Государственный комитет
РФ
по науке и технике

Академия наук
РФ

ВСЕРОССИЙСКИЙ ИНСТИТУТ НАУЧНОЙ И ТЕХНИЧЕСКОЙ ИНФОРМАЦИИ

на правах рукописи

ГЕЛЬБУХ
Александр Феликсович

УДК 801.73:681.3 (043.3)

ЭФФЕКТИВНО РЕАЛИЗУЕМАЯ МОДЕЛЬ МОРФОЛОГИИ ФЛЕКТИВНОГО ЕСТЕСТВЕННОГО ЯЗЫКА

Специальность — 05.13.17
Теоретические основы информатики

Диссертация на соискание ученой степени
кандидата технических наук

Москва 1994

A. Gelbukh. *An effectively implementable model of inflective natural language morphology* (in Russian). Ph.D. thesis, All-Russian Institute for Scientific and Technical Information (VINITI), Moscow, Russia, 1995. PDF.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

На современном этапе развития науки, техники и культуры процессы переработки и обработки информации выдвигаются на одно из ведущих мест в процессе общественного производства, пронизывая все сферы человеческой деятельности. Все большее значение приобретают методы и средства обработки информации на естественном языке — от простейших систем подготовки документов до информационно-поисковых систем, систем машинного перевода и программ общения с пользователем на естественном языке. Необычайно широк спектр приложений, так или иначе связанных с обработкой естественно-языковых текстов; столь же различна глубина их проникновения в структуру текста.

В зависимости от необходимой глубины проникновения в структуру текста указанные программы, как правило, оперируют блоками, соответствующими всей или, чаще, начальной части следующей последовательности обработки текста: буквы входного текста — слова — фразы — смысл — ... — новый смысл — структура фраз — значения слов — буквы выходного текста [93]. Первым (а также последним) шагом обработки естественноязыкового текста является блок, опознающий (анализирующий) и строящий (синтезирующий) различные формы слов — блок морфологического анализа и синтеза.

Актуальность проблемы морфологического анализа и синтеза словоформ определяется тем, что блок морфологического анализа является необходимой частью большинства работающих с естественноязыковыми текстами программ самого различного уровня и назначения; большинство таких систем нуждается также в блоке синтеза. Ввиду системного характера задачи и большого объема обрабатываемой информации к морфологическому блоку предъявляются жесткие требования по эффективности и быстродействию.

Задача заключается в том, чтобы разработать алгоритмы, методы и лингвистические модели, позволяющие автоматически осуществлять точный и полный морфологический анализ и синтез, а также решать ряд смежных задач, таких как нормализация слов, обучение пользователя грамматике, обнаружение и исправление грамматических ошибок и опечаток, интерпретация неправильных и незнакомых слов. К разрабатываемым алгоритмам предъявляется ряд более жестких, чем в известных алгоритмах, требований, в том числе — требование предельно высокой скорости работы и малого объема занимаемой оперативной памяти при работе на современных персональных ЭВМ, а также требование полного отделения программ от лингвистической информации в рамках модели, описывающей строение некоторого класса флективных языков.

Целью исследования в настоящей диссертации является разработка принципов, алгоритмов, программ и соответствующих лингвистических моделей, позволяющих создать эффективную по быстродействию и занимаемой оперативной памяти ЭВМ систему автоматического морфологического анализа, синтеза, нормализации слов, обнаружения и исправления ошибок, функционирующую на современных персональных ЭВМ и допускающую встраивание в интегрированные пакеты обработки текстовой информации.

Предметом исследования является (1) изучение морфологического строения флективных языков, в частности русского, в связи с задачей его формального описания в той мере, в какой это необходимо для построения программы автоматической морфологической обработки текста; (2) способы представления словаря и морфологической информации в связи с задачей ускорения доступа к хранящемуся на дисковом накопителе словарю; (3) алгоритмы морфологического анализа, синтеза, нормализации слов, обнаружения и исправления ошибок.

Научная новизна работы заключается в том, что автором впервые разработана структура словаря, позволяющая получить все гипотетические основы слова при предельно возможном быстродействии, то есть за одно элементарное обращение к дисковой памяти; разработана оригинальная языково-независимая (в некотором классе языков) модель морфологического строения флективного языка, основанная на разбиении словоформ на произвольное число равноправных в техническом отношении морфов; разработан метод исправления ошибок в тексте на флективном языке, превосходящий по быстродействию известные методы исправления ошибок данного класса; впервые предложен метод упорядочения процесса перебора альтернатив при исправлении опечаток, заключающийся в проверке гипотез в порядке возрастания времени, необходимого для каждой проверки.

Методы исследования. Исследование проводилось путем изучения закономерностей морфологического строения флективных языков, в первую очередь — русского; разработки конкретных морфологических таблиц и словаря для русского языка; изучения методов представления словаря на устройстве дисковой памяти и алгоритмов доступа к нему; разработки алгоритмов морфологического анализа, синтеза, нормализации слов, обнаружения и исправления ошибок; практической реализации этих алгоритмов на ЭВМ; статистической обработки результатов экспериментов.

Практическая значимость работы заключается в том, что в результате проведенных исследований создана библиотека процедур, осуществляющих автоматический морфологический анализ, синтез, нормализацию слов, обнаружение и исправление ошибок на персональной ЭВМ. Данное программное средство позволяет существенно повысить эффективность реализованных на персональных ЭВМ диалоговых поисковых систем и систем подготовки документов, а также служит инструментом дальнейшей лингвистической обработки текстов, такой как сбор различной статистики, поиск и выделение из текста фрагментов по различным условиям, синтаксический анализ и др.

Основные научные результаты:

- Разработана формальная модель морфологического строения флективного языка, позволяющая в некотором классе языков полностью отделить программы от лингвистических данных и допускающая эффективную реализацию на ее основе алгоритмов морфологического анализа, синтеза, нормализации слов, обнаружения и исправления ошибок.
- Разработана структура словаря, позволяющая достичь предельного быстродействия алгоритма поиска основ слов в словаре, а также алгоритмы поиска в таком словаре и алгоритм формирования словаря нужной структуры.
- Разработаны и практически реализованы на ЭВМ быстродействующие и экономичные по использованию оперативной памяти алгоритмы морфологического анализа, синтеза, нормализации слов, обнаружения и исправления ошибок.
- Предложен новый метод упорядочения процесса перебора альтернатив при исправлении ошибок, заключающийся в генерировании в первую очередь тех гипотез, проверка которых требует минимального времени.
- При участии автора разработан и реализован на ЭВМ интерфейс стандартной библиотеки морфологических процедур.
- При участии автора создана система морфологических таблиц, описывающая строение русского языка в рамках разработанной автором модели.
- При участии автора создан машинный морфологический словарь русского языка (на основе известных источников), включающий около 130 тыс. лексем общеупотребительной и специальной лексики.

Реализация результатов работы. На основании разработанных алгоритмов, программ, лингвистического обеспечения и словаря создана стандартная библиотека процедур морфологического анализа, синтеза, нормализации слов, обнаружения и исправления ошибок. Имеется опыт интеграции данной библиотеки в реализованную на персональной ЭВМ информационно-поисковую систему. Имеется также опыт реализации разработанного алгоритма исправления опечаток в системе грамматической проверки текста, основанной на иной морфологической модели.

Апробация работы. Основные научные результаты работы представлялись на конкурс работ молодых ученых ВНТИЦентра (1990; III место), докладывались автором на IV Всесоюзной школе-семинаре при Институте кибернетики АН УССР, конференции «Программное обеспечение новой информационной технологии», III Международной конференции «Программное обеспечение ЭВМ», 1-й Ежегодной Всесоюзной конференции SUUG, конференции «Использование программных средств ПЭВМ для автоматизации учрежденческой деятельности», Международном форуме «Тех-Екс'90 — обмен технологиями» (Болгария), на научно-технических семинарах сектора ПО баз знаний отдела автоматизации информационных процессов ВНТИЦентра и отдела ОТОИ ВИНИТИ.

Публикации. По теме диссертации опубликовано 11 работ — 3 статьи, тезисы докладов, технические отчеты.

Структура и объем работы. Диссертационная работа состоит из введения, четырех глав и приложений.

В первой главе обосновывается необходимость совершенствования методов морфологической обработки текстов на естественном языке. Дается обзор существующих подходов к решению проблемы, применяемых методов и разработанных к настоящему времени морфологических систем, анализируются их достоинства и недостатки. Формулируются задачи исследования.

Во второй главе излагается разработанный автором метод организации морфологического словаря, позволяющий при морфологическом анализе получать всю необходимую для анализа слова информацию при одном обращении к дисковой памяти. Приводятся алгоритмы формирования словаря и поиска информации в нем.

Третья глава является центральной в работе. В ней излагается разработанная автором языково-независимая в некотором классе естественных языков формальная лингвистическая модель морфологического строения естественного языка, а также структура разработанной автором морфологической системы, основанной на данной модели. Приводятся алгоритмы морфологического анализа, синтеза, нормализации слов, обнаружения, объяснения и исправления ошибок.

В четвертой главе отдельно рассматривается разработанный автором алгоритм исправления опечаток, примененный, в частности, в описываемой морфологической системе. Показывается, что по эффективности данный алгоритм превосходит применяемые в настоящее время. Приводится несколько вариантов алгоритма.

ГЛАВА 1.
ПРОБЛЕМА МОРФОЛОГИЧЕСКОЙ ОБРАБОТКИ ТЕКСТОВ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

1.1. Место и роль морфологической подсистемы в системах обработки текстов на естественном языке

В настоящее время в самых различных областях информатики возникает потребность в решении задач, связанных с морфологическими преобразованиями слов естественного языка. К таким задачам относятся, прежде всего, морфологический анализ и синтез словоформ, а также обнаружение и исправление ошибок в текстах, нормализация слов, автоматизированное определение словарных характеристик новых слов и некоторые другие задачи. В большинстве случаев решение с высокой точностью задач данного класса сводится, в том или ином виде, к решению задач морфологического анализа и синтеза.

Потребность в морфологических операциях над текстом возникает в системах самого различного масштаба и назначения, обрабатывающих текст на естественном языке. Достаточно назвать задачи автоматического и автоматизированного перевода, извлечения информации из текста, автоматического индексирования баз данных в информационно-поисковых системах, сжатия текстовых баз данных, проверки грамматической правильности текста, создания электронных словарей и обучающих систем, проведения лингвистических исследований, и многие другие задачи.

Под морфологическими операциями понимаются операции над текстом, связанные с явлением словоизменения (более точно — морфологического словоизменения), то есть образования различных форм слов. Образование форм слов свойственно подавляющему большинству крупных языков мира, в частности, всем европейским языкам. Даже в языках со слабо развитой системой словоизменения большая часть слов образует несколько различных форм, называемых словоформами данного слова. Например, англ. *book — books — book's — books', ask — asks — asked — asking*. В русском же языке типичное существительное образует 12 форм, прилагательное — 40, глагол — 225 форм. Например: *стол — стола — столами*; *красный — красная — красны*; *читать — читающимися — читая*. Уточним, что число форм, образуемых в русском языке лексемами разных частей речи, может быть оценено по-разному, и приведенные здесь цифры носят лишь иллюстративный характер.

Однако во многих случаях конкретная буквенная цепочка, напр. *столами*, не может быть непосредственно использована при обработке текста. Так, в информационных системах любая информация обычно привязана к собственно слову, лексеме, а не к отдельной словоформе. Однако в текстах, обрабатываемых системой, слова представлены в виде различных словоформ. Первой задачей морфологического анализатора является сведение всех форм слова к единому виду, представленному либо числом, либо стандартизированной буквенной цепочкой.

Пусть, например, с понятием `стол' в некоторой базе данных связана некоторая информация, скажем, количество или цена. Хранение в базе данных отдельных записей для каждой из форм слова *стол* не является приемлемым. Очевидно, это означало бы 12-кратное дублирование информации. Кроме того, по такой базе данных было бы невозможно установить, сколько *столов* имеется на складе, поскольку цепочка *столов* не совпадает с цепочкой *стол*. По той же причине невозможно извлечение полезной для такой базы информации из текста, сообщающего, что на складе имеется три *стола*.

Задача обработки форм слов не является тривиальной, поскольку хранение в явном виде всех таких форм, а также информации, сопутствующей каждой форме в отдельности, не является приемлемым в случае русского языка ввиду огромного количества таких форм. Так, из-за большого размера массива всех словоформ, составленного по словарю [69], скорость поиска и выборки информации из такого массива оказывается недостаточной. Следовательно, морфологическая система должна «вычислять» различные формы слов. Более того, наличие в системе правил изменения слов играет принципиальную роль при решении некоторых задач морфологической обработки текста. Так, при исправлении грамматических ошибок существенна правильная интерпретация системой неграмматичных форм, например, **хочут*. Не менее важной является задача автоматизированного определения словарных характеристик слов текста, отсутствующих в системном словаре, что также является возможным только при наличии в системе некоторых правил словоизменения.

Такие правила действительно существуют в естественном языке и могут быть сформулированы в виде, необходимом для автоматического образования словоформ. Например, для слов типа *стол, сад*

правило состоит в присоединении к соответствующей цепочке справа буквенных цепочек *-а, -у, ..., -ах*. Соответственно, правило автоматического определения слова по словоформе состоит в отделении такой цепочки от словоформы. Однако полный набор подобных правил для русского языка образует сложную систему. Например, наборы окончаний для слов *стул*, *стог*, *круг* отличаются от приведенного выше, ср.: *стол-ы, стул-ья; стог-а, круг-и*. Кроме того, слова других частей речи присоединяют несколько суффиксов, ср.: *сильн-ейш-ий, чит-а-ющ-ий-ся*, причем не все сочетания таких суффиксов возможны для конкретного слова.

Языки с подобным типом словоизменения называются флективными. Существует ряд языков, обладающих более простым — агглютинативным — типом словоизменения, однако в рамках настоящей работы такой тип может формально рассматриваться как упрощенный частный случай флективного.

Таким образом, морфологическая система должна обладать моделью морфологического строения данного естественного языка. Кроме того, для описания свойств отдельных слов система должна обладать словарем большого объема. Однако для практически работающей морфологической системы существенным является не столько само наличие такого аппарата, сколько высокоэффективная программная реализация алгоритмов, решающих на его основе разнообразные задачи обработки текста. Как правило, морфологической обработке подвергается каждое слово текста, то есть каждая группа в среднем из восьми букв. Следовательно, большое значение имеет высокое быстродействие соответствующих алгоритмов.

Наибольшую трудность при программной реализации представляет быстрый доступ к информации, хранимой в виде крупного морфологического словаря. При морфологическом анализе, то есть автоматическом распознавании словоформ, известную трудность представляет также комбинаторный характер задачи, приводящий к необходимости перебора различных вариантов разбиения буквенной цепочки. Например: *сильнейший-*, сильнейши-й, сильнейш-ий, сильн-ейш-ий*. Здесь знаком * обозначена пустая цепочка. При рассмотрении различных вариантов разложения анализируемой буквенной цепочки многие существующие системы производят несколько обращений к дисковой памяти. В главе 2 нами показана возможность получения всей информации, необходимой для анализа буквенной цепочки, при одном обращении к дисковой памяти.

Морфологическая обработка текста включает в себя решение всех или некоторых из следующих задач. Морфологическим анализом называется установление по словоформе исходного слова — лексемы, а также морфологических характеристик данной словоформы, таких как род, падеж, число и т.д. Морфологическим синтезом называется задача обратного сопоставления — нахождения формы заданного слова по заданным падежу, числу и т.д. Автоматическим обнаружением орфографической ошибки называется установление существования или несуществования данной словоформы в языке. Автоматизированным исправлением ошибки называется нахождение по ошибочной словоформе неошибочных словоформ, близких к данной в некотором определенном смысле.

В области морфологической обработки текстов на естественном русском языке рядом исследователей достигнуты значительные результаты. Крупные практически эксплуатируемые системы созданы, в частности, научными коллективами под руководством Г.Г.Белоногова, И.А.Большакова, Ю.Д.Апресяна, М.Г.Мальковского, О.С.Кулагиной. Обзор методов, разработанных авторами этих систем, дан в пунктах 1.2.1 — 1.2.4.

1.2. Состояние вопроса

1.2.1. Методы и системы морфологического анализа

Задаче морфологического анализа словоформ естественного языка посвящено множество исследований и практических разработок. Такие исследования проводились, в частности, в рамках работ по созданию систем автоматического перевода [50, 81, 84], автоматизированных информационных систем [43, 52, 99], систем орфографического контроля и коррекции текста [8, 17, 44, 155]. Созданные в настоящее время системы морфологического анализа различаются как применяемыми методами анализа, так и особенностями реализации.

Наиболее распространенными являются следующие два подхода к задаче морфологического анализа, которые условно могут быть названы анализом «справа налево» и «слева направо».

В первом случае слово предварительно выделяется из контекста, далее без участия словаря от него отсекается конечный набор аффиксов и таким образом выделяется основа слова, и затем

производится попытка поиска такой основы в словаре основ. Данный способ является, по-видимому, более распространенным. Более широко, к данному типу могут быть отнесены любые методы, использующие бессловарное приведение слова к виду, допускающему поиск определенной цепочки в словаре.

Во втором случае в словаре основ ищется предполагаемая основа анализируемого слова, а затем на основании полученной словарной информации производится попытка интерпретации оставшейся правой части слова как набора аффиксов. В пункте 2.1.1 обосновывается целесообразность именно такой модели для выполненной автором разработки.

Из существующих систем морфологического анализа одной из наиболее интересных является система, разработанная в ВИНИТИ под руководством проф. Г.Г.Белоногова [18, 20, 32]. Многолетний опыт практической эксплуатации данной системы показал ее высокую эффективность [11, 28]. Система разработана в рамках решения задачи информационного поиска [10, 14, 15, 25, 36]. В системе используется принцип анализа «справа налево». Алгоритм морфологического анализа в системе проф. Г.Г.Белоногова включает следующие этапы [19, 20, 26]:

Разбиение текста на слова. Словом считается только последовательность русских букв и дефисов. Разбиение производится априорно, без привлечения лингвистической информации. В некоторых случаях, например, при наличии дефиса, разбиение может быть пересмотрено в процессе словарного анализа. Специальных мер по обработке переноса слов не предпринимается.

Сопоставление со словарем коротких и служебных слов. Данный небольшой по размеру словарь содержит полные формы слов, как правило, неизменяемых. Сопоставление с таким списком осуществляется значительно быстрее, чем анализ слова по общему алгоритму. Если слово найдено в данном словаре, его анализ на этом завершается. Фактически данный словарь содержит следующие классы слов. Во-первых, короткие слова, обработка которых принятым в системе общим алгоритмом анализа невозможна. Особая обработка таких слов является специфичной для данной системы и определяется спецификой применяемого в системе общего алгоритма анализа. Во-вторых, наиболее часто встречающиеся в тексте слова. Особая обработка таких слов является типичной для систем морфологического анализа и значительно ускоряет процесс анализа (см. пункт 3.3.3). В-третьих, слова с особыми признаками, которые должны быть исключены из обработки общим алгоритмом. Например, цепочки *AT* и *THE* как английские слова. Фактически данная часть словаря является словарем так называемых стоп-слов (информационно считающихся пустыми).

Приближенное выделение основ слов. На данном этапе от буквенной цепочки отделяется справа набор аффиксов и выделяются возможные основы. Одному слову может соответствовать несколько гипотетических основ. При дальнейшем сопоставлении со словарем каждая из таких гипотез подвергается проверке. В системе применяются методы, позволяющие до обращения к словарю существенно снизить число гипотетических основ слова; как правило, слову соответствует только одна гипотетическая основа. Для этого используется сформулированная проф. Г.Г.Белоноговым концепция наличия в ряде естественных языков, в частности, в русском языке, сильной корреляции между грамматическими признаками слов и буквенным составом их концов [22, 23]. В специальном списке пятибуквенных концов слов определяется элемент, наилучшим образом совпадающий с конечной частью анализируемого слова. Анализируемому слову приписываются только гипотетические основы, соответствующие найденной пятибуквенной финали.

Точный морфологический анализ. Для каждой найденной на этапе приближенного выделения основ гипотетической основы производится поиск в словаре основ. Наличие информации об основе как об определенной буквенной цепочке позволяет применять мощные методы ускорения поиска в словаре, в частности, хеширование. Поиск также существенно ускоряется благодаря методу «слияния» текста со словарем, заключающемуся в том, что входной массив словоформ предварительно упорядочивается по гипотетическим основам, а затем «сливается» (в смысле [76]) с аналогично упорядоченным словарем. Если основа слова обнаружена в словаре, по приписанной ей словарно грамматической информации и по таблицам окончаний производится проверка возможности присоединения к данной основе данного окончания и определяются грамматические характеристики словоформы, после чего анализ данной словоформы завершается. Изложенный метод является особенностью пакетной обработки текста и неприменим при последовательной обработке слов.

Приближенный морфологический анализ. Если ни одна из гипотетических основ слова не обнаружена в словаре, слову назначаются грамматические характеристики, приписанные пятибуквенной финали, найденной на этапе приближенного выделения основы. Затем слово обрабатывается так, как если бы соответствующая основа была найдена в словаре. Данный прием также

основан на сформулированном проф. Г.Г.Белоногoвым принципе корреляции между грамматическими характеристиками слов и буквенным составом их концов [13, 24]. Системой отдельно обрабатывается информация о наличии в словоформе частицы *-ся*/*-сь*.

В рамках программно-лингвистического комплекса реализованы подсистемы, выполняющие ряд операций над текстом, например, нормализацию слов [12, 30, 106]. Подсистема нормализации слов реализована в виде отдельного алгоритма и использует дополнительный набор лингвистических данных [94, 105].

Таким образом, к особенностям реализации описываемой системы можно отнести, с одной стороны, высокое быстродействие, высокую степень полноты анализа, высокую, хотя и не полную, степень отделения лингвистической информации от алгоритмов, возможность приближенного морфологического анализа, реализацию элементов словообразовательного анализа [29]. С другой стороны, к особенностям системы относятся использование значительного объема оперативной памяти ЭВМ, отсутствие возможности последовательной, пословной обработки входного буквенного потока, ограниченные возможности морфологического синтеза, ограниченные возможности интерпретации ошибочных слов и автоматического исправления ошибок.

Примером подхода, основанного на полностью алгоритмическом и, следовательно, языково-зависимом представлении информации, является система, реализованная в ИВАН проф. С.А.Старостиным в рамках автоматизированного рабочего места лингвиста STAR. Словарем системы является непрепарированный текст словаря [69]. Алгоритмически реализована интерпретация словарных помет, используемых в [69], что, в частности, дает возможность синтеза всех форм каждого включенного в словарь слова [70]. При анализе буквенной цепочки по заложенным в систему правилам строятся все возможные варианты исходной формы данного слова, то есть без обращения к словарю решается задача нормализации слова. Далее для каждого построенного варианта производится попытка поиска в словаре [69]. В случае, если данная исходная форма найдена в словаре, проверяется, есть ли среди всех форм данного словарного слова формы, совпадающие с анализируемой цепочкой.

Обширный набор использованных при создании системы правил обеспечивает высокую точность приближенного анализа отсутствующих в словаре слов и синтеза различных их форм, а также широкие возможности интерпретации неграмматичных форм слов и объяснения и исправления грамматических ошибок (но не случайных опечаток). Особенностью системы является полный учет ударения. При реализации системы не ставилась задача ее оптимизации, и, в частности, задача достижения высокого быстродействия.

К достоинствам примененного С.А.Старостиным процедурного подхода следует отнести широкий набор учитываемых системой лингвистических явлений; гибкость и естественность описания как правил, так и исключений из них и особенностей изменения отдельных групп слов; высокую точность приближенного анализа и синтеза; широкие возможности применения к задаче обучения языку. К недостаткам системы С.А.Старостина могут быть отнесены низкое быстродействие, языковая зависимость, затрудненность модификации лингвистической информации, невозможность использования имеющейся в системе лингвистической информации для решения других задач обработки текста, например, задачи исправления случайных опечаток.

Комплексная морфологическая система разработана на факультете ВМК МГУ коллективом исследователей под руководством М.Г.Мальковского в рамках создания системы искусственного интеллекта TULIPS-2 [86]. Система осуществляет широкий набор морфологических операций. Так, кроме анализа и синтеза реализованы интерпретация и исправление как случайных опечаток, так и грамматических ошибок. Морфологическая подсистема системы TULIPS-2 функционирует во взаимодействии с другими блоками системы [87], что накладывает отпечаток на общую структуру реализации подсистемы [89].

В системе реализован принцип анализа «справа налево», при котором от анализируемой цепочки сначала отделяются постфиксы, и затем по каждому варианту производится обращение к словарю основ [88]. Никакие специальные процедуры доступа к словарю не применяются, вместо этого для хранения словаря и выборки информации из него используется система управления базой данных, предоставляемая встроенными механизмами языка реализации — оператором GET языка ПЛЭНЕР-БЭСМ. Для ускорения анализа могут применяться так называемые предсказания — априорные сведения об ожидаемых свойствах анализируемой словоформы. Такие предсказания могут генерироваться как самим анализатором по словарной информации, полученной при анализе ранее обработанных словоформ, так и другими блоками системы TULIPS-2 с учетом семантических, прагматических и др. факторов.

В лингвистической модели системы изменяемые слова в общем случае рассматриваются как состоящие из основы, окончания и, возможно, постфикса, например, частицы *-ся*/*-сь*. Словарными свойствами основ слов являются номер *словоизменительного класса*, обобщающего понятие части речи, номер *парадигматического класса*, обобщающего понятие набора окончаний, номер *чередования*, задающего алгоритм изменения отдельных букв основы, и номер *исключения*, описывающее некоторые особенности изменения данного слова. При каждой основе в словаре указаны также значения ее словарных характеристик, таких как род существительного или переходность глагола.

В набор перечисленных выше стандартных объектов морфологической модели включаются только объекты (классы, чередования, исключения), соответствующие большому количеству словарных слов. Уникальные и редко встречающиеся чередования трактуются в системе наряду с явлением супплетивизма основ. Для каждого такого случая в словарь заносится отдельная основа (например, *зай-*, *зайд-*, *заш-* для *зайти*). Кроме морфологической информации, словарь системы включает для каждого слова синтаксическую, семантическую и иную необходимую информацию.

Тесное взаимодействие с различными блоками системы искусственного интеллекта TULISP-2 является как сильной, так и слабой стороной морфологической подсистемы. В частности, в выполненной реализации она не может использоваться в качестве автономной морфологической системы. Использование для организации доступа к словарным данным встроенной системы управления базой данных языка ПЛЭНЕР-БЭСМ, несомненно, оправданное при реализации морфологического блока системы TULISP-2, также оставляет за рамками рассмотрения вопросы вычислительной эффективности алгоритма морфоанализа.

Существенным моментом при реализации морфологического анализатора словоформ флективного языка, в отличие от случая агглютинативных языков, является описание правил чередования букв в основах слов. Во многих системах, ориентированных на достижение высокого быстродействия при анализе, правила чередования не рассматриваются [1, 83, 90]. Вместо этого в словарь заносятся все формы основы слова. Например: *сильн-*, *силен-*; *глубок-*, *глубоч-*, *глубж-*. К недостаткам такого метода относятся как дублирование информации и увеличение объема словаря, так и фактическое отсутствие в системе информации о правилах чередования букв. Впрочем, увеличение объема словаря может компенсироваться использованием специального формата хранения словарных статей. Отсутствие же информации о чередованиях может рассматриваться и как достоинство системы, поскольку приводит к значительному сокращению объема таблиц, упрощению алгоритмов и снижению трудоемкости составления лингвистического обеспечения.

Другим возможным способом исключения из рассмотрения чередований является перенесение записи явного вида буквенных цепочек не в основы, а в наборы окончаний (точнее — финалей) лексем. Так, слову *сильный* при этом соответствует одна основа *сил-* и набор окончаний *ьный, ьного, ..., ен, ьна, ьно, ьны*; супплетивные лексемы типа *человек*/*люди* имеют пустую словарную основу. Достоинством метода является простота алгоритмов, к недостаткам относится большой объем хранимых в оперативной памяти таблиц финалей: так, в выполненной с участием автора (вне рамок настоящей работы) реализации объем грамматических таблиц, построенных по словарю на базе словаря [69], составил около ста килобайт.

Третьим возможным подходом является описание чередований в виде правил или алгоритмов. При этом невозможным становится анализ «слева направо», поскольку для нахождения гипотетических основ необходимо до обращения к словарю произвести отсечение окончаний и применить правила чередования. К достоинствам метода относится естественность и «лингвистичность» описания рассматриваемых явлений. К недостаткам данного метода относится большая сложность правил чередования букв и большое количество возникающих в процессе анализа ложных гипотетических основ. Упомянутые два метода могут комбинироваться.

Описание чередований в виде правил, однако, является более естественным для систем, осуществляющих синтез, а не анализ словоформ. Одним из методов анализа, не получившим, однако, широкого распространения, является «анализ через синтез» [5]. Данный прием применяется в тех случаях, когда правила чередований и синтеза форм записываются в системе в виде, не допускающем преобразование их в эффективно реализуемый алгоритм разложения словоформы при анализе. В этом случае для каждой лексемы, гипотетически порождающей данную словоформу, синтезируются все ее формы, из них выбираются те, которые совпадают с анализируемой словоформой, и таким образом определяются грамматические характеристики исходной словоформы.

Среди зарубежных разработок наибольшее распространение получила разработанная финским исследователем Киммо Коскенниеми система KIMMO [113, 115, 133], получившая также название

двухуровневой модели [148]. На базе разработанной К. Коскенниеми модели созданы как инструментальные средства, позволяющие без дополнительного программирования разрабатывать описания морфологической структуры естественного языка [124, 134, 145, 160, 167], так и библиотеки процедур, позволяющие использовать морфологические процедуры в прикладных программах на языках Си [113], Лисп [125], Пролог [119].

Система является языково-независимой в широком классе языков различной структуры. Так, экспериментальные реализации морфологических процессоров выполнены в рамках KIMMO-модели для японского [110], арабского [116], шведского [117], немецкого [126, 164], финского [129, 147, 149], английского [135, 159], аккадского [140], румынского [144], церковнославянского [151], французского [152], а также иврита, тагальского, турецкого и некоторых других языков [113]. Реализации для некоторых языков потребовали введения в исходную модель дополнительных возможностей [116, 126, 142]. Для некоторых языков были реализованы описания только отдельных подсистем морфологического строя, например, только имен или только глаголов [126], либо были опущены периферические явления [113]. Некоторые реализации посвящены мертвым или малораспространенным языкам, что повлекло сильное упрощение описания [140, 151].

В системе последовательно реализован взгляд на описание морфологии естественного языка как на формально-математическую задачу [111, 166], что сделало возможным, в частности, применение чисто математического аппарата при разработке модели [130] и оценке ее вычислительной эффективности [114]. В частности, морфологический анализатор и синтезатор в KIMMO-модели представляют собой преобразователи, работающие по принципу конечного автомата [131, 132].

KIMMO-модель представляет собой разновидность контекстно-зависимой формальной грамматики. Подобные модели, более подробно рассмотренные в пункте 1.2.3, широко использовались при создании систем морфологического синтеза [34, 35], однако использование их для решения задачи анализа затрудняется их низкой вычислительной эффективностью. К. Коскенниеми наложил ряд ограничений на вид применяемых правил преобразований и на алгоритм их интерпретации. Введенные ограничения позволили программно реализовать интерпретатор правил в виде конечного автомата, что повысило вычислительную эффективность анализатора.

Основные отличия формальной грамматики, используемой в KIMMO-модели, состоят в следующем. Во-первых, используется только двухуровневое представление данных, соответствующее исходному тексту и результату анализа, в отличие от многоуровневых моделей. Во-вторых, в описание контекста в правилах входят в произвольной смеси символы обоих уровней, в отличие от моделей, допускающих строго поуровневое преобразование текста. В-третьих, что, по-видимому, наиболее существенно, правила рассматриваются как неупорядоченное множество и применяются не поочередно, а все одновременно. Последнее обстоятельство делает набор правил преобразования текста симметричным относительно обоих уровней, то есть относительно операций анализа и синтеза словоформ. Таким образом на базе одного и того же набора лингвистических правил и одной и той же программы, интерпретирующей эти правила, строится как морфологический анализатор, так и программа синтеза.

В настоящее время многие исследования посвящены расширению и модификации исходной KIMMO-модели. Так, разработаны методы описания морфологической структуры языка, представляющие слово не в виде цепочки атомарных символов, а в виде набора более сложно структурированных объектов, что упрощает составление правил преобразований [164]. Некоторые исследователи рассматривают правила более сложной и гибкой структуры [118]. Описание морфологии высокофлективных и в особенности интрофлективных [101], напр. семитских, языков, потребовало расширения двухуровневой модели, например, усложнения алгоритмов [116] или введения дополнительных уровней [142].

KIMMO-модель является интересной в чисто лингвистическом плане, поскольку она не отражает непосредственно бытующие в настоящее время представления о природе морфологии различных языков, а также не является языком описания трансформаций текста общего вида. Таким образом, KIMMO-модель несет в себе нетривиальное утверждение об определенной природе механизма словоизменения в описываемом языке, не совпадающее с общепринятыми представлениями. Опыт составления описаний для различных языков показал приложимость представлений о природе словоизменения, заложенных в систему KIMMO, к широкому ряду естественных языков различных типов. Данный факт до сих пор не нашел теоретического обоснования.

К достоинствам KIMMO-модели относится прежде всего высокая степень языковой независимости при большей, чем в других моделях, простоте составления системы правил

преобразований. Языковая независимость и основанные на строгих и хорошо известных программистам математических конструкциях механизмы реализации способствовали разработке различных версий эффективного программного обеспечения системы программистами разных стран на базе различных языков программирования. Наличие же эффективных программных средств способствовало разработке различных версий лингвистического обеспечения для многих языков мира. Таким образом, на базе модели KIMMO возник уникальный в лингвистической практике феномен широко распространенной стандартной языково-независимой морфологической системы с развитым набором программных средств и хорошо изученным корпусом лингвистических описаний.

Необходимо, однако, заметить, что языковая независимость исходной KIMMO-модели ограничена. Как указывалось выше, исходная модель оказалась недостаточной для описания морфологии многих языков, в частности, таких высокофлективных языков, как русский. Более того, модель оказывается недостаточной для описания всех фактов морфологии живого естественного языка, включая периферические. В связи с этим системы, построенные на базе KIMMO-модели, нашли применение скорее как чисто лабораторный инструмент лингвистического исследования, чем как эффективная практически применимая подсистема прикладных программ.

Быстродействие основанных на KIMMO-модели систем также не является предельно возможным для морфологической системы. По-видимому, это связано со слишком высокой степенью общности модели. По существу, KIMMO-модель является не только языково-независимой, но и «проблемно-независимой» в том смысле, что лежащий в ее основе принцип преобразования текста [111, 132] не является в достаточной степени специфически морфологическим, а отражает гораздо более широкий класс преобразований. Последний факт затрудняет также и составление лингвистического обеспечения.

В связи с этим возникает важный в методологическом плане вопрос об оптимальной степени общности и языковой независимости математической модели, лежащей в основе программной реализации морфологической системы. Как правило, в рамках одной морфологической системы составляются или могут быть составлены различные массивы лингвистической информации, хотя бы в процессе составления и отладки лингвистического обеспечения. Как показал опыт разработки предлагаемой автором модели, реализуемые при этом взгляды на конкретный естественный язык, скажем русский, оказываются столь различны, что фактически речь всегда идет о реализации описания разных языков, принадлежащих, однако, к некоторому определенному классу, например, классу флективных языков. По-видимому, программная реализация морфологической системы должна быть языково-независимой в данном, достаточно широком, классе, однако должна учитывать особенности данного класса и особенности морфологической обработки текста по сравнению с произвольным его преобразованием. Более общо, среди всех возможных приложений данной программной реализации должны быть выделены общие черты, учитываемые при разработке реализуемой математической модели. KIMMO-модель, по-видимому, является примером недостаточно оправданного увеличения общности математической модели.

Другим развиваемым в настоящее время зарубежными исследователями подходом являются модели так называемой парадигматической морфологии. В рамках этого подхода основное внимание уделяется определению иерархической системы морфологических классов и описанию механизмов наследования подклассами свойств надклассов [121]. Подобный подход практически реализован, например, в системе MOLUSC [120]. Данный подход является удобным при описании высокофлективных языков, например, латинского. Предпринимались также попытки использования данного подхода для описания морфологии некоторых славянских языков.

1.2.2. Методы составления словника, сжатия словаря и алгоритмы поиска в словаре

Вопрос составления и использования словаря имеет два различных аспекта: во-первых, составление и обновление словника словаря и, во-вторых, техническая реализация хранения словаря и алгоритмы выборки информации из машинного вида словаря.

Задача составления и обновления словника словаря выходит за рамки чисто научной или технической задачи. В технологический процесс составления и обновления словарей крупных информационных систем оказываются, как правило, вовлечены целые коллективы сотрудников [85]. В нашей стране создан и функционирует Машинный фонд русского языка — национальная лингвистическая служба, в задачи которой входит, в частности, составление и постоянное обновление словарей [4, 79, 98]. Сопровождение словарей требует разработки специальных, подчас сложных,

программных инструментальных средств [73, 74], а также лингвистических методов [11, 102]. Так, в ВИНИТИ под руководством проф. Г.Г.Белоногова создана специальная словарная служба [11], включающая подсистемы лингвистической обработки машинных словарей [9, 68], морфологического и синтаксического анализа. Последние две подсистемы используются при автоматическом извлечении новых слов из текстов по некоторой предметной области. Важной задачей при составлении специальных словарей является оптимальный выбор включаемых в словарь слов, например, отбор только стоп-слов [97] или только терминов определенной предметной области [102]. Задача извлечения из текста значимых терминов может решаться как автоматизированно, так и чисто автоматически на базе статистических методов, как, например, в системе MOSAIC [146].

Методы хранения словарей в машинной форме, а также алгоритмы поиска информации в них и ее выборки во многом определяют эффективность реализации морфологической системы, в особенности в режиме морфологического анализа. При выборе способа кодирования крупного словаря учитывается, с одной стороны, необходимость минимизации занимаемого им объема дисковой памяти, и с другой стороны, необходимость минимизации времени поиска информации и ее выборки.

Указанные два требования — увеличения быстродействия и уменьшения занимаемого объема — как правило в практике программирования считаются противоречивыми. Однако в случае очень больших дисковых файлов, в частности, в случае очень крупного словаря, наблюдается не обратная, а прямая зависимость между временем поиска информации в словаре и его объемом. Это объясняется увеличением времени доступа к определенному участку файла с ростом его объема, а также уменьшением эффективности метода кеширования при доступе к данному файлу. Это обстоятельство определяет недопустимость хранения в словаре всех словоформ лексем в явном виде.

Различными исследователями как в нашей стране, так и за рубежом разработаны методы организации хранения словаря [60, 82], в частности экономного кодирования словарной информации [46, 153] и сжатия текста словаря [41, 150]. Простой и эффективный метод сжатия алфавитно упорядоченного словаря указан У. Купером [123].

Основными способами представления словарей являются представление в виде алфавитно упорядоченного списка либо в виде дерева, набора из двух «встречных» деревьев или более сложного графа [38, 122, 128]. Такие способы организации информации, как, например, B-деревья, применяемые при организации баз данных общего назначения, сравнительно редко используются в морфологических системах, поскольку не обеспечивают достаточно эффективной выборки информации. Подобные способы хранения обеспечивают эффективное добавления и удаления информации, что, однако, не является столь же существенным для морфологической системы, как поиск и выборка информации. На выбор способа представления словаря влияет также необходимость обеспечения возможности поиска по искаженному ключу при решении задачи исправления опечаток [128]. Наряду со специальными способами хранения словаря разработаны различные алгоритмы выборки информации из словаря, учитывающие специфику задачи выборки информации по ключу в виде буквенной цепочки [51, 122, 163].

Словарь морфологической системы располагается, как правило, на устройстве дисковой памяти. В некоторых системах словарь располагается в оперативной памяти ЭВМ, что упрощает алгоритмы доступа к нему и многократно ускоряет поиск. Однако для размещения словаря в оперативной памяти ЭВМ необходима крайне высокая степень сжатия его информации. При этом также ограничены объем словника и объем информации, хранящейся в словарных статьях. Хранение словаря в оперативной памяти ЭВМ налагает ограничения на характеристики используемого оборудования. Кроме того, система, требующая размещения словаря в оперативной памяти ЭВМ, как правило, не может использоваться в качестве подпрограммы какой-либо прикладной программы, также использующей значительный объем оперативной памяти. Затрудняется также работа многозадачных операционных систем, таких как OS/2, Windows, UNIX. По изложенным соображениям хранение большей части словаря в оперативной памяти представляется нам скорее неудачным решением, во всяком случае при использовании распространенных в настоящее время аппаратных средств и операционных систем.

На практике используются различные методы кеширования словаря, а также методы организации словаря, предполагающие хранение в оперативной памяти небольшого набора наиболее часто употребляемых в тексте слов. Однако при этом быстродействие системы в целом определяется скоростью выборки информации из той части словаря, которая все же хранится на диске. Таким образом, все наши построения прямо переносятся на этот случай в применении к размещенной на диске основной части словаря.

Как правило, морфологические словари ориентированы на пословный поиск, другими словами, на последовательный поиск информации о словах, поступающих в порядке появления их в тексте. Другим способом организации выборки информации из словаря является метод «слияния» словаря с предварительно отсортированным в алфавитном порядке массивом всех слов, имеющихся в достаточно большом анализируемом текстовом документе. Данный способ значительно упрощает структуру файла словаря и снижает требования к быстродействию алгоритма выборки информации из словаря, при высоком быстродействии системы в целом в среднем на одно входное слово. Такой метод применен, в частности, в системе, созданной под руководством проф. Г.Г.Белоногова [20, 21]. Однако данный метод применим только при обработке информации в пакетном режиме, не рассматриваемом в настоящей работе.

1.2.3. Методы и системы морфологического синтеза

Необходимость осуществления морфологического синтеза возникает при решении ряда задач, в частности, задач автоматического перевода, нормализации слов, исправления ошибок, обучения языку и др. Как правило, в рамках одной прикладной задачи возникает необходимость осуществления как морфологического анализа, так и синтеза. В связи с этим многие комплексные морфологические системы, в том числе почти все системы, рассмотренные в пункте 1.2.1, содержат подсистемы морфологического синтеза.

В системах морфологического анализа и синтеза по возможности используется единый набор лингвистической информации. Так, если системе известен набор окончаний для слов какого-либо парадигматического класса, например, *стол-*, -а, -у, ..., -ах*, и анализ производится путем отделения от словоформы таких окончаний, то синтез производится путем присоединения к основе слова этих окончаний. При этом возникает ряд дополнительных технических проблем, не возникающих в процессе анализа, например, проблема выбора основы словоформы из набора связанных с данной лексемой основ. Выбранный в системе способ представления словаря должен обеспечивать поиск необходимых основ в словаре.

Примером системы синтеза, первоначально разработанной не как часть системы анализа, может служить блок синтеза русских словоформ системы автоматического перевода ЭТАП-1 [2] и ЭТАП-2 [6], разработанной научным коллективом под руководством Ю.Д.Апресяна. Однако общая структура используемой в системе лингвистической модели и лингвистическое обеспечение системы допускает, по мнению ее авторов, использование и при морфологическом анализе [77]. В частности, аналогичный аппарат применяется в системе для морфологического анализа словоформ английского [6] и французского [7] языков.

В системе последовательно проводится принцип полного отделения лингвистической информации от алгоритмов. В частности, вся лингвистическая информация, как морфологическая и словарная, так и синтаксическая, в исходном виде записаны в системе на специальном формальном языке [6, 107, 108]. Последовательное проведение в системе идей модели «Смысл <=> Текст» обеспечивает, тем не менее, четкое отделение морфологического компонента системы от синтаксического и других блоков [92, 93].

Система снабжена дескриптивным описанием морфологического строения русского языка. Чередования букв в основах исключены из рассмотрения путем введения в словарь всех форм основ лексемы, например, *спинк-(а)* и *спинок-(*)*. Хотя в редких случаях не исключается указание отдельных морфов непосредственно в словарных статьях, в большинстве случаев для указания в словаре парадигмы лексемы применяются ссылки на следующие стандартные объекты: стандартные списки, маски, форматы и трафареты.

Стандартный список представляет собой перечень окончаний, соответствующих элементам парадигмы некоторого класса лексем, например: (*спинк-*) *-а, -у, -е, ..., -ах*. Маски применяются для получения из стандартных списков различных вариантов ущербных парадигм. Так, соответствующая маска исключает из стандартного списка для основы *спинк-* лексемы *спинка* окончание *-** родительного падежа множественного числа, а другая маска исключает для основы *спинок-* той же лексемы, наоборот, все окончания, кроме указанного. Таким образом обеспечивается возможность использования одного и того же стандартного списка в различных случаях, требующих построения ущербной парадигмы для лексемы или отдельной основы.

Словарная информация задается отдельно для каждой основы каждой лексемы. Различные части парадигмы основы могут задаваться отдельными «строками» словаря. Например, отдельно может быть задана подпарадигма превосходной степени прилагательного, или только деепричастие. Одной

словарной основе соответствует одна или несколько «строк», в совокупности описывающих часть парадигмы лексемы, образуемую данной основой. Дальнейшими способами сокращения словарной информации являются форматы — списки часто встречающихся «строк» словаря, и трафареты — списки часто встречающихся сочетаний таких «строк».

Достоинством принятого в системе способа задания лингвистической информации является четкость структурирования, компактность и единообразность задания лингвистической информации. С другой стороны, не является достаточно последовательным подход к описанию разбиения словоформы флективного языка на ряд равноправных в техническом отношении фрагментов, например: *чит-а-ющ-ими-ся*. Наличие в словарных статьях приводимых в явной орфографической форме префиксов и суффиксов затрудняет использование имеющейся лингвистической информации для решения задачи морфологического анализа.

Наряду с описаниями морфологии естественного языка, в той или иной мере симметричными относительно операций морфологического анализа и синтеза, иногда применяются специфические, ориентированные только на синтез словоформ описания. Как правила, такие описания оказываются проще, чем предназначенные для морфологического анализа, отличаются более высокой степенью отделения лингвистической информации от алгоритмов и требуют более простого и менее специализированного программного обеспечения. Исторически более ранние морфологические модели были ориентированы именно на синтез словоформ [71, 72, 78].

Одним из способов построения формальных моделей морфологии, ориентированных на синтез словоформ, является использование механизма контекстно-зависимых формальных грамматик [33]. Особенно удобным данный способ является для описания морфологии агглютинативных языков [72]. Для высокофлективных языков, таких как русский, синтезирующая грамматика задается в виде многоуровневого преобразователя.

Примером использования механизма формальных грамматик для описания морфологии русского языка является экспериментальная система морфологического синтеза, созданная научным коллективом под руководством проф. И.А.Большакова [34, 35]. Для синтеза словоформ используется порождающая грамматика с начальным символом {словоформа} и упорядоченным набором продукционных правил вида

Q Г- А --> Г,

представляющим собой собственно лингвистическую информацию. Здесь А — преобразуемый символ, Q — условие применимости правила — логическое выражение, включающее грамматические характеристики синтезируемой лексемы, словарную информацию типа класса склонения и условия на левый и/или правый контекст. Г представляет собой как правило цепочку, на которую следует заменить А. Однако в общем случае Г представляет собой мини-программу редактирования текста, включая контекст. Например, Г может содержать инструкции перестановки А с левым или правым соседним символом, уничтожения соседних символов и т.д.

В системе, наряду с полным отделением лингвистической информации от алгоритмов, последовательно проводится принцип равноправия составляющих словоформу частей — морфов. Например, технически равноправны морфы, составляющие словоформу *чит-а-ющ-ий-ся*. Для описания русского языка вводятся дополнительно вводятся пять промежуточных уровней представления текста, то есть общее число рассматриваемых уровней достигает семи. Использование хорошо изученного в теории и практике программирования механизма формальных грамматик позволило авторам системы разработать эффективный интерпретатор продукционных правил. Программное обеспечение системы является языково-независимым в широком классе языков. Так, с его помощью были разработаны реализации лингвистического обеспечения для английского, сербохорватского, арабского (только глаголы) и латинского языков [48].

К недостаткам использованного метода, однако, относятся сложность системы продукционных правил и недостаточное быстродействие при интерпретации правил. Основным же недостатком подхода, основанного на использовании формальных грамматик указанного вида, является несимметричность лингвистического обеспечения относительно процессов синтеза и анализа текста. В данном случае основным источником несимметричности является зависимость порядка работы алгоритма от порядка записи продукционных правил, а также необратимость некоторых операций редактирования текста, фигурирующих в правой части правил. Так, поскольку правила упорядочены в расчете на использование их при синтезе словоформ, построение непосредственно на их основе

морфологического анализатора путем простой перестановки левой и правой частей каждого правила оказывается невозможным.

Указанных недостатков во многом лишена рассмотренная подробнее в пункте 1.2.1 KIMMO-модель, резко ограничивающая допустимый вид продукционных правил и число уровней представления словоформ. В частности, продукционные правила в KIMMO-модели никак не упорядочены и симметричны относительно операций анализа и синтеза. Другими словами, правила в этой модели описывают не *преобразования* синтезируемого или анализируемого текста, а *соответствия* между цепочками символов, задающих различные, глубинное и поверхностное, представления словоформы. Описание морфологии в виде набора соответствий, а не в виде правил преобразований, является, по-видимому, наиболее удобной формой представления лингвистической информации в системе, обеспечивающей как анализ, так и синтез словоформ.

1.2.4. Методы автоматического обнаружения ошибок

Обнаружение грамматических ошибок является в настоящее время наиболее распространенным приложением алгоритмов морфологического анализа и развитых в рамках задачи морфологического анализа методов организации словаря [27, 104]. Это объясняется следующими тремя обстоятельствами.

Во-первых, как показывает практика, методы представления словаря как набора допустимых словоформ, не опирающиеся на понятия лингвистики и, в частности, на представления о лексемах, их парадигмах и классах словоизменения, дают менее удовлетворительные результаты как при сжатии словаря, так и при организации поиска отдельной словоформы в нем.

Во-вторых, надежное обнаружение грамматических ошибок требует привлечения информации о контексте каждой словоформы, в частности, привлечения элементов синтаксического анализа [141]. Поэтому при расширении системы обнаружения ошибок необходимо не только установление возможности или невозможности существования в языке данной словоформы, но и установление ее грамматических и синтаксических характеристик, что не может быть достигнуто без морфологического анализа.

В-третьих, при практической работе пользователя с комплексной системой обнаружения ошибок возникает необходимость использования системой лингвистических знаний, более обширных, чем просто информация о допустимых словоформах. Так, при внесении нового слова в словарь пользователь должен иметь возможность оперировать понятиями лексемы и ее парадигмы, а не только отдельной словоформы. Системы, не допускающие такой возможности, с одной стороны, требуют от пользователя утомительного, да и приводящего к ошибкам, ввода десятков или сотен форм одного и того же слова, и, с другой стороны, имеют, как правило, чрезмерно увеличенный, плохо сжатый пользовательский словарь новых словоформ.

Таким образом, основным методом автоматического обнаружения грамматических ошибок следует считать метод, основанный на морфологическом анализе проверяемого текста [16, 17]. Практически каждая система морфологического анализа может быть использована для решения задачи обнаружения ошибок, и перечисленные в пункте 1.2.1 системы действительно используются для решения этой задачи [31, 88].

При использовании алгоритма морфологического анализа исключительно для решения задачи обнаружения ошибок могут быть предложены некоторые, весьма незначительные, его упрощения. Так, полная система морфологического анализа должна вырабатывать по словоформе уникальный идентификатор лексемы, хранящийся, как правило, в словаре [6]. В случае системы обнаружения ошибок без элементов синтаксического анализа необходимость определения идентификатора лексемы в некоторых случаях отпадает. Достигнутое таким образом уменьшение объема словаря позволяет в отдельных системах расположить словарь непосредственно в оперативной памяти ЭВМ. Достоинства и недостатки такого подхода рассмотрены в пункте 1.2.2.

В некоторых системах обнаружения опечаток вместо полного морфологического анализа используется упрощенный морфологический анализ [37, 39, 47]. Как правило, такое упрощение отрицательно сказывается на качестве работы системы. Современные достижения лингвистики и развитие методов морфологического анализа позволяют в системах аналогичного класса использовать полный морфологический анализ.

Кроме методов морфологического анализа для решения задачи обнаружения опечаток используются неморфологические методы, которые можно разделить на словарные и бессловарные. К словарным методам относятся методы, опирающиеся на использование неструктурированного списка

всех допустимых словоформ. Обычно такой список алфавитно упорядочивается для облегчения поиска словоформ. Применяются также иные методы сжатия такого списка и методы поиска в нем, например, хеширование. Используемые при этом алгоритмы являются намного более простыми, чем алгоритмы морфологического анализа. Однако подобный метод, с успехом применяющийся при проверке текстов на языках с бедной морфологией, например, английском [165], по указанным выше причинам оказывается несостоятельным при работе с высокофлективными языками, в том числе с русским языком.

Одними из основных несловарных методов обнаружения грамматических ошибок являются методы, подвергающие проверке не целые словоформы, а их части. В частности, метод *n*-грамм заключается в проверке каждого сочетания *n* подряд стоящих символов проверяемого текста. Получили распространение варианты метода с $n = 2$ — метод диграмм [42], и с $n = 3$ — метод триграмм [171]. По заведомо правильному достаточно большому тексту, например, по словарю, составляется таблица *n*-грамм. Таблица содержит по одному биту на каждое теоретически возможное сочетание *n* букв, причем соответствующий бит устанавливается в 1, если такое сочетание букв встречается хотя бы в одном слове языка, и в 0, если оно не встречается в текстах на данном языке. При проверке помечаются как неправильные словоформы, содержащие хотя бы одну недопустимую *n*-грамму.

Применяются варианты метода *n*-граммного контроля, учитывающие частоту встречаемости *n*-грамм в тексте. В целях уменьшения объема таблицы *n*-грамм различаются три или четыре уровня частоты встречаемости *n*-грамм. При проверке как неправильные помечаются словоформы, содержащие хотя бы одну недопустимую *n*-грамму или более одной низкочастотной *n*-граммы.

Достоинствами метода *n*-грамм является крайне экономное расходование оперативной памяти ЭВМ и полный отказ от размещения какой-либо информации на дисковом устройстве, высокое быстродействие, простота алгоритма, слабая зависимость «лингвистического обеспечения» от представительности текста, использованного для первоначального накопления статистики *n*-грамм, а также от степени флективности языка. Серьезным недостатком метода является низкое качество проверки текста. В связи с этим в настоящее время бессловарные методы в качестве самостоятельного средства контроля орфографии выходят из употребления. Однако, как показано в пункте 4.8, подобные методы могут с успехом применяться для некоторых специальных целей в сочетании со словарными методами, в частности, для решения задачи извлечения редких правильных слов из массива случайных буквенных цепочек. Последняя задача возникает в связи с задачей исправления опечаток, см. пункт 1.2.5.

Имеется два типа ошибок, совершаемых системой в процессе орфографического контроля текста: во-первых, признание правильного слова неправильным, и, во-вторых, признание неправильного слова правильным. Ошибки первого типа, то есть ложное сигнализирование опечатки, характерны для словарных методов. Они связаны с неполнотой словаря и как правило легко устраняются путем внесения неопознанных слов в словарь. Ошибки второго типа, при которых система пропускает реальные опечатки, характерны для несловарных методов типа *n*-граммного контроля. Такие ошибки являются в гораздо большей степени неприемлемыми для практически используемой системы орфографического контроля. Кроме того, они являются практически неустранимыми. Так, в слове **очепатка* нет ни одной недопустимой или малочастотной триграммы.

Методом, занимающим промежуточное положение между словарными и бессловарными методами и совмещающим их достоинства и недостатки, является метод хеш-кодов, представляющий собой особого рода способ сжатия словаря с частичной потерей информации. Например, в одном из вариантов реализации системной служебной процедуры spell операционной системы UNIX используется восемь битовых массивов. С каждым из массивов связана определенная хеш-функция. При подготовке массивов системой просматривается эталонный текст или словарь, и в каждом массиве соответствующий бит устанавливается в 1, только если в эталонном тексте встретилось слово, дающее соответствующее значение хеш-функции. При проверке текста по слову вычисляются все восемь хеш-функций, и рассматривается логическое произведение соответствующих битов массивов. Правильными считаются слова, для которых оно равно 1.

Данный метод обладает тем дополнительным по сравнению с методом триграмм достоинством, что его точность прямо зависит от объема памяти, доступного для хранения таблиц. Недостатком является как большой объем используемой оперативной памяти ЭВМ, так и зависимость точности обнаружения ошибок при фиксированном объеме используемой оперативной памяти от общего количества словоформ в словаре, что делает данный метод малоприемлемым для контроля текстов на высокофлективных языках, в частности, — на русском языке.

1.2.5. Методы автоматизированного исправления ошибок

Исследованию методов автоматического исправления ошибок посвящено большое количество работ [45, 96, 100, 154, 135, 162], созданы и получили распространение специализированные системы, предназначенные для автоматического обнаружения и исправления ошибок [39, 40, 91]; кроме того, подсистемы исправления ошибок включены в некоторые комплексные морфологические системы [88]. Задача исправления ошибок является в некотором смысле менее детерминированной и хуже определенной, чем задачи морфологического анализа и обнаружения ошибок, поскольку подразумевает решение вопроса о том, что человек «на самом деле» имел в виду под ошибочным словом. В целях совершенствования алгоритмов исправления ошибок изучаются механизмы совершения ошибок в текстах и закономерности появления в словах тех или иных конкретных искажений [80, 95, 109, 156].

Методы автоматической коррекции ошибок могут быть условно разделены на словарные и несловарные, с ранжированием предлагаемых вариантов по вероятности быть правильными и без такого ранжирования, учитывающие узкий либо широкий контекст ошибочного слова либо рассматривающие слово вне контекста. Таким образом, задача исправления ошибок в широком смысле может быть поставлена в следующей формулировке: с учетом имеющейся информации, для ошибочного слова найти варианты его исправления и выдать наиболее вероятные из них. Технически алгоритм решения такой задачи может быть построен двумя способами.

Первый способ заключается в априорном сужении пространства поиска вариантов исправления, а также в определенном упорядочении процесса перебора гипотез с прекращением поиска после получения «приемлемого» варианта. При этом используются различные вероятностные соображения либо информация о внешнем контексте слова. Необходимость применения данного способа, как правило, обусловлена низким быстродействием применяемого в системе метода проверки отдельной гипотезы. Действительно, пусть проверка одной гипотезы по морфологическому словарю занимает 0,1 сек. Тогда ненаправленный полный перебор вариантов исправления одной ошибки типа пропуска или замены буквы в слове из 10 букв занял бы 1 мин., а двух таких ошибок (**интиллегент*) в одном слове — 10 часов.

Второй способ заключается в получении всех возможных вариантов исправления, и, если найдено более одного варианта, применении затем некоторого алгоритма их ранжирования. Данный способ может быть назван способом апостериорного ранжирования вариантов. Рассмотрение конкретных алгоритмов апостериорного ранжирования вариантов выходит за рамки настоящей работы. Достаточно заметить, что собственно апостериорное ранжирование вариантов является, как правило, гораздо более простой задачей, чем использование вероятностных соображений в процессе поиска вариантов. Поэтому при рассмотрении методов исправления ошибок с апостериорным ранжированием мы ограничимся рассмотрением методов получения вариантов исправления без их ранжирования, а также без учета контекста.

Методы первого типа обычно являются несловарными в том смысле, что словарная информация не используется в процессе порождения гипотез. При этом морфологический словарь все же используется для проверки каждой порожденной гипотезы. Методы второго типа, как правило, используют словарную информацию непосредственно в процессе порождения гипотез.

Различные методы упорядочения перебора гипотез основываются на полученной экспериментально либо теоретически информации о наиболее вероятных типах ошибок. Так, в [143] для набираемых на клавиатуре ЭВМ английских текстов приводится таблица вероятностей замены одной буквы на другую и вероятности пропуска одной буквы в контексте других; данная информация определена экспериментально по представительному корпусу опечаток, встретившихся в очень большом массиве текстов. При исправлении опечатки в первую очередь просматриваются гипотезы, содержащие наиболее вероятные искажения. К недостаткам данного метода по сравнению с другими методами того же класса относится, в частности, отсутствие учета значительного разброса статистических показателей для различных документов и в особенности для различных операторов, осуществлявших ввод этих документов.

Более перспективным представляется подход, основанный на содержательной модели процесса совершения ошибок. Так, в [42, 44] развит подход, основанный на клавиатурной модели процесса совершения ошибок: совершение ошибки рассматривается как ошибка движения пальцев оператора при наборе слова на типовой клавиатуре ЭВМ. Соответственно в первую очередь предлагается, например, рассматривать замены букв в словах на буквы, близкие к данной на клавиатуре ЭВМ. Недостатком данного метода является, в частности, зависимость применяемых оценок от, вообще говоря,

неизвестного, типа клавиатуры, применявшейся при набивке документа, а также отсутствие учета других факторов, влияющих на механизм совершения ошибок.

Кроме того, общим недостатком несловарных методов коррекции ошибок является то, что порядок перебора гипотез никак не соотносится со структурой словаря и, таким образом, является практически случайным по отношению к этой структуре. Как правило, доступ в произвольном порядке к информации, хранящейся во внешней памяти, оказывается во много раз медленнее, чем доступ к такой информации в порядке, согласованном с ее структурой.

Некоторым обобщением понятия модели процесса совершения ошибок является понятие метрики, или расстояния между буквенными цепочками. Вариантами исправления ошибочного слова могут считаться слова, близкие к данной буквенной цепочке в смысле выбранной функции расстояния. Выбор функции расстояния между словами может быть как чисто эмпирическим, так и опирающимся на некоторую модель процесса совершения ошибки.

Так, во многих системах рассматривается фонетический критерий близости слов, опирающийся на представление о процессе совершения ошибки как на ошибочный выбор записи данного фонетического слова [127]. Данный метод получил название Soundex-метода. В смысле данного критерия близости слова physics и *fizyx являются близкими, что делает возможным автоматическое исправление неправильного слова. Однако фонетические ошибки в текстах даже на английском языке не составляют большую часть всех ошибок, а для русского языка данный метод вообще едва ли актуален.

Критерий близости специального вида применяется при автоматическом и автоматизированном исправлении искажений при оптическом вводе текстов с помощью программ оптического распознавания символов (OCR). В этом случае характерными являются искажения типа *u --> ll, e <--> o, r --> l', a <--> s*, и некоторые другие специфические искажения.

Предложено также большое число способов измерения расстояния между словами, найденных эмпирическим путем [49, 161]. Некоторые из них основываются на вероятностных методах коррекции помех в информационном канале [138, 139, 143], другие основываются на подсчете с весами числа элементарных операций, переводящих одно слово в другое [136, 137], третьи — на подсчете для двух слов числа их общих подцепочек, например триграмм [112]. Получили также распространение методы, предполагающие применение некоторого простого критерия близости не к самим словам, а к построенным по ним цепочкам — их кодам. Например, используются фонетические коды, так называемый альфа-код и другие способы стандартизации буквенных цепочек [3, 157, 158].

С методами сравнения кодов слов связаны методы исправления ошибок, при которых для заданной ошибочной цепочки тем или иным способом априорно ограничивается область словаря, в которой ищутся возможные варианты исправления. Небольшой объем полученного подсловаря делает возможным либо полный его просмотр, либо проверку по нему большого числа гипотез. Так, словарь может быть организован в виде набора подсловарей или ассоциативных цепочек, соответствующих определенному значению кода слов [3, 63, 64]. Применяются также другие специальные методы организации физической структуры словаря, ориентированные на поиск цепочек, близких к заданной в смысле определенного критерия близости. Примером такой организации словаря является графовое или иерархическое его представление [128]. Недостатком метода является затрудненность использования данного словаря для решения задач, не связанных с задачей исправления ошибок, например, задачи морфологического анализа.

Другим способом ограничения области поиска вариантов исправления ошибки является метод синтаксических и семантических предсказаний, примененный, например, в системах TULIPS [88] и DYPAR [75]. При анализе искаженного слова выделяется множество слов, допустимых для данной предметной области в данном синтаксическом, семантическом, прагматическом и т.д. контексте на месте ошибочного слова. Метод обладает рядом несомненных достоинств, однако может быть применен только в составе мощной системы искусственного интеллекта с развитым семантическим компонентом.

Все рассмотренные методы решения задачи исправления ошибок представляют определенный интерес, поскольку при практическом решении данной задачи наилучший результат может быть достигнут сочетанием различных методов, а также различных критериев близости слов и различных моделей процесса внесения ошибок в тексты.

1.3. Постановка задачи исследования

Итак, в задачах морфологического анализа, синтеза, нормализации слов, автоматического обнаружения и исправления орфографических ошибок морфологическим преобразованиям подвергается каждое отдельное слово текста. Поэтому ключевым моментом реализации морфологической системы является ее быстродействие. Кроме того, морфологическая обработка текста является не самостоятельной задачей, а подзадачей конкретной прикладной задачи, например, в системе информационного поиска. Ввиду этого другим важнейшим показателем эффективности морфологической подсистемы является требование минимизации объема занимаемой оперативной памяти ЭВМ при использовании крупного словаря.

Развитие аппаратных средств вычислительной техники привело в последние годы к значительному повышению быстродействия распространенных ЭВМ и увеличению объема доступной оперативной памяти. Но проблемы быстродействия и минимизации расхода оперативной памяти остались актуальными, поскольку одновременно многократно возросли объемы обрабатываемых с помощью ЭВМ текстов, а также стали широко применяться многозадачные операционные системы типа Microsoft Windows и OS/2, предъявляющие жесткие требования экономии оперативной памяти каждой из функционирующих в системе задач.

Применяемые в настоящее время морфологические системы не обладают в достаточной степени необходимым сочетанием высокого быстродействия и малого объема занимаемой оперативной памяти и имеют ряд других недостатков. Актуальной остается задача разработки универсальной инструментальной морфологической системы, встраиваемой в произвольную прикладную программу и осуществляющей широкий набор морфологических операций над текстом на основе единой лингвистической модели, единого набора лингвистических данных и комплекса взаимосвязанных алгоритмов.

Основной задачей настоящего исследования является разработка лингвистической модели, методов организации хранения и поиска информации в морфологическом словаре, методов и алгоритмов осуществления морфологических операций над текстом, с целью построения универсальной, языково-независимой в некотором классе языков, включая русский и другие европейские языки, программной морфологической системы, способной осуществлять ряд морфологических операций над текстом на естественном языке, в частности, задачи точного и приближенного морфологического анализа, синтеза, нормализации слов, автоматического обнаружения, объяснения и исправления орфографических ошибок, и др.

Кроме основных технических требований — высокого быстродействия и малого объема занимаемой оперативной памяти, к системе предъявляется ряд специальных требований, таких как полнота анализа и синтеза, возможность обработки слов, содержащих пробелы и специальные символы, например, *OS/2 2.1*, возможность выделения слов из слитных композитных образований типа *тридцатидвухразрядный*, обработка переноса слов при анализе, возможность пословной обработки текста, одновременной работы с несколькими словарями, задающими, например, различные естественные языки, и др. Система должна носить инструментальный характер, что подразумевает универсальность используемого набора лингвистической информации и возможность работы совместно и под управлением прикладной программы, предоставляющей системе входные данные и использующей результаты ее работы.

Ключевым моментом достижения высокого быстродействия при малом объеме занимаемой оперативной памяти является является организация хранения и поиска информации в морфологическом словаре. Структура же морфологического словаря во многом определяет архитектуру морфологической системы в целом. Поэтому для последующих глав нами выбран следующий порядок изложения. В главе 2 излагается предлагаемая структура словаря, определившая выбор используемой в системе лингвистической модели. Далее в главе 3 излагается структура этой модели. В отдельную главу 4 вынесен частный, но сложный и существенный вопрос реализации в такой модели алгоритма исправления опечаток. При разбиении материала по главам также учитывалась возможность независимого использования идей, изложенных в каждой главе. В этом смысле материал главы 2 представляется наименее сложным и не зависящим от остальных глав.

ВЫВОДЫ

1. Существует практическая потребность в разработке универсальной инструментальной морфологической системы, удовлетворяющей определенному набору требований, в частности

требованию высокого быстродействия и малого объема занимаемой оперативной памяти при использовании крупного морфологического словаря, а также языковой независимости в некотором классе языков, включающем русский язык.

2. Имеются объективные предпосылки решения поставленной задачи, заключающиеся в преимущественно суффиксальном характере словоизменения в языках выбранного класса, а также в особенностях флективного и агглютинативного типов словоизменения в данных языках.

3. Рядом исследователей разработаны методы и системы, осуществляющие отдельные задачи морфологической обработки текста. Однако применяемые в настоящее время для анализа текстов на русском языке системы не удовлетворяют всей совокупности предъявляемых требований, в частности, требованию максимального быстродействия при условии экономии занимаемой оперативной памяти ЭВМ.

4. В настоящей работе поставлена задача разработки моделей, методов и алгоритмов, позволяющих построить универсальную, языково-независимую в некотором классе языков, быстродействующую, экономичную по расходу оперативной памяти ЭВМ морфологическую систему, удовлетворяющую также ряду дополнительных требований.

ГЛАВА 2.
МИНИМИЗАЦИЯ ЧИСЛА ОБРАЩЕНИЙ К ДИСКОВОЙ ПАМЯТИ ПРИ СЛОВАРНОМ МОРФОЛОГИЧЕСКОМ АНАЛИЗЕ

2.1. Постановка задачи минимизации числа обращений к диску

2.1.1. Задача морфологического анализа. Два возможных подхода

Рассмотрим задачу морфологического анализа словоформ естественного флективного языка. Будем рассматривать только случай суффиксального словоизменения; префиксы либо рассматриваются как часть основы, либо считаются отделенными от буквенной цепочки до обращения к процедуре поиска в словаре. Не рассматриваются любые явления изменения буквенного состава основы слова, такие как чередования гласных и т.п.; в подобных случаях будем считать, что лексема имеет несколько различных основ, хранимых в словаре по отдельности. Например: *замок-*(*), *замк-(а*) — две разных основы, каждая из которых по отдельности имеется в словаре. Здесь * означает пустую цепочку.

В данной постановке задача морфологического анализа упрощенно сводится к следующей вычислительной задаче. Рассматривается входная буквенная цепочка (словоформа), являющаяся формой некоторого словарного слова (лексемы). Необходимо расчленить буквенную цепочку на начальную часть — основу, имеющуюся в словаре, и комплекс суффиксов, причем так, чтобы удовлетворялись некоторые дополнительные условия, определяемые грамматической информацией при основе. Например: *чит-ающимися*, где *чит-* — основа, *-а-ющ-ими-ся* — тема + суффикс + окончание + частица, возможные для данной глагольной основы.

Существует два возможных подхода к решению такой задачи: анализ «справа налево» и «слева направо». При первом подходе делается попытка вычленить конечную часть словоформы, похожую на комплекс суффиксов, и затем проверить наличие в словаре оставшейся начальной части — гипотетической основы. При втором подходе делается попытка найти в словаре некоторую начальную часть цепочки, а затем проверить, что оставшаяся правая часть образует возможный для данной основы комплекс суффиксов. При обоих подходах могут быть неудачные попытки, когда приходится пробовать другое разбиение.

Как правило, выборка информации из словаря является наиболее долгой операцией при морфологическом анализе. Более точно, долгой операцией является происходящее при этом обращение к дисковой памяти. Поэтому и в случае первого, и в случае второго подхода актуальной является задача сокращения числа обращений к дисковой памяти, происходящих за время полного цикла анализа словоформы.

Поскольку средняя длина слова в научно-техническом тексте составляет 8 — 10 букв, подход «слева направо» может потребовать до 10 обращений к словарю. Как правило, подход «справа налево» требует гораздо меньшего числа обращений к словарю, что обычно является аргументом в пользу выбора именно этого подхода при реализации морфологических систем. Однако по сравнению с подходом «справа налево» подход «слева направо» имеет ряд преимуществ как по простоте реализации, так и по заложенным в нем возможностям.

Так, в первом случае приходится анализировать комплекс суффиксов без всяких априорных предположений о грамматических характеристиках лексемы или хотя бы о ее части речи. Во втором же случае, когда основа и, следовательно, ее грамматическая информация уже известны, проверка «пригодности» данного комплекса суффиксов значительно облегчается.

Далее, первый способ требует априорного определения конца слова во входном буквенном потоке. Мы будем предполагать, независимо от лингвистической содержательности такого подхода, что технически возможным является внесение в словарь слов, содержащих в качестве символов пробелы и другие разделители, напр. *OS/2 2.1, во что бы то ни стало*. Внесение подобных образований в словарь бывает удобным с технической стороны. Кроме того, мы предполагаем, что, слово может не быть ограниченным пробелом или разделителем. Независимость алгоритма от наличия символа-разделителя в конце слова не только абсолютно необходима при анализе словоформ таких языков, как японский или немецкий [103], но и полезна при анализе сложных русских слов и терминов, напр. *водопыленепроницаемый* [65]. При допущенных предположениях задача членения текста на отдельные слова не является тривиальной и должна решаться апостериорно с учетом словарной информации.

Таким образом, задача максимального сокращения числа обращений к диску является тем более актуальной, что ее решение позволит эффективно реализовать обладающий рядом преимуществ

подход «слева направо» к задаче морфологического анализа. Пример такой реализации составляет предмет третьей главы настоящей работы.

2.1.2. Постановка задачи построения морфологической СУБД

Под *начальной подцепочкой V* буквенной цепочки *W* понимается любая ее начальная часть, возможно, совпадающая с *W*. Будем также говорить, что *V вложена слева* в *W*. Таким образом, отношение вложения слева есть отношение нестрогого частичного порядка на множестве буквенных цепочек.

Как показано в пункте 2.1.1, в рассматриваемом случае задача считывания из словаря информации об основе сводится к задаче поиска в нем некоторой начальной подцепочки анализируемой буквенной цепочки. В процессе морфологического анализа часто бывает необходимо произвести поиск в словаре нескольких гипотетических основ данной цепочки. Рассмотрим задачу выборки из словаря всех основ, являющихся начальными подцепочками анализируемой цепочки. Поскольку словарь содержит конечное число основ, для решения данной задачи не требуется ограничивать цепочку справа, то есть априорно указывать конец слова во входном тексте.

Ниже показывается, что количество обращений к диску, необходимых для поиска всех гипотетических основ данной словоформы в словаре, может быть сведено к тому минимуму, который необходим для поиска одной конкретной цепочки, а именно к единице.

Поскольку для решения задачи поиска в словаре в указанной постановке не требуется никакой лингвистической информации, можно рассматривать задачу построения словаря и поиска в нем как задачу управления базой данных. Ключи записей такой базы — основы словоформ, а тела записей — грамматическая и иная информация, связанная с данной основой.

Как правило, задача выборки информации из базы данных по ключу K есть задача нахождения в базе всех записей, ключи которых *равны* K. Здесь K — цепочка букв определенной длины, то есть одна гипотетическая основа словоформы.

В рассматриваемом же случае задача выборки информации по ключу K ставится как задача нахождения в базе всех записей, ключи которых являются *начальными подцепочками* K. Здесь K — цепочка неопределенной длины, а именно весь входной текст, начиная с анализируемой словоформы, но без априорного указания ее конца. Существующие СУБД не ориентированы на решение такой задачи.

Результатом поиска является множество всех гипотетических основ словоформы, даже содержащих пробелы и другие разделители слов. Например, для входного текста «*во что бы то ни стало решить задачу*» будут найдены гипотетические основы *в*-(), *во*-(), *во что бы то ни стало*-(). Здесь пустые скобки означают отсутствие окончания.

Относительно устройства хранения данных предполагается, что информация хранится блоками постоянной длины (напр. 1 килобайт), и элементарным обращением к устройству является считывание одного (любого) такого блока. Далее, предполагается, что такое элементарное обращение является относительно медленной операцией, определяющей быстродействие всего процесса морфологического анализа. Данные предположения являются типичными для дисковых устройств современных ЭВМ и файловых систем современных ОС. Предполагается также, что соображения экономии оперативной памяти не позволяют считывать в память одновременно весь словарь или значительную его часть.

Предметом настоящей главы является описание разработанной автором структуры базы данных, при которой обработка одного запроса (анализ одной словоформы) требует не более одного обращения к дисковой памяти.

Задача не является тривиальной. Так, для анализа слова *дела* необходимо найти в словаре основы *дел*-(*о*) и *д*-(*ела*), разнесенные в словаре [69] примерно на 800 позиций по алфавиту, и, следовательно, при обычном размещении словаря по алфавиту расположенные в далеких его блоках. Поскольку гипотетические основы, пусть даже реально не существующие, также требуют поиска в словаре, такая ситуация встречается при анализе многих слов. Например, при анализе слов *пять, гать, мять*; *тепло, кресло, стало* приходится пробовать по нескольку гипотетических основ. Поскольку каждое слово следует испытывать еще и как неизменяемое, можно сказать, что ситуация, когда гипотетические основы разнесены далеко по алфавитному порядку, встречается регулярно.

2.2. Структура базы данных для морфологического анализа

Рассматривается только задача морфологического анализа. Необходимые для морфологического синтеза дополнения к рассматриваемой структуре даны в пункте 2.6.

Рассмотрим словарь, упорядоченный лексикографически. Уточним, что лексикографический порядок определяется так, что начальные подцепочки любой цепочки не больше самой этой цепочки, например: *пар* < *паровоз* < *паровозный*.

Лексикографический порядок, в частности, позволяет применить известный способ сжатия текста, предложенный Купером [123], сводящийся к тому, чтобы вместо повторения в ключе очередной записи тех его первых букв, которые совпадают с первыми буквами ключа предыдущей записи, указать только их количество.

Зададимся некоторой величиной словарного блока. Добавляя в файл, где это требуется, нули, добьемся того, чтобы ни одна словарная статья не пересекала границу блоков. Это всегда возможно, если только в файле нет слишком длинных записей, превышающих размер блока. Относительное увеличение объема словаря при этой операции не превышает среднего отношения размера записи к размеру блока, что в нашей реализации составило около 3%. Сжатие по Куперу применяется только внутри каждого блока, не затрагивая его начальную запись.

В памяти размещается индексный массив, содержащий для каждого блока ключ его первой записи. Заметим, что в индексном массиве можно хранить каждый ключ не целиком, а только до той позиции (включая), по которой он отличается от ключа последней записи предыдущего блока — этого достаточно для четкого различения блоков при поиске. Поскольку сам индексный массив автоматически получается алфавитно упорядоченным, его также можно разбить на блоки и сжать их по Куперу.

В выполненной автором реализации объем индексных массивов словаря, содержащего около 250 тысяч основ, составил 12.5 Кбайт.

Рассмотрим алгоритм поиска в таком словаре. При поиске ищется алфавитное место предъявленной цепочки в индексном массиве. Алфавитным местом записи называется позиция в упорядоченном массиве, содержащая лексикографически максимальную из имеющихся в словаре записей, не большую данной; если словарь содержит несколько таких записей с одинаковыми ключами, то следует рассматривать последнюю из них по расположению в словаре. С диска считывается соответствующий блок основного массива словаря. В нем ищется алфавитное место той же цепочки. Цепочка обрезается по первой же букве, по которой она не совпадает с найденной таким образом в словаре цепочкой, а область поиска в словаре ограничивается записями, расположенными по словарю выше последней просмотренной. Затем процесс повторяется для такой укороченной цепочки. Такой прием, а также основные идеи приемов, рассмотренных в пункте 3.3.1, сообщены автору Г.Б.Савиной.

Например, рассмотрим поиск для цепочки *парой*. Пусть ее алфавитное место в словаре занято ключом *парол*-(ь). Обрезая цепочку по позиции несовпадающих букв *л/й*, получаем цепочку *паро* и повторяем поиск. Пусть ее алфавитное место занято ключом *пар*-(*а*) (найдена гипотетическая основа). Ограничим область поиска, исключив из нее все записи, начиная с найденной основы. Повторение поиска даст еще одну гипотетическую основу *пар*-(*). Снова ограничим область поиска. Пусть теперь алфавитное место цепочки *пар* занято ключом *охран*-(*а*). Поскольку первые буквы уже не совпадают, поиск окончен. Найдены две гипотетические основы.

Рассмотренный алгоритм и/или структура словаря нуждаются в модификации, поскольку в процессе поиска возможны обращения к разным блокам словаря, как для *парол*-(ь) и *охран*-(*а*).

Предлагается продублировать в каждый блок словаря все те записи из других блоков, при поиске которых на первой итерации возможно обращение к данному блоку. Покажем, что увеличение объема словаря при этом будет незначительным.

Заметим, что записи, которые могут понадобиться на второй и дальнейших итерациях поиска, как видно из приведенного выше примера, имеют ключи, вкладывающиеся слева в ключ, найденный при первой итерации. В примере это ключ *парафин*-, а искомыми были записи с ключами *пар*- и, возможно, *па*- и *п*-. Таким образом, в каждый блок достаточно продублировать те записи, ключи которых вкладываются слева в ключ хотя бы одной записи из данного блока.

На самом деле дублировать придется только записи с ключами, вкладывающимися в ключ первой записи блока. Для доказательства рассмотрим вспомогательное

Утверждение 1. Пусть A и B — буквенные цепочки, $A < B$ лексикографически, и K — начальная подцепочка B. Тогда либо K — начальная подцепочка A, либо $A < K$.

Доказательство. Пусть B состоит из букв $b_1b_2...b_n$, K — из букв $k_1k_2...k_l$, $l <= n$, а ключ A первой записи блока — из букв $a_1a_2...a_m$. Поскольку K вкладывается слева в B, то K состоит из тех же букв: $k_1=b_1$, ..., $k_l=b_l$. Пусть K не вкладывается слева в A. Если A вкладывается слева в K, то K лексикографически больше A. Иначе некоторая буква k_i ключа K не совпадает с буквой a_i ключа A. Будем считать, что i — минимальный номер несовпадающих позиций в A и K, и, следовательно, в A и B, поскольку $a_j=k_j=b_j$ при $j < i$. Поскольку A лексикографически меньше B, имеем $a_i < b_i$. Возвращаясь к тому, что $k_i=b_i$ — первая слева буква в K, отличающаяся от соответствующей буквы в A, снова получаем, что K лексикографически больше A. Данный вывод получен в предположении, что K не вкладывается слева в A. Утверждение доказано.

Таким образом, если некоторый ключ вкладывается слева в ключ одной из записей данного блока, то он либо вкладывается в ключ первой записи блока, либо лексикографически больше него. В последнем случае запись, имеющая такой ключ, уже расположена в данном блоке и не требует дублирования. Действительно, ее ключ больше ключа первой записи блока. Но ее ключ вкладывается слева в ключ некоторой записи из данного блока и поэтому лексикографически меньше него. Следовательно, запись расположена между двумя записями одного блока и сама расположена в нем.

Поскольку в качестве ключа B можно рассматривать любой ключ из блока, доказано, что дублированию в блок подлежат только записи с ключами, вкладывающимися слева в ключ первой записи блока. Это, как правило, короткие основы, часто испытываемые в качестве гипотетических основ, однако число их невелико. В составленном при участии автора словаре на основе словаря [69] 512-байтный блок содержал после сжатия по Куперу в среднем около 30 записей, а продублировано в него было в среднем одна — три, редко четыре — пять записей. Объем словаря увеличился менее чем на 10%.

Кроме того, если про некоторые основы дополнительно известно, что их они вообще не могут присоединять суффиксы либо присоединяют только суффиксы, лексикографически меньшие соответствующего остатка первого ключа блока, то такие записи можно в данный блок не дублировать. Так можно избежать дублирования словарных статей предлогов типа *о, по* и т.п. В случае же наличия суффиксов некоторую дополнительную выгоду можно извлечь, изменив алфавитный порядок букв так, чтобы первые буквы наиболее часто (по словарю) используемых окончаний оказались в начале алфавита. Под первой записью блока здесь и далее понимается первая запись *без учета* продублированных.

Заметим, что, при расположении продублированных записей в начале блока рассмотренный выше алгоритм поиска в одном блоке не претерпевает никаких изменений, т.к. блок по-прежнему остается лексикографически упорядоченным. Однако рассмотренная структура словаря гарантирует, что этот алгоритм находит все гипотетические основы анализируемого слова в первом же блоке, считанном с устройства дисковой памяти.

2.3. Алгоритмы формирования, выдачи и изменения словаря

2.3.1. Алгоритм формирования структуры словаря

Рассмотрим алгоритм формирования словаря предложенной структуры, осуществляющий импорт базы данных из файла, записи которого (словарные статьи) упорядочены лексикографически по основам слов. Записи входного файла поступают на вход алгоритма последовательно по одной, выходной массив формируется также последовательно по мере поступления входных записей. Алгоритм является однопроходным и не требует хранения в памяти больших массивов информации.

Предварительно рассмотрим полезную конструкцию, которую назовем стеком вложенных ключей. Это LIFO-собой стек (т.е. организованный по принципу «последним пришел — первым ушел»), элементами которого являются словарные статьи формируемого словаря. Стек вложенных ключей обновляется при поступлении каждой новой записи из входного файла.

В начале работы стек пуст. При поступлении из входного файла каждой очередной записи с вершины стека последовательно удаляются все элементы, ключи которых не вкладываются слева в ключ поступившей со входа записи. Процесс удаления останавливается, когда стек исчерпан либо на вершине стека находится запись, вкладывающаяся слева в прочтенную. Затем новая запись помещается на вершину стека.

В каждый момент стек содержит последовательность записей, считая от наиболее глубокой до находящейся на вершине стека, нестрого упорядоченную по отношению вложения слева, а, следовательно, и по алфавиту. Доказательство легко провести методом индукции по числу прочтенных

записей, и мы его опускаем. Более того, после внесения в стек очередной прочтенной записи в стеке находятся *все* записи словаря, ключи которых вкладываются слева в ключ текущей записи.

Действительно, поскольку входной файл упорядочен лексикографически, все они расположены выше по файлу, чем текущая запись, и ранее вносились в стек. Но они не были удалены из стека. Для доказательства предположим противное: пусть запись с ключом *K*, вкладывающимся слева в ключ текущей записи *B*, была удалена из стека на шаге чтения некоторой записи с ключом *A*. Это означает, что *K* не вкладывается слева в *A*. Из утверждения 1 следует, что *K* лексикографически больше *A*, и соответствующая запись расположена после *A* во входном файле. Обнаруживается противоречие.

Заметим попутно, что ввиду упорядоченности ключей в стеке по отношению вложения слева эти ключи могут быть размещены в одном и том же буфере, и для каждого ключа следует отдельно хранить только его длину. Например, если в буфере расположена цепочка *паровозный*, то последовательность длин 3, 7, 10 задает последовательность ключей *пар*, *паровоз*, *паровозный*. Таким образом, объем памяти, необходимый для хранения ключей в стеке, практически не зависит от их количества. Заметим также, что разность между соседними длинами в последовательности представляет собой число повторенных букв при сжатии такой последовательности по методу Купера.

Описание свойств стека вложенных ключей окончено. Рассмотрим порядок работы приведенного алгоритма на примере. Буфер содержит буквы расположенных в стеке ключей, а собственно стек — их длины и ассоциированную с ними грамматическую информацию (в примере мы ее не показываем). Пусть входной файл содержит записи с ключами *пар*-*, *паровоз*-*, *паровозн*-(*ый*), *пароход*-*, *самолет*-*. Они читаются одна за другой:

Буфер: _ _ _ _ _ _ _ _ _ (пуст)
Стек: (пуст)

На входе: *пар*-*
Удалять из стека нечего.
Добавляется: *пар* — копируем в буфер.
Буфер: *п а р* _ _ _ _ _ _
Стек: вершина: 3 (буквы в буфере, т.е. *пар*)

На входе: *паровоз*-*
Вершина вкладывается во входную запись — не удаляем.
Добавляется: *паровоз* — копируем в буфер.
Буфер: *п а р о в о з* _ _
Стек: 3 (*пар*), вершина: 7 (*паровоз*)

На входе: *паровозн*-(*ый*)
Вершина вкладывается во входную запись — не удаляем.
Добавляется: *паровозн* — копируем в буфер.
Буфер: *п а р о в о з н* _
Стек: 3 (*пар*), 7 (*паровоз*), вершина: 8 (*паровозн*)

На входе: *пароход*-*
Вершина не вкладывается во входную запись — удаляем.
Стек: 3 (*пар*), вершина: 7 (*паровоз*)
Вершина не вкладывается во входную запись — удаляем.
Стек: вершина: 3 (*пар*)
Вершина вкладывается во входную запись — не удаляем.
Добавляется: *пароход* — копируем в буфер.
Буфер: *п а р о х о д* _ _
Стек: 3 (*пар*), вершина: 7 (*пароход*)

На входе: *самолет*-*
Вершина не вкладывается во входную запись — удаляем.
Стек: вершина: 3 (*пар*)
Вершина не вкладывается во входную запись — удаляем.
Стек: (пуст)
Удалять из стека нечего.
Добавляется: *самолет* — копируем в буфер

Буфер: *с а м о л е т _ _*
Стек: вершина: 7 (*самолет*)

Перейдем теперь к алгоритму формирования структуры базы данных. Требуется расположить прочтенные со входа записи в блоках, продублировав в каждый блок все записи, ключи которых вкладываются слева в ключ первой записи блока. Лексикографическое упорядочение будет получено автоматически, поскольку так упорядочен входной файл.

Входной файл читается запись за записью и копируется, с сокращением по методу Купера, в выходной файл. Вместе с этим обновляется стек вложенных ключей. В каждый момент известно, сколько байт уже выведено в выходной массив. Если длина очередной предназначенной к выводу записи такова, что выводимая запись должна пересечь границу блоков, *вместо* ее вывода будут предприниматься следующие действия.

Вывод очередного блока завершается, выходной массив дополняется нулями до границы блока, начинается вывод нового блока. На выход копируется все содержимое стека вложенных ключей, начиная с самой глубокой записи и кончая вершиной. Заметим, что на вершине стека расположена новая, не уместившаяся в предыдущем блоке запись. Остальные элементы стека суть дублируемые в данный блок записи, ключи которых вкладываются слева в ключ первой записи блока. При выводе стека он сжимается по методу Купера. При формировании нового блока в его заголовок можно поместить уменьшенное на единицу количество записей в стеке вложенных ключей. Это число, равное количеству продублированных в блок записей, облегчает распаковку (экспорт) внутреннего представления базы данных в текстовый файл, а также операции пополнения словаря.

Далее процесс чтения новых записей и копирования их в выходной массив продолжается. Следующая после вывода блока запись обычным образом сокращается по методу Купера по сравнению с прочтенной перед ней. При достижении конца входного файла база данных сформирована.

Поясним работу алгоритма на примере. Пусть входной массив состоит из записей с ключами: *автомобиль*-*, *пар*-*, *паровоз*-*, *паровозн*-(*ый*), *пароход*-*, *самолет*-*, *яхт*-(*а*). Пусть, например, первый блок словаря оказался переполненным после поступления на вход слова *пароход*-*. Следовательно, данный блок выводится в выходной файл, а запись с ключом *пароход*-* становится первой записью нового блока. В выходной файл выводятся нули до тех пор, пока текущая позиция в файле не окажется кратной размеру блока. Затем выводится содержимое стека, содержащего, как показано в предыдущем примере, записи с ключами *пар*-*, *пароход*-*. Затем на вход поступают и обрабатываются следующие записи — *самолет*-*, *яхт*-(*а*). Словарь разбивается на блоки следующим образом:

*B*1:		*автомобиль-*, пар-*, паровоз-*, паровозн-(ый)* 000
*B*2:	*пар-**	*пароход-*, самолет-*, яхт-(а)* 000000000000

В блок *B*2 оказалось продублировано слово *пар*-*. Рассмотрим поиск в таком словаре цепочек *паровозы* и *пары*. Алфавитное место цепочки *паровозы* расположено в нем после ключа *паровозн*-, то есть в блоке *B*1; в том же блоке расположена и искомая основа *паровоз*-. Алфавитное же место цепочки *пары* расположено после ключа *пароход*-, то есть в блоке *B*2, в то время как искомая основа *пар*- находится в блоке *B*1. Но поскольку она продублирована в блок *B*2, считывания обоих блоков не требуется. Считывание же блока *B*2 необходимо уже для проверки того, не является ли цепочка *пары* неизменяемым словом или словом с пустым окончанием. Рассмотрение примера окончено.

Рассмотрим формирование индексных массивов. При формировании очередного блока необходимо, в дополнение к перечисленным выше действиям, вывести в отдельный текстовый файл ключ записи с вершины стека, т.е. первой новой записи блока. После того, как вся база данных сформирована, этот текстовый файл обрабатывается с помощью той же самой процедуры. Полученный файл имеет ту же структуру, что и основной массив базы данных, но гораздо меньший объем, и тела записей в нем пусты, то есть сохранены только ключи. Он представляет собой индексный массив первого уровня.

По результатам обработки индексного массива первого уровня аналогично строится индексный массив второго уровня, и т.д. до получения массива, содержащего только один блок. В выполненной автором реализации словарь, содержащий около 250 тысяч записей, потребовал два индексных массива.

При выводе ключа в индексный массив он усекается по позиции (сама позиция остается) первого несовпадения с ключом последней занесенной в предыдущий блок записи. Как уже отмечалось,

этого достаточно для четкого различения блоков. Заметим, что индексные массивы не содержат никакой информации, кроме собственно усеченных ключей.

Заметим, что в стек вложенных ключей можно не помещать записи, ключи которых не могут присоединять окончаний, например, предлоги. Однако это требует нежелательного привлечения содержательной лингвистической информации в процессе формирования базы данных.

Алгоритм будет правильно обрабатывать случаи наличия в словаре записей с одинаковыми ключами, когда это вообще возможно, то есть когда группа записей с одинаковыми ключами не превышает по размеру один блок.

Для целей облегчения поиска в базе данных полезно поместить в базу некоторую дополнительную информацию о записи, как, например, длину вложенного ключа (см. пункт 2.4). Например, если словарь содержит ключи *пар*, *паровоз*, *паровозный*, то длина вложенной основы для *паровозный* будет 7 (*паровоз*), для *паровоз* — 3 (*пар*), для *пар* — 0 (нет вложенных ключей). Для этой цели следует вместе с каждой записью выводить длину ключа, лежавшего на вершине стека вложенных ключей непосредственно перед моментом вталкивания в стек данного ключа, либо 0, если стек был пуст.

2.3.2. Алгоритм выдачи словаря в текстовой форме

Алгоритм распаковки базы данных предложенной структуры является однопроходным, выходной файл формируется последовательно по мере чтения входного файла. Алгоритм не требует хранения в памяти больших массивов информации. Для распаковки базы данных требуется только основной файл базы, файлы индексных массивов не требуются.

Блоки файла базы данных читаются последовательно. В прочтенном блоке производится раскрытие сжатых по методу Купера ключей записей. Затем в выходной текстовый файл выдаются все записи, кроме первых, продублированных в данный блок. Число продублированных в блок записей указано в заголовке блока. Нули на конце блока игнорируются.

Наличие явного указания в заголовке блока числа продублированных записей не является обязательным. Можно предложить следующие три модификации алгоритма, не использующие эту информацию.

В простейшем случае, когда в базе не разрешено повторение ключей, достаточно игнорировать в каждом новом блоке все первые записи, лексикографически меньшие последней записи предыдущего блока, поскольку это и есть продублированные в данный блок записи.

В случае наличия в базе одинаковых ключей, что обычно для морфологического словаря, можно применить тот же прием, если первая запись блока гарантировано не имеет ключа, равного ключу последней записи предыдущего блока. Это достигается несложной модификацией алгоритма формирования базы, рассмотренного в пункте 2.2. Ввиду тривиальности такой модификации мы опускаем ее детали.

Наконец, если границы блоков могут проходить между записями с одинаковыми ключами, но известно, что в базе не может быть записей с одинаковыми и ключами, и информационными телами, то следует в каждом блоке отбрасывать все первые записи до тех пор, пока не выполнится одно из условий: либо встречена запись с ключом, большим ключа последней записи предыдущего блока, либо встречена запись, совпадающая с последней записью предыдущего блока, считая и ключ, и тело. В последнем случае такая запись также игнорируется, но следующая уже выдается в выходной файл.

Процесс распаковки базы данных формирует текстовый упорядоченный файл, являющийся точной копией входного файла алгоритма формирования базы данных.

2.3.3. Алгоритм изменения, удаления и добавления записей словаря

Предложенная структура базы данных не обеспечивает удобного изменения сформированной базы, поскольку операции изменения словаря являются не столь критичными по быстродействию, как операция поиска в нем. Как показано в пункте 3.3.4, для целей морфологического анализа изменение крупного словаря не является необходимым.

При изменении небольшого по объему словаря, либо при добавлении большого количества слов, либо в случае, когда время работы алгоритма актуализации словаря несущественно, производится распаковка имеющейся базы в текстовый вид, внесение в текстовый файл всех изменений, например, слияние с отсортированным массивом новых записей, и затем формирование новой базы данных. Соответствующие алгоритмы, как показано в пунктах 2.3.1 и 2.3.2, имеют достаточное быстродействие.

Ввиду их однопроходности при этом используются предоставляемые современными операционными системами, такими как UNIX или OS/2, средства конвейерной обработки данных в памяти ЭВМ. В случае, если ОС, как, например, MS-DOS, не имеет таких средств, конвейерная обработка организуется программой обновления базы.

Однако если нужно быстро внести в словарь на диске одно новое слово, применяются специальные приемы. Один из возможных методов состоит в следующем. При формировании базы заранее оставляются пустыми некоторые блоки, например, каждый двадцатый. При внесении нового слова весь отрезок словаря, начиная с блока, содержащего алфавитное место новой записи, должен переформатироваться работающими «по конвейеру» алгоритмами распаковки и упаковки. Переформатирование заканчивается на блоке, вид которого не изменился в данном процессе.

То, что участок, подвергаемый переформатированию, будет иметь небольшую протяженность, гарантируется двумя соображениями. Во-первых, заранее оставленные пустыми блоки компенсируют рост объема словаря и поглотят волну перемещения его записей. Во-вторых, блоки, в ключ первой записи которых не вкладывается слева ключ новой записи, не должны подвергаться изменениям, кроме как вследствие раздвижки словаря. Когда пустых блоков станет слишком мало, необходимо произвести полное переформатирование словаря.

Удаление и изменение записей производится аналогично. Вместо удаления лишние записи могут быть просто помечены специальной пометой и игнорироваться при поиске. Такая помета ставится на всех копиях записи в разных блоках словаря. При возрастании количества помеченных записей база полностью переформировывается.

2.4. Алгоритм поиска гипотетических основ в словаре

Существуют различные варианты алгоритма поиска в словаре рассматриваемой структуры и различные формы организации его индексного массива. В выполненной автором реализации хранящийся в памяти индексный массив составлен из ключей первых записей блоков, урезанных так, как это описано в пункте 2.2. Сам индексный массив имеет в точности такую же структуру, как и основной массив базы данных, однако информационные тела всех записей пусты. Он также разбит на блоки, и по ним составлен еще один такой же индексный массив, объем которого меньше размера одного блока.

В момент инициализации базы данных дополнительно к индексному массиву первого уровня строится массив M целых чисел, размер которого равен числу блоков индексного массива первого уровня. Его элемент m_i равен суммарному количеству записей (не считая продублированные) во всех блоках первого индексного массива с номерами, меньшими i. Количество записей в блоке хранится в заголовке блока, поэтому процедура построения M оказывается однопроходной и быстрой. Конечно, можно хранить m_i и прямо в заголовке i-го блока.

Рассмотрим процедуру поиска для цепочки W. Сначала просматривается индексный массив второго уровня. Пусть алфавитное место W в его единственном блоке есть n-е слово блока. Тогда производится поиск алфавитного места той же цепочки в n-м блоке первого индексного массива. Пусть это k-е слово блока. Тогда с диска считывается блок основного массива с номером m_n+k. По построению базы данных, этот блок содержит все записи, ключи которых вкладываются слева в W, и более обращений к диску не производится. Поиск во всех трех блоках производятся одной и той же процедурой поиска в блоке.

Поскольку время работы процедуры поиска в блоке существенно меньше времени считывания блока с диска, оптимизация такой процедуры не является актуальной. Могут быть предложены различные быстрые способы поиска, например, с древесной организацией ключей в одном блоке. Однако в выполненной автором реализации достаточным оказался простой просмотр всех записей одного блока с его начала. Позиции записей, ключи которых вкладываются слева в W, заносятся в LIFO-стек. Поиск прекращается, когда блок исчерпан либо найдена запись с ключом, лексикографически большим W.

Тот факт, что ключи записей сокращены по методу Купера, только ускоряет процедуру сравнения, поскольку часто позволяет совсем не производить сравнение ключа очередной записи с W.

При поиске в блоке индексного массива результатом является номер записи, на которой остановился процесс поиска; стек позиций вложенных записей не используется.

При поиске в блоке основного массива результатом является все множество записей, позиции которых занесены процедурой в стек. Они извлекаются из стека, начиная с его вершины. Заметим, что при этом первой строится наиболее длинная гипотетическая основа, для которой вероятность оказаться

правильной максимальна. Так, при анализе слова *паровозный* сначала будет получена гипотетическая основа *паровозн-*, затем *паровоз-*, затем *пар-*.

Можно рассмотреть две модификации процедуры поиска, не использующие дополнительный стек. Во-первых, ссылки на записи с вложенными ключами могут храниться непосредственно в блоке. Во-вторых, в блоке может храниться информация о длине вложенного ключа (см. пункт 2.3.1). В этом случае можно последовательно повторять поиск в блоке основного массива, соответственно обрезая входную цепочку W и ограничивая число записей в блоке так, чтобы исключить из области поиска запись, найденную на предыдущей итерации (см. пункт 2.2).

2.5. Алгоритм решения одной задачи, относящейся к проблеме поиска ближайшей цепочки

При решении задачи автоматического исправления опечаток методами, изложенными в главе 4, возникает необходимость найти для буквенной цепочки некоторую ближайшую цепочку. Рассмотренные в главе 4 алгоритмы оперируют величинами, вычисляемыми для данной отсутствующей в словаре цепочки на основе информации, имеющейся в словаре. Поскольку сама исходная цепочка в словаре отсутствует, задача извлечения необходимой информации в этом случае не сводится к рассмотренной в пункте 2.1.2. Однако и в этом случае может быть поставлен и, как показано ниже, положительно решен вопрос об извлечении из словаря всей необходимой информации при не более чем одном обращении к дисковой памяти.

Постановке задачи необходимо предпослать ряд определений, а также дать анализ некоторых свойств определяемых понятий.

Для данной цепочки W рассмотрим пару слов словаря W_1 и W_2, таких что $W_1 < W_2$, W у W_1 и $W < W_2$. Эти слова называются ближайшими в словаре к слову W, а W расположено непосредственно между W_1 и W_2. В случае, если W меньше первого слова словаря, в качестве W_1 рассматривается пустая цепочка. В случае, если W больше последнего слова словаря, в качестве W_2 рассматривается цепочка, состоящая из специального символа конца алфавита. По определению, этот символ больше по порядку, чем любая буква алфавита.

Через D_1 (D_2) обозначается минимальный номер позиции, по которой слово W отличается от W_1 (W_2), т.е. (увеличенная на единицу) длина их общей начальной подцепочки. Через d_2 обозначается буква, стоящая в W_2 на позиции D_2. Позицией несовпадения D для слова W относительно данного словаря называется наибольшее из чисел D_1 и D_2. Ближайшей большей буквой d называется буква d_2, если $D=D_2$, в противном случае символ конца алфавита. Например, пусть словарь содержит фрагмент: «*массив, машина, ...*»; тогда для W=«*маска*» имеем $D_1=D=4$, $D_2=3$, d_2=«*ш*», d есть символ конца алфавита. Здесь позиции букв в слове нумеруются с единицы. Значения D и d характеризуют ближайшие к W слова словаря. Как показано в пункте 4.3, эффективное определение этих значений необходимо для решения проблемы исправления опечаток в тексте.

Другими словами, позицией несовпадения D для слова W относительно множества слов M называется (увеличенная на единицу) наибольшая длина общей начальной подцепочки W и слова из M. Ближайшей большей буквой d называется минимальная по алфавитному порядку буква, стоящая на позиции D в слове из M, совпадающем с W по начальному отрезку длины D, однако большая буквы, стоящей на позиции D в слове W, если последняя существует. Здесь максимум и минимум рассматриваются по множеству M. В подсчете минимума для d участвует также символ конца алфавита. Доказательство эквивалентности двух определений тривиально.

Отсюда непосредственно вытекают следующие свойства значений D и d. Обозначим через M и M' множества цепочек, а через D и D', d и d' соответствующие значения для цепочки W относительно M и M'. Аналогичные обозначения вводятся для M''.

Утверждение 2. Пусть M' есть подмножество M. Тогда D' не больше D, и если $D'=D$, то d' не меньше d.

Утверждение 3. Пусть M' есть множество, состоящее из начальных подцепочек слов множества M. Тогда D' не больше D, и если $D'=D$, то d' не меньше d.

Утверждение 4. Пусть M есть объединение множеств M' и M''. Тогда D есть максимальное из чисел D' и D''. Если $D'=D''$, то d есть минимальное из значений d' и d'', в противном случае d есть то из этих значений, которое соответствует максимальному из значений D' и D''.

Утверждение 5. Пусть M не содержит начальной подцепочки W. Тогда приписывание ко всем словам M справа любых цепочек не изменяет значений D и d.

Доказательства всех утверждений непосредственно вытекают из экстремальной природы значений D и d. Другие полезные утверждения относительно свойств D и d приводятся в пунктах 4.2.1 и 4.3.

Предполагается, что определение значений D и d относительно списка слов, расположенного в оперативной памяти ЭВМ, не представляет труда. В частности, такие значения определяются процедурой поиска цепочки в блоке, как в блоке основного файла словаря, так и в блоке индексного массива. Рассмотрение вводных определений окончено.

Рассматривается задача нахождения значений D и d относительно словаря с блочной структурой. Проблема состоит в том, чтобы в случае, если W_1 и W_2 расположены в разных блоках основного файла словаря, избежать считывания дополнительного блока с дискового устройства. Таким образом, показывается, что определение значений D и d возможно при одном обращении к дисковой памяти.

Алгоритм 1. Значение D определяется в процессе работы алгоритма поиска, рассмотренного в пункте 2.4. В начале работы алгоритма D=0, d есть признак конца алфавита. В процессе работы алгоритма поиска в словаре после выполнения поиска в каком-либо списке, будь то блок индексного массива или блок основного файла, выполняются следующие действия. Пусть найденные процедурой поиска в данном списке значения относительно данного списка есть D' и d'. Если $D' > D$, то D устанавливается в D', а d устанавливается в d'. Если $D'=D$ и $d' < d$, то d устанавливается в d'. Иначе никаких действий не производится. По завершении процесса поиска, рассмотренного в пункте 2.4, D и d имеют требуемые значения. Доказательство этого факта составляет дальнейшее содержание данного пункта. Другими словами, D есть максимальное значение среди частных значений, полученных относительно всех просмотренных в процессе поиска списков, а d есть минимальное значение среди частных значений, соответствующих данному значению D. *Конец алгоритма 1*.

Условие $(d' < d)$ может быть ослаблено до условия (d' не равно d) или даже до условия (d' не есть символ конца алфавита). Однако в модификации алгоритма 1, рассмотренной в пункте 4.3, используется именно данная форма условия.

В процессе работы алгоритма 1 не выполняется считывания дополнительных блоков с дискового устройства, то есть необходимые величины определяются при одном элементарном обращении к дисковой памяти.

Обозначим через D' значение, определенное относительно считанного в память блока основного файла словаря, а через D'' значение, определенное относительно всего индексного массива первого уровня (а не только одного его блока). Аналогично обозначим остальные величины, например, W'_1 и W''_1.

Для доказательства правильности работы алгоритма 1 достаточно показать, что D есть максимум из D' и D'', а d соответствующим образом определяется значениями d' и d''. Конкретно, если $D' > D''$, то $d=d'$; если $D'' > D'$, то $d=d''$; если $D'=D''$, то либо $d''=d'$ и d равно этому значению, либо одно из d'', d' есть символ конца алфавита, и тогда d равно другому из них.

Действительно, отношение между индексным массивом второго уровня и индексным массивом первого уровня полностью аналогично отношению между индексным массивом первого уровня и основным файлом словаря. Если индексный массив первого уровня содержит более одного блока, то значения D'' и d'' определяются полностью аналогичным образом по значениям, найденным при анализе индексного массива второго уровня, играющим теперь роль D'' и d'', и значениям, найденным при анализе одного блока индексного массива первого уровня, играющим роль D' и d'. То же верно для любого числа уровней индексных массивов. Поскольку операция взятия максимума ассоциативна, значение D есть максимум среди значений, определенных относительно каждого просматриваемого в процессе поиска блока, то есть относительно единственного блока индексного массива второго уровня, соответствующего блока индексного массива первого уровня, и соответствующего блока основного файла. Аналогичное рассуждение верно для d.

Согласно алгоритму считывания блока с диска, рассмотренному в пункте 2.4, в оперативную память ЭВМ считан блок, соответствующий элементу W'_1 индексного массива; в частности, W'_1 есть усеченное первое слово этого блока.

Поскольку блок основного файла является подмножеством всего словаря, то по утверждению 2, D' не больше D, а если $D'=D$, то d' не меньше d. Поскольку индексный массив содержит начальные подцепочки некоторого подмножества слов словаря, то по утверждениям 2 и 3, D'' не больше D, а если $D''=D$, то d'' не меньше d. Поэтому для доказательства того, что D есть максимальное из значений D' и D'', а d — соответствующим образом найденное минимальное значение, достаточно показать, что либо $D'=D$ и $d'=d$, либо $D''=D$ и $d''=d$, либо выполняются оба условия.

Рассмотрим отдельно случаи, когда считанный с диска блок содержит W_1 и W_2, когда он содержит только W_1 и когда он содержит только W_2.

Случай первый: оба соседних с W по алфавиту слова находятся в одном блоке. Другими словами, $W'_1=W_1$ и $W'_2=W_2$. Следовательно, $D=D'$ и $d=d'$.

Случай второй: $W_2=W'_2$ есть первое слово считанного блока, а W_1 есть последнее слово предыдущего блока, отсутствующего в оперативной памяти ЭВМ. Следовательно, W''_1 есть начальная подцепочка W_2. Поскольку $W < W_2$ и неверно, что $W < W''_1$, то, по утверждению 1 из пункта 2.2, W''_1 есть начальная подцепочка W. По построению индексного массива, последняя буква W''_1 отличается от соответствующей буквы W_1. Поскольку W''_1 есть общая начальная подцепочка цепочек W и W_2, очевидно, что D_1 не больше, а D_2 больше длины W''_1. Следовательно, $D_1 < D_2$, и по определению $D=D_2$ и $d=d_2$.

Рассмотрим блок основного файла. Поскольку $W_2=W'_2$, то $D'_2=D_2$ и $d'_2=d_2$. Поскольку W'_2 есть первая запись блока в смысле, определенном в пункте 2.2, перед ним в блоке могут быть расположены продублированные в блок слова, являющиеся его начальными подцепочками. Поскольку $W < W'_2$, W'_1 есть начальная подцепочка W'_2, следовательно D'_1 не больше D'_2. Отсюда $D'=D'_2=D_2=D$ и $d'=d'_2=d_2=d$.

Случай третий: W_1 есть последнее слово считанного блока, а W_2 есть первое слово следующего блока, отсутствующего в оперативной памяти ЭВМ. Следовательно, $W''_1 < W_1$, W''_2 есть начальная подцепочка W_2, $W'_1=W'_1$, W'_2 есть цепочка, состоящая из символа конца алфавита.

Рассмотрим блок основного файла. Очевидно, $D'=D'_1$ и d' есть символ конца алфавита. Кроме того, $D'_1=D_1$. Рассмотрим индексный массив. Пусть W_1 совпадает с W_2 по первым k-1 позициям, но не по k позициям. По построению индексного массива слово W''_2 является начальным отрезком длины k слова W_2, т.к. W_1 есть последнее слово блока. В частности, W''_2 не больше W_2. Поскольку W расположено по алфавиту между W_1 и W_2, оно совпадает с W_1, W''_2 и W_2 по k-1 позициям, но отличается от W''_2, а следовательно и от W_2, по позиции k, поскольку $W < W''_2$. Следовательно $k=D''_2=D_2$, причем $d''_2=d_2$ есть совпадающая k-я буква слов W''_2 и W_2.

Рассмотрим два варианта. Пусть $D_1 > D_2$. Тогда $D=D_1=D'_1=D'$ и $d=d'$, поскольку оба значения по определению есть символ конца алфавита. Пусть теперь D_1 не больше D_2. Поскольку D''_1 не больше D_1, которое не больше $D_2=D''_2$, то $D''=D''_2=D_2=D$ и $d''=d''_2=d_2=d$.

Правильность работы алгоритма 1 доказана. В заключение докажем эквивалентность трех условий на d в алгоритме 1. Поскольку алгоритм поиска слова в словаре сначала просматривает индексный массив, и только затем блок основного файла, достаточно показать, что в рассмотренных выше терминах d'' всегда не меньше, чем d', если только последнее значение не есть символ конца алфавита. Действительно, в последнем варианте третьего из рассмотренных выше случаев расположения W в блоке d' есть символ конца алфавита. Во всех остальных случаях выполняются соотношения $d'=d$ и d'' не меньше d.

2.6. База данных для морфологического анализа и синтеза и алгоритмы работы с ней

Морфологическим синтезом называется построение словоформы по заданному номеру — идентификатору — лексемы и морфологическим характеристикам. База данных с рассмотренной выше структурой допускает поиск только по буквенной цепочке — ключу записи. База же, поддерживающая как морфологический анализ, так и синтез, должна допускать поиск записи по идентификатору основы.

Поясним понятие идентификатора основы. В общем случае предполагается, что для одной лексемы в словаре может храниться несколько записей с разными ключами, соответствующими морфологическим основам ее словоформ. Например, для лексемы ЧЕЛОВЕК в словаре хранятся отдельно записи с ключами *человек*-* и *люд*-(*и*); для лексемы ГЛУБОКИЙ — записи с ключами *глубок*-(*ий*), *глубоч*-(*айш-ий*), *глубж*-(*е*). Каждой такой записи соответсвует некоторое уникальной число, называемое далее идентификатором основы. Считается, что имеется возможность по идентификатору лексемы установить идентификаторы соответствующих ей основ, и наоборот. Например, идентификатор лексемы может быть равен идетификатору одной из ее основ.

Существуют разные подходы к задаче назначения идентификаторов основ. Так, идентификатор основы может каким-либо образом быть связанным с физическим расположением соответствующей записи в базе данных, например, содержать номер блока файла базы данных. В этом случае задача поиска нужной основы в базе тривиальна.

Однако такой подход обладает рядом недостатков. Во-первых, трудности возникают, когда лексема имеет несколько разных основ, и нужно найти не ту, идентификатор которой связан с идентификатором лексемы. Во-вторых, зависимость идентификаторов от структуры базы приводит к тому, что идентификаторы лексем изменяются при каждом изменении состава словаря, например, при внесении нового слова, что, как правило, недопустимо.

Поэтому предполагается, что идентификатор основы не содержит информации о ее местоположении в базе данных, а хранится в теле записи при основе. Вычисление идентификатора основы по идентификатору лексемы выполняется системой морфологического синтеза (см. главу 3) и не входит в задачу поиска в базе данных.

Таким образом, задача поиска основы по идентификатору сводится к стандартной для СУБД задаче поиска записи по значению поля. Мы предполагаем, что все основы имеют разные идентификаторы. Мы также предполагаем, что идентификаторы основ следуют в каком-то смысле подряд, то есть для словаря, содержащего N основ, их идентификаторы — номера — расположены в интервале от 1 до числа, близкого к N. Так, в выполненной автором реализации идентификаторы основ назначаются в хронологическом порядке, по мере внесения основ в словарь, и далее закрепляются за основами.

В этом случае задача поиска основы по идентификатору решается стандартными средствами, например, с помощью инвертированного индексного массива. Такой массив состоит из записей постоянной длины. Его запись номер n содержит номер блока, в котором размещена основа с идентификатором n, и номер этой основы в этом блоке.

Алгоритм поиска основы по номеру n состоит в том, что считывается n-я запись инвертированного индексного файла, содержащая номер блока и номер записи в блоке; считывается блок основного массива; производится восстановление сокращенных по методу Купера ключей записей — основ словоформ, вплоть до восстановления нужной записи.

Алгоритм формирования базы данных (пункт 2.3.1) дополняется следующими шагами. При выводе в формируемый массив очередной записи в отдельный временный файл выводится ее идентификатор, номер блока и номер записи в блоке. По окончании формирования базы данных для морфологического анализа этот временный файл упорядочивается по идентификаторам, и затем однопроходным алгоритмом преобразуется в инвертированный индексный массив.

Алгоритм внесения изменений в базу данных (пункт 2.3.3) дополняется так, что при изменении положения каждой записи в базе это изменение фиксируется в соответствующей записи инвертированного индексного массива.

Алгоритм распаковки базы данных (пункт 2.3.2) остается без изменений. Идентификаторы основ не изменяются и не назначаются при операциях с базой данных, поскольку они являются одним из полей информационного тела записей базы. Инвертированный индексный массив не используется алгоритмом распаковки словаря.

Предложенный алгоритм синтеза вдвое менее эффективен, чем алгоритм анализа (пункт 2.4), поскольку в общем случае требует ровно двух обращений к диску. Однако в наиболее важных приложениях задачи морфологического синтеза, таких как нормализация слов или исправление ошибок, поиск основы по идентификатору производится гораздо реже, чем анализ словоформы, поскольку необходимая основа к моменту синтеза часто оказывается уже известной (см. главу 3). В целом же быстродействие приложений морфологического синтеза обычно бывает не столь критичным.

Заметим, что для нужд морфологического синтеза к базе данных добавляется один файл. Никаких изменений в другие файлы базы данных не вносится. Таким образом, эти изменения никак не влияют на эффективность работы программы в режиме морфологического анализа. Более того, если словарь более не должен поддерживать синтез, дополнительный файл может быть просто отброшен.

ВЫВОДЫ

1. Предложена структура базы данных, ориентированной на выполнение решение следующей типичной подзадачи морфологического анализа: найти все записи, ключи которых являются начальными подцепочками предъявленной не ограниченной справа цепочки. Объем информации, хранимой в оперативной памяти, невелик; увеличение объема хранимой на диске информации по

сравнению с традиционной структурой словаря также невелико: для словаря, содержащего 250 тысяч основ, эти показатели составляют около 12.5 Кбайт и 10% соответственно.

2. Показано, что для нахождения всего искомого множества записей достаточно одного обращения к дисковой памяти, что является теоретическим пределом быстродействия для крупного словаря при данных ограничениях на требуемый объем оперативной памяти.
3. Показано, что для решения одной частной задачи, относящейся к проблеме поиска ближайшего ключа и используемой при решении задачи исправления орфографических ошибок, также достаточно одного обращения к дисковой памяти.
4. Предложены алгоритмы формирования (импорта) и распаковки (экспорта) такой базы данных. Алгоритмы не требуют большого объема оперативной памяти, являются быстрыми и однопроходными. Последнее позволяет использовать средства конвейерной обработки данных в памяти ЭВМ при внесении изменений в базу данных.
5. Предложенная структура базы данных может эффективно применяться и для решения задач морфологического синтеза, включая задачи нормализации слов и исправления грамматических ошибок. Используемый при этом дополнительный индексный массив не влияет на эффективность работы программы в режиме анализа и может быть отброшен, если база не используется в режиме синтеза.

ГЛАВА 3.
МОДЕЛЬ МОРФОЛОГИИ ФЛЕКТИВНОГО ЕСТЕСТВЕННОГО ЯЗЫКА

3.1. Постановка задачи построения модели морфологии флективного естественного языка

3.1.1. Основные определения

Под лексемой естественного языка понимается объект, обладающий рядом свойств, а также связанный с непустым множеством объектов, называемых словоформами данной лексемы. Это множество называется парадигмой лексемы. Одни из свойств лексемы называются ее морфологическими характеристиками, например, род существительного; рассмотрение других ее свойств, например, семантических сведений, не входит в задачи морфологической системы. Под характеристикой лексемы понимается сочетание названия свойства и его значения, например: {свойство «род», значение «средний»}.

Лексема выражает понятие, с которым в конкретном случае могут быть связаны какие-либо внеязыковые характеристики, обычно меняющиеся со временем, например, количество неких предметов на некотором складе в данный момент. Поэтому в морфологических системах лексемы идентифицируются каким-либо номером или буквенной цепочкой. По такому идентификатору с лексемой связывается информация вне морфологической подсистемы. Мы будем использовать в качестве идентификаторов лексем числа, гарантируя тем самым их уникальность.

Словоформа некоторой лексемы есть конечная непустая буквенная цепочка, связанная с рядом свойств, таких как, например, падеж существительного. Все свойства словоформы относятся к сфере морфологии, со словоформой прямо не связаны никакие внеязыковые характеристики.

Явление омонимии словоформ состоит в том, что одинаковые буквенные цепочки могут определять разные словоформы одной лексемы или словоформы разных лексем. Например: *вижу машины* vs. *нет машины*; *русская печь* vs. *печь пироги*. Явление вариативности словоформ состоит в том, что словоформы одной лексемы, обладающие одинаковыми наборами характеристик, задаются разными буквенными цепочками. Например: *дверями* vs. *дверьми*.

Характеристиками словоформы, отнесенной к конкретной лексеме, мы для краткости будем называть как характеристики, связанные собственно со словоформой, так и характеристики лексемы.

Среди всех словоформ лексемы выделяется одна, называемая ее словарной, или канонической формой. В нашем определении словарная форма существует всегда: *ножницами — ножницы*; *рада, рады — рад*. Мы будем условно обозначать идентификаторы лексем их словарными формами, например, КРАСНЫЙ — условная запись числового идентификатора соответствующей лексемы.

Под задачей точного морфологического анализа слова понимается установление идентификатора соответствующей лексемы и установление морфологических характеристик соответствующей словоформы, как словарных, так и текстовых. Например: *красными* --> лексема = КРАСНЫЙ, часть речи = прилагательное, падеж = творительный, число = множественное.

Под задачей морфологического синтеза понимается обратное сопоставление, то есть определение буквенной цепочки по идентификатору лексемы и набору характеристик словоформы. Мы будем допускать указание неполного набора характеристик. Например, в случае, если не указано никаких характеристик, в задачу синтеза входит построение всех форм лексемы.

Под задачей морфологического анализа текста понимается разбиение буквенного потока на слова и морфологический анализ каждого слова. Задача разбиения текста на слова даже в случае русского языка не является тривиальной и может требовать лингвистической информации. Например, текст *во что бы то ни стало решить задачу* содержит три слова. Другой пример: *женщина-депутат* vs. *вице-президент*. Трудность при разбиении текста на слова представляет также перенос слов. Так, текст

```
Вышли в эфир теле-              избрании боль-
и радиопередачи для вице-       шинства женщин-
президентов о пере-             депутатов
```

не может быть правильно разбит на слова без привлечения лингвистической и словарной информации. Заметим, что данный текст содержит лексемы *теле-, вице-президент, переизбрание, женщина,*

депутат, но не содержит слов **телеи, *вице-, президент, *вицепрезидент, перо, избрание, боль, *женщина-депутат*. Последняя композитная конструкция не является словарной лексемой.

Ввиду наличия явлений омонимии и вариативности как анализ, так и синтез могут быть неоднозначными. Морфологический анализ называется полным, если строятся все варианты разбора омонимичной буквенной цепочки. Всюду, где не оговорено противное, под морфологическим анализом понимается полный и точный анализ.

Под задачей нормализации, или лемматизации, понимается сопоставление буквенной цепочке идентификатора лексемы либо словарной формы. Заметим, что задача нормализации с помощью идентификатора лексемы в нашем случае является частью задачи анализа, а нормализация с помощью словарной формы требует также синтеза этой формы. Например: *людьми — человек*; *шел, шедшими — идти*. Очевидно, что при омонимии анализируемой цепочки нормализация неоднозначна.

Под задачей обнаружения ошибок понимается установление того, является ли данная буквенная цепочка словоформой какой-либо лексемы. Мы будем рассматривать эту задачу как часть задачи морфологического анализа.

Под ошибкой в связи с задачей исправления ошибок понимается буквенная цепочка, полученная преобразованием заданного вида из словоформы некоторой лексемы. Под задачей исправления ошибок понимается установление исходной («правильной») цепочки. Задача исправления ошибок сводится к двум подзадачам: во-первых, нахождение полного прообраза данной цепочки при всех преобразованиях рассматриваемого вида в множестве словоформ всех лексем; во-вторых, определение вероятности каждого построенного варианта исправления. В данной работе исследуется в основном первая из этих подзадач. Установление конкретного преобразования, исказившего словоформу, называется объяснением ошибки. Как правило, исправление и объяснение ошибки неоднозначны.

Алгоритмы исправления ошибок существенно зависят от того, какой класс преобразований, т.е. подразумеваемый механизм совершения ошибок, рассматривается. В данной главе уделяется внимание классу, названному нами грамматическими ошибками; глава 4 посвящена задаче исправления ошибок типа опечатки.

Под задачей приближенного морфологического анализа понимается установление вероятных морфологических характеристик цепочки, не являющейся словоформой какой-либо известной лексемы. В задачу приближенного морфологического анализа может входить установление всей вероятной парадигмы подходящей лексемы, что делает возможным синтез других форм данного слова и, в частности, его приближенную нормализацию путем построения словарной формы. Как и в случае задачи исправления ошибок, задача приближенного анализа сводится к поиску всех «аналогичных» лексем и выбору наиболее вероятных соответствий. Конкретное содержание понятия «аналогичности» также может быть различным.

Под морфологической программной системой понимается набор программных средств и лингвистической информации, позволяющий решать комплекс задач, относящихся к сфере морфологии, то есть к сфере связи между буквенным представлением слов, с одной стороны, и набором лексем данного естественного языка, их парадигм и морфологических характеристик, с другой стороны. Таким образом, к задачам морфологической системы относятся все перечисленные выше задачи.

3.1.2. Требования, предъявляемые к морфологической системе

Условия эксплуатации накладывают ряд технических требований на морфологическую систему, например, требование высокого быстродействия при условии работы с крупным словарем. Кроме того, морфологическая подсистема является инструментальным средством, интегрируемым в прикладную систему, такую, как информационно-поисковая система или система машинного перевода, что также накладывает на нее ряд требований, например, требование экономии оперативной памяти ЭВМ. Конкретно, мы будем считать, что доступный объем оперативной памяти много меньше объема любой существенной части словаря. В выполненной автором реализации часть составленного при участии автора лингвистического обеспечения, содержащаяся в оперативной памяти ЭВМ, занимает 14.5 Кбайт (без учета индексных массивов словаря, см. главу 2).

Требование экономии оперативной памяти распространяется также и на программное обеспечение. Поэтому всюду, где это возможно, одна задача сводится к другой, возможно, с некоторыми модификациями соответствующего алгоритма. Такое сведение к другой задаче производится даже в том случае, когда для решения данной конкретной задачи можно предложить и лучший независимый алгоритм. Это относится к блокам исправления ошибок и приближенного

анализа. Основными относительно независимыми блоками системы являются процедуры анализа и синтеза.

Далее, требуется однопроходная схема работы системы, при которой входной поток данных поступает на вход системы небольшими порциями, а результаты его обработки становятся доступны по мере поступления входной информации. Требование экономии оперативной памяти существенно также для современных многозадачных операционных систем, таких как Windows, UNIX, OS/2; требование однопроходности существенно, в частности, для применения средств конвейерной обработки в UNIX и OS/2.

Относительно словаря предполагается, что он имеет значительный объем. Это диктуется уже следующим соображением. Идентификаторы лексем не должны меняться при изменении словаря, поскольку к таким идентификаторам в прикладной системе привязана отдельно хранящаяся информация, собранная за длительное время, и эта информация не должна зависеть от изменений в морфологической подсистеме. Таким образом, идентификаторы лексем должны храниться в словаре, а не вычисляться по позиции лексемы в нем. Для 100-тысячного словаря только суммарный объем идентификаторов составляет 300 Кбайт, в то время как реальные словари содержат обычно существенно большее количество основ.

Основным предположением относительно физической природы словаря, с которым должна работать система, является то, что он размещается на устройстве дисковой памяти, позволяющем считывать любой его небольшой участок (см. тж. пункт 2.1.2). Предполагается, что считывание информации с диска есть наиболее длительная по времени операция всего цикла анализа словоформы.

Наконец, система должна удовлетворять требованию отделения лингвистической информации от программной, ставшему стандартным для лингвистических систем. Оно обосновывается тем, что программы и массивы лингвистической информации создаются и модифицируются разными людьми и в разное время. Программные средства технологического пакета могут быть полностью заменены без необходимости изменения лингвистической информации. На базе той же лингвистической информации могут решаться новые задачи. Наоборот, пакет программных средств может быть настроен на обработку текстов на ином языке. Последнее свойство называется языковой независимостью программного пакета. Языковая независимость возможна в некотором более или менее широком классе языков, имеющих определенные сходные черты.

В инструментальной лингвистической системе необходима такая организация лингвистической модели, чтобы однажды построенное описание языка позволяло решать различные задачи. Это тем более существенно, когда необходимо решать несколько задач одновременно, например, задачи анализа и синтеза при нормализации слов словарными формами. В таких случаях в памяти ЭВМ должна храниться только одна копия лингвистической информации, а не отдельно модель анализа и отдельно модель синтеза.

Аналогично, языковая независимость существенна в прикладных системах, имеющих дело с несколькими языками одновременно, например, в системах автоматического перевода. Свойство языковой независимости позволяет загружать в память ЭВМ только одну копию программного кода морфологической системы, работающую с различными массивами лингвистических данных.

3.1.3. Общая структура лингвистической модели флективного естественного языка

Точкой соприкосновения лингвистической и программной части морфологической системы является модель языка, определяющая основные понятия, их структуру и свойства для выбранного класса языков, а также определяющая метаязык, используемый для создания и отладки лингвистической информации. Таким образом, модель языка определяет класс языков, на работу с которыми ориентирована данная система.

Флективным языком мы называем язык, словоформы которого образуются путем соединения, конкатенации буквенных цепочек — флексий, или морфов, с каждой из которых связан набор морфологических характеристик. Морфы сочетаются в словоформах языка в соответствии с определенными закономерностями. Первый из морфов словоформы определяет лексему и является специфическим для данной лексемы, остальные являются стандартными для данного языка. Например, *читающийся*:

```
чита-      --> лексема=ЧИТАТЬ, часть речи=глагол,
  -ющ-     --> залог=действительный, время=настоящее,
```

```
-ий-  --> падеж=имен., род=мужск., число=единств.,
  -ся --> возвратность=возвратный.
```

Сочетание значений морфологических характеристик, кумулятивно выражаемое одним морфом, будем называть грамматемой. В приведенном примере «падеж = именительный, род = мужской, число = единственное» есть грамматема, связанная с морфом *-ий-*. Элемент грамматемы, т.е. отдельное свойство с указанием его значения, будем называть граммемой.

Таким образом, рассматривается только суффиксальное словоизменение. Для простоты не рассматриваются префиксы, хотя они легко могут быть введены в модель. Модель не распространяется на интрофлективные языки, то есть не рассматриваются инфиксы. Напротив, агглютинация рассматривается как частный случай флективности, то есть модель может применяться для описания агглютинативных языков. Ради упрощения программного и лингвистического обеспечения системы не рассматриваются характерные для флективных языков фузионные явления на стыках морфов. Такие явления сводятся к введению новых морфов или супплетивных основ.

Задача морфологического анализа в данном случае сводится к расчленению буквенной цепочки на основу и суффиксальные морфы и определению характеристик, связанных с каждой частью. Задача синтеза сводится к поиску основы лексемы и морфов, обладающих необходимыми характеристиками, и соединению их в должном порядке. В рассматриваемой модели такое соединение производится механически, без изменения буквенного состава морфов.

3.2. Лингвистическая модель морфологии флективного естественного языка

3.2.1. Основные особенности лингвистической модели языка

Число морфов, составляющих словоформу, фиксировано для словоформ данной части речи. Другими словами, словоформа разбита на определенное число позиций, занятых морфами. Позиции нумеруются с нуля, так что основа занимает нулевую позицию, первый суффикс — первую, и т.д. Рассматриваются также пустые морфы, занимающие соответствующую позицию. С пустым морфом часто связывается пустой набор характеристик либо характеристики, выражающие отсутствие какого-либо свойства. Например: *древн-*-ий* vs. *древн-ейш-ий*. В данном случае * означает значащий ноль, однако иногда приходится вводить незначащие, фиктивные пустые морфы, только ради сохранения единообразия описания слов данной части речи. Они скрыты от конечного пользователя, поскольку при выдаче результатов морфемного разбора такие позиции опускаются.

Лингвистическое обеспечение системы состоит из двух блоков: таблицы, задающие общие закономерности грамматики данного языка, и словарь, задающий лексику языка и особенности изменения отдельных слов. Таблицы содержат информацию о морфах, грамматемах и связях между ними. Словарь содержит информацию об основах и их свойствах.

Грань между грамматической и лексической информацией не является четкой. Многие явления словоизменения могут с равным успехом либо трактоваться как относящиеся к классу слов и помещаться в таблицы, либо трактоваться как относящиеся к конкретным словам и помещаться в словарь. Это позволяет выбирать оптимальное соотношение объема размещенного на диске словаря и размещенных в памяти таблиц.

В таблицах декларативно представлена информация трех типов, одного типа — о связи между грамматемами и морфами, стоящими на одной позиции, и двух типов — о связи между разными позициями словоформы. Эти три типа информации следующие:

1) Морфы соответствуют грамматемам. Так, именительному падежу единственного числа прилагательных соответствует морф *-ый-*, а творительному множественного — морф *-ыми-*.

2) Морфы управляют морфами. Так, в причастиях типа *читающий*, *читаемый* после *-ющ-* пишется *-ий-*, но после *-ем-* пишется *-ый-*. Это синтактика морфов одной позиции по отношению к морфам другой.

3) Грамматемы управляют грамматемами. Так, можно писать *(чита)-ющ-(ий)-ся* и нельзя *(чита)-ем-(ый)-ся*, поскольку граммема возвратности не совместима с граммемой страдательного залога. Это синтактика грамматем одной позиции по отношению к грамматемам другой.

Существенно, что среди правил двух последних типов фигурируют только правила вида «*после* того-то можно/нельзя/нужно писать то-то», но не рассматриваются правила вида «*перед* тем-то пишется то-то». Такое ограничение упрощает модель и ее реализацию. Заметим, что в инвентарь понятий модели

не входят такие понятия, как мягкость/твердость основы либо окончания, следовательно, правила вида «*-ий-* пишется после мягкого или шипящего» не являются выразимыми. Вместо этого в таблицах конкретно указывается, что после *-ющ-* пишется *-ий-*. Ввиду небольшого объема таблиц это не приводит к существенному увеличению объема лингвистической информации, но упрощает ее структуру.

Для полного описания грамматики языка необходимо задать следующие таблицы: таблицу морфов, список морфов, список грамматем, список масок. Данные четыре таблицы необходимо представить отдельно для каждой позиции каждой части речи. Всюду далее при описании таблиц будем иметь в виду таблицы для некоторой фиксированной позиции фиксированной части речи.

3.2.2. Таблица морфов

В простейшем случае структуру представления лингвистической информации в модели можно проиллюстрировать схемой рис. 1 [6, 20]. Строки таблицы морфов образуют парадигмы, разные строки соответствуют различным классам словоизменения. Грамматемы соответствуют столбцам таблицы и образуют ее шапку, но для удобства они вынесены в отдельный список. Словарь содержит основы слов с указанием соответствующих строк таблицы морфов.

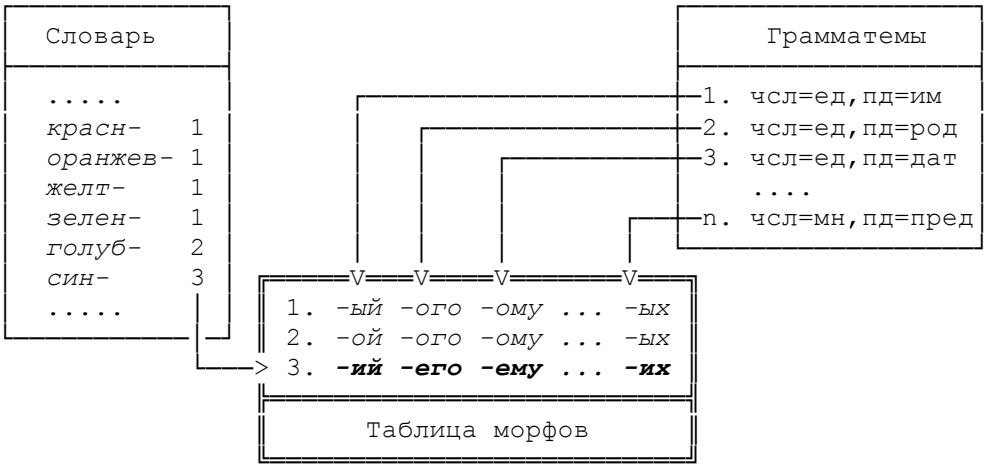


Рис.1. Таблица морфов

Алгоритмы как анализа, так и синтеза очевидны. При анализе входная цепочка разбивается на основу, имеющуюся в словаре, и морф, имеющийся в таблице. Существенно, что рассматриваются только такие разбиения, при которых найденный морф содержится в строке, указанной в словаре при найденной основе; остальные варианты разбиения считаются ошибочными и игнорируются. По найденным основе и морфу устанавливается номер лексемы, хранящийся при основе в словаре, и набор характеристик, заданный грамматемой, соответствующей столбцу таблицы морфов, содержащему найденный морф. При выборе столбца учитывается только указанная строка таблицы. Как разбиение буквенной цепочки, так и выбор столбца, содержащего морф, неоднозначны. При полном анализе рассматриваются все возможные варианты как разбиения цепочки, так и выбора столбца, содержащего морф в нужной строке.

Аналогично при синтезе в списке грамматем выбирается грамматема, содержащая нужные характеристики, то есть определяется столбец таблицы. В словаре выбирается основа по номеру лексемы. При основе указан номер строки таблицы. К выбранной основе присоединяется морф, стоящий в таблице на пересечении выбранного столбца и выбранной строки. Как выбор основы, так и выбор грамматемы могут быть неоднозначны, так, несколько грамматем содержат граммему «число=единственное». При полном синтезе строятся все варианты.

Таким образом, таблицей морфов, построенной для определенной позиции словоформ определенной части речи, называется прямоугольная матрица, составленная из морфов — буквенных цепочек. Ее столбцы соответствуют всем возможным грамматемам для данной позиции, а строки — различным вариантам словоизменения, касающимся данной позиции. Заметим, что столбцы таблицы соответствуют некоторой формализации понятия морфемы, а строки — понятия парадигматического класса в ограничении на данную позицию.

3.2.3. Маски и чередование основ

Для отражения явлений супплетивизма и чередования букв одной лексеме может соответствовать несколько основ. Такие основы называются чередующимися основами. Например, лексеме РЕБЕНОК соответствуют чередующиеся основы *ребенок*-(*), *дет*-(*и*) (супплетивизм) и *ребенк*-(*а*) (чередование, формально сведенное к супплетивизму). Все три основы образуют отдельные, независимые записи в словаре. Однако каждая из этих основ имеет ущербную парадигму, то есть может сочетаться не со всеми грамматемами. Так, для основы *дет*- запрещены все формы единственного числа.

Для описания ущербности парадигм введем понятие маски [6]. Маска представляет собой строку некоторых знаков, скажем, «+» и «-». Длина строки равна длине строки таблицы морфов, или, что то же самое, количеству грамматем данной позиции данной части речи. Каждый из этих знаков, соответствующий некоторому столбцу таблицы морфов, задает информацию о возможности выбора данного столбца при анализе и синтезе форм некоторой основы. Таким образом, маска запрещает, маскирует некоторые столбцы таблицы морфов. Список используемых в системе для данной позиции данной части речи масок называется списком масок. Пример использования масок дан на рис. 2.

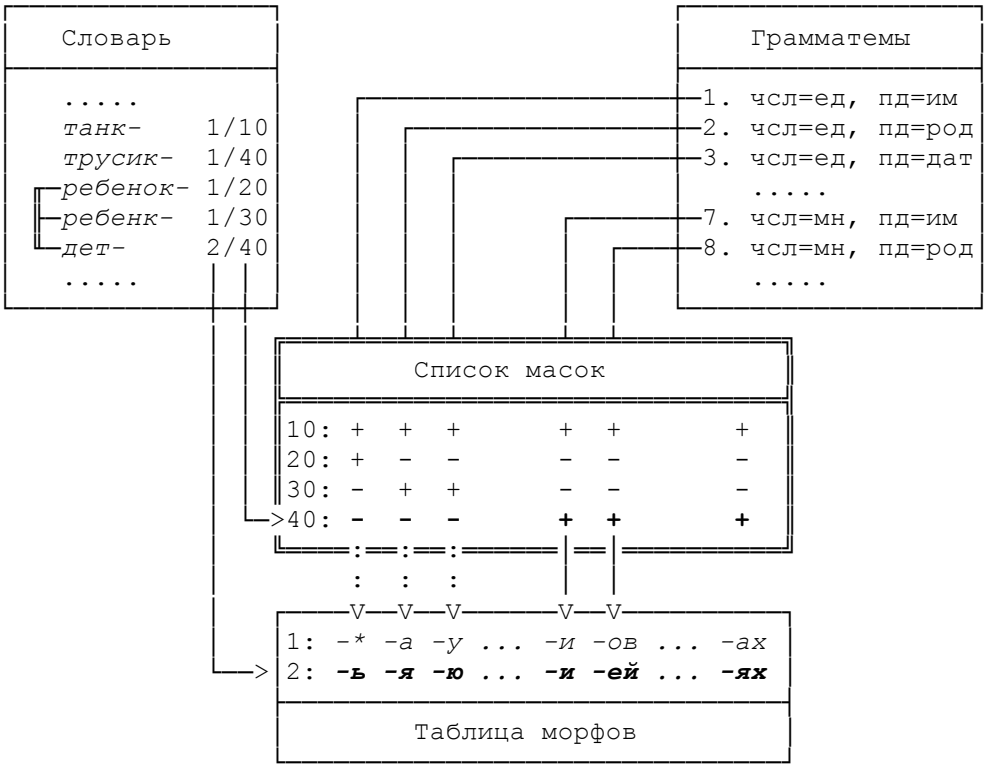


Рис.2. Маски и чередование основ

На рис. 2 в словаре кроме номеров строк таблицы морфов основам приписаны также номера масок. Они даны в строках словаря через косую черту. Чередующиеся основы одной лексемы условно показаны соединенными. Здесь основе лексемы ТАНК разрешены все формы, поскольку соответствующая маска номер 10 содержит все «+». Формы данной основы строятся по строке номер 1 таблицы морфов: *танк*-*, *танк-а*, ..., *танк-ах*. Лексема ТРУСИКИ образует формы по той же строке 1 таблицы морфов, но маска 40 разрешает ей только формы множественного числа. Лексема РЕБЕНОК имеет все формы, но конкретная ее основа *дет*-, как показано на рис. 2 стрелками, образует только формы множественного числа, соответственно маске 40; остальные формы данной лексемы образуются другими ее основами.

Именно так удается применить механизм масок к совместному описанию трех явлений: ущербность парадигмы, чередование, супплетивизм. Четвертое применение масок — описание вариативности, например: *дверями* / *дверьми*, *псалтырем* / *псалтырью*. При этом в масках различных основ одной лексемы области, помеченные знаком «+», просто перекрываются. Механизм масок и

чередования основ позволяет построить практически любую парадигму без существенного дублирования информации путем комбинирования одних и тех же строк морфов.

Маски задают информацию о возможности или невозможности выбора данного столбца парадигмы. Однако понятие возможности можно трактовать шире, включая в него понятия затрудненности, факультативности и т.п. Так, в выполненной автором реализации знаки «%» и «&» в масках выражают соответственно затрудненность и факультативность образования формы; употребление символа «*» в масках рассматривается в пункте 3.2.6.

При описании чередований в виде отдельных основ общее количество основ в словаре возрастает. Однако сжатие словаря по методу Купера в значительной мере компенсирует рост его объема, поскольку вместо, например, фрагмента *глубж-*, *глубок-*, *глубоч-* в сжатом словаре хранится 0+*глубж-*, 4+*ок-*, 5+*ч-*.

Случаи чередований могут формально обслуживаться не только механизмом масок и чередования основ, но и добавлением новых, лингвистически не содержательных позиций либо формальным изменением морфемного членения слова. Так, например, беглое *-о-* в словах *ребенок*, *станок* можно было бы формально описать как форматив *-ок-/-к-*, занимающий отдельную позицию, со строкой таблицы морфов вида *-ок-*, *-к-*, ..., *-к-*. Другим вариантом описания того же чередования является введение строки таблицы морфов вида *-ок, -ка, ..., -ках*. Однако столь искусственные приемы обычно являются нежелательными.

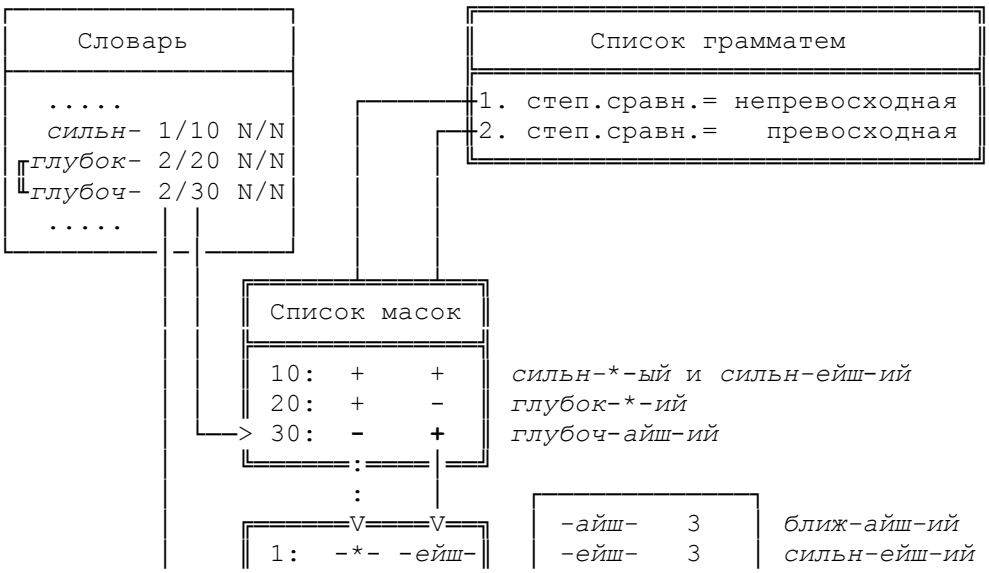
3.2.4. Сочетание нескольких суффиксов

В модели принимается, что для одной формальной части речи каждая словоформа формально делится ровно на N позиций, возможно, выраженных пустыми морфами, причем N не зависит ни от лексемы, ни от словоформы. Позиции нумеруются с нуля, нулевая позиция соответствует основе. Некоторые позиции в словоформе могут быть фиктивными и вводятся лишь для единообразия описания.

Для учета нескольких позиций строятся таблицы морфов, списки масок и грамматем для каждой позиции в отдельности, кроме нулевой позиции. Заметим, что роль таблиц для нулевой позиции играет словарь основ.

Рассмотрим строение русских прилагательных, принимая во внимание возможность образования превосходной степени: *сильн-ейш-ий*. В этом случае информация, приведенная на рис. 1, является недостаточной. Более точно, таблицы рис. 1 описывают позицию номер 2, занимаемую окончанием, например, *-ий*. Пример таблиц для позиции номер 1 прилагательных приведен на рис. 3. Соответствующей модификации подвергается также словарная информация. Каждая основа прилагательного в словаре снабжается отдельно номером строки таблицы морфов и номером маски для позиции номер 1 и аналогичными номерами для позиции номер 2.

Для полноты изложения на рис. 3 приведена еще одна таблица — список морфов. Она рассматривается в пункте 3.2.5 и не требуется для понимания настоящего изложения.



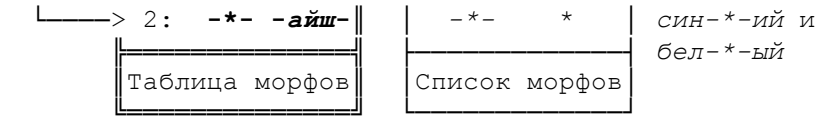


Рис.3. Таблицы для позиции 1 прилагательных

Для описания лексем, присоединяющих несколько суффиксов, вводятся также обсуждаемые ниже механизмы взаимной зависимости между морфами и между морфемами разных позиций одной словоформы.

3.2.5. Синтактика морфов

Для каждой позиции словоформы все предыдущие позиции в модели формально играют роль, подобную роли основы, а все последующие подчиняются ей аналогично тому, как в простейшем случае выбор единственного суффиксального морфа подчинен основе. Так, после *-ейш-* всегда пишется *-ий*, но не *-ый*. В соответствии с этим составляются как бы мини-словари морфов для каждой позиции. В них для морфов указываются номера строк таблиц морфов для *всех* следующих позиций, подобно тому, как для основ номера строк таблицы морфов указывается в словаре основ. Такие списки называются списками морфов.

Для примера рассмотрим список морфов позиции 1 для прилагательных, данный на рис. 3. Остальные таблицы для позиции 1 даны на рис. 3, а для позиции 2 — на рис. 1. Другими словами, в словоформах *сильн-*-ый, сильн-ейш-ий* морфы *-*-, -ейш-* выбираются согласно рис. 3, а *-ый, -ий* — согласно рис. 1.

Числа в списке морфов позиции 1 являются номерами строк таблицы морфов позиции 2. Таким образом, морфу *-ейш-* соответствует строка номер 3 на рис. 1: *-ий, -его, -ему, ..., -их*. При этом словарная информация игнорируется.

В списке морфов вместо числа допустимо специальное значение «*», означающее отсутствие выбора конкретной строки. В этом случае используется значение, указанное при морфе предыдущей позиции, в данном случае — при основе в словаре. Если при морфе предыдущей позиции указано «*», просматривается предыдущий к нему, и т.д. Если же просмотр значения в словаре так и не привел к определению номера, выдается сообщение об ошибке. На рис. 3 морфу -*- соответствует значение «*», следовательно, для определения выбираемого номера строки таблицы морфов позиции 3 необходимо обратиться к словарной информации, которая различается для слов *син-*-ий* и *бел-*-ый*.

На рис. 3 словарная информация приведена не полностью. Приведем ее здесь полностью для лексемы СИЛЬНЫЙ: *сильн-* 1/10 1/10. Первая пара чисел относится к позиции 1: строка 1, маска 10 на рис. 3. Вторая пара чисел относится к позиции 2: строка 1, маска 10 на рис. 1. Поскольку на рис. 1 не приведен список масок, будем считать, что маска 10 содержит все «+».

Рассмотрим пример работы алгоритма синтеза форм лексемы СИЛЬНЫЙ по таблицам рис. 1 и 3 со словарной информацией «*сильн-* 1/10 1/10». Обратим внимание на механизм выбора окончаний *-ему* и *-ому* в двух случаях. Выбор *-ому* в *сильному* определяется словарными свойствами данной лексемы, в то время как выбор *-ему* в *сильнейшему* определяется грамматикой языка и не зависит от словарных свойств конкретной лексемы.

```
На входе : СИЛЬНЫЙ, степ.сравн.=превосх., падеж=дательный
В словаре: сильн-, 1-я позиция: 1/10, 2-я позиция: 1/10
Позиция 0: морф сильн- (из словаря)
Позиция 1: строка  1   (указана при сильн- позиции 0)
           столбец 2   (превосх. степ. в списке грамматем)
           маска   +   (маска 10 позиции 1)
           морф -ейш-  (в таблице морфов рис. 3)
Позиция 2: строка  3   (указана при -ейш- позиции 1)
           столбец 3   (дат. падеж в списке грамматем)
           морф -ему   (в таблице морфов рис. 1)
           маска   +   (маска 10 позиции 2)
На выходе: сильнейшему

На входе : СИЛЬНЫЙ, степ.сравн.=непрев., падеж=дательный
В словаре: сильн-, 1-я позиция: 1/10, 2-я позиция: 1/10
```

```
Позиция 0: морф сильн-  (из словаря)
Позиция 1: строка  1    (указана при сильн- позиции 0)
           столбец 1    (непрев. степ. в списке грамматем)
           маска   +    (маска 10 позиции 1)
           морф -*-     (в таблице морфов рис. 3)
Позиция 2: строка  *    (указана при -*- позиции 1)
           строка  1    (указана при сильн- позиции 0)
           столбец 3    (дат. падеж в списке грамматем)
           морф -ому    (в таблице морфов рис. 1)
           маска   +    (маска 10 позиции 2)
На выходе: сильному
```

Алгоритм анализа действует аналогично. Заметим, что для последней позиции список морфов не требуется, поскольку в нем не указывается никакая информация. На практике он все же составляется в целях облегчения обнаружения ошибок в таблице морфов.

Для упрощения структуры описания и алгоритмов в модели все словоформы одной лексемы рассматриваются как содержащие одно и то же число позиций. Однако это не всегда соответствует содержательному морфемному делению словоформ, например: *чит-а-ть-** vs. *чит-а-ющ-ие-ся*. Знаком * обозначен значащий ноль — отсутствие частицы *-ся*. Необходимо ввести обозначение для фиктивных позиций, содержащих незначащий ноль. Обозначим его знаком #, например: *чит-а-ть-#-**. Фиктивная позиция не обязательно соответствует пустой грамматеме (пустому множеству граммем), но на практике это обычно так.

Для описания фиктивных позиций вводится еще один сорт числовых значений в списке морфов. Вместо номера строки таблицы морфов указывается, со специальной пометой «–», номер столбца той же таблицы, то есть номер определенной грамматемы в списке грамматем. Фрагмент списка морфов и соответствующих таблиц для глаголов дан на рис. 4. Данный упрощенный пример не соответствует реальному лингвистическому обеспечению системы.

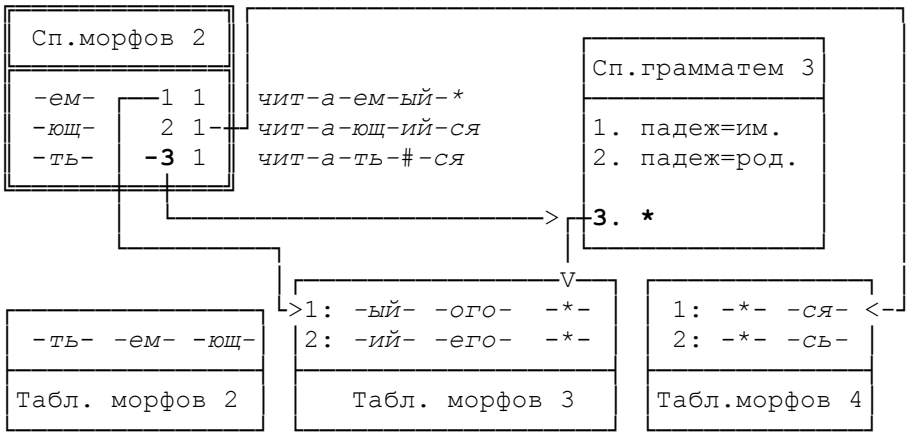


Рис. 4. Фрагмент таблиц для нескольких позиций

В случае, когда в качестве номера строки по изложенным выше правилам найден номер столбца фиктивной позиции, таблица морфов по данной позиции игнорируется, соответствующий морф полагается равным пустой цепочке, однако соответствующая грамматема учитывается. При выдаче результатов морфного разбора словоформы фиктивные позиции могут быть скрыты от конечного пользователя системы, то есть разбор цепочки *читать* может быть показан как *чит-а-ть-**, а некак *чит-а-ть-#-**. Другой механизм описания фиктивных позиций приведен в пункте 3.2.6.

3.2.6. Синтактика морфем

Формальной морфемой называется номер столбца таблицы морфов. Морфеме соответствует определенная грамматема в списке грамматем, а также определенный столбец в списке масок. Там, где это не вызывает недоразумений, морфемы упоминаются не по номерам, а по характерному морфу, знаку

маски или грамматеме. Например, если говорится о морфеме *-ейш-* или морфеме «степень сравнения = превосходная», то имеется в виду морфема номер 2 на рис. 3.

Взаимосвязь морфем различных позиций может заключаться в несовместимости или в затрудненности совмещения этих морфем. Под несовместимостью морфем понимается невозможность их совмещения в одной словоформе, например, морфемы страдательного залога и возвратности несовместимы: **чит-а-ем-ый-ся.* В модели не задается информация о несовместимости соответствующих грамматем, вместо этого несовместимыми объявляются соответствующие им морфемы. В модели выражается только попарная несовместимость морфем. В редких случаях, когда требуется выразить более сложную зависимость, одна морфема разбивается на несколько, по-разному сочетающихся с другими, причем это обычно отвечает лингвистической природе описываемого явления.

Для выражения информации о совместимости морфем каждой морфеме сопоставляется по одной маске на морфемы последующих суффиксов. Заметим, что, поскольку матрица попарной совместимости симметрична, она всегда может быть выражена с помощью треугольной матрицы, задаваемой через маски только на последующие морфемы.

Аналогично тому, как это рассмотрено в пункте 3.2.6, для указания номера маски допускается специальное значение «*», означающее отсутствие маски. В этом случае номер определяется предыдущей позицией. Если и она дает номер «*», просматривается предыдущая к ней позиция, и т.д. вплоть до словаря основ. Если этот процесс не позволяет определить номер маски, выдается сообщение об ошибке. На практике номер маски в большом числе случаев определяется словарной информацией.

В то время как символ «–» в маске трактуется как запрещение морфемы, символ «+» трактуется как ее разрешение. Другие возможные значения символов масок даны в пункте 3.2.3. Так, морфема *-айш-* разрешает падежные окончания даже для тех основ, для которых словарь указывает маску, запрещающую их. Например, для основы *глубоч-* запрещены формы типа *глубоч-*-ий*, однако формы типа *глубоч-айш-ий* возможны.

Изложенный механизм не является достаточным. В большинстве случаев возникает необходимость запретить одни морфемы какой-либо позиции, не запрещая и не разрешая другие. Для этого кроме «+» и «–» вводится значение элемента маски «*». В случае, если элементом маски для некоторой морфемы является «*», предпринимаются те же действия, что и для номера маски «*», то есть просматривается маска, налагаемая предыдущей позицией.

Вместо указания номера маски, разрешающей или запрещающей отдельные морфемы, может указываться сразу номер морфемы, со специальной пометой «–». На данный случай полностью переносятся правила, изложенные в пункте 3.2.5 для фиктивных позиций. Отличие указания такого номера от указания маски с единственным знаком «+» состоит в том, что в последнем случае позиция считается значащей, а не фиктивной.

Каждой морфеме соответствует два сорта информации: грамматема и набор масок на все последующие позиции. Поэтому удобным местом для указания такого набора является список грамматем. На рис. 5 дан пример, продолжающий пример рис. 4.

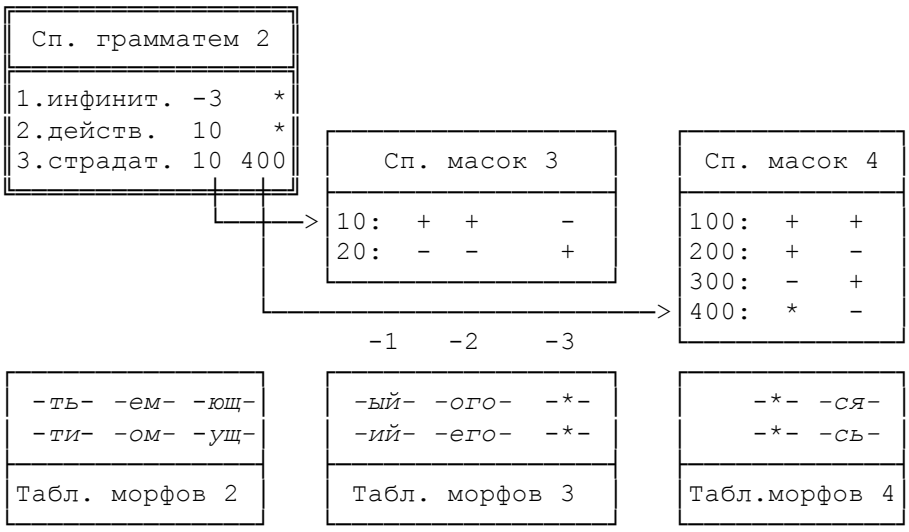


Рис. 5. Синтактика морфем

Рассмотрим управление позицией 3 со стороны позиции 2. Инфинитив выбирает столбец 3 по фиктивной позиции 3: *чит-а-ть-#-**; альтернативой был бы выбор маски 20, и тогда позиция 3 считалась бы значимой: *чит-а-ть-*-*-**. Действительный и страдательный залог разрешают падежные окончания и запрещают фиктивный столбец 3. Словарная информация о данных морфемах не требуется.

Рассмотрим управление позицией 4 со стороны позиции 2. Инфинитив и действительный залог не задают номера маски, поэтому используется управление со стороны позиции 1 либо словарная информация, что дает возможность возвратным глаголам выбирать маску 300 (*смеяться*), а непереходным глаголам выбирать маску 200 (*идти*). Страдательный залог запрещает возвратность, но не запрещает и не разрешает невозвратную форму. В последнем случае снова используется управление со стороны позиции 1 либо словаря. Таким образом, при наличии морфемы страдательного залога значение маски для морфемы возвратности берется из маски 400, запрещающей ее; однако для морфемы невозвратности значение маски берется соответственно из масок 100 для *читать*, 200 для *идти*, и 300 для *смеяться*. Поскольку в этом случае для глагола *смеяться* запрещены все возможные морфемы позиции 4, словоформ с морфемой страдательного залога у данной лексемы быть не может. Заметим, что никакой дополнительной словарной информации, указывающей на этот факт, не требуется.

3.2.7. Списки грамматем

Рассмотрим подробнее структуру списка грамматем. Встречаются случаи, когда некоторые морфемы совпадают во всех отношениях, кроме соответствующих им грамматем. Такие морфемы объединяются в одну морфему, которой соответствует множество грамматем. Такая ситуация возникает, например, для русских прилагательных: *нет белой бумаги, к белой бумаге, белой бумагой, о белой бумаге*. В этом случае столбцу таблицы морфов, содержащему морф *-ой*, ставится в соответствие множество из четырех грамматем:

```
род=женский, число=единственное, падеж=родительный;
род=женский, число=единственное, падеж=дательный;
род=женский, число=единственное, падеж=творительный;
род=женский, число=единственное, падеж=предложный.
```

Окончательно список грамматем имеет следующий вид. Элемент этого списка включает, во-первых, одну или несколько грамматем — наборов значений характеристик, и, во-вторых, перечисление номеров масок на каждую из следующих позиций.

Все таблицы, такие как таблицы морфов, списки морфов и списки масок, существуют для каждой позиции, кроме нулевой, у каждой части речи. Поскольку морфы нулевой позиции определяются словарем, не существуем таблицы морфов нулевой позиции, а, значит, и списков морфов и масок. Особенностью, отличающей списки грамматем от всех других таблиц, является то, что список грамматем строится и для нулевой позиции. В него включаются грамматические характеристики, выражаемые основами, такие, как род существительных или переходность глаголов. Более подробно списки грамматем нулевой позиции рассмотрены в пункте 3.2.8.

3.2.8. Части речи и словарь основ

Грамматической частью речи называется значение признака «часть речи», входящее в набор характеристик лексемы. Оно является результатом морфологического анализа словоформы.

На рис. 1, 2 и 4 представлены фрагменты таблиц для прилагательных, существительных и глаголов соответственно. Внутри каждой части речи таблицы представляют собой взаимосвязанную систему, однако таблицы одной части речи никак не связаны с таблицами другой. Формат и семантика таблиц одинаковы для разных частей речи, и алгоритмы работы с ними не различаются.

Такие отдельные подсистемы таблиц называются в модели техническими, или формальными, частями речи. Каждая формальная часть речи задается независимой реализацией системы таблиц. Фактически, таблицы для разных формальных частей речи описывают отдельные языки. Например, в рамках единого лингвистического обеспечения возможно существование формальных частей речи «русские существительные» и «английские существительные».

Параметром формальной части речи является *только* число позиций. Оно определяет количество таблиц, необходимых для описания данной части речи. В действующей системе

формальные части речи не имеют названия. Название технической части речи является условностью, принятой в документации и в комментариях к лингвистическому обеспечению системы. Оно не определяет значение грамматической части речи лексемы. Более точно, каждая лексема в словаре относится к какой-либо определенной формальной части речи, однако название грамматической части речи, входящее в набор ее характеристик, может быть произвольным.

Как правило, технические части речи хорошо соответствуют общепринятым грамматическим частям речи. Так, к технической части речи «существительные» обычно относятся именно существительные в общепринятом понимании этого термина. Однако это соответствие не всегда удобно сохранять. Формы таких существительных как *булочная, учащийся* удобнее строить по таблицам прилагательных и глаголов соответственно. Все неизменяемые слова, такие как наречия, предлоги, союзы, удобно отнести к одной части речи «неизменяемые». Собственно грамматическая часть речи является при этом словарной информацией, различной для таких лексем одной технической части речи, как *звуковой, часовой, первый, самый*.

Заметим, что словарь содержит для нулевой позиции ту информацию, которая для остальных позиций содержится в списках начертаний, таблицах морфов и списках масок, поэтому данные три таблицы не строятся для нулевой позиции. Однако словарь не содержит грамматем. Грамматемы, выражаемые основами, собраны в список грамматем нулевой позиции. Именно из списка грамматем нулевой позиции формальной части речи прилагательные лексема *звуковой* получает грамматему «часть речи = прилагательное», а лексема *часовой* получает из того же списка характеристики «часть речи = существительное, род = мужской, одушевленность = одушевленное».

Поскольку граммемы, выражаемые основой, могут быть совместимыми или нет с граммемами других позиций, в данном списке, как и в остальных списках грамматем, указываются номера масок на все последующие позиции. Например, непереходность глагола указывает на невозможность образования форм страдательного залога или возвратных форм, и соответствующая маска указывается в списке грамматем нулевой позиции при каждой грамматеме, содержащей граммему непереходности.

Таким образом, не требуется специальной словарной информации для отражения явлений, являющихся законами грамматики языка. Например, для существительных женского рода нет запрета на образование формы творительного падежа на *-ою*: *с мамою*. Для существительных мужского рода такая форма запрещена маской, указанной в списке грамматем нулевой позиции. Таким образом, для отражения данного явления никакой дополнительной информации, кроме выбора указания номера грамматемы нулевой позиции, в словаре не требуется. Естественным образом описываются и исключения. Так, в списке грамматем существительных имеется такая строка, которая содержит грамматему как для мужского рода, но маски как для женского: *с папою*.

Однако невозможность образования какой-либо формы может также быть индивидуальным свойством основы, не связанным с ее морфологическими характеристиками. Например, для лексемы *мечта* затруднено образование формы родительного падежа множественного числа, в то время как лексема *пята* чаще всего употребляется именно в этой форме: *до пят*. Наиболее часто необходимость запрещения основе форм возникает в связи с чередованиями. Подробнее данный вопрос изложен в пункте 3.2.3.

Для отражения индивидуальных свойств основы в словаре при основе также указывается набор масок на все последующие позиции. Маски, указанные в словаре, имеют меньший приоритет в смысле алгоритма определения значения маски из пункта 3.2.6, чем маски из списка грамматем нулевой позиции. Последнее обстоятельство приводит к тому, что в масках, номера которых указываются в списках грамматем, все значения, не равные «–», обычно равны «*»; в масках же, указанных в словаре, обычно большинство значений равны «+».

Набором номеров масок данной части речи называется массив чисел, содержащий по одному номеру маски для каждой позиции данной части речи, начиная с первой. Аналогично определяется набор номеров строк таблиц начертаний. Вместо номеров строк таблицы начертаний и номеров масок в наборе могут указываться номера морфем фиктивных позиций, в точности так же, как это определено в пунктах 3.2.5 для морфов и 3.2.6 для морфем, соответственно. Таким образом, каждая запись словаря включает, возможно, в опосредованном или уплотненном виде, следующие поля:

- текст основы
- информация об идентификаторе лексемы
- номер технической части речи — набора таблиц
- номер грамматемы для нулевой позиции

- набор номеров масок
- набор номеров строк таблицы начертаний
- различная служебная информация

В реализации структура файла словаря отличается от указанной и является более сложной; см. напр. пункт 3.3.1.

3.3. Некоторые вопросы реализации

3.3.1. Идентификаторы основ и правила распределения основ

Различным основам одной и той же лексемы соответствует один общий идентификатор лексемы. Для целей морфологического анализа достаточно возможности установить по основе идентификатор соответствующей лексемы. Для целей морфологического синтеза и нормализации слов необходима система перекрестных ссылок между основами одной лексемы, например: *люд-ьми* --> *человек*-*, *станк-ами* --> *станок*-*. Другими словами, необходимо обеспечить возможность идентифицировать основы одной лексемы по отдельности. Приписанное записи словаря уникальное число называется идентификатором основы.

Следующий прием обеспечивает взаимное соответствие между идентификаторами лексем и основ. Используется то обстоятельство, что идентификаторы лексем могут выбираться морфологической системой произвольно, при условии, что идентификатор лексемы не меняется при изменении словаря. При внесении новой лексемы в словарь определяется максимальное значение идентификатора основы среди всех, уже имеющихся в словаре. Оно в точности соответствует размеру инвертированного индексного массива словаря, см. пункт 2.6. Обозначим это число через N. Первая вносимая в словарь основа лексемы получает идентификатор N, следующая N+1, и т.д. Кроме того, в запись в словаре вносится номер основы внутри лексемы. Разность идентификатора основы и номера основы внутри лексемы есть число, постоянное для основ одной лексемы, и оно уникально в словаре. Это число объявляется идентификатором лексемы.

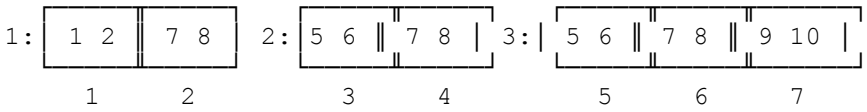
Таким образом, идентификатор лексемы L и идентификатор ее i-й основы S связаны уравнением $L = S – i$. Для хранения идентификатора основы нужно столько же бит, сколько для хранения идентификатора лексемы, однако дополнительная память размером порядка 5 — 6 бит требуется для хранения номера основы. Ниже показано, что дополнительного расхода памяти можно почти всегда избежать. Никакая дополнительная информация для перекрестных ссылок между основами одной лексемы в словаре не хранится.

В пункте 3.2.1 указано, что в словаре хранится информация о нерегулярных явлениях, связанных с лексикой языка. Однако распределение основ лексемы по формам, ими образуемым, часто носит регулярный характер. Например, наблюдается следующая закономерность: у многих существительных одна основа отвечает за формы именительного и винительного падежей единственного числа, а другая — за все остальные формы. Например: *станок*-* vs. *станк-ами*. Другими словами, у многих лексем формальной части речи существительных есть ровно две основы, первая из которых имеет маску «+–...–», а вторая — маску «–+...+». Данное явление в лингвистике носит название чередования. В предлагаемой модели отсутствуют средства выражения данного явления как регулярного, однако такая регулярность автоматически обнаруживается (см. пункт 3.3.4) и используется для оптимизации хранения и доступа к данным словаря.

Понятие набора номеров масок дано в пункте 3.2.8. Основе в словаре соответствует набор номеров масок. Правилом распределения основ называется упорядоченное множество из нескольких наборов номеров масок. Таблицей правил распределения основ называется автоматически формируемый массив из нескольких правил распределения основ. Размер такой таблицы крайне невелик, в нее заносятся только самые частотные правила. Однако при этом оказывается, что подавляющему большинству лексем словаря соответствует правило, имеющееся в таблице. Это означает, что для данной лексемы в таблице есть такое правило, что первая основа лексемы имеет набор номеров масок, равный первому набору в правиле, вторая — второму, последняя — последнему.

Таблица правил хранится в оперативной памяти ЭВМ, занимая около 300 байт. Для большинства лексем в словаре вместонабора номеров масок указывается номер соответствующего набора в таблице правил. По этому номеру восстанавливается не только сам набор, но и все правило. Более того, поскольку номер i основы внутри лексемы соответствует номеру набора масок в правиле, необходимость его хранения в словаре также отпадает. Например, пусть одна основа лексемы имеет

набор масок (5 6), а другая набор (7 8). Пусть таблица правил содержит три правила, состоящие в сумме из семи наборов:



Тогда по номеру набора 4 восстанавливается как набор (7 8) и правило номер 2, так и то, что данная основа вторая среди основ данной лексемы. Последнее необходимо для определения идентификатора лексемы по идентификатору основы.

Более важной, чем экономия места в словаре, функцией механизма правил распределения основ является ускорение работы алгоритмов синтеза и нормализации. Рассмотрим фрагмент алгоритма синтеза одной формы слова по другой его форме, например единственного числа слова *дет-ей*. На рис. 2 даны соответствующие таблицы. При анализе с диска считана основа *дет-*, имеющая маску 40. Необходимо выяснить, от какой основы образуется форма родительного падежа единственного числа лексемы РЕБЕНОК. Если в таблице правил нет соответствующего правила, будет считана с диска первая основа *ребенок-*, она имеет маску 20, запрещающую нужную форму. После этого будет считана основа *ребенк-*, и по ее маске 30 построена форма *ребенк-а*. Если же в таблице есть нужное правило { (20) (30) (40) }, ссылка на него хранится при основе *дет-*, и по нему сразу устанавливается, что необходимо считать с диска вторую основу лексемы.

3.3.2. Атрибуты текста

Рассмотрим процесс морфологического анализа текста. В главе 2 показано, что по тексту может быть найдена в словаре запись, ключ которой является начальной его подцепочкой. Такая запись дает гипотетическую основу первой лексемы текста. Однако вид реального текста может отличаться от того вида, в котором хранятся ключи записей в словаре. Например, словоформа в тексте может начинаться с заглавной буквы. Простое приведение всех букв к одному регистру не дает удовлетворительных результатов. Так, в режиме обнаружения ошибок система должна обнаруживать отклонения в правописании таких слов, как *Москва*, *ВИНИТИ*, *ВНТИЦентр*, *сПб*.

Известную трудность представляет быстрая обработка переноса слов. Как показано в пункте 3.1.1, при переносе слов неясно, был ли в словоформе дефис. Словоформа может быть перенесена по дефису, и этот дефис будет удален из нее процедурой склеивания перенесенных слов. Поэтому в словаре основы хранятся без дефисов.

Стандартным видом буквенной цепочки называется цепочка, полученная из исходной выполнением следующих операций. Большие буквы заменяются соответствующими малыми. Повторные пробелы удаляются. Перенесенные строки склеиваются. Конец каждой строки преобразуется в пробел. Все знаки дефиса удаляются. Русская буква «ё» заменяется на «е». Выполняются некоторые другие аналогичные преобразования.

Информация о произведенных преобразованиях кодируется специальным образом. Полученный короткий код называется массивом атрибутов текста. Большая часть слов в обычном тексте уже представлена в стандартном виде, для таких цепочек массив атрибутов пуст. По массиву атрибутов и цепочке в стандартном виде можно восстановить исходную цепочку букв. Для цепочек с одинаковым стандартным видом возможно сравнение их атрибутов. Например, результатом сравнения атрибутов цепочки *москва* с атрибутами цепочки *Москва* является сообщение «первая буква должна быть заглавной».

Ключи записей словаря хранятся в стандартном виде. В словаре хранятся также их атрибуты. Для большинства основ массив атрибутов пуст; непустой массив атрибутов имеется у основ, содержащих букву «ё», дефис, заглавные буквы.

В выполненной автором реализации пустой массив атрибутов не требует хранения в словаре никакой дополнительной информации, например, бита наличия или отсутствия такого массива. В словаре хранится длина всей записи, соответствующей данной основе, а длины всех ее полей, кроме массива атрибутов, определяются их значениями. Следовательно, длина массива атрибутов определяется как разность между длиной всей записи и суммой длин ее полей.

3.3.3. Работа с набором документов и словарей. Словари высокочастотных слов и стоп-слов

Назначение морфологической подсистемы состоит в том, чтобы избавить использующую ее прикладную систему от работы с буквенным представлением текста. Поскольку задача членения входного текста на словоформы является частью задачи морфологического анализа, на вход системы подается не отдельная словоформа, а весь анализируемый текст. Система может хранить часть текста в своих внутренних буферах, просматривать его на несколько символов вперед, и т.д. Однако входной текст не обязательно должен храниться в памяти ЭВМ или в некотором файле. Для определения входного анализируемого текста достаточно указать системе имя процедуры, поставляющей по запросу системы символы входного текста. Она называется процедурой ввода текста.

Аналогично, синтез текста также не является пословной операцией, поскольку синтез словоформы не всегда возможен вне контекста. Так, морфологическая подсистема должна производить выбор правильного варианта синтеза таких слов, как *о*/*об*/*обо*, *в*/*во*, не различимых на синтаксическом уровне.

Документом называется объект, являющийся источником или приемником текстовой информации, с указанием режима и параметров ее обработки. Документы делятся на анализируемые и синтезируемые. Необходимым атрибутом анализируемого документа является процедура ввода текста, однако кроме нее требуется указание целого ряда параметров его обработки. Основной операцией над анализируемым документом является получение идентификатора и характеристик очередной лексемы. Основной операцией над синтезируемым документом является вывод в него словоформы очередной лексемы на основании идентификатора лексемы и набора морфологических характеристик. Кроме того, реализовано большое число вспомогательных операций.

При практической работе системы часто возникает необходимость работы с несколькими документами одновременно. Такая возможность реализуется путем создания нескольких независимых объектов-документов и поочередного выполнения операций над ними.

Со словарем также связан некоторый объект. Возможно существование нескольких таких объектов, работающих с разными файлами словарей. Аналогично, возможно существование нескольких объектов, представляющих разные наборы морфологических таблиц. Каждый словарь связан с определенным объектом, представляющем таблицы, ссылки на которые хранятся при основах лексем в данном словаре. Однако обычно несколько словарей связаны с одним и тем же объектом, представляющем таблицы. На рис. 5 дан пример типичной конфигурации прикладной системы. Двойной рамкой выделены массивы информации, хранящиеся на внешних носителях.

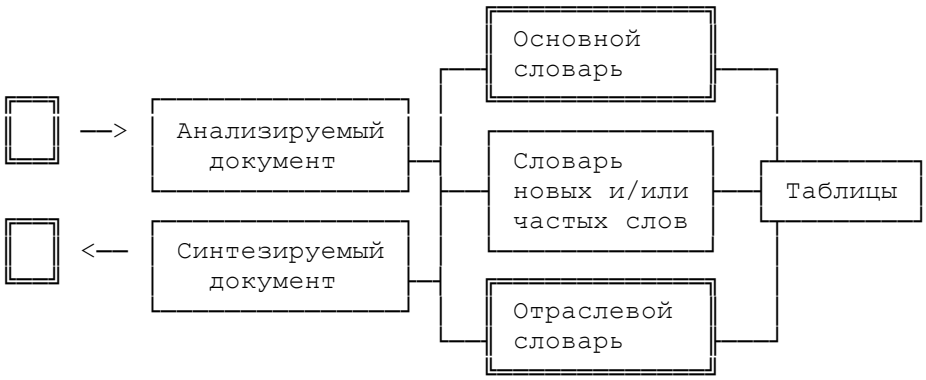


Рис. 5. Типичная конфигурация прикладной системы

Поскольку словарь новых слов расположен в оперативной памяти ЭВМ и имеет небольшой объем, внесение новых слов в него не представляет затруднений. На него не распространяются требования, изложенные в главе 2. Его структура может не совпадать со структурой основного словаря, рассмотренной в главе 2. Поэтому новые слова не вносятся в основной словарь, это было бы длительной операцией. Когда словарь новых слов становится слишком большим, пользователем выполняется процедура слияния словарей, рассмотренная в пункте 2.3.3.

Каждый объект-словарь в реализации имеет собственный кеш-буфер, размер которого задается прикладной программой. Любой словарь небольшого размера, например, отраслевой, может быть

целиком или почти целиком размещен в оперативной памяти ЭВМ путем указания достаточного размера кеш-буфера.

Изложенный механизм используется также для организации обработки высокочастотных слов без обращения к дисковому словарю. Словарь высокочастотных слов содержит около 300 коротких слов — союзы, предлоги и т.п. Предполагается, что если слово найдено в данном словаре, большой дисковый словарь не дает новых по сравнению с найденными вариантов анализа данной цепочки. Условия, при которых такое предположение верно, рассмотрены ниже.

Система позволяет прикладной программе определять порядок просмотра словарей в процессе анализа словоформы, активизировать и деактивизировать словари, а также прерывать цикл анализа словоформы. Словарь высокочастотных слов используется прикладной программой следующим образом. Ему назначается размер кеш-буфера, достаточный для размещения всего словаря в оперативной памяти ЭВМ. В цепочке словарей, ассоциированных с документом, данный словарь располагается перед большим дисковым словарем. В процессе анализа очередной буквенной цепочки, в случае, если хотя бы один вариант анализа найден в данном словаре, большой дисковый словарь деактивируется до конца цикла анализа данной цепочки. Таким образом, для слов, занесенных в словарь высокочастотных слов, обращения к диску не производятся.

Необходимо заметить, что исключениями из приведенного правила являются такие варианты анализа, при которых из анализируемого текста выделяются словоформы, заканчивающиеся на дефис, расположенный в конце строки, что специально индицируется системой путем установки соответствующего флага в атрибутах проанализированной словоформы. Таким образом, случаи наличия высокочастотного слова в позиции, которая может быть позицией переноса, не допускают анализа по словарю высокочастотных слов. В противном случае в тексте

произведены из-
мерения параметров

вместо слова *измерение* был бы ошибочно обнаружен предлог *из*. Поэтому при обнаружении подобного варианта, если большой словарь еще не деактивирован, то деактивируется, напротив, словарь высокочастотных слов, а сам вариант игнорируется. Если же подобный вариант не является первым вариантом, найденным по словарю высокочастотных слов, обработка текста продолжается по изложенному выше алгоритму.

Для составления словаря высокочастотных слов используются следующие автоматические процедуры: процедура частотного анализа текста, процедура замыкания словника и процедура построения подсловаря. Процедура частотного анализа текста составляет по входному массиву текстов упорядоченный по частоте встречаемости в тексте список лексем. Из данного списка выделяются наиболее частотные лексемы, что дает словник словаря высокочастотных слов; при необходимости в данный словник лингвистом могут быть внесены изменения. Затем словник пополняется рассмотренной ниже процедурой замыкания. С помощью процедуры построения подсловаря из большого словаря извлекается морфологическая информация, соответствующая каждой лексеме словника, и формируется отдельный словарь, имеющий рассмотренную в главе 2 внутреннюю структуру.

В целях обеспечения полноты анализа необходимо, чтобы для любой буквенной цепочки выполнялось одно из двух условий: либо словарь высокочастотных слов не дает вариантов ее анализа (и в этом случае просматривается большой дисковый словарь), либо он дает все варианты ее анализа, которые дает большой словарь. Процедура замыкания словаря обеспечивает выполнение этого условия. Заметим, что, по построению словаря высокочастотных слов, его словник является подмножеством словника большого словаря.

Подмножество M множества лексем словаря (или набора из нескольких словарей) D называется замкнутым относительно D, если для любого контекста K множество лексем, словоформы которых могут быть выделены в начале K по словарю D, либо является подмножеством M, либо не пересекается с M. Заметим, что, например, лексема *стол* может быть выделена в начале контекста «*столы накрыты*», но не в начале контекстов «*столпотворение*» или «*на столе*».

Таким образом, M не является замкнутым относительно D, если существует пара словоформ — словоформа m некоторой лексемы из множества M, и словоформа d некоторой лексемы в множестве D, но не в M, — таких, что выполняется одно из двух условий: либо $m=d$, либо одна из них, например m, является начальной частью другой, например d, причем за этой начальной частью в d следует символ, который может рассматриваться как разделитель слов в тексте. Например, следующие лексемы образуют совпадающие словоформы: *горе* и *гореть* (*горю*), *поле* и *полить* (*полей*). Следующие лексемы

образуют вложенные словоформы: *диван* и *диван-кровать*, *во* и *во что бы то ни стало*. Следующие лексемы *не* образуют вложенных словоформ: *стол* и *столпотворение*.

Замыканием множества *M* в *D* называется минимальное замкнутое подмножество *D*, содержащее *M*. Замыканием также называется процесс построения такого множества. Алгоритм работы процедуры замыкания словника состоит в генерации всех словоформ каждой лексемы словника и анализе полученных буквенных цепочек по словарю *D*. В случае, если такой анализ возможен, соответствующая лексема переносится из *D* в словник и добавляется к множеству еще не проверенных слов словника. Процесс заканчивается, когда все слова словника проверены. Для обнаружения более длинных словоформ применяется некоторая модификация изложенного в пункте 4.5 алгоритма исправления неодиночных опечаток, а именно, исправления произвольного числа пропусков букв в конце слова, при условии, что первый добавленный символ является разделителем слов. Таким образом по словоформе *во* могут быть построены словоформы *во что бы то ни стало* и *во-первых* (но не *вопрос*).

Фактически быстродействие подобной процедуры достаточно велико, поскольку мало количество подлежащих проверке вариантов. Поскольку в словарь высокочастотных слов включаются в основном неизменяемые слова, количество их словоформ невелико. Что же касается алгоритма построения более длинной словоформы, то он почти всегда заканчивает работу сразу же после неудачной попытки добавить к слову хотя бы один символ, который обязан быть разделителем слов, то есть не буквой. По изложенным причинам увеличение словника в процессе его замыкания как правило является небольшим. Необходимо, однако, отметить, что в некоторых случаях приходится удалять из словника отдельные слова, приводящие к значительному росту его объема при замыкании.

При добавлении слов к основному словарю необходимо корректировать словарь, замкнутый относительно него. Такая коррекция производится особым алгоритмом. А именно, сначала с помощью изложенной выше процедуры производится попытка замыкания множества вновь внесенных слов относительно существующего словаря высокочастотных слов. Ввиду малого объема обоих множеств слов быстродействие такой процедуры достаточно велико. Если в данных двух множествах найдена хотя бы одна пара лексем, имеющих совпадающие или вложенные словоформы, то процесс замыкания словаря высокочастотных слов производится по общему алгоритму. Если такой пары не существует, что и наблюдается чаще всего на практике, никаких дополнительных действий более не предпринимается.

Словарем стоп-слов называется словарь, составленный из слов, обработка которых должна производиться особым образом. Чаще всего особая обработка заключается в блокировании выдачи результатов анализа таких слов в выходной массив. Например, при составлении поисковых образов документов, хранимых в информационно-поисковой системе по тематике «программирование», должны игнорироваться такие неинформативные в данном случае слова, как *программа*, *файл*, *бит*. Существует три способа организации словаря стоп-слов. Первый способ заключается в применении рассмотренного выше механизма, исключая механизм замыкания словаря. Заметим, что словарь высокочастотных слов в нашем определении является частным случаем словаря стоп-слов. Однако исключение механизма замыкания словаря может приводить к потере точности анализа. Второй способ заключается в организации массива идентификаторов лексем стоп-слов, например, в виде хеш-таблицы. Недостатком данного метода является дополнительный расход оперативной памяти ЭВМ в адресном пространстве прикладной программы. Третий способ заключается в формировании замкнутого словаря по словнику словаря стоп-слов и хранении массива идентификаторов добавленных при замыкании лексем. Последний массив имеет небольшой объем и просматривается только при обнаружении в тексте слов, принадлежащих стоп-словарю.

3.3.4. Подготовка лингвистического обеспечения

Для подготовки лингвистического обеспечения были разработаны и реализованы программные средства компиляции и отладки исходных текстов массивов лингвистической информации. Исходные тексты представляют собой структурированные текстовые файлы, содержащие описание таблиц и словарей.

Таблицы помещаются каждая в отдельный файл. Сроки и столбцы идентифицируются числами с десятичной дробью, что позволяет в процессе сопровождения системы вставлять новые строки и столбцы на соответствующие места без изменения нумерации уже имеющихся, поскольку такое изменение привело бы к необходимости исправления номеров во всех остальных массивов.

Компилятор таблиц преобразует дробные номера строк в порядковые номера, выполняет синтаксический и формально-логический контроль текста таблиц, обеспечивает обнаружение

одинаковых строк таблиц. Текст морфов в списках и таблицах морфов приводится к стандартному виду (пункт 3.3.2). Атрибуты морфов помещаются в отдельные массивы. Практически единственное назначение данного шага — обработка русской буквы «ё» в окончаниях слов типа *конь-е* (см. пункт 3.3.2).

Исходные тексты словарей существуют в трех форматах: формат с полностью заданной информацией, формат с указанием грамматических классов, и формат словаря А.А.Зализняка [69].

Файлы словаря в формате с полностью заданной информацией содержат отдельные лексемы с нерегулярным изменением. При каждой лексеме указан перечень ее основ и заполнены все поля словарной информации при каждой из них (см. пункт 3.2.8). Имеются средства определения небольших групп одинаково изменяющихся слов. Общий размер таких файлов, формируемых вручную, невелик. Кроме того, файлы в таком формате порождаются автоматически программой внесения незнакомых слов в словарь.

Файлы словаря с указанием грамматических классов формируются автоматически процедурой обработки словаря [69]. Они содержат перечень слов в исходной форме, каждому из которых приписано имя грамматического класса. Допускаются одинаковые имена классов для слов, детали изменения которых могут быть установлены по их окончаниям. Например, словам *делать* и *здороваться* в словаре А.А.Зализняка [69] приписано одно и то же имя класса «нсв 1а».

Для обработки словаря в формате с указанием классов составляется файл описания классов. Каждому классу сопоставлен набор основ и полей для занесения в словарь морфологической системы. Специальным образом кодируется преобразование исходной формы слова в каждую из основ. Выбор между различными вариантами изменения внутри одного класса производится по шаблонам на конечную часть слова. Допускается объявление некоторых имен классов синонимами. Имеется набор макросредств для упрощения определения шаблонов.

Таким образом, в файле описания таблиц на специальном языке записаны инструкции вида «для классов нсв 1 а или b: если исходная форма кончается на *-ся*, то удалить пять последних букв и полученную основу записать в словарь, приписав такую-то часть речи, такие-то номера строк и масок; если исходная форма кончается на *-ть*, то удалить три последних буквы и приписать другие номера масок...» и т.д. Файлы описания классов, составленные для классов словаря [69], составляют большую часть разработанного при участии автора лингвистического обеспечения системы.

В выполненной автором реализации программного обеспечения системы массивы правил распределения основ (см. пункт 3.3.1) формируются при обработке файлов словаря в формате с указанием классов. Такой файл упорядочивается по классам. Если в процессе обработки файла число слов с одним и тем же классом превышает некоторое пороговое значение, формируется правило, соответствующее данному классу.

В качестве файла в формате словаря [69] был использован сам этот словарь. Первоначально при участии автора была предпринята попытка программной поэлементной обработки индексов, употребляющихся в [69]. Однако по мере роста числа подлежащих обработке индексов структура такой программы стала неоправданно сложной, а ее понимание и сопровождение лингвистом, не имеющим навыков программирования, стали затрудненными. Вместо этого программным образом было выполнено преобразование словаря А.А.Зализняка [69] в формат словаря с указанием классов. В качестве имен классов используются имеющиеся в [69] индексы, например, «п 1*а». Преобразование заключалось в приведении информации, приписанной словам в [69], к более единообразному виду. Например, производились следующие преобразования:

```
м 1а (растение)       ——>  м 1а
нсв 1а (°) 4 (-ить)   ——>  нсв 1а (°)
н; союз               ——>  н
                           союз
п 1а; мо              ——>  п 1а
                           мо <п 1а>
```

Ввиду не совсем формального синтаксиса словаря [69], среди ста тысяч его строк встречаются такие, формальная обработка которых представляет значительную трудность. Так, например, символ «;» не всегда употребляется в качестве разделителя вариантов, выражения в круглых скобках не всегда бывают семантическими замечаниями, и т.д.

Кроме собственно словаря [69], файлы в таком же формате составляются для отраслевых словарей и словарей новых слов. При формировании основного словаря системы объединяется информация, содержащаяся в файлах разных форматов.

3.4. Алгоритмы морфологического анализа, синтеза и других операций над текстом

3.4.1. Точный и полный морфологический анализ словоформ

На вход алгоритма подается буквенная цепочка, начинающаяся анализируемой словоформой. На выходе формируется идентификатор лексемы, набор морфологических характеристик словоформы и разбор словоформы на морфы. Результат не является однозначным.

Алгоритм анализа построен по следующему принципу. Сначала словоформа разбивается на морфы. При разбиении учитывается только информация из списков морфов, считая словарь как список морфов нулевой позиции. Затем устанавливаются соответствующие морфам морфемы. Затем проверяется их совместимость. Два первых шага не являются однозначными. Выполняется полный перебор всех возможных вариантов. Как показано в главе 2, полный перебор всех вариантов требует только одного обращения к дисковой памяти.

Рассмотрим алгоритм анализа подробнее. Наборами здесь называются наборы соответствующих чисел, содержащие по одному числу на каждую позицию, начиная с нулевой или с первой.

В практической реализации как сам алгоритм, так и структуры используемых данных оптимизированы. То же касается всех приводимых в работе алгоритмов.

Шаг 1. В словаре ищутся все гипотетические основы данной цепочки, то есть записи, ключи которых — основы — являются ее начальными подцепочками. Как показано в главе 2, поиск требует одного обращения к диску. Алгоритм поиска в словаре выдает первой самую длинную из таких цепочек, т.е. как раз ту, для которой вероятность оказаться правильной максимальна. Дальнейшие шаги выполняются отдельно для каждой гипотетической основы в отдельности.

Шаг 2. По информации при основе определяется техническая часть речи и фиксируется набор таблиц.

Шаг 3. Построение набора морфов. Для этого в списке морфов первой позиции ищутся морфы, вкладывающийся слева в часть анализируемой цепочки, расположенную после основы. Первым анализируется наиболее длинный из таких морфов. Затем, для каждого из найденных вариантов по оставшейся части анализируемой цепочки производится поиск в списке морфов следующей позиции, и т.д.

Одновременно строится набор номеров строк таблиц начертаний. Значение элемента этого набора для данной позиции полагается равным значению, указанному в списке морфов при морфе предыдущей позиции, который к этому моменту уже найден. Если это значение есть «*», используется значение при морфе предыдущей позиции, как это указано в пункте 3.2.5. Однако, если для данной позиции вместо номера строки таблицы начертаний был обнаружен номер морфемы (см. пункт 3.2.5), позиция считается фиктивной, и поиск в списке морфов данной позиции не производится.

Дальнейшие шаги выполняются отдельно для каждого построенного набора морфов. Используемая на данном шаге информация: списки морфов.

Шаг 4. Проверка. Проверяется, может ли символ, расположенный после разобранной части цепочки, служить разделителем словоформ. Если нет, вариант отвергается. Например: *вор-ов-?ство* vs. *воровств-о*.

Шаг 5. Преобразование набора морфов в набор морфем. Для фиктивных позиций номера морфем установлены на шаге 3. По каждой нефиктивной позиции производится поиск найденного на шаге 3 морфа в найденной на шаге 3 строке таблицы начертаний данной позиции. Номер столбца, в котором в данной строке найден морф, является искомым номером морфемы. Поскольку такая операция неоднозначна, дальнейшие шаги выполняются отдельно для каждого построенного набора морфем. Используемая на данном шаге информация: таблицы морфов.

Шаг 6. Преобразование набора морфем в набор значений масок. Релевантными являются только значения масок для нефиктивных позиций. Обозначим набор номеров морфем через M. Для каждой фиксированной позиции i определяется номер m маски, указанный для данной позиции в элементе номер $M(i\text{-}1)$ списка грамматем позиции i-1. Если этот номер маски не равен «*», определяется элемент номер $M(i)$ маски номер m для позиции i. Если номер маски равен «*» или найденное значение

элемента маски равно «*», процесс повторяется, с использованием списка грамматем позиции *i*-2, и т.д. Если значение маски не определено после просмотра списка грамматем нулевой позиции, используется значение номера маски, данное в словаре. Используемая информация: таблицы масок, списки грамматем.

Шаг 7. Проверка. Если хотя бы одно из значений масок равно «–», вариант отвергается. Производятся некоторые другие проверки (см. пункт 3.4.3).

Шаг 8. Формирование результата. В качестве результата анализа для каждого варианта, не отвергнутого на шаге 7, выдается идентификатор лексемы и конкатенация грамматем, соответствующих значениям из набора морфем, для всех позиций. Идентификатор лексемы определяется так, как указано в пункте 3.3.1. В случае, когда одной морфеме приписано несколько грамматем (см. пункт 3.2.7), каждая из них порождает отдельный вариант. Если среди значений набора масок имеются признаки затрудненности, факультативности, вариативности, соответствующая особенность словоформы отмечается. Морфный разбор словоформы строится по набору номеров морфов. При выдаче морфного разбора словоформы фиктивные позиции игнорируются (см. пункт 3.5.2).

3.4.2. Точный и полный морфологический синтез словоформ

На вход алгоритма синтеза подается идентификатор лексемы и набор граммем (см. пункт 3.1.3). Может также указываться информация об атрибутах текста синтезируемой цепочки, например, «первая буква заглавная». На выходе алгоритма формируется буквенная цепочка, представляющая словоформу данной лексемы, обладающую указанными характеристиками, а также полный перечень ее характеристик. Результат не является однозначным. Так, при указании пустого множества граммем формируется вся парадигма лексемы.

Алгоритм синтеза построен по следующему принципу. По спискам грамматем составляются наборы морфем, соответствующие требуемым граммемам. По таблицам морфов определяются морфы, составляющие словоформу. Понятие набора морфем дано в пункте 3.4.1. Рассмотрим алгоритм синтеза подробнее.

Шаг 1. Из словаря извлекается первая основа лексемы, определяется техническая часть речи и фиксируется набор таблиц.

Шаг 2. Построение набора морфем. В каждом списке грамматем ищутся такие, которые не противоречат входному набору граммем. Грамматема противоречит граммеме, если содержит граммему с тем же признаком, но с другим значением. Например, грамматема «число = единственное, падеж = творительный» противоречит граммеме «падеж = дательный», но не противоречит граммеме «время = настоящее».

Из найденных значений всеми способами составляются наборы морфем. Дальнейшие шаги выполняются отдельно для каждого построенного набора морфем. Используемая на данном шаге информация: списки грамматем.

Шаг 3. Определение словарной информации о масках для всех основ. В случае, если при считанной с диска основе указана ссылка на правило распределения основ (см. пункт 3.3.1), дополнительного обращения к дисковому словарю не требуется. В противном случае с диска считывается информация о всех основах лексемы. Используемая информация: списки правил распределения основ.

Шаг 4. Построение набора значений масок и отсев вариантов. Набор масок строится так, как указано в пункте 3.4.1, шаг 6 алгоритма. Варианты, не прошедшие проверку шага 7 того же алгоритма, отвергаются.

Шаг 5. Построение набора морфов. С диска считывается информация о номерах строк таблиц морфов для выбранной основы. Набор морфов строится слева направо. Для очередной позиции определяется номер строки таблицы морфов так, как это указано в пункте 3.4.1, шаг 3 алгоритма. Позиции, помечаемые при этом как фиктивные, обязаны уже быть помечены как фиктивные на шаге 4. В набор морфов заносится морф, стоящий в найденной строке таблицы морфов в столбце, определяемом набором морфем, построенным на шаге 2.

Шаг 6. Формирование результата. В качестве результата синтеза для каждого варианта выдается конкатенация найденных морфов по всем нефиктивным позициям. Набор характеристик строится так, как указано в пункте 3.4.1, шаг 8 алгоритма.

3.4.3. Анализ реального текста

Поскольку задача членения текста на словоформы также входит в число задач морфологического анализа, а также в связи с необходимостью выполнения над текстом определенных преобразований, задача анализа текста не сводится к задаче анализа отдельной словоформы, рассмотренной в пункте 3.4.1.

Следующие особенности следует учесть при обработке реального текста, состоящего из многих слов (ср. пункт 3.3.2): наличие в тексте заглавных букв и других символов, требующих преобразования, таких, как русская буква «ё»; наличие переносов слов; необходимость продвижения анализатора вдоль текста после анализа очередной словоформы, и ряд других особенностей. Кроме того, системой выполняется внутренняя буферизация текста, обеспечивающая, например, ввод и правильный анализ слов типа *во что бы то ни стало*.

Результатом анализа является также информация, связанная с атрибутами текста (см. пункт 3.3.2). Например, сообщение «первая буква должна быть заглавной» вероятнее всего сигнализирует об ошибке типа ошибки в слове **москва*, однако сообщение «первая буква заглавная», дважды выдаваемое при анализе фразы «*Пришла Света*.», может быть проинтерпретировано прикладной программой, например синтаксическим анализатором, либо как ошибочное, либо как нормальное. Кроме того, последнее сообщение содержательно является результатом анализа текста, представляя собой синтаксически релевантную информацию.

До выполнения собственно анализа по алгоритму пункта 3.4.1 текст во внутреннем буфере системы приводится к стандартному виду. После анализа словоформы атрибуты текста сравниваются с атрибутами, хранящимися в словаре, и по результатам сравнения генерируются различные сообщения. В зависимости от режима работы системы (см. пункт 3.4.6), некоторые из них считаются ошибочными. Варианты анализа, породившие такое сообщение, отвергаются на шаге 7 алгоритма анализа из пункта 3.4.1. Другие сообщения считаются нормальными и выдаются в качестве результатов анализа.

Необходимо уточнить также шаг 4 того же алгоритма анализа. Поскольку в стандартном виде текста отсутствует дефис, проверка того, может ли данный символ ограничивать слово, производится не только по виду самого символа, но и по его атрибутам, в частности, по наличию перед ним дефиса или переноса. Таким образом обеспечивается обнаружение словоформы *женщина* в цепочке *женщина-депутат* даже в случае наличия переноса по дефису. Однако, как видно из алгоритма анализа, если бы слово *женщина-депутат* имелось в словаре (ср. *вице-президент*), оно было бы обнаружено в тексте.

В связи с этим возрастает значение проблемы неоднозначности разбиения текста на слова. Так, тексты *во что бы то ни стало* и

о пере-
избрании

неоднозначно разбиваются на слова. Системой выбирается на каждом шаге наиболее длинная анализируемая цепочка, хотя имеются средства, позволяющие прикладной программе самостоятельно управлять продвижением анализатора по тексту и даже возвращаться на несколько слов назад.

3.4.4. Анализ сложных слов и текста без пробелов

Алгоритм анализа из пункта 3.4.1 с небольшой модификацией используется для анализа некоторых сложных слов и композитных образований даже в том случае, когда априорно трудно указать место или места членения композитной словоформы. Например: *молокозавод, газоводопыленепроницаемый, тридцатидвухразрядный*. Для распознавания сложных слов в парадигмы лексем включается специальный соединительный морф, например, соединительная гласная или пустой морф. Морфеме, соответствующей такому морфу, приписывается граммема композитности. Тогда шаг 4 алгоритма анализа дополняется возможностью выделения слова, ограниченного соединительным морфом. Алгоритм синтеза из пункта 3.4.2 практически без модификации может синтезировать такие формы слов с граммемой композитности, например: *газ-о-, пыл-е-, тридцат-и-*.

Кроме того, в словарь включаются квазилексемы или квазиосновы обычных лексем, являющиеся частями композитных образований, например: (*зелено*)-*окий*, (*легко*)-*крылый*, *электро-*, *вице-*. Такие квазиосновы имеют специальный словарный признак, запрещающий их употребление вне композитных образований.

Мы трактуем распознаваемые композитные образования как свободные сочетания. Только на этапе дальнейшего содержательного анализа текста может быть установлена сочетаемость частей

композитного образования. Ср.: *ветропросвист экспрессов, крылолет буеров* (И.Северянин); *электропочта*. Сложные же слова, не являющиеся свободными сочетаниями, включаются в словарь как обычные слова, например: *четвероногий*.

Необходимость массового анализа сложных слов возникает при анализе текстов по химии или текстов на естественном языке с развитой системой словосложения, например немецком. Аналогичный аппарат может быть использован при анализе текста, не разделенного на слова пробелами, например, текста на японском языке.

3.4.5. Нормализация слов и синтез первой формы

Задача нормализации слов идентификаторами лексем сводится к задаче морфологического анализа. Из алгоритма анализа исключаются только заключительные операции, связанные с выдачей характеристик словоформы. В этом случае из списков грамматем исключаются и не хранятся в оперативной памяти ЭВМ сами грамматемы, что приводит к экономии оперативной памяти. Номера масок в списках грамматем оставляются.

Задача нормализации слов словарными формами сводится к задаче нормализации слов идентификаторами лексем и последующего синтеза первой формы данной лексемы.

Однако выбор требуемого набора характеристик представляет затрудняется тем, что различные слова имеют в словарной форме различные наборы характеристик. Например: *ножницы, некого, щец, рад*. Для облегчения синтеза первой формы на лингвистическое и программное обеспечение системы накладывается следующее требование: синтез всех форм каждой лексемы должен производиться так, чтобы на выходе алгоритма синтеза формы генерировались в «естественном» порядке. Например: *сильный, сильного, сильному, ..., сильнейший, сильнейшего, сильнейшему...*

Пусть зафиксирован такой порядок перебора морфем в алгоритме синтеза из пункта 3.4.2, при котором наборы морфем синтезируются в лексикографическом порядке. Опыт составления таблиц показывает, что при «естественном» перечислении парадигмы в каждой позиции указанное условие упорядоченности выдачи при синтезе в большинстве случаев выполняется автоматически.

Для слов с ущербной парадигмой словарной формой обычно считается первая возможная форма при перечислении его форм в таком «естественном» порядке, ср.: *некого*. Таким образом, синтез первой формы сводится к синтезу всех форм, то есть к синтезу форм с указанием пустого множества требуемых граммем. Однако процесс синтеза останавливается после получения первого варианта.

Алгоритм нормализации словарными формами часто требует обращений к дисковой памяти на одно меньше, чем отдельно алгоритм анализа и алгоритм синтеза, поскольку необходимая на шаге 1 алгоритма синтеза из пункта 3.4.2 информация об одной из основ слова уже находится в оперативной памяти ЭВМ после анализа входной цепочки.

Точной нормализацией по семантике называется замена имеющегося в словаре слова на идентификатор группы близких по значению слов. Например: *красный, желтый, ..., синий* --> группа ЦВЕТА. Нормализацией представителем называется замена такого слова на близкое по значению слово-представитель группы квазисинонимов. Например: *металл, железо, чугун, сталь* --> *металл*. Понятие близости по значению зависит от конкретного приложения. Поэтому такая нормализация производится с помощью определяемого прикладной программой словаря синонимов.

Для этого составляется массив перекодировки идентификаторов лексем в идентификаторы групп или представителей групп. Идентификаторами лексем в системе являются числа в диапазоне от нуля до числа, приблизительно равного количеству основ в словаре системы (см. пункт 3.3.1). Поэтому размер массива перекодировки для крупного словаря синонимов не превышает 300 килобайт.

При первоначальной подготовке массива перекодировки текст словаря синонимов подвергается морфологическому анализу. Принимаются во внимание только варианты, являющиеся словарными формами. Например, из вариантов анализа цепочки *стекло* как СТЕЧЬ и СТЕКЛО автоматически выбирается второй. Из оставшихся вариантов выбираются такие, для которых все слова в группе синонимов относятся к одной части речи. Например, для группы «*плита, печь*» вариант анализа слова *печь* как глагола отбрасывается. Немногочисленные оставшиеся после этого случаи неоднозначности анализа разрешаются вручную. В элемент массива, соответствующий полученному идентификатору очередного слова словаря, заносится номер группы синонимов либо идентификатор представителя группы.

Затем, в процессе эксплуатации системы, каждый идентификатор, полученный при анализе, заменяется на соответствующий элемент массива, что требует одного лишнего обращения к дисковой памяти. Возможна также замена идентификаторов непосредственно в морфологическом словаре, в этом

случае лишнее обращение к дисковой памяти не требуется. При необходимости неоднозначной нормализации или в случае небольшого словаря синонимов вместо массива перекодировки используются соответствующие структуры данных.

3.4.6. Обнаружение, объяснение и исправление ошибок и интерпретация ошибочных слов

Понятие ошибки и постановка задач обнаружения, исправления и объяснения ошибки даны в пункте 3.1.1. Под интерпретацией ошибочно написанного слова понимается формирование результатов его морфологического анализа, то есть установление идентификатора лексемы и набора морфологических характеристик, выраженных данной неправильной формой. В частности, интерпретация ошибки обеспечивает «понимание» окказионально образованных словоформ в случаях, когда правильная форма не существуетили выражается аналитически. Например: **уйденный*, **радый*, **лучшайший*.

Под частичным исправлением ошибки в слове, не допускающем исправление ошибки в собственном смысле, понимается нахождение неошибочной словоформы, близкой к исходной по грамматическим характеристикам. В качестве меры близости может быть взято, например, расхождение по минимально возможному числу граммем. Примеры даны на рис. 6. Варианты частичного исправления ошибок даны в скобках. В графе «Верно» даны истинные варианты исправления.

Словоформа	Интерпретация	Объяснение	Исправление	Верно
читаемыйся	ЧИТАТЬ, имен. страд.возвр.	Страд. несовместимо с возвр.	невозможно (*читаемый*, *читающийся*)	невозможно
ножницей	НОЖНИЦЫ, единств.твор.	Лексема не имеет единств. твор.	невозможно (*ножницами*)	*ножницами*
лучшайший	ЛУЧШИЙ превосх.имен.	Лексема не имеет превосх.	невозможно (*лучший*)	*самый лучший*
человеками	ЧЕЛОВЕК, множ.твор.	неверный выбор основы, д.б. *люд-*	*людьми*	
ста=оками	СТАНОК, множ.твор.	неверный выбор основы, д.б. *станк-*	*станками*	
читаущийсь	ЧИТАТЬ, имен. действ.возвр.	После *-а-* д.б.*-ущ-* После *-ий-* д.б.*-ся*	*читающийся*	
чмтающийся читающмйся	невозможно до исправления	Замена *и* на *м*	*читающийся*	

Рис. 6. Интерпретация, объяснение и исправление ошибок

Как показано на рис. 6, ошибки делятся на две резко различные по своим свойствам группы: те, исправление которых происходит на основе их интерпретации, и те, интерпретация которых происходит на основе их исправления. Ошибки первого рода называются грамматическими ошибками, ошибки второго рода называются опечатками.

Опечатки не связаны с грамматическими явлениями. Они часто затрагивают небольшой участок слова. Объяснение таких ошибок не представляет интереса для конечного пользователя, поскольку он знает правильную форму слова. Пользователь способен исправить опечатку самостоятельно. Например, сообщение системы «слова *хоччет* нет в словаре» является адекватной реакцией на данную ошибку.

В отличие от опечаток, грамматические ошибки тесно связаны с несоблюдением в тексте законов грамматики. Такие ошибки могут целиком изменить буквенный состав слова, например

**человеками* vs. *людьми*. Конечный пользователь системы, возможно, не способен самостоятельно исправить подобную ошибку. Поэтому для него представляет интерес как ее автоматическое исправление, так и объяснение. Например, сообщение системы «слова *хочут* нет в словаре» является явно недостаточно информативным для данного пользователя.

Исправлению опечаток посвящена глава 4 настоящей работы; здесь рассматривается только обработка грамматических ошибок. Этот тип ошибок является более редким, чем опечатки. Из пользователей компьютеров такие ошибки чаще совершают лица, плохо владеющие русским языком. Объяснение грамматических ошибок необходимо как в системах, предназначенных для лиц, для которых русский язык не является родным, так и в обучающих системах. Обработка грамматических ошибок требует привлечения большего объема лингвистической информации, чем обработка опечаток.

Предлагаемая система объясняет и исправляет следующие типы грамматических ошибок, приведенные на рис. 6:

- Несовместимые морфемы (**чита-ем-ый-ся*);
- Недопустимая морфема (**ножниц-ей*, **рад-ый*);
- Неверный выбор основы (**станок-ами*, **человек-ами*);
- Неверный выбор морфа (**красн-ий*, *солдат-ов*);

Неверным выбором морфа называется употребление для выражения каких-либо характеристик словоформы морфа, существующего в языке и действительно выражающего данные характеристики, но употребляющегося с лексемами других грамматических классов.

Алгоритм анализа, рассмотренный в пункте 3.4.1, построен так, что разбор словоформы на морфы на шаге 3 производится только на основании просмотра списков морфов. Никакая дополнительная информация не используется. Грамматическими ошибками считаются такие ошибочные словоформы, для которых возможно построение набора морфов.

Для интерпретации грамматически ошибочного слова с ошибкой типа «несовместимые морфемы» или «недопустимая морфема» на шаге 7 алгоритма варианты, не прошедшие проверку, не отвергаются. Первое из указанных сообщений выдается, если значение «–» в маске обусловлено маской, наложенной на шаге 6 граммatемой ненулевой позиции; второе — соответственно граммatемой нулевой позиции либо маской, номер которой указан в словарной информации. В последнем случае производится попытка синтеза словоформы с установленными при интерпретации характеристиками. Если синтез такой формы возможен, выдается сообщение «неверный выбор основы».

Для интерпретации слова с ошибкой типа «неверный выбор морфа» на шаге 5 в целях построения набора морфем просматривается не только выбранная строка таблицы морфов, а вся таблица. При упрощенной схеме исправления такой ошибки устанавливается морф, стоящий в установленном столбце таблицы в строке, определяемой набором номеров строк, построенном на шаге 3.

Однако наиболее надежным способом исправления любой проинтерпретированной ошибки является синтез словоформы с установленным набором характеристик. При исправлении ошибок типа «несовместимые или недопустимые морфемы» синтез нужной формы обычно невозможен. В этом случае производится частичное исправление ошибки. При этом в первую очередь варьируются граммемы, принадлежащие к грамматемам запрещенных или несовместимых морфем.

Если в слове не удается обнаружить одиночную грамматическую ошибку, производится попытка обнаружения некоторого сочетания однотипных или неоднотипных ошибок.

При разбиении слов на технические части речи учитывается, что интерпретация и исправление ошибок действуют только внутри одной части речи. Так, морфы одной части речи не могут и не должны быть распознаны при основе другой части речи. Данное условие делает нежелательным совмещение в пределах одной технической части речи лексем резко различных грамматических классов, например, существительных и числительных, даже если они имеют одинаковое число позиций.

Задача обработки грамматических ошибок решается без привлечения дополнительной лингвистической информации и без существенного изменения алгоритмов анализа и синтеза. Алгоритм обработки таких ошибок, в отличие от обработки ошибок типа опечатки, практически не требует дополнительных обращений к дисковой памяти и по быстродействию близок к общему алгоритму анализа.

Введение более развитых алгоритмов исправления ошибок дало бы лучшие результаты, но неоправданно усложнило программную часть системы и увеличило бы объем ее лингвистического обеспечения. Применение более развитых алгоритмов возможно на уровне прикладной программы.

3.4.7. Реализация обработки ошибок

Интерпретация и исправление ошибки в слове может быть произведены многими способами. Однако нахождение всех мыслимых гипотез не является оправданным. Во-первых, одни из таких гипотез являются более правдоподобными, чем другие, причем появление на выходе системы менее вероятных гипотез нежелательно, если имеются более вероятные. Во-вторых, быстродействие различных методов интерпретации и исправления ошибок резко различно. Поэтому более медленный способ исправления ошибки не применяется, если применение более быстрого дало удовлетворительные результаты.

С другой стороны, наряду с вариантом анализа слова как правильного в некоторых случаях требуется получить сразу варианты с исправлением ошибок некоторого типа. Например, пусть в словаре имеются слова *Панама* (страна) и *панама* (головной убор). При анализе текста «*Панама ей очень идет!*» желательно получить оба варианта — как вариант *Панама* без ошибки, так и вариант *панама* с сообщением «первая буква заглавная», что интерпретируется системой как ошибка в слове.

С целью управления процессом интерпретации и исправления ошибок все множество типов ошибок, интерпретируемых системой, разбивается на три ступени. К первой ступени по умолчанию относятся некоторые ошибки в атрибутах текста, такие, как «первая буква заглавная» или «должно быть ё». Ко второй ступени относятся грамматические ошибки. К третьей ступени относятся опечатки.

Анализ каждого слова может производиться в один, два или три этапа. На каждом этапе интерпретируются ошибки соответствующей ступени либо менее серьезные. Для этого в алгоритме анализа разрешается или запрещается проведение определенных проверок, как это рассмотрено в пункте 3.4.6, в зависимости от номера этапа. Если найдена интерпретация слова, последующие этапы не выполняются. Таким образом, почти все слова анализируются за один этап.

Кроме того, каждому типу ошибок присваивается определенный «вес». На выходе системы генерируются варианты исправления ошибок только минимально возможного для данного слова веса, но только из тех вариантов, которые получены за минимальное число этапов. Распределение весов не совпадает с распределением ошибок по ступеням, поскольку последнее производится с учетом быстродействия каждого метода.

Как распределение типов ошибок по ступеням, так и их веса могут изменяться и контролироваться прикладной программой путем установки и маскирования определенных битов в слове состояния документа. При неудовлетворительных результатах интерпретации слова прикладная программа может повторить анализ с другими параметрами.

3.4.8. Приближенные анализ, синтез и связанные с ними задачи. Пополнение словаря

Под задачами приближенного морфологического анализа и синтеза мы понимаем задачи осуществления соответствующих операций над словом, встретившемся в тексте, но отсутствующем в словаре системы. Например: определить падеж, род и число встретившегося в тексте слова *картриджами*; построить словоформу родительного падежа множественного числа этого слова. Как правило, такие задачи решаются неоднозначно, например: *нет *картриджев, *картридж, картриджей*. Новое слово всегда сначала встречается в виде буквенной цепочки в тексте, поэтому первой приближенной операцией над словом всегда оказываестя мофологический анализ.

Задача осуществления приближенных морфологических операций над словом сводится к более общей задаче, имеющей важное самостоятельное значение — к задаче приближенного определения словарных грамматических показателей неизвестного слова, то есть определения его класса словоизменения. После того, как класс словоизменения определен, все морфологические операции над словом выполняются в обычном порядке; кроме того, новое слово может быть внесено в словарь системы.

Определение грамматических показателей новых слов производится методом, аналогичным применяемому в [13, 22, 23, 24], с некоторыми модификациями. Мы не сочли целесообразным составление специального программного обеспечения для решения задачи приближенного анализа, несмотря на некоторое замедление процесса приближенного анализа примененным нами методом. Использование специального алгоритма привело бы к неоправданному расходу оперативной памяти либо к еще большему замедлению процесса анализа при загрузке специального программного модуля в память как оверлейного сегмента. Вместо этого используется обычный алгоритм анализа в сочетании со специально составленным словарем.

Словарь для приближенного морфологического анализа составляется автоматически на основе обычного морфологического словаря и имеет аналогичный формат. На вход алгоритма формирования такого словаря подаются словарные статьи основного словаря системы, сгруппированные по лексемам. Так, словарные статьи основ *точн-*(*ый*) и *точен-** рассматриваются как одна группа. От каждой основы группы последовательно отбрасываются слева одна, две и т.д. буквы, при условии, что отбрасываемые буквы одинаковы во всех основах лексемы, и ни одна из основ после отбрасывания буквы не становится пустой цепочкой. Для лексемы ТОЧНЫЙ таким образом порождаются группы {*-очен-, -очн-*}, {*-чен-, -чн-*}, {*-ен-, -н-*}; для лексемы ЛЕД — группа {*-ед-, -ьд-*}.

Набор полученных квазилексем упорядочивается, и из одинаковых квазилексем в словаре остается только одна. Так, квазилексема {*-ен-, -н-*: прилагательное, ...} с набором грамматических признаков лексемы ТОЧНЫЙ может быть получена также и из лексемы ПРОЧНЫЙ; из таких одинаковых наборов сохраняется только один. Наборы, имеющие различную грамматическую информацию, считаются разными. Так, квазилексема {*-ен-, -н-*: существительное, ...} с набором грамматических признаков лексемы БУДНИ не совпадает с квазилексемой, полученной из ТОЧНЫЙ и ПРОЧНЫЙ. Подобные квазилексемы отражают варианты классов словоизменения имеющихся в словаре реальных лексем.

При объединении одинаковых квазилексем в специальное поле словарной статьи либо в отдельный файл для каждой квазилексемы записывается число объединенных в ней одинаковых квазилексем в исходном массиве. Это число отражает частотность соответствующего класса словоизменения относительно словаря (но не относительно текста).

С целью экономии объема словаря малочастотные квазилексемы могут быть удалены из словаря. Уже удаление квазилексем с частотностью меньше трех практически не влияет на точность результатов приближенного морфологического анализа, но значительно сокращает словарь квазилексем даже по сравнению с исходным словарем реальных лексем.

Рассмотрим алгоритм приближенного морфологического анализа с использованием построенного таким образом словаря классов словоизменения (квазилексем). На вход алгоритма подается буквенная цепочка, не опознанная по основному словарю системы. От этой цепочки последовательно отбрасываются первая, вторая и т.д. буквы, а оставшаяся правая часть цепочки подвергается анализу по словарю классов словоизменения. Процесс останавливается на цепочке, успешно проанализированной по данному словарю. На исходную неопознанную буквенную цепочку переносятся результаты анализа усеченной цепочки и грамматические показатели найденной квазилексемы.

Так, для цепочки *картриджами* может оказаться успешным анализ правой подцепочки *-джами* как формы творительного падежа квазиосновы *-дж-*, унаследовавшей грамматические показатели лексем ХАДЖ, КОЛЛЕДЖ, КОТТЕДЖ, БРИДЖ. Соответственно возможным становится синтез формы *картриджей*.

При разрешении неоднозначности анализа используется информация о частотности классов, то есть о количестве усеченных основ исходного словаря, «склеенных» в одну квазиоснову словаря классов (см. выше). Так, пусть для неопознанной цепочки *легитимен* найдены три квазиосновы *-ен-*, унаследовавшие показатели сооветственно существительного БУДНИ, прилагательного ПРОЧНЫЙ и прилагательного УДОБНЫЙ. Однако первые две квазилексемы имеют частотность порядка десятков, последняя — около девяти тысяч. В этом случае выбирается набор грамматических показателей прилагательного УДОБНЫЙ, и может быть построена нормализованная форма исходного слова *легитимный*.

Когда рассмотренный выше процесс приводит к нескольким высокочастотным квазилексемам, приближенный анализ оказывается неоднозначным. Когда все найденные квазилексемы малочастотны, производится попытка продолжить процесс отсечения букв от исходной цепочки. Частоты полученных при этом квазилексем при сравнении с частотами первоначально полученных квазилексем учитываются с понижающим коэффициентом на каждую лишнюю отсеченную букву. В нашей реализации было выбрано значение 30, по числу букв в алфавите.

Тот факт, что при разрешении неоднозначности должны использоваться показатели частотности, определенные не по тексту, а по словарю, интуитивно очевиден. Действительно, хотя в тексте чаще всего встречаются предлоги, вероятность появления *нового* предлога крайне низка, как низка и частотность предлогов в словаре. Наоборот, вероятность появления в тексте нового термина, образованного по продуктивной модели, велика, как велика и доля в словаре таких слов. Более точно, чем вероятнее слово определенного класса может быть встречено в тексте как *новое*, тем чаще такие

слова включаются в словарь и тем больше их доля в словаре. Наличие и статистические показатели всех слов в тексте, кроме новых, не влияют ни на процесс формирования словаря, ни на определение вероятности отнесения очередного нового слова к определенному грамматическому классу.

В приведенном рассуждении не учитывется информация о частотности отдельных морфологических форм слов, например, формы творительного падежа. Такая информация не может быть получена автоматически по имеющемуся морфологическому словарю. Однако если такая информация имеется, она учитывается при разрешении неоднозначности приближенного анализа.

Изложенный алгоритм обладает рядом ограничений. Так, не рассматриваются квазилексемы с пустыми основами. Снятие такого ограничения привело бы не только к необходимости пересмотра алгоритмов доступа к словарю, но и к неоправданному росту неоднозначности анализа и к появлению на выходе алгоритма недостаточно надежных вариантов приближенного анализа. Изложенный же алгоритм не позволяет производить анализ слов, требующих пустых квазиоснов, то есть тех слов, результаты автоматического анализа которых в любом случае были бы недостаточно надежны.

Для назначения грамматических показателей таким словам с целью внесения их в словарь используется принципиально другой алгоритм, работающий в интерактивном режиме под управлением человека. Используется словарь квазилексем, аналогичный рассмотренному выше, однако включающий также и пустые основы. В целях сокращения объема словаря в словарь включаются только квазилексемы, полученные отбрасыванием от основ реальных лексем максимальной левой общей подцепочки всех основ лексемы. Таким образом, полученный словарь характеризуем типичные классы словоизменения, включая типичные случаи чередования букв в основе.

Оператору предлагается упорядоченный по алфавиту набор новых слов, составленный по некоторому корпусу текстов. Оператор помечает в этом наборе элементы, являющиеся словоформами одно лексемы. Автоматически из словаря выбираются все квазилексемы (гипотезы), каждая из которых образует все помеченные оператором словоформы. Выбранный набор гипотез упорядочивается в соответствии с имеющейся информацией об их частотности, и оператору предъявляется парадигма наиболее вероятной квазилексемы с указанием ее грамматических характеристик.

Если парадигма содержит неверные формы или неверные грамматические показатели, такие, как часть речи или род, оператор выбирает на экране дисплея нужную позицию парадигмы или нужную графу грамматических характеристик. По всем отобранным лексемам автоматически строится набор возможных вариантов значения выбранного поля. Из данного набора вариантов оператор выбирает правильный, после чего в наборе гипотез оставляются только те, которые содержат выбранный вариант. Процесс завершается, когда все позиции парадигмы оказываются правильными.

Работа оператора облегчается тем, что на экране выделяются цветом те позиции парадигмы, по которым в наборе гипотез вообще имеется неоднозначность. Из них же автоматически выбирается и отмечается цветом та позиция, для которой точное определение словоформы с наибольшей вероятностью приведет к сокращению набора гипотез. Как показали эксперименты, для большей части новых слов бывает достаточно выбора двух-трех позиций парадигмы.

3.5. Программная реализация системы

Программная реализация системы была выполнена на языке программирования Си. Ядро системы было реализовано в виде библиотеки процедур в двух вариантах: в виде статически связываемой библиотеки и динамически связываемой библиотеки (DLL). Кроме того, был реализован набор программ для подготовки словарей и лингвистических таблиц.

Исходный текст процедур ядра системы на языке Си составил 11275 строк и занимает 271 килобайт. Исходный текст программ подготовки словаря составил 6623 строк и занимает 156 килобайт, а текст компилятора лингвистических таблиц — соответсвенно 2317 строк и 51 килобайт. Общий объем исходного текста программ системы — 20215 строк, или 478 килобайт.

Интерфейс ядра системы с прикладной программой составляет 69 функций и 22 именованные константы.

Лингвистическое обеспечение системы для русского языка состоит из таблиц и описания процесса преобразования словаря. Внутренний словарь системы построен путем специальной переработки текста словаря А.А.Зализняка [69] и некоторых других словарей. Описание процесса переработки состоит из 10 тысяч строк исходного текста на специально разработанном языке и занимает 310 килобайт. Лингвистические таблицы состоят из 1022 строк исходного текста и занимают 50 килобайт. Размер таблиц в скомпилированном виде, используемом системой непосредственно, — 12,5 килобайт.

ВЫВОДЫ

1. Разработана модель морфологического строения флективного естественного языка, эффективно реализуемая на современных ЭВМ. Под эффективностью реализации понимается небольшая потребность в оперативной памяти в сочетании с высоким быстродействием.
2. Предложенная модель является языково-независимой в некотором классе флективных и агглютинативных языков; лингвистическая информация в модели отделена от алгоритмов. Опыт практической разработки и эксплуатации лингвистического обеспечения имеется для русского языка.
3. Модель позволяет интерпретировать, объяснять и исправлять некоторый класс характерных неоднобуквенных грамматических ошибок, совершаемых лицами, плохо владеющими русским языком.
4. На основании предложенной модели автором разработаны и программно реализованы алгоритмы решения задач точного и полного морфологического анализа и синтеза, а также ряд смежных задач, таких как нормализация слов, обнаружение, интерпретация и исправление ошибок и опечаток, приближенный анализ новых слов. Решение смежных задач не требует привлечения дополнительной лингвистической информации, практически не требует использования дополнительных блоков программного обеспечения и требует минимальной модификации алгоритмов анализа и синтеза.
5. Система отличается, в частности, следующими особенностями. При анализе и нормализации слову назначается уникальный числовой идентификатор, словарно закрепленный за словом и не меняющийся при изменении словаря, причем диапазон значений идентификаторов близок к количеству основ в словаре. При анализе текст автоматически с привлечением лингвистической информации членится на слова, что предполагает также обработку сложных случаев переноса слов и анализ сложных слов и композитных образований.
6. При участии автора разработан и реализован на ЭВМ интерфейс стандартной библиотеки морфологических процедур, предназначенной для решения задач точного и полного морфологического анализа и синтеза, нормализации слов, обнаружения, интерпретации и исправления ошибок и опечаток, и ряда других задач морфологической обработки текста. Использование библиотеки полностью избавляет прикладную программу от необходимости работы с поверхностным, буквенным представлением текста.
7. При участии автора созданы система морфологических таблиц, описывающая морфологическое строение русского языка в рамках предложенной модели, а также машинный морфологический словарь русского языка (на основе словаря [69] и некоторых специальных словарей из фондов ВНТИЦентра), включающий около 130 тыс. лексем общеупотребительной и специальной лексики.

ГЛАВА 4.
ИСПРАВЛЕНИЕ ОРФОГРАФИЧЕСКИХ ОШИБОК С ПОМОЩЬЮ ПЕРЕБОРА, УПРАВЛЯЕМОГО МОРФОЛОГИЧЕСКИМ СЛОВАРЕМ

4.1. Постановка задачи словарного исправления орфографических ошибок

4.1.1. Основные определения

Орфографически ошибочным словом называется буквенная цепочка, полученная некоторым преобразованием из словоформы некоторой лексемы, имеющейся в системном словаре. Под исправлением ошибки в таком слове называется установление исходной словоформы. Исходная словоформа определяется неоднозначно. В данной постановке задача исправления ошибки называется также задачей полного словарного исправления. Результатом попытки исправления ошибки в пределах некоторого класса преобразований может быть также установление невозможности ее исправления, то есть несуществования в словаре словоформы, из которой данная цепочка получается каким-либо преобразованием заданного класса.

Методы и вычислительная трудоемкость исправления ошибок резко различны для разных классов рассматриваемых преобразований. Грамматическими называются ошибки, связанные с искажениями, затрагивающими различные неповерхностные морфологические уровни текста; исправление некоторого класса таких ошибок с привлечением лингвистической информации рассмотрено в пункте 3.4.6. Опечатками называются ошибки, связанные с поверхностным, буквенным представлением слова, то есть с искажениями непосредственно буквенной цепочки, представляющей словоформу.

В более общей постановке под исправлением опечаток можно понимать алгоритм исправления ошибок, не опирающийся на морфологические, фонетические, синтаксические и т.п. представления. В данной главе рассматриваются методы исправления опечаток в слове, взятом вне контекста.

С целью объяснения и исправления ошибок выделяется некоторый класс элементарных искажений. Например, замена одной буквы на другую, перестановка гласной буквы через согласную, сдвиг руки на одну позицию при набивке части слова на клавиатуре. Сложными называются ошибки, являющиеся комбинацией нескольких элементарных. Например, замена двух букв в одном слове. Элементарное искажение называется локальным, если оно по определению затрагивает небольшой отрезок буквенной цепочки, например, не больше трех букв. Цепочка называется словом с одиночной ошибкой, если она содержит только одно элементарное искажение. Цепочка называется словом с изолированными ошибками, если она содержит только локальные элементарные искажения, затрагивающие непересекающиеся участки, удаленные друг от друга на некоторое фиксированное количество символов. Например, цепочка **эликтрефикацея* содержит три изолированные ошибки, а цепочка **каррова* содержит неизолированные ошибки.

Соответственно различаются три возможных постановки задачи исправления опечаток: исправление одиночных опечаток, исправление не более чем n элементарных опечаток в слове, исправление изолированных опечаток. Постановка задачи исправления опечаток как задачи исправления изолированных опечаток представляется наиболее оправданной, поскольку вероятность совершения опечатки на участке текста определяется скорее количеством букв, чем количеством слов на данном участке.

Одним из важнейших классов элементарных искажений является класс однобуквенных ошибок, включающий замены, вставки, пропуски букв и перестановки двух соседних букв. Одиночные ошибки данного класса составляют около 90% всех искажений в текстах. К таким опечаткам может быть сведено и большинство грамматических ошибок. В последнем случае ошибка может быть правильно исправлена и проинтерпретирована, но ее объяснение не является правильным. Например, *сильнейшвй* vs. *сильнейшый*. Объяснение ошибок последнего типа рассмотрено в пункте 3.4.6.

Таким образом, исправление опечаток определенных классов, в том числе однобуквенных, является практически важной задачей. Ввиду большой вычислительной сложности применяемых в настоящее время методов возможность быстро и без привлечения значительного объема дополнительной информации решать задачу исправления опечаток является важной характеристикой

морфологической системы, ориентированной на практическое использование. Алгоритмы исправления ошибок в русских словах должны учитывать особенности русского языка как высоко флективного.

4.1.2. Формальная постановка задачи исправления опечаток в слове, взятом вне контекста

Процедурой сопоставления со словарем, или процедурой анализа, называется машинная процедура, на вход которой подается произвольная цепочка букв, а на выходе генерируется булево значение — *допустимо* или *недопустимо*. В процессе работы этой процедуры используется словарь — структура данных, особенности физической организации которой учитываются при оценке эффективности алгоритма. Допустимые слова называются также словами, имеющимися в словаре. К процедуре сопоставления со словарем предъявляются также дополнительные требования, уточняемые в дальнейшем изложении.

Фиксируется класс преобразований, приводящих к ошибкам, исправляемым данным алгоритмом. Например, класс изолированных однобуквенных опечаток. Буквенные цепочки, являющиеся прообразами данной цепочки при применении преобразований выбранного класса, называются близкими к ней.

Вариантом исправления слова называется близкая к нему цепочка, имеющаяся в словаре. Правильным вариантом исправления называется тот из вариантов, который в действительности имелся в виду автором данного текста. Правильность варианта устанавливается человеком-экспертом на основании анализа контекста. Заметим, что так определенный правильный вариант существует не всегда, поскольку истинная буквенная цепочка может либо сама отсутствовать в словаре, либо содержать ошибку, не входящую в класс исправляемых данным алгоритмом. Например: **каррова,* **Якабсон.*

Под задачей полного исправления опечаток понимается задача нахождения всех вариантов исправления. Под задачей почти полного исправления опечаток понимается задача нахождения в среднем подавляющего большинства вариантов исправления. С практической точки зрения такая постановка задачи исправления опечаток эквивалентна задаче полного исправления опечаток, поскольку класс исправляемых опечаток в любом случае не исчерпывает всех практически возможных ошибок. Может быть также рассмотрена задача определения для каждого найденного варианта вероятности того, что он является правильным. Такая задача называется ранжированием вариантов.

Предполагается, что общая схема работы алгоритма исправления опечаток является циклической. На каждом шаге формируется некоторая близкая к исходной цепочка. Затем она предъявляется процедуре сопоставления со словарем. Если она оказывается допустимым словом, она добавляется к выходному набору найденных вариантов исправления. Работа алгоритма заканчивается, когда формирование очередной новой близкой цепочки невозможно.

Цепочки, подаваемые алгоритмом на вход процедуры сопоставления со словарем, называются гипотезами. Множество гипотез является подмножеством множества цепочек, близких к исходной, но не обязательно совпадает с ним, поскольку на основании какой-либо дополнительной информации некоторые близкие цепочки могут быть отвергнуты априорно. Оптимизация алгоритма исправления опечаток заключается как в уменьшении числа гипотез, так и в выборе наилучшего порядка их проверки.

Алгоритм основан на переборе, управляемом морфологическим словарем, если при построении очередной гипотезы используется информация, полученная при анализе предыдущей гипотезы. В этом случае кроме булева значения *допустимо*/*недопустимо* процедура сопоставления со словарем поставляет некоторую дополнительную информацию.

4.1.3. Критерии оптимизации алгоритма исправления опечаток

Рассматриваются два основных показателя эффективности алгоритма: число обращений к процедуре сопоставления слова со словарем и число обращений к дисковой памяти, производимых в процессе работы этой процедуры. В зависимости от конкретной реализации процедуры анализа каждый из этих показателей может определять время работы алгоритма. В дальнейшем для краткости любой из данных показателей будет называться просто временем работы алгоритма.

Каждый из данных показателей учитывается в четырех вариантах. Число соответствующих событий подсчитывается от начала работы алгоритма до получения первого варианта исправления, до получения правильного варианта, последнего варианта исправления и до завершения работы алгоритма.

Заметим, что моментом завершения работы алгоритма является не момент выдачи последнего варианта исправления, а момент проверки последней из порождаемых им гипотез.

Необходимость сокращения времени от начала работы алгоритма до ее завершения очевидна, в особенности при пакетном режиме работы системы. В интерактивном режиме более важным является время ожидания правильного варианта, поскольку в этом случае пользователь может остановить работу алгоритма после получения правильного варианта. Поскольку правильный вариант может быть получен не раньше первого варианта, время ожидания первого варианта также должно минимизироваться. Более того, при интерактивной работе с психологической точки зрения важно сократить время ожидания ответа системы.

Также крайне существенной является минимизация среднего времени ожидания последнего варианта исправления. Например, при почти полном исправлении опечаток работа алгоритма принудительно завершается по истечении среднего времени нахождения последнего варианта. Следовательно, время работы алгоритма в случае почти полного исправления опечаток определяется средним временем ожидания последнего варианта исправления. Другой задачей, требующей сокращения времени получения последнего варианта, является задача ранжирования вариантов, рассмотренная в пункте 4.7.

Таким образом, желательным поведением алгоритма является такое, при котором после начала его работы почти сразу появляются варианты исправления, а затем, после неизбежной паузы, выдается сообщение о том, что других вариантов нет. Напротив, нежелательным поведением при том же общем времени работы является такое, при котором варианты выдаются равномерно в течение всего времени работы алгоритма.

Очевидно, число обращений к процедуре анализа равно числу порожденных гипотез. Для подсчета числа обращений к дисковой памяти необходимо рассмотреть более конкретные предположения о структуре используемого словаря. Предполагается, что словарь представляет собой большой массив, хранимый на диске или другом устройстве с блочным доступом. В каждый момент в оперативной памяти может находиться только относительно малый участок словаря, называемый блоком. Считывание нового блока в память и является элементарным обращением к дисковой памяти. При очередном обращении к процедуре анализа новый блок считывается только в том случае, если он не остался в памяти от предыдущего обращения. Словарь упорядочен лексикографически, так что слова, близкие в алфавитном порядке, находятся в одном и том же блоке словаря. Таким образом, требование минимизации числа обращений к диску состоит в требовании группировать вместе гипотезы, близкие по алфавиту, при циклическом обращении к процедуре анализа.

В предлагаемых в настоящей работе алгоритмах минимизация времени ожидания правильного варианта является простым следствием минимизации времени ожидания первого и последнего вариантов. Никаких специальных мер для ускорения нахождения именно правильного варианта не предпринимается. Это продиктовано, во-первых, отказом от использования дополнительных эмпирических данных о языке и механизме совершения опечаток.

Во-вторых, поскольку, как показывают эксперименты, время ожидания последнего варианта близко к времени ожидания первого варианта, существенное изменение стратегии порождения гипотез только ухудшило бы все показатели эффективности. Так, близкая к оптимальной стратегия получения первого варианта, рассмотренная в [42], дает несколько худшие результаты, чем предлагаемые в настоящей работе алгоритмы, см. пункт 4.6. Необходимо уточнить, что алгоритм [42] близок к оптимальному в классе алгоритмов, не учитывающих словарную информацию и особенности структуры словаря. Однако требования учета особенностей структуры словаря и учета эмпирических данных, приводимых в [42], являются противоречивыми.

Ниже в пункте 4.2 рассматриваются упрощенные варианты алгоритмов, исправляющие ошибки только одного типа по словарю упрощенной структуры. В пункте 4.3 приведен алгоритм, использующий реальный морфологический словарь, рассмотренный в главе 2, и морфологическую модель, рассмотренную в главе 3. В пунктах 4.4 и 4.5 приводится обобщение алгоритма исправления опечаток. Данный алгоритм исправляет все однобуквенные и некоторые неоднобуквенные изолированные ошибки.

4.2. Исправление одиночной ошибки типа замены буквы по словарю упрощенной структуры

4.2.1. Минимизация времени от начала до завершения работы алгоритма

В настоящем пункте для простоты предполагается, что словарь представляет собой просто лексикографически упорядоченный набор слов. Задача состоит в том, чтобы для данной цепочки букв, не содержащейся в словаре, найти все слова из словаря, близкие к ней. В качестве критерия близости для простоты рассматриваются только однобуквенные замены: два слова считаем близкими, если их длины одинаковы и они различаются ровно одной буквой. Алгоритмы, исправляющие более широкий класс ошибок по реальному морфологическому словарю, рассмотрены в пунктах 4.3 — 4.5.

На вход алгоритма подается буквенная цепочка *W*. Рассмотрим введенные в пункте 2.5 понятия позиции несовпадения *D* и ближайшей большей буквы *d*. Введем обозначение *w* для буквы, стоящей в цепочке *W* на позиции *D*. Основным свойством позиции *D* является то, что никакие изменения исходной цепочки *W*, оставляющие неизменными первые *D* букв цепочки *W*, не могут привести к допустимой цепочке. Основным свойством буквы *d* является то, что никакие изменения цепочки *W*, оставляющие неизменными первые $D-1$ букв и приводящие к появлению на позиции *D* любой буквы, *большей* по алфавиту, чем *w*, но меньшей, чем *d*, также не приводят к допустимой цепочке.

Существо приводимых в данной главе алгоритмов заключается в априорном отказе от гипотез, порождаемых подобными преобразованиями *W*. Существенно, что определенные в пункте 2.5 значения *D* и *d* являются наилучшими среди значений с данным свойством, устанавливаемых на основании анализа только пары соседних слов словаря. Однако не менее существенно то, что при замене этих значений на другие, дающие более слабые оценки, с сохранением указанных основных свойств, правильность работы приводимых в данной главе алгоритмов не нарушается, хотя эффективность может ухудшиться.

Дополнительным требованием к процедуре сопоставления слова со словарем является требование установления в процессе сопоставления позиции несовпадения *D* и ближайшей большей буквы *d*. Применяемые обычно алгоритмы поиска в словаре в любом случае получают такую информацию, но не выдают ее в качестве одного из результатов работы. Заметим, что для получения требуемой информации необходим анализ только двух ближайших по алфавиту к данному слов словаря. В случае древесной организации словаря необходимая информация также может быть легко получена.

Предлагаемый алгоритм перебирает гипотезы в алфавитном порядке, в соответствии с установленным в пункте 4.1.3 требованием. Получаемая на каждом шаге информация о ближайшей большей букве используется для априорного отсева гипотез, содержащих на позиции несовпадения *D* буквы, меньшие по алфавиту. На каждой итерации алгоритм производит на некоторой позиции *V* в исходном слове *W* замену буквы на новую букву *v*. Позиция *V* называется текущей варьируемой позицией.

Алгоритм 1

Шаг 1. В качестве варьируемой позиции *V* выбирается первая позиция цепочки *W*.

Шаг 2 начинает внешний цикл. Варьируемая позиция *V* фиксирована. В качестве первой буквы для замены *v* выбирается первая буква алфавита.

Шаг 3 начинает внутренний цикл. Буква на варьируемой позиции *V* заменяется на выбранную букву *v*. Полученная гипотеза предъявляется процедуре сопоставления со словарем.

Если гипотеза найдена в словаре, она выдается в качестве варианта исправления ошибки. Следующей буквой для замены *v* становится буква, следующая по алфавиту за текущим значением *v*, а если такой нет, то символ конца алфавита. Заметим, что если гипотеза найдена в словаре и $D_2=V$, то в качестве следующего значения *v* может также быть взято d_2.

Если гипотеза не найдена в словаре, рассматриваются возвращенные процедурой анализа номер *D* и буква *d*. Если $D > V$, следующее значение *v* выбирается как в предыдущем случае. Если $D=V$, выбирается $v=d$, что приводит к уменьшению числа гипотез. Заметим, что в этом случае *d* может быть равно символу конца алфавита. Если $D < V$, значением *v* становится символ конца алфавита. Последняя возможность осуществляется только один раз, в момент наибольшей глубины рекурсии на шаге 4.

Шаг 4. Предварительно рассмотрим план действий на данном шаге. Для обеспечения правильного алфавитного порядка перебора гипотез необходимо организовать перебор текущей варьируемой позиции *V* в две фазы — прямой и обратный проход. В момент, когда *v* сравняется с буквой, стоящей на позиции *V* в исходном слове *W*, вместо испытания гипотезы, совпадающей с

исходным словом, необходимо рекурсивно применить алгоритм к оставшейся части слова. Когда *v* есть символ конца алфавита, следует выйти из рекурсии.

Итак, рассмотрим букву *v*, выбранную для замены. Если она по алфавитному порядку меньше буквы, стоявшей на данной позиции в исходной неправильной цепочке, или если выполняется обратный проход по позициям слова, производится переход к шагу 3. Если на прямом проходе по позициям цепочки *W* буква для замены *v* оказалась по алфавитному порядку больше буквы в исходном слове *W* на позиции *V*, осуществляется вход в рекурсию. Буква *v* запоминается в стеке, номер позиции *V* увеличивается, осуществляется переход к шагу 2. Если значением *v* является символ конца алфавита, то осуществляется выход из рекурсии: номер варьируемой позиции *V* уменьшается, из стека выбирается ранее запомненная буква *v*, и выполняется переход к шагу 3. Тем самым продолжается перебор на данном уровне рекурсии, прерванный ранее на прямом проходе. Однако, если номер позиции *V* не может быть уменьшен, работа алгоритма заканчивается.

Конец алгоритма 1.

Для работы алгоритма 1 необходим один стек из *N*–1 байта для хранения запоминаемых при входе в рекурсию букв, где *N* не превышает как число букв в слове *W*, так и число букв в самом длинном слове словаря. Дополнительной памяти для хранения проверяемой гипотезы не требуется, поскольку все изменения можно производить прямо в цепочке *W*, каждый раз возвращая измененную букву на место при смене текущей позиции.

Порядок изменения значения *V* легко понять на примере алфавитно упорядоченного множества гипотез замены буквы в слове *дом*. Показаны только буквы *а* и *я*:

аом, *V*=1
д**а**м, *V*=2
до**а**, *V*=3
до**я**, *V*=3
д**я**м, *V*=2
яом, *V*=1

Содержательно смысл алгоритма 1 состоит в том, что он порождает не все, а только необходимые гипотезы.

Следует подчеркнуть, что данный алгоритм, в отличие от [44] или [143], не требует никаких априорных знаний о языке, составе словаря и механизме совершения ошибок в словах. Фактически не требуется даже информация об алфавите данного языка. На шаге 2 в качестве первой буквы алфавита может быть выбран первый символ кодового набора ЭВМ, т.е. 0 или -1. В этом случае уже на следующей итерации цикла шага 3 будет найдено правильное значение, и только в редких случаях количество итераций шага 3 увеличится на одну. Таким образом, алгоритм правильно работает и в том случае, когда в словаре есть слова, содержащие небуквенные символы, например, *OS/2*, или когда состав и язык словаря вообще не известны заранее.

Алгоритм фактически варьирует буквы только в пределах той начальной части слова, которая совпадает с начальной частью какого-либо слова из словаря. Часто это позволяет сократить исследуемый участок слова приблизительно вдвое. Кроме того, для работы алгоритма не требуется априорно определять конец слова в предъявленном потоке букв. Это может оказаться важным, когда в словаре есть слова, содержащие пробел и/или знаки препинания, например, «*и т.д.*», «*во что бы то ни стало*». Подробнее данный вопрос рассмотрен в пункте 3.1.1. Потребность в оперативной памяти ЭВМ при этом априорно ограничена максимальной длиной слова в словаре.

При полном переборе число гипотез равно *N* x *L*, где *N* — число букв в слове *W*, а *L* — число букв в алфавите. Алгоритм позволяет в среднем сократить примерно вдвое число просматриваемых позиций слова и в несколько раз — среднее число испытываемых букв алфавита. Более того, он не требует априорного знания ни *N*, ни *L*.

Данный алгоритм оптимизирует как количество обращений к процедуре анализа, так и количество обращений к дисковой памяти, рассчитанные от начала до конца работы алгоритма, то есть до испытания последней гипотезы. Однако время получения первого, правильного и последнего вариантов исправления ошибки не минимизируется. Модификация алгоритма, существенно улучшающая данные показатели, рассмотрена в пункте 4.2.2.

4.2.2. Минимизация времени нахождения первого и последнего вариантов исправления

Потребуем минимума числа проверяемых гипотез и обращений к дисковой памяти, рассчитанных до получения первого и до получения последнего вариантов. Улучшение этих показателей не приводит к существенному ухудшению общего времени работы алгоритма по сравнению с алгоритмом, рассмотренным в пункте 4.2.1.

Задача состоит в такой модификации алгоритма 1, чтобы он выдавал почти все варианты в начале своей работы, а затем проверял гипотезы, большая часть из которых неверна. Существует два способа такой организации работы. Первый состоит в том, что сначала проверяются наиболее вероятные гипотезы, а затем наименее вероятные. Данный способ рассмотрен, например, в [42].

Второй способ состоит в том, что сначала проверяются гипотезы, проверка которых происходит быстро, а затем гипотезы, требующие значительного времени. В данной работе применяется последний способ. Используется то, что проверка не каждой гипотезы приводит к новому обращению к дисковой памяти. В начале работы проверяются гипотезы, не требующие обращений к дисковой памяти. Количественно это множество гипотез составляет большую часть всех проверяемых гипотез. Кроме того, это же множество гипотез с наибольшей вероятностью содержит все варианты исправления.

Будем использовать буквенные обозначения, введенные в пункте 4.2.1. Все слова, близкие к исходному слову W и отличающиеся от него только по далеким позициям, т.е. имеющие с ним общий начальный участок существенной длины, расположены по алфавитному порядку близко друг от друга и, весьма вероятно, в одном с ним или близких блоках словаря. Напротив, слова, отличающиеся от W по первым буквам, расположены в разных блоках словаря, и анализ каждого из них требует считывания нового блока, т.е. элементарного обращения к дисковой памяти. Кроме того, именно при варьировании непервых букв слова происходит наиболее интенсивный априорный отсев гипотез на шаге 3 алгоритма 1.

Искомая модификация алгоритма состоит в том, что варьируемая позиция V изменяется от конца слова к его началу. Необходимо заметить, что никакие исправления цепочки W на участке слова за позицией несовпадения D не приведут к допустимому слову, поскольку уже отрезок слова W до позиции D не является начальным отрезком никакого слова из словаря. Поэтому варьированию подлежат буквы только на позициях меньше D. Во многих случаях это соображение сразу отсекает большое число заведомо ложных гипотез, как, например, в слове * *эликтрификация*.

Алгоритм 2

Шаг 1. В качестве варьируемой позиции V выбирается D.

Шаг 2 начинает внешний цикл. Варьируемая позиция V фиксирована. Буквы алфавита перебираются не от начала к концу алфавита, как в алгоритме 1, а от буквы w, стоящей на позиции V в исходном слове W, до конца алфавита; затем от начала алфавита до w. Такой порядок перебора способствует оптимальному использованию блока словаря, уже имеющегося в оперативной памяти ЭВМ от предыдущего обращения к процедуре анализа. Таким образом, в качестве первой буквы для замены v выбирается буква, следующую по алфавиту за w.

Шаг 3 Выбор нового значения v производится так же, как в алгоритме 1 из пункта 4.2.1. Затем производится поправка на циклический сдвиг алфавитного порядка, рассмотренный при описании шага 2. Именно, если v было больше, чем w, а стало символом конца алфавита, значение v заменяется на первую букву алфавита. Если v было меньше, чем w, а стало больше, чем w, или стало равно символу конца алфавита, то значение v заменяется на символ конца алфавита. Может также быть использовано то, что если $D_1 < V$, то переход на первую букву алфавита производить не обязательно.

Шаг 4. Если перебор букв на шаге 3 не закончился, осуществляется переход к шагу 3. Иначе, то есть если v есть символ конца алфавита, номер варьируемой позиции V уменьшается и осуществляется переход к шагу 2. Однако если V=0, работу алгоритма завершается.

Конец алгоритма 2.

Алгоритм не требует дополнительной оперативной памяти, кроме фиксированного количества переменных.

Данный алгоритм является более простым, чем алгоритм 2, и по совокупности всех оптимизируемых показателей намного более эффективным. Порядок изменения значения V

иллюстрируется на примере порядка обращений к процедуре анализа при *W*=«*дом*». Показаны только буквы *а* и *я*:

дом
*до**я***, *V*=3
*до**а***, *V*=3
*д**я**м*, *V*=2
*д**а**м*, *V*=2
***я**ом*, *V*=1
***а**ом*, *V*=1

Порядок перебора гипотез не является алфавитным, но близок к нему на каждом отдельном участке. Большая часть гипотез, причем именно из начального участка списка, расположена в одном и том же блоке словаря, оставшемся в оперативной памяти ЭВМ от первоначального анализа слова *W*. Поэтому до момента считывания первого нового блока с диска значительная часть гипотез оказывается проверенной, и в среднем несколько вариантов исправления найдено.

Поскольку алгоритм 2 не оптимален по отношению к общему числу обращений к диску, по сравнению с алгоритмом 1 он приводит к лишним обращениям. Такие лишние обращения могут возникать в момент нарушения алфавитного порядка гипотез, т.е. в момент обращения к процедуре анализа с гипотезой, лексикографически меньшей предыдущей. Из алгоритма 2 видно, что на тех его участках, где алфавитный порядок соблюдается, лишних обращений не происходит. Алфавитный порядок нарушается в алгоритме 2 один раз на каждую позицию слова, при переходе от конца алфавита к началу при переборе заменяемых букв. При переходе от текущей позиции к предыдущей алфавитный порядок следования гипотез не нарушается.

Следовательно, производится не более *N*-1 лишних обращений к диску, где *N* — длина слова. На самом деле их гораздо меньше, поскольку в качестве *N* может быть взята средняя длина общего участка всех слов одного блока, которая даже для очень крупного морфологического словаря обычно не превышает трех — четырех букв. Следовательно, за время от начала до конца работы алгоритм 2 совершает на три — четыре обращения к диску больше, чем алгоритм 1, что составляет около 2% общего числа обращений.

4.2.3. Экспериментальная проверка эффективности упрощенных алгоритмов исправления опечаток

Автором проводились эксперименты с использованием рассматриваемой в настоящей работе морфологической системы. Использовался крупный морфологический словарь, содержащий около 100 тысяч лексем. Как и ожидалось априорно, распределения исследуемых показателей оказались резко асимметричными. Наряду с эмпирическим средним свойства асимметричных распределений хорошо отражаются таким статистическим показателем, как медиана — число, превышаемое исследуемой случайной величиной в 50% случаев. Медиана лучше, чем эмпирическое среднее, отражает местоположение «основной массы» значений изучаемой величины и менее чувствительна к пусть значительным, но редким отклонениям. Следуя работе [42], мы использовали в качестве статистических показателей медианы.

Число обращений к диску, производимое алгоритмом 2 до выдачи последнего варианта, оказалось в 7.5 раза в среднем и в 59 раз по медиане меньше общего количества обращений к диску до конца работы алгоритма, то есть до момента проверки последней гипотезы. Другими словами, обычно все варианты определялись почти сразу, и затем, после некоторой паузы, выдавалось сообщение о том, что других вариантов нет. В отличие от алгоритма 2, алгоритм 1 затрачивал столько же времени, но выдавал найденные варианты иногда в начале, иногда в середине, а иногда в конце работы. Легко видеть, что моментом наибольшей продуктивности алгоритма 1 является момент наибольшей глубины рекурсии, очевидным образом зависящий от входного слова. В алгоритме 2 работа как раз начинается с этого момента, независимо от вида исходного слова.

Число обращений к диску, производимое до выдачи первого варианта, было в среднем в 18.1 раза меньше, чем до конца работы алгоритма, в то время как в алгоритме 1 среднее время ожидания первого варианта составляет около 50% общего времени работы алгоритма. Число обращений к диску до нахождения правильного варианта в 13.1 раз меньше общего числа обращений.

4.3. Использование крупного морфологического словаря

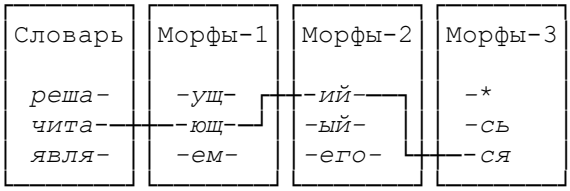
Модифицируем теперь изложенные в пунктах 4.2.1 и 4.2.2 методы с учетом использования не словаря словоформ, а словаря основ, содержащего информацию о построении парадигм в закодированном виде. Конкретно, используется словарь основ, рассмотренный в главе 2, и морфологический анализатор, рассмотренный в главе 3. При этом существенно используются только следующие информационные массивы: индексные массивы словаря (пункты 2.2, 2.4), поле ключа записи в самом словаре (пункты 2.2, 3.2.8) и списки морфов (пункт 3.2.5), причем в списке морфов используется только поле буквенной цепочки.

Схемы алгоритмов 1 и 2 не пересматриваются. Рассматривается способ получения определенных в пункте 2.5 позиции несовпадения D и ближайшей большей буквы d. Полученные значения непосредственно используются в алгоритмах 1 и 2 данной главы.

Предлагается для нахождения значений D и d использовать алгоритм 1 из пункта 2.5 со следующими изменениями. В качестве алгоритма поиска в словаре рассматривается алгоритм морфологического анализа, определенный в пункте 3.4.1, а в качестве процедур поиска в отдельном списке рассматриваются как процедуры поиска в словаре основ, так и процедуры просмотра списков морфов. При этом значение D, определяемое процедурой поиска в списке морфов, учитывается с поправкой, рассмотренной ниже. Приведенное в пункте 2.5 замечание об ослаблении условия в алгоритме не распространяется на данный случай.

Пусть процедура поиска в отдельном списке, как в словаре, так и в списке морфов, определяет значения D и d обычным образом, так, как они определены в пункте 2.5. При этом производится необходимая поправка, обеспечивающая отсчет D от начала всей анализируемой цепочки W. Например, для слова **читающего* процедура поиска цепочки *-ющего* в первом списке морфов дает D=2, d=«*щ*»; однако предполагается, что произведена поправка, обеспечивающая D=6, считая от начала цепочки **читающего*. Если при практическом кодировании алгоритма в качестве D используется адрес в памяти ЭВМ, то значение D автоматически получается правильным.

Для доказательства правильности работы алгоритма рассмотрим алгоритм морфологического анализа из пункта 3.4.1. Словоформа рассматривается как конкатенация подцепочек, первая из которых имеется в словаре основ, следующая — в списке морфов первой позиции, и т.д. Например, *чита-ющ-ий-ся*:



Алгоритм морфологического анализа путем просмотра словаря осуществляет поиск начальной подцепочки анализируемого слова в словаре, затем поиск начальной подцепочки оставшейся части слова в первом списке морфов, и т.д. Попытка разложения слова на такие подцепочки производится всеми возможными способами.

Произведением множеств буквенных цепочек M_1 и M_2 называется множество $M = M_1 \times M_2$ всех цепочек, начальная часть которых совпадает с некоторой цепочкой из M_1, а оставшаяся часть — с некоторой цепочкой из M_2. Таким образом, множество всех правильных словоформ есть подмножество произведения словаря и всех списков морфов. То, является ли данный элемент такого произведения правильной словоформой, устанавливается морфологическим анализатором на основании словарной информации. Здесь игнорируется разбиение лингвистических таблиц по частям речи; данный вопрос рассмотрен ниже.

Согласно замечанию, приведенному в пункте 4.2.1, вместо значений D и d в алгоритмах исправления опечаток могут быть использованы более слабые значения, конкретно, большее значение D и меньшее значение d. В частности, согласно утверждению 2 из пункта 2.5, могут быть использованы значения, определенные для данной цепочки W относительно множества, содержащего множество всех правильных словоформ. В качестве такого множества используется указанное произведение, поскольку учет грамматической информации неоправданно усложнил бы алгоритм.

При этом производится больше обращений к процедуре сопоставления со словарем, однако заведомо не больше, чем при полном переборе всех возможных гипотез. Поскольку списки морфов

хранятся в оперативной памяти ЭВМ, лишних обращений к дисковой памяти по сравнению со случаем использования списка только правильных словоформ не производится. Таким образом, задача сводится к нахождению значений D и d относительно произведения словаря основ и всех списков морфов.

Рассмотренная выше поправка на значение D означает, что фактически рассматривается не поиск части анализируемой цепочки в списке морфов, а поиск всей цепочки в множестве, являющемся произведением списка морфов на начальную часть цепочки, являющуюся произведением цепочек из предыдущих списков. Таким образом, доказательство правильности работы алгоритма определения D и d сводится к следующему утверждению. Обозначим анализируемую цепочку W.

Утверждение 1. Пусть $M = M_1 \times M_2 \times ... \times M_n$. Рассмотрим все возможные способы выделения в цепочке W некоторой начальной части w, являющейся элементом $M_1 \times M_2 \times ... \times M_{k-1}$, либо пустой цепочкой при k=1. Обозначим через D и d значения, определенные для W относительно M, а через D' и d' значения, определенные относительно множества M' вида $\{w\} \times M_k$. Уточним, что M', D' и d' определены неоднозначно. Тогда D есть максимальное среди всех значений D', а d есть минимальное среди всех значений d', соответствующих данному D.

Доказательство. Рассмотрим объединение M'' всех множеств вида M', и соответствующие ему величины D'' и d''. По утверждению 4 пункта 2.5, приведенные в условии минимальное и максимальное значения есть в точности D'' и d''. Следовательно, доказательство утверждения состоит в установлении равенств $D=D''$ и $d=d''$.

Докажем, что D'' не больше D, а d'' не меньше d. Каждая цепочка из любого из множеств M' может быть дополнена справа до некоторой цепочки из M. Следовательно, M'' получается из M путем выделения подмножества и усечения цепочек справа. По утверждениям 2 и 3 пункта 2.5, D'' не больше D и, если $D''=D$, то d'' не меньше d. Однако, как будет показано ниже, действительно $D''=D$, следовательно, d'' не меньше d.

Обратно, докажем, что D не больше D'', а d не меньше d''. Рассмотрим непосредственно соседние с W в M слова W_1 и W_2, позиции D_1 и D_2, и букву d_2, участвующие в определении D и d из пункта 2.5. Рассмотрим любое разложение слова W_1 на подцепочки, принадлежащие $M_1, ..., M_n$. Пусть первые k-1 членов разложения образуют начальную подцепочку слова W, а первые k членов не образуют такой подцепочки. В этом случае позиция несовпадения D_1 расположена внутри k-го члена разложения. Рассмотрим слово W'_1, полученное из W_1 отбрасыванием всех членов разложения, начиная с k+1. Поскольку W_1 не больше W, то W'_1 не больше W, и позиция несовпадения с ним $D'_1=D_1$. Заметим, что, поскольку все начальные члены разложения W'_1 образуют начальную подцепочку W, то W'_1 принадлежит некоторому из множеств вида M'. Аналогично рассматривается W'_2, $W'_2 > W$, W_2 принадлежит некоторому множеству вида M', позиция несовпадения с ним $D'_2=D_2$, и буква $d'_2=d_2$.

Рассмотрим множество $Q = \{W'_1, W'_2\}$. Поскольку W'_1 не больше W и $W'_2 > W$, слово W расположено между W'_1 и W'_2. Как показано выше, относительно Q верны те же самые значения $D'_1=D_1$, $D'_2=D_2$ и $d'_2=d_2$, а следовательно, D и d. Поскольку как W'_1, так и W'_2 принадлежат некоторым множествам вида M', то Q есть подмножество M'' и, по утверждению 2 пункта 2.5, D не больше D'', и если $D=D''$, то d не меньше d''. Таким образом, показано, что $D=D''$, а следовательно $d=d''$. Утверждение 1 доказано.

Рассмотренная выше модель представления словоформы как элемента произведения словаря и списков морфов предполагает существование только одной формальной части речи. Рассмотрим ситуацию, когда имеется несколько формальных частей речи, введенных в пункте 3.2.8. Множество правильных словоформ одной части речи p содержится в произведении M^p соответствующего подмножества словаря и соответствующих списков морфов. Множество всех словоформ содержится в объединении множеств M^p, которое и используется для определения ослабленных значений D и d. Утверждение 1 обобщается на случай объединения произведений множеств. Из утверждения 4 пункта 2.5 непосредственно следует, что в качестве множеств вида M' достаточно взять все такие множества, построенные по каждому из M^p в отдельности. Но именно такие множества и просматриваются алгоритмом морфологического анализа из пункта 3.4.1.

Другой не рассматривавшейся до сих пор особенностью изложенной в главе 2 модели морфологической системы являются введенные в пункте 3.2.5 фиктивные позиции морфного разбора словоформ. Алгоритм морфологического анализа при разборе текста слева направо руководствуется в каждый момент текущим состоянием списка номеров строк таблиц морфов. На шаге 3 алгоритма пункта 3.4.1 при обнаружении в данном списке специального значения соответствующий список морфов не просматривается. С учетом данного механизма введенное выше понятие произведения множеств не

может быть использовано при описании множества допустимых словоформ. Рассмотрим необходимое обобщение данного понятия.

Обобщенным произведением M множеств $M_1, M_2, ..., M_n$ называется множество всех буквенных цепочек, представимых в виде конкатенации подцепочек $w_1w_2...w_n$ так, что каждая подцепочка w_k является либо элементом M_k, либо пустой цепочкой, в зависимости от значения некоторого $(k$-1)-местного предиката $P_k(w_1, ..., w_{k-1})$. Если значение $P_k(w_1, ..., w_{k-1})$ есть *истина*, то w_k может пробегать все M_k, в противном случае w_k есть пустая цепочка, независимо от наличия пустой цепочки в M_k.

Заметим, что данное выше определение произведения множеств является частным случаем обобщенного произведения, при всех $P_k(...)$, являющихся тождественно истинными. Содержательно это соответствует случаю, когда в списках морфов отсутствуют введенные в пункте 3.2.5 специальные номера морфем.

Все предыдущие рассуждения почти дословно, с некоторыми очевидными модификациями, переносятся на случай обобщенного произведения множеств. В частности, как формулировка, так и доказательство утверждения 1 переносятся на случай обобщенного произведения, причем под разложением цепочки всюду понимается разложение в смысле обобщенного произведения. Предикаты $P_k(...)$ определяются с помощью построения списка номеров строк таблиц морфов, как это рассмотрено в пункте 3.4.1. Следовательно, алгоритм морфологического анализа действительно просматривает все множества вида M' из формулировки утверждения 1.

В заключение рассмотрим вопрос определения соответствующих значений, обозначим их D' и d', относительно словаря основ. Как показано в пункте 2.5, значение D' являются максимальным среди значений, определяемых процедурами просмотра блоков словаря. В настоящем пункте показано, что значение D являются максимальным среди значений, определяемых процедурами просмотра списков морфов, и значения D'. Следовательно, D есть максимум среди значений, определяемых всеми процедурами просмотра списков, как списков морфов, так и блоков словаря. Рассмотренная в данном пункте модификация алгоритма 1 из пункта 2.5 вычисляет именно такой максимум. Аналогично доказывается правильность определяемого алгоритмом значения d.

4.4. Исправление опечаток разных типов

Рассматривается проблема исправления различных типов, в частности, всех типов однобуквенных опечаток и некоторых типов неоднобуквенных опечаток. Под однобуквенной опечаткой понимается опечатка типа замены, пропуска, вставки буквы и перестановки двух соседних букв [42]. Рассматриваемое множество неоднобуквенных опечаток конкретно не фиксируется. Предполагается только, что для любой цепочки число возможных априорных гипотез исправления в ней неоднобуквенных опечаток значительно меньше числа априорных гипотез исправления однобуквенных опечаток. Неоднобуквенные опечатки предполагаются также локальными, см. пункт 4.1.1. Например, в [42] рассматривается несколько типов часто встречающихся неоднобуквенных опечаток, таких как перестановка гласных букв через согласную или согласных через гласную: **очепатка*.

Предлагается следующая модификация алгоритмов 1 и 2 из пунктов 4.2.1 и 4.2.2. Заметим сначала, что для работы алгоритмов 1 и 2 не требуется, чтобы гипотезы отличались ровно одной буквой от исходного слова. В алгоритмах 1 и 2 первым действием шага 3 является порождение гипотезы, отличающейся от исходного слова по текущей варьируемой позиции V на выбранную букву для замены v. Порожденная гипотеза не обязательно должна совпадать с исходным словом по позициям, которые больше V.

Предлагаемая модификация алгоритма состоит в том, что кроме замены буквы в исходной цепочке на букву v проверяется также гипотеза, в которой производится вставка буквы v, что сразу делает возможным исправление опечаток типа пропуска буквы. Если буква исходного слова на позиции V+1 совпадает с v, проверяются гипотезы перестановки и наличия лишней буквы. Проверяются также любые гипотезы с локальным неоднобуквенным исправлением, которое затрагивает только позиции слова начиная с V и дает на позиции V букву v.

Алгоритм 3, являющийся модификацией алгоритма 2, иллюстрирует параллельное исправление опечаток разных типов. Основными требованиями, предъявляемыми к порядку перебора гипотез, остаются следующие: алфавитный порядок перебора гипотез для одной варьируемой позиции V и перебор варьируемых позиций V от конца слова к началу. Гипотезой для данной позиции V и буквы v может быть выбрана любая буквенная цепочка, совпадающая с исходным словом по позициям,

которые меньше V, и имеющая на позиции V букву v. Практически формируются гипотезы, соответствующие выбранному набору типов исправляемых опечаток.

Простым типом опечаток называется такой, для которого по данной цепочке порождается мало гипотез, например, число гипотез имеет порядок длины слова. Приводятся лишь отличия модифицированного алгоритма от алгоритма 2. По сравнению с алгоритмом 2 в алгоритме 3 добавлены действия на шаге 2 и изменен вид шага 3. Доказательство правильности выбора значений D и d на шаге 3 приводится ниже.

Алгоритм 3.

Шаг 1. Начальная варьируемая позиция V выбирается, как в алгоритме 2.

Шаг 2. Очередная варьируемая позиция V выбирается, как в алгоритме 2. После того, как она выбрана, составляется список простых гипотез для этой позиции: из слова удаляется одна буква, переставляются соседние буквы, а также порождаются неоднобуквенные гипотезы всех необходимых типов, совпадающие с исходной цепочкой в точности по первым V буквам. В среднем такой список содержит две-три гипотезы.

Шаг 3. Фиксируется буква для замены v. Порождаются две гипотезы путем замены буквы позиции V на v и вставки буквы v на позицию V. Из списка, составленного на шаге 2, выбираются гипотезы, имеющие на позиции V данную букву v. Затем гипотезы проверяются в алфавитном порядке. В качестве новых значений D и d выбираются любые из значений, полученных при проверке гипотез. Затем производятся остальные действия шага 3 алгоритма 2.

Шаг 4. Выбираются следующее значение v или V и производится переход к шагу 2 или шагу 3 как в алгоритме 2.

Алгоритм 3 требует дополнительной памяти для хранения списка простых гипотез. Размер дополнительной памяти в среднем составляет несколько более $3N$, где N — минимум из длины предъявленного слова и длины самого длинного слова словаря. Обычно в списке бывает две гипотезы, поскольку неоднобуквенные гипотезы встречаются нечасто.

Однако, если все необходимые гипотезы вычисляются на шаге 3 каждый раз заново, дополнительной памяти не требуется. Поскольку сложность такого вычисления невелика, подобная модификация алгоритма может быть предпочтительной.

Количество гипотез, подлежащих словарной проверке, всего лишь примерно вдвое превышает аналогичный показатель для алгоритма 2. Действительно, вместе с каждой гипотезой типа замены буквы теперь проверяется и гипотеза об ошибке типа пропуска буквы. Два других типа гипотез — перестановка и вставка — дают дополнительно не более $(2N\text{-}1)$ проверок, что составляет небольшую часть от общего их числа. Неоднобуквенные гипотезы также не увеличивают существенно числа проверок, поскольку их число также пропорционально N.

Как показывают эксперименты, число обращений к дисковой памяти увеличивается менее чем вдвое по сравнению с алгоритмом 2.

Доказательство правильности выбора значений D и d на шаге 3 алгоритма 3. Алгоритм 3 по сути сводится к синхронному выполнению действий алгоритма 2 для нескольких серий гипотез, соответствующих каждая определенному типу искажений, например, серия гипотез пропуска буквы. Необходимо выбрать значения D и d для выбора v на следующей итерации алгоритма независимо для каждой из серий. Покажем, что для всех проверяемых на одном шаге алгоритма 3 гипотез эти значения совпадают или различаются несущественно. Все проверяемые на одном шаге цепочки имеют общий начальный отрезок длины V, т.е. по букву v включительно. Пусть D' есть значение D для одной из таких цепочек. Ввиду экстремальных свойств значений D и d, установленных в пункте 2.5, если D' не превышает V, то значения D и d для всех таких цепочек совпадают; если $D' > V$, то значение D для всех таких цепочек превышает V. Однако действия алгоритма 2 в случае $D > V$ не зависят от конкретных значений D и d. Доказательство закончено. Все приведенные рассуждения непосредственно переносятся на случай модификации алгоритма 1, а не алгоритма 2.

4.5. Исправление неодиночных опечаток

Алгоритмы 1 — 3 предназначены для исправления одиночных локальных опечаток. Следующая их модификация дает соответствующий алгоритм исправления неодиночных изолированных опечаток. В любом из алгоритмов 1 — 3, на шаге 3, кроме проверки гипотезы по словарю необходимо рекурсивно применить данный алгоритм к проверяемой гипотезе. Однако во

внутреннем относительно рекурсии экземпляре алгоритма не производится варьирование букв на позициях, меньших текущей варьируемой позиции V вышестоящего экземпляра алгоритма плюс некоторый заранее фиксированный отступ. Рассмотрим предлагаемый алгоритм подробнее.

Фиксируется желаемое расстояние t между одиночными опечатками в слове с изолированными опечатками. Например, при t=4 опечатки в слове *к*а*мпьют*о*р* считаются изолированными, а в слове *к*а*мпь*б*тер* — нет: одиночные опечатки расположены слишком близко. При определении расстояния между одиночными опечатками принимается во внимание длина s участка, затрагиваемого локальной опечаткой, в сформированной гипотезе. Например, для гипотезы лишней буквы s=0, поскольку лишняя буква удалена; для гипотезы пропуска буквы s=1, поскольку одна буква вставлена; для перестановки соседних букв s=2; для перестановки гласной через согласную s=3.

Предлагаемая модификация алгоритмов 1, 2 или 3 является рекурсивным алгоритмом. Параметром рекурсии является увеличенная на единицу длина начального участка слова, опечатки в котором алгоритмом не рассматриваются, обозначим данный параметр V_0. Алгоритм начинает работу с V_0=1. Даются только отличия алгоритма от алгоритмов 1 и 2.

Алгоритм4

Шаг 1. Если алгоритм является модификацией алгоритма 1, в качестве начального значения V выбирается V_0. Если алгоритм является модификацией алгоритма 2, шаг 1 выполняется как в исходном алгоритме.

Шаг 2. Шаг 2 выполняется как в исходном алгоритме.

Шаг 3. Шаг 3 выполняется как в исходном алгоритме. Однако, для каждой проверяемой гипотезы, если определенное для данной гипотезы в процессе проверки по словарю значение D не меньше $V+s+t$, то производится попытка исправления в ней другой опечатки с помощью рекурсивного применения данного алгоритма с параметром $V_0=V+s+t$. Все используемые в алгоритме величины, например, V, v, D, d, являются локальными для данного экземпляра алгоритма и не изменяются после рекурсивного применения другого экземпляра.

Шаг 4. Шаг 2 выполняется как в исходном алгоритме. Однако, если при уменьшении V получено значение $V=V_0$, экземпляр алгоритма заканчивает работу.

Обоснование выбора V_0 на шаге 3 состоит в следующем. Попытка исправления опечаток на позициях i и j эквивалентна попытке исправления опечаток на позициях j и i. Следовательно, достаточно производить попытки исправления опечаток на позициях, образующих неубывающую последовательность. Более того, в данном случае такая последовательность является возрастающей, и именно в этом состоит эффективность алгоритма. Действительно, наиболее долгой операцией является замена букв в коротком начальном участке слова, а в данном случае число таких замен не больше, чем в исходном алгоритме.

Эффективность работы алгоритма 4 практически не отличается от эффективности работы соответствующего исходного алгоритма. Фактически число гипотез, проверяемых алгоритмом на внутренних уровнях рекурсии, невелико, а глубина рекурсии крайне мала. Действительно, на глубоких уровнях рекурсии алгоритм обрабатывает множество слов со значительной общей начальной частью, а именно, общей частью, по длине превышающей t в несколько раз. Другими словами, алгоритм осуществляет перебор, управляемый словарем, являющимся таким подмножеством основного словаря, которое состоит из слов с заданной начальной частью значительной длины. Однако в реальном словаре такие подмножества крайне невелики. Поскольку число гипотез, проверяемых алгоритмами 1, 2 и 3, по порядку не превышает число слов в словаре, оно также невелико. Кроме того, большие значения D встречаются крайне редко.

Число обращений к дисковой памяти, производимых алгоритмом 4, почти всегда совпадает с числом обращений, производимых исходным алгоритмом. Действительно, все проверяемые алгоритмом 4 на внутренних уровнях рекурсии гипотезы имеют существенную общую начальную часть с одной из гипотез, проверяемых исходным алгоритмом, и их алфавитное место почти всегда расположено в том же блоке словаря. Поскольку они проверяются одновременно с исходной гипотезой, лишнего обращения к дисковой памяти почти никогда не производится. Рассмотрение алгоритма исправления изолированных опечаток закончено.

Локальные опечатки называются непересекающимися, если затрагиваемые ими участки слова не пересекаются. Алгоритм 4 исправляет такие опечатки при t=0, однако показатели эффективности такого процесса — число проверяемых гипотез и число обращений к дисковой памяти — могут быть

хуже аналогичных показателей для алгоритма, исправляющего одиночные ошибки. Кроме того, при t=0 невозможна проверка гипотез о наличии лишней буквы, поскольку для этого случая s=0 и алгоритм никогда не заканчивает работу, ср.: *$опечаааааааааатка$. Ниже дается более подходящий для случая t=0 вариант алгоритма.

Следующее обобщение алгоритмов 1 — 4 дает алгоритм исправления не более чем n непересекающихся или изолированных, в зависимости от значения t, элементарных опечаток в слове. Здесь n — некоторое фиксированное число. Параметром рекурсии, кроме V_0, является глубина рекурсии r. Алгоритм начинает работу при r=1. На шаге 3 производится рекурсивное обращение к данному алгоритму с r=r+1. Однако рекурсивное обращение производится только в том случае, если $r < n$. Заметим, что если n=1, данный алгоритм сводится к одному из алгоритмов 1 — 3, а если n есть большое число, данный алгоритм сводится к алгоритму 4.

Наконец, требование того, чтобы опечатки были непересекающимися, может быть снято путем удаления слагаемого s из выражения V_0=V+s+t на шаге 3 рассмотренных алгоритмов. При t=0 такой алгоритм исправляет n элементарных опечаток в слове независимо от их взаимного расположения. Однако в данном случае необходимы следующие изменения логики работы алгоритма.

Назовем замкнутым класс преобразований со следующим свойством: попытки повторного исправления ошибок данного класса на той же позиции в множестве гипотез, порожденных по некоторой цепочке W, не приводят к появлению новых гипотез. Замкнутыми являются классы ошибок типа замены буквы и перестановки букв. Действительно, повторная замена буквы на некоторой позиции в цепочке W приводит лишь к одной из гипотез одиночной замены. Под исправление ошибки на некоторой позиции V понимается преобразование, не затрагивающее начальный участок длины V-1 цепочки W и по определению затрагивающее букву на позиции V.

Назовем класс F_1 преобразований компенсирующим некоторый другой класс F_2, если имеет место следующее свойство: попытки исправления ошибок класса F_1 в множестве гипотез исправления ошибок класса F_2 на той же позиции в цепочке W не приводят к новым гипотезам по сравнению с множеством гипотез исправления в W одиночных ошибок всех элементарных классов, исправляемых алгоритмом, на всех позициях.

Взаимно компенсирующими являются, например, классы вставки и пропуска буквы. Действительно, при вставке гипотетически пропущенной буквы в гипотезу, полученную из W путем удаления из W гипотетически лишней буквы получается одна из гипотез типа замены буквы. Классы пропуска и перестановки двух соседних букв также являются взаимно компенсирующими. Класс наличия лишней буквы является компенсирующим для класса перестановки двух соседних букв, поскольку перестановка и удаление буквы эквивалентны удалению буквы на следующей позиции. Напротив, перестановка букв не компенсирует класс наличия лишней буквы. Например: *$опечакта$ ⟶ *$опечакта$ ⟶ *$опечатка$. Последняя гипотеза не сводима ни к одному из элементарных классов. Замкнутые классы компенсируют сами себя.

Предлагаемое изменение логики работы алгоритма для исправления многократных пересекающихся опечаток состоит в следующем. Параметром рекурсии является, кроме V_0 и r, еще и класс F гипотезы, являющейся входной цепочкой данного алгоритма. Так, при рекурсивном выполнении на шаге 3 исправления опечатки в гипотезе, полученной из входного слова W путем перестановки соседних букв, в качестве параметра рекурсии передается F=«перестановка». При $r > 1$ и V=V_0 на шагах 2 и 3 алгоритма порождаются гипотезы только тех классов, которые не компенсируют F. Таким образом избегаются попытки, например, дважды заменить букву на данной позиции. Очевидно, при $n > 2$ данный прием нуждается в некоторой модификации, однако практическая ценность алгоритма при $n > 2$ невелика ввиду низкой точности поиска вариантов исправления.

При программной реализации алгоритма проверка того, компенсирует ли данный класс преобразований класс F, производится по 1-битовой квадратной матрице совместимости классов. Пусть, например, алгоритм исправляет четыре элементарных класса опечаток — замена, вставка, пропуск одной буквы и перестановка соседних букв. Тогда для хранения матрицы совместимости достаточно одной 16-битной ячейки памяти ЭВМ. Рассмотрение алгоритма исправления не более чем n возможно пересекающихся опечаток закончено.

В заключение приведем вариант последнего из рассмотренных алгоритмов, исправляющий не более чем n-кратные изолированные опечатки. Например, цепочка *$апечаекта$ содержит три опечатки, в том числе двукратную изолированную. Опечатки подобного типа встречаются на практике, хотя и нечасто. При t=0 фиксируется константа t'. В рассмотренном выше алгоритме условием продолжения рекурсии было $r < n$. Предлагаемая модификация состоит в том, что условием продолжения рекурсии

является выражение ($V > V_0+t'$ или $r < n$). При $t'=0$ или большом значении t' данный алгоритм сводится соответственно к одному из рассмотренных выше.

4.6. Экспериментальная проверка эффективности алгоритма исправления опечаток

Экспериментальная проверка алгоритма 3 проводилась с использованием морфологического анализатора, рассмотренного в главе 3, с алгоритмом доступа к словарю, рассмотренным в главе 2. Использовался морфологический словарь, составленный на основе словаря А.А.Зализняка [69] и некоторых дополнительных источников, содержащий более 120 тысяч лексем. Проверка производилась на массиве реально встретившихся в текстах опечаток, предоставленном автору для этой цели проф. И.А.Большаковым. Данный массив является в точности тем массивом, для которого приводится статистика в [42], что позволяет сравнить эффективность предлагаемого подхода с методами, рассмотренными в [42].

В контрольном массиве ошибочных слов было 507 слов. Из них для 482 слов был найден правильный вариант исправления. Правильность варианта устанавливалась человеком-экспертом на основе анализа контекста. Для 21 слова не было найдено ни одного варианта исправления — это неоднобуквенные опечатки типа **выячвлялясь* вместо *выявлялась*. В семи случаях ошибка в цепочке не была обнаружена — например, в слове *исковых* вместо *искомых*. В четырех случаях среди предложенных вариантов не было правильного — это также неоднобуквенные ошибки типа **коет* вместо *койот*.

В таблице 1 показаны средние значения и медианы распределения следующих параметров: числа проверенных по словарю гипотез, числа считанных с дискового устройства блоков словаря и числа найденных вариантов исправления. Каждый параметр подсчитывался до выдачи программой первого найденного варианта, до выдачи правильного варианта, до выдачи последнего из найденных вариантов, и до конца поиска, т.е. до момента проверки последней из гипотез. Первое число в каждой паре представляет собой медиану соответствующей величины (см. пункт 4.2.3), второе — среднее значение. К предмету настоящего пункта имеет отношение первая строка каждого раздела таблицы; величины, приведенные во второй строке, обсуждаются в пункте 4.8.

	до первого варианта		до правильн варианта		до последн. варианта		до конца поиска	
	мед	среднее	мед	среднее	мед	среднее	мед	средн
Число гипотез	16 15	26.65 25.37	19 18	33.25 30.76	26 26	47.97 42.24	215 149	215 150
Обращен. к диску	1 1	6.61 5.69	2 2	9.21 7.64	2 2	16.78 13.17	118 76	120 77
Вариа=ты			1	1.40	1	2.15		

Табл. 1. Медианы и средние значения показателей

В [42] приводится только один из наших показателей — число гипотез, испытанных до получения правильного варианта. Наилучшие достигнутые в [42] значения — медиана 23 и среднее 49.5. Как видно из таблицы, предлагаемый алгоритм дает несколько лучшие показатели: 19 и 33.25 соответственно. Однако основным показателем эффективности предлагаемого метода является число обращений к дисковому устройству.

Как видно из таблицы, для нахождения всех вариантов по медиане достаточно *двух* обращений к диску, а 118 обращений нужно для проверки всех гипотез, т.е. для установления того, что других вариантов исправления нет. Столь резкая асимметрия относительно среднего момента выдачи вариантов является одной из целей, преследовавшихся при разработке предлагаемых алгоритмов.

4.7. Ранжирование вариантов

В отличие от подхода, рассмотренного в [42], предлагаемые алгоритмы выдают варианты исправления без ранжирования вариантов их по вероятности быть правильным. Поскольку предлагаемый алгоритм находит правильный вариант в среднем быстрее, чем алгоритм, изложенный в [42], представляется нецелесообразным существенно изменять предлагаемый алгоритм в целях ранжирования вариантов в ущерб быстродействию алгоритма в целом. Поскольку среднее число вариантов равно приблизительно двум, и они выдаются алгоритмом с крайне небольшой задержкой, такое ранжирование может быть проведено апостериорно, на основании методов, рассмотренных в [42].

В проведенных автором экспериментах по апостериорному ранжированию вариантов использовался упомянутый в пункте 4.6 массив реально встретившихся в текстах опечаток, предоставленный автору проф. Большаковым. Ранжирование производилось на основании методов, изложенных в [42, 45, 95], а также разработанных автором.

Подсчитывались показатели только для тех слов, для которых было найдено более одного варианта. Без ранжирования в среднем порядковый номер правильного варианта был 1.80, и лишь в 43.3% случаев правильный вариант выдавался первым и в 34.6% — вторым. После ранжирования средний номер правильного варианта составил 1.37, правильный вариант был первым в 78.0%, вторым в 16.6% и третьим в 2.0% случаев.

Подсчитывались также показатели для всего исследованного массива слов, т.е. как для слов, для которых было найдено несколько вариантов исправления и проведено их ранжирование, так и для слов, для которых был найден только один вариант исправления. Правильный вариант выдавался первым в 83.3% случаев до ранжирования и в 93.3% — после.

Апостериорное ранжирование вариантов обладает тем недостатком, что при интерактивной работе время ожидания выдачи вариантов в этом случае равно полному времени работы алгоритма. Предположим, что реальное время работы алгоритма определяется количеством обращений к дисковой памяти. Как видно из Табл. 1, при апостериорном ранжировании вариантов время ожидания правильного варианта увеличивается в 60 раз. Поэтому при интерактивной работе алгоритма применяется следующий прием.

Проводится почти полное исправление опечаток в слове. Другими словами, после проверки достаточного числа гипотез, превышающего среднее число гипотез, проверяемых до получения последнего варианта, работа алгоритма приостанавливается. Из Табл. 1 видно, что таким пороговым значением числа гипотез для нашего словаря может служить число 50. Проводится апостериорное ранжирование найденных вариантов, после чего варианты выдаются пользователю. Затем работа алгоритма исправления опечаток возобновляется. Найденные варианты выдаются пользователю по мере их обнаружения. Однако как правило на данном втором этапе алгоритм не обнаруживает ни одного нового варианта.

Поскольку среднее время ожидания последнего варианта незначительно отличается от среднего времени ожидания первого варианта, при рассмотренной схеме ранжирования пользователю выдаются почти всегда ранжированные варианты с задержкой, незначительно большей, чем без ранжирования.

4.8. Бессловарный отсев гипотез

Обратимся к задаче улучшения показателей, рассчитанных до завершения поиска. Ситуация парадоксальна: обычно все варианты исправления бывают получены практически сразу, а основное время тратится на установление того, что других вариантов нет.

Как показано в пункте 4.6, показатели, связанные с получением первого, правильного и последнего варианта исправления, не могут быть существенно улучшены с помощью каких-либо дополнительных методов отсева гипотез. Однако стратегия выбора следующей буквы *v* в алгоритмах 1 и 2, эффективная при работе с большей частью слова, является неэффективной при варьировании первых двух-трех букв слова. Действительно, процедура анализа в этом случае почти всегда возвращает значение *d*, равное текущей букве *v*, и в качестве следующей буквы берется просто следующая по алфавиту за *v* буква, то есть априорного отсева гипотез не происходит. Вместе с тем, на фазу варьирования первых двух-трех букв слова приходится основная часть затрат общего времени работы алгоритма 2: каждая новая буква из первых двух-трех букв слова, как правило, требует считывания нового блока словаря.

Таким образом, для сокращения общей продолжительности работы алгоритма необходимо априорно отсеять часть гипотез, варьирующих первые буквы слова. Для этой цели могут быть применены бессловарные методы отсева, такие, как диграммный и триграммный контроль [42].

Модификация алгоритмов 1 и 2 состоит в том, что на шаге 3, перед обращением к процедуре анализа, гипотеза подвергается три- или диграммному контролю; такой контроль для краткости далее называется *n*-граммным. Если в гипотезе найдено недопустимое сочетание букв, обращения к процедуре анализа не происходит, а в качестве следующей буквы *v* берется следующая по алфавиту буква.

Однако такой отсев целесообразнее применять не на всех этапах работы алгоритма, а только на этапе варьирования первых двух-трех букв слова. Во-первых, получение массива *n*-грамм по крупному морфологическому словарю требует синтеза всех словоформ. Гораздо проще собрать статистику *n*-грамм по первым буквам основ, хотя синтезировать словоформы с короткими основами при этом все же требуется. Для формирования массива триграмм по первым трем буквам словоформ применявшегося автором словаря (см. пункт 4.7) необходимость в синтезе словоформ возникла для 0.3% лексем. Во-вторых, что более важно, массивы триграмм и особенно диграмм, собранные только по первым буквам слов, получаются более разреженными, и отсев по ним происходит более интенсивно.

Однако преимуществом метода *n*-граммного контроля всего слова перед *n*-граммным контролем только первых букв слова является возможность в некоторых случаях сузить интервал поиска опечатки, что позволяет в таких случаях полностью избежать варьирования первых букв слова. По данным, приведенным в [42], диграммный контроль позволяет сузить интервал поиска в 11% случаев. Лучшие результаты дает двойной контроль: на всем протяжении слова — по *n*-граммам, собранным по всей длине слов словаря, и на начальном участке слова — по *n*-граммам, собранным по первым буквам слов словаря. Однако для хранения двух массивов *n*-грамм требуется дополнительный объем оперативной памяти.

Автором проводились эксперименты с триграммным контролем первых букв слова. Результаты даны в Табл. 1 во второй строке каждого раздела. Как и ожидалось, существенно изменились только показатели, рассчитанные до конца поиска, однако в этом случае улучшение значительно.

ВЫВОДЫ

1. Предложены алгоритмы поиска вариантов исправления опечаток в слове, взятом вне контекста, с использованием морфологического словаря. Перебор проверяемых гипотез исправления ошибки в слове управляется морфологическим словарем, что значительно снижает общее число проверяемых гипотез. Предложенные алгоритмы по эффективности превосходят известные. Так, число гипотез, испытываемых до получения правильного варианта исправления, в среднем в 2.12 раза меньше, чем в [42].
2. Впервые предложен принцип упорядочения процесса перебора альтернатив, заключающийся в проверке в первую очередь тех гипотез, проверка которых может быть произведена наиболее быстро.
3. Варианты алгоритма исправления опечаток предложены, в частности, для решения следующих задач: исправление одиночных однобуквенных опечаток и неоднобуквенных опечаток некоторого (произвольного) класса; исправление неодиночных изолированных опечаток; исправление не более чем *n* элементарных опечаток в слове; исправление изолированных *n*-кратных опечаток, и др.
4. Алгоритмы осуществляют полный, то есть исчерпывающий, поиск вариантов исправления ошибок рассматриваемых классов. Предложен также вариант каждого алгоритма, осуществляющий почти полный поиск вариантов. Общее время работы алгоритма в этом случае сокращается примерно в 60 раз.
5. Предложенные алгоритмы минимизируют одновременно следующие восемь показателей эффективности: число проверяемых гипотез и число обращений к дисковой памяти, производимых алгоритмом до нахождения первого, правильного, последнего вариантов исправления и до завершения процесса поиска вариантов, то есть до момента проверки последней из гипотез.
6. Алгоритмы минимизируют показатели, существенные не только при пакетной, но и при интерактивной работе. Так, специально минимизируется время ожидания ответа системы и время получения правильного варианта исправления. Вместе с тем минимизируется общее время работы алгоритма. Таким образом, особенностью алгоритма является получение, как правило, всех вариантов исправления на самой ранней стадии работы алгоритма, составляющей около 2% общего времени работы.

7. Предложены модификации алгоритмов, выдающие найденные варианты исправления ранжированными или почти всегда ранжированными по вероятности быть правильным.
8. Алгоритмы не опираются ни на какие дополнительные массивы информации, кроме морфологического словаря, а также не используют никакие эмпирические данные или эвристики. Исключением являются некоторые предложенные модификации алгоритмов, использующие массивы три- или диграмм.
9. Для работы алгоритмов не требуется хранения каких-либо массивов информации в оперативной памяти ЭВМ. Объем программного кода, реализующего предложенные алгоритмы, невелик.

ЗАКЛЮЧЕНИЕ

В итоге работы над темой диссертации решена задача разрабо-тки принципов, моделей, методов и алгоритмов, составляющих в совокуп-ности универсальную, языковонезависимую в некотором клас-се языков, быстродействующую, экономичную по расходу оперативной памяти ЭВМ морфологическую систему, удовлетворяющую ряду допол-нительных требований. В процессе исследования получены следующие научные результаты:

1. Разработана модель морфологического строения флективного естественного языка, эффективно реализуемая на современных ЭВМ. Под эффективностью реализации понимается небольшая потребность в оперативной памяти в сочетании с высоким быстродействием. Модель является языково-независимой в некотором классе языков, включающем русский, и отвечает определенной совокупности предъявленных требований.
2. На основании предложенной модели в комплексе и с единых позиций разработаны и программно реализованы высокоэффективные алгоритмы решения задач точного и полного морфологического анализа и синтеза, а также ряда смежных задач, таких как нормализация слов, обнаружение, интерпретация и исправление ошибок и опечаток, приближенный анализ новых слов. Указанные алгоритмы и программы в совокупности составляют универсальную инструментальную языковонезависимую в некотором классе языков морфологическую систему.
3. При участии автора созданы система морфологических таблиц для русского языка в рамках предложенной модели, а также машинный морфологический словарь русского языка (на базе известных источников), включающий около 130 тыс. лексем общеупотребительной и специальной лексики.
4. Применительно к задаче морфологического анализа и синтеза предложена структура базы данных, ориентированной на высокоэффективное решение следующей задачи: найти все записи, ключи которых являются начальными подцепочками предъявленной не ограниченной справа буквенной цепочки. Для нахождения всего искомого множества записей достаточно одного обращения к дисковой памяти, что является пределом возможного быстродействия для крупного словаря при ограничении занимаемого объема оперативной памяти. Разработаны однопроходные алгоритмы формирования и распаковки такой базы данных.
5. Предложены алгоритмы поиска вариантов исправления опечаток в слове, взятом вне контекста, с помощью перебора, управляемого морфологическим словарем. При этом минимизируются совместно следующие показатели: время нахождения первого, правильного, последнего вариантов исправления и время завершения процесса поиска. Особенностью алгоритмов является получение, как правило, всех вариантов исправления на самой ранней стадии работы алгоритма, составляющей около 2% общего времени работы. Экспериментально показано, что по эффективности разработанные алгоритмы превосходят известные.
6. Варианты высокоэффективного алгоритма исправления опечаток предложены, в частности, для решения задачи исправления одиночных однобуквенных опечаток и неоднобуквенных опечаток некоторого (произвольного) класса, а также задачи исправления различных типов неодиночных опечаток.
7. Созданы программные средства, реализующие перечисленные выше алгоритмы. Программы составлены в основном на языке Си++ и содержат более 20 000 строк исходного текста.

Основные результаты диссертации отражены в публикациях [53, 54, 55, 56, 57, 58, 59, 61, 62, 66, 67].

Литература

1. Автоматическое индексирование на базе пакета прикладных программ АИДОС // Методич. материалы и документация по пакетам прикладных программ, М.: МЦНТИ, 1982, вып. 16.
2. Альшванг В.Д. и др. Математическое обеспечение системы автоматического перевода ЭТАП-1 // Прикладные и экспериментальные лингвистические процессоры. Новосибирск: ВЦ СО АН СССР, 1982.
3. Андреевски А., Дебили Ф., Флур К. Об одном важном свойстве лексики естественных языков и его использовании при автоматическом исправлении опечаток // Прикладные и экспериментальные лингвистические процессоры, Новосибирск: ВЦ СО АН СССР, 1982, с. 98-109.
4. Андрющенко В.М. Концепция и архитектура машинного фонда русского языка. Дисс... докт. фил. наук, М.: ВИНИТИ, 1989.
5. Анно Е.И. Система морфологического анализа с синтезом словоформ // Семиотика и информатика, М., 1978, вып. 10.
6. Апресян Ю.Д. и др. Лингвистическое обеспечение системы автоматического перевода ЭТАП-2. М.: Наука, 1989.
7. Апресян Ю.Д. и др. Лингвистическое обеспечение системы французско-русского автоматического перевода ЭТАП-1. II. Французская морфология. Французский комбинаторный словарь: Препр. Ин-та рус. яз. АН СССР (Пробл. группа по эксперим. и прикл. лингвистике) М., 1984, N 154.
8. Бабко-Малая О.Б., Шемраков В.А. Методы и системы автоматизированного обнаружения и коррекции текстовых ошибок. Препринт N 5, Л.: Библиотека АН СССР, 1987, 46 С.
9. Белоногов Г.Г. и др. Автоматизация лингвистической обработки словарей // НТИ, сер. 2, 1983, N 11.
10. Белоногов Г.Г. и др. Автоматизированная обработка научно-технической информации // Итоги науки и техники, сер. Информатика, М.: ВИНИТИ, 1984, т. 8.
11. Белоногов Г.Г. и др. Автоматизированная словарная служба ВИНИТИ // Системные исследования ГАСНТИ: тез. докл. XV Всес. научн. сем., ч. 2, М.: 1985.
12. Белоногов Г.Г. и др. Автоматическая нормализация слов и словосочетаний // НТИ, сер. 2, 1985, N 1.
13. Белоногов Г.Г. и др. Автоматический морфологический анализ «новых» слов // Системные исследования ГАСНТИ: тез. докл. XIV Всес. научн. сем., ч. 2, М.: 1983.
14. Белоногов Г.Г. и др. Автоматическое индексирование документов // Вопр. инф. теории и практики, 1985, N 53, с. 57-61.
15. Белоногов Г.Г. и др. Автоматическое индексирование для диалоговых пакетов прикладных программ // Системные исследования ГАСНТИ: тез. докл. XIV Всес. научн. сем., ч. 2, М.: 1983.
16. Белоногов Г.Г., Зеленков Ю.Г. Алгоритм автоматизированного исправления орфографических ошибок в текстах. М.: ВИНИТИ, 1986, 14 с.
17. Белоногов Г.Г., Зеленков Ю.Г. Алгоритм автоматического обнаружения орфографических ошибок в текстах. М.: ВИНИТИ, 1986, 15 с.
18. Белоногов Г.Г. и др. Алгоритм многоступенчатого морфологического анализа русских слов // НТИ, сер. 2, 1983, N 1, с. 6-10.
19. Белоногов Г.Г. и др. Алгоритм многоступенчатого морфологического анализа текстов // Системные исследования ГАСНТИ: тез. докл. XIII Всес. научн. сем., ч. 2, М.: 1982.
20. Белоногов Г.Г., Зеленков Ю.Г. Алгоритм морфологического анализа русских слов // Вопр. инф. теории и практики. М.: 1985, N 53, с. 62-93.
21. Белоногов Г.Г., Зеленков Ю.Г. Алгоритм формирования машинного словаря словоизменительных основ слов для морфологического анализа. М.: ВИНИТИ, 1986, 16 с.
22. Белоногов Г.Г. Об использовании метода аналогии при автоматической обработке текстовой информации // Проблемы кибернетики, М., 1974, вып. 28.
23. Белоногов Г.Г., Давыдова И.М. О возможности определения грамматических классов слов по буквенным кодам слов // НТИ, сер. 2, 1967, N 8.
24. Белоногов Г.Г. и др. Определение грамматических признаков «новых» слов // Инженерная лингвистика: Ученые записки, ч. 2, — Л.: ЛГПИ им. Герцена, 1971, т. 458.
25. Белоногов Г.Г. и др. Оценка эффективности ИПС с автоматическим индексированием документов // НТИ, сер. 2, 1986, N 8.

26. Белоногов Г.Г. и др. Принципы многоступенчатого морфологического анализа // Интерактивные системы: Тез. докл. IV школы-семинара, Сухуми, 1982.
27. Белоногов Г.Г. и др. Проблемы автоматического обнаружения и исправления ошибок в научно-технических текстах // НТИ, сер. 2, 1982, N 6, с. 29-31.
28. Белоногов Г.Г. и др. Результаты функционирования в ВИНИТИ системы обнаружения орфографических ошибок в режиме опытной эксплуатации // Вопросы информационной теории и практики, 1984, N 51, с. 24-44.
29. Белоногов Г.Г. и др. Словообразовательные классы русских слов // НТИ, сер. 2, 1985, N 12.
30. Белоногов Г.Г. и др. Таблица подстановок для нормализации слов // Вопр. инф. теории и практики, 1985, N 53.
31. Белоногов Г.Г. и др. Экспериментальная система автоматизированного обнаружения и исправления орфографических ошибок в текстах // НТИ, сер. 2, 1984, N 3, с. 2-27.
32. Белоногов Г.Г., Кузнецов Б.А. Языковые средства автоматизированных систем, М.: Наука, 1983.
33. Бидер И.Г. Синтез словоформ с помощью преобразовательных грамматик // НТИ, сер. 2, 1979, N 6.
34. Бидер И.Г., Большаков И.А., Еськова Н.А. Формальная модель русской морфологии // ИРЯ АН СССР, Проблемная группа по экспериментальной и прикладной лингвистике, вып. 112, М., 1978, I — 60 с., II — 59 с.
35. Бидер И.Г., Большаков И.А. Экспериментальная система морфологического синтеза для естественных языков // Материалы VI межреспубликанской школы-семинара «Интерактивные системы», Тбилиси, 1984, с. 5-15.
36. Болошин И.А., Белоногов Г.Г., Кузнецов Б.А., Пащенко Н.А. Состояние и перспективы развития лингвистического обеспечения ГАСНТИ // Системные исследования ГАСНТИ: Тез. докл. XIII Всес. научн. сем., ч. 2, М., 1982.
37. Большаков И.А. Автоматическая проверка правильности дефисных образований // НТИ, сер. 2, 1986, N 2, с. 28-31.
38. Большаков И.А., Емелин Е.В. Алгоритм минимизации графового представления словарей // Известия АН СССР. Техническая кибернетика, 1987, N 4.
39. Большаков И.А. ДИСКОР — диалоговая система коррекции текстов // НТИ, сер. 2, 1986, N 5, с. 8-15.
40. Большаков И.А. ДИСКОР — диалоговая система обнаружения и исправления ошибок в русских и английских текстах // Материалы VI межреспубликанской школы-семинара «Интерактивные системы», Тбилиси, 1984, с. 16-25.
41. Большаков И.А. Количественный анализ методов сжатия крупных машинных морфологических словарей // НТИ, сер. 2, 1990, N 3. с. 28-32.
42. Большаков И.А. Минимизация перебора альтернатив при автоматизированном исправлении искаженных слов // Семиотика и информатика, 1990, вып. 31, с. 124-149.
43. Большаков И.А., Солодовник И.А. Некоторые результаты инвентаризации программных средств ГАСНТИ // НТИ, сер. 1, 1986, N 8.
44. Большаков И.А. О чисто автоматической коррекции текстов с опорой на клавиатурную модель типовых ошибок // НТИ, сер. 2, 1987, N 3.
45. Большаков И.А. Проблемы автоматической коррекции текстов на флективных языках // Итоги науки и техники. Теория вероятностей. Математическая статистика. Техническая кибернетика. т. 28, М.: ВИНИТИ, 1988, с. 111-139.
46. Большаков И.А. Русский морфологический словарь для IBM-совместимых персональных компьютеров // НТИ, сер. 2, 1990, N 1, с. 26-32.
47. Большаков И.А. Упрощенный морфологический анализ при автоматической проверке правильности текстов // НТИ, сер. 2, 1985, N 6, с. 22-28.
48. Большаков И.А. Формальная модель латинской морфологии. Препринт ИРЯ АН СССР, вып. 124, 125. М.: 1979.
49. Братчиков И.Л. Метод обнаружения и исправления искажений в русских словоформах, основанный на функции расстояния // Семиотические аспекты формализации интеллектуальной деятельности. Тезисы докладов и сообщений школы-семинара «Кутаиси-85», М.: ВИНИТИ, 1985, с. 339-395.
50. Варга Д. Проблемы осуществления морфологического анализа при машинном переводе // НТИ, сер. 2, 1964, N 4.

51. Волков В.Н., Иванов А.В. Реализация алгоритма распознавания и выборки слов с использованием функции совпадения // Программирование, 1982, N. 2, с. 90-92.
52. Выходцева Л.Н. Алгоритм морфологического анализа русской назывной фразы // Разработка и совершенствование лингвистического обеспечения информационного поиска: Сб. научн. трудов ГПНТБ СССР, М., 1982.
53. Гельбух А.Ф. Минимизация количества обращений к диску при словарном морфологическом анализе // НТИ, сер. 2, 1991, N 6.
54. Гельбух А.Ф. Минимизация числа гипотез и обращений к дисковой памяти при словарном исправлении опечаток // НТИ, сер. 2, 1993, N 5, с. 23-30.
55. Гельбух А.Ф. Модель морфологии флективного естественного языка // Материалы III Международной конференции «Программное обеспечение ЭВМ», Тверь, НПО Центрпрограммсистем, 1990, с. 27-31.
56. Гельбух А.Ф. Морфологический анализ/синтез и проверка русских текстов // Тезисы 1-й Ежегодной Всесоюзной конференции SUUG, Москва, 1990.
57. Гельбух А.Ф. Простая оболочка системы точного морфологического анализа и синтеза текстов на естественном языке // Использование программных средств ПЭВМ для автоматизации учрежденческой деятельности, АН СССР и НПО ЦПС, Калинин, 1990.
58. Гельбух А.Ф. Пустая языково-независимая оболочка системы точного морфологического анализа и синтеза текстов на естественном языке // Международный форум «Тех-Екс'90 — обмен технологиями», Болгария, Пловдив, 1990.
59. Гельбух А.Ф. Эффективно реализуемая модель морфологии флективного языка // НТИ, сер. 2, 1992, N 1.
60. Гинзбург М.П. Словари для автоматического индексирования (структура и технология) // Вопр. инф. теории и практики, 1979, вып. 9.
61. Добрушина Е.Р., Савина Г.Б., Гельбух А.Ф. Разработка программных и лингвистических средств для создания баз знаний о научных исследованиях и разработках // Отчет ВНТИЦ N 02900 056145, Москва, 1989, 35 с.
62. Добрушина Е.Р., Савина Г.Б., Гельбух А.Ф. Система точного морфологического анализа и синтеза // Программное обеспечение новой информационной технологии, АН СССР и НПО ЦПС, Калинин, 1989.
63. Долгополов А.С. Об автоматическом корректоре текстов // НТИ, сер. 2, 1985, N 3, с. 27-28.
64. Долгополов А.С. Программа автоматической коррекции текстов // НТИ, сер. 2, 1986, N 4, с. 26-29.
65. Домбровский М. и др. Система обработки сложных терминов в тексте // НТИ, сер. 2, 1980, N 12.
66. Загацкий Б.А., Перцов Н.В., Гельбух А.Ф. Лингвистический процессор базы знаний о научных исследованиях и разработках // Отчет ВНТИЦ N 02900 003868, Москва, 1990, 82 с.
67. Загацкий Б.А., Перцов Н.В., Гельбух А.Ф. Система синтаксического анализа фраз русского языка // Международный форум «Тех-Екс'90 — обмен технологиями», Болгария, Пловдив, 1990.
68. Загика Е.А. Алгоритмы лингвистической обработки словарей // Вопр. инф. теории и практики, 1985, N 53, с. 10-25.
69. Зализняк А.А. Грамматический словарь русского языка. Словоизменение. М.: Русский язык, 1987, 878 с.
70. Зализняк А.А. Русское именное словоизменение. М.: Наука, 1967.
71. Зарецкая Е.Н. Автоматический синтез словоформ, чередующих гласную с нулем // Семиотика и информатика, М., 1977, вып. 9.
72. Исхакова Х.Ф. Автоматический синтез форм существительного в татарском языке // НТИ, сер. 2, 1968, N 3.
73. Казакевич О.А. Автоматизация лексикографических работ. Автоматические словари (обзор зарубежных публикаций) // НТИ, сер. 2, 1985, N 9.
74. Казаков Е.А. Исследование процесса построения информационно-поисковых тезаурусов с применением ЭВМ. Дисс... канд. техн. наук, М., 1975.
75. Карбонелл Дж., Хейз Ф. Стратегии преодоления коммуникативных неудач при анализе неграмматичных языковых выражений // Новое в зарубежной лингвистике: вып. XXIV. Компьютерная лингвистика, 1989, с. 48-106.
76. Кнут Д. Искусство программирования для ЭВМ. Том 3. Сортировка и поиск. М:, 1979.

77. Коровина Т.И., Румшинский Б.Л., Семенова В.Э., Цинман Л.Л. Аппарат для описания морфологии флективных языков: Препр. Ин-та рус. яз. АН СССР. (Пробл. группа по эксперим. и прикл. лингвистике), М., 1977, N 91.
78. Коровина Т.И. Синтез русских именных словоформ с помощью ЭВМ // НТИ, сер. 2, 1975, N 3.
79. Котов Р.Г. Машинный фонд русского языка и автоматизация словарно-терминологического обеспечения информационной службы // Машинный фонд русского языка: идеи и суждения, М.: Наука, 1986.
80. Красиков Ю.В. Теория речевых ошибок (на материале ошибок наборщика). — М.: Наука, 1980, 160 с.
81. Кулагина О.С. Исследования по машинному переводу, М.: Наука, 1979.
82. Курбаков К.И. Кодирование и поиск информации в автоматическом словаре. — М.: Советское радио, 1968, 214 с.
83. Лахути Д.Г. и др. Автоматическое индексирование текстов в документальных ИПС // Кибернетическая лингвистика, М.: Наука, 1983.
84. Леонтьева Н.Н. Информационная модель машинного перевода // НТИ, сер. 2, 1985, N 10.
85. Леонтьева Т.М. Организация и использование системы словарей в ИПС // Вопр. разработки механиз. ИПС для ЦСИФ по химии и хим. промышленности, 1970, вып. 25.
86. Мальковский М.Г. Диалог с системой искусственного интеллекта. М.: МГУ, 1985, 216 с.
87. Мальковский М.Г. TULIPS — обучаемый решатель задач, понимающий естественный язык // Труды IV Межд. объедин. конф. по иск. инт., т. 6, М.: Изд. ВИНИТИ, 1975, с. 34-43.
88. Мальковский М.Г., Волкова И.А. Анализатор системы TULIPS-2. Морфологический уровень // Вестн. Моск. Ун-та, сер. 15, 1981, N 1, с. 70-76.
89. Мальковский М.Г. Анализатор программы, понимающей естественный язык // Обработка символьной информации, вып. 2, М.: Изд. ВЦ АН СССР, 1975, с. 138-154.
90. Марчук Ю.Н., Тихомирова Б.Д., Щербинин В.И. Система машинного перевода с английского языка на русский // Машинный перевод и автоматизация информационных процессов, М., 1975.
91. Матвеев С.А., Сотникова Р.А. Система автоматической коррекции ошибок в словосочетаниях // Программирование, 1984, N 5, с. 68-74.
92. Мельчук И.А. Опыт разработки фрагмента системы понятий и терминов для морфологии // Семиотика и информатика, 1975, вып. 6.
93. Мельчук И.А. Опыт теории лингвистических моделей «Смысл <=> текст». М.: Наука, 1974.
94. Новоселов А.П., Хорошилов А.А. Алгоритм автоматической нормализации слов // Вопр. инф. теории и практики, 1985, N 53, с. 26-30.
95. Партыко З.В. Анализ искажений, возникающих при вводе текстов в ИПС «Ассистент» // НТИ, сер. 2, 1982, N 1, с. 21-26.
96. Партыко З.В. Методы машинной корректуры и машинного редактирования. — М.: Книга, 1983, 40 с.
97. Пащенко Н.А., Никульцева И.Г., Яровенко О.И. Составление и использование словарей стоп-слов при автоматической обработке текстов // Вопр. инф. теории и практики, 1985, N 53.
98. Рогожникова Н.П. Машинный фонд русского языка и словарное дело // Машинный фонд русского языка: идеи и суждения, М.: Наука, 1986.
99. Рыбаков Ф.И., Руднев Е.А., Петухов В.А. Автоматическое индексирование на естественном языке, М.: Энергия, 1980.
100. Салмина Н.Ю., Ходашевский И.А. Методы и средства автоматического исправления орфографических ошибок // НТИ, сер. 2, 1986, N 10, с. 25-28.
101. Скаличка В. К вопросу о типологии // ВЯ, 1966, N 4, с. 29-30.
102. Скороходько Э.Ф., Стогний А.А. Некоторые вопросы создания банков терминов: лингвистический аспект // НТИ, сер. 2, 1986, N 8.
103. Сухотин Б.В. Выделение морфем в текстах без пробелов между словами. М.: Наука, 1984.
104. Сучилин В.А. Обнаружение ошибок в научно-технических текстах средствами автоматизированной ИПС // НТИ, сер. 2, 1983, N 2, с. 39.
105. Хорошилов А.А. Автоматическая нормализация слов в системах обработки научно-технической информации. Дисс... канд. техн. наук, М.: ВИНИТИ, 1987, 144 с.
106. Хорошилов А.А., Новоселов А.П. Автоматическая нормализация слов, выражающих сказуемое // Системные исследования ГАСНТИ: Тез. докл. XV Всес. науч. сем., ч. 2, М., 1985.

107. Цинман Л.Л. Язык для записи лингвистической информации в системе автоматического перевода ЭТАП (опыт «практической логики») // Семиотика и информатика, 1986, N 27, с. 82-120.
108. Цинман Л.Л. и др. Алгоритм синтаксического анализа в системе ЭТАП-2: Препр. Ин-та рус. яз. АН СССР (Пробл. группа по эксперим. и прикл. лингвистике) М., 1986, N 174.
109. Шгурман Я.П., Партыко З.В. Анализ искажений при вводе реферативной информации в систему «Ассистент» // НТИ, сер. 2, 1982, N 3, с. 17-31.
110. Alam, Y.S. A two-level morphological analysis of Japenese // Texas Linguistic Forum 22, 1983, p. 229-252.
111. Anderson, S.R. Morphology as a parsing problem // Wallace, K., ed. Morphology as a computational problem. UCLA Occasional Paper no. 7: Working Papers in Morphology, Department of Linguistics, University of California, Los Angeles, 1988, p. 4-21.
112. Angell, R.C., G.E.Freund, P.Willett. Automatic spelling correction using a trigram similarity measure // Inform. Proc. & Manag., 1983, v. 19, no. 4, pp 255-261.
113. Antworth, E.L. PC-KIMMO: a two-level processor for morphological analysis. Occasional Publications in Academic Computing No. 16. Dallas: Summer Institute of Linguistics, 1990, 273 p.
114. Barton, G.E., at al. Computational complexity and natural language. MIT Press, 1987.
115. Bear, J. A morphological recognizer with syntactic and phonological rules // Proceedings of Coling'86, Association for Computational Linguistics, 1986, p. 272-276.
116. Beesley, K.R. Computer analysis of Arabic morphology: a two-level approach with detours. 1989.
117. Blaberg, O. A two-level description of Swedish // Folia Linguistica 14, 1985, p. 43-62.
118. Black, A.W., at al. Formalisms for morphographemic description // ACL Proceedings, Third European Conference, Association for Computational Linguistics, 1987, p. 11-18.
119. Boisen, S. Pro-KIMMO: a Prolog implementation of two-level morphology // Wallace K., ed. Morphology as a computational problem. UCLA Occasional Paper no. 7: Working Papers in Morphology, Department of Linguistics, University of California, Los Angeles, 1988, p. 31-53.
120. Cahill, L.J. Syllable-based morphology // Proceedings of 13th International Conference on Computational Linguistics, 1990, p. 48-53.
121. Calder, J. Paradigmatic Morphology // Proceedings of 4th Conference of European Chapter of the ACL, 1989, p. 58-65.
122. Comer, D., V.Y.Shen. Hash-bucket search: a fast technique for searching an English spelling dictionary // Software — Practice & Experience, 1982, v. 12, p. 669-682.
123. Cooper, W.S. The storage problem // Mech. Translat., 1958, p. 74-83.
124. Dalrymple, M., at al. DKIMMO/TWOL: a development environment for morphological analysis. Stanford, CA: Xerox Paolo Alto Research Center and Center for the Study of Language and Information, 1987.
125. Gajek, O., at al. KIMMO: LISP implementation // Texas Linguistic forum 22, 1983, 187-202.
126. Gorz, G., D.Paulus. A finite state approach to German verb morphology // Proceedings of Coling'88, Association for Computational Linguistics, 1988, p. 212-215.
127. Hall, P.A.V., G.R.Dowling. Approximate string matching // Computing surveys, 1980, v. 12, no.4, p. 381-402.
128. Ito, T., M.Kizawa. Hierarchical file organization and its application to similar-string matching // ACM Trans. of Database Systems, 1982, v. 8, no. 3, p. 410-433.
129. Jappinen, H., at al, eds. Morphological analysis of Finnish word forms. Publications of the Kielikone project, series A, report no. 1, 1986.
130. Jappinen, H., at al. Knowlege engineering approach to morphological analysis // Jappinen H., at al, eds. Morphological analysis of Finnish word forms. Publications of the Kielikone project, series A, report no. 1, 1986.
131. Jappinen, H., M.Ylilammi. Associative model of morphological analysis: an empirical inquiry // Computational Linguistics 12 (4), 1986, p. 257-272.
132. Karlsson, F., K.Koskenniemi. A process model of morphology and lexicon // Folia Linguistica 14, 1985, p. 207-231.
133. Karttunen, L. KIMMO: a general morphological processor // Texas Linguistic Forum 22, 1983, p. 163-186.
134. Karttunen, L., K.Koskenniemi, R.Kaplan. A compiler for two-level phonological rules. Stanford, CA: Xerox Paolo Alto Research Center and Center for the Study of Language and Information, 1987.

135. Karttunen, L., K.Wittenburg. A two-level morphological analysis of English, Texas Linguistic Forum 22, 1983, p. 217-228.
136. Kashyap, R.L., B.J.Oommen. An effective algotithm for string correction using generalized edit distances. I. Description of the algorithm and its optimality // Information Sciences, 1981, v. 23, no. 2, p. 123-142.
137. Kashyap, R.L., B.J.Oommen. An effective algotithm for string correction using generalized edit distances. II. Computational complexity of the algorithm and some applications // Information Sciences, 1981, v. 23, no. 3, p. 210-217.
138. Kashyap, R.L., B.J.Oommen. Pattern matching with noisy substrings // Intern. J. Computer Math., 1983, v. 13, p. 17-40.
139. Kashyap, R.L., B.J.Oommen. Spelling correction using probablistic methods // Pattern Recognit. Lett., 1984, v. 2, no. 3, p. 147-154.
140. Kataja, L., K.Koskenniemi. Finite-state description of Semitic morphology: a case study of ancient Akkadian // Proceedings of Coling'88, Association for Computational Linguistics, 1988, p. 313-315.
141. Kawai, A., K.Sugihara, N.Sugie. Sentence structure standardization method for detection errors in English sentences // Systems, Computers, Controls, 1983, v. 14, no. 2, p. 84-92.
142. Kay, M. Nonconcatenative finite-state morphology // ACL Proceedings, Third European Conference, Association for Computational Linguistics, 1987, p. 2-10.
143. Kernighan, M.D., K.W.Church, W.A.Gale. A spelling correction program based on a noisy channel model // Proc. of Coling'90. vol. 2, Helsinki, 1990, p. 205-210.
144. Khan, R. A two-level morphological analysis of Rumanian // Texas Linguistic Forum 22, 1983, p. 253-270.
145. Khan, R., at al. KIMMO user's manual // Texas Linguistic Forum 22, 1983, p. 203-215.
146. Kirschner, Z. MOSAIC — a method of automatic extraction of significant terms from texts // Explizite Beschreibung der Sprache und automatische Textbearbeitung, Matematicko-fyzikalni fakulta UK, Praha, 1983, 120 p.
147. Koskenniemi, K. An application of the two-level model to Finnish // Karlsson, F., Koskenniemi, K. A process model of morphology and lexicon. Folia Linguistica 14, 1985, p. 19-42.
148. Koskenniemi, K. Two-level morphology: a general computational model for word-form recognition and production // Publication No. 11. University of Helsinki: Department of General Linguistics, 1983.
149. Koskenniemi, K., K.W.Church. Complexity, two-level morphology and Finnish // Proceedings of Coling'88, Association for Computational Linguistics, 1988, p. 335-340.
150. Linoff, C, C.Stanfill. Compression of indexes with full positional information in very large text databases // Proc. ACM, SIGIR'93, 1993, p. 88-95.
151. Linstedt, J. A two-level description of Old Church Slavonic morphology // Scando-Slavica 30, 1984, p. 165-189.
152. Lun, S. A two-level morphological analysis of French // Texas Linguistic Forum 22, 1983, p. 271-278.
153. Nix, P. Experience with a space efficient way to store a dictionary // Commun. ACM, 1981, v. 24, no. 5, p. 257-288.
154. Peterson, J.L. Computer programs for detection and correction spelling errors // Commun. ACM, 1980, v. 23, no. 12, p. 676-687.
155. Pollock, J.J. Spelling error detection and correction by computer: some notes and bibliography // Journal of documentation, 1982, v. 38, no. 4, p. 282-291.
156. Pollock, J.J., A.Zamora. Collection and characterization of spelling errors in scientific and scolarly texts // J. Amer. Soc. Inf. Sci., 1983, v. 34, no. 1, p. 51-58.
157. Pollock, J.J., A.Zamora. Automatic spelling correction in scientific and scolarly text // Commun. ACM, 1984, v. 27, no. 4, p. 358-368.
158. Pollock, J.J., A.Zamora. System design for detection and correction of spelling errors in scientific and scolarly texts // J. Amer. Soc. Inf. Sci., 1984, v. 35, no. 2, p. 104-109.
159. Russel, G.J., at al. A dictionary and morphological analizer for English // Proceedings of Coling'86, Association for Computational Linguistics, 1986, p. 277-279.
160. Simons, G.F. A tool for exploring morphology // Notes on Linguistics no. 44, 1989, p. 51-59.
161. Smith, G. De V. A comparision of three string matching algorithms // Software — Practice & Experience, 1982, v. 12, p. 57-66.
162. Subieta, K. A simple method of data correction // Prace IPI PAN, 1983, no. 527, 12 p.
163. Tharp, A.L., Tai Kuo-Chung. The practicality of text signatures for accelerating string searching // Software — Practice & Experience, 1982, v. 12, p. 35-44.

164. Trost, H. The Application of Two Level Morphology to non-concatenative German morphology // Proc. of 13th International Conference on Computational Linguistics, 1990, p. 371-376.
165. Turba, T.N. Checking for spelling and typographical errors in computer-based text // ACM Sigplan Notes, 1981, v. 16, no. 6, p. 51-60.
166. Wallace, K., ed. Morphology as a computational problem. UCLA Occasional Paper no. 7: Working Papers in Morphology, Department of Linguistics, University of California, Los Angeles, 1988.
167. Weber, D.J., at al. AMPLE: a tool for exploring morphology. Occasional Publications in Academic Computing No. 12, Dallas: Summer Institute of Linguistics, 1988, 252 p.
168. Yannakoudakis, E.J., D.Fawthrop. The rules of spelling errors // Inform. Proc. & Manag., 1983, v. 19, no. 2, p. 87-99.
169. Yannakoudakis, E.J., D.Fawthrop. An intelligent spelling error corrector // Inform. Proc. & Manag., 1983, v. 19, no. 2, p. 101-108.
170. Zamora, A. Automatic detection and correction of spelling errors in large data base // J. Amer. Soc. Inf. Sci., 1980, v.31, no. 2, p. 51-57.
171. Zamora, E.M., J.J.Pollock, A.Zamora. The use of trigram analysis for spelling error detection // Inf. Proc. & Manag., 1981, v. 17, no. 6, p. 305-316.