Метод порождения правил синтаксической сегментации для ATN

Е.С. Манушкин, Э.С. Клышинский

Аннотация. В статье рассматривается метод автоматической генерации правил для этапа синтаксической сегментации, предваряющего этап синтаксического анализа при автоматической обработке текстов. Для этого используются множества терминалов и пар терминалов, с которых могут начинаться или которыми могут заканчиваться цепочки, выводимые из правил.

Ключевые слова: синтаксическая сегментация, генерация правил, сеть ATN-переходов.

Введение

Данная работа является продолжением статьи [1]. В ней был приведен формализм, позволяющий автоматически порождать правила синтаксической сегментации из грамматики, описывающей предложения на естественном языке и выраженной с помощью расширенных бэкусовских нормальных форм (БНФ).

Напомним, что этап синтаксической сегментации применяется при автоматической обработке текстов (АОТ). Данный этап предшествует этапу синтаксического анализа. Синтаксическая сегментация текста проводится с целью выделения априорной информации о структуре предложения на основе выделения его фрагментов [2] или составных конструкций. Одной из задач синтаксической сегментации является определение границ структурных единиц (фрагментов, сегментов, групп) предложения. Путем анализа входного предложения можно выдвинуть некоторые предположения о том, с какого слова начинаются те или иные синтаксические группы и каким словом они заканчиваются. Вместо того чтобы перебирать все возможные варианты разбора предложения, система синтаксического анализа будет проводить разбор исходя из того, что слова фрагмента принадлежат заданной синтаксической группе. Это позволяет резко сократить количество вариантов разбора входного предложения и существенно уменьшить вычислительные затраты.

В данной работе была поставлена цель показать универсальность метода, предложенного в [1], его применимость для правил синтаксического анализа, представляемых в формах, близких по концепции к контекстно-свободным грамматикам (КС-грамматикам). В качестве примера взяты грамматики расширенных сетей переходов (ATN сетей). Предлагаемый метод проверен на грамматиках русского и английского языка, представленных в форме ATN, полученных из имеющихся у автора грамматик, представленных в форме БНФ.

Обзор методов представления правил синтаксического анализа

Рассмотрим некоторые варианты записи правил синтаксического анализа, применяемые при АОТ.

Link Grammar

Оригинальный подход к синтаксическому анализу английского языка был предложен группой американских исследователей университета Карнеги-Меллон. Проект получил название Link Grammar – грамматика соединений. Разработчики метода отталкивались от свойства проективности, лежащего в основе многих

языков, в том числе и английского: если каждую пару слов из предложения соединить дугами отношений, то эти дуги не пересекаются [3].

Грамматика соединений состоит из набора слов (терминальных символов грамматики), каждому из которых приписан набор соединителей (коннекторов), объединенных формулой. В формуле коннекторов используются следующие связки: оператор дизъюнкции (OR), конъюнкции (&), оператор приоритета (круглые скобки), а так же операторы неограниченного повторения @ (аналог оператора Клини +) и факультативности (фигурные скобки). Каждый коннектор, в свою очередь, имеет тип, обозначаемый заглавной буквой латинского алфавита, и направление ('+' – правосторонний, '-' – левосторонний).

Входная последовательность слов является предложением языка, определенного грамматикой, если пары слов этой последовательности можно соединить дугами, состоящими из коннекторов одинакового типа и противоположных по направлению, удовлетворяющих формуле коннекторов, и при этом выполнены следующие условия (перевод терминов взят из [4]):

1. условие проективности (Planarity): дуги не пересекаются;

2. условие полноты связей: (Connectivity): дуги соединяют все слова во входной последовательности;

3. условие упорядоченности (Ordering): соединяемые дугами слова должны следовать в том порядке, в каком следуют однонаправленные коннекторы в формуле, объединенные знаком конъюнкции (это означает, что операция & несимметрична); например, для терминального символа W: A+ & B+ слово, соединяющееся с коннектором A+, должно быть левее слова, соединяющегося с B+, во входной последовательности;

4. условие исключения (Exclusion): одну пару слов можно соединить только одной дугой.

Приведем простой пример словаря для английского предложения: The cat chased a snake (пример взят из [3]).

Слово	Формула коннекторов
a, the	D+
snake, cat	D- & (O- or S+)
chased	S- & O+

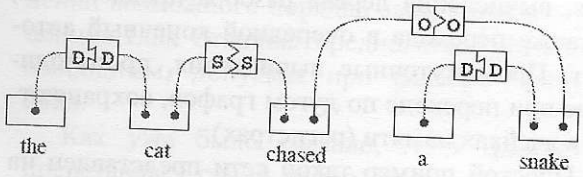

Рис. 1. Пример разбора с использованием LinkGrammar

Разбор этого предложения выглядит, как показано на иллюстрации (Рис. 1):

Чтобы избежать проблемы избыточности, слова разбиваются на классы (для английского языка слова разбиты на 23 класса). Имеются слова-исключения, не входящие ни в один класс, для которых в грамматику добавляются отдельные правила.

Алгоритм анализа предложения основан на подходе динамического программирования [3, 5]. Скорость рекурсивного алгоритма экспоненциально растет при увеличении длины входной последовательности слов. Однако разработчики утверждают, что за счет введения стадии обработки, предшествующей основному алгоритму (Pruning), применения техники memorization, а также использования некоторых других усовершенствований основного алгоритма, разбор предложения происходит за время $O(n^3)$, где n – количество слов в предложении.

Грамматика соединений имеет программную реализацию (Link Parser) с открытыми исходными кодами на языке ANSI C и широко применяется в различных проектах, связанных с анализом текстов.

Расширенные сети переходов (ATN)

Модель расширенных сетей переходов является продолжением идей КС-грамматик и конечных автоматов. Основной разработчик этой модели представления КС- грамматик – Вильям Вудс. ATN сеть представляет собой набор конечных автоматов, представленных в виде графов, в дугах которых содержатся терминальные либо нетерминальные символы КС-грамматики, причем нетерминальные символы реализуют переход в некоторый конечный автомат сети.

Вудс предложил представлять терминальные и нетерминальные символы в виде набора правил, применяемых для проверки параметров согласования слов или фрагментов предложе-

ния, вычисления дерева разбора предложения, а также перехода в очередной конечный автомат. Промежуточные вычисления, производимые при переходе по дугам графов, сохраняются в ячейках памяти (регистрах).

Простой пример такой сети представлен на Рис. 2.

Верхний граф на иллюстрации является стартовым, дуги, помеченные NP и PP, реализуют переход в соответствующий конечный автомат, вершины q4, q5, q7, q8 и q10 – конечные состояния автоматов.

Алгоритм обхода сети предполагает наличие стека. При переходе по нетерминальной дуге, в стек записывается предыдущее положение курсора (т.е. номер графа, номер вершины и номер дуги, по которой производится переход), затем происходит обход соответствующего графа сети. При попадании в конечное состояние одного из графов, из стека достается последнее положение курсора, и переход осуществляется в соответствующую вершину соответствующего графа.

В своей статье [6] Вудс дает формальное описание языка для представления сети в форме расширенной КС-грамматики. В угловых скобках записаны нетерминальные символы, тогда как терминальные символы обозначены большими латинскими буквами. Вертикальной чертой обозначается оператор «или», звездочкой – оператор Клини (Рис. 3).

Терминальные символы (CAT, PUSH, TST, POP, JUMP, TO etc.) обозначают команды, которые производят переход в другой граф, сравнение параметров входного слова предложения, а также построение дерева разбора предложения (подробное описание этих команд и представление ATN сети для предыдущего примера приведены далее).

Из описания языка видно, что запись ATN сети состоит из набора записей вида:

```
(Вершина1 (Набор команд 1) (Набор команд 2) … (Набор команд n)),
```

где `(Набор команд 1)`, `(Набор команд 2)` … `(Набор команд n)` – дуги, исходящие из вершины 1.

Однако это не единственная форма представления ATN сетей, которой пользуются разработчики [7].

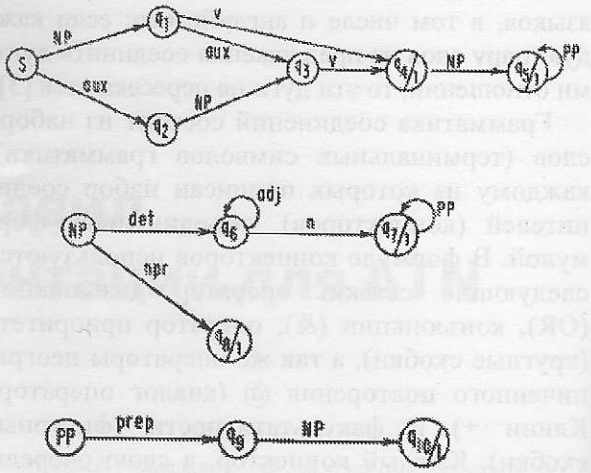

Рис. 2. Пример сети переходов в ATN-грамматике

```
⟨transition network⟩ → (⟨arc set⟩⟨arc set⟩*)
⟨arc set⟩ → (⟨state⟩⟨arc⟩*)
⟨arc⟩ → (CAT ⟨category name⟩⟨test⟩⟨action⟩* ⟨term act⟩)|
        (PUSH ⟨state⟩⟨test⟩⟨action⟩* ⟨term act⟩)|
        (TST ⟨arbitrary label⟩⟨test⟩⟨action⟩* ⟨term act⟩)|
        (POP ⟨form⟩⟨test⟩)
⟨action⟩ → (SETR ⟨register⟩⟨form⟩)|
           (SENDR ⟨register⟩⟨form⟩)|
           (LIFTR ⟨register⟩⟨form⟩)
⟨term act⟩ → (TO ⟨state⟩)|
             (JUMP ⟨state⟩)
⟨form⟩ → (GETR ⟨register⟩)|
         *|
         (GETF ⟨feature⟩)|
         (BUILDQ ⟨fragment⟩⟨register⟩*)|
         (LIST ⟨form⟩*)|
         (APPEND ⟨form⟩⟨form⟩)|
         (QUOTE ⟨arbitrary structure⟩)
```

Рис. 3. Пример записи ATN-грамматики у Вудса [4]

Модель расширенных сетей переходов широко используется в системах автоматической обработки текстов, в том числе и отечественными разработчиками. Например, отечественная компания ПРОМТ утверждает, что их «алгоритмическая основа «translation engines» построена на формализме расширенной сети переходов» (http://www.promt.ru/company/technology/promt/).

Формализм Бэкуса-Наура (БФН)

БНФ является распространенным способом записи правил КС-грамматик, так как хорошо подходит для описания искусственных языков. Добавление механизма согласования дает возможность применять БНФ для записи правил синтаксического анализа естественных языков.

а также делает данную нотацию схожей по идеологии с концепцией расширенных сетей переходов. Введение механизма согласований позволяет расширить грамматику до контекстно-зависимой. Терминальным и нетерминальным символам в этом случае приписывается набор параметров, которые сравниваются с входными словами и с параметрами других символов. Параметры символов являются лексическими характеристиками слов, например, падеж, род, число и так далее. Каждый параметр имеет свой групповой идентификатор и тип, указывающий действия, производимые с параметрами с одинаковым идентификатором, например: сравнить значение параметра с входным словом или присвоить параметру значение, такое же, как у остальных параметров, с тем же идентификатором и т. д.

Для построения дерева зависимостей всем терминалам и нетерминалам из правой части правил приписываются их номера и номера их родительских символов для построения дерева разбора предложения.

Применение метода для нотации ATN

Как было показано выше, для синтаксического анализа текста используются различные подходы, многие из которых не были рассмотрены. Для наших целей их можно разделить на две группы: методы, в той или иной форме основанные на записи грамматик БНФ, и методы, основанные на кардинально иных принципах динамического объединения правил, не позволяющих записать единую грамматику.

Метод генерации правил синтаксической сегментации для грамматики, записанной в явной форме в формате расширенных БНФ, уже был разработан для нотации, используемой в системе машинного перевода Crosslator 2.0 [8, 1]. Как показали исследования, данный метод также может быть применен и к некоторым другим нотациям, так или иначе использующим концепцию КС-грамматик. Мы выбрали нотацию ATN сетей как наиболее близкую по концепции к нотации БНФ. Исследования показали, что метод не может быть напрямую применен к такой записи правил, как, например, Link Grammar. Это связано с тем, что предлагаемый метод основывается на определении возможного порядка слов, тогда как концепция Link Grammar предполагает его вполне свободным, допуская при разборе пропуски слов.

Как уже было сказано, ATN грамматика представляет собой набор графов, дугам которых может быть приписан некоторый алгоритм согласования. Продукции, записанные в БНФ, представляются в виде самостоятельных графов. Для каждой продукции такой граф будет иметь вид транспортной сети всего лишь с двумя вершинами, в которых возможно ветвление, - это начальная и конечная вершины. Это означает, что БНФ грамматики могут быть конвертированы в ATN. Но на практике графы в ATN грамматиках намного сложнее. Например, кроме ветвлений, такие графы могут содержать циклы, в том числе нулевые. Очевидно, такие графы нельзя преобразовать обратно в продукции, записанные в БНФ (хотя вполне возможно, что их можно преобразовать в расширенные БНФ), и, следовательно, предложенные ранее методы вычисления необходимо изменить.

Необходимые теоретические сведения

Наше представление расширенной сети переходов несколько отличается от представления, предложенного Вудсом [6]. Здесь графу приписывается набор параметров, значения которых граф получает после удачного обхода. Дугам приписываются терминальные либо нетерминальные символы в том формате, который был представлен в публикации [1]. Графы будем обозначать буквой G с нижним индексом либо без него, когда граф может быть идентифицирован по умолчанию.

Напомним основные определения из [1], которые нам понадобятся в дальнейшем (запись несколько изменена):

1) Параметр: $\mathbf{p}=\langle n, v\rangle$, где n — имя параметра, а v — его значение, например, <“число”, “множ”>;

2) Входное слово: $\mathbf{a}=\langle s_a, P_a\rangle$, где s – нормальная форма и часть речи входного слова, а $P_a=\{\mathbf{p}\}$ – множество параметров, приписанных к данному слову, например, <“СТЭК”, {<“род”, “муж”>, <“число”, “ед”>,<“падеж”, “род”>}>;

3) Типизированный параметр $\mathbf{p}_t=\langle n, v, t\rangle$, где t – тип параметра, показывающий, какая

операция должна над ним совершаться, например, <“число”, “множ”, согласование>;

4) Терминальный символ a=<s_a, P’$_a$>, где s_a – нормальная форма, P’$_a$={p_t} – множество типизированных параметров терминала a;

5) Нетерминальный символ A=<s_A, P’$_A$>, где s_A – название нетерминала, P’$_A$ - набор типизированных параметров нетерминала A.

Заметим, что к параметрам слова или терминала также может относится параметр, обозначающий его часть речи.

Так же напомним правило сравнения терминала и входного слова. После стандартных сравнений терминала с лексемой, параметры входного слова и терминала сравниваются по следующему правилу:

$$b=\mathbf{a}:\ b=\langle s_b,\ P'_b\rangle,\ \mathbf{a}=\langle s,\ P_a\rangle,\ s_b \sim s_a,\ \forall i\in[1,n]\ \exists j\in[1,m]:\ P'_b(i)=P_a(j) \quad (1)$$

Здесь b – терминал, **a** – входное слово, n и m – количество параметров у b и **a** соответственно, $s_b \sim s_a$ означает, что нормальная форма терминала успешно сравнима с нормальной формой входного слова (нормальная форма терминала может, например, сравниваться с любой нормальной формой в случае, когда мы ищем слово с заданной частью речи и набором параметров), $P'_b(i)=P_a(j)$ означает, что i-й параметр множества P'_b успешно сравним с j-м параметром множества P_a.

При наличии в правиле нескольких символов может потребоваться проверить согласование их параметров. Например, в группе существительного все прилагательные должны согласовываться с главным словом по роду, числу и падежу. Общепринятой точкой зрения является тот факт, что согласовываться могут не только отдельные слова, но и группы [9]. Именно в связи с этим нетерминалам также приписывается множество типизированных параметров. Для проверки согласования заводится множество типизированных параметров, в которое помещаются неповторяющиеся значения параметров всех пройденных в данной продукции символов. При размещении параметра в зависимости от его типа проверяется согласование с уже имеющимися в данном множестве значениями. Если при этом возникает противоречие хотя бы у одного параметра (например, помещается параметр <“падеж”, “им”, согласование>, а в множестве уже хранится значение <“падеж”, “род”>), процесс размещения параметров считается неуспешным и все внесенные на данном шаге изменения откатываются назад. Из этого же множества берутся значения параметров, приписываемые позже нетерминалу при его успешном разборе.

Итак, пусть выбран текущий узел, из которого выходят дуги $c_1, c_2, \ldots, c_k$. Прохождение дуги будет проводиться следующим образом. Если дуге приписан терминал, то он проверяется с текущим словом во входном предложении в соответствии с правилом (I). Если дуге приписан нетерминал, мы пытаемся разобрать приписанное ему правило по имеющемуся графу.

Дуга может быть пройдена, если успешно проверен приписанный ей символ и процесс размещения параметров успешно завершен.

Так как разбор может вестись рекурсивным спуском, пройденная дуга и ее параметры сохраняются в стеке. В этом случае при неуспехе прохождения одной из дуг мы всегда можем откатить изменения и попробовать разобрать другие варианты.

Заметим здесь, что приведенный алгоритм относится лишь к согласованию параметров в графе, но не к разбору входного предложения.

Для автоматической генерации правил синтаксической сегментации, как и в предыдущей работе [1], нам потребуется вычислять множества FIRST, LAST, FIRST2 и LAST2. В теории компиляторов данные множества определяются для цепочек терминальных и нетерминальных символов, входящих в правила грамматики. Элементами указанных множеств являются терминальные символы (в случае FIRST и LAST) либо упорядоченные пары терминальных символов (в случае FIRST2 и LAST2), с которых могут начинаться (FIRST, FIRST2) либо заканчиваться (LAST, LAST2) терминальные цепочки символов, выводимые из правил грамматики.

При переходе к ATN грамматике добавляется новое понятие – граф. Кроме того, в нашем представлении ATN грамматики параметры нетерминала не обязаны совпадать с параметрами графа, на который ссылается этот нетерминал, т.е. нетерминальный символ не всегда совпадает с графом, на который он ссылается. В связи с

этим переопределим множества FIRST, LAST, FIRST2 и LAST2 для ATN грамматики.

Элементами множества FIRST(G) являются терминалы, с которых могут начинаться терминальные цепочки, выводимые из графа G. Элементами множества FIRST2(G) являются упорядоченные пары терминалов, с которых могут начинаться терминальные цепочки, выводимые из графа G. Элементами множеств LAST(G) и LAST2(G) являются, соответственно, терминалы и упорядоченные пары терминалов, на которых могут заканчиваться терминальные цепочки, выводимые из графа G.

Для нетерминалов данные множества вычисляются так же, как для графов, на которые ссылаются эти нетерминалы. Например, если X – нетерминал ATN грамматики, а G_X – граф, на который ссылается этот нетерминал, то $FIRST(X) = FIRST(G_X)$.

Для терминальных символов множества FIRST и LAST будут состоять из единственного элемента – данного терминала, т. е. $FIRST(a)=\{a\}$ и $LAST(a)=\{a\}$.

Здесь мы будем вести рассуждения с учетом согласования символов в грамматике. Поэтому нас будут интересовать модифицированные множества FIRST, LAST, FIRST2 и LAST2. Штрихом обозначим множества, вычисление которых проводится с учетом согласования параметров.

Для определения множества FIRST'(G) следует рассмотреть все дуги графа G, исходящие из его начальной вершины. Пусть α – символ, приписанный такой дуге. Тогда:

- если α является терминалом, то $\alpha \in FIRST'(G)$;
- если α является нетерминалом $A = \langle s_A, P'_A \rangle$, то $\forall b = \langle s_b, P'_b \rangle$, $b \in FIRST(G)$ $\exists a = \langle s_b, P'_a \rangle : a \in FIRST'(G)$ и $P'_a = P'_b \cap P'_A$.

Аналогично определяется множество LAST', для которого рассматриваются дуги, входящие в конечные вершины графа G (здесь мы считаем, что конечных вершин в графе может быть несколько).

Введем новое множество ONLY', содержащее в себе все цепочки длиною в один символ, непосредственно порождаемые данным графом. Данное множество определяется аналогично множествам FIRST' и LAST'. Для его создания рассматриваются отдельные терминалы или дуги, ведущие из начальной в конечные вершины графа.

Для генерации правил синтаксической сегментации, нам понадобится вычисление функций FIRST2 и LAST2 с учетом согласования параметров. Для этого определим множества DirectFIRST2'(G) и DirectLAST2'(G), состоящие из пар терминалов, встречающиеся в терминальных цепочках, которые непосредственно выводятся из графа G.

Рассмотрим все пары дуг такие, что первая дуга из пары c_1 исходит из начальной вершины, а вторая дуга c_2 исходит из вершины, в которую ведет первая дуга. Пусть α_1 и α_2 – символы, приписанные соответственно дугам c_1 и c_2. Тогда

$\forall x = \langle s_x, P'_x \rangle \in ONLY'(\alpha_1)$

$\forall y = \langle s_y, P'_y \rangle \in FIRST'(\alpha_2)\ \exists ab \in DirectFIRST2'(G)$:

$a = \langle s_x, P'_a \rangle$, $b = \langle s_y, P'_b \rangle$, $P'_a = P'_b = P'_x \cap P'_y$.

Аналогично определяется множество DirectLAST2', с той разницей, что c_1 будет обозначать дугу, ведущую в один из конечных узлов графа, а c_2 – дугу, ведущую в вершину, из которой исходит c_1. Здесь нас будут интересовать элементы множеств $ONLY'(\alpha_1)$ и $LAST'(\alpha_2)$.

Пары терминалов, входящие в множества DirectFIRST2' и DirectLAST2', могут встречаться также и в середине цепочек, выводимых из графов грамматики. В этом случае возникает неоднозначность. Для ее разрешения введем множество MIDDLE2'(G) упорядоченных пар терминалов, которые могут встречаться в середине цепочек, выводимых из графа G грамматики. Отсекая пары терминалов, входящие и в DirectFIRST2', DirectLAST2', и в MIDDLE2', мы устраним возникающую неоднозначность.

Опишем алгоритм вычисления множества MIDDLE2'(G).

Для каждой вершины $\mathbf{n}_i$ графа G рассмотрим все пары дуг такие, что первая дуга из пары исходит из рассматриваемой вершины, а вторая исходит из вершины, в которую ведет первая. Положим c_1 и c_2 – дуги одной из таких пар, α_1 и α_2 – символы, приписанные соответственно дугам c_1 и c_2, $\mathbf{n}_i$ – вершина, в которую ведет c_2. Тогда возможны следующие варианты.

1) Если в вершину $\mathbf{n}_i$ ведет хотя бы одна дуга и из вершины $\mathbf{n}_i$ исходит хотя бы одна дуга,

то $\forall x \in LAST'(\alpha_1)$ $\forall y \in FIRST'(\alpha_2)$ $\exists ab \in MIDDLE'(G)$: $a = \langle s_x, P'_a \rangle$, $b = \langle s_y, P'_b \rangle$, $P'_a = P'_b = P'_x \cap P'_y$;

2) Если в вершину n_i ведет хотя бы одна дуга и из вершины n_j не исходит ни одной дуги, то $\forall x \in LAST'(\alpha_1)$ $\forall y \in FIRST'(\alpha_2)$, $y \notin ONLY'(\alpha_2)$ $\exists ab \in MIDDLE'(G)$: $a = \langle s_x, P'_a \rangle$, $b = \langle s_y, P'_b \rangle$, $P'_a = P'_b = P'_x \cap P'_y$;

3) Если в вершину n_i не ведет ни одна дуга и из вершины n_j исходит хотя бы одна дуга, то $\forall x \in LAST'(\alpha_1)$, $x \notin ONLY'(\alpha_1)$ $\forall y \in FIRST'(\alpha_2)$ $\exists ab \in MIDDLE'(G)$: $a = \langle s_x, P'_a \rangle$, $b = \langle s_y, P'_b \rangle$, $P'_a = P'_b = P'_x \cap P'_y$;

4) Если в вершину n_i не ведет ни одна дуга и из вершины n_j не исходит ни одна дуга, то $\forall x \in LAST'(\alpha_1)$, $x \notin ONLY'(\alpha_1)$ $\forall y \in FIRST'(\alpha_2)$, $y \notin ONLY'(\alpha_2)$ $\exists ab \in MIDDLE'(G)$: $a = \langle s_x, P'_a \rangle$, $b = \langle s_y, P'_b \rangle$, $P'_a = P'_b = P'_x \cap P'_y$.

5) Для всех нетерминалов A, которыми помечены дуги, $MIDDLE'(A) \subset MIDDLE'(G)$.

Графически разобранные случаи показаны на Рис. 4.

Определим множества backFIRST2 и backLAST2.

$backFIRST2(ab)=\{G_i\}$: $ab \in DirectFIRST2'(G_i)$, $\forall G_j, \forall G_k$: $ab \notin MIDDLE'(G_j)$, $ab \notin DirectLAST2'(G_k)$.

$backLAST2(ab)=\{G_i\}$: $ab \in DirectLAST2'(G_i)$, $\forall G_j, \forall G_k$: $ab \notin MIDDLE'(G_j)$, $ab \notin DirectFIRST2'(G_k)$.

Эти множества определяют, какие графы могут порождать цепочки символов, начинающиеся (заканчивающиеся) парой символов ***ab***.

При этом, как видно из формул, пары терминалов, входящие в данные множества, встречаются либо только в начале, либо только в конце цепочек, выводимых из графов грамматики.

Метод генерации правил

В генерации правил используются описанные выше множества backFIRST2 и backLAST2. Заметим, что можно задать аналогичные множества для единственного терминала: backFIRST и backLAST, которые также можно использовать в генерации правил.

Будем говорить, что пара терминалов ***ab*** является характеристической, если множество backFIRST2(***ab***) или backLAST2(***ab***) содержит лишь один элемент.

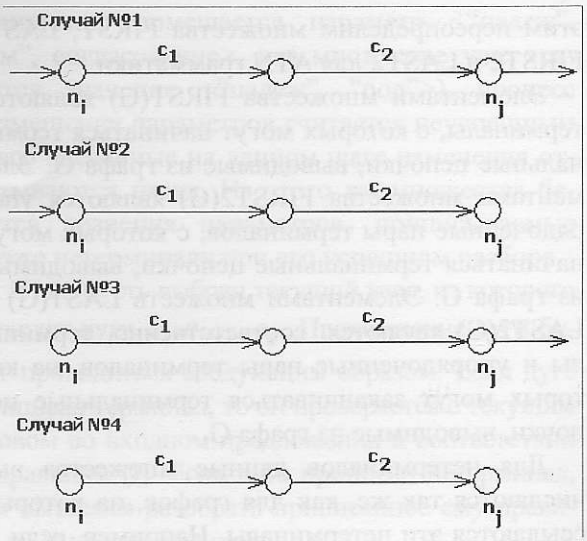

Рис. 4. Различные случаи, возникающие при генерации множества MIDDLE'

Будем говорить, что пара терминалов ***ab*** является характеристической по порогу r, если мощность множества backFIRST2(***ab***) или backLAST2(***ab***) не превышает r.

Если пара терминалов является характеристической, то, встретив ее во входном предложении, мы можем высказать обоснованное предположение, что с данной пары разбор должен вестись по единственному графу либо, соответственно, закачиваться им. Таким образом, весь дальнейший разбор должен вестись так, чтобы к моменту прохождения данной пары мы начали разбор с данного графа (или завершили этой парой разбор данного графа).

На основе этой информации правила синтаксической сегментации строятся следующим образом. Для множеств $backFIRST2(ab)=\{G_i\}$ с мощностью, равной единице, формируются правила синтаксической сегментации вида «если встретилась пара ***ab***, то разбор ведем по графу G_i». Аналогично для множеств $backLAST2(ab)=\{G_i\}$ с мощностью, равной единице, формируются правила вида «если встретилась пара ***ab***, то словами, соответствующими терминалам ***ab***, следует заканчивать разбор по графу G_i». Заметим, что при мощности множеств, превышающей единицу, составляется соответствующее количество правил, однако их применение будет носить комбинаторный характер. При отсутствии ограничений с данной позиции может начаться разбор по не-

скольким десяткам различных графов, тогда как применение правил даст уменьшение их количества до соответствующего порога.

Результаты эксперимента

Для проверки изложенных положений был проведен вычислительный эксперимент. Была разработана программа, которая преобразовывает имеющиеся грамматики, представленные в БНФ, в грамматики ATN, а также вычисляет значение множеств backFIRST2 и backLAST2 для полученных грамматик. На вход программы были поданы реально действующие грамматики русского и английского языков, разработанные для системы машинного перевода «Crosslator 2.0». По результатам вычислений были выделены наборы правил, соответствующие парам терминалов.

Грамматика английского языка, использовавшаяся для экспериментов, содержала 557 графов. В результате генерации множеств backFIRST2 и backLAST2 было обнаружено, что из 5117 пар терминалов, 2148 пар однозначно идентифицируют графы, по которым следует начинать разбор фрагмента текста, и из 445 пар – 153 однозначно идентифицируют графы, по которым следует заканчивать разбор фрагмента. Для русского языка была взята грамматика, состоявшая из 144 графов. При этом из 2073 пар терминалов 123 пары однозначно идентифицируют графы, с которых следует начинать разбор фрагмента, и из 72 пар – 24 однозначно идентифицируют граф, по которому следует заканчивать разбор.

Анализ полученных правил показал, что все корректные пары и в самом деле являются претендентами на правила синтаксической сегментации. Под каждым правилом, приведенным ниже, показан позитивный пример. Формат терминала здесь следующий: [нормальная форма: часть речи; параметры].

Примеры для английской грамматики представлены в Табл. 1. Примеры для русской грамматики представлены в Табл. 2.

Здесь знак «+» означает, что параметр должен согласовываться с другими параметрами вне зависимости от своего значения.

Полученные правила были обработаны вручную и добавлены в базу правил синтаксической сегментации системы «Кросслятор 2.0» (работающей с системой правил в формате

Табл. 1. Примеры правил, полученные для грамматики английского языка

Пара терминалов	Граф	Пример предложения
['AND':conj;]['AFTER':prep;]	CPPDW	..., and after all I didn't really do anything Friday and Saturday
['IN_ORDER_TO':part;] ['HAVE':verb;]	TO_INF2	In order to have something you must give something
['SOME':pronoun;]['OF':prep;]	OBJ_PRON	Some of the most useful things
['SINCE':conj;]['THERE':adv;]	ADV_CL	Since there exists no accepted definition of grammatical English, many somewhat arbitrary choices had to be made
['NO':part;]['SUCH':pronoun;]	S_NP2	No such file or directory

Табл. 2. Примеры правил, полученные для грамматики русского языка

Пара терминалов	Граф	Пример предложения
['КАКОЙ': poss_pron; gender +, number +, case +] [noun; gender +, number +, case +]	IPP1	Кто мог с максимальной точностью сообщить роботу по радио, когда он был в безопасности, а в какой момент ему следовало спрятаться?
['КОТОРЫЙ': poss_pron; gender +, number +, case +] [pers_pron; gender +, number +, case +]	IPP1	Это "Дезинто", который я строю, чтобы приступить к работе
['НЕ_ТОЛЬКО':conj;]['НЕ':particle;]	&O4	... уже полнеющий, с первыми признаками зоба, он не только не женился, но от него исходил какой-то холостяцкий дух, как от некоторых людей будто бы исходит дух святости.
['НАПРИМЕР':parenthesis;][',':sign;]	SWORD	Так, например, ведутся работы по автоматизированному наполнению морфологических словарей за счет анализа несловарной лексики

Табл. 3. Результаты тестирования работы системы

Предложение	Правило	Ускорение, %
Such a small amount of different people think about water-cooling their computer.	['SUCH':pronoun;]['A':article;] начинает SDET4	50
I could not imagine myself that such a small amount of different smart people think about cooling their computer by water pipe.	['SUCH':pronoun;]['A':article;] начинает SDET4	25
There was a very nice girl with a big black dog running to me.	['THERE':adv;][":verb;] начинает THERE	30
Some of my good friends come to me to summer vacation with list of some of duties.	['SOME':adj;]['OF':prep;] начинает SNP_3	20
Little boy comes to read an interest book about big adventures of little mice.	[":verb;]['TO':part;] начинает VG	10

БНФ). Было проведено тестирование работы системы с применением полученных правил и без них, которое дало следующие результаты (Табл.3).

Время синтаксического анализа каждого предложения было получено по результатам нескольких прогонов, так как на время анализа отдельного предложения существенно влияла дискретизация процессов в операционной системе. По этой же причине ускорение усреднялось до «круглых» значений.

Сам процесс сегментации в этом случае пытается применить сгенерированные правила к предложению. При успешном сравнении шаблона в текст добавляется метка, требующая начать разбор с данного места по определенному правилу (задаваемому нетерминалом из списка правил синтаксического анализа).

Ускорение синтаксического анализа достигалось за счет того, что предварительно для каждого правила вычислялось его множество FIRST, состоящее, однако, из нетерминалов, которыми могут начинаться цепочки, выводимые из данного правила. Перед началом разбора каждого правила проверялось, не помечен ли текущий символ меткой правила, полученной в результате работы этапа сегментации. Если такая метка находилась, то проверялось, не входит ли правило, рекомендованное этапом сегментации, в множество FIRST текущего правила. Если правило не находилось, то разбор считался неуспешным, и рассмотрение продукций не производилось. Таким образом, заранее отсекались целые ветви синтаксического анализа, на разбор которых не тратилось время.

Заключение

В данной статье предложен метод автоматического формирования правил синтаксической сегментации на основе формализма ATN-переходов. Принимая во внимание предыдущие работы по этой теме, связанные с обработкой правил в формате расширенных БНФ, можно говорить об общности предложенных решений. Однако метод имеет свои ограничения. Так, например, промежуточные исследования показали неприменимость метода к классу формализмов, подобных Link Grammar. Это связано с тем, что метод требует четкого определения последовательности идущих терминалов и нетерминалов, тогда как Link Grammar предполагает произвольное сцепление отдельных фрагментов, допускающее пропуск частей входного предложения.

Кроме того, предложенный метод всё еще не лишен некоторой неоднозначности, при которой два терминала, обладающие не совпадающими множествами параметров могут одновременно примениться к одному и тому же фрагменту входного предложения. В этом случае может быть выбрано несколько вариантов разбора вместо предполагаемого единственного. Вопрос частично снимается объединением результатов, даваемых терминалами с вложенными множествами, однако для общего случая результаты всё еще не получены.

Несмотря на возможную неоднозначность, метод позволяет автоматически генерировать правила, в случае успеха ускоряющие проведение синтаксического анализа на 10-50%.

Литература

1. Клышинский Э.С., Манушкин Е.С. Метод автоматического порождения правил синтаксической сегментации для задач анализа текстов на естественном языке // Информационные технологии и вычислительные системы 4/2009
2. Кобзарева Т.Ю., Лахути Д.Г., Ножов И.М. Модель сегментации русского предложения // Труды международного семинара Диалог'2001, том 2, Аксаково, 2001
3. Sleator D., Temperley D. Parsing English with a Link Grammar // Carnegie Mellon University Computer Science technical report CMU-CS-91-196, October 1991
4. Ножов И.М. Морфологическая и синтаксическая обработка текста (модели и программы) // internet-публикация диссертации Ножова И.М. «Реализация автоматической синтаксической сегментации русского предложения».
5. Grinberg D., Lafferty J., Sleator D. A Robust Parsing Algorithm For Link Grammars // Proceedings of the Fourth International Workshop on Parsing Technologies, pp. 111-125
6. Woods W. A. Transition Network Grammars for Natural Language Analysis // Communications of the Association for Computing Machinery, vol. 13, no. 10, 1970
7. Лебедев А.С. Механизм семантического поиска авторефератов // Научно-техническая конференция студентов, аспирантов и молодых специалистов МИЭМ. Тезисы докладов. М.: МИЭМ, 2009, с.101-102
8. Жирнов Р.В., Клышинский Э.С., Максимов В.Ю. Модуль фрагментарного анализа в составе системы машинного перевода. Crosslator 2.0 // Вестник ВИНИТИ, 2005 г. НТИ. Серия 2. № 8 сс. 31-33.
9. Тестелец Я.Г. Введение в общий синтаксис / М.: РГГУ, 2001 г. 798 с.

Манушкин Евгений Сергеевич. Аспирант, МГИЭМ. Окончил Московский государственный институт электроники и математики в 2007 году. Автор 5 статей. Область научных интересов: обработка текстов на естественном языке. E-mail: EugeneLebowsky@mail.ru

Клышинский Эдуард Станиславович. Доцент, МИЭМ. Окончил Московский государственный институт электроники и математики в 1997 году. Кандидат технических наук. Автор более 70 статей и 3-х монографий. Область интересов: обработка текстов на естественном языке, искусственный интеллект, многоагентные системы. E-mail: klyshinsky@mail.ru.