[13] Аджемов А.С., Синева И.С. Метод аналогово-цифровых преобразований на основе сплайн интерполяции // Электросвязь. №2. 1998. С. 37-39.

[14] Рябенький В.С. Введение в вычислительную математику: Учеб. пособие: Для вузов. М.: Физматлит, 1994.

[15] Каханер Д., Моулер К., Нэш С. Численные методы и математическое обеспечение / Пер. с англ. М.: Мир, 1998.

[16] Булычев Ю.Г. Метод опорных интегральных кривых решения задачи Коши для обыкновенных дифференциальных уравнений // Журн. вычисл. матем. и матем. физ. 1988. №10. С. 1482-1490.

[17] Булычев Ю.Г. Методы численно-аналитического интегрирования дифференциальных уравнений // Журн. вычисл. матем. и матем. физ. 1991. №9. С. 1305-1319.

[18] Назаров А.Н. О новом подходе к идентификации битового трафика в широкополосных цифровых сетях интегрального обслуживания // СПб.: Совет безопасности Российской Федерации, Межрегиональная конференция «Информационная безопасность регионов России», 13-15 октября 1999. Тезисы докладов.

# Использование лексических и синтаксических анализаторов в задачах распознавания для естественных языков

А.Б. Холоденко

В статье рассматривается задача коррекции результатов распознавания речи и/или печатных текстов в предположении, что результат распознавания должен быть правильным с точки зрения некоторой фиксированной формальной грамматики. Исследуются случаи регулярных и контекстно-свободных грамматик. Предложены алгоритмы для обработки результатов распознавания и получены оценки их сложности.

## 1. Введение

Как показывают многочисленные эксперименты (см., например, [1]), построение систем распознавания речи и (или) текста без использования лексической и грамматической информации наталкивается на серьезные трудности. Критерием оценки качества таких систем является сравнение результатов их работы с человеческим восприятием. Но даже человек может распознать отдельные звуки лишь с вероятностью 80–85%, хотя при распознавании слитной речи даже в условиях сильного шума надежность ее распознавания человеком приближается к 100%.

В настоящее время основными моделями грамматик, используемых в системах распознавания речи являются статистические грам-

матики, такие как биграммы, триграммы, $n$-граммы и их обобщения (см., например, [2]). Хотя эти модели успешно применяются для европейских языков, их использование для русского языка сталкивается с рядом трудностей. Так, более свободный порядок слов «размывает» статистические закономерности, а большое число словоформ приводит к быстрому разрастанию системы.

В связи с этим более распространенными грамматиками для русского языка стали грамматики, не использующие статистических зависимостей и марковских цепей (см., например, [3]).

В этой работе предложена модель коррекции результатов распознавания, в которой результат работы базового алгоритма распознавания, представленный в виде графа вариантов разбора, анализируется и фильтруется с тем, чтобы оставить лишь те варианты, которые являются правильными с точки зрения некоторой фиксированной формальной грамматики.

## 2. Основные определения и формальная постановка задачи

Дадим вначале несколько формальных определений.

Пусть $A = \{a_1, \ldots, a_r\}$ – фиксированный конечный алфавит.

**Определение 1.** *Обобщенной цепочкой* над алфавитом $A$ назовем связный ориентированный граф без ориентированных циклов, каждому ребру которого приписан некоторый символ из алфавита $A$.

Заметим, во-первых, что в любом ориентированном графе без ориентированных циклов можно занумеровать вершины таким образом, что все ребра будут вести из вершин с меньшим номером в вершины с большим номером.

Во-вторых, в таком графе всегда существует по крайней мере одна вершина, в которую не входит ни одного ребра (такие вершины будем называть начальными) и по крайней мере одна вершина, из которой не выходит ни одного ребра (такие вершины мы будем называть конечными). Доказывается это индукцией по числу вершин



в графе. В решаемой задаче начальная вершина – это начало распознаваемого фрагмента, а конечная вершина – его конец. У распознаваемого фрагмента может быть только одно начало и один конец, поэтому в определении обобщенной цепочки разумно дополнительно потребовать, чтобы в графе содержалась единственная начальная вершина и единственная конечная вершина.

**Определение 2.** Обобщенную цепочку будем называть *нагруженной*, если каждому ребру приписано дополнительно вещественное неотрицательное число. Это число мы будем называть *штрафом* и интерпретировать как штраф, который система должна заплатить, чтобы включить в конечный вариант распознавания именно этот символ.

**Определение 3.** Длиной обобщенной цепочки назовем количество вершин в соответствующем ей графе.

Теперь можно сформулировать общую формальную постановку задачи.

Пусть фиксирована некоторая формальная грамматика Г.

**Задача 1.** Для данной цепочки требуется проверить, существует ли путь, ведущий из начальной вершины в конечную, такой, что соответствующая ему цепочка является конкатенацией выводимых в грамматике Г подцепочек.

**Задача 2.** Для данной цепочки требуется проверить, существует ли путь, ведущий из начальной вершины в конечную такой, что соответствующая ему цепочка выводима в грамматике Г.

Если такой путь существует, то предъявить его (любой, все, с наименьшим штрафом).

## 3. Случай регулярных грамматик

Разберем сначала случай, когда грамматика Г является регулярной. По теореме Клини [4] существует конечный детерминированный автомат, порождающий эту грамматику.

Предположим вначале, что наша обобщенная цепочка линейна. Тогда задача сводится к построению разбиения нашей цепочки на

подцепочки, допускаемые фиксированным конечным детерминированным автоматом.

Пусть автомат $A_D = \{A, Q, Q', \varphi, q_0\}$ реализует грамматику $\Gamma$, $\alpha$ – цепочка символов над алфавитом $A$. Положим $\beta := A_D(\alpha), \beta \in E_2^n$, где $n = len(\alpha)$. $\beta_i = 1 \Longleftrightarrow \varphi([\alpha]_i, q_0) \in Q'$, то есть начало цепочки $\alpha$ длины $i$ является допустимым словом. Заметим, что для вычисления вектора $\beta$ требуется в точности $n$ операций.

Рассмотрим множество $\{\alpha_{k,n}\}_{k=1}^n; \beta_{k,n} := A_D(\alpha_{k,n})$. Введем матрицу $C = (c_{ij})$ следующего вида. Ее $k$-ая строка начинается с $(k-1)$ нуля, после чего стоит вектор $\beta_{k,n}$. Ясно, что $c_{ij} = 1 \Longleftrightarrow$ слово $\alpha_{j,i}$ является допустимым.

Очевидно, что матрицу $C$ можно построить за $\sum\limits_{k=1}^{n}(n-k+1) = \frac{n(n-1)}{2} \approx \frac{n^2}{2}$ операций.

Введем вектора $\vec{k} = (k_1, \ldots, k_{n+1})$ и $\vec{q} = (q_1, \ldots, q_{n+1})$. $k_s = 1$ тогда и только тогда, когда перед $s$-ым символом во входной цепочке можно вставить пробел. Если это можно сделать, то $q_s$ показывает на конец предыдущего слова при таком разбиении. Положим $k_1 := 1$ (перед первым символом всегда можно вставить пробел).

Для $s > 1$: $k_s := 1 \Longleftrightarrow \exists l$ такое, что $k_l = 1$ и $c_{s-1,l} = 1$. В этом случае положим $q_s := l$. Это означает, что перед $s$-ым символом можно вставить пробел тогда и только тогда, когда существует такой номер $l$, что на $l$-ом месте можно вставить пробел, а цепочка $\alpha_{l,s-1}$ допустима.

Очевидно, что цепочка допускает разбиение тогда и только тогда, когда можно вставить пробел после последнего символа, то есть $k_{n+1} = 1$. При этом, двигаясь по вектору $\vec{q}$ из конца в начало, мы получим само разбиение.

Ясно, что для вычисления $k_s$ требуется $s$ операций, а на вычисление всего вектора $\vec{k}$ требуется $\sum\limits_{s=1}^{n+1} s = \frac{n(n+1)}{2} \approx \frac{n^2}{2}$.

Таким образом, общее число операций составляет $n^2$, и мы доказали следующее утверждение.



**Утверждение 1.** *Существует алгоритм, который для любой цепочки за квадратичное относительно длины этой цепочки время строит ее разбиение на подцепочки, допускаемые данной регулярной грамматикой.*

Если цепочка была нагруженной, то в матрице $C$ мы будем хранить штрафы за использование данного слова, а в векторе $\vec{k}$ – величину штрафа, накопленного нами к этому моменту. При этом число $l$ из алгоритма выбирается из условия минимизации штрафа.

Отсутствие разбиения для момента $s$ означает, что $k_s = +\infty$.

Если мы хотим получить все возможные пути, то необходимо хранить не одно значение $l$, а все подходящие значения.

**Замечание 1.** В этом случае выражение «за квадратичное время» нужно понимать в следующем смысле: за квадратичное время можно построить ориентированный граф, являющийся обобщенной цепочкой над алфавитом всевозможных допустимых слов. При этом реальное число путей, содержащееся в этой обобщенной цепочке может быть даже экспоненциально. (Пример: $A = \{a\}; \Gamma = a^*$).

Перейдем теперь к произвольным обобщенным цепочкам.

Построим систему множеств $\{S_i\}_{i=1}^n$, где $n$ – длина обобщенной цепочки.

Положим $S_1 = q_0$. Пусть множества $S_i$ уже построены для всех $i < k$. Пусть $N = \{i_1^{(k)}, \ldots, i_{j_k}^{(k)}\}$ – номера вершин, из которых ребра ведут в вершину $k$; $A_{i_1^{(k)}}, \ldots, A_{i_{j_k}^{(k)}}$ – множества символов, стоящих на этих ребрах. Тогда

$$S_k = \bigcup_{r \in N} \left[ \bigcup_{q \in S_r; a \in A_r} \varphi(q, a) \right].$$

Множество $S_k$ определено корректно, так как все элементы в множестве $N$ строго меньше $k$ (в силу способа нумерации).

Содержательно это означает, что на первом шаге мы находимся в начальном состоянии, а дальше мы из каждой вершины для

всех доступных в ней состояний совершаем все возможные переходы, причем множества состояний для каждой вершины получаются объединением множеств состояний, полученных при переходе в нее из какой-то одной.

Теперь вектор $\vec{\beta}$ определяется следующим образом: $\beta_i = 1 \iff S_i \cap Q' \neq \emptyset$.

Дальнейший ход алгоритма полностью аналогичен рассмотренному выше случаю. Элемент матрицы $c_{i,j}$ кладется равным нулю (бесконечности), если не существует пути, ведущего из $j$-ой вершины в $i$-ую.

Так как из $j$-ой вершины в $k$-ую может вести не более $|A|$ ребер, а число вершин, подлежащих обработке, не превосходит $k$, то для вычисления множества $S_k$ необходимо произвести не более $|A| \cdot |Q| \cdot k$ операций. Таким образом, для вычисления $k$-ой строки матрицы $C$ необходимо затратить $const \cdot k$ операций.

Следовательно, всего алгоритм требует $const \cdot n^2$ операций.

**Замечание 2.** Если язык, задаваемый регулярной грамматикой, конечен, то оценки можно улучшить. Так, например, для построения одного разбиения в случае линейной обобщенной цепочки требуется $const \cdot n$ операций, где $const$ зависит от словаря и не зависит от анализируемой цепочки.

## 4. Случай контекстно-свободных грамматик

Перейдем теперь к случаю контекстно-свободных грамматик. Для этого приведем вначале несколько определений и общеизвестных свойств КС-грамматик (без доказательства).

В дальнейшем терминалы грамматики мы будем обозначать строчными латинскими буквами, нетерминалы – заглавными, а пустую цепочку – $\lambda$.

**Утверждение 2.** *Если язык, порождаемый КС-грамматикой не*



*содержит пустой цепочки, то грамматику можно привести к виду, в котором не будет правил с пустой правой частью.*

**Определение 4.** Вывод цепочки называется *правым*, если в процессе вывода всегда раскрывается самый правый нетерминал в промежуточных цепочках.

**Определение 5.** Для любого вывода существует правый вывод, имеющий то же самое дерево вывода, что и данный.

Известно множество алгоритмов для проверки, является ли данная цепочка выводимой в данной КС-грамматике (см., например, [5]).

Приведем известный алгоритм разбора, который мы будем в дальнейшем обобщать.

В силу утверждения 2 можно считать, что в грамматике нет правил с пустой правой частью.

Введем помимо существующих нетерминальных символов еще два новых специальных символа: $\emptyset$ и $\epsilon$.

Для каждого момента времени $t$ введем множество $S_t$ по следующим правилам:

1) Правило $\emptyset \to .S; 0$, где $S$ – начальный нетерминал грамматики принадлежит множеству $S_0$.
2) Если правило $A \to \tilde{Q}.x\tilde{R}; n$ принадлежит множеству $S_{i-1}$ и $x$ – $i$-ый символ входной цепочки, то правило $A \to \tilde{Q}x.\tilde{R}; n$ принадлежит множеству $S_i$.
3) Если правило $A \to \tilde{Q}.B\tilde{R}; n$ принадлежит множеству $S_i$, то все правила вида $B \to .\tilde{T}; i$ также принадлежат множеству $S_i$, если в грамматике существует правило $B \to \tilde{T}$.
4) Если правило $B \to \tilde{T}.; n$ принадлежит множеству $S_i$ и правило $A \to \tilde{Q}.B\tilde{R}; m$ принадлежит множеству $S_n$, то правило $A \to \tilde{A}B.\tilde{R}; m$ также принадлежит множеству $S_i$.

Правило 2 называют считыванием, а правила 3 и 4 – восполнени-

ем.

Можно доказать, что цепочка выводима тогда и только тогда, когда в множестве $S_n$ есть правило $\emptyset \to S.;0$. В множестве $S_k$ может находиться не более $const \cdot k$ правил. Для вычисления множества $S_k$ (при вычисленных $S_t$; $t < k$) необходимо не более $const \cdot k^2$ операций, а для вычисления всех множеств – не более $const \cdot n^3$ операций.

Для обобщения этого алгоритма сделаем следующее. Будем строить множества $S_i$ в порядке возрастания номеров вершин в графе. При этом индуктивность описания множеств $S_i$ сохраняется, а единственным отличием от основного алгоритма будет являться условие применения правила 1.

Теперь вектор $\vec{\beta}$ строится по правилу $\beta_i = 1 \iff$ множество $S_i$ содержит правило $\emptyset \to S.;0$. Дальнейшие действия полностью повторяют алгоритм для регулярного случая.

Для построения самого разбиения необходимо хранить также информацию о том, как именно было получено то или иное правило. Эта информация никак не используется на этапе построения множеств $S_i$ и не изменяет количество операций.

Таким образом, получаем общий результат для случая контекстно-свободных грамматик: задача коррекции может быть решена за время порядка $const \cdot n^3$ для задачи проверки правильности обобщенной цепочки; время порядка $const \cdot n^4$ для задачи проверки возможности разбиения (случай построения одного разбиения) и $const \cdot n^5$(случай всех возможных разбиений).

## 5. Заключение

В свете всего вышеизложенного можно сделать вывод, что предложенная модель может быть использована для коррекции результатов распознавания для тех ситуаций, когда грамматика и словарь могут быть полностью описаны заранее, например, в системах с ограниченной предметной областью.

Описанные в работе алгоритмы были реализованы в программе, предназначенной для распознавания сильно испорченных текстов,



причем качество распознавания текста программой превысило качество распознавания текста человеком.

Работа выполнена на кафедре математической теории интеллектуальных систем механико-математического факультета МГУ под руководством д.ф.-м.н. Д.Н. Бабина.

## Список литературы

[1] Kosarev Yu., Machovikova I., Machovikov A., Piotrowski R., Tseitlin S. Language Perception Modeling Based on Analysis of Children's Speech // Speach and computer. St.-Peterburg, Russia, 26-28 October 1998.

[2] Zitouni I., Smaili K., Haton J.-P. Variable-length Class Sequences Based on a Hieratchical Approach: MCnv // Speach and computer. St.-Peterburg, Russia, 26-28 October 1998.

[3] Zharkov I. Segmentation of continuous speech with stress as a word-forming element // Speach and computer. St.-Peterburg, Russia, 26-28 October 1998.

[4] Кудрявцев В.Б., Алешин С.В., Подколзин А.С. Введение в теорию автоматов. М.: Наука, 1985.

[5] Ахо А., Ульман Д. Теория синтаксического анализа, перевода и компиляции. М.: Мир, 1978. Т. 1,2.