnumb) кесли

если (class1 – элемент множества {прилаг, констр, сущ})

то Вычисление-вида-случая (Rc, k1, Arls, Matr, class1, sem1, casemark1);

Выбор casemark из

‘Case1’: Buildsemdes1 (List1),

где List1 – список параметров Rc, k1, Arls, Matr, Sembase, Semdes, Performers, casemark1;

‘Case2’: Buildsemdes2 (List1);

‘Case3’: Buildsemdes3 (List1);

‘Case4’: Buildsemdes4 (List1);

‘Case5’: Buildsemdes5 (List1);

‘Case6’: Buildsemdes6 (List1);

‘Case7’: Buildsemdes7 (List1)

квыбор

кон

9.3. Заключительные этапы разработки алгоритма сборки семантического представления входного текста по его матричному семантико-синтаксическому представлению

9.3.1. Основные идеи алгоритма “Отображение-ситуаций”

К моменту вызова алгоритма сформированы массив Performers и начальная конфигурация массива описания ситуаций Sitdescr. В массиве Performers представлены семантические единицы (первичные и составные), соответствующие конструктам (числовым значениям параметров), существительным и сочетаниям видов “Группа прилагательных + Существительное”, “Число + Существительное”, “Число + Группа прилагательных + Существительное”, “Количественное числительное + Существительное”, “Количественное числительное + Группа прилагательных + Существительное”.

Напомним (см. параграф 7.3), что количество заполненных строк массива Sitdescr равно количеству глаголов и причастий в тексте. В столбце mrk размещается метка ситуации (связь с МССП Matr осуществляется через элементы в этом столбце); столбец expr (сокращение от “expression” – “выражение”) предназначен для хранения семантических описаний ситуаций (событий), упоминаемых в тексте (см. таблицу в подразделе 7.3.1).

В рассматриваемом алгоритме “Отображение-ситуаций” преобразование информации осуществляется в два последовательных этапа. Первый этап представляет собою цикл по m от 1 до nt, где m – номер строки классифицирующего представления Rc, nt – количество элементов текста. В этом цикле информация о семантико-синтаксических отношениях в сочетаниях “Глагольная форма (глагол или причастие) + Зависимый фрагмент предложения” отображается в элементах столбца expr массива Sitdescr (каждый

из таких элементов является описанием определенной ситуации, упоминаемой во входном тексте).

При этом под зависимым фрагментом предложения понимается конструкт, либо существительное, либо сочетание одного из видов “Группа прилагательных + Существительное”, “Число + Существительное”, “Число + Группа прилагательных + Существительное”, “Количественное числительное + Существительное”. “Количественное числительное + Группа прилагательных + Существительное”.

Примерами зависимых фрагментов предложения являются выражения “в 2002-м году”, “европейские научные издательства”, “двухтонных контейнеров”, “5 контейнеров”, “двенадцать персональных компьютеров”.

Пример. Пусть B1 = “Сколько двухтонных контейнеров с индийской керамикой, поступивших из Новороссийска, было отправлено фирме “Парус”?

Тогда на первом этапе выполнения алгоритма выражения, являющиеся элементами столбца expr массива Sitdescr, пополняются информацией о семантико-синтаксических связях в сочетаниях “поступивших + двухтонных контейнеров”, “поступивших + Новороссийска”, “отправлено + двухтонных контейнеров”, “отправлено + фирме “Парус”.

В результате выполнения данного этапа алгоритма для вопроса B1 массив Sitdescr приобретет следующую конфигурацию:

mrk	expr
e1	*Ситуация(e1, поступление2 * (Время, нек мом * (Раньше, #сейчас#) : t1) (Объект1, произв контейнер1 * (Элем, нек множ * (Кач-состав, контейнер1 * (Вес, 2/тонна) : S1) : y1)(Место1, нек город * (Название, 'Новороссийск‘) : x2))*
e 2	*Ситуация (e2, отправка1 * (Время, нек мом * (Раньше, #сейчас#) : t2) (Объект1, y1)(Адресат, нек фирма * (Название, 'Парус‘) : x3))*

Рис. 9.1. Структура массива Sitdescr на промежуточном этапе построения семантического представления входного текста

Вспомогательный массив Used длины nt позволяет избежать многократного повторения в столбце expr массива Sitdescr семантического представления одного и того же выражения, обозначающего объект или множество объектов. Первоначально для каждого m от 1 до nt Used [m] = 0. Если для некоторого k строка Performers [k] включается в состав некоторой строки массива Sitdescr, то Used [k] := 1. Поэтому в состав других строк массива Sitdescr (если такая необходимость возникает) включается не строка Performers [k] , а переменная, являющаяся окончанием строки Performers [k].

Например, в первую строку массива Sitdescr (рис. 9.2) входит выражение *произв контейнер1 * (Элем, нек множ * (Кач-состав, контейнер1 * (Вес, 2/тонна) : S1) : y1* , являющееся элементом Performers [3]. Во второй же строке массива Sitdescr вместо этого выражения использована переменная *y1*.

Фрагменты предложения, непосредственно управляемые глагольными формами (глаголами или причастиями), назовем зависимыми элементами 1-го уровня.

Второй этап алгоритма “Отображение-ситуаций” заключается в поиске по МССП Matr таких конструктов или сочетаний с существительным, которые непосредственно управляются зависимыми элементами 1-го уровня, т.е. в поиске зависимых элементов 2-го уровня. Например, в вопросе В1 фрагмент “двухтонных контейнеров” управляется глагольной формой “было отправлено**”** и поэтому является зависимым элементом 1-го уровня. В то же время сочетание “двухтонных контейнеров” управляет сочетанием “с индийской керамикой”.

Формальное представление *descr1* информации, передаваемой сочетанием вида “Зависимый элемент 1-го уровня X + Зависимый элемент 2-го уровня Y”, приписывается справа с помощью конъюнкции к элементу Sitdescr [k, expr], где k – порядковый номер ситуации, участника которой обозначает зависимыый элемент 1-го уровня X.

Вспомогательный массив Conj первоначально заполняется нулями. Если в результате выполнения второго этапа алгоритма к элементу Sitdescr [k, expr] приписывается справа с помощью конъюнкции (conjunction) некоторое

выражение, то Conj[k] := 1. Это значение 1 является сигналом об обрамлении элемента Sitdescr [k, expr] скобками (,) перед включением этого элемента в семантическое представление входного текста.

Пример. В результате выполнения второго этапа алгоритма для вопроса В1 к элементу Sitdescr [1, expr] с помощью конъюнкции будет приписано справа выражение *Содержание1 (y1, нек множ * (Кач-состав, керамич-изделие * (Географич-локализация, нек страна * (Назв,'Индия'))))* .

Так как использовалась конъюнкция, то элементу Conj [1] будет присвоено значение 1. В итоге массив Sitdescr приобретет следующую конфигурацию:

mrk	expr
e1	*Ситуация(e1, поступление2 * (Время, нек мом * (Раньше, #сейчас#) : t1) (Объект1, произв контейнер1 * (Элем, нек множ * (Кач-состав, контейнер * (Вес, 2/тонна) : S1) : y1)(Место1, нек город * (Назв, 'Новороссийск') : x2)) ∧ Содержание1 (y1, нек множ * (Кач-состав, керамич-изделие * (Географич-локализация, нек страна * (Название,'Индия')))*
e2	*Ситуация (e2, отправка1 * (Время, нек мом * (Раньше, #сейчас#) : t2) (Объект1, y1)(Адресат, нек фирма * (Название, 'Парус') : x3))*

Рис. 9.2. Структура массива Sitdescr на заключительном этапе построения семантического представления входного текста

9.3.2. Описание алгоритма “Отображение-ситуаций”

Рассматриваемый ниже алгоритм предназначен для отображения информации об упоминаемых в тексте ситуациях (событиях) в массиве Sitdescr .

Внешняя спецификация алгоритма

Вход : Rc – массив – классифицируюшее представление входного текста; nt - целое число – длина входного текста (количество строк в Rc и Matr); kindtext –

строка – обозначение вида входного текста; Matr – МССП текста; Performers – двумерный массив, содержащий семантические образы участников ситуаций; maxnumbsit – цел – количество ситуаций, упоминаемых в тексте.

Выход : Sitdescr - массив, отображающий информацию о ситуациях, упоминаемых во входном тексте; Used[1 : nt] – одномерный массив для хранения признаков неоднократности использования структурированной семантической единицы в строках массива Sitdescr; Conj[1 : maxnumbsit] - одномерный массив для хранения признаков использования конюнкции в строках массива Sitdescr.

Алгоритм “Отображение-ситуаций”

Нач {Обработка прямых смысловых связей между глагольной формой и существительным или конструктом}

Цикл для j1 от 1 до nt Used [j1] := 0 кцикл

Цикл для j2 от 1 до maxnumbsit Conj [j2] := 0 кцикл

Цикл для m от 1 до nt {первый проход строк из Rc}

Class1 := Rc [m, tclass]

Если (class1 = сущ) ИЛИ (class1 = констр)

То нач d := nmasters[m]; {найдено количество единиц текста, управляющих m–й единицей текста}

Если d >0

То цикл для q от 1 до d

Нач p1 := Matr [m, posdir, q]; class2 := Rc[p1, tclass]

Если (class2 = глаг) ИЛИ (class2 = прич)

То var2 := Matr [p1, mark]; {метка ситуации, которую обозначает управляющая глагольна форма} numbsit := Numb (var2);

role := Matr [m, reldir, q] кесли

если (class1 = сущ)

то если kindtext не входит в множество {specqs-relat2, specqs-quant1}

то если Used [m] = 0 то actant := Performers[m]

иначе если (class1 = сущ)

то actant := Varbuilt(Matr[m, mark]) кесли;

если (class1 = констр) то actant := Rc[m, unit]) кесли

иначе {в случае kindtext входит в множество {specqs-relat2, specqs-quant1}}

если ((Used[m] = 1) ИЛИ ((Used[m] = 0) И (Matr[m, mark] = S1))

то actant := Varbuilt(Matr[m, mark])

иначе actant := Performers[m] кесли

кесли {kindtext}

{Varbuilt (x_j) = x_j ; Varbuilt (S_j) = y_j }

Sitdescr [numbsit, expr] :=

Sitdescr [numbsit, expr] + '(' + role + ',' + actant + ')'

Кцикл {конец цикла по q}

Кцикл {конец цикла по m }

{Пример 1. Пусть рассматривается вопрос В1 = "Откуда поступили 5 алюминиевых двухтонных контейнеров", и перед применением алгоритма "Отображение-ситуаций" выполнялось соотношение

Sitdescr[1, expr] = *Ситуация (e1, поступление2 * (Время, нек мом * (Раньше, #сейчас#) : t1)* .

Тогда после применения алгоритма "Отображение-ситуаций" при определенном выборе размеченного концептуального базиса будет иметь место соотношение

Sitdescr [1, expr] = *Ситуация (e1, поступление2 * (Время, нек мом * (Раньше, #сейчас#) : t1) (Место1, x1)(Объект1, произв контейнер1 * (Элем, нек множ * (Колич, 5) (Кач-состав, контейнер1 * (Вес, 2/тонна)(Материал, алюминий)) : S1) : y1*}

цикл для k1 от 1 до nt

{заполнение Sitdescr – второй проход Rc– обработка единиц текста, управляемых существительными, зависящими от глагольной формы}

class1:=Rc[k1, tclass]

если (class1= сущ) ИЛИ (class1= констр)

{примеры сочетаний: "контейнеров с индийской керамикой" (class1= сущ), "с лампами по 60 ватт" (class1= констр) }

то posmaster: := Matr[k1, posdir, 1]

{позиция словоформы “контейнеров” или словоформы “ лампами“ для указанных выше сочетаний}

если posmaster = 0 то вывод (‘Неправ. текст’)

иначе class2 := Rc[posmaster, tclass]

если class2 = сущ

то rel1 := Matr [k1, reldir, 1]

varmaster := Matr[posmaster, mark]

если (class1= констр) то arg2 := sembase [k1] кесли

если (class1= сущ)

то vardep:=Matr[k1, mark]

если Used[k1] = 0 то arg2 := Performers[k1]

иначе arg2 := vardep кесли

letter := первый символ цепочки varmaster {переменная varmaster соответствует существительному, управляющему единицей текста в позиции k1, причем для самого этого существительного управляющей единицей является глагольная форма – глагол или причастие}

если letter=’x’ то descr1:=rel1(varmaster, arg2)

иначе {т.е. в случае letter=’S’}

semhead:= первый элемент sembase[posmaster]

descr1 := rel1 + ‘(произв‘ + semhead + ‘*(Элем,‘

+ varmaster + ‘),’ + arg2 + ‘)’

{Затем с помощью конъюнкции $\wedge$ к выражению Sitdescr [numbsit, expr] , характеризующему рассматриваемую ситуацию с номером. numbsit, справа приписывается выражение descr1 }

Пример: Пусть B1 = “Какие писатели из Томска участвовали в конференции?”, и после выполнения первого цикла (для m от1 до nt) массив Sitdescr имет следующую конфигурацию:

mrk	expr
e1	*Ситуация(e1, участие * (Время, нек мом * (Раньше, #сейчас#) : t1)(Агент1, произв писатель * (Элем, нек множ * (Кач-состав,писатель) :S1) : y1)* *(Событие1, нек конференция : x1))*

Рис. 9.3. Структура массива Sitdescr после выполнения цикла с параметром m, изменяющимся от1 до nt.

Во втором цикле (для k1 от1 до nt) выполняется оператор Descr1:= *Географич-локализация (y1, нек город * (Назв, ‘Томск ‘) : x2)* , а затем массив Sitdescr приобретает новую конфигурацию:

Sitdescr

mrk	expr
e1	*Ситуация(e1, участие * (Время, нек мом * (Раньше, #сейчас#) : t1)(Агент1,, произв писатель * (Элем, нек множ * (Кач-состав, писатель) : S1) : y1)(Событие1, нек конференция : x1)) ∧ Географич-локализация (y1, нек город * (Назв, ‘Томск ‘) : x2)* .

Рис. 9.4. Структура массива Sitdescr после выполнения цикла с параметром k1, изменяющимся от1 до nt.

Конец примера}

конец

possit := Matr[posmaster, posdir, 1]

varsit := Matr[possit, mark] ; numbsit := Numb(varsit)

Sitdescr [numbsit, expr] := Sitdescr [numbsit, expr] + ‘∧’ + descr1

Если Conj [numbsit] = 0 то Conj [numbsit] := 1 кесли

{признак использования конъюнкции в строке Sitdescr [numbsit, expr] }

кесли кесли кесли кцикл

конец {алгоритма }

9.3.3. Описание алгоритма “Заключит-операции“

Рассматриваемый алгоритм предназначен для передачи информации, отраженной в масиве описания ситуаций Sitdescr, в финальном значении строки Semrepr - семантического представления входного текста.

Внешняя спецификация

Вход : Rc – массив – классифицируюшее представление входного текста; Rm – массив – морфологическое представление входного текста; kindtext – строка, характеризующая вид входного текста; mainpos – целое число – позиция вопросительного слова в начале текста; Matr – МССП текста; Performers – двумерный массив, содержащий семантические образы участников ситуаций; numbsit – цел – количество ситуаций, упоминаемых во входном тексте, т.е. количество заполненных строк массива Sitdescr; numbqswd – цел – количество вопросительных слов во входном тексте; Sitdescr - массив, отображающий информацию о ситуациях, упоминаемых во входном тексте; Semrepr – строка - исходное значение семантического представления текста.

Выход : Semrepr – строка - финальное значение СП входного текста.

Описание вспомогательных алгоритмов

Функция Right

Аргументы: pos1 – номер строки из классифицирующего представления Rc входного текста, т.е. номер позиции единицы входного текста; class1 – строка – обозначение класса единицы текста. **Значение:** pos2 – позиция ближайшей справа (к позиции pos1) единицы текста, относящейся к классу class1.

Функция Stringvar

Аргументы: *R* - буква латинского алфавита, *N* - натуральное число.

Значение: строка вида *R S* , где *S* – строка, представляюшая натуральное число *N*.

Пример: Если *R* - буква ‘t’, *N* – число 2, то *Stringvar(R, N)* – это строка *t2*.

Алгоритм

Нач pos2 := pos1

цикл-до pos2 = pos2 + 1; class2 := Rc [pos2, tclass]

выход-при (class2 = class1) конец

Алгоритм “Заключит-операции“

Начало цикл для k от 1 до maxnumbsit

event := Sitdescr [k, expr]

если Conj[k] = 1 то event := '(' + event + ‘)’ кесли

если k = 1 то situations := event

иначе situations := situations + ‘∧’ + event кесли кцикл

если maxnumbsit > 1 то situations := '(' + situations + ‘)’ кесли

{строка situations описывает ситуации, упоминаемые во входном тексте}

Выбор kindtext из

Stat: Semrepr := Situations

{Пример. Пусть T1 = “Профессор Игорь Новиков преподает в Томске”.

Тогда сначала Semrepr := пустая строка.

После завершения работы алгоритма BuildSem

Semrepr = *Ситуация(e1, преподавание * (Время, #сейчас#)(Агент1, нек чел * (Квалиф, профессор)(Имя, ‘Игорь‘)(Фамилия, 'Новиков’) : x2)(Место1, нек город * (Название, 'Томск’) : x3)).* }

Imp: Semrepr := Semrepr + ‘∧’ + *Situations* + *‘)’*

{Пример. Пусть T2 = "Доставь ящик с деталями на склад № 3".

Тогда сначала Semrepr := ‘*(Команда(#Оператор#, #Исполнитель#, #сейчас#, e1)*‘ .

После завершения работы алгоритма Buildsem

Semrepr = *(Команда(#Оператор#, #Исполнитель#, #Сейчас#, e1) ∧ Цель (e1, доставка1*(Объект1, нек ящик * (Содерж1, нек множ * (Кач-состав, деталь)) : x1)(Место2, нек склад * (Номер, 3) : x2))* }

Genqs: Semrepr := Semrepr + + situations + ‘)))’;

{Пример. Пусть T3 = “Проходила ли в Азии международная научная конференция “COLING”?”. Тогда сначала Semrepr := ‘*Вопрос(x1, (x1≡Ист-знач (‘* .

После завершения работы алгоритма Buildsem

Semrepr := ‘*Вопрос(x1, (x1≡ Ист-знач (Ситуация (e1, прохождение2* (Время, нек мом * (Раньше ,#сейчас#) : t1)(Событие, нет конф* (Вид1, междун) (Вид2, научная) (Название, ‘COLING’) : x2) (Место, нек континент* (Название, ‘Азия’) :x3)))))*. }

Specqs-relat1, Specqs-relat2:

нач Если Semrepr = ‘*Вопрос(x1,'*

{Пример. T4 = “Какое издательство опубликовало роман «Ветры Африки»?”}

то Semrepr = Semrepr + situations + ‘)’

{Пример. Для вопроса T4 Semrepr := ‘*Вопрос(x1, Ситуация(e1, опубликование * (Время, нек мом * (Раньше, #сейчас#) : t1) (Агент2, нек издательство: x1) (Объект3, нек роман1* (Название, ‘Ветры Африки’) :x3)))’* }

иначе {т.е. если Semrepr = ‘*Вопрос (S1, (Кач-состав (S1,'* }

posmainnoun:= Right (mainpos, сущ)

{Пример . Пусть T5 = “ С какими зарубежными издательствами сотрудничает писатель Игорь Сомов?”. Тогда mainpos = 2 (позиция вопросительного слова “какими“, posmainnoun := 4 (позиция слова “издательствами”) }

sem1 := sembase [posmainnoun]

если sem1 не включает символ ‘*’

то semhead := sem1

иначе loc1 := Matr [posmainnoun, locunit]

semhead := Arls [loc1, sem]

{Пример. Для вопроса Т5 sem1 := *издательство * (Вид-географич, зарубежное)* ,

semhead := *издательство* }

кесли

Semrepr = Semrepr + sem1 + *') ∧ Описание (произв'* + semhead + *'* (Элем . S1) : y1,'* + situations + '))'

{Пример. Для вопроса Т5 = " С какими зарубежными издательствами сотрудничает писатель Игорь Сомов?"

Semrepr := *Вопрос (S1, (Кач-состав (S1, издательство * (Вид-географич, зарубежное)) ∧ Описание(произв издательство* (Элем, S1) : y1, Ситуация(e1, сотрудничество * (Время, #сейчас#) (Агент1, нек чел* (Профессия, писатель)(Имя, 'Игорь')(Фамилия, 'Сомов'): x1)(Организация1, y1)))))* . }

Specqs-rol: {Пример 1. Пусть Т6 = "Кем выпускается препарат "Зиннат"?".

Тогда сначала Semrepr := *Вопрос (* .}

Unknowns := *'x1'*

Если numbqswd > 1

То цикл для k от 1 до numbqswd – 1

Vrb := Stringvar ('x', k) ; Unknowns := unknowns + '∧' + vrb кцикл

Unknowns := '(' + unknowns + ')' кесли

Semrepr := Semrepr + unknowns + ',' + situations + ')'

{конец Specqs-rol}

{ Пример 1. После завершения работы алгоритма BuildSem в случае вопроса Т6 = "Кем выпускается препарат "Зиннат"?"

Semrepr = *Вопрос (x1, Ситуация (e1, выпуск1 * (Время, #сейчас#) (Агент1, x1)(Продукция1, нек препарат1 * (Название, 'Зиннат') : x2)))*

Пример 2. Пусть Т7 = "Откуда и для кого поступил трехтонный алюминиевый контейнер?". Тогда сначала Semrepr := 'Вопрос ('.

После завершения работы алгоритма BuildSem

Semrepr = *Вопрос ((x1 ∧ x2) , Ситуация (e1, поступление2 * (Время, нек мом * (Раньше, #сейчас#) : t1) (Место1, x1) (Адресат, x2) (Объект1, нек контейнер * (Вес, 3 тонна))(Материал, алюминий) : x3))) .* }

Specqs-quant1

posmainnoun:= Right (mainpos, сущ)

{Пример . Пусть T8 = “Сколько человек участвовало в создании статистического сборника?”. Тогда mainpos = 1 (позиция вопросительного слова “ сколько”, posmainnoun := 2 (позиция слова “ человек”) }

sem1 := sembase [posmainnoun]

если sem1 не включает символ ‘*’

то semhead := sem1

иначе loc1 := Matr [posmainnoun, locunit]

semhead := Arls [loc1, sem]

{Пример. Для вопроса T8 sem1 := *чел* , semhead := *чел* }

кесли

Semrepr = Semrepr + sem1 + *‘) ∧ Описание (произв’* + semhead + *‘* (Элем . S1)* :

y1,’ + situations + ‘))’

{Пример. Для вопроса T8 = “Сколько человек участвовало в создании статистического сборника?”

Semrepr := ‘Вопрос(x1, ((x1≡ Колич(S1)) ∧ Кач-состав (S1, чел) ∧ Описание(произв чел (Элемент, S1) : y1, Ситуация(e1, участие1* (Время, нек мом * (Раньше, #сейчас#) : t1) (Агент1, y1)(Вид-деятельности, создание1* (Продукт1, нек сборник1 * (Область1, статистика) : x2))))) ‘ .*

Specqs-quant2: *Semrepr := Semrepr + situations + ‘)))’*

{Пример. Пусть T9 = “Сколько раз Иван Михайлович Семенов летал в Мексику?”

Тогда сначала Semrepr := *Вопрос(x1, ((x1≡ Колич(S1)) ∧ Кач-состав (S1, сит) ∧ Описание(произв сит * (Элемент, S1) : e1, .*

После завершения работы алгоритма BuildSem

Semrepr = *Вопрос(x1, ((x1≡ Колич(S1)) ∧ Кач-состав (S1, сит) ∧*

*Описание(произв сит * (Элемент, S1) : e1, Ситуация (e1, полёт * (Время, нек мом * (Раньше, #сейчас#) : t1)(Агент1, нек чел.*(Имя, ‘Иван’)(Отчество, ‘Михайлович’)(Фамилия, ‘Семёнов’): x2)(Место2, нек страна* (Название, ‘Мексика’):x3))))) .* }

9.3.4. Полный алгоритм сборки семантического представления текста

Описание алгоритма BuildSem (“Сборка-СемП”)

Внешняя спецификация

Вход : Rc – массив – классифицируюшее представление входного текста T; Rm – массив – морфологическое представление текста T; nt - целое число – длина текста T (количество строк в Rc и Matr); kindtext – строка, характеризующая вид входного текста; mainpos – целое число – позиция вопросительного слова в начале текста; Matr – МССП текста; Arls – проекция лексико-семантического словаря Lsdic на входной текст.

Выход : Performers – массив; Sitdescr - массив; Semrepr – строка - К-представление входного текста (семантическое представление входного текста, являющееся выражением некоторого стандартного К-языка).

Алгоритм BuildSem

Нач Подготовка-к-постр-СемП (Rc, Rm, Matr, kindtext, mainpos, Semrepr)
Начало-постр-СемП (Rc, Rm, Matr, kindtext, numbqswd, Arls, Performers, Sitdescr, Semrepr)
Отображение-ситуаций (Rc, Matr, Performers, Sitdescr)
Заключит-операции (Rc, kindtext, numbqswd, Matr, Performers, Sitdescr, Semrepr)
Конец

Таким образом, разработанный в данной главе алгоритм BuildSem (“Сборка-СемП”) преобразует МССП вопроса, команды или сообщения из широкого многообразия текстов на естественном (русском) языке в семантическое представление, являющееся выражением стандартного К-языка, задаваемого рассматриваемым размеченным концептуальным базисом – компонентом лингвистического базиса.

9.4. Алгоритм семантико-синтаксического анализа текстов на естественном (русском) языке

9.4.1. Описание алгоритма SemSyn (“Семантико-синтаксич-анализ-текста”)

Внешняя спецификация

Вход : *Lingb* – лингвистический базис; T - текст из множества *Texts(Tform(Lingb))* , где *Tform* - текстообразующая система, являющаяся одним из компонентов *Lingb*.

Выход : Semrepr – строка - К-представление входного текста (семантическое представление входного текста, являющееся выражением некоторого стандартного К-языка).

Алгоритм

Нач BuildMatr (T, Rc, Rm, Arls, kindtext, Matr, mainpos, numbqswd)

BuildSem (Rc, Rm, Arls, kindtext, Matr, mainpos, numbqswd, maxnumbsit, Semrepr) кон

Пример. Пусть T1 = «В каком московском издательстве в 2001-м году вышла работа по искусственному интеллекту «Основы обработки знаний» профессора Сомова ?». Лингвистический базис *Lingb* может быть определен таким образом, что на разных стадиях обработки T1 алгоритм SemSyn построит двумерные массивы Rc и Rm – классифицирующее и морфологические представления текста T1 (см.примеры в подразделе 7.1.1), рассмотренные в подразделе 7.1.2 двумерные массивы Arls (проекцию лексико-семантического словаря Lsdic на входной текст), Arvfr (проекцию словаря глагольно-предложных семантико-синтаксических фреймов Vfr на входной текст), Arfrp (проекцию словаря предложных семантико-синтаксических фреймов Frp на входной текст), а затем матричное семантико-синтаксическое представление (МССП) Matr (см. пример в параграфе 7.2).

На заключительном этапе работы процедура BuildSem построит К-представление текста Т1, являющееся строкой Semrepr вида

Вопрос (х1, Ситуация(е1, выход 1 (Издатель, нек издательство1* (Место, Москва) : х2)(Время, 2001/год)(Информ-объект, нек работа2 * (Название, ‘Основы обработки знаний’)(Область1, иск-интеллект)(Авторы, нек чел *(Квалификация, профессор)(Фамилия, ‘Сомов‘) : х4) : х3)))) .*

9.4.2. Обсуждение разработанного алгоритма семантико-синтаксического анализа текстов

Разработанный выше алгоритм SemSyn, базирующийся на построенной в главе 6 формальной модели лингвистической базы данных (ЛБД) и на введенном понятии матричного семантико-синтаксического представления (МССП), устанавливает смысловые отношения между элементарными значащими единицами входного текста, отражая эти отношения посредством МССП, а затем строит семантическое представление (СП) текста, являющееся выражением некоторого СК-языка (К-представлением). Входные ЕЯ-тексты могут выражать высказывания (сообщения), команды, специальные вопросы (т.е. вопросы с вопросительными словами), общие вопросы (т.е. вопросы с ответом «Да»/ «Нет»)и могут, в частности, включать причастные обороты и придаточные определительные предложения.

Алгоритм SemSyn позволяет устанавливать возможные смысловые отношения, в частности, в сочетаниях «Глагол + Предлог + Существительное», «Глагол + Существительное», «Существительное1 + Предлог + Существительное2», «Число + Существительное», «Прилагательное + Существительное», «Существительное1 + Существительное2», «Причастие + Существительное», «Причастие + Предлог + Существительное», «Вопросительно-относительное местоимение или местоименное наречие, играющее роль вопросительного слова + Глагол», «Предлог + Вопросительно-относительное местоимение + Глагол».

Вместе с результатами глав 6 и 7 алгоритм SemSyn выражает принципиально новый подход к семантико-синтаксическому анализу ЕЯ-текстов.

Чтобы продемонстрировать преимущества этого нового подхода по сравнению с подходами, отраженными в современной научной литературе, сравним алгоритм BuildMatr с алгоритмом семантико-синтаксического анализа ЕЯ-текстов, изложенным в монографии Дж.Ф. Люгера “Искусственный интеллект. Стратегии и методы решения сложных проблем”. 4-е издание этой монографии было опубликовано на английском языке в 2002-м году, перевод на русский язык опубликован в 2004 г. В книге, в частности, отмечается, что 4-е издание содержит обновленный материал по вопросам обработки естественных языков.

Процедура Sentence; {Анализ предложения}

Начало

вызвать процедуру noun_phrase для получения представления подлежащего;

вызвать процедуру verb_phrase для получения представления сказуемого с зависимыми словами;

с помощью объединения и ограничения связать понятие существительного, возвращаемое для подлежащего, с агентом графа для глагольной конструкции

конец

Процедура noun_phrase;

Начало вызвать процедуру noun для получения представления существительного;

выбор случая:

Неопределенный артикль и единственное число: понятие, определяемое существительным, является общим;

Определенный артикль и единственное число: связать маркер с понятием, определяемым существительным;

Множественное число: указать, что существительное во множественном числе

конец выбора случая

конец;

Процедура verb_phrase;

Начало вызвать процедуру verb для получения представления глагола;

если с глаголом связано дополнение

то начало вызвать процедуру noun_phrase *для получения представления дополнения;*

с помощью объединения и ограничения связать понятие, определяемое дополнением, с дополнением, соответствующим сказуемому

конец

конец;

Процедура verb*;*

Начало получить падежный фрейм для глагола конец;

Процедура noun*;*

Начало получить понятие для существительного конец.

Приведенный выше алгоритм отражает основные характерные черты доминирующего как в отечественной, так и зарубежной научной литературе подхода к описанию алгоритмов семантико-синтаксического анализа ЕЯ-текстов. Этими характерными чертами являются отсутствие модели лингвистической базы данных (заменяемое отдельными неформальными примерами используемых данных), отсутствие формального или достаточно четкого неформального описания структуры входных текстов и, как следствие, отсутствие в публикациях реальных алгоритмов семантико-синтаксического анализа (ССА) текстов или даже подробных методов выполнения ССА.

По существу, приведенный выше текст с названием *Процедура Sentence* является не алгоритмом, а лишь *пожеланием* разработать такой алгоритм. Разные специалисты в области компьютерной обработки ЕЯ разработают по этому пожеланию *разные* алгоритмы. Это относится не только к приведенному выше фрагменту из монографии Дж.Ф. Люгера, но и к подавляющему большинству публикаций, посвященных семантико-синтаксическому анализу ЕЯ-текстов.

Результаты данной монографии, изложенные в главах 6 - 9, дают не только продвижение вперед, но и *качественный скачок* в области разработки формальных средств и методов проектирования алгоритмов ССА ЕЯ-текстов. Разработчики ЛП впервые получили широко применимый *формальный аппарат для описания структуры данных,* с которыми работает алгоритм ССА, а также

детальный метод описания алгоритмов ССА и *оригинальный алгоритм ССА*, базирующийся на формальной модели ЛБД.

Существенным преимуществом разработанного алгоритма SemSyn является явный учет многозначности слов, что чрезвычайно важно для приложений.

Анализ построенного алгоритма SemSyn показывает работоспособность предложенного в главе 7 нового метода выполнения преобразования “ЕЯ-текст ➔ СП текста”. Важная особенность этого метода и алгоритма SemSyn заключается в том, что они не предусматривают использования синтаксического уровня представления (как результата выполнения синтаксического анализа) текста. Разработка алгоритма SemSyn показала, что такие традиционные понятия синтаксиса, как, например, подлежащее и дополнение, являются избыточными для компьютерной обработки ЕЯ-текста: семантическое представление текста может быть построено без опоры на эти понятия.

В этой связи можно отметить, что с учетом характера используемых данных из ЛБД и принципов применения этих данных для построения СП ЕЯ-текста без выполнения синтаксического анализа текста центральные идеи алгоритма SemSyn имеют некоторые общие черты с идеями компьютерной семантики русского языка.

Например, согласно работе (Тузов 2001), процесс компьютерного анализа текста делится на три части: морфологический анализ, предварительная пословная обработка текста и собственно семантический анализ текста. Последний этап характеризуется как выбор конкретного морфо-семантического значения словоформы и связывания всех слов предложения в единую семантическую структуру, причем на данном этапе используется семантический словарь.

С точки зрения материалов глав 7 - 9 данной книги, морфологический анализ и предварительная пословная обработка текста примерно соответствуют построению компонентно-морфологического представления текста. Для реализации собственно семантического анализа текста в данной монографии разработан сложный структурированный алгоритм семантико-синтаксического анализа ЕЯ-текстов SenSyn, описанию которого посвящены главы 6 – 9.

Однако уровень проработанности вопросов формального описания структуры ЛБД, структуры промежуточных данных и алгоритма преобразования ЕЯ-текстов в семантические представления в главах 6 – 9 данной монографии значительно выше, чем в публикациях по компьютерной семантике русского языка (Тузов 2001; Лезин, Каневский, Тузов 2002; Тузов 2003). В частности, в указанных публикациях отсутствуют формальная модель ЛБД и четкое описание алгоритма построения СП текста.

Разработка аппарата СК-языков в главах 2, 3 и применение этого аппарата в модели ЛБД (глава 6) и в алгоритме SemSyn позволили преодолеть трудности принципиального характера, касающиеся отображения содержания команд, а также вопросов нескольких видов: с вопросительными словами “какие”, “каких” и т.д., со словом “сколько”, относящимся к количеству предметов, и с ответом “Да /Нет”.

Алгоритм семантической сборки BuildSem, являющийся частью алгоритма SemSyn, существенно использует ряд новых выразительных возможностей, предоставляемых определением класса СК-языков.

Пример 1. Пусть В1 = “ С какими зарубежными издательствами сотрудничает писатель Игорь Сомов?”. Тогда для некоторого лингвистического базиса Lingb алгоритм SemSyn построит по вопросу В1 его К-представление (КП) в виде цепочки

Semrepr1= *Вопрос (S1, (Кач-состав (S1, издательство * (Вид-географич, зарубежное)) ∧ Описание(произв издательство* (Элем, S1) : y1, Ситуация(e1, сотрудничество * (Время, #сейчас#) (Агент1, нек чел* (Профессия, писатель)(Имя, ‘Игорь’)(Фамилия, ‘Сомов’): x1)(Организация1, y1)))))* .

Фрагментами цепочки Semrepr1 являются: (а) составное обозначение понятия *издательство * (Вид-географич, зарубежное)* , (б) семантическая характеристика произвольного элемента множества *произв издательство* (Элем, S1) : y1*, (в) составное обозначение объекта *нек чел* (Профессия, писатель)(Имя, ‘Игорь’)(Фамилия, ‘Сомов’): x1* . Правило Р[5] позволило связать метку (переменную) *y1* с произвольным элементом искомого множества *S1* , а затем использовать только эту метку для последующих ссылок на эту характеристику.

Пример 2. Пусть B2 = “Проходила ли в Азии международная научная конференция “COLING”?”. Тогда в рамках некоторого лингвистического базиса Lingb алгоритм SemSyn построит КП вопроса B2 в виде цепочки

Semrepr2 = *Вопрос(x1, (x1≡ Ист-знач (Ситуация (e1, прохождение2* (Время, нек мом * (Раньше ,#сейчас#) : t1)(Событие, нет конф* (Вид1, междун) (Вид2, научная) (Название, ‘COLING’) : x2) (Место, нек континент* (Название, ‘Азия’) :x3)))))*.

В выражении Semrepr2 цепочка *Ист-знач* интерпретируется как обозначение функции, аргументом которой является СП высказывания, а значением – логическая величина Истина или Ложь.

Таким образом, использование СК-языков для построения СП входных текстов лингвистического процессора позволило расширить возможности отображения особенностей смысловой структуры входных текстов по сравнению с другими известными подходами к построению СП ЕЯ-текстов. В частности, это относится к командам и к текстам с составными описаниями множеств.

По глубине проработки вопросов преобразования компонентно-морфологического представления текста в его СП и ясности описания предложенных решений разработанный алгоритм не имеет аналогов как в отечественной, так и зарубежной научной литературе на английском языке.

Содержание данной главы и глав 6 - 8 отражено в публикациях (Фомичев 1978б – 1980, 1986а – 1989, 1990в – 1992б, 2002а, б, в; Фомичев, Волчков 1999; Фомичев, Люстиг 2004; Fomichov 1992 – 1994, 1996a – 1998a, 2002a, 2002b, 2005в – 2005e; Fomichov, Akhromov 2001; Fomichov, Chuykov 2000; Fomichov, Kochanov 2001; Fomichov, Lustig 2001; Fomichova, Fomichov 2004).

9.5. Применение разработанного алгоритма к проектированию русскоязычных интерфейсов прикладных компьютерных систем

9.5.1. Русскоязычные интерфейсы баз данных и баз знаний

Работоспособность предложенного структурированного алгоритма семантико-синтаксического анализа текстов на естественном (русском) языке, разработанного в главах 8 и 9, доказана тем, что на его основе в рамках учебного процесса сконструирован целый спектр лингвистических процессоров (ЕЯ-интерфейсов) баз данных и баз знаний, реализованных в программных средах Turbo-Pascal 7.0, Borland-Pascal 7.0, Delphi 4.0, Delphi 5.0, C, C++, Visiul C++, Action Script, PHP.

Реализация ЕЯ-интерфейсов (русскоязычных интерфейсов) осуществлялась в МИЭМ в рамках курсового и дипломного проектирования, а также в “МАТИ” – Российском государственном технологическом университете им. К.Э. Циолковского в рамках выполнения курсовых работ по дисциплине “Проектирование лингвистических процессоров” и дипломного проектирования.

Рассмотрим характеристики наиболее интересных из этих программных реализаций.

ПРИМЕР 1. ЕЯ-интерфейс “Фармацевт” предназначен для обработки запросов к базам данных, хранящим сведения о различных лекарственных препаратах, а также для ввода в базу данных содержания сообщений о препаратах. Для реализации использовалась программная среда C++ 3.1 фирмы Borland.

Примерами входных текстов ЛП являются вопросы “Откуда поступил анальгин?”, “Сколько стоит анальгин?”, “Кто призводит анальгин?”, “Для кого поступил анальгин?”, “Когда и откуда поступил анальгин?”, “Откуда и для кого поступил анальгин?”, “Какие препараты, выпускаемые фирмой “GlaxoWelcome”, предназначены для больных астмой?”, “В каких странах выпускается препарат бекатит?”, “Какие европейские страны выпускают препарат серетид?” и сообщения “Анальгин поступил из Польши”, “Бромгексин поступил на склад”, “Со склада поступил инсулин”.

ПРИМЕР 2. “РУСЛАН-1” (РУСскоязычный Лингвистический АНализатор – Первая версия) – семантико-синтаксический анализатор вопросов к БД и команд интеллектуальному транспортно-погрузочному роботу. Включает подсистемы морфологического анализа и семантико-синтаксического анализа. Реализация в среде Turbo-Pascal 7.0.

ПРИМЕР 3. ЛП “СКЛАД ГПС”. Модельная предметная область: обработка ЕЯ-запросов к БД оператора автоматизированного склада ГПС. Реализация в среде Borland Delphi 4.0, язык Object Pascal. Отлажена в среде Windows 2000. Пример запроса: “Откуда и для кого поступили 3 двухтонных алюминиевых контейнера?”

ПРИМЕР 4. ЕЯ-интерфейс интеллектуальной вопросо-ответной системы АНТЕК-1 (АНАлиз ТЕКстов, версия 1) является семантико-синтаксическим русскоязычным анализатором вопросов и сообщений о событиях (о публикации научных работ, участии в научных конференциях, разработке новых приборов, получении и отправке товаров и т.д.).

Интерфейс реализован на языке C++ с использованием библиотек классов MFC и OLE DB Template Library фирмы Microsoft. Это позволило использовать в качестве хранилища базы знаний реляционные СУБД Microsoft Access или Microsoft SQL-Server, которые позволяют оперировать большими объемами данных.

При реализации алгоритмов поиска в базе данных существует возможность использовать для этих целей средства самой СУБД, что позволяет в некоторых случаях значительно упростить задачу поиска.

Пример 5. В среде Visual C++ разработан прототип русскоязычного интерфейса интеллектуальной базы данных, предназначенной для подбора вин и составления ресторанной винной карты в ходе взаимодействия конечного пользователя с Web–сайтом Российской ассоциации сомелье (РАС) и Web–сайтом Интернет-магазина, разработанного при поддержке РАС .

Ниже приведено несколько примеров входных текстов этого интерфейса; каждый из текстов может являться входом алгоритма семантико-синтаксического анализа, разработаннjuj в данной монографии:

• Посоветуйте белое французское вино к рыбному столу. • Какие красные вина, производимые в провинции Бургундия, импортирует фирма “Радуга”? • По какой технологии производят вино “Божоле”?

• Сколько фирм поставляют сухое белое вино “Бельканто”? • Выращивают ли виноград “Изабелла” в провинции Прованс?

•Откуда и кто поставляет сухое белое вино “Бельканто”?

•Из какого винограда производят красное вино “Божоле”?

•Сколько раз в апреле заказывали французское вино “Мерло“?

ПРИМЕР 6. Одна из исходных версий разработанного в данной главе алгоритма семантико-синтаксического анализа текстов была применена Я.В. Ахромовым (в рамках дипломного проектирования на кафедре информационных технологий МАТИ) к конструированию лингвистического процессора анимационной системы, предназначенной для имитации взаимодействия с интеллектуальным транспортно-погрузочным роботом, действующим в аэропорту (см. рисунок 9.5).

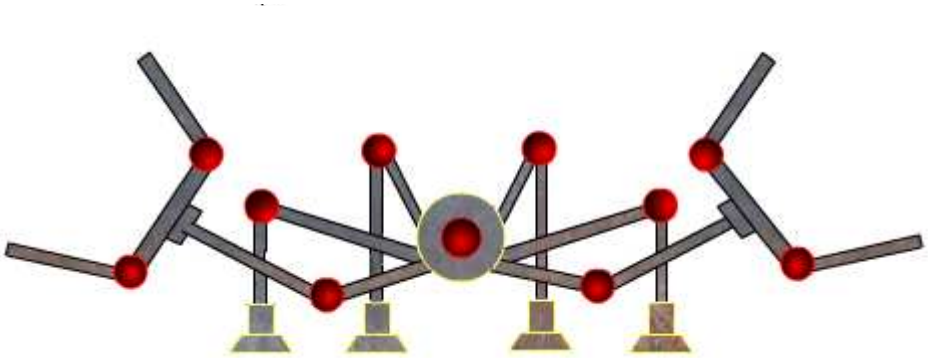

Рис. 9.5. Внешний вид возможного авиаробота

Анимационная система с ЛП, обрабатывающим команды и вопросы авиароботу , проектировалась с использованием среды разработки Flash 5. Этот выбор был сделан в связи с тем, что среда Flash 5 может быть использована как графическая программа, программа обработки звука и среда программирования.

Для реализации ЛП использовался язык программирования Action Script, встроенный в среду Flash 5.

Работа ЛП и связанных с ним подсистем направлена на осуществление общения оператора погрузочно-разгрузочных работ с авиароботом. Разработанный робот является только виртуальной моделью и не имеет реальных прототипов на практике. Задача оператора заключается в том, чтобы вводить команды авироботу средствами естественного языка, например: “Погрузи 3 контейнера в самолёт авиакомпании SAS, рейс (SK-6787).” Авиаробот является в этой системе некоторым интеллектуальным агентом, который способен оценивать текущую ситуацию с погрузкой в соответствии с динамическим расписанием погрузочно-разгрузочных работ и отслеживать верность запрашиваемой оператором команды.

Входной текст ЛП (команда или вопрос) подвергается анализу на корректность. Например, если оператор введет количество контейнеров для погрузки больше, чем заявлено в расписании погрузочно-разгрузочных работ, система должна будет отреагировать на это сообщением об ошибке. Другим примером может быть ситуация, при которой оператор указывает рейс, который уже произвел взлет. Тогда система должна будет уведомить оператора, что данный рейс является недоступным для погрузочных работ.

Помимо этого, оператор может задавать авироботу вопросы нескольких типов, например, “Сколько контейнеров было погружено на рейс (SK-6787)?”, “Грузил ли контейнеры к терминалу <номер терминала>?”, “Сколько контейнеров погрузил на все рейсы авиакомпании <имя авиакомпании>?”.

Это даёт возможность оператору следить за ходом работы авиаробота, за динамикой погрузки, прогнозировать следующие погрузочные работы.

Разработку семантико-синтаксического анализатора письменных вопросов и сообщений авироботу можно рассматривать как шаг на пути организации устного взаимодействия оператора и транспортно-погрузочного авиаробота.

Одной из базовых процедур алгоритма BuildMatr (алгоритма построения матричного семантико-синтаксического представления входного текста) является алгоритм “Найти-множ-тематич-ролей” (параграф 8.7), существенно использующий в работе словарь глагольно-предложных семантико-

синтаксических фреймов (см. параграф 6.5). Применяя терминологию статей (Баллард 1989; Хейз, Гауптман, Карбонелл, Томита 1989), можно сказать, что работа алгоритма “Найти-множ-тематич-ролей” основывается на применении семантических падежных фреймов.

Во второй из указанных статей на основе экспериментальных исследований, проведенных в Университете Карнеги-Меллон, сделан вывод о перспективности использования семантических падежных фреймов для семантико-синтаксического анализа устной речи. В этой связи можно сделать заключение о перспективности использования разработанного в данной главе алгоритма семантико-синтаксического анализа ЕЯ-текстов (алгоритма SemSyn) при проектировании анализаторов устной речи, т.е. при решении одной из актуальных проблем разработки лингвистических информационных технологий.