

## ПРОБЛЕМЫ АВТОМАТИЧЕСКОЙ КОРРЕКЦИИ ТЕКСТОВ НА ФЛЕКТИВНЫХ ЯЗЫКАХ

*И. А. Большаков*

### § 1. Введение

Программы автоматического обнаружения и исправления ошибок в текстах на естественных языках (назовем их автокорректорами — АК, хотя терминология еще не сложилась) получают все большее распространение.

Говоря точнее, АК производят автоматически лишь обнаружение ошибок и то лишь орфографического типа, а собственно коррекция ведется обычно при участии человека. Но и при таком ограниченном автоматизме точность и производительность выверки текстов с помощью АК повышаются столь значительно, что АК становятся неотъемлемой частью процесса подготовки разнообразных деловых документов, статей, рефератов, книг [6, 99, 105, 124, 133]. Быстрое же развитие персональных компьютеров позволяет предсказать распространение АК в будущем в миллионах копий, ибо в них нуждается каждый научный работник, инженер, студент, делопроизводитель.

Не составляет исключения и сфера научно-технической информации (НТИ). Каждый центр — генератор НТИ должен иметь АК для ликвидации ошибок в постоянно растущих документальных и фактографических базах данных [6], и такое средство следует предоставить каждому нуждающемуся в нем сотруднику.

Западные фирмы поставляют десятки различных АК, постоянно упоминаемых, например, в журналах «Byte», «Personal Computing» и др. В подавляющем большинстве эти АК ориентированы на английский язык (ср. немногие работы по другим европейским языкам [57, 119]). Английский обладает, как известно, низкой флективностью: одна английская основа позволяет сформировать лишь немного различных словоизменительных форм (словоформ). Так, у английских глаголов и существительных не более четырех разных форм, у прилагательных — не более трех. В этих условиях на худой конец

можно хранить в АК различные словоформы языка по-отдельности, не прибегая к морфологическому анализу.

В высокофлективных языках, к которым относятся, в частности, все славянские, от одной основы могут образовываться до нескольких сот различных словоформ (см. славянские глаголы или венгерские существительные). В этих условиях в АК неизбежны средства морфологического анализа той или иной сложности, а непосредственное использование западных АК и перенос методов их работы на неанглоязычные тексты едва ли даст удовлетворительные результаты, если исключить метод «пробой силы» — неограниченное наращивание объема оперативной памяти (ОП) и быстройдействия ЭВМ — в массе пока еще недоступный и в западных странах.

Очевидно, что у АК флективных языков, которые, например, для русского и польского языков уже появились [6, 9, 116], необходимо, с одной стороны, сохранить все достоинства англоязычных программных средств, а с другой — обеспечить удобные методы морфологического анализа и компактного хранения более сложных словарей.

Трудности создания новых неанглоязычных АК не только в том, что им нужны морфологические анализаторы и новые словари, но и в том, что сама структура существующих АК известна плохо. После превращения АК в коммерческий продукт западная научно-техническая литература стала избегать описаний внутренней их структуры, приводя лишь чисто внешние и пользовательские их параметры — минимальный объем ОП, необходимую операционную среду и т. п. При этом сообщаемый обычно параметр — объем словаря — не всегда можно расшифровать, дается ли он в лексемах или словоформах, а для английского языка два эти способа измерения дают не менее чем двойное различие.

Настоящая работа анализирует основные свойства современных АК естественных языков с учетом как их пользовательских возможностей, так и внутренней структуры, как черт, не зависящих от языка, так и черт, тесно связанных с языковой флективностью. Тексты на языках программирования (исходные коды программ) и на иных искусственных языках нами не рассматриваются: их автоматическая коррекция существенно проще.

Описывая АК в целом, мы привлекаем известные характеристики англоязычных образцов, а в части АК для русского языка опираемся на отечественный, особенно — личный опыт [9]. Но нами не делается попыток дать полную картину или проследить историю проблематики. Скорее, мы ставим задачу написать дополнение к обзорам, содержащимся в [102], в монографии Питерсона [100] (похоже, единственной, которая содержит полный листинг программы функционирующего АК) и в [2]. Там, где сведения отсутствуют, мы приводим наши

приближенные оценки и суждения, пусть несколько субъективные. При выборе подтемы мы наверняка были субъективны. В частности, почти не затрагивалась проблематика подбора кандидатов на замену искаженного слова, но подробно изучается проблема сжатого хранения словаря.

Что же касается немногочисленных лингвистических примеров, то для русского языка автор эксплуатировал знание его как родного, а когда осмеливался на примеры из других языков, сознавал ограниченность своей языковой компетенции. Впрочем, возможная непредставительность наших лингвистических примеров едва ли влияет на основные положения, справедливые для широкого круга флективных языков. Из европейских лишь немецкий язык ввиду свободного формирования в нем многокоренных слов явно не покрывается нашим анализом.

## § 2. Методы обнаружения ошибок

Известно, по крайней мере, три метода автоматизированного обнаружения орфографических ошибок в текстах — статистический, полиграммный и словарный [99].

При статистическом методе из текста одна за другой выделяются составляющие его словоформы, а их перечень по ходу проверки упорядочивается согласно частоте встречаемости. По завершении просмотра текста упорядоченный перечень предъявляется человеку для контроля, например, через экран дисплея. Орфографические ошибки и описки в сколь-нибудь грамотном тексте несистематичны и редки, так что искаженные ими слова оказываются где-то в конце перечня. Заметив их здесь, контролирующее лицо может автоматизированно найти их в тексте (например, с помощью программы текстового редактора) и исправить.

При полиграммном методе [99] все встречающиеся в тексте двух- или трехбуквенные сочетания (биграммы и триграммы) проверяются по таблице их допустимости в данном естественном языке. Если в словоформе не содержится недопустимых полиграмм, она считается правильной, а иначе — сомнительной, и тогда предъявляется человеку для визуального контроля и, если нужно, исправления.

При словарном методе все входящие в текст словоформы, после упорядочения или без него, в своем исходном текстовом виде или после морфологического анализа, сравниваются с содержимым заранее составленного машинного словаря. Если словарь такую словоформу допускает, она считается правильной, а иначе предъявляется контролеру. Он может оставить слово, как есть; оставить его и вставить в словарь, так что далее в сеансе подобное слово будет опознаваться системой без замечаний; заменить (исправить) слово в данном месте;

потребовать подобных замен по всему дальнейшему тексту; отредактировать слово вместе с его окружением. Операции над сомнительным участком текста, указанные или иные возможные, могут комбинироваться исходя из замысла проектировщика АК.

Результаты неоднократных исследований [99] показали, что только словарный метод и экономит труд человека и ведет к минимуму ошибочных действий обоих родов — пропуска текстовых ошибок, с одной стороны, и отнесения правильных слов к сомнительным, с другой. Поэтому словарный метод стал доминирующим, хотя полиграммный метод иногда и применяются как вспомогательный.

### § 3. Автоматизация процесса исправления

Можно предложить три степени автоматизации процесса коррекции текста:

- 1) только обнаружение ошибок,
- 2) обнаружение их и выдвижение гипотез (альтернатив, кандидатов) по исправлению,
- 3) обнаружение ошибок, выдвижение гипотез и принятие одной из них (если хотя бы одна выдвинута системой) в качестве автоматически вносимого исправления.

Без первой степени АК немислим.

Вторая и третья степень возможны только при словарном методе. Уже вторая существенно облегчает внесение исправлений, ибо в большинстве случаев исключает перенабор сомнительного слова. Особенно полезны найденные альтернативы, когда контролирующее текст лицо нетвердо знает данный естественный язык или конкретную терминологическую область. Однако выдвижение гипотез требует больших переборов с поиском по словарю. Поэтому современные АК часто имеют средство выдвижения гипотез лишь в качестве факультативного, запускаемого, если требуется, избирательно для данного сомнительного слова.

Третья степень автоматизации заманчива и одновременно опасна. Заманчивость заключается в полной автоматизации процесса исправления. Опасность же в том, что ни один словарь, в том числе — заключенный в человеческом мозгу, никогда не бывает исчерпывающе полным. Когда незнакомое слово встречает система, основанная на неполном словаре, она может «исправить» его на ближайшее ей знакомое, порой резко искажив исходный смысл текста. Особо опасно править собственные имена лиц, фирм, изделий. Заманчиво уметь пропускать (обходить) собственные имена и сугубо специальные

термины, априори полагая их правильными, но безошибочные способы обхода, особенно — терминов, нам не известны.

Чисто автоматическому исправлению мог бы способствовать автоматический синтаксический и семантический анализ проверяемого текста, но он еще не стал принадлежностью обычных АК. И даже при его наличии лишь человек сможет диагностировать быстро меняющиеся совокупности собственных имен, терминов и аббревиатур, а также окказионализмы — случайно появляющиеся словесные новации.

В связи со сказанным полная автоматизация исправлений может применяться лишь в любом из следующих ограничительных условий:

1) Текст имеет вид перечня терминов и терминологических словосочетаний в стандартной их форме, так что в АК достаточно иметь словарь, замкнутый по объему и проблематике. При этом все термины между собой «непохожи» (например, в словаре нет одновременно АДСОРБЦИЯ и АБСОРБЦИЯ).

2) Ошибки носят характер замены кодов исходных букв на коды литер, совпадающих или близких к исходным по начертанию. Например, заменяются коды КОИ (советский аналог кодов ASCII) русских букв А, В, С, Е, У на коды латинских букв A, B, C, E, Y; латинские буквы l и O — на цифры 1 и 0 и т. п. Сюда же отнесем повторы одной и той же литеры, возникающие из-за продленного нажима клавиши дисплея или его неисправности. В подавляющем большинстве, если в словоформе более 2—3 букв, такие исправления абсолютно правильны.

Можно применять автоматическое исправление и тогда, когда текст содержит лишь минимум личных имен, и в системе предусмотрен постконтроль автоматически исправленных мест человеком. Чтобы в последнем случае (с постконтролем или без него) уменьшить число неверных исправлений, можно привлечь результаты исследований типовых ошибок, допускаемых человеком при вводе текстов в машину. Характер этих ошибок существенно зависит от конкретного языка.

В английском языке, где орфография зиждется на этимологическом (историческом) принципе и расхождение между звуковым и письменным обликом слов велико, машинистки и операторы часто совершают школьно-орфографические ошибки, имитируя сходное звучание неверными буквами [130, 131]. Поэтому в процессе исправления здесь в первую очередь целесообразны пробные шаги, сохраняющие «звучание» буквенных цепочек. Основанный на этих соображениях способ исправления назван в [131] интеллектуальным (intelligent), но лучше сохранить этот термин для систем будущего, включающих полный анализ текста.

В славянских языках письмо базируется на принципах,

близких к фонематическому (отдельные буквы отражают фонемы современного языка), и пишущему гораздо легче увязывать звуки и фиксирующие их на письме буквы. Поэтому машинисткам легче избегать школьно-орфографических ошибок [13, 27, 31]. Как показано в [13], на первое место в русском языке выдвигаются недостатки профессиональной подготовки машинисток — неточные или несинхронизованные между руками нажимы клавиш, пропуски и лишние нажимы. Выявлен характер этих погрешностей, и на этой основе предложено ранжировать гипотезы исправления, внешне равновозможные, по их статистическому правдоподобию. Для этого предложена клавиатурная модель типовых ошибок, прямо зависящая от расположения клавиш на стандартной (для данной страны) клавиатуре и от привычной привязки к зонам клавиш разных пальцев машинистки (оператора).

#### § 4. Орфография, синтаксис, стилистика

Практически все современные АК обнаруживают лишь орфографические ошибки, включая описки при вводе данных. Из орфографических ошибок не обнаруживаются те, которые переводят одну правильную словоформу в другую правильную, но, возможно, неуместную в данном контексте. Таковой может быть, например, замена одного окончания русского существительного другим, тоже верным.

Системы, пытающиеся находить ошибки и иных родов, носят пока чисто экспериментальный характер, время промышленного их применения еще не пришло [67, 68, 82, 91]. Это объясняется тем, что пока отсутствуют быстрые и эффективные синтаксические анализаторы естественных языков, рассчитанные на открытые тексты, т. е. не имеющие ограничений лексического и синтаксического характера.

Среди ошибок неорфографического характера важен еще класс стилистических ошибок. Если словарь, придаваемый АК, снабдить стилистическими пометами, а отдельные слова в нем связать, где необходимо, синонимическими отсылками, то слова проверяемого текста, не соответствующие его жанру, можно обнаружить и заменить на допустимые синонимы. Такие системы называют системами с тезаурусом, но из широко известных тезаурусных отношений в них реализуется лишь синонимия.

Неверными словоупотреблениями стилистические ошибки, однако, не исчерпываются. Возможны синтаксические обороты, допустимые в языке в целом, но не в жанре данного текста. Современные АК обычно не умеют их обнаруживать, это еще сложнее, чем обычный синтаксический анализ.

## § 5. Влияние класса компьютера. Диалоговый и пакетный режимы

Автокорректоры, особенно в прошлом, как правило, не были универсальны в отношении класса ЭВМ, будучи ориентированы либо на большую, либо на малую (мини-, микро-) машину. Такое различие в данном случае важно, ибо только большие машины могут вместить в ОП практически неограниченные словари и средства ускоренного формирования гипотез замены. К счастью, прогресс вычислительной техники быстро повышает конкурентоспособность малых машин.

Существенно влияет класс базовой машины и на режим работы АК. Упрощенно можно представить смену характерных режимов работы АК по мере развития компьютеров следующим образом.

Когда строились первые АК, были доступны лишь большие машины, как правило, с пакетным режимом работы. Выбор режима был, по существу, детерминирован, и создателю АК необходимо было лишь обеспечить высокую эффективность в заданных рамках. Для этого брался достаточно большой проверяемый текст (в десятки и сотни тысяч знаков), его слова при просмотре сводились воедино и упорядочивались, а затем одним проходом сравнивались с упорядоченным по тем же правилам словарем [5]. Это давало минимальные затраты машинного времени на единицу текста. Однако пользователю приходилось ждать полного результата проверки, а исправление автоматически обнаруженных ошибок оказывалось довольно трудоемким, ибо протокол исправления ошибок приходилось составлять, вводить в ЭВМ и обрабатывать в пакете отдельно.

По мере распространения миникомпьютеров и ускорения их центральных процессоров стал популярным диалоговый режим работы. В условиях, когда машина снабжена жестким дисковым накопителем емкостью хотя бы в несколько мегабайт, оказалось необременительным проверять тексты слово за словом. Для каждого из них, сколько раз они бы в тексте ни встречались, производилось обращение к словарям, расположенным в ОП или на диске. Удельные затраты машинных ресурсов оказывались несколько выше, зато пользователю предоставлялось возможность снять очередное затруднение, как только оно возникало, так что суммарное время обработки текста могло даже снизиться.

Быстро прогрессирующие микрокомпьютеры были снабжены сначала оперативной памятью 64—128 Кбайт и только гибкими (флоппи-) дисками, малоемкими и медленными. Потребовались существенные усилия по созданию придаваемых АК словарей, а режим работы пришлось сменить полностью. Даже если обращаться к флоппи-диску в среднем за каждым вторым словом, храня каждое первое в ОП, скорость проверки

абсолютно правильного текста не могла достигать и 10 слов/с. При этом должен постоянно перегружаться не всегда надежный, хрупкий дисконвод.

Большинство АК были переведены в этих условиях в полупакетный режим. Очередной, достаточно большой фрагмент проверяемого текста заносился в ОП, из него создавался словоуказатель с отсылками внутрь обрабатываемого фрагмента, а затем по указателю производилось сравнение со словарем, считываемым с флоппи-диска тоже по кускам. Ненайденные словоформы либо как-то отмечались в исходном тексте, либо запоминались отдельно в виде своих адресов. Подобная проверка велась до конца проверяемого файла без вмешательства человека. Далее файл вызывался снова и предъявлялся для контроля тех строк, где были замечены сомнительные слова.

Нетрудно доказать, что наивысшая скорость проверки в описанном режиме достигается, когда отводимые для очередных порций текста и словаря участки ОП равны, а суммарный их объем (равный емкости ОП за вычетом объема всех программ и резидентной части операционной системы) максимально возможный. Время ожидания проверки текста в несколько тысяч знаков не превышает в таком режиме минуты, и это казалось приемлемым в сопоставлении с рядом иных сложных программных средств на тех же микрокомпьютерах (например, с СУБД).

По мере увеличения ОП микрокомпьютеров до 640 и более Кбайт и их оснащения жесткими дисками типа «Винчестер» наблюдается возврат к чисто диалоговому режиму, ибо из-за существенного уменьшения времени обращения к ОП и диску этот режим снова оказывается эргономичным. Спираль смены режимов АК завершает свой второй оборот.

Уместно добавить, что в АК с диалогом практически всегда возможна и пакетная работа. Пользователь заранее готовит пакетное задание (косвенный файл, каталогизированную процедуру или макрос — название зависит от операционной системы), где указывается режим работы и обрабатываемый файл. Запуск задания осуществляется самим пользователем либо операционной системой — в порядке пакетной очереди или по таймеру. В зависимости от типа запуска и наличия на установке работ, параллельных автокоррекции, говорят о собственно пакете или работе на заднем плане.

После запуска система проверяет указанный файл, обнаруживает в нем сомнительные места и заносит их вместе со строками контекста в отдельный протокольный файл. Запросов к пользователю при этом не вырабатывается, исправлений не вносится, и словарь не пополняется.

Пакетная работа без участия оператора может быть и более сложной, включая множество команд, характерных для

диалога. Нужно лишь заранее предусмотреть исходы операций с тем, чтобы система нормально всех их завершала. Имеется в виду, например, самообучение по орфографически безупречным текстам, замена некоторых слов (сокращений — на их полные варианты, иностранных слов — на их транслитерацию или перевод) по отдельному словарю подстановок. Но нельзя чисто автоматически и всегда безошибочно совершать исправления или иные операции, число, место и характер которых заранее неизвестны.

Если АК задуман сравнительно давно, то непосредственно перевести его в диалоговый режим нелегко, но и в своем пакетном варианте, при постоянно совершенствующихся словаре и алгоритме работы с ним, АК продолжает быть высокоэффективным техническим средством.

### § 6. Комплексирование с текстовыми редакторами и форматизаторами

Сложившаяся привычная последовательность подготовки текстов на вычислительной установке включает ввод текста с помощью текстового редактора, автокоррекцию, форматирование и вывод на печатающее устройство (или фотонаборный автомат) [60, 88]. Большие преимущества имела бы интегрированная система, охватывающая все эти функции и уместяющаяся со всеми программами и вспомогательными файлами в ОП. Созданные на единых принципах, компоненты такой системы могли бы ориентироваться на единую операционную среду, тип дисплея, формат вспомогательных файлов и др. Но объем ОП у микрокомпьютеров до последнего времени был недостаточным, а постоянная смена программ и данных в ОП с помощью так называемых оверлеев существенно снижала скорость работы. Поэтому еще не потеряло актуальность комплексирование АК с автономными программами, выполняющими иные этапы подготовки текста, а также придание АК отдельных функций этих программ.

Запускаемый для проверки готовых текстов АК остро нуждается в операциях внутрискрипного редактирования. Действительно, по некоторым наблюдениям, до 10% всех ошибок в текстах имеют характер разрыва или слияния правильных слов. Исправления таких погрешностей операциями замены принципиально возможны, но громоздки. Можно, конечно, выдвигать гипотезы по склеиванию или расщеплению слов. При наличии же средств внутрискрипного редактирования достаточно подвести курсор в нужное место строки и однократно нажать клавишу раздвига или сдвига литер. Только текстовым редактированием обычно исправляются сложные (не однобуквенные) ошибки.

Когда же проверяются тексты, вводимые пользователем в самом сеансе диалоговой работы, процессы текстового редактирования и автокоррекции сливаются полностью. Многие западные АК сопрягаются с текстовыми редакторами тех же фирм. Текстовый редактор при этом всегда может работать автономно от АК, а АК — только в особой, автономной его версии.

Однако сохранить высшие качества и текстового редактирования (просмотр подготавливаемого текста вперед и назад, поиск подстроки по всему тексту и др.), и автокоррекции довольно трудно как с программной, так и с пользовательской точек зрения. Так, просмотры в обоих направлениях требуют хранения в ОП всего обрабатываемого файла или крупного его фрагмента, а это ведет к нехватке ОП для прочих задач. Перемещение по тексту относительно точки, до которой уже доведена автокоррекция, требуют уточнения дальнейшей ее стратегии коррекции, обычно строго последовательной. Легче всего сочетать предельно высококачественный диалоговый АК с построчным текстовым редактором, когда точка наблюдения текста может по желанию пользователя произвольно перемещаться внутри строки, но номера строк могут лишь нарастать. Очередная введенная или заранее подготовленная строка текста сразу проверяется средствами АК и, если нужно, подвергается доводке автокорректором или вручную.

Форматирование, т. е. завершающее квазиполиграфическое оформление документов, сложилось независимо от текстового редактирования и автокоррекции еще в 70-х годах [60]. Минимально оно включало выравнивание правого края строк, стандартизацию оформления абзацев, центровку заголовков, представление номеров страниц.

По мере роста требований к полиграфическому оформлению возникла необходимость переноса слов на границе строк, использование различных шрифтовых гарнитур (курсив, полужирный шрифт и т. п.) включая буквы разной ширины, многоколоночную печать и, как предел, — сложные математические формулы и таблицы.

Технология машинной форматизации обычно включала разметку текста за дисплеем — вставление в подготовленный текст специальных командных строк (визуально выделяющихся, например, точкой на первой позиции), а иногда — наборов спецзнаков между отдельными словами и даже между буквами отдельных слов.

Конечно, лучше всего форматизовать уже полностью готовый, выверенный текст. Если же рассчитывать и на те ситуации, когда текст может правиться и после форматизации, то программа автокорректора существенно усложняется, а время ожидания пользователя за дисплеем — несколько увеличивается.

В практике подготовки деловых документов возможны, однако, случаи, когда требования к внешнему оформлению сравнительно невысоки и сводятся лишь к указанному ранее минимуму. В этих случаях можно вообще не вносить никаких форматизирующих помет, приняв необременительные соглашения о способе ввода текста (абзац начинается хотя бы одним пробелом, заголовок — знаком табуляции и т. п.). Такой форматизатор нетрудно включить как факультативное средство в любой АК. Учесть пользователя правилам разметки при этом не нужно.

Во избежание протяженных межсловных интервалов при выравнивании правого края выходного текста в форматизаторы часто включают подпрограмму переноса слов со строки на строку. В славянских языках правила переносов опираются, в основном, на слоговое деление. Отклонение от этого принципа, например, в русском языке делается лишь в отношении приставок и одиночных гласных в начале и в конце слова. С помощью небольших таблиц (приставки, классификаторы букв, диаграммы или их веса) в славянских языках можно достичь довольно высокого качества переносов. Но получить то же качество теми же средствами в других языках удастся не всегда. Так, в английском языке, где переносы во многом опираются на морфемное членение словоформ и произношение, пришлось бы опираться на словарь, храня в каждом его слове места всех возможных переносов.

## § 7. Использование морфологического анализа

Даже если словоформы хранятся в словаре АК целиком, перед сравнением с ними текстовых словоформ необходимо подавить в последних их регистровые особенности (прописные и строчные буквы обычно имеют разные коды) и особенности шрифтовых гарнитур (если таковые есть). В англоязычной литературе этот предварительный этап обработки называют нормализацией.

В флективных языках хранить все словоформы технически невозможно — требования по объему памяти возрастают в 5—10 раз. Нужен автоматический морфологический анализ, придающий этапу нормализации совершенно новый смысл.

Под автоматическим морфологическим анализатором обычно понимается программа, которая в ответ на входную буквенную цепочку  $S$  (являющуюся правильной словоформой данного естественного языка) выдает пару  $(L, G)$ , где  $L$  — номер (по словарю) или нормализованное имя лексемы, к которой относится  $S$ ,  $G$  — набор грамматических характеристик словоформы. Применительно, например, к славянским языкам компонентами  $G$  являются род, число, падеж, лицо, вид, наклонение и т. п. (конкретный состав зависит от части речи).

В принципе число выходных пар такого вида может быть

равным 0, 1, 2... Нуль означает несуществующее или незнакомое анализатору слово; 1 — однозначный разбор; 2, 3... — многозначный разбор, в морфологии нередкий. Вот русский пример: *новым* — новый (ед., муж., твор.)/новый (ед., сред., твор.)/новый (мн., дат.)

Здесь роль *L* выполняет словарная форма прилагательного, *G* изображено набором характеристик в скобках (ед./м. — число, муж./сред. — род, твор./дат. — падеж), знак «/» разделяет альтернативы анализа.

В полном разборе такого вида АК не нуждаются. Здесь достаточно иметь так называемый акцептор, выдающий в ответ на *S* бинарный признак: 0 — недопустимая или незнакомая словоформа; 1 — допустимая (т. е. правильная) словоформа. Полный анализатор легко превратить в акцептор, но акцептор можно построить экономнее.

Предположим, что каждая словоформа языка распадается на начальную часть (основу), в процессе словоизменения инвариантную, и конечную часть (флексию), которая своими вариантами различает словоформы одной и той же лексемы. Если хранить в словаре основы с номерами их словоизменительных классов, а в отдельных грамматических таблицах — флексии, которые допустимы при основах разных классов, то алгоритм акцептора становится очевидным. Нужно отсекают от словоформы справа одну букву за другой и проверять, представляет ли отсеченная подцепочка допустимую флексию, если это так, — содержится ли остаток в словаре основ, а если подтвердилось и это, — допускает ли класс обнаруженной основы данную флексию. Как только допустимый разбор обнаружен, анализ можно прекратить, а иначе продолжить его вплоть до допустимого минимума длины основы или максимума длины флексии.

Морфологический анализ в его описанном виде эквивалентен тому, что иногда и называют нормализацией суффиксов. Кроме выделения основы и флексии нормализация включает замену последней на тот ее вариант, который условно принят при данной основе за «стандарт». Но простое расчленение с проверкой на допустимость для АК вполне достаточно.

Языковые особенности могут существенно усложнить приведенную картину:

- 1) Флексия часто является сочленением некоторого числа (1—6) суффиксальных морф, обладающих сложными законами сочетаемости. Набор этих морф не столь велик (в славянских языках — порядка 100), но если инвентаризовать все допустимые составленные из них флексии, то последних окажется раз в 10 больше.

- 2) На границе основы и флексии происходят многочисленные морфологические альтернации: руск. *сказать* — *скажу*, серб. *нога* — *нози*. В этих условиях, чтобы формально сохра-

нить схему СЛОВОФОРМА=НЕИЗМЕНЯЕМАЯ ОСНОВА+ ФЛЕКСИЯ, можно переместить границу деления на 1—2 буквы влево (после чего компоненты правильнее называть уже псевдоосновой и псевдофлексией). Количество классов основ при этом умножается примерно на число типов альтернатив, увеличивается и число флексий. Другим выходом является хранение для таких лексем двух или даже трех основ, но это ведет к расширению словаря.

3) Альтернативы могут происходить не только на границе основы, но и внутри нее (рус. *сплотить* — *сплачивать*) или выражаться в исчезновении одной или даже двух букв (рус. *стенка* — *стенке*, серб. *отац* — *оца*). Здесь можно либо перемещать границу деления еще левее, либо хранить несколько основ, либо перейти к алгоритмической обработке альтернатив, либо, наконец, хранить для таких случаев словоформы целиком, отказавшись от тотального морфоанализа.

Анализ содержимого современных научно-технических текстов, а также ненужность для АК полного морфоанализа дают основание ограничиться комбинацией двух первых способов, ибо удельный вес сложных альтернатив относительно велик лишь в текстах бытовых жанров.

Итак, для целей автокоррекции можно делить словоформы на две части (из них вторая может быть пустой), не подвергая основы и флексии в указанном их понимании никакому дальнейшему анализу. Основы делятся на классы эквивалентности, внутри каждого из которых совокупность присоединяемых флексий одинакова. Для акцептора не нужно рассматривать разные флексии независимо, поэтому и они разбиваются на максимальные классы эквивалентности, внутри каждого из которых собираются все флексии, присоединяемые к одним и тем же классам основ.

Априорно ограничим количество классов основ и флексий. В морфологии, покрывающей все возможные словоформы русского языка, нужно около 120 классов основ, но если отказаться от принципа тотальности, можно взять заметно меньшую величину, например,  $N=40$ . Удобство такого ограничения для возможности сжатия данных известно: в двух смежных байтах можно поместить три величины из диапазона  $0 \dots 39$ , на этом основан известный код RADIX-50. В тот же диапазон укладываются и алфавиты всех славянских языков (включая буквы с диакритическими знаками), причем для славянских языков хватает 32 буквы.

Задача морфологического описания естественного языка для АК формулируется тогда следующим упрощенным способом. Из всего инвентаря как словоизменительных, так и словообразовательных парадигм языка нужно выделить те наиболее используемые парадигмы, а в случае весьма неравномерно используемых парадигм — наиболее используемые

подпарадигмы в общем количестве  $N$ , которые требуют для своего двучленного описания не более  $N$  классов флексий.

Такая проблема была эвристически решена в [10] для английского и русского языков. От словарей объемом примерно в 2 тыс. основ удалось добиться продуктивности (отношение суммарного числа покрываемых словоформ к числу основ) около 6,2 и 51,2 соответственно. При расширении словарей до 20 тыс. основ и некоторого пересмотра их классов продуктивность снизилась лишь до 5,5 и 42,5 соответственно. Английский словарь основ покрывал при этом более 40 тыс. словоформ, русский — более 800 тыс.

Объяснение сохраняющихся высоких показателей заключено в охвате некоторыми классами основ словообразовательных парадигм. Так, к парадигме русского прилагательного типа СЕРЫЙ (основа *сер-*) присоединено наречие СЕРО и существительное СЕРОСТЬ, к парадигме глагола ИНФОРМИРОВАТЬ (основа *информ-*) присоединены ИНФОРМАЦИЯ и ИНФОРМАТОР, к глаголу ПЛАНИРОВАТЬ — ПЛАНИРОВАНИЕ и т. п. Нетрудно убедиться, что в других славянских языках имеются прямые аналоги этим расширенно понимаемым русским парадигмам.

Не существует однозначного решения, как быть с непокрываемыми сформированным морфословарем словоформами, к которым относятся морфологически своеобразные слова, от самых частотных до самых редких. Одним возможным решением служит создание для них непосредственно словарей словоформ, к одному из которых, высокочастотному, следует обращаться при поиске до словаря основ, а к другому, малочастотному, — после. Это решение имеет следующие достоинства:

— при обращении к этим словарям морфологического анализа не требуется;

— новые встретившиеся в тексте слова по указанию пользователя легко вставлять в подобные словари словоформ с тем, чтобы далее в сеансе эти слова были системе известны;

— возможно самообучение автокорректоров новой терминологии по текстам, в орфографической корректности которых пользователь уверен;

— аналогично тому, как каждой основе в словаре основ придается номер ее морфологического класса, можно ввести в словарях словоформ классы, характеризующие их дефисную сочетаемость (если раздельно хранить буквенные цепочки, составляющие русские словоформы типа *п-ка*, *S-функция*, *русско-польский* и т. п.), возможность их использования в качестве точечных сокращений и аббревиатур (типа *рус.*, *япон.*, *СЭВ*, *КПСС*) [11].

Однако в указанной классификации нуждается менее 10% всех русских словоформ. Поэтому можно включить непокрываемые анализом словоформы в общий словарь основ (например,

присвоив им класс 0). В качестве другого решения можно взять это выделенное значение номера класса основ в качестве префикса еще одной серии классификационных номеров из интервала 0...39. Один из двучленных номеров можно оставить за словоформами (т. е. вырожденным классом основ), а прочие 39 значений приписать дополнительным классам основ, пусть не таким частым и количественно обширным. Приближенные расчеты показали, что средняя продуктивность словаря русских основ сохраняется при этом не ниже 25, что оставляет словарь достаточно компактным.

## § 8. Частотная декомпозиция и объем словаря

Даже если использовать морфологический анализ и иные методы сжатого кодирования словарей (см. далее), разместить в ограниченной ОП микрокомпьютера словарь основ целиком, включающий в научно-технических текстах до 20—50 тыс. основ, не всегда удается. Возникает идея частотной декомпозиции словаря, т. е. разбиения его на части, одни из которых размещаются в ОП, а другие — на дисковом накопителе.

В [99] применительно к ЭВМ с обширной памятью и к английскому языку предложено сделать словарь словоформ трехчастным. Первая часть в ОП, содержит порядка 200—300 наиболее употребительных словоформ, вторая, тоже в ОП, содержит 1—2 тысяч так называемых документных словоформ, третья, на диске, включает все словоформы, не вошедшие в первый словарь.

Последовательность проверки слова из текста здесь такова. Сначала оно ищется среди общеупотребительных. Если оно находится, то признается правильным, а иначе поиск продолжается в документном словаре (в начале сеанса он пуст). Если слово найдено здесь, поиск кончается, а иначе продолжается на диске. При обнаружении там слова оно признается правильным и заносится для облегчения проверки последующей части того же текста в документный словарь. Если же текстовое слово не найдено нигде, то пользователь может в диалоге с АК признать его допустимым и даже вставить в документный словарь. Новые поступления в этот словарь эпизодически переносятся в словарь на диске, обогащая его на дальнейшее.

В [99] приводится ожидаемое процентное содержание нахождения текстовых слов в различных словарях такой трехчастной системы — 50%, 45% и 5%. Наши эксперименты подтвердили, что 300 наиболее ходовых английских словоформ, даже без тщательной подгонки под жанр и предметную область документа, охватывают 50% слов из текста. Но в славянских языках покрытие аналогичных текстов словарем словоформ того же объема из-за существенной флективности даже весьма употребительных служебных слов не превышает 30%.

Существенный ущерб приносит флективность и в части документных слов. Прежде всего 2 тысячи документных словоформ не всегда удается разместить в ОП. При уменьшении же этой цифры, например, до 200 покрытие документным словарем снижается для английских текстов примерно до 15%, а для русского — ниже 10%. В итоге этот словарь полезно сохранять, в основном, для новых поступлений.

Для обеспечения сравнимой с английским языком степени покрытия текста на флективном языке словарями, находящими в ОП, придется дополнить словари ОП хотя бы небольшим словарем основ. Тогда сочетание «ходовой словарь в 300 словоформ + документный словарь в 200 словоформ + внутренний словарь из 200 основ» (цифры примерные) сможет обеспечить покрытие половины славянского текста, но только при достаточном согласовании словарей с жанром и тематикой проверяемых текстов.

### § 9. Сжатое кодирование словарей

Процессоры мини- и микрокомпьютеров по скорости немногим уступают процессорам больших ЭВМ, чего нельзя сказать об объемах ОП и дисковой памяти. Поэтому столь остро стоит для АҚ на малых машинах проблема сжатия словарей. Перечислим некоторые способы сжатия, пригодные для АҚ.

Под словарями вообще будем понимать совокупность буквенных цепочек переменной, но ограниченной длины, снабженных при необходимости числовыми реквизитами. Цепочки могут являться словоформами, основами или последовательностями суффиксов/флексий естественных языков. Реквизиты — это словоизменятельные классы или какие-либо иные характеристики цепочек.

Исходный вид таких словарей — текстовой, но для непосредственного использования в программах АҚ они обычно переводятся в рабочие бинарные файлы. Как при загрузке их целиком в ОП, так и при привлечении их по частям с диска (в том числе — «электронного») нужно предельно компактно их закодировать.

В литературе предложены необратимые и обратимые способы сжатия. При необратимых способах вместо буквенных цепочек хранятся их сжатые бинарные образы, например, хеш-коды. Сжатие получается сравнительно высоким (в 4 и более раз), но в общем случае, особенно — при сменном содержимом словаря, приходится довольствоваться опознаванием слов, содержащихся в словаре, лишь с высокой вероятностью, в то время как обратимое сжатие гарантирует 100-процентное точное восстановление и опознавание. Рассмотрим лишь обратимые способы.

В [4] предложено хранить словоформы с 4 реквизитами в виде длин словообразовательных и словоизменятельных основ и классов двух этих основ. Хотя при этом хранятся лишь наиболее частотные словоформы, а прочие «выводятся» с помощью их реквизитов, сжатие здесь затруднено.

Максимальный эффект достигается при хранении основ. Дополнительную экономию в 1,5 раз дает размещение кодов трех смежных букв в двухбайтовом слове (способ, близкий к RADIX-50). Прочие способы сжатия существенно зависят от того, размещен ли бинарный файл в ОП или на диске.

В случае ОП важными оказываются изменяемость/неизменяемость словаря после первоначальной загрузки и наличие/отсутствие в словаре повторяющихся цепочек (с разными реквизитами). При изменяемости приходится применять хеширование с промежуточными структурами из подписков коллизий и с отдельным пулом буквенных цепочек. При неизменяемости словаря можно применять: заранее подстроенное под данный словарь хеширование, адресующее к пулу цепочек; индексный массив из отсылок к пулу для дихотомического поиска в последнем; побуквенное графовое представление словарей со сращиванием совпадающих начал и концов слов-цепочек.

Последний способ подробно изучен в [14]. Найден алгоритм минимизации графа, доказаны его минимизационные свойства и реализуемость по времени и объему на малых машинах. Проведены эксперименты по кодированию этим алгоритмом словарей с числом составляющих от нескольких десятков до тысячи. Сжимаемые словари включали наиболее употребительные словоформы, основы и флексии русского и английского языков. Для русских флексий способ дал примерно шестикратный выигрыш по сравнению с дихотомическим способом, для основ — проигрыш, а в прочих случаях отмечались умеренные выигрыши (в 1,5—2 раза) в зависимости от языка, направления просмотра цепочек и других факторов.

Хеширование, даже подстроенное под словарь, дает результаты, по памяти несколько худшие, чем дихотомический способ.

Дисковые словари целесообразно хранить логическими блоками, кратными блокам дискового накопителя. Отсылка из ОП совершается к блоку в целом, так что при поиске внутри него остается рассчитывать лишь на внутриблочное лексикографическое упорядочение. Хотя линейный просмотр замедляет поиск, зато память здесь можно сэкономить несколькими способами:

1) способом Купера [49, 87], когда в очередной цепочке хранится длина той ее части, которая совпадает с началом предыдущей цепочки;

2) дополнением к способу Купера, когда наряду с длиной совпадающей части хранится бит, указывающий, является ли

первая несовпадающая буква смежной по алфавиту с буквой на той же позиции в предыдущей цепочке;

3) кодированием наиболее употребительных окончных подцепочек однобуквенными кодами, дополняющими алфавит до 40 «литер»;

4) кодированием литер, полученных в результате применения предыдущих способов, еще и по Хаффмену;

5) умалчиванием особо часто повторяющихся значений числового реквизита.

В словарях русских основ использование способов 1, 2 и 3 дало выигрыш в объеме памяти порядка 2, а с учетом хранения лишь основ и способа RADIX-50 — около 130. В словарях словоформ использование способом 1, 2 и 5 дало выигрыш порядка 3, а с учетом RADIX-50 — 4,5. Если исходить из многочисленных экспериментов по кодам Хаффмена, то можно ожидать, что указанные кумулятивные показатели удастся поднять едва ли более, чем в 1,2 раза.

В случае, если слить словари основ и словоформ, кумулятивный коэффициент сжатия упадет для русского языка примерно до 100, оставаясь все еще значительным. Здесь уместно отметить, что для англоязычных АК считается полезным сжатие даже на 15 %.

## § 10. Внешнее и внутреннее кодирование проверяемых текстов

Проверяемые с помощью АК тексты, как готовые, так и вводимые в сеансе работы, размещаются в текстовых файлах строками переменной длины. Предельная длина строки согласуется с дисплеем (80) или с принтером (80—132 позиции). Разрывать слова границами строк, в том числе — переносами, не рекомендуется, поскольку части слов на разных строках рассматриваются типовым АК как отдельные слова. По тем же причинам не рекомендуется использовать дефисные сокращения (типа *sv-va* вместо *свойства*).

Внешнее кодирование текстов обычно осуществляется с помощью стандартных кодов — ASCII, КОИ или др. Сочетание прописных и строчных букв может при этом соответствовать одному из трех случаев:

1) Тексты набраны только прописными буквами и регистр отдельных букв для последующих представлений этих текстов не существует.

2) Тексты представлены прописными и строчными буквами в строгом соответствии с орфографией, причем разные регистры имеют разные коды.

3) Тексты набраны только прописными буквами, но содержат особую разметку тех букв, которым надлежит быть прописными при окончательном оформлении текстов. Разметка заключается в предварении или окаймлении одной или несколь-

ких смежных букв специальным знаком, например, « ». Буквы внутри выделенных разметок участников должны быть прописными, прочие же — строчными. Подобная разметка является как бы одним из элементов полиграфической разметки (см. выше).

Свойства пользовательского дисплея должны быть как-то согласованы с характером кодирования букв, допуская, если нужно, оба регистра букв одного алфавита, а для многоязычных полиалфавитных АК — хотя бы прописные буквы разных алфавитов.

Вариант представления букв после проверок обычно должен сохраняться.

Внутреннее кодирование не обязано повторять внешнее. Удобно хранить очередной фрагмент текста в буфере ОП из строк, умещающихся на экране дисплея (80 позиций). Слишком длинные строки расщепляются, причем граница переноса помещается между словами текста без их разрывов.

Порядок следования внутренних кодов букв можно выбирать по ситуации. Если, например, внешние коды идут не подряд (подобно русским буквам в коде ДКОИ), то их можно перевести в непрерывный интервал. Для ускорения поиска в словарях иногда удобно перерасположить эти коды в порядке убывания средней частоты их встречаемости — либо в любой позиции внутри слова данного языка, либо как-то иначе (например, в непервой или непоследней позиции в слове).

Хотя АК может быть рассчитан на тексты с несколькими разными способами различения буквенных регистров (см. выше), его внутренние коды естественно брать инвариантными. Даже в случае, когда таблица внешних кодов охватывает менее 256 значений (например, 128), внутри АК разумно пользоваться всеми 256 допустимыми однобайтовыми комбинациями.

Может оказаться полезным принять два способа внутреннего кодирования: один — для хранения и воспроизведения текстов, другой — для хранения словарей. Действительно, тексты разнообразны по сочетанию алфавитов и регистров букв, а в словарях можно либо вообще не хранить сведений о регистрах, либо хранить их в ограниченном числе комбинаций. Имеются в виду варианты первой прописной буквы у имен типа *Москва* и всех прописных букв у аббревиатур типа *СССР* (слова типа *ЛенЦНТИ* и *микроЭВМ*, где комбинации регистров более сложны, в русском языке пока очень редки).

На выбранный способ внутреннего кодирования могут опираться способы внутреннего упорядочения словарей. В частности, если один из них использует упорядочение по частоте встречаемости букв, то для облегчения совместимости тот же порядок может быть принят и для прочих словарей. Отличие такого порядка от общего алфавитного или навязываемого таблицей *ASCII* (КОИ) ничему не мешает.

## § 11. Вспомогательные средства

В развитый АК могут быть встроены и многие иные сервисные средства, нацеленные на оперативную подстройку параметров системы и ее базы данных, т. е. словарей и поисковых структур. В качестве регулируемых системных параметров могут быть взяты:

— размер контекста, демонстрируемого вместе с сомнительным словом. Этот размер может регулироваться от 1 до 10—12 строк и разбиваться на зоны, в средней из которых находится сомнительный участок;

— предельное число гипотез замены;

— включение/выключение проверки чисел, буквенно-цифровых смесей, иностранных вкраплений, дефисных образований, точечных сокращений, правильность сочетания буквенных регистров и др.;

— включение/выключение демонстрации на экране встречных в тексте управляющих символов; введение и стирание стартопных символов, окаймляющих участки текста, не подлежащие проверке; ускорение основных операций АК за счет исключения отчетных (см. ниже);

— включение/выключение средств полиграфической подготовки текста, что может включать разметку мест допустимых переносов и выделение «истинно прописных» букв (в текстах, набранных целиком прописными буквами);

— установка параметров выводного файла, форматизируемого автономно, т. е. силами АК (предельное число строк на странице, число литер в строке и т. п.).

Операции над системной базой данных включают многостороннюю поддержку словарей. Словари внутри ОП можно очищать, выводить в указанный файл, загружать из указанного файла. Внешние словари можно сортировать, сливать или фильтровать (т. е. вычеркивать составляющие одного файла из содержимого другого файла).

Полезны также средства специализации рабочих словарей. По умолчанию специализацию лучше не проводить, и для работы автоматически привлекать ядерный словарь, содержащий лексику, принятую во многих научно-технических областях. Специализация, совершаемая по указанию пользователя, подразумевает присоединение к ядерному словарю 1—3 небольших профильных словарей. Так, для биологических текстов нужно слить с ядром словарь биологических терминов.

Слова, накопленные в данном сеансе, по требованию могут включаться в ядерный словарь в расчете на дальнейшие сеансы.

При частотной декомпозиции словарей системы полезно накапливать в процессе работы сведения об употребительности наиболее ходовых словоформ с целью коррекции соответствующего словаря. Для словарей основ целесообразно иметь от-

дельную диалоговую подпрограмму их автоматизированного составления и коррекции. В расчете на пользователя, не обученного лингвистике, создать такую подпрограмму непросто.

Одновременно с работой над текстами полезно автоматически составлять статистический отчет о числе литер, слов и строк, считанных из проверенных файлов, объемах поддерживаемых системой структур (справочных массивов, строковых пулов, словарных файлов на диске), числе нахождения слов в каждом из словарей системы и иных ее показателях.

В АК разумно предусмотреть также разнообразные средства обучения пользователя, а именно:

— специально вызываемый пользователем на экран компактный (порядка 5—8 экранов) файл, содержащий описание основных свойств системы;

— сообщения «помощи», по просьбе пользователя описывающие 1—2 строками любую из команд, применимую в текущий момент;

— разъяснения только что совершенной пользователем ошибки;

— предупреждения, поясняющие результаты отдельных шагов и причины возникающих задержек обработки.

Ведению системы, т. е. динамической локализации ее остаточных ошибок, тонкому освоению, расширению и оптимизации помогают средства трассировки, которые должны включаться только факультативно. При трассировке выдаются номера или имена активизируемых процедур и привлекаемых частей дисковых словарей, показывается перемещение точки внимания вдоль проверяемого текста, процесс отсева гипотез замены, реально произведенные автоматические замены и т. п.

## § 12. Выдвижение кандидатов на исправление

Этап выдвижения кандидатов на исправление сомнительной буквенной цепочки в тексте не отделим от вопроса, какие из многочисленных допускаемых естественным языком (и словарем данного АК) словоформ считать «сходными» с данной текстовой цепочкой.

Концептуально просто решается эта проблема, когда предполагаются лишь так называемые однобуквенные ошибки, т. е. 1) пропуски буквы на любом месте в слове, 2) вставки лишней буквы в любом месте, 3) замены одной буквы, 4) перестановки двух смежных букв. Статистика ошибок при массовом вводе текстов показывает, что однобуквенные ошибки имеют место более, чем в 85% случаев [37].

По определению полагается, что однобуквенная ошибка сохраняет ошибочное слово сходным с первоначальным его вариантом. Тогда выделить кандидатов можно либо перебирая словоформы в словаре (или его части) и сравнивая с текстовым словом, пытаясь обнаружить в словаре «однобуквенное» исправле-

ние текстовой цепочки, либо порождая все мыслимые варианты восстановления в текстовой цепочке однобуквенной ошибки и предъявляя «правильные» слова словарю. Первый вариант подходит лишь для словарей словоформ, причем для больших словарей неэкономичен, а для словарей основ вообще едва ли осуществим. Но и второй вариант не столь уже практически прост.

Действительно, если в алфавите данного естественного языка  $N$  букв, а длина текстовой цепочки равна  $L$ , то пробных шагов нужно:  $L-1$  — для восстановления всех перестановок,  $N(L+1)$  — для пропусков,  $(N-1)L$  — для замен,  $L$  — для вставок, всего  $2NL+N+L-1$  шагов. При средней длине слова 10 для английского языка, в котором 26 букв плюс апостроф, это составит 576 шагов, а для русского (без учета ё) — 681. Для более длинных слов эти оценки нужно примерно пропорционально увеличить.

Это означает, что при восстановлении даже однобуквенных ошибок вместо одного обращения к словарю требуется несколько сот. Расположение словарей, хотя бы частичное, на внешних накопителях может сделать подбор всех вариантов неэргономичным (3—10 с/слово) даже при жестких дисках, в случае же флоппи-дисков это вообще неприемлемо.

Чтобы уменьшить расходы, можно применить отсеиваемых словарю вариантов полиграммным методом (см. выше), искать лишь заранее ограниченное небольшое число вариантов либо ограничиться проверкой по словарю лишь вариантов, которые наиболее вероятны из-за низкой квалификации операторов—машинисток (клавиатурные ошибки) или их малограмотности. Задача существенно облегчается, когда весь словарь удается разместить в оперативной памяти.

Если речь идет о двух- и более буквенных ошибках, то полный перебор осуществим лишь на очень мощных установках. Чтобы помочь этому процессу, предложены различные критерии сходства — сходство звучания, обобщенное расстояние между цепочками по Левенштейну, длина максимальной общей подстроки или минимальной объемлющей строки [27], число общих триграмм [38] и др.

В случае триграмм предложено создавать «инвертированные» файлы, по триграммам отсылающие к нужным «сходным» словам. Подобные вспомогательные файлы крайне громоздки (несколько сот Кбайт) и поэтому нагружают установку порой больше, чем собственно словарь. К тому же триграммное сходство в большинстве своем означает просто «однобуквенное» сходство. Можно, тем самым, констатировать, что подбор однобуквенных вариантов исправления — дело слишком громоздкое и не окупающее расходов машинных ресурсов.

Возвращаясь к однобуквенным ошибкам, хотелось бы найти такую организацию словаря АҚ, которая концентрировала бы «сходные» цепочки в как можно более узких подобластях.

Поскольку однобуквенные ошибки либо оставляют длину слова неизменной (перестановки и замены) либо меняют ее на  $\pm 1$  (пропуски и вставки), один или возможных исходов — разбить словарь на зоны с равной длиной составляющих. Но такой прием возможен лишь для словоформ. Он снижает эффективность хранения словаря и несколько противоречит требованиям первого, более важного этапа обнаружения ошибок.

Можно вообще усомниться в том, что сходство цепочек и их линейное упорядочение в словарях основ удастся когда-нибудь конструктивно совместить, одновременно удовлетворив оба этапа работы АК. Когда же список кандидатов найден, их можно упорядочить перед предъявлением человеку исходя из

— сходства звучания (особенно — в английском языке),

— вероятностей пальцевых ошибок оператора (особенно — в русском языке),

— вероятностей появления конкретных слов в тексте (для этого нужно хранить в словаре сведения об употребительности отдельных слов).

Более подробный обзор методов подбора кандидатов см. в [2].

### § 13. Некоторые перспективы

Наше рассмотрение показывает, что многие черты современных АК продиктованы не столько особенностями естественных языков, сколько ограниченными возможностями современных ЭВМ, особенно — малых. К таким чертам относятся всевозможные приемы сжатия словарей и снижения переборов при поиске. Если не обращать внимания на техническую сторону дела, то, например, задачу подбора кандидатов при однобуквенных ошибках можно сформулировать на языке Пролог всего десятком утверждений [40]. (Обращение к словарю при этом формулируется, как удовлетворение текстовой цепочки предикату «Есть в словаре».) Не намного больше утверждений требуют операции выделения цепочек из текста. По мере увеличения производительности современных Пролог-машин такие программы станут выполняться за приемлемое время сначала на больших, а затем — и на малых установках.

Морфологический анализатор для флективных языков типа русского, по нашему мнению, останется и в перспективе. Он не только снижает требования к вычислителю, но и позволяет в каком-то смысле допускать незнакомые слова, по морфологическим свойствам «близкие» к уже известным системе.

Грандиозной и решающей перспективой для АК является, конечно, введение в них синтаксического анализа. Только он позволит проверять тексты с определенной степенью их «понимания», всегда обнаруживать замены одного правильного слова на другое правильное, но неуместное, а также выявлять синтаксические и стилистические ошибки.

## ЛИТЕРАТУРА

1. *Андреевски А., Дебили Ф., Флур К.*, Об одном важном свойстве лексикн естественных языков и его использовании при автоматическом исправлении опечаток. Прикладные и экспериментальные лингвистические процессоры. Новосибирск: ВЦ СО АН СССР, 1982, 98—109 (РЖМат, 1983, 2В1029)
2. *Бабко-Малая О. В., Шемраков В. А.*, Методы и системы автоматизированного обнаружения и коррекции текстовых ошибок. Препринт № 5. Л.: БАН СССР, 1987, 46 с.
3. *Белоногов Г. Г. и др.* Проблема автоматического обнаружения и исправления ошибок в научно-технических текстах. НТИ, 1982, сер. 2, № 6, 29—31
4. — и др. Алгоритм многоступенчатого морфологического анализа русских слов. НТИ, 1983, сер. 2, № 1, 6—10
5. — и др. Экспериментальная система автоматизированного обнаружения и исправления орфографических ошибок в текстах. НТИ, 1984, сер. 2, № 3, 20—27
6. — и др. Результаты функционирования в ВИНТИ системы обнаружения орфографических ошибок в режиме опытной эксплуатации. Вопросы информационной теории и практики. 1984, № 51, 24—44
7. —, *Кузнецов Б. А.*, Языковые средства автоматизированных систем. М.: Наука, 1983, 217 с.
8. —, —, *Новоселов А. П.*, Автоматизированная обработка научно-технической информации. Лингвистические аспекты. Итоги науки и техники. Информатика. 8. М.: ВИНТИ, 1984
9. *Большаков И. А.*, Автоматическое обнаружение и исправление ошибок как технологическая предпосылка смысловой обработки текста. Семiotические аспекты формализации интеллектуальной деятельности: Тез. докл. и сообщ. М.: ВИНТИ, 1983, 179—182
10. —, Упрощенный морфологический анализ при автоматической проверке правильности текстов. НТИ, 1985, сер. 2, № 6, 22—28
11. —, Автоматическая проверка правильности дефисных образований. НТИ, 1986, сер. 2, № 2, 28—31
12. —, ДИСКОР — диалоговая система коррекции текстов. НТИ, 1986, сер. 2, № 5, 8—15
13. —, О чисто автоматической коррекции текстов с опорой на клавиатурную модель типовых ошибок. НТИ, 1987, сер. 2, № 3, 27—31
14. —, *Емелин Е. В.*, Алгоритм минимизации графового представления словарей. Изв. АН СССР, Тех. кибернет. 1987, № 4, 3—13 (РЖМат, 1987, 12Г367)
15. *Бояринов И. М. и др.*, Использование помехоустойчивого кодирования для защиты информации от ошибок оператора. Автоматика и телемеханика, 1983, № 2, 5—49
16. *Братчиков И. Л.*, Метод обнаружения и исправления искажений в русских словоформах, основанный на функции расстояния. Семiotические аспекты формализации интеллектуальной деятельности: Тез. докл. и сообщ. М.: ВИНТИ, 1985, 393—395
17. *Волков В. Н., Иванисов А. В.*, Реализация алгоритма распознавания и выборки слов с использованием функции совпадения. Программирование, 1982, № 2, 90—92 (РЖМат, 1982, 7В1016)
18. *Дедиков Э. А., Чен Р. Н.*, Организация корректирующего машинного словаря имен с помощью аддитивной функции хеширования. Проблемы бионки (Харьков). 1982, № 28, 14—19
19. *Деннинг В., Эсиг Г., Маас С.*, Диалоговые системы «человек—ЭВМ». Адаптация к требованиям пользователя. М.: Мир, 1984
20. *Долгополов А. С.*, Машинное распознавание орфографических искажений информации. Управляющие системы и машины (Киев). 1976, № 5, 79—82 (РЖМат, 1976, 2В922)

21. —, Об автоматическом корректоре текстов. НТИ, 1985, сер. 2, № 3, 27—28
22. —, Недвоичные коды, исправляющие вставки, выпадения и замены символов. Проблемы передачи информации. Вып. 1, 1985 (РЖМат, 1985, 7Г52)
23. —, Языковая оптимизация коммуникативных систем. НТИ, 1985, сер. 2, № 9, 12—15
24. —, Программа автоматической коррекции текстов. НТИ, 1986, сер. 2, № 4, 26—29
25. *Епископоян Р. Л.* Метод сокращения времени на автоматическую нейтрализацию орфографических ошибок. Управляющие системы и машины (Киев). Вып. 6, 1983, 82—84
26. *Коростелев Л. Ю.* Некоторые особенности обработки неопознанных слов в системе машинного перевода. НТИ, 1985, сер. 2, № 4, 23—28
27. *Красиков Ю. В.* Теория речевых ошибок (на материале ошибок наборщика). М.: Наука, 1980, 160 с.
28. *Курбаков К. И.* Кодирование и поиск информации в автоматизированном словаре. М.: Советское радио, 1968, 214 с.
29. *Левенштейн В. И.* Двоичные коды с исправлением выпадений, вставок и замещений символов. ДАН СССР, 163, 1965, № 4, 845—848
30. *Матвеев С. А., Сотникова Р. А.* Система автоматической коррекции ошибок в словосочетаниях. Программирование, 1984, № 5, 68—74 (РЖМат, 1985, 1Г469)
31. *Партыко З. В.* Анализ искажений, возникающих при вводе текстов в ИИС «АССИСТЕНТ». НТИ, 1982, сер. 2, № 1, 21—26
32. —, Методы машинной корректуры и машинного редактирования. М.: Книга, 1983, 40 с.
33. *Поздняк М. В. и др.* Система автоматизированного обнаружения и исправления ошибок. Вопросы информационной теории и практики, 1984, № 51, 12—23
34. *Салмина Н. Ю., Ходашинский И. А.* Методы и средства автоматического исправления орфографических ошибок. НТИ, 1986, сер. 2, № 10, 25—28
35. *Сушилин В. А.* Обнаружение ошибок в научно-технических текстах средствами автоматизированной ИПС. НТИ, 1983, сер. 2, № 2, 39
36. *Штурман Я. П.* Анализ систем автоматизированного обнаружения орфографических ошибок. НТИ, 1985, сер. 2, № 9, 21—24
37. —, *Партыко З. В.* Анализ искажений при вводе реферативной информации в систему «АССИСТЕНТ». НТИ, 1982, сер. 2, № 3, 17—31
38. *Angel R. C. et al.* Automatic spelling correction using a trigram similarity measure. Inform. Proc. & Manag., 1983, 19, № 4, 255—261
39. *Alberga C. N.* String similarity and misspellings. Commun. ACM, 1967, 10, № 5, 302—313
40. *Berghel H. L.* A logical framework for correction of spelling errors in electronic documents. Inform. Proc. & Manag., 1987, 23, № 5, 477—494
41. *Blair C. R.* A program for correcting spelling errors. Inform. & Control, 1960, 3, 60—67 (РЖМат, 1962, 9В403)
42. *Bloom B. H.* Space/time trade-offs in hash coding with allowable errors. Commun. ACM, 1977, 13, № 7, 422—426
43. *Bourne C. P.* Frequency and impact of spelling errors in bibliographic data bases. Inform. Proc. & Manag., 1977, 13, № 1, 1—12
44. *Burkhard W. A.* Partial match retrieval. Bit., 1976, 16, № 1, 13—31
45. —, Associative retrieval tree hash coding. J. Comp. & Syst. Sci., 1977, 15, № 3, 280—299
46. *Chen C. H.* Finite sample consideration in statistical pattern recognition. IEEE Comp. Conf. on Patt. Recogn., 1978, 188—192
47. *Cobham A.* Representation of a word function as the sum of two functions. Math. Syst. Theory, 1978, 11, № 4, 31—36 (РЖМат, 1979, 1В702)
48. *Comer D., Shen V. Y.* Hash-bucket search: a fast technique for searching in English spelling dictionary. Software—Practice & Experience, 1982, 12, 669—682

49. *Cooper W. S.*, The storage problem. *Mech. Transl.*, 1958, 5, № 2, 78—83
50. *Cornew R. W.*, A statistical method of spelling correction. *Inform. & Control*, 1968, 12, 79—93
51. *Damerau F.*, A technique for computer detection and correction of spelling errors. *Commun. ACM*, 1964, 7, № 3, 171—176
52. *Doster W.*, *Schurmann J.*, An application of the modified Viterbi algorithm in text recognition. *Proc. 5th Int. Conf. Patt. Recogn.*, Miami Beach, Fl., 1980, 855—863
53. *Dunlavey M. R.*, On spelling correction and beyond. *Commun. ACM*, 1981, 24, № 9, 608
54. *Durham I. et al.*, Spelling correction in user interfaces. *Commun. ACM*, 1981, 26, № 10, 764—773
55. *Feyock S.*, Transition diagram — based CAI/HELP systems. *Int. J. Man—Mach. Studies*, 1977, № 9, 339—413 (PЖMar, 1978, 3B791)
56. *Findler N. V.*, *Leenwen J. V.*, A family of similarity measures between two strings. *IEEE Trans. on Pattern Analysis & Mach. Intell.*, 1979, 1, № 1, 116—118
57. *Foisy A.*, *Lapalme G.*, Structuration d'un dictionnaire français pour la detection de fautes d'orthographe. *Actes de journées sur la manipulation de documents*, Rennes, Le Chesnay, 4—6 Mai, 1983, 176—180
58. *Fourney G. D.*, The Viterbi algorithm. *IEEE Proc.*, 1983, 61, № 3, 268—278
59. *Friedman T. H.*, An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software*, 1977, 3, 209—226
60. *Furuta R.*, *Scofield J.*, *Shaw A.*, Document formatting systems: survey, concepts, and issues. *Comp. Surveys*, 1982, 14, № 3, 417—472 (PЖMar, 1983, 6B1057)
61. *Galli E. J.*, *Yamada H. M.*, An automatic dictionary and verification of machine readable text. *IBM Syst. J.*, 1967, 6, № 3, 192—207
62. —, Experimental studies of computer-assisted correction of unorthographic text. *IEEE Trans. on Engineering Writing and Speech*, 1968, 11, № 2, 73—84
63. *Gingardella J. J.*, et al. Spelling correction by representation using digital computer. *IEEE Trans. on Engineering Writing and Speech*, 1967, 10, № 2, 57—65
64. *Graham S. L.*, *Rhodes S. P.*, Practical syntactic error recovery. *Commun. ACM*, 1975, 18, № 11, 639—650
65. *Grentas E. C.*, *Rosenbaum W. S.*, Automatic spelling verification: towards a system solution for the office. *Inf. Techn. Proc. 3rd Jerusalem Conf. Inf. Techn.*, 1978, 225—231
66. *Hall P. A. V.*, *Dowling G. R.*, Approximate string matching. *Comp. Surveys*, 1980, 12, № 4, 381—402 (PЖMar, 1981, 6B1085)
67. *Heidorn G. E. et al.*, The EPISTLE text critiquing system. *IBM Syst. J.*, 1982, 21, № 3, 305—326
68. *Hendrix G. G.*, Human engineering for applied natural language processing. *Proc. 5th Int. Joint Conf. Artif. Intell.*, Cambridge, Mass., 1977, 1, 183—191
69. *Hirschberg D. S.*, A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 1975, 18, № 6, 341—343 (PЖMar, 1976, 1B1323)
70. —, Algorithms for the longest common subsequence problem. *J. ACM*, 1977, 34, № 4, 664—675 (PЖMar, 1978, 6B1168)
71. *Hsu W. J.*, *Du M. W.*, Computing a longest common subsequence for a set of strings. *Bit.*, 1984, 24, № 1, 45—49 (PЖMar, 1984, 10Г414)
72. —, —, New algorithms for LCS problem. *J. Comp. & Syst. Sci.*, 1984, 29, № 2, 133—152 (PЖMar, 1985, 5Г291)
73. *Hunt J. W.*, *Szymanski T. G.*, A fast algorithm for computing longest common subsequences. *Commun. ACM*, 1977, 20, № 5, 350—353

74. Ito T., Kizawa M., Hierarchical file organization and its application to similar—string matching. ACM Trans. on Database Systems, 1982, 8, № 3, 410—433
75. Joseph D. M., Wong R. L., Correction of misspellings and typographical errors in free text medical English information storage and retrieval systems. Math. Inform. Med., 1979, 18, № 4, 228—234
76. Karzewski I., Chodorowski I., Michalewicz M., Automatic correction. Warszawa, 1975
77. Kashyap R. L., Oommen B. J. An effective algorithm for string correction using generalized edit distances. I. Description of the algorithm and its optimality. Inform. Sci., 1981, 23, № 2, 123—142
78. —, —, An effective algorithm for string correction using generalized edit distances. II. Computational complexity of the algorithm and some applications. Inform. Sci., 1981, 23, № 3, 210—217
79. —, —, Pattern matching with noisy substrings. Proc. of Compsac 81, IEEE Comp. Software and Appl. Conf., Nov., 18—20. 1981, 119—125
80. —, —, A common basis for similarity measures involving two strings. Int. J. Comp. Math., 1983, 13, 17—40
81. —, —, Spelling correction using probabilistic methods. Pattern Recognit. Letters, 1984, 2, № 3, 147—154 (PЖMar, 1984, 12Г49)
82. Kawal A. et al., Sentence structure standardization method for detection of errors in English sentences. Systems, Computers, Controls, 1983, 14, № 2, 84—92
83. Klarner D., Sets of words which omit specified words and subwords. Proc. Kon. Ned. Awad. Wetensch., 1978, A81, № 2 (PЖMar, 1979, 1B1310)
84. Krause J., Natural language access to information systems: an evaluation study of its acceptance by end—users. Inform. Syst., 1980, 5, № 4, 297—318
85. Lowrance K., Wagner R. A., An extension of the string—to—string correction problem. J. ACM, 1975, 22, № 3, 177—183
86. Maguire M., Computer recognition of textual keyboard inputs from naive users. Behaviour & Inform. Technology, 1982, 1, № 2, 93—111
87. Martin J., Computer data—base organization. New Jersey: Prentice Hall Int., 1975, 615 pp.
88. Meirowitz N., Van Dam A., Interactive editing systems. Comp. Surveys, 1982, 14, № 3, 321—416
89. Miller G. A. et al., Length—frequency statistics for written English texts. Inform. & Control, 1958, 1, № 4, 370—389 (PЖMar, 1959, 11406)
90. —, Friedman E. A., The Reconstruction of mutilated English texts. Inform. & Control, 1957, 1, № 1, 38—55 (PЖMar, 1959, 1865)
91. Milles L. A. et al., Text critiquing with EPISTLE system: an author's aid to better syntax. AFIPS Conf. Proc. 50, May, 1981, 649—655
92. Mitton R., Spelling checkers, spelling correctors, and misspelling of poor spellers. Inform. Proc. & Manag., 1987, 23, № 5, 495—505
93. Mor M., Fraenkel A. S., Retrieval in an environment of faulty texts or faulty queries. Improv. Database Usabil. & Responsib.: 2nd Int. Conf. Databases, N. Y. et al., 1982, 405—425
94. Morgan H. L., Spelling correction in system programs. Commun. ACM, 1970, 13, № 2, 90—94
95. Morris R., Cherry L. L., Computer detection of typographic errors. IEEE Trans. on Prof. Commun., 1975, M18, 54—63
96. Mukherjee S. R., Sloan M., Positional representation of English words. IEEE Trans. on Prof. Commun., 1985, 38, 587—591
97. Muth F. E., Tharp A. L., Correction human errors in alphanumeric terminal input. Inform. Proc. & Manag., 1977, 13, № 6, 329—337
98. Nix R., Experience with a space efficient way to store a dictionary. Commun. ACM, 1981, 24, № 5, 297—298

99. *Peterson J. L.*, Computer programs for detection and correction spelling errors. *Commun. ACM*, 1980, 23, № 12, 676—687 (PЖMar, 1981, 11B1133)
100. —, Computer programs for spelling correction: an experiment in program design. Berlin et al.: Springer, 1980, 213 pp. (PЖMar, 1981, 7B1337K)
101. —, A note on undetected typing errors. *Commun. ACM*, 1986, 29, № 7, 633—637
102. *Pollock J. J.*, Spelling errors detection and correction by computer: some notes and bibliography. *J. of Documentation*, 1982, 38, № 4, 282—291
103. —, *Zamora A.*, Collection and characterization of spelling errors in scientific and scholarly texts: *J. Amer. Soc. Inf. Sci.*, 1983, 34, № 1, 51—58
104. —, —, Automatic spelling correction in scientific and scholarly text. *Commun. ACM*, 1984, 27, № 4, 358—368
105. —, —, System design for detection and correction of spelling errors in scientific and scholarly text. *J. Amer. Soc. Inf. Sci.*, 1984, 35, № 2, 104—109
106. *Prescott R. L.*, On spelling error detection. *Commun. ACM*, 1981, 24, № 5, 331—332
107. *Rieseman E. M.*, *Hanson A. R.*, A contextual postprocessing system for error correction using binary n-grams. *IEEE Trans. on Computer*, 1974, 23, № 5, 480—493
108. *Robinson P.*, *Singer D.*, Another spelling correction program. *Commun. ACM*, 1981, 24, № 5, 296—297
109. *Schek H. J.*, Tolerating fuzziness in key-fords by similarity searches. *Kybernetes*, 1977, 6, 175—184
110. *Sellers K. H.*, An algorithm for the distance between two finite sequences. *J. Combin. Theory (A)*, 1974, 16, № 2, 253—258
111. *Shapiro M.*, The choice reference points in best—match file searching. *Commun. ACM*, 1977, 20, № 5, 339—343
112. *Shinghal R. et al.*, A simplified heuristic version of recursive Bayes algorithm for using context in text recognition. *IEEE Trans. on System, Man and Cybernetics*, 1978, 8, № 5, 412—414
113. *Tonssaint G. T.*, A bottom—up and top—down approach in using context in text recognition. *Int. J. Man—Mach. Studies*, 1979, 11, № 2, 201—202
114. —, —, Experiments in text recognition with modified Viterbi algorithm. *IEEE Trans. on Pattern Anal. & Mach. Intell.*, 1979, 1, 184—192
115. *Smit G. de V.*, A comparison of three string matching algorithms. *Software—Practice & Experience*, 1982, 12, 57—66
116. *Subieta K.*, A simple method of data correction. *Prace IPI PAN*, 1983, № 527, 12 pp. (PЖMar, 1984, 8Г393)
117. *Szanser A. J.*, Automatic error correction in natural languages. *Inform. Storage & Retrieval*, 1970, 5, № 4, 167—174
118. —, Bracketing technique in elastic matching. *Comp. J.*, 1973, 16, № 2, 132—134
119. *Szelenyi G. R.*, Formale und Semantische Prufmoglikeiten in Textverarbeitung. *Office Manag.*, 1983, № 1, 30—33
120. *Tagliacozzo R. et al.*, Orthographic error patterns of author names in catalog searches. *J. Libr. Autom.*, 1970, 3, № 2, 93—101
121. *Tanaka E.*, *Kasai T.*, Correcting method of garbled languages using ordered key letters. *Electr. & Commun. in Japan*, 1972, 55, № 6, 127—133
122. *Tarp A. L.*, *Kue-chung Tai*, The practicality of text signatures for accelerating string searching. *Software—Practice & Experience*, 1982, 12, 35—44
123. *Thorelli L. E.*, Automatic correction of errors in text. *Bit.*, 1962, 2, 45—65

124. Turba T. N., Checking for spelling and typographical errors in computer-based text. ACM Sigplan Notes, 1981, 16, № 6, 51—60
125. Ullman J. R., A binary n-gram technique for automatic correction of substitution, detection, insertion, and reversal error in words. Comp. J., 1977, 20, № 2, 141—147.
126. Van Nes F. L., Analysis of keying errors. Ergonomics, 1976, 19, № 2, 165—174
127. Wagner R. A., Fisher M. J., The string to string correction problem. J. ACM, 1976, 23, № 1, 13—16
128. Wong C. K., Chandra A. K., Bounds for the string editing problem. J. ACM, 1976, 23, № 1, 13—16 (ПЖМат, 1977, 9B755)
129. Yannakoudakis E. J., Expert spelling error analysis and correction. Inf. 7: Intelligent Inf. Retr. Proc. Conf. ASLIB Inf. Group & Inf. Retrieval, London, 1983, 39—50
130. —, Fawthrop D., The rules spelling errors. Inform. Proc. & Manag., 1983, 19, № 2, 87—99
131. —, —, An intelligent spelling error corrector. Inform. Proc. & Manag., 1983, 19, № 2, 101—108
132. Yianilos P. N., A dedicated comparator matches symbol strings fast and intelligently, Electronics, 1983, 56, № 24, 113—117
133. Zamora A., Automatic detection and correction of spelling errors in large data base. J. Amer. Soc. Inf. Sci., 1980, 31, № 2, 51—57
134. Zamora E. M., Pollock J. J., Zamora A., The use of trigram analysis for spelling error detection. Inform. Proc. & Manag., 1981, 17, № 6, 305—316

ВЫПУСКИ И ТОМА, ОПУБЛИКОВАННЫЕ РАНЕЕ

- |   |   |
|---|---|
| <p>Алгебра, Топология. «1962» (1964)*.</p> <p>Геометрия. «1963» (1965).</p> <p>Алгебра. «1964» (1966).</p> <p>Алгебра. Топология. Геометрия.</p> <p>«1965» (1967), «1966» (1968), «1967» (1969), «1968» (1970), «1969» (1970), «1970» (1972), тома 10 (1971), 11 (1974), 12 (1974), 13 (1975), 14 (1977), 15 (1977), 16 (1978), 17 (1979), 18 (1981), 19 (1981), 20 (1982), 21 (1983), 22 (1984), 23 (1985), 24 (1986), 25 (1987), 26 (1988).</p> <p>Проблемы геометрии. Тома 7 (1976), 8 (1977), 9 (1979), 10 (1978), 11 (1981), 12 (1981), 13 (1982), 14 (1983), 15 (1984), 16 (1984), 17 (1985), 18 (1987), 19 (1987), 20 (1988).</p> <p>Математический анализ. Теория вероятностей. Регулирование. «1962» (1964).</p> <p>Математический анализ. «1963» (1965), «1964» (1966), «1965» (1966), «1966» (1967), «1967» (1969), «1968» (1969), «1969» (1971), «1970» (1971), тома 10 (1973), 11 (1973), 12 (1974), 13 (1975), 14 (1977), 15 (1977), 16 (1978), 17 (1979), 18 (1980), 19 (1981), 20 (1982), 21 (1983), 22 (1984), 23 (1985), 24 (1986), 25 (1987), 26 (1988).</p> <p>Теория вероятностей. «1963» (1965).</p> <p>Теория вероятностей. Математическая статистика. Теоретическая кибернетика</p> | <p>«1964», (1966), «1966» (1967), «1967» (1969), «1968» (1970), «1969» (1970), «1970» (1971), тома 10 (1972), 11 (1974), 12 (1975), 13 (1976), 14 (1977), 15 (1978), 16 (1978), 17 (1979), 18 (1981), 19 (1982), 20 (1983), 21 (1983), 22 (1984), 23 (1985), 24 (1986), 25 (1987), 26 (1988).</p> <p>Современные проблемы математики. Тома 1 (1973), 2 (1973), 3 (1974), 4 (1975), 5 (1975), 6 (1976), 7 (1976), 8 (1977), 9 (1977), 10 (1978), 11 (1978), 12 (1978), 13 (1979), 14 (1979), 15 (1980), 16 (1980), 17 (1981), 18 (1981), 19 (1982), 20 (1982), 21 (1982), 22 (1983), 23 (1983).</p> <p>Современные проблемы математики. Новейшие достижения. Тома 24 (1984), 25 (1984), 26 (1985), 27 (1985), 28 (1986), 29 (1986), 30 (1987), 31 (1987), 32 (1988), 33 (1988).</p> <p>Современные проблемы математики. Фундаментальные направления. Тома 1 (1985), 2 (1985), 3 (1985), 4 (1985), 5 (1985), 7 (1985), 8 (1985), 9 (1986), 10 (1986), 11 (1986), 12 (1986), 13 (1986), 14 (1987), 15 (1987), 16 (1987), 17 (1988), 18 (1988), 20 (1988), 21 (1988), 22 (1988), 26 (1988), 27 (1988), 28 (1988), 29 (1988), 30 (1988), 31 (1988), 32 (1988), 33 (1988), 34 (1988).</p> |
|---|---|

\* Число в кавычках — название, в скобках — год издания.