Модели и методы анализа иерархически структурированных текстов

Models and methods for the analysis of hierarchically structured texts

Скатов Д. С. (ds@dictum.ru),
Ерехинская Т. Н. (te@dictum.ru), **Окатьев В. В.** (oka@dictum.ru)
ООО «Диктум», Нижний Новгород, Россия

В статье обсуждается задача анализа иерархически структурированных текстов (законы, кодексы, стандарты). Дано описание задачи, способы применения результатов анализа и обзор разработок в области. Описаны разработанные модели и методы анализа иерархически структурированных текстов.

1. Введение

Большинство исследователей под обработкой естественно-языкового текста традиционно понимают обработку текста, представляющего собой набор предложений без выраженной структуры. В настоящее время становится актуальной задача обработки документов, обладающих высокой степенью формализованности и, как следствие, внутренней иерархической структурой. К ним относятся, в первую очередь, юридические тексты, нормативно-техническая документация, описания стандартов. Интерес к анализу подобных текстов обусловлен тем, что учет информации об иерархии в системах поиска и анализа документов позволяет предоставить их пользователям новые инструменты, повышающие эффективность работы ([7, 8, 9, 10]).

При поиске информации в коллекции сложно структурированных текстов пользователю недостаточно одного лишь списка релевантных документов в качестве поисковой выдачи по причине больших объемов и высокой сложности документов. Повышение эффективности поиска в таких документах может быть достигнуто, если пользователь будет получать в качестве поисковой выдачи не только документы, но и цитаты из них — точные дословные выдержки из текста, обладающие смысловой законченностью. Цитаты могут быть получены с помощью анализа иерархической структуры текстов, и далее могут быть уточнены с применением синтаксического анализа. В результате пользователь получает компактную поисковую выдачу, в которой отсечен значительный объем информации, нерелевантный запросу. Способ поиска информации, основанный на извлечении цитат из текстов, описан в [14].

Задача извлечения цитат тесно связана с задачами автоматического реферирования текстов [9, 10], поэтому предложенные в статье модели и методы могут быть эффективно использованы в этой области.

Сложность задачи анализа иерархически структурированных текстов обусловлена следующими их свойствами:

1) как правило, разметка заголовков и маркеров (с помощью стилей, тэгов и т.д.) в документе присутствует лишь частично или отсутствует;
2) заголовки из разных уровней иерархии могут быть неотличимы по виду;
3) заголовок и ссылка на него в тексте могут иметь одинаковый вид;
4) богатство конфигураций непрерывных текстовых фрагментов: предложение может состоять из нескольких таких фрагментов, один фрагмент может включать несколько предложений, группа предложений может быть вложена в предложение в виде комментария.

Эти сложности преодолены в рамках подхода, рассматриваемого в статье.

2. Обзор

До последнего времени не было разработано математических моделей, пригодных для построения систем анализа иерархически организованных текстов ([1, 2, 3]). В работах [4, 5, 6] рассматривается задача деления на предложения линейного текста, при этом возможное наличие в тексте иерархической структуры не учитывается. В публикации [7]

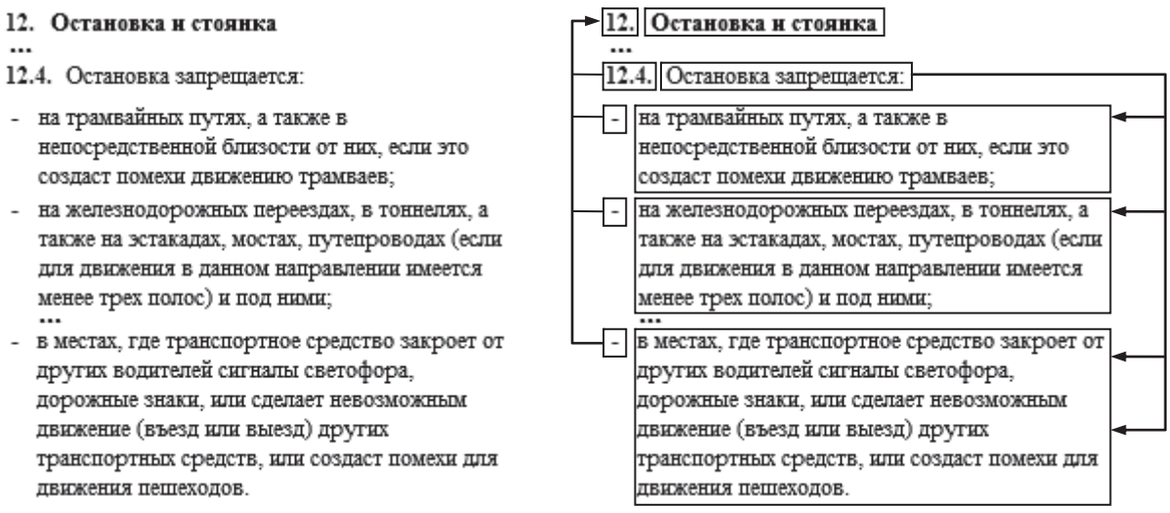

Рис. 1. Фрагмент текста ПДД (слева) и результат анализа его иерархической структуры (справа)

признается важность учета иерархии текстов при поиске, описывается модель поведения пользователей при работе со структурированными документами, анализируются примеры возможных текстовых иерархий. В работе [8] приведены краткие сведения о поиске в иерархически структурированных документах. В публикациях [9, 10] приводится информация об использовании иерархической структуры в задаче автореферирования текстов. Однако во всех упомянутых работах не описаны модели и методы анализа сложно структурированных текстов, которые были бы применимы для решения широкого спектра задач поиска информации.

В HTML-документах информация об иерархии содержится в гипертекстовой разметке. Ее дерево легко можно получить с помощью одной из множества существующих библиотек [15, 16]. В plain-текстах стандартизированная и тривиально определяемая разметка (подобная HTML) отсутствует: перед анализом иерархии определяющую ее разметку (прежде всего — заголовки и маркеры) фактически необходимо выявить, при этом учитывая, что различные тексты, как правило, обладают различной разметкой. В сравнении с методами очистки веб-страниц [17, 18], по сути удаляющих из документа информацию об иерархии, в представленном далее подходе эта информация используется для поиска и получения цитат.

Предложенный в статье подход к анализу иерархической структуры связан с анализом нумерации (маркировки) разделов документа. Определение нумерованных (маркированных) фрагментов позволяет более точно определить границы предложений и сформировать их в виде, удобном для последующей обработки. Полученная в результате разметка текста может быть использована в поисковых системах, в частности, она используется в системе извлечения цитат «Dictum» [12, 13].

Конечно, зная характерную структуру *конкретного документа*, легко разработать утилиту, которая строит дерево этого документа. На практике требуется анализировать сотни документов с *различной внутренней структурой*, что возможно в рамках предлагаемого в данной статье подхода.

Поясним суть задачи анализа иерархии текста на примере текста Правил Дорожного Движения [11]. Рассмотрим фрагмент, содержащийся в п. 12.4 этого текста. На Рис. 1 слева показан исходный текст.

Под иерархическим анализом текста понимается выполнение следующих действий:

1) определение в исходном тексте элементов маркировки — заголовков,
2) определение вложенности групп заголовков,
3) определение отдельных предложений (цитат) из фрагментов текста, расположенных на разных уровнях иерархии.

На Рис. 1 справа показана разметка, полученная в результате анализа иерархии: заголовки и маркеры связаны отношением подчинения (в левой части рисунка), текстовые фрагменты — отношением следования (в правой части рисунка). Из текстовых фрагментов по отношению следования могут быть выделены цитаты.

3. Линейная структура текста

3.1. Текстовый фрагмент

Рассмотрим фрагмент *txt* входного текста. *Текстовым фрагментом* назовем набор (l, r, txt, pl, pr, P).

- *txt* будем называть *значащей частью фрагмента*.
- $P \in D$ представляет собой *разделитель* — символ, который следует после значащей части фрагмента. Между фрагментом и его разделителем допускается наличие пробельных символов. Разделитель является свойством фрагмен-

```
1  Сигналы регулировщика имеют следующее значение :
2  руки вытянуты в стороны или опущены :
3  со стороны левого и правого бока разрешено движение трамваю прямо,
   безрельсовым транспортным средствам прямо и направо, пешеходам разрешено
   переходить проезжую часть ;
```

Рис. 2. Пример разделителей в тексте

та, но не входит в значащую часть *txt*. Пример изображен на Рис. 2: в строках 1–3 части текста, обведенные в сплошную рамку, представляют собой значащие части (*txt*) фрагментов, а символы, заключенные в пунктирную рамку — их разделители.

- Параметры *pl, pr* принимают булевские значения и означают соответственно наличие слева и справа от фрагмента символа переноса строки. В примере на Рис. 2 для трех представленных фрагментов $pl = pr = true$.

В процессе анализа иерархии строится разбиение входного текста α на непересекающиеся фрагменты, в совокупности образующие упорядоченное множество.

3.2. Ячейка

Ячейка представляет собой запись некоторого порядкового значения (величины, номера) в фиксированной системе нумерации. Определим ячейку как набор (T, L, C, os, N). Первые четыре компонента определяют *систему нумерации ячейки*, последний компонент — *числовой эквивалент ячейки* в ее системе нумерации.

- $T \in$ {*числовая, буквенная*} — определяет тип системы нумерации ячейки;
- $L \in$ {*латинская, русская, арабская, римская*} — представляет собой локализацию системы нумерации. Напр., (1) *ячейка «б» записана в системе с русской локализацией, ячейка «XI» — с римской*.
- Для латинской и русской локализаций предусматриваются многопозиционные системы нумераций. В них допускаются порядковые значения, составленные из нескольких символов алфавита. Напр., при такой нумерации допустимы ячейки (2) *«аа» и «аб»*.
- $C \in$ {*верхний, нижний*} — регистр системы нумерации, которой принадлежит данная ячейка. Напр., (3) *ячейка «А» записана в системе с верхним регистром*. Свойство регистра не задано для ячеек с арабской локализацией.
- $os \in D$ представляет собой открывающий символ данной ячейки. Напр., (4) *ячейка «.а» имеет открывающий символ «.»*. Если ячейка не имеет открывающего символа, то $os = null$.
- N — числовой эквивалент ячейки в ее системе нумерации. Строится функция φ: если $\alpha \in D^*$ представляет собой запись какого-либо порядкового значения в некоторой системе нумерации, то $\varphi(\alpha) = N$ представляет собой число, однозначно определяющее эту запись в данной системе нумерации; иначе $\varphi(\alpha) = \infty$. Полагается $N = \varphi(\alpha)$. Напр.: (5) *α = «VII» — это запись порядкового значения в системе нумерации с римской локализацией, верхним регистром, а число $\alpha = 7$ однозначно определяет эту запись в данной системе нумерации. α = «остановка» не является записью порядкового значения в какой-либо системе нумерации, поэтому $\varphi(\alpha) = \infty$.*

На множестве всех ячеек вводятся отношения *равенства* «=» и *непосредственного следования* «$\prec$». Отношение равенства «=» определяется как равенство векторов:

$$Cell_1 = (T_1, L_1, C_1, os_1, N_1) = (T_2, L_2, C_2, os_2, N_2) = Cell_2 \Leftrightarrow$$
$$\Leftrightarrow os_1 = os_2,\ T_1 = T_2,\ L_1 = L_2,\ C_1 = C_2,\ N_1 = N_2.$$

Отношение непосредственного следования «$\prec$» определим так: $Cell_1 \prec Cell_2 \Leftrightarrow (os_1 = os_2,\ T_1 = T_2,\ L_1 = L_2,\ C_1 = C_2)\ \&\ N_2 - N_1 = 1$. Напр., (6) *«a» $\prec$ «b», но «a» $\not\prec$ «c»*.

Множество всевозможных ячеек, распознаваемых системой, описывается композицией конечных автоматов, реализация которых и выполняет определение ячеек в тексте.

3.3. Заголовок

Заголовок представляет собой конечный набор ячеек с дополнительными свойствами. Определим его набором: (T, *Cells*, *txt*, *pos*, *os*, *cs*, *pr*, *t*).

- $T \in$ {*пустой, маркированный, нумерованный, корневой*} — определяет тип заголовка.
  - *Маркированный заголовок* используют для обозначения серии однотипных пунктов. В неразмеченных текстах такие заголовки представляются некоторым декоративным символом — маркеро м, напр. (7) *«*» или «–»*.
  - *Пустой заголовок* часто встречается в неформатированных текстах. В начале каждой строки, которая логически представляет собой новый пункт, можно разместить виртуальный маркер ∅.

- *Корневой заголовок* вводится в рассмотрение искусственно как будущий корень иерархии.
- *Нумерованный* заголовок определяется свойством *Cells*. Они различаются согласно порядку нумерации. Если *T* = *нумерованный*, то *Cells* = (*Cell*$_1$, ... , *Cell*$_k$) представляет собой набор ячеек, образующих данный заголовок, иначе *Cells* = *null*. Напр., (8) *заголовок «1.б» имеет следующий набор ячеек*:

$((\text{числовая}, \text{арабская}, \varnothing, \text{null}, 1), (\text{буквенная}, \text{русская}, \text{нижний}, ".", 2))$.

- *txt* представляет собой часть исходного текста, определяющую заголовок (если *T* = *пустой*, то *txt* = *null*).
- *pos* задает расположение *txt* в исходном тексте, если $T \neq$ *пустой*, и ту позицию, куда может быть вставлен виртуальный маркер, если *T* = *пустой*.
- Параметры $os, cs \in D$ представляют собой соответственно открывающий и закрывающий символы заголовка. Напр., (9) *заголовок «(1.а)» имеет открывающий символ* $os = «(»$, *закрывающий символ* $cs = «)»$.
- Для каждого текста можно указать набор префиксов — слов, которые могут открывать заголовок. Множество префиксов *P* упорядочивается лексикографически, и его элементы нумеруются согласно этому порядку, начиная с нуля. Тогда *pr* представляет собой номер префикса данного заголовка. Напр., (10) *если P* = {*раздел*, *статья*}, *то заголовок «Статья 1.1» имеет префикс pr* = 1.
- В процессе анализа некоторые фрагменты могут быть определены как названия соответствующих заголовков — тогда фрагмент изымается из множества *F* и становится атрибутом *t* заголовка. Напр., (11) *в тексте «Статья 6.1. Порядок исчисления сроков, установленных законодательством о налогах и сборах» имеется заголовок с префиксом и фрагмент после него, причем фрагмент является названием данного заголовка*.

Множество всех заголовков *H* строится так, чтобы части текстов, соответствующие различным заголовкам из *H*, не пересекались. Упорядочим элементы *H* в порядке их следования в тексте, и установим отношения *непосредственного следования* «$\prec$» и *непосредственного подчинения* «$\lhd$».

Пусть $H_1, H_2 \in H$,
$H_j = (T_j, Cells_j, txt_j, pos_j, os_j, cs_j, pr_j, t_j)$, $j = 1,2$.
Отношение непосредственного следования «$\prec$» определяется следующей цепочкой проверок:

- $T_1 \neq T_2 \Rightarrow H_1 \not\prec H_2$;
- $T_1 \neq$ *нумерованный* & T_2 = *нумерованный* $\Rightarrow$ $H_1 \prec H_2$;
- Пусть $T_1 = T_2$ = *нумерованный*, тогда:
- $cs_1 \neq cs_2 \Rightarrow H_1 \not\prec H_2$;
- $os_1 \neq os_2 \Rightarrow H_1 \not\prec H_2$;
- $pr_1 \neq pr_2 \Rightarrow H_1 \not\prec H_2$;
- $length(Cells_1) \neq length(Cells_2) \Rightarrow H_1 \not\prec H_2$;
- $length(Cells_1) = 1$ & $cs_1 = "."$ & $pr_1 = \text{null}$ & $(t_1 = \text{null}$ & $t_2 \neq \text{null} \mid t_2 = \text{null}$ & $t_1 \neq \text{null}) \Rightarrow H_1 \not\prec H_2$;
- Если $length(Cells_1) > 1$, то:
  - $\exists\, j \in \{1, length(Cells_1) - 1\}$: $Cell_{1j} \neq Cell_{2j} \Rightarrow H_1 \not\prec H_2$;
  - $k = length(Cells_1)$, $\forall\, j \in \{1, k-1\}$ $Cell_{1j} = Cell_{2j} \Rightarrow$ $H_1 \prec H_2 \Leftrightarrow Cell_{1,k}.T = Cell_{2,k}.T$ & $Cell_{1,k} \prec Cell_{2,k}$;
- $T_1 = T_2 =$ *маркированный* $\Rightarrow$ $H_1 \prec H_2 \Leftrightarrow \text{ord}\, H_2 - \text{ord}\, H_1 = 1$;
- $T_1 = T_2 =$ *пустой* $\Rightarrow$ $H_1 \prec H_2 \Leftrightarrow \text{ord}\, H_2 - \text{ord}\, H_1 = 1$;
- в остальных случаях $H_1 \not\prec H_2$.

Отношение непосредственного подчинения «$\lhd$» определяется цепочкой проверок:

- ! ($T_1 = T_2 =$ *нумерованный*) $\Rightarrow H_1 \not\lhd H_2$;
- ! ($length(Cells_2) > 1$ & $length(Cells_2) - length(Cells_1) = 1) \Rightarrow H_1 \not\lhd H_2$;
- $pr_1 \neq pr_2 \Rightarrow H_1 \not\lhd H_2$;
- $\exists\, j \in \{1, length(Cells_1)\}$: $Cell_{1j} \neq Cell_{2j} \Rightarrow H_1 \not\lhd H_2$;
- в остальных случаях $H_1 \lhd H_2$.

4. Иерархическая структура текста

4.1. Древовидная модель иерархии

Иерархию в тексте представим деревом, в котором узлы соответствуют объектам, а ветви задают отношение подчинения (включения). Если у некоторого объекта-вершины имеются непосредственные объекты-потомки, то они находятся с родительской вершиной в отношении непосредственного подчинения.

Будем рассматривать *линейные компоненты текста — заголовки* и *фрагменты* — как вершины дерева. Расширим множество заголовков и фрагментов дополнительным объектом — *параграфом*. *Параграф* — это вершина, которой могут быть подчинено некоторое кол-во фрагментов.

Т.о., моделью иерархической структуры текста является дерево (V, E) (V — множество его вершин, E — множество ребер), заданное как бинарное отношение непосредственного подчинения на V. Множество вершин состоит из элементов вида (T, Obj), где тип элемента задан параметром $T \in$ {*заголовок, фрагмент, параграф*}, а параметр Obj обозначает объект типа T, представляющий собой данную вершину.

Задача анализа иерархии состоит в построении дерева (V, E), моделирующего иерархическую структуру этого текста. Это построение выполняется в три прохода:

1) на первом проходе строится дерево T_1, состоящее только из заголовков;
2) второй проход выявляет фрагменты и присоединяет их к дереву T_1 в линейном порядке, результатом является дерево T_2; также выполняется определение некоторых фрагментов как названий соответствующих заголовков;
3) третий проход выполняет коррекцию дерева T_2 с целью определения иерархии фрагментов; результатом является дерево T_3.

4.2. Определение иерархии заголовков

Введенные отношения непосредственного следования и подчинения на множестве H используются в методе определения иерархии для этого множества.

Полагаем, что множество H упорядочено по линейному расположению заголовков в тексте. В начало множества H добавим заголовок H_0 с T = *корневой*.

Построим процедуру формирования дерева. В ней параметры *from, to* представляют собой некоторый диапазон элементов множества H, *parent* — элемент H, который является родительским на данном уровне рекурсии, *Father* — узел в дереве T_1, который является предком для других узлов на данном уровне рекурсии.

```
procedure Recurrent (from, to, parent, Father) {
    prev := from; next := from + 1;
    пока (prev != to) {
        если (next = to) {
            V.T := заголовок;
            V.Obj := H_prev ∈ H;
            добавить_потомка (T_1, Father, V);
            если (next — prev != 1)
                Recurrent (prev+1, next, prev, V);
            прервать_цикл;
        }
        если (H_prev ≺ H_next) {
            V.T := заголовок;
            V.Obj := H_prev ∈ H;
            добавить_потомка (T_1, Father, V);
            если (next — prev != 1)
                Recurrent (prev+1, next, prev, V);
            prev := next;
            next:=next+1;
            продолжить_цикл;
        }
        если (parent != null) {
            если (H_next ◁ H_parent) {
                V.T := заголовок;
                V.Obj := H_prev ∈ H;
                добавить_потомка (T_1, Father, V);
                если (next — prev != 1)
                    Recurrent (prev+1, next, prev, V);
                prev := next;
                next:=next+1;
                продолжить_цикл;
            }
        }
        next := next+1;
    }
}
```

Теперь построение дерева можно выполнить вызовом рекуррентной процедуры: **Recurrent** (0, *lenght*(H), null, *root*(T_1)).

4.3. Определение и добавление фрагментов

Для добавления фрагментов к дереву будем выполнять проход по дереву T_1 в ширину (breadth-first). При построении T_2 используется тот факт, что фрагменты заключены между заголовками текста. В процессе прохода по дереву фиксируются участки текста, заключенные между смежными заголовками, в этих участках текста выявляются фрагменты, которые добавляются как потомки соответствующих заголовков.

4.4. Коррекция вершин и финальная обработка

Чтобы T_2 приобрело требуемый вид T_3, необходимо применить к нему процедуру коррекции вершин. Она выполняется в три этапа.

На первом этапе осуществляется вынесение вершин-фрагментов из внутренних уровней во внешние. Решение о перемещении некоторой группы вершин принимается на основании свойств вершин, окружающих эту группу.

Появление иерархии в структуре текста наблюдается в двух ситуациях:

1) Разбиение множества предложений на иерархические группы: совокупность предложений разделяется на множества, каждому из которых присваивается определенный уровень иерархии и позиция в этой иерархии;
2) Разбиение одного предложения на иерархические части: компоненты, на которые таким образом разбивается предложение, могут быть разделены текстовыми вставками, независимыми от данного предложения — например, комментариями.

Возможна также композиция этих ситуаций. Обрабатывать ситуацию (а) позволяют вершины-заголовки, (б) — вершины-фрагменты, подчиняющие себе другие вершины. Вершины-фрагменты выявляются на *втором этапе*.

На третьем этапе выполняется введение в дерево вершин-параграфов и подчинение последовательно идущих фрагментов этим вершинам так, чтобы вершины-параграфы и их потомки отражали структуру абзацев исходного текста.

Детально эти этапы рассмотрены в работе [12].

5. Заключение

Полученные результаты позволяют сделать вывод, что разработанные модели и методы анализа иерархической структуры текста могут быть эффективно использованы в ряде приложений поиска и анализа текстов. Решенная задача является актуальной, т.к. возможность учета иерархии в текстах в настоящее время востребована, однако моделей и методов, которые могут быть использованы для решения широкого класса задач, ранее представлено не было. В рамках проведенных исследований разработана математическая модель линейной и иерархической структур текста, а также методы выявления структуры текста. На основе разработанных моделей и методов построен прототип системы извлечения цитат Dictum, который доступен в режимах браузера и ICQ-клиента на сайте [13].

Исследование проводилось малым инновационным предприятием ООО «Диктум» при поддержке Фонда содействия развитию малых форм предприятий в научно-технической сфере, проект № 6466 «Разработка компьютерной системы извлечения цитат из текстов на естественном языке». Подробные результаты исследований опубликованы в работе [12].

Литература

1. *Кормалев Д. А., Куршев Е. П.* Приложение технологии извлечения информации из текста: теория и практика. // Переяславль-Залесский: ИПС РАН, 2003.
2. *Шереметьева С. О.* Теоретические и методологические проблемы инженерной лингвистики. // М.: ВИНИТИ, 1998.
3. *Лахути Д. Г.* Автоматический анализ естественно-языковых текстов. // М.: ВИНИТИ, 2003.
4. *http://www.aot.ru/docs/fragman.html.*
5. *Липатов А. А., Мальцев А. А.* Методы автоматизации построения и пополнения двуязычных словарей с использованием корпусов параллельных текстов. // Труды международной конференции «Диалог 2006». М.: Изд-во РГГУ, 2006.
6. *Palmer D.* SATZ — An Adaptive Sentence Segmentation System // Report No. UCB/CSD-94–846, Computer Science Division (EECS). University of California: December 2004.
7. *Hertzum M., Lalmas M. and Frokjer E.* How Are Searching and Reading Intertwined during Retrieval from Hierarchically Structured Documents? // INTERACT 2001. Japan, July 2001.
8. *Lalmas M., Reid J. and Hertzum M.* Information-seeking Behaviour in the Context of Structured Documents // ECIR 2003: European conference on IR research No. 25. Pisa, ITALIE: 2002. Vol. 2633, pp. 104–119.
9. *Браславский П., Колычев И.* Автоматическое реферирование веб-документов с учетом запроса. // Интернет-математика 2005. Автоматическая обработка веб-данных. М.: 2005. С. 485–501.
10. *Yang Ch. C., Wang, F. L.* Fractal Summarization for Mobile Devices to Access Large Documents on the Web // Proceedings of the WWW2003. Budapest, Hungary: May 20–24, 2003.
11. *Правила* Дорожного Движения Российской Федерации (в ред. Постановлений Правительства РФ от 08.01.1996 N 3, от 31.10.1998

N 1272, от 21.04.2000 N 370, от 24.01.2001 N 67, от 28.06.2002 N 472, от 07.05.2003 N 265, от 25.09.2003 N 595, от 14.12.2005 N 767, от 28.02.2006 N 109).

12. *Окатьев В. В., Гергель В. П., Алексеев В. Е., Таланов В. А., Баркалов К. А., Скатов Д. С., Ерехинская Т. Н., Котов А. Е., Титова А. С.* Отчет о выполнении НИОКР по теме: «Разработка пилотной версии системы синтаксического анализа русского языка» (инвентарный номер ВНТИЦ 02200803750) // М.: ВНИТЦ, 2008.
13. *http://www.dictum.ru*
14. *Окатьев В. В., Баркалов К. А.* Патент на изобретение «Способ поиска информации», RU 2320005, // М.: Федеральная служба по интеллектуальной собственности, патентам и товарным знакам, 2008. Бюллетень № 8.
15. *http://htmlparser.sourceforge.net/*
16. *http://homepage.mac.com/pauljlucas/software/html_tree/*
17. *http://www.jafsoft.com/detagger/*
18. *http://www.prcyonline.info/totext.html*