Метод автоматического порождения правил синтаксической сегментации для задач анализа текстов на естественном языке

Э.С. Клышинский, Е.С. Манушкин

Аннотация. В статье изложен метод автоматической генерации правил синтаксической сегментации на основе вычисления множеств FIRST, LAST, FIRST2 и LAST2 для формальной грамматики, записанной в виде бэкусовских нормальных форм, предназначенной для синтаксического анализа текстов на естественном языке.

Ключевые слова: автоматическая обработка текстов, синтаксическая сегментация, машинное обучение.

Введение

При классическом подходе к решению задач автоматической обработки текстов (АОТ) путем полного анализа вычислительные затраты растут экспоненциально с ростом длины анализируемых фрагментов текста. Кроме того, создание систем правил для АОТ сопоставимо по сложности с созданием программного обеспечения для таких систем. Количество правил в реальных задачах таково, что в них так же легко запутаться, как в многочисленном и объемном программном коде. В связи с этим в последнее время начали активно развиваться методы, позволяющие автоматически генерировать правила для АОТ. Среди прочего это произошло благодаря тому, что, с одной стороны, в сети Интернет накопилось достаточное количество одноязыковых и параллельных корпусов текстов, другой стороны, само развитие вычислительной техники позволило перейти к срешению подобных задач.

Ярким примером системы, работа которой основана на обработке таких параллельных текстов, может служить система Google Translate (http:\\translate.google.com\). Однако используемая для этой системы генеративная модель, предназначенная для трансфера при машинном переводе, не обеспечивает решения таких задач, как составление рефератов, машинный диалог, анализ содержимого текстов. В связи с этим задача полного анализа текстов на естественном языке сохраняет свою актуальность.

На основе работы с корпусами текстов развиваются статистические методы и для полного анализа. Так, например, ведутся работы по автоматизированному наполнению морфологических словарей за счет анализа несловарной лексики [1]. На основе статистического выделения n-грамм проводится снятие омонимии во входных текстах в ходе предсинтаксического анализа [2]. Однако генерация правил для самого сложного этапа – синтаксического анализа – до сих пор не проводится.

Для ускорения работы синтаксического анализа обычно используется предшествующий ему этап синтаксической сегментации, который выделяет априорную информацию о структуре предложения на основе выделения его фрагментов [3] или составные конструкции (об автоматическом

выделении составных терминов, например, [4,5]). 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, а также 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. Также будут четко очерчены границы синтаксической группы, и система не будет предпринимать неуспешных попыток выйти за эти границы. За счет этого резко сокращается количество вариантов разбора входного предложения и, как результат, существенно уменьшаются вычислительные затраты.

Настоятельная необходимость выделения структуры предложения еще до начала этапа синтаксического анализа становится очевидной при изучении структуры этого предложения. Сама по себе входная информация является неодозначной в связи с явлением лексической омонимии. Так, одна и так же словоформа может представлять собой две формы одного и того же слова («мамы» – единственное число родительный падеж и множественное число именительный падеж), различных слов с одной частью речи («листа» – от «лист/листья» и «лист/листы» или «кошка» как животное с параметром одушевленности или как крюк с параметром неодушевленности), так и слов с различными частями речи («стекло» как глагол или как существительное). Неоднозначность входных данных позволяет говорить о неоднозначности разбора их входных данных. Кроме того, сами правила разбора являются принципиально неоднозначными. В [6] приведен следующий пример: «Привет защитникам города от фашистских оккупантов». С точки зрения синтаксиса данная фраза может быть разобрана двумя способами. С одной стороны, привет передается лицам, защищавшим город от фашистских оккупантов. С другой стороны, привет защитникам города передают сами оккупанты. И если на момент выпуска первого издания книги второе прочтение фразы имело вполне фантастический контекст, то в свете последних политических событий в некоторых странах СНГ подобное прочтение все больше приобретает реальные рамки.

Таким образом, синтаксический анализ текстов на естественном языке является неоднозначным по своей природе, что влечет за собой экспоненциальную скорость роста вычислительных затрат. В связи с этим априорная информация о структуре предложения будет существенно сказываться на производительности системы синтаксического анализа, не давая ей порождать большое количество лишних вариантов.

Если синтаксический анализ предложения ведется с использованием формальной грамматики, записанной в виде бэкусовских нормальных форм (БНФ), то каждой синтаксической группе должно быть сопоставлено некоторое правило, входящее в эту грамматику. Таким образом, задача синтаксической сегментации (в части, определения априорной структуры предложения, анализируемого с использованием БНФ) сводится к сопоставлению слов входного текста правилам формальной грамматики. Следует заметить, что составление в той или иной форме формальной грамматики для задач машинного синтаксического анализа текстов является вполне распространенной практикой, хотя и представляется весьма сложной задачей.

Выделение и написание правил синтаксической сегментации также является трудоемкой и сложной задачей, а организация связей между правилами синтаксического анализа и этапа синтаксической сегментации приводит к дополнительным затруднениям. Автоматическая генерация правил синтаксической сегментации позволит снизить нагрузку на людей, задействованных в создании правил, и повысить качество анализа самих правил.

В данной работе было поставлено целью описание метода автоматизированного порождения правил синтаксической сегментации на основе уже существующей формальной атрибутивной грамматики, записанной в виде БНФ и предназначенной для синтаксического анализа текстов на естественном языке. Построенный метод был проверен на примере имеющейся у авторов формальной грамматики для анализа предложений русского языка.

1. Необходимые теоретические сведения

Будем считать, что нам дана формальная грамматика синтаксического анализа текста на естественном языке, записанная в виде бэкусовских нормальных форм. Для унификации знаки препинания во входном предложении также будем считать словами.

В дальнейших рассуждениях мы будем использовать множества FIRST(α) и LAST(α), определение которых можно найти в [7]. Заметим только, что в данной работе мы будем рассматривать грамматики, не содержащие e-цепочки, в связи с чем вычисление множеств FIRST(α) и LAST(α) значительно упрощается.

Введем новое множество ONLY(α), которое будет содержать множество цепочек, выводимых из α и состоящих из единственного терминального символа.

ONLY(α)={a}, a: $\alpha \Rightarrow^* a$, $a \in$ VT.

Заметим, что если $a \in$ VT, то ONLY(a)={a}.

На основе полученной информации можно рассчитать значения функций FIRST2 и LAST2, показывающих, с каких пар терминалов могут начинаться цепочки, генерируемые при помощи правила (в случае FIRST2), или какими парами терминалов такие цепочки могут заканчиваться (в случае LAST2). При этом будем считать, что грамматика не содержит *e*-символов. В связи с этим алгоритм будет иметь некоторые отличия от алгоритма, применяемого при работе с LL(2) грамматиками [7].

Если α – любая последовательность символов грамматики, то FIRST2(α) – это множество упорядоченных пар терминалов, с которых могут начинаться терминальные цепочки, выводимые из α. FIRST2(α) ={ab : $\alpha \Rightarrow^* ab\beta$}.

Если α – любая последовательность символов грамматики, то LAST2(α) – это множество упорядоченных пар терминалов, которыми могут заканчиваться терминальные цепочки, выводимые из α. LAST2(α)={ab : $\alpha \Rightarrow^* \beta ab$}.

FIRST2 в нашем случае рассчитывается по следующему алгоритму. Пусть рассматривается продукция вида $B \rightarrow \alpha_1\alpha_2\beta$, причем β может являться произвольной (в том числе и пустой) цепочкой символов. В этом случае возможны следующие варианты.

1. Если $b \in$ ONLY(α_1), то $bc \in$ FIRST2(B), где $c \in$ FIRST(α_2).
2. Если $\alpha_1 \in$ VN, то FIRST2(α_1)$\subset$FIRST2(B);

LAST2 вычисляется аналогичным образом, но при этом рассуждения применяются не слева направо, а справа налево к продукциям вида $B \rightarrow \beta\alpha_2\alpha_1$.

Кроме того, введем множества DirectFIRST2 и DirectLAST2, определяемые только по первому пункту приведенного выше алгоритма. Эти множества будут содержать в себе лишь те пары терминалов, которые появляются именно в данном правиле, без «наследования» пар терминалов от других правил.

Легко видеть, что мощности множеств DirectFIRST2 и DirectLAST2 не превосходят мощности множеств FIRST2 и LAST2.

Введем также новое множество MIDDLE2(α), которое будет показывать, какие пары терминалов могут образовываться в середине цепочек, выводимых из α. Пусть $\exists \alpha \Rightarrow^* \beta$: $\beta=\beta_1\beta_2...\beta_n$, $\beta_i \in$ VT, $i \in [1,n]$. Тогда MIDDLE2(α)={ab}, где a и $b \in$ VT: $\exists j \in [2,n-2]$, $\beta_j\beta_{j+1}=ab$. Из определения видно, что значение MIDDLE2 определено только для цепочек, порождающих терминальные цепочки длиною не менее 4 символов. Нас будут интересовать только те пары, которые возникают на стыке символов, за исключением случаев, когда оба символа – терминалы.

Расчет значений, входящих в множество MIDDLE2, будет вестись по следующему алгоритму. Пусть рассматривается продукция вида $B \rightarrow \alpha A_1A_2\beta$, причем α и β могут являться произвольными (в том числе и пустыми) цепочками символов.

Пусть a и $b \in$ VT. Тогда для α и β возможны следующие варианты.

1. Пусть $\alpha=e$ и $\beta\neq e$. Тогда MIDDLE2(α)={*ab*}, $a\in$ LAST(A_1), $a\notin$ ONLY(A_1), $b\in$ FIRST(A_2).
2. Пусть $\alpha\neq e$ и $\beta=e$. Тогда MIDDLE2(α)={*ab*}, $a\in$ LAST(A_1), $b\in$ FIRST(A_2), $b\notin$ ONLY(A_2).
3. Пусть $\alpha=e$ и $\beta=e$. Тогда MIDDLE2(α)={*ab*}, $a\in$ LAST(A_1), $a\notin$ ONLY(A_1), $b\in$ FIRST(A_2), $b\notin$ ONLY(A_2).
4. Пусть $\alpha\neq e$ и $\beta\neq e$. Тогда MIDDLE2(α)={*ab*}, $a\in$ LAST(A_1), $b\in$ FIRST(A_2).
5. MIDDLE2(α) $\cup$ MIDDLE2(A_1) $\cup$ MIDDLE2(A_2) $\cup$ MIDDLE2(β) $\subset$ MIDDLE2(B).

Для ясности изложения будем исследовать результаты на примере. Пусть имеется следующая грамматика:

<SENT>::=<A1>
<A1>::=<A3><A4>|<A4> (1)
<A3>::=[1]|[1]<A3>|[2]<A3>
<A4>::=[2][3]|[2][4]|[4][5][6]

Здесь терминалы ограничиваются квадратными, а нетерминалы – треугольными скобками.

Тогда множества FIRST и LAST для продукций будут следующими:

FIRST(SENT) = {[1],[2],[4]} LAST(SENT) = {[3],[4],[6]}
FIRST(A1) = {[1],[2],[4]} LAST(A1) = {[3],[4],[6]} (2)
FIRST(A3) = {[1],[2]} LAST(A3) = {[1]}
FIRST(A4) = {[2],[4]} LAST(A4) = {[3],[4],[6]}
ONLY(SENT) = {∅}
ONLY(A1) = {∅}
ONLY(A3) = {[1]}
ONLY(A4) = {∅}

На основании вычисления данных множеств можно рассчитать множества FirectFIRST2, DirectLAST2 и MIDDLE.

FirectFIRST2 (SENT) = {∅} DirectLAST2 (SENT) = {∅}
FirectFIRST2 (A1) = {[1][2], [1][4]} DirectLAST2 (A1) = {∅}
FirectFIRST2 (A3) = {[1][1], [1][2], [2][1], [2][2]} DirectLAST2 (A3) = {[1][1], [2][1]}
FirectFIRST2 (A4) = {[2][3], [2][4], [4][5]} DirectLAST2 (A4) = {[2][3], [2][4], [5][6]}
MIDDLE2(SENT) = {∅}
MIDDLE2(A1) = {∅}
MIDDLE2(A3) = {∅}
MIDDLE2(A4) = {∅}

2. Учет согласования слов

Приведенное описание хорошо подходит для формальных языков. Но в естественных языках наблюдается такое явление, как согласование и управление. Так, например, существительное в русском языке управляет своими прилагательными, требуя, чтобы те находились в том же числе, падеже и роде. В каких-то языках (например, в китайском) такое явление отсутствует, в каких-то (как в английском) ощущается слабо, но для таких языков, как русский, учет данного явления обязателен. В связи с этим дополним предложенный метод учетом возможности согласования слов.

Для согласования слов будем использовать параметры. Определим параметр как пару p=<N, V>, где N — имя параметра, а V — его значение. Под именем параметра здесь будем понимать лексические характеристики слов: падеж, род, число и так далее. Для каждой характеристики определен набор значений: падеж может быть именительным, родительным, ... ; число единственным и множественным. Таким образом, каждое слово в тексте будет обладать не только нормальной формой и частью речи, но и набором параметров, указывающих на форму, в которой находится данное вхождение слова. Все эти данные могут быть получены в результате морфологического анализа. Таким образом,

входное слово будет записываться как $\overline{a} =< a, P_w >$, где a – нормальная форма и часть речи входного слова, а $P_w=\{p\}$ – множество параметров, приписанных к данному слову.

В ходе синтаксического анализа параметры могут использоваться в различных ситуациях: нам может быть необходимо проверить значение параметра на предмет совпадения или несовпадения с конкретным значением, проверить согласование параметров между собой, добавить новый параметр к слову и так далее. В связи с этим для терминальных символов введем еще один вид параметров, типизированных, обладающих типом: $p_t=<N, V, T>$, где T – тип параметра. Тип параметра будет определять набор действий, производимых при помощи данного параметра. Например, при поиске объекта действия мы будем искать существительное с параметром «падеж не равен именительному».

Каждому терминалу и нетерминалу будет приписываться некоторое (возможно пустое) множество типизированных параметров. Не нарушая общности рассуждений, разобьем все параметры в зависимости от приписанного им типа на два множества: множество $P_c=\{p_t\}$ параметров, используемых для проверки согласования конструкций, и множество $P_o=\{p_t\}$ остальных параметров. Множество всех параметров символа обозначим как $P_t=P_c \cup P_o$.

Таким образом, терминал будет записываться как $\overline{a} =< a, P_t^a >$, нетерминал запишется как $\overline{A} =< A, P_t^A >$. Здесь a – терминал без добавления к нему параметров, A – нетерминал без добавления к нему параметров, P_t^X – множество параметров для символа X.

При сравнении терминального символа с конкретным словом типизированные параметры терминала сравниваются с параметрами этого слова. Если помимо стандартных операций, используемых при сравнении терминала с лексемой, все сравнения типизированных параметров проведены успешно, то считается, что терминал успешно сравнился со словом. Заметим, что сравнивается полное множество параметров терминала P_t. При этом сравнение множеств параметров проводится следующим образом:

$$\overline{b} = \overline{a} : \overline{b} =< b, P_t^{\overline{b}} >, \overline{a} =< a, P_w^{\overline{a}} >, a \sim b, \forall i =\in [1,n]\ \exists j \in [1,m] : P_{ti}^{\overline{b}} \sim P_{wj}^{\overline{a}}. \quad \text{(I)}$$

Здесь $\overline{b}$ – терминал, $\overline{a}$ – входное слово, n и m – количество параметров у $\overline{b}$ и $\overline{a}$, соответственно, $a \sim b$ означает, что терминал успешно применим к входному слову (без учета согласования параметров), $P_{ti}^{\overline{b}} \sim P_{wj}^{\overline{a}}$ означает, что i-й параметр множества P_t^b успешно сравним с j-м параметром множества P_w^a.

С учетом наличия параметров согласования продукцию можно переписать в виде $\overline{B} \to \overline{\alpha_1} ... \overline{\alpha_n}$, где $\overline{B} =< B, P_c^B >, \overline{\alpha_1} =< \alpha_1, P_c^{\alpha_1} >, ..., \overline{\alpha_n} =< \alpha_n, P_c^{\alpha_n} >$. При проверке согласования значения параметров берутся для терминалов – от соответствующих слов из текста, а для нетерминалов – из результатов разбора успешно примененных продукций.

Для каждой продукции $\overline{B} \to \overline{\alpha_1} ... \overline{\alpha_n}$ можно определить множество имен параметров, используемых для согласования: $C_N = \bigcup_{i=1}^{n} \bigcup_{j=1}^{m_i} N_{ij}$, где m_i – количество параметров согласования у i-го символа продукции, N_{ij} – имя j-го параметра i-го символа продукции. В этом случае можно сказать, что мощность множества, объединяющего значения каждого согласовывающегося параметра, $V_{Ni} = \bigcup_{j=1}^{m_i} V_{ij}$ равна 1, либо она равна 2 и включает в себя значение параметра, согласовывающееся с любым другим значением (в случае, если такой параметр определен).

$$\left|V_{Ni} - 0\right| = 1, \qquad \text{(II)}$$

где 0 – значение параметра, сравнимое с любым другим значением данного параметра.

Таким образом, продукция может считаться успешно сравнившейся, если для каждого терминального символа в ней выполняется условие (I), разбор каждого нетерминала завершен успешно и для продукции в целом выполняется условие (II). Значения параметров для нетерминалов берутся из описанных выше объединенных множеств значений параметров.

В соответствии с вышеизложенным нам необходимо модифицировать и правила определения пар, входящих в FIRST2' или LAST2'. Для этого будем использовать следующий алгоритм.

Определим множество FIRST', состоящее из терминалов, входящих в FIRST($\overline{A}$), с приписанными к ним множествами параметров согласования.

Пусть $\overline{A} =< A, P_c^{\overline{A}} >, \overline{a} =< a, P_c^{\overline{a}} >$, причем $\overline{A} \in VN, \overline{a} \in VT$. Тогда $\overline{a} \in FIRST'(\overline{A})$ при условии, что $\exists \overline{b} =< a, P_c^{\overline{b}} >: \overline{A} \to \overline{b}\overline{\alpha}$, причем $a \in FIRST(A)$ и $P_c^{\overline{a}} = P_c^{\overline{A}} \cap P_c^{\overline{b}}$

То есть множеству $FIRST'(\overline{A})$ принадлежит не сам символ $\overline{b}$, с которого могут начинаться выводимые из $\overline{A}$ цепочки, а его модификация $\overline{a}$, в которой множество параметров определяется как пересечение параметров согласования нетерминала $\overline{A}$ и терминала $\overline{b}$, а остальная информация берется от $\overline{b}$.

Множества LAST' будет определяться аналогично с той разницей, что из $\overline{A}$ будут выводиться цепочки вида $\overline{A} \to \overline{\alpha b}$. Для множества ONLY' выводимые цепочки будут иметь вид $\overline{A} \to \overline{b}$.

Определим теперь множество DirectFIRST2' как множество пар согласуемых терминалов с приписанными к ним типизированными параметрами. Пусть рассматривается продукция вида $\overline{B} \to \overline{\alpha_1 \alpha_2}\beta$, причем β может являться произвольной (в том числе и пустой) цепочкой символов. Пусть $\overline{B} =< B, P_c^{\overline{B}} >, \overline{a} =< a, P_c^{\overline{a}} >, \overline{b} =< b, P_c^{\overline{b}} >$, причем $\overline{a}, \overline{b} \in VT$. Тогда $\overline{a}\overline{b} \in$ DirectFIRST2'($\overline{B}$), если $\overline{a} \in$ ONLY'($\overline{\alpha_1}$), $\overline{b} \in$ FIRST'($\overline{\alpha_2}$) и $P_c^{\overline{ab}} = P_c^{\overline{a}} \cap P_c^{\overline{b}}$. Поскольку нас будет интересовать взаимное согласование символов, из множеств параметров каждого из символов мы можем отбросить те параметры, которые участвуют в согласовании, но присутствуют лишь в одном из терминалов.

Аналогичным образом переопределим множество DirectLAST2' на множестве пар терминалов, содержащих типизированные параметры.

Расчет значений, входящих в множество MIDDLE2', будет вестись по следующему алгоритму. Пусть рассматривается продукция вида $\overline{\mathrm{B}} \to \alpha\overline{\mathrm{A}_1\mathrm{A}_2}\beta$, причем α и β могут являться произвольными (в том числе и пустыми) цепочками символов.

Пусть $\overline{a}, \overline{b} \in VT$. Тогда для α и β возможны следующие варианты.

1. Пусть α=e и β≠e. Тогда $\overline{a}\overline{b} \in$ MIDDLE2'($\overline{\mathrm{B}}$), $\overline{a} \in$ LAST'($\overline{\mathrm{A}_1}$), $\overline{a} \notin$ ONLY'($\overline{\mathrm{A}_1}$), $\overline{b} \in$ FIRST'($\overline{\mathrm{A}_2}$).

2. Пусть α≠e и β=e. Тогда $\overline{a}\overline{b} \in$ MIDDLE2'($\overline{\mathrm{B}}$), $\overline{a} \in$ LAST'($\overline{\mathrm{A}_1}$), $\overline{b} \in$ FIRST'($\overline{\mathrm{A}_2}$), $\overline{b} \notin$ ONLY'($\overline{\mathrm{A}_2}$).

3. Пусть α=e и β=e. Тогда $\overline{a}\overline{b} \in$ MIDDLE2'($\overline{\mathrm{B}}$), $\overline{a} \in$ LAST'($\overline{\mathrm{A}_1}$), $\overline{a} \notin$ ONLY'(A_1), $\overline{b} \in$ FIRST'($\overline{\mathrm{A}_2}$), $\overline{b} \notin$ ONLY'($\overline{\mathrm{A}_2}$).

4. Пусть α≠e и β≠e. Тогда $\overline{a}\overline{b} \in$ MIDDLE2'($\overline{\mathrm{B}}$), $\overline{a} \in$ LAST'($\overline{\mathrm{A}_1}$), $\overline{b} \in$ FIRST'($\overline{\mathrm{A}_2}$).

5. MIDDLE2'(α) ∪ MIDDLE2'($\overline{A_1}$) ∪ MIDDLE2'($\overline{A_2}$) ∪ MIDDLE2'(β) ⊂ MIDDLE2'($\overline{B}$).

При этом для вариантов 1-4 $P_c^{\overline{ab}} = P_c^{\overline{a}} \cap P_c^{\overline{b}}$.

Формально в этом случае возможен вариант, когда одна и та же пара терминалов будет входить в перечисленные множества несколько раз с различными наборами согласовываемых параметров. Это будет происходить в связи с тем, что единичный терминал может порождаться из данного нетерминала более чем одним способом. При этом при порождении различными способами будет формироваться различный набор параметров.

Также определим множества backFIRST2 и backLAST2.

backFIRST2($\overline{a}\overline{b}$)={ $\overline{A}$ }: $\overline{a}\overline{b} \in$ DirectFIRST2'($\overline{A}$), $\forall \overline{B}$: $\overline{a}\overline{b} \notin$ MIDDLE2' ($\overline{B}$)

backLAST2($\overline{a}\overline{b}$)={ $\overline{A}$ }: $\overline{a}\overline{b} \in$ DirectLAST2'($\overline{A}$), $\forall \overline{B}$: $\overline{a}\overline{b} \notin$ MIDDLE2' ($\overline{B}$)

Эти множества определяют, какие нетерминалы могут начинаться или заканчиваться данной парой согласуемых терминалов. При этом считаем, что пара $\overline{a}\overline{b}$ не может образовываться в середине выводимых цепочек за счет сцепления хвоста и головы двух цепочек (для этого в определение функций были введена проверка отсутствия в множестве MIDDLE).

Приведем пример. Пусть грамматика (1) с учетом введения параметров выглядит следующим образом

<SENT>::=<A1>
<A1>::=<A3; p1><A4; p1>|<A4; p1> (3)
<A3; p1>::=[1; p1]|[1; p1]<A3; p1>|[2; p1]<A3; p1>
<A4; p1>::=[2; p1][3]|[2; p1][4]|[4][5; p1][6]

После знака точки с запятой добавлены имена параметров, по которым проводится согласование. Для данной грамматики могут быть получены соответствующие множества.

FIRST'(SENT) = {[1; p1], [2;p1], [4]} LAST'(SENT) = {[3],[4],[6]}
FIRST'(A1) = {[1; p1],[2; p1],[4]} LAST'(A1) = {[3],[4],[6]}
FIRST'(A3) = {[1; p1],[2; p1]} LAST'(A3) = {[1; p1]}
FIRST'(A4) = {[2; p1],[4]} LAST'(A4) = {[3],[4],[6]}

ONLY'(SENT) = {∅} MIDDLE2(SENT) = {∅}
ONLY'(A1) = {∅} MIDDLE2(A1) = {∅}
ONLY'(A3) = {[1; p1]} MIDDLE2(A3) = {∅}
ONLY'(A4) = {∅} MIDDLE2(A4) = {∅}

DirectFIRST2'(SENT) = {[1; p1][1; p1], [1; p1][2; p1], [2; p1][1; p1], [2; p1][2; p1], [2][3], [2][4], [4][5], [1][4]}
DirectFIRST2'(A1) = {[1; p1][1; p1], [1; p1][2; p1], [2; p1][1; p1], [2; p1][2; p1], [2][3], [2][4], [4][5], [1][4]}
DirectFIRST2'(A3) = {[1; p1][1; p1], [1; p1][2; p1], [2; p1][1; p1], [2; p1][2; p1]}
DirectFIRST2'(A4) = {[2][3], [2][4], [4][5]}

DirectLAST2'(SENT) = {[2][3], [2][4], [5][6]}
DirectLAST2'(A1) = {[2][3], [2][4], [5][6]}
DirectLAST2'(A3) = {[1; p1][1; p1], [2; p1][1; p1]}
DirectLAST2'(A4) = {[2][3], [2][4], [5][6]}

Из множеств FIRST2' и LAST2' могут быть получены множества backFIRST2 и backLAST2.

backFIRST2([1; p1][1; p1])={A3} backLAST2([1; p1][1; p1])={A3}
backFIRST2([1; p1][2; p1])={A1, A3} backLAST2([2; p1][1; p1])={A3}
backFIRST2([1][4])={A1} backLAST2([2][3])={A4}

backFIRST2([2; p1][1; p1])={A3}
backFIRST2([2; p1][2; p1])={A3}
backFIRST2([2][3])={A4}
backFIRST2([2][4])={A4}
backFIRST2([4][5])={A4}
backLAST2([2][4])={A4}
backLAST2([5][6])={A4}

Как видно из примера, пары терминалов применяются не к каждому вхождению пар слов в тексте, а лишь при условии согласования этих слов (там, где это необходимо). Пусть запись p1=v2 означает, что параметр с именем p1 принимает значение v2. Пусть на вход поступает следующая цепочка лексем: [2;p1=v2][1;p1=v1][2;p1=v2][4;p2=v4] (согласование первых двух слов не проведено). Тогда, если бы множества backFIRST и BackLAST рассчитывались без учета согласования параметров, то успешно (и ошибочно) было бы определено, что начало фрагмента должно разбираться по правилу A3, тогда как с учетом согласования параметров мы (вполне корректно) не сможем выделить такой информации о структуре входного текста. Однако в ряде случаев (например, [2, 4]) определение может быть проведено вполне успешно и без учета согласования параметров.

Таким образом, введение согласования лексем по параметрам позволяет повысить точность выделения информации о структуре входной цепочки, снижая при этом количество распознаваемых цепочек.

3. Метод генерации правил

Итак, на основании результатов вычисления можно решить обратную задачу: определить, какие правила могут начинаться или заканчиваться данным терминалом или парой терминалов. Для этого были введены множества backFIRST2 и backLAST2. Аналогично можно решить задачу и для единственного терминала, задав множества backFIRST и backLAST.

Будем говорить, что пара терминалов $\overline{a}\,\overline{b}$ является характеристической, если мощность множества backFIRST2($\overline{a}\,\overline{b}$) или backLAST2($\overline{a}\,\overline{b}$) равна единице.

Будем говорить, что пара терминалов $\overline{a}\,\overline{b}$ является характеристической по порогу R, если мощность множества backFIRST2($\overline{a}\,\overline{b}$) или backLAST2($\overline{a}\,\overline{b}$) не превышает R.

Аналогично можно определить и отдельные характеристические терминалы.

Количество характеристических пар терминалов по сравнению с количеством характеристических терминалов должно возрастать. Такой результат может быть получен как исходя из общих соображений, так и в соответствии с тем фактом, что LL(2) и LR(2) грамматики обладают большей выразительной и различительной силой при написании самих правил, хотя и увеличивают объем обрабатываемой информации (например, [8]).

Если пара терминалов является характеристической, то встретив ее во входном потоке мы можем высказать обоснованное предположение, что с данной пары начинается (или заканчивается) единственное правило. Таким образом, весь дальнейший разбор должен вестись так, чтобы к моменту прохождения данной пары мы начали разбор с данного правила (или завершили этой парой разбор данного правила).

На основе этой информации можно построить правила синтаксической сегментации. Для множеств backFIRST2($\overline{a}\,\overline{b}$)={$\overline{A}$} с мощностью, равной единице, можно сформировать правила синтаксической сегментации вида «если встретилось $\overline{a}\,\overline{b}$, то со слова, соответствующего терминалу $\overline{a}$, начинать разбор по правилу $\overline{A}$». Аналогично для множеств backLAST2(A) с мощностью, равной единице, можно сформировать правила вида «если встретилось $\overline{a}\,\overline{b}$, то со слова, соответствующего терминалу $\overline{b}$, заканчивается разбор по правилу $\overline{A}$». Заметим, что при мощности множеств, превышающей единицу, также можно составить соответствующее количество правил, однако их применение будет носить ве-

роятностный характер. Однако даже применение таких правил дает эффект. При отсутствии ограничений с данной позиции может начаться разбор по нескольким десяткам различных правил, тогда как применение правил даст уменьшение их количества до R/.

Применение подобных правил позволяет решить поставленную задачу: еще до этапа синтаксического анализа выделить априорную информацию о структуре текста, поданного на вход системы.

4. Результаты эксперимента

Для проверки изложенных положений нами был проведен вычислительный эксперимент. Была разработана программа, которая вычисляла значения множеств backFIRST2' и backLAST2'. На вход программы были поданы реально действующие грамматики русского и английского языков, разработанные для системы машинного перевода «Crosslator 2.0» [9]. По результатам вычислений были выделены наборы правил, соответствующие терминалам или парам терминалов.

Грамматика английского языка, использовавшаяся для экспериментов, содержала около 450 правил. В результате генерации множеств FIRST2 и LAST2 было обнаружено, что более 900 пар терминалов однозначно идентифицируют начало правила и порядка 100 пар терминалов однозначно идентифицируют окончание правила. Для русского языка была взята грамматика примерно из 130 правил. Для нее было получено более 100 пар в множестве backFIRST2 и всего 13 в множестве backLAST2.

Полученные результаты в основном являлись декартовым произведением небольшого количества конкретных слов, встречающихся рядом. Это существенно снижает вероятность применения правил.

Однако выяснилось, что результаты работы могут служить для проверки корректности грамматики. Так, среди пар терминалов было найдено порядка десятка таких, которые не допускаются реальной грамматикой языка. Например, встретилась пара терминалов, каждый из которых является предлогом. Исследование соответствующих множеств помогло локализовать место, где возникает ошибка.

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: [часть речи; нормальная форма; параметры].

[pn_pers;'WHO';][mod;'MAY';form prf] начинает фрагмент WHO_PHRASE
Jane went to ask who may show us the way.
[part;'NO';][noun;'MATTER';] начинает фрагмент NO_MATTER
no matter
[pn_poss;';][noun;';] начинает фрагмент DEF_OBJECT
his cat
[prep;'OF';][article;';] и [prep;'OF';][part;'NO';] начинают фрагменты S_DET
one of the finest smells

5. Обсуждение

Следует заметить, что терминал в нашей нотации проверяет не всю информацию, приписанную к слову в тексте, а только ее часть. Так, например, проверяется лишь интересующее нас подмножество параметров (один терминал ищет произвольный глагол, а другой – в мужском роде), может не проверяться нормальная форма слова (может искаться как конкретный, так и произвольный предлог). В связи с этим один и тот же терминал может применяться к целому классу слов входного текста. Однако в этом случае справедливо и обратное: несколько различных терминалов могут применяться к одному и тому же слову.

В связи с этим возникает дополнительная неоднозначность: к одному и тому же месту в тексте может быть применено несколько конкурирующих правил. Однако подобная ситуация не так

страшна. Дело в том, что мы в любом случае существенно сокращаем количество вариантов разбора, достигая тем самым поставленной цели. Вместо проверки всех возможных гипотез мы строим лишь несколько. Кроме того, вариативность результатов может быть изначально присуща этапу синтаксической сегментации (так как входной текст сам по себе может быть синтаксически неоднозначен), и нам не придется менять ход синтаксического анализа.

Следует также заметить, что порождаемые правила носят достаточно простой и частный характер. Однако метод гарантирует точность и однозначность их применения в ходе синтаксической сегментации, тогда как ручное создание подобных правил требует тщательной проверки.

Таким образом, разработанный метод может использоваться как дополнение к имеющимся технологиям создания правил для синтаксической сегментации.

Литература

1. Ляшевская О.Н., Кобрицов Б.П., Сичинава Д.В. Автоматизация построения словаря на материале массива несловарных словоформ // Интернет-математика 2007
2. Сокирко А.В., Толдова С.Ю. Сравнение эффективности двух методик снятия лексической и морфологической неоднозначности для русского языка // Международная конференция «Корпусная лингвистика 2004». С.-Пб., 2004
3. Кобзарева Т.Ю., Лахути Д.Г., Ножов И.М. Модель сегментации русского предложения // Труды международного семинара Диалог'2001, том 2, Аксаково, 2001
4. Большакова Е.И., Васильева Н.Э. Терминологическая вариантность и ее учет при автоматической обработке текстов // Труды одиннадцатой национальной конференции по искусственному интеллекту КИИ-2008, том 2, Дубна, 2008
5. Добров Б.В., Лукашевич Н.В., Сыромятников С.В. Формирование базы терминологических словосочетаний по текстам предметной области // Труды пятой всероссийской научной конференции "Электронные библиотеки: Перспективные методы и технологии, электронные коллекции". - 2003, с. 201-210.
6. Попов Э.В. Общение с ЭВМ на естественном языке. Изд. 2 – М.: УРСС. 2004 г. 360 с.
7. Ахо А.В., Сети Р., Ульман Д.Д. Компиляторы: принципы, технологии и инструменты // М.: Вильямс, 2003.
8. Parr T.J., Quong R.W. LL and LR Translator Need k // SIGPLAN Notices, Vol. 31, 1996.
9. 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.

Клышинский Эдуард Станиславович. Доцент, МГТУ им. Н.Э. Баумана. Окончил Московский государственный институт электроники и математики в 1997 году. Кандидат технических работ. Автором более 60 статей и 2 монографий. Область научных интересов: обработка текстов на естественном языке, искусственный интеллект, многоагентные системы. klyshinsky@mail.ru.

Манушкин Евгений Сергеевич. Аспирант, МГИЭМ. Окончил Московский государственный институт электроники и математики в 2007 году. Автор 4 статей и докладов. Область научных интересов: обработка текстов на естественном языке. EugeneLebowsky@mail.ru.