РОССИЙСКАЯ АКАДЕМИЯ НАУК
ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР ИМ. А. А. ДОРОДНИЦЫНА РАН

На правах рукописи

КУДИНОВ ПАВЕЛ ЮРЬЕВИЧ

АДАПТИВНЫЕ МЕТОДЫ ИЗВЛЕЧЕНИЯ ИНФОРМАЦИИ ИЗ СТАТИСТИЧЕСКИХ ТАБЛИЦ, ПРЕДСТАВЛЕННЫХ В ТЕКСТОВОМ ВИДЕ

05.13.17 – теоретические основы информатики

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель
доктор физико-математических наук
К. В. Воронцов

Москва – 2011

Оглавление

Введение

Диссертационная работа посвящена разработке методов извлечения статистических данных из таблиц, представленных в текстовом виде. Предлагается методология полуавтоматической обработки информации, основанная на динамическом обучении с привлечением эксперта. Исследуются инкрементные алгоритмы машинного обучения, не совершающие ошибок на обучающей выборке. Описывается полуавтоматическая система извлечения статистических показателей из таблиц, разработанная на основе предложенной методологии.

Актуальность темы. Социальная, экономическая, демографическая, финансовая статистика собирается и публикуется в бумажном и электронном виде различными организациями. Многие источники, например Росстат, ВЦИОМ, OECD, банки и финансовые организации предоставляют статистические данные в табличном виде (Рис. 1). Число таблиц, созданных ежегодно, измеряется сотнями тысяч. При этом в разных источниках могут использоваться разные термины и структуры таблиц для описания одних и тех же явлений. Это осложняет поиск, агрегацию и анализ динамики изменения статистических показателей.

В настоящее время не существует единого удобного способа поиска статистической информации по всем источникам. Организациям, которые занимаются анализом статистики, приходится обрабатывать таблицы вручную или с помощью примитивных программных средств, адаптация которых к быстро меняющимся источникам требует большого количества ресурсов. При этом обработка каждой новой коллекции таблиц занимает много времени, что ограничивает возможность оперативного анализа статистических данных. Таким образом, создание поисковой системы над множеством доступных источников

является актуальной проблемой.

Исходными данными для поисковой системы являются коллекции таблиц, получаемые из интернета с помощью роботов (crawler), либо загружаемые вручную. Поиск производится не по документам, как в классических поисковых системах, а по *статистическим показателям*, записанным в статистических таблицах. Каждый показатель характеризуется названием измеряемого статистического явления, единицей измерения, периодом времени и значением. Основной функцией системы поиска статистической информации является выдача, агрегирование, фильтрация, группировка статистических показателей и представление их в виде графиков, диаграмм или таблиц. В качестве запросов может использоваться один или несколько фрагментов названий статистических показателей, период времени, регионы и т. п.

Распределение численности занятых в экономике регионов Российской Федерации по возрастным группам в 2000 г. (в процентах)

	Всего	в том числе в возрасте, лет						Средний возраст, лет
		до 20	20-29	30-39	40-49	50-59	60-72	
Российская Федерация	**100**	**2,0**	**21,5**	**27,2**	**30,3**	**14,1**	**5,0**	**39,3**
Центральный федеральный округ	**100**	**1,6**	**20,0**	**26,6**	**30,1**	**15,7**	**6,1**	**40,1**
Белгородская область	100	1,8	19,8	28,4	30,5	12,6	6,9	39,9
Брянская область	100	1,9	22,8	27,7	30,7	12,3	4,6	38,8
Владимирская область	100	2,2	21,0	26,5	30,6	14,4	5,2	39,4
Воронежская								

Рис. 1. Пример статистической таблицы.

Известные методы информационного поиска ориентированы на обработку текстовых документов или изображений и не предназначены для обработки информации, представленной в табличной форме. Для создания поисковой системы по статистическим данным необходимо сначала решить задачу извлечения статистических показателей из произвольных таблиц. Основной проблемой является многообразие возможных способов записи одних и тех

же показателей в таблицах. Исходные таблицы могут содержать неполные, противоречивые, по-разному агрегированные данные, строиться на разной терминологии и иметь разнородную, плохо формализованную структуру. Существенной проблемой является большое количество опечаток и ошибок, которые появляются на этапе набора таблиц человеком или при применении автоматических методов распознавания текстов (OCR — Optical Character Recognition). Такие особенности исходных данных значительно снижают релевантность поисковой выдачи и делают задачи верификации и агрегирования статистических показателей трудновыполнимыми.

В настоящее время много научных исследований посвящено извлечению данных из таблиц. Но большинство методов либо не являются обучаемыми, либо ориентированы на узкое множество исходных таблиц. Адаптация этих методов к более широкому классу таблиц приведёт к неуправляемому росту сложности программной реализации. Методы, основанные на обучении по фиксированной выборке, неэффективно используют время эксперта, т.к. требуют разметки представительной обучающей выборки, необходимый объём которой трудно оценить заранее.

Более рациональным представляется режим диалога, при котором система предъявляет размеченные ею таблицы, а эксперт только исправляет допущенные системой ошибки. Эти исправления формируют обучающую выборку в режиме адаптивного или инкрементного обучения (incremental learning). При этом к алгоритму обучения предъявляются *требование корректности* — он не должен совершать ошибок на всех исправлениях, сделанных экспертом ранее.

Корректные инкрементные методы обучения, способные эффективно обучаться по выборкам длиной в сотни тысяч объектов и более, в литературе

практически не изучались.

Таким образом, актуальной задачей является как создание автоматизированной системы для извлечения статистических показателей из текстовых таблиц, так и разработка корректных инкрементных методов обучения, достаточно эффективных на выборках большого объёма.

Цель диссертационной работы состоит в создании общей методологии автоматизированной обработки информации с привлечением экспертов и разработке корректных инкрементных методов классификации.

Методы исследования. Извлечение информации из статистических таблиц представляется в виде последовательности задач распознавания: определения типа ячеек, особенностей структуры таблиц, выделения названия таблицы и содержимого каждой ячейки. Некоторые из них формулируются как задачи классификации, для решения которых в данной работе применяются инкрементные методы машинного обучения и методы оптимизации для улучшения качества классификации. Для распознавания содержимого ячеек таблицы используются методы информационного поиска и анализа текстов.

На защиту выносятся следующие научные результаты и положения:

1. Методология построения обучаемых систем извлечения информации из плохо структурированных текстовых данных, основанная на применении корректных инкрементных алгоритмов классификации.

2. Алгоритм инкрементного обучения композиций случайных деревьев (Random Incremental Forest, RIF).

3. Методы и технология извлечения статистических показателей из таблиц, представленных в текстовом виде.

Научная новизна. Постановка задачи извлечения статистических показателей из больших коллекций разнородных таблиц, представленных в текстовом виде, является новой. Автором предложена методология разметки обучающей выборки и корректный инкрементный алгоритм машинного обучения, основанный на решающих деревьях и их композициях.

Практическая значимость. Предложенная технология является основой для построения поисковой системы по статистическим таблицам. Использование инкрементных методов позволяет существенно сократить трудозатраты по формированию обучающих выборок.

Модули извлечения данных из таблиц были использованы в проектах Международного научно-образовательного Форсайт-центра Института статистических исследований и экономики знаний Национального исследовательского университета «Высшая школа экономики» при обработке экспертных оценок, представленных в виде статистических таблиц различного типа.

Апробация работы. Результаты работы докладывались и обсуждались на следующих научных конференциях:

— XI Всероссийская объединенная конференция «Интернет и современное общество» IMS–2008 (Санкт-Петербург, 2008 год);

— XVI международная научная конференция студентов, аспирантов и молодых учёных «Ломоносов–2009» (Москва, 2009 год);

— 14-я всероссийская конференция «Математические методы распознавания образов» (Суздаль, 2009 год);

— 8-я международная конференция «Интеллектуализация обработки информации» ИОИ'10 (Пафос, Республика Кипр, 2010 год);

— 6-я международная конференция «Служба экономических и социологических данных» ESDS-2010 (Лондон, Великобритания, 2010 год);

— 15-я всероссийская конференция «Математические методы распознавания образов» (Петрозаводск, 2011 год).

Описания отдельных результатов работы включены в отчёты по проектам РФФИ №№08-07-00305-а, 10-07-00609-а, 11-07-00480-а.

Личный вклад. Вклад автора работы в результаты, выносимые на защиту, является определяющим.

Публикации. Материалы диссертации опубликованы в 7 статьях [3–8, 24], из них 2 работы [8, 24] — в журналах, включённых в Перечень ведущих рецензируемых научных журналов и изданий.

Структура и объём диссертации. Работа состоит из оглавления, введения, трёх глав, заключения и списка литературы. Содержание работы изложено на 105 страницах. Список литературы включает 46 наименований. Текст работы иллюстрируется 39 рисунками и 7 таблицами.

Краткое содержание работы по главам.

В первой главе описывается общая проблематика извлечения статистической информации из таблиц, рассматриваются и анализируются известные подходы. Формулируется концепция построения поисковой системы по статистическим таблицам. Описываются особенности входящих в неё подсистем, реализация каждой из которых является сложной научно-технической задачей. Детально анализируется основная часть поисковой системы — подсистема извлечения статистических показателей из таблиц, представленных в текстовом виде. Предлагается методология, лежащая в основе разработки этой подсистемы.

Во второй главе вводятся основные понятия, определения и обозначения, описываются особенности статистических таблиц, приводится постановка задачи извлечения статистических показателей. Излагаются все этапы

решения поставленной задачи. Для этапов, представляющих из себя задачи классификации, приводятся их постановки, перечисляются признаки классифицируемых объектов. Во второй части предлагается новый инкрементный алгоритм, основанный на построении композиции случайных деревьев. Доказывается теорема о его корректности.

В третьей главе описывается архитектура предложенного решения, особенности реализации процедуры динамического обучения и интерфейса пользователя. Приводятся результаты экспериментального сравнения известного алгоритма на задачах репозитория UCI и реальной выборке.

В заключении подводятся итоги работы.

Глава 1

Концепция извлечения информации в полуавтоматическом режиме

Статистическая информация включает социально-демографические, экономические и финансовые статистические данные, необходимые для проведения фундаментальных и прикладных исследований, построения прогнозов и принятия решений. Ввиду большого количества источников необходима система, которая позволила бы объединить всю статистическую информацию в одной базе. Разработка такой системы является сложной научно-технической задачей, поскольку отсутствует методология извлечения данных из произвольных источников информации.

В главе описываются существующие решения по извлечению информации из таблиц, рассматриваются особенности статистических таблиц и предлагается методология полуавтоматического режима обработки таблиц.

1.1. Разработка системы поиска статистической информации

Система поиска статистической информации состоит из нескольких основных подсистем:

- подсистема поиска новых источников данных (статистических таблиц);
- подсистема извлечения статистической информации из найденных таблиц;

- подсистема поиска релевантных статистических показателей и их отображения.

1.1.1. Поиск новых источников данных

Поисковые роботы (crawlers) известных поисковых систем предназначены для прохода по всем страницам интернета или конкретных веб-сайтов и занесения их содержимого в поисковую базу данных. При поиске статистических данных интересны только те HTML-страницы, на которых находятся статистические таблицы. Таким образом, каждая страница должна быть проанализирована на предмет наличия в ней статистических таблиц. Существует два основных способа успешного распознавания этих таблиц — использование эвристических приёмов [21] и методов машинного обучения [41].

Первый способ подразумевает использование фиксированных правил определения того, является ли таблица информационной, или она используется для вёрстки HTML страницы. В работе Hsin-Hsi Chen и др. [21] предлагается руководствоваться двумя правилами для фильтрации неинформативных таблиц:

1) таблица содержит как минимум две ячейки;

2) таблица не содержит большого количества гиперссылок (определяется на основе порога).

Идея второй способа, предложенного Yalin Wang и Jianying Hu [41], заключается в использовании нескольких групп признаков, учитывающих разметку таблицы, тип содержимого и особенности содержимого текстовых ячеек. Значения признаков вычисляются на основе анализа HTML кода таблиц. Для классификации применяются известные методы машинного обучения,

такие как решающие деревья и машины опорных векторов (support vector machine, SVM).

Значительное число работ [10, 12, 20, 27, 31, 34] посвящено извлечению таблиц из изображений и текстовой псевдографики. Эти методы также могут быть использованы как средство получения структурированных таблиц в формате HTML.

Все эти методы позволяют получить статистические таблицы в структурированном формате HTML из различных типов источников, включая интернет и преобразованные в изображения печатные издания. Полученные таблицы могут использоваться в качестве исходных данных для описываемой в диссертации системы извлечения статистической информации.

1.1.2. Методы извлечения статистической информации из таблиц

Извлечение статистической информации из таблиц сводится к определению смысла каждого числового значения. Под смыслом понимается полное описание измеряемого статистического явления, единица измерения и период времени.

1.1.2.1. Этапы извлечения статистических показателей

Рассмотрим процедуру извлечения статистических показателей из таблиц (рис. 1.1). В диссертации предлагается свести эту задачу к последовательному решению разных типов задач распознавания:

1) распознавание типа ячеек;

2) распознавание суперстрок;

3) распознавание вложенных ячеек;

4) извлечение названия таблицы из окружающего текста;

5) поиск содержимого ячеек и названия таблицы в словарях;

6) извлечение периода времени;

7) извлечение единиц измерения;

8) построение статистических показателей.

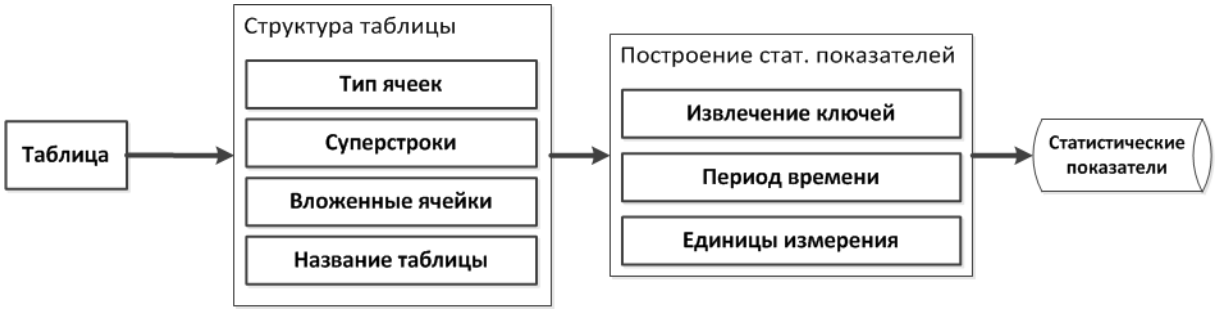


Рис. 1.1. Этапы обработки таблицы.

Прежде всего предлагается определить *логическую структуру* таблицы. Для этого происходит распознавание типа каждой ячейки, а также распознавание суперстрок и вложенных ячеек. Затем извлекается название статистического показателя. Оно формируется из множества ключей, которые извлекаются из содержимого ячеек описания и названия таблицы. После этого определяется период времени и единица измерения каждой ячейки данных на основе анализа содержимого определяющих её ячеек описания. На последнем шаге вся собранная информация объединяется в одной структуре, называемой статистическим показателем.

Рассмотрим особенности каждого шага подробнее.

Распознавание типа ячеек. Ячейки в статистических таблицах бывают трёх типов: пустые, ячейки данных и ячейки описания.

Пустая ячейка не содержит информации. Ячейка данных содержит значение статистического показателя, а ячейки описания в совокупности описывают его смысл.

В задаче распознавания типа ячеек таблицы объектами являются сами ячейки, а классами — три описанных типа ячеек. На рис. 1.2 тёмно-серым цветом выделены ячейки описания, светло-серым — ячейки данных, а остальные ячейки — пустые.

Распределение численности занятых в экономике регионов Российской Федерации по возрастным группам в 2000 г. (в процентах)

	Всего	в том числе в возрасте, лет						Средний возраст, лет
		до 20	20-29	30-39	40-49	50-59	60-72	
Российская Федерация	**100**	**2,0**	**21,5**	**27,2**	**30,3**	**14,1**	**5,0**	**39,3**
Центральный федеральный округ	**100**	**1,6**	**20,0**	**26,6**	**30,1**	**15,7**	**6,1**	**40,1**
Белгородская область	100	1,8	19,8	28,4	30,5	12,6	6,9	39,9
Брянская область	100	1,9	22,8	27,7	30,7	12,3	4,6	38,8
Владимирская область	100	2,2	21,0	26,5	30,6	14,4	5,2	39,4
Воронежская								

Рис. 1.2. Пример статистической таблицы.

Большая часть статистических таблиц содержит блок ячеек описания сверху и слева. Такие таблицы относятся к классу таблиц *простой структуры*, т.к. для каждой ячейки данных множество ячеек описания определяется тривиальным образом — необходимо взять все ячейки описания из данной строки и столбца. Для выделенной на рис. 1.2 ячейки данных множество ячеек описания включает ячейки «Брянская область», «20-29», «в том числе в возрасте, лет». Пример статистической таблицы простой структуры из коллекции Росстата[1] представлен на рис. 1.3.

[1] $http://www.gks.ru/bgd/regl/b10_13/IssWWW.exe/Stg/d1/01-05.htm$

1.5. ТЕМПЫ РОСТА (СНИЖЕНИЯ) ПРОИЗВОДИТЕЛЬНОСТИ ТРУДА ПО ВИДАМ ЭКОНОМИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ
(в процентах к предыдущему году)

	2003	2004	2005	2006	2007	2008	2009
Всего в экономике	**107,0**	**106,5**	**105,5**	**107,5**	**107,5**	**104,8**	**95,8**
Сельское хозяйство, охота и лесное хозяйство	105,6	102,9	101,8	104,3	105,0	110,7	105,0
Рыболовство, рыбоводство	102,1	104,3	96,5	101,6	103,2	95,5	109,2
Добыча полезных ископаемых	109,2	107,3	106,3	103,3	103,1	101,0	107,5
Обрабатывающие производства	108,8	109,8	106,0	108,5	108,4	102,6	96,1
Производство и распределение электроэнергии, газа и воды	103,7	100,7	103,7	101,9	97,5	102,1	96,3
Строительство	105,3	106,8	105,9	115,8	112,8	109,1	91,4
Оптовая и розничная торговля; ремонт автотранспортных средств, мотоциклов, бытовых изделий и предметов личного пользования	109,8	110,5	105,1	110,8	104,8	108,1	92,1
Гостиницы и рестораны	100,3	103,1	108,5	109,2	108,0	109,2	87,1
Транспорт и связь	107,5	108,7	102,1	110,7	107,5	106,5	100,1
Операции с недвижимым имуществом, аренда и предоставление услуг	102,5	101,3	112,4	106,2	117,1	107,9	96,7

Рис. 1.3. Пример таблицы простой структуры из коллекции Росстата.

Извлечение названия таблицы из окружающего текста. Название таблицы также задаёт смысл описанных в ней статистических показателей. Обычно название состоит из одного предложения и расположено перед таблицей. При этом оно может отличаться стилевым оформлением.

Распознавание суперстрок. Кроме таблиц простой структуры для удобства представления статистических показателей используются такие средства как *суперстроки* и *вложенные ячейки*.

Суперстроки предназначены для объединения в одной таблице статистических показателей, которые отличаются только в одной из характеристик: года, вида продукции, территории и пр. Пример таблицы из коллекции Росстата[2] с суперстроками представлен на рис. 1.4. В данном случае все статистические показатели определяются ячейками описания в шапке таблицы (первые три строки), а также ячейками «2010 г.» и «2011 г.». Таким образом,

[2] $http://www.gks.ru/bgd/free/B11_00/IssWWW.exe/Stg/dk06/7-0.htm$

Динамика численности безработных

на конец месяца

	Общая численность безработных			*Численность официально зарегистрированных безработных*		
	тыс. человек	*в % к*		*тыс. человек*	*в % к*	
		соответствующему периоду предыдущего года	*предыдущему периоду*		*соответствующему периоду предыдущего года*	*предыдущему периоду*
2010г.						
Январь	6832	105,1	110,7	2202	129,0	102,6
Год (в среднем за месяц)	***5636***	***88,9***		***1875***	***90,0***	
2011г.						
Январь	5815	85,1	107,8	1609	73,1	101,2

Рис. 1.4. Пример таблицы с суперстроками из коллекции Росстата.

все строки таблицы могут быть разделены на обычные строки, содержащие как ячейки описания, так и ячейки данных, и суперстроки, которые логически разделяют таблицу на несколько частей.

Во многих работах по определению структуры таблиц в изображениях [10], или представленных в текстовой псевдографике, предлагаются методы распознавания суперстрок. Однако суперстроки рассматриваются только для одного случая — когда строка состоит из одной ячейки. Между тем во многих таблицах суперстроки могут иметь более сложный вид, поэтому в настоящей работе задача распознавания суперстрок ставится как задача классификации.

Распознавание вложенных ячеек. Ещё одним часто встречающимся приёмом оформления таблиц является использование вложенных ячеек (рис. 1.5). Его игнорирование приводит к неправильному пониманию сути отражённой в таблице информации. Чаще всего такой приём используется в таблицах

Росстата, в которых несколько ячеек могут быть сдвинуты на один уровень вправо. При этом таблицы с несколькими уровнями вложенности занимают незначительную долю всех таблиц, поэтому задача распознавания нескольких уровней не представляет практического интереса.

Женщины и мужчины по возрастным группам

	2002				2010			
	тыс. человек		распределение по полу, %		тыс. человек		распределение по полу, %	
	женщины	мужчины	женщины	мужчины	женщины	мужчины	женщины	мужчины
Все население	**77562**	**67605**	**53**	**47**	**76275**	**65639**	**54**	**46**
в том числе в возрасте, лет:								
0-9	6515	6825	49	51	7224	7612	49	51
10-19	11390	11817	49	51	7366	7694	49	51
20-29	10982	11097	50	50	12143	12370	50	50
30-39	10113	9939	50	50	10575	10293	51	49
40-49	12575	11578	52	48	10688	9753	52	48
50-59	8411	7008	55	45	11652	9356	55	45
60-69	8633	5695	60	40	6898	4478	61	39
70-79	6739	3070	69	31	6630	3156	68	32
80 лет и более	2144	516	81	19	3099	927	77	23

Рис. 1.5. Пример таблицы с вложенными ячейками из коллекции Росстата.

В работе Шигарова [10] рассматриваются таблицы, содержащие вложенные ячейки. Для их учёта строится дерево вложенности ячеек. Однако сама

вложенность определятся на основе визуального сдвига. Этот способ хорошо работает на таблицах, представленных в виде псевдографики, однако он не будет так же хорошо работать на таблицах из некоторых коллекций Росстата и других коллекций таблиц, где вложенность может быть определена только на основе анализа содержимого ячеек.

Поиск содержимого ячеек в словарях. Терминология описания статистической информации не является устойчивой для большинства статистических явлений. Несмотря на то, что для российской статистики существуют различные классификаторы, такие как ОКАТО[3], ОКВЭД[4], ОКП[5] и пр., они описывают не все социально-экономические разделы, и не все составители таблиц ими пользуются. На рис. 1.6 показан пример того, каким образом упоминается город Москва в большинстве статистических таблиц.

Рис. 1.6. Синонимы ключа «г. Москва».

Таким образом, важной задачей является приведение названий статистических показателей к единому виду. Для решения этой задачи вводится понятие *ключа* как элементарной семантической единицы статистического яв-

[3] Общероссийский классификатор административно-территориальных объектов.

[4] Общероссийский классификатор видов экономической деятельности.

[5] Общероссийский классификатор продукции.

ления. Примером ключей может служить регион (территория), единица измерения, профессия, продукция. Название статистического показателя формируется из множества ключей, извлечённых из содержимого ячеек. Для каждого ключа может быть определено множество его эквивалентных словесных представлений.

Важной особенностью многих таблиц является и то, что в тексте одной ячейки может содержаться несколько ключей. На рис. 1.7 сверху показан пример одной из таблиц с текстом ячейки, разделённым на семантические части. Некоторым частям могут соответствовать ключи, на рис. 1.7 они выделены разными цветами. Поэтому задачу построения названия статистического показателя следует формулировать как задачу поиска всевозможных словесных представлений ключей во всех ячейках описания данной числовой ячейки.

Рис. 1.7. Поиск всех ключей в тексте ячейки описания.

Извлечение периода времени. Даты в статистических таблицах представляются в числовой и словесной формах, а также они могут быть подвергнуты уточнениям различных типов, выраженным в словесной форме, например «на конец года». Статистические данные по учебным заведениям обычно заданы для учебного года. Обычно каждому статистическому измерению соответствует некоторый период времени. Он может иметь две границы, например «прирост населения за год» или левая граница может быть не задана явно, например «численность населения на конец года».

Таким образом, для каждого извлекаемого статистического показателя

необходимо получить интервал времени $[t_1, t_2]$. При этом левая граница интервала может быть не известной или вообще не заданной.

Пример 1. *Текст одной из ячеек описания равен «2010». Период времени:* $[01.01.2010, 31.12.2010]$.

Пример 2. *Текст одной из ячеек описания равен «на начало года». Текст другой равен «2009». Период времени:* $[\varnothing, 01.01.2009]$

Пример 3. *Текст одной из ячеек описания равен «в 2009/2010 учебном году». Период времени:* $[01.09.2009, 01.06.2010]$

Пример 4. *Текст одной из ячеек описания равен «январь 2010 года». Период времени:* $[01.01.2010, 31.01.2010]$.

Извлечение единиц измерения. Обычно единицы измерения определяются в одной или нескольких ячейках описания. Предполагается, что всё множество ключей содержит все возможные ключи, имеющие отношение к единицам измерения. Это означает, что они будут найдены на этапе поиска всех ключей в ячейках описания.

1.1.2.2. Методы извлечения информации из статистических таблиц

Извлечение данных из таблиц (table extraction) является хорошо известной задачей. Многие существующие методы извлечения табличной информации из документов подробно рассматриваются в обзорах авторов e Silva A.C. и др. [33], Embley D.W. и др. [35], Lopresti D. и др. [28], Zanibbi R. и др. [45, 46]. Основная часть работ в этом направлении посвящена определению физической структуры таблицы, когда она не известна. Исходными данными

в этом случае являются документы в чистом текстовом (plain text) виде [17], либо изображения [42].

Методы, основанные на анализе изображений и псевдографике фокусируются на определении линий, разграничивающих ячейки таблиц, а также распознавании текстового содержимого каждой ячейки. Для решения этих задач используются как обучаемые, так и необучаемые алгоритмы.

Есть также работы, посвящённые анализу HTML кода для извлечения данных. Незначительная часть исследований посвящена семантическому анализу содержимого таблиц или построению систем, позволяющих отвечать на вопросы в свободной форме, основываясь на информации, полученной из таблиц [32]. В работе [23] описывается метод извлечения данных из таблиц, основанный на необучаемых методах, которые работают для ограниченного множества таблиц, не содержащих суперстрок и вложенных ячеек.

В работе Шигарова [10] предлагается комплексная система извлечения статистических данных из изображений, содержащих таблицы, основанная на анализе метафайлов EMF. Основным преимуществом системы является то, что она позволяет решить задачу полностью, т. е. введя документ в любом формате, можно получить множество статистических показателей. Существенным недостатком предложенной системы является то, что её использование ограничено таблицами определённой структуры, построенными по конкретным правилам. Это связано с тем, что в ней не исследуются различные типы суперстрок и вложенных ячеек.

В работе [21] предлагается необучаемый метод извлечения данных из таблиц. Он основан на мере сходства ячеек и построенной на этой основе мере сходства строк и столбцов. Сходство ячеек определяется на основе анализа нескольких факторов: близость строк, наличие одинаковых слов или слов из

одной категории (единицы измерения, даты, денежные единицы), количество цифр в тексте ячейки. Далее на основе пороговых правил восстанавливается структура таблицы. Ячейки описания в предложенной модели могут быть только в первых столбцах и/или первых рядах, а объединённые ячейки всегда принимаются за ячейки описания.

В работе David W. Embley и Cui Tao [18] предлагается решение задачи автоматического распознавания цены и свойств автомобилей в объявлениях об их продаже, представленных в виде таблиц и размещенных в интернете. Предлагаемое решение состоит в полном описании содержания исходных данных и ручном составлении «выделяющей онтологии» (extracting ontology), позволяющей в автоматическом режиме обрабатывать автомобильные объявления. Выделяющая онтология состоит из полного описания предметной области, включая полное перечисление свойств объектов (в данном случае автомобилей) и всех возможных значений каждого свойства. Однако использование предложенного авторами язык описания выделяющих онтологий не представляется возможным для определения всех объектов и статистических явлений, которые могут встретиться в статистических таблицах.

Tengli и др. [36] предложили подход, основанный на обучаемых методах, пригодных для распознавания таблиц фиксированной структуры из фиксированной прикладной области. Имеется обучающая выборка таблиц, на основе которой происходит построение исходной базы известных значений ячеек с текстом. Рассматриваются только таблицы простой структуры. При распознавании каждой новой таблицы происходит поиск значений в базе. Пополнения базы в процессе обучения не предполагается. Несмотря на то, что система показывает неплохие результаты (ошибка на уровне 80%) на таблицах из той же совокупности, откуда была взята обучающая выборка, данный метод не

применим для таблиц, содержащих статистические данные.

В работе [44] предпринимается попытка объединения схожих данных из различных таблиц. В ней перечисляются поддерживаемые логические структуры таблиц. Для определения типа ячеек используется вероятностная модель, основанная на максимизации функции правдоподобия. Вероятность оценивается на основе априорных знаний о типичном содержимом ячеек каждого из типов. Объединение таблиц сводится к задаче кластеризации содержимого ячеек описания, которое называется *атрибутом*. Каждый атрибут представляется в виде вектора, компонентами которого являются частоты ячеек данных, связанные с этим атрибутом. Мера близости между атрибутами определяется как скалярное произведение в пространстве векторов.

Общим недостатком известных методов является предположение о том, что в одной текстовой ячейке может содержаться только один ключ. Такое предположение является существенным ограничением при работе с реальными данными. В большинстве существующих подходов не проводится анализ содержимого ячеек с целью объединения значений одних и тех же статистических показателей, полученных из разных типов таблиц.

Таким образом, известные подходы к извлечению информации из таблиц предлагают алгоритмы, предназначенные для обработки выбранного разработчиками набора таблиц. Между тем, в задаче построения поисковой системы необходимо рассматривать максимально широкий круг таблиц, чтобы обеспечить полноту информации. Доработка алгоритмов для каждого набора таблиц является трудоёмким процессом, поэтому необходимо использовать *адаптивные обучаемые процедуры*, которые позволяют обрабатывать большее количество источников данных без вмешательства в программный код системы. Следовательно, известные методы не позволяют организовать поис-

ковую систему над статистическими показателями, обеспечивающую полноту и достоверность полученной информации.

1.1.3. Инструменты поиска и отображения статистических показателей

Существенным недостатком имеющихся в открытом доступе хранилищ статистических данных, таких как Росстат, Мировой банк и др., является отсутствие возможности быстрого поиска статистической информации по названию показателей.

Названия статистических показателей, извлечённых из таблиц, могут состоять из нескольких ключей. Возникает задача сравнения названий нескольких показателей, а также возможности их объединения, например, для формирования временного ряда. Если при анализе множеств ключей, описывающих названия двух показателей, обнаруживается, что они относятся к одному и тому же явлению, то сравнению подвергаются единицы измерения статистических показателей. В случае, когда они различаются, для создания единого временного ряда требуется приведение значений показателей к одной единице измерения.

Таким же образом могут быть получены двумерные таблицы, построенные по принципу объединения множеств статистических показателей, с совпадающими типами ключей, но имеющих различные значения для двух фиксированных типов ключей, или одного типа ключей и периода времени. Например, данные по численности населения могут быть объединены в таблицу, в которой по строкам будет записана информация о территории, а по столбцам — различные периоды времени.

Одним из преимуществ единой поисковой системы по статистическим

данным является то, что в ней могут быть представлены данные об одних и тех же статистических явлениях, полученных от разных организаций. Поэтому особый интерес представляет возможность сравнения временных рядов и таблиц, построенных на основе данных из различных источников, что позволяет выявить противоречивую статистическую информацию. В связи с этим, статистические данные могут объединяться в таблицах и временных рядах только в случае их непротиворечивости.

При поиске необходимой информации можно использовать несколько способов поиска. Простой поиск по части названия статистического показателя должен выдавать временные ряды из показателей, в названиях которых содержатся соответствующие фрагменты.

Более детализированный поиск может позволять постепенно уточнять запрос, предлагая выбрать из сгруппированных по типу значений ключей различных показателей. Например, при запросе «численность учеников средних школ» поисковая система может предложить уточнить регион, затем город и период времени.

Возможно также использование конструкторов таблиц. Они позволят формировать макет таблицы (то есть, определять её возможное представление), а также последовательно задавать структуру таблицы, указывая какие типы ключей располагать в заголовках.

Реализация такой поисковой системы является сложной научно-технической задачей, требующей эффективной индексации данных, их кластеризации и фильтрации, а также разработки удобных пользовательских интерфейсов.

1.2. Концептуальная основа адаптивных методов извлечения статистической информации

В основе концепции данного исследования лежит комплексный подход к извлечению статистической информации, включающий динамическое распознавание логической структуры таблиц, поиск семантических единиц в ячейках описания, извлечение периодов времени и единиц измерения. Предлагаемый подход основан на применении адаптивных методов при извлечении информации из статистических таблиц разного типа, описанных в главе 1.

Адаптивные методы включают инструменты создания обучающих выборок, пополнения словарей и динамических методов машинного обучения в процессе извлечения статистической информации из текстовых таблиц.

Комплексный подход разрабатывается для достижения следующих основных целей:

- увеличение множества таблиц, доступных для корректной обработки;
- уменьшение количества рутинных операций по обработке;
- повышение вычислительной эффективности используемых методов машинного обучения.

Составители статистических таблиц используют различные типы логической структуры таблиц и различные средства для разделения таблицы на ячейки и блоки. Поэтому предлагаемое решение учитывает больше особенностей логической структуры таблиц, таких как суперстроки и вложенные ячейки. Ввиду того, что эти особенности могут проявляться различными способами, постоянная доработка алгоритмов анализа структуры таблиц приведёт к неуправляемому росту их сложности. Поэтому решение задачи определения

логической структуры таблицы предлагается свести с решению трёх задач классификации — классификации типов ячеек, суперстрок и вложенных ячеек. Это позволяет учитывать ранее неизвестные особенности обрабатываемых таблиц посредством добавления новых признаков.

Решение задач классификации предполагает предварительное обучение алгоритмов. Обучение должно проводиться на представительной выборке, охватывающей большинство особенностей статистических таблиц. Однако для составления такой выборки необходимо вручную обработать большое количество таблиц и принять решение о включении очередной таблицы в обучающую выборку. В этом случае необходимо выполнить полную разметку таблицы. Такая процедура предполагает большие трудозатраты. Динамическое пополнение обучающей выборки является частью процесса динамического обучения и подразумевает дополнение обучающей выборки только теми таблицами, которые были обработаны с ошибками.

При динамическом обучении таблицы обрабатываются по очереди. На начальном этапе экспертами размечается начальная обучающая выборка S_0, которая может быть небольшого размера. На полученной обучающей выборке производится начальное обучение алгоритма $A_0 = \mu(S_0)$, где μ — метод обучения.

Далее процесс динамического обучения состоит из последовательности чередующихся шагов классификации и дообучения. На стадии классификации i-ой таблицы алгоритму A_{i-1} классификации не известно значение целевого признака. Предполагается, что эксперты просматривают обработанные таблицы в специализированном пользовательском интерфейсе и верифицируют результаты обработки. Все исправления экспертов сохраняются и преобразуются в обучающую выборку S_i и используются для дообучения алгоритма

классификации A_{i-1}. В этом случае эксперт исправляет только ошибки классификации, что существенно уменьшает объём работы. Если ошибки распознавания редки, то динамический режим существенно снижает трудоёмкость обучения в сравнении с обычной практикой, когда обучающая выборка формируется заранее (offline learning). После этого происходит дообучение алгоритма: $A_i = \mu(A_{i-1}, S_i)$. Отметим, что шагом динамического обучения может быть не одна таблица, а несколько, например, коллекция таблиц из одного сборника Росстата.

Примером online-алгоритма может служить модифицированный таким образом алгоритм k ближайших соседей (kNN), чтобы новые объекты после классификации добавлялись в обучающую выборку. Алгоритмы Rule Induction (RISE [16], SLIPPER [15], IREP [14] и др.) также можно модифицировать таким образом, чтобы правила продолжали строиться или модифицироваться по ходу работы алгоритма.

Известны также инкрементные методы построения решающих деревьев (Incremental Tree Induction, ITI) [39, 40], инкрементного бустинга Learn++ [26] и применения генетических алгоритмов в методе (Incremental Learning Using Genetic Algorith, ILUGA) [22]. Исследования последних лет в области инкрементного обучения были направлены в основном на повышение эффективности работы алгоритмов на сверхбольших выборках. В предлагаемой методологии не менее важным аспектом является корректность алгоритмов.

Обычно допускается возможность ошибки алгоритма классификации на обучающей выборке. Однако в описанном режиме эксперт ожидает, что все введённые им объекты будут классифицироваться правильно, ведь в противном случае схожие ошибочные ситуации должны быть обработаны экспертом несколько раз. Поэтому алгоритм классификации, используемый для ре-

шения задач классификации, должен удовлетворять *требованию корректности*, то есть при последующих модификациях он не должен делать ошибок на ранее классифицированных объектах. Будем говорить, что метод динамического обучения корректен, если он удовлетворяет требованию корректности.

Определение 1. *Метод динамического обучения μ корректен, если* $\forall x \in S_0 \cup \cdots \cup S_{i-1} \Rightarrow A_i(x) = A_{i-1}(x), i > 0$.

Требование корректности позволяет существенно сократить время работы эксперта, однако невозможно обеспечить корректность алгоритма на противоречивой выборке. Поэтому интерфейс разметки таблиц должен указывать на факты различия меток классов для объектов с одинаковым признаковым описанием. Такие ситуации означают либо ошибку в действиях эксперта, либо неполноту признакового пространства, в котором представляют объекты обучающей выборки. В этом случае разработчики системы должны иметь возможность быстрого добавления новых признаков, а инкрементные алгоритмы не должны требовать выполнения обучения заново на всей накопленной обучающей выборке.

Таким образом, к алгоритмам динамического обучения предъявляются требования корректности и вычислительной эффективности.

1.3. Основные выводы

1. Поисковая система по статистической информации в таблицах должна состоять из трёх основных подсистем: поиска новых таблиц, извлечения информации из таблиц и поискового инструментария. Реализация каждой из этих систем является сложной научно-технической задачей.

2. Проведённый обзор известных методов поиска таблиц показал, что существуют надёжные способы нахождения таблиц в интернете, на изображениях и в текстовой псевдографике. Использование найденных такими способами таблиц в качестве исходных данных для разрабатываемой системы является целесообразным.

3. В литературе по извлечению статистической информации из таблиц авторы предполагают, что в каждой текстовой ячейке содержится только одна семантическая единица (территория, период времени, вид продукции и т. п.). Такое предположение существенно сужает множество обрабатываемых таблиц и не позволяет рассматривать многие источники данных (например, многие таблицы Росстата).

4. Процедура извлечения статистической информации из таблиц сводится к определению логической структуры и смысла каждого числового значения. Большинство известных методов определения логической структуры таблиц являются необучаемыми. Ввиду описанных в главе свойств исходных таблиц (типы ячеек, наличие суперстрок и вложенных ячеек), определение особенностей логической структуры таблиц целесообразно ставить в виде совокупности задач классификации.

5. В связи с тем, что доработка известных необучаемых алгоритмов для каждого нового набора таблиц является трудоёмкой задачей, требующей постоянного вмешательства в программный код, для расширения множества исходных данных целесообразно использовать адаптивные процедуры.

6. Важным требованием к адаптивным процедурам обучения (динамическим алгоритмам) является безошибочная классификация всех объек-

тов обучающей выборки — требование корректности. Однако в литературе не изучались одновременно корректные и вычислительно эффективные алгоритмы динамического обучения. Поэтому разработка таких алгоритмов является актуальной задачей.

Глава 2

Методы поиска и распознавания статистических данных

В данной главе вводятся основные понятия, определения и обозначения, используемые в постановке и решении задачи извлечения статистической информации. Описываются особенности статистических таблиц, приводится постановка задачи извлечения статистических показателей, излагаются все этапы её решения. Формулируются задачи классификации, являющиеся частью этапа определения логической структуры таблиц, определяются признаки классифицируемых объектов.

Описанные задачи распознавания решаются динамическими алгоритмами машинного обучения. Проводится сравнение инкрементных алгоритмов и обосновывается выбор алгоритма построения дерева решений ITI. Предлагается новый корректный алгоритм построения композиции случайных деревьев решений (RIF).

2.1. Особенности структуры статистических таблиц

Статистические таблицы, рассматриваемые в данном исследовании, отличаются от других таблиц наличием чёткого разделения ячеек на данные и их описания, а также различными особенностями разметки, используемыми для более удобного и компактного представления статистической информации. Такие особенности исходных осложняют автоматический анализ и извлечение информации из таблиц.

2.1.1. Понятия и определения

Социально-демографическая, экономическая, финансовая информация объединяет разнородные *системы статистических показателей*, каждая из которых отражает целостную статистическую характеристику социально-экономического явления.

Ключом k будем называть элементарную, неделимую характеристику измеряемого явления. Множество всех ключей обозначим через $\mathcal{K}$. Каждому ключу $k_i \in \mathcal{K}$ соответствует множество его допустимых словесных описаний D_i. Оно включает все словесные эквиваленты одной и той же статистической характеристики, представленной в текстовом виде. Множества словесных описаний могут быть получены из Общероссийских классификаторов или наполнены вручную.

Примерами ключей могут служить города, виды экономической деятельности, продукции, единицы измерения. При этом каждый ключ может выражаться в тексте по-разному, например, для города Москвы это могут быть «г. Москва» (самое распространённое употребление), и «город Москва, столица Российской Федерации» (название из общероссийского классификатора административно-территориальных объектов).

Обычно статистические таблицы содержат ключи из общероссийских классификаторов, таких как ОКАТО, ОКВЭД, ОКП и пр. Однако во многих таблицах можно встретить не входящие в эти классификаторы тексты ячеек.

Статистическим показателем будем называть четвёрку $s = \langle K, U, d, v \rangle$, определяющую множество ключей $K = \{k_1, \ldots, k_n\} \subset \mathcal{K}$, единицу измерения $U \subset K$, интервал времени $d = [t_1, t_2]$, где t_1, t_2 — даты, и значение $v \in \mathbb{R}$. Множество K полностью описывает смысл (географическое положение, вид

экономической деятельности, вид продукции и т.п.) измеряемого социально-экономического явления. Единица измерения также является подмножеством ключей специального типа. Пространство всех статистических показателей будем обозначать через $\mathcal{I}$.

Рассмотрим квадратную сетку $G^{M\times N}$ из M строк и N столбцов и зададим её полное покрытие непересекающимися прямоугольными областями. Элемент покрытия будем называть *ячейкой*. Множество всех ячеек обозначим через C. Каждой ячейке $c \in C$ поставим в соответствие текстовую строку $text(c)$, возможно пустую, которую назовём *содержимым ячейки* или её *текстом*. Положение ячейки $c \in C$ в сетке будем определять с помощью четырёх координат — номеров первой и последней строки сетки, первого и последнего столбцов сетки: $coord(c) = (r_1(c), c_1(c), r_2(c), c_2(c))$.

Определение 2. *Таблицей будем называть пару* $\langle G^{M\times N}, C\rangle$, *состоящую из прямоугольной сетки и заданного на ней множества ячеек.*

Пространство всех таблиц обозначим через $\mathcal{T}$. Будем полагать, что каждая ячейка c имеет тип $type(c) \in \{data, key, empty\}$. Первый тип соответствует ячейке данных, второй — ячейке описания, третий — пустой (неинформативной) ячейке.

Ячейка данных — это ячейка, содержащая ровно одно числовое значение. *Ячейка описания* представляет собой текстовую строку и содержит один или несколько ключей в виде их словесных описаний. При этом описания ключей могут быть выражены в словесной, числовой или смешанной формах. Примером числовой формы может служить строка «35-45», обозначающая диапазон возрастов. Примером смешанной формы может служить описание «дети, не достигшие 18 лет» из общероссийского классификатора информации по социальной защите населения (ОКИЗН).

Определим отображение $\mathfrak{R} : C_V \to 2^{C_K}$, где C_V — множество ячеек данных, а C_K — множество ячеек описания. Это отображение ставит в соответствие каждой ячейке данных множество ячеек описания, т. е. задаёт *логическую структуру* таблицы.

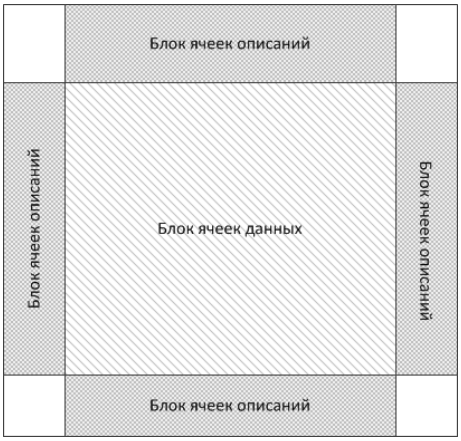


Рис. 2.1. Статистическая таблица простой структуры.

Определение 3. *Статистической таблицей T будем называть четвёрку*

$$\langle G^{M\times N}, C_V, C_K, \mathfrak{R}\rangle.$$

Множество всех статистических таблиц обозначим через $\mathcal{T}_s$. Будем полагать, что если таблица T является статистической, то для неё определено отображение $\mathfrak{S} : \mathcal{T} \to \mathcal{T}_s$.

Подмножество ячеек $X \subseteq C, X \neq \varnothing$, образующих прямоугольную область, будем называть *блоком ячеек*. Будем говорить, что при некотором малом значении $\beta \geqslant 0$ блок X имеет тип $t \in \{data, key, empty\}$, если

$$\frac{1}{|X|}\sum_{c\in X}[type(c) \in \{t, empty\}] > 1 - \beta.$$

Чем меньше значение β, при котором выполнено неравенство, тем точнее блок соответствует своему типу.

Введём формальное определение таблицы простой структуры.

Определение 4. *Статистическая таблица имеет простую структуру, если она содержит ровно один блок данных, с каждой стороны которого может быть не более одного блока ячеек описания, при этом их общее число больше нуля (рис. 2.1).*

2.1.2. Ключи в статистических таблицах

Поиск статистической информации осложнён тем, что текст ячейки не равносилен поисковому ключу. С одной стороны, он может быть представлен в составе группы ключей. В этом случае текст ячейки содержит несколько ключей или вариантов их сочетания.

Пример 5. *В качестве простого примера рассмотрим ключи {«волокна», «волокна хлопковые», «хлопковые ткани», «ткани»} и ячейку с текстом «Волокна хлопковые ткани» (рис. 2.2). В данном случае текст в ячейке может быть представлен в виде набора ключей тремя способами: {«волокна хлопковые», «ткани»}, {«волокна», «хлопковые ткани»} и {«волокна», «ткани»}.*

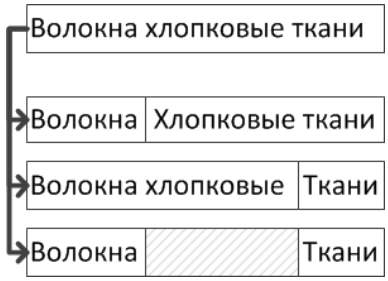


Рис. 2.2. Варианты извлечения ключей из текста ячейки описания.

С другой стороны, каждому ключу $k_i \in \mathcal{K}$ соответствует множество его допустимых словесных описаний D_i. Оно включает все словесные эквиваленты одной и той же статистической характеристики, представленной в текстовом виде. Множества словесных описаний могут быть получены из Общероссийских классификаторов или наполнены вручную.

Пример 6. *Для ключа, соответствующего семантической единице «город Москва», множество D может содержать следующие словесные эквиваленты: «Москва», «город Москва», «г. Москва», «столица Российской Федерации».*

2.2. Задача извлечения статистических показателей

В данном разделе рассматривается задача извлечения статистических показателей. Приводится её постановка и решение, основанное на описанной в первой главе методологии.

2.2.1. Постановка задачи

Пусть на сетке $G^{M \times N}$ из M строк и N столбцов задано множество ячеек C с сопоставленным им текстом, имеется множество ключей $\mathcal{K}$, а также множества эквивалентных описаний D_i каждого ключа $k_i \in \mathcal{K}$ в словесной форме. Задача состоит в том, чтобы построить множество всех статистических показателей $S \subset I$, представленных в таблице T, то есть построить отображение $\mathfrak{M} : \mathcal{T} \to 2^{\mathcal{I}}$.

2.2.2. Решение задачи

Решение этой задачи разбивается на два основных этапа: распознавание статистической таблицы, т.е. построение отображения $\mathfrak{S} : \mathcal{T} \to \mathcal{T}_s$, а затем построение отображения $\mathfrak{I} : \mathcal{T}_s \to 2^{\mathcal{I}}$.

Первый этап состоит из следующих шагов.

1. Распознавание типа каждой ячейки, т. е. построение множеств C_V и C_K.
2. Распознавание суперстрок.
3. Распознавание вложенных ячеек.

4. Чтение таблицы, т. е. построение отображения $\mathfrak{R}$.

Второй этап состоит из следующих шагов.

1. Чтение ячеек описания — извлечение ключей из текста всех ячеек описания.

2. Извлечение названия таблицы из текста.

3. Извлечение периода времени и единицы измерения.

4. Построение множества статистических показателей.

2.2.2.1. Распознавание типа ячеек таблицы

Рассмотрим множество ячеек C. Каждая ячейка $c \in C$ может быть представлена четвёркой координат, которые задают её положение в сетке $G^{M \times N}$: $(r_1(c), c_1(c), r_2(c), c_2(c))$, где

$$\forall c \Rightarrow 1 \leqslant r_1(c) \leqslant r_2(c) \leqslant M, 1 \leqslant c_1(c) \leqslant c_2(c) \leqslant N.$$

Задача распознавания состоит в том, чтобы каждой ячейке $c \in C$ поставить в соответствие её тип, то есть построить отображение $f : C \to \{data, key, empty\}$. Основной проблемой в распознавании типа ячейки является то, что не всякая ячейка, содержащая число, является ячейкой данных.

Для всех ячеек таблицы предлагается следующее признаковое пространство:

1) $numbers(c)$ — количество чисел;

2) $words(c)$ — количество слов;

3) $|text(c)|$ — количество символов;

4) $\big(r_1(c) + r_2(c)\big)/2M$ — вертикальное положение;

5) $\big(c_1(c) + c_2(c)\big)/2N$ — горизонтальное положение;

6) $r_2(c) - r_1(c)$ — число вертикально объединённых элементов сетки;

7) $c_2(c) - c_1(c)$ — число горизонтально объединённых элементов сетки.

Таким образом, имеем задачу классификации, в которой объектами являются ячейки таблицы с описанными выше признаками, а классами — три типа ячеек.

2.2.2.2. Распознавание суперстрок

Всё множество значений статистических показателей в таблице может быть разделено на несколько частей в соответствии со значениями одной из характеристик. Например, данные по мужской и женской занятости, данные за разные годы, абсолютные и относительные данные одних и тех же показателей. Для совмещения таких данных в одной таблице составители часто используют суперстроки (рис. 2.3).

Рассмотрим задачу классификации строк таблицы на два класса: «обычная строка» и «суперстрока». Для каждой строки будем строить следующий набор признаков:

1) $f_1(c)$ — количество ячеек в строке;

2) $f_2(c) = N$ — ширина таблицы;

3) $f_3(c)$ — высота строки;

4) $f_4(c)$ — количество пустых ячеек;

5) $f_5(c) = \big(c_1(c) + c_2(c)\big)/2N$ — количество непустых ячеек.

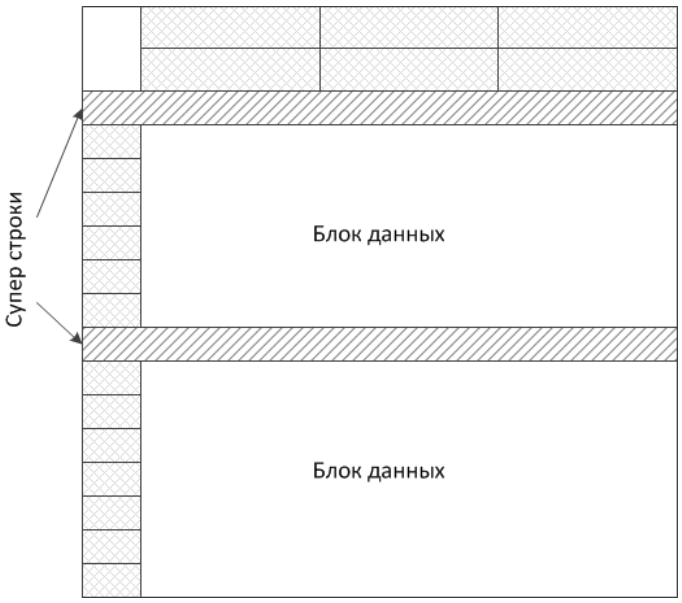


Рис. 2.3. Статистическая таблица, содержащая суперстроки.

После того как все строки таблицы были классифицированы, построим множество C_{super}, состоящее из ячеек суперстрок.

2.2.2.3. Распознавание вложенных ячеек

Вложенные ячейки схематично изображены на рис. 2.4. Рассмотрим задачу классификации, в которой объектами являются пары последовательно идущих ячеек $p = (x_1, x_2)$ в левом блоке ячеек описания, разделённых на три класса: «ячейки находятся на одном уровне», «x_2 сдвинута вправо относительно x_1» и «x_2 сдвинута влево относительно x_1». Для этих объектов вычисляется следующий набор признаков:

1) $f_1(p)$ — заканчивается ли текст x_1 на «:»;

2) $f_2(p)$ — количество начальных пробельных символов в тексте x_2;

3) $f_3(p)$ — тип первого непробельного символа в x_2: «цифра», «буква» или «знак»;

4) $f_4(p)$ — является ли первая буква в x_1 прописной;

5) $f_5(p)$ — является ли первая буква в x_2 прописной;

6) $f_6(p) = r_2(x_1) - r_1(x_1)$ — высота x_1;

7) $f_7(p) = c_2(x_2) - c_1(x_2)$ — высота x_2.

На основе решения этой задачи классификации для каждой ячейки c можно определить ячейку $c_{out}(c)$, для которой c является вложенной. Если такой ячейки не существует, положим её равной $c_{out}(c) = \varnothing$.

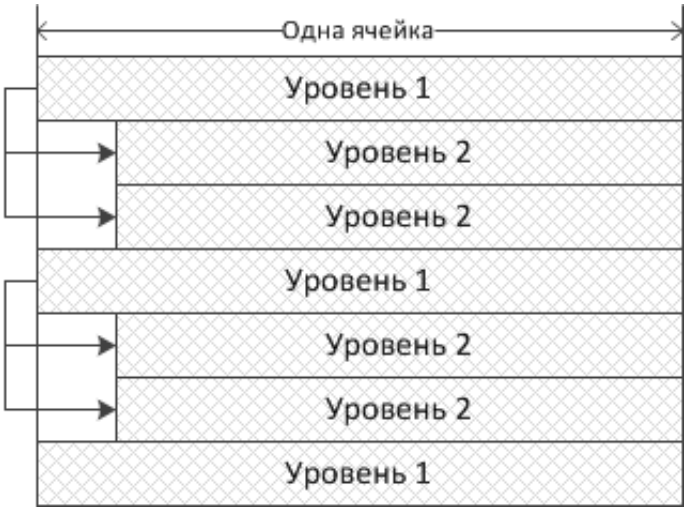


Рис. 2.4. Статистическая таблица, содержащая вложенные ячейки.

2.2.2.4. Восстановление структуры статистической таблицы

После того как были определены типы ячеек, суперстроки и вложенные ячейки, необходимо восстановить структуру статистической таблицы, то есть решить задачу поиска блока с данными и блоков с описаниями. Суперстроки не участвуют в определении блоков данных и ячеек, поэтому их нужно временно исключить из таблицы.

На рис. 2.5 блоки $A_1, A_2, A_3, A_4 \subset C$ — множества ячеек описаний, а $D \subset C$ — множество ячеек данных. Данную конфигурацию блоков можно задать числами x_1, y_1, x_2, y_2. Первое число равно минимальному номеру столбца

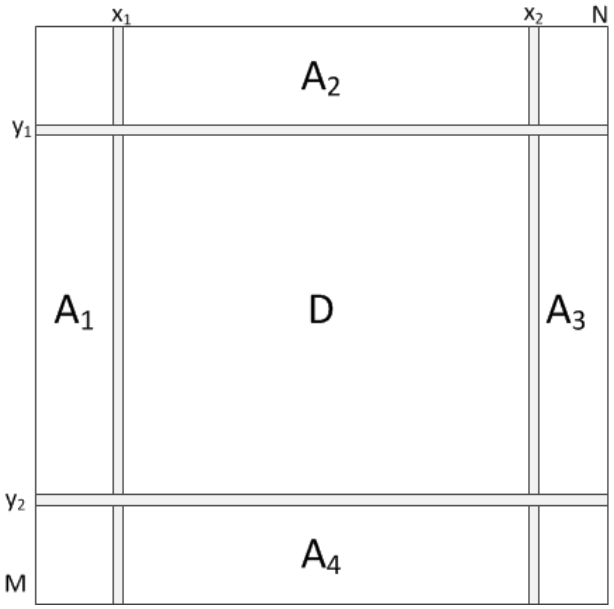


Рис. 2.5. Поиск блоков в таблице.

блока данных D, второе — минимальному номеру строки блока D, третье — минимальному номеру столбца блока A_3, четвёртое — минимальному номеру строки блока A_4.

При отсутствии блока A_3 будем полагать $x_2 = N + 1$, при отсутствии блока A_4 — $y_2 = M+1$. Аналогично, если отсутствует блок A_2, то $y_1 = 1$, если отсутствует блок A_1, то $x_1 = 1$. Тогда условие наличия хотя бы одного блока описаний A_1, A_2, A_3, A_4 запишется в виде $(x_1, y_1, x_2, y_2) \neq (1, 1, N+1, M+1)$. Область допустимых значений Y для y_1 и y_2 определяется как

$$Y = \{y \in \overline{1, M+1} \mid \nexists c \in C : r_1(c) < y < r_2(c)\},$$

то есть множество всех строк сетки, которые не разрезают ни одну ячейку. Аналогичным образом определяется множество допустимых значений для x_1 и x_2:

$$X = \{x \in \overline{1, N+1} \mid \nexists c \in C : c_1(c) < x < c_2(c)\}.$$

Пусть f — отображение $C \to \{data, key, empty\}$, построенное на предыдущем шаге. Среди всевозможных допустимых значений x_1, y_1, x_2, y_2 необ-

ходимо найти такие, при которых для каждого блока типа t общее число ячеек, не имеющих тип t или $empty$, было бы минимальным. Обозначим через $f_{A_i} = \sum\limits_{c \in A_i} [f(c) = data], i = 1, 2, 3, 4$, $f_D = \sum\limits_{c \in D} [f(c) = key]$ — количества объектов «чужого» типа в блоках, $[a = b] = 1$, если $a = b$.

Таким образом, имеем следующую задачу минимизации

$$\begin{cases} \sum\limits_{i=1}^{4} f_{A_i} + f_D \longrightarrow \min\limits_{x_1, y_1, x_2, y_2}; \\ f_{A_i} < \beta |A_i|, i = 1, 2, 3, 4; \\ f_D < \beta |D|; \\ x_1, x_2 \in X, x_1 < x_2; \\ y_1, y_2 \in Y, y_1 < y_2; \\ (x_1, y_1, x_2, y_2) \neq (1, 1, N + 1, M + 1). \end{cases} \tag{2.1}$$

Если задача (2.1) не имеет решения, значит рассматриваемая таблица не является статистической. В этом случае описываемый метод не применим к данной таблице.

Идея решения задачи (2.1) состоит в следующем (Алгоритм 1). Изначально все ячейки матрицы воспринимаются как ячейки данных (матрица D). Начальная ошибка полагается равной e. Затем происходит удаление столбца или строки с каждой из сторон матрицы. Вычисляется функционал ошибки e_α. Если ни одна из операций по удалению не уменьшает значение ошибки, алгоритм останавливается. По получившейся матрице D тривиально восстанавливаются значения (x_1, y_1, x_2, y_2).

2.2.2.5. Извлечение названия таблицы

Название таблицы обычно находится в тексте документа перед обрабатываемой таблицей (рис. 2.6). В простом случае достаточно проанализировать

Алгоритм 1 FindBlocks: поиск блоков данных и ключей

Вход: $A \in \{data, key, empty\}^{M \times N}$ — матрица результатов классификации;

Выход: $\gamma = (x_1, y_1, x_2, y_2)$

$D = \overline{\Theta_{MN}}$ — матрица единиц размера $M \times N$;

$E_{key} = [A = data]; E_{data} = [A = key];$ — матрицы ошибок, если все ячейки описания или данных

$E = E_{data} * D + E_{key} * \overline{D};\ e = \sum\limits_{i=1}^{N} \sum\limits_{j=1}^{M} E_{ij}; s = 0;$

повторять

для $\alpha \in \{0, \pi/2, \pi, 3\pi/2\}$

$D_\alpha = rotate(D, \alpha);$

$D_{\alpha i,1} = 0, \forall i = \overline{1, N}$ — удалить левый столбец

$E_\alpha = E_{data} * D_\alpha + E_{key} * \overline{D_\alpha};\ e_\alpha = \sum\limits_{i=1}^{N} \sum\limits_{j=1}^{M} E_{\alpha ij};$

$\alpha^* = \underset{\alpha}{\text{Argmin}} e_\alpha;$

если $e_{\alpha^*} \leqslant e$ **то**

$D = D_{\alpha^*};\ e = e_{\alpha^*};$

иначе

$s = 1;$

пока $s = 1$

Получить γ из D;

вернуть γ;

текст перед таблицей и использовать в качестве названия предложение, которое находится непосредственно перед ней.

2.14. ОСНОВНЫЕ ПОКАЗАТЕЛИ МАЛЫХ ПРЕДПРИЯТИЙ ПО ВИДАМ ЭКОНОМИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ в 2009 г.

	Число малых предприятий (на конец года)		Среднесписочная численность работников (без внешних совместителей)		Оборот малых предприятий	
	тыс.	в процентах к итогу	тыс. человек	в процентах к итогу	млрд. руб.	в процентах к итогу
Добыча полезных ископаемых	5,7	100	43,7	100	64,6	100

Рис. 2.6. Таблица и её название.

Однако в случае, когда исходные данные представлены в виде HTML кода, необходимо анализировать дерево тегов HTML. Оно однозначно определяется по корректному коду. Корнем дерева будем считать тег body. Узлами дерева являются вложенные теги и текстовые блоки. Дочерние узлы упорядочены в соответствии с порядком их следования в коде. Код HTML для таблицы на рис. 2.6 представлен на рис. 2.7.

```
<body link="#0000ff" vlink="#800080">
  <b>
    <font face="Arial" size="1">
      <p align="CENTER">
        "2.14. ОСНОВНЫЕ ПОКАЗАТЕЛИ МАЛЫХ ПРЕДПРИЯТИЙ ПО ВИДАМ ЭКОНОМИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ в 2009 г."
      </p>
    </font>
  </b>
  <table border cellspacing="1" bordercolor="#000000" cellpadding="1" width="529">
    <tbody>…</tbody>
  </table>
```

Рис. 2.7. Фрагмент HTML кода.

Для определения названия используется следующий эвристический метод, основанный на предположении, что название таблицы представляет собой одно предложение.

Ближайшее к таблице предложение считается названием и определяется по HTML дереву. Сначала для этого из исходного дерева рекурсивно удаляются все узлы, не содержащие текстовых блоков. Это делается с целью исключения из анализа несодержательных узлов дерева, используемых для

форматирования и вёрстки документа. Затем определяется ближайший узел к таблице. Для этого выполняется следующая последовательность действий:

1. Рассматриваем ближайшего предшественника узла с тегом table.
2. Если ближайший предшественник найден, остановка.
3. Иначе поднимаемся вверх и среди его дочерних узлов выбираем последний.

После этого из полученного узла извлекается последний по порядку текстовый узел. Он разделяется на предложения, и последнее предложение рассматривается как название таблицы.

2.2.2.6. Чтение таблицы

После того как была восстановлена структура таблицы и получены блоки $D, A_1, \ldots, A_4$, необходимо восстановить отображение $\mathfrak{R} : C_V \to 2^{C_K}$. Рассмотрим эту процедуру для таблицы простой структуры. Для каждой ячейки $c \in D$ построим множество ячеек $\mathfrak{R}(c) = A_1(c) \cup \cdots \cup A_4(c)$, где через $A_i(c) \subseteq A_i, i = 1, 2, 3, 4$ обозначено множество ячеек описания ячейки для $c \in D$ в блоке A_i.

Пусть ячейка c имеет координаты $\big(r_1(c), c_1(c), r_2(c), c_2(c)\big)$ на сетке $G^{M \times N}$. Ввиду того, что таблица имеет простую структуру, множества $A_1(c), \ldots, A_4(c)$ могут быть выписаны явно. Обозначим интервалы $[c_1(c), c_2(c)], [r_1(c), r_2(c)]$ через $cols(c), rows(c)$ соответственно. На рис. 2.8 $rows(a_2) \subseteq rows(c) \subseteq rows(a_1)$ и $cols(a_4) \subseteq cols(c) \subseteq cols(a_3)$.

Ячейка $c_1 \in A_h, h = 1, 3$ является ячейкой описания для ячейки $c_2 \in D$ тогда и только тогда, когда интервал строк ячейки c_1 содержит, либо содер-

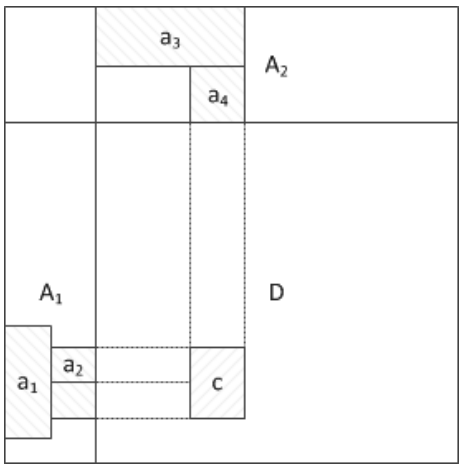


Рис. 2.8. Чтение таблицы.

жится в интервале строк ячейки c_2. Аналогично звучит условие для блоков A_2 и A_4.

Таким образом, ячейки описания во всех блоках могут быть найдены с помощью следующих выражений.

$$A_h(c) = \{a \in A_h : rows(a) \subseteq rows(c)\} \cup \{a \in A_h : rows(c) \subseteq rows(a)\}, h = 1, 3;$$

$$A_v(c) = \{a \in A_v : cols(a) \subseteq cols(c)\} \cup \{a \in A_v : cols(c) \subseteq cols(a)\}, v = 2, 4.$$

Множество ячеек описания из суперстрок R_{super}, построенных на предыдущем этапе, определим через множество $R_{super}(c)$ ячеек в ближайшей для c суперстроке. Пусть $\overline{R_{super}}(c) = \{a \in R_{super} : r_1(c) < r_2(a)\}$. Тогда $R_{super}(c) = \underset{a \in \overline{R_{super}}(c)}{\text{Argmin}} (r_1(c) - r_2(a))$. Множество ячеек описания из суперстрок определяется следующим образом: $A_R(c) = \{a \in R_{super} : r_2(a) \leqslant r_2(c_1)\}$.

Для учёта вложенных ячеек необходимо построить множество $A'_h(c) = \{c_{out}(a), a \in A_h\}, h = 1, 3$. В результате отображение $\mathfrak{R}$ для ячейки c определяется следующим образом:

$$\mathfrak{R}(c) = A_1(c) \cup A'_1(c) \cup A_2(c) \cup A_3(c) \cup A'_3(c) \cup A_4(c) \cup A_R(c).$$

Поиск ключей в тексте ячеек описания. Задача состоит в том, чтобы каждой ячейке-описанию $c \in C_K$ поставить в соответствие множество ключей $K(c) \subset \mathcal{K}$, то есть построить отображение $\mathfrak{F} : C_K \to 2^{\mathcal{K}}$. При этом необходимо учитывать, что, во-первых, допустимы ошибки в словах, а во-вторых — разделение строки на ключи не однозначно.

Будем рассматривать исходную строку $x = text(c)$ в виде последовательности слов $(x_1, \ldots, x_n)$. Для решения проблемы ошибок в словах предлагается использовать нечёткий поиск слов по словарю. В этом случае для каждого слова в строке может быть найдено несколько соответствий в словаре. Таким образом, для каждого слова x_i будет получено множество $\tilde{X}_i$ наиболее близких с точки зрения некоторого расстояния $\rho(s,t)$ слов: $\tilde{X}_i = \{x \in \mathcal{W} : \rho(x, x_i) < \tau\}$, где τ — параметр, а $\mathcal{W}$ — множество всех слов и их словоформ из $\mathcal{D} = D_1 \cup \cdots \cup D_{|\mathcal{K}|}$.

Поскольку разделение строки на ключи не однозначно, необходимо произвести поиск всевозможных подстрок, состоящих из слов $(x_{i_1}, \ldots, x_{i_2}), \forall i_1, i_2 : 1 \leqslant i_1 \leqslant i_2 \leqslant n$ исходной строки среди всех словесных описаний ключей $\mathcal{D}$. При этом вместо каждого слова x_i может использоваться любой элемент множества $\tilde{X}_i$. Каждому результату поиска будет сопоставлено словесное описание из $\mathcal{D}$ и оценка сходства, основанная на расстоянии ρ. Для построения отображения U необходимо выбрать множество попарно не пересекающихся подстрок, имеющих максимальную суммарную оценку сходства.

Пусть $\mathcal{X} = \{(\tilde{x}_i, \ldots, \tilde{x}_j), 1 \leqslant i \leqslant j \leqslant n, \tilde{x}_l \in \tilde{X}_l, \forall l = \overline{i,j}\}$ — семейство всех подстрок строки x, включающее всевозможные комбинации близких слов. Каждой строке $\tilde{x} \in \mathcal{X}$ поставим в соответствие оценку $\alpha(\tilde{x}) = \sum_{l=i}^{j} \rho(\tilde{x}_l, x_l)$. Чем больше значение $\alpha(\tilde{x})$, тем меньше $\tilde{x}$ соответствует словарю $\mathcal{W}$. Обозначим через $\theta(\tilde{x})$ множество номеров слов, входящих в строку $\tilde{x}$.

Множество всех строк из $\mathcal{X}$, являющихся описаниями ключей, обозначим через $\mathcal{X}_\mathcal{D} = \mathcal{X} \cap \mathcal{D}$. Для построения множества $\mathcal{X}_\mathcal{D}$ необходимо осуществить поиск каждой строки из $\mathcal{X}$ в множестве всех словесных описаний $\mathcal{D}$. Рассмотрим множество всех подмножеств $\mathcal{X}_\mathcal{D}$, состоящих из строк, попарно не содержащих одно и то же слово: $\overline{\mathcal{X}_\mathcal{D}} = \{S \in 2^{\mathcal{X}_\mathcal{D}} : \forall s_1, s_2 \in S \Rightarrow \theta(s_1) \cap \theta(s_2) = \varnothing\}$. Каждое множество из $\overline{\mathcal{X}_\mathcal{D}}$ состоит из строк $s_1, \ldots, s_m$, которые отстоят от соответствующих описаний ключей на расстояние $\alpha(s_1), \ldots, \alpha(s_m)$. При этом все эти строки могут не содержать некоторые из слов $(x_1, \ldots, x_n)$ исходной строки x. Для построения отображения $\mathfrak{F}$ необходимо выбрать множество строк из $\overline{\mathcal{X}_\mathcal{D}}$, содержащее наибольшее число слов и состоящее из строк, имеющих минимальное значение α. Обозначим через $\overline{\Theta}(S) = \{\overline{1,n}\} \setminus \bigcup\limits_{s \in S} \theta(s)$ множество номеров слов исходной строки x, не содержащихся ни в одной из строк множества $S \in \overline{\mathcal{X}_\mathcal{D}}$. Для поиска оптимального множества слов предлагается функционал, учитывающий пропуск слов и нечёткость поиска слов.

$$E(S) = \gamma|\overline{\Theta}(S)| + \sum_{s \in S} \alpha(s) \to \min_{S \in \overline{\mathcal{X}_\mathcal{D}}}.$$

Здесь γ — параметр, отвечающий за то, насколько сильно нужно штрафовать за пропуск слов. Его значение оптимизируется исходя из реальной выборки. Пусть S^* — множество строк, на котором достигается минимум функции $E(S)$. Обозначим через $key(s)$ ключ, имеющий словесное описание s, где s строка. Тогда $\mathfrak{F}(c) = \{key(s), s \in S^*\}$, где $c \in C_K$ — ячейка описания статистической таблицы.

Предложенный метод построения отображения $\mathfrak{F}$ состоит из двух шагов, существенно влияющих на его эффективность: нечёткий поиск слов и построение множества $\overline{\mathcal{X}_\mathcal{D}}$.

Поиск слов. Так как в тексте ячейки таблицы могут встречаться ошибки, поиск слов должен быть нечётким. Для этого можно использовать некое расстояние $\rho(a, b)$ между строками a и b и искать слова, находящиеся на сравнительно небольшом расстоянии друг от друга. Существует довольно много методов нечёткого поиска в словарях, основанных на суффиксных деревьях [37], динамическом программировании [9, 25] и индексировании.

Одним из самых известных методов является расстояние Левенштейна [9]. Оно оценивает количество необходимых операций вставки, удаления и замены символов для перевода одной строки в другую. Расстояние Левенштейна может быть рассчитано по алгоритму динамического программирования Вагнера-Фишера (алгоритм 2), сложность которого составляет $O(nm)$, где m и n — длины строк.

Алгоритм 2 LevenshteinDistance: Алгоритм Вагнера-Фишера

Вход: s_1 — строка длины m, s_2 — строка длины n;

$A(0,0) = 0$;

для $j = 1 \dots n$

$A(0,j) = A(0,j-1) + 1$;

для $i = 1 \dots m$

$A(i,0) = A(i-1,0) + 1$;

для $j = 1 \dots n$

$D(i,j) = min(A(i-1,j)+1, A(i,j-1)+1, A(i-1,j-1)+[s_1(i) \neq s_2(j)])$;

вернуть $A(m,n)$;

Для нахождения всех слов, близких с точки зрения расстояния Левенштейна к данному слову, необходимо попарно сравнить его со всеми словами из словаря $\mathcal{W}$. Он должен содержать все часто употребляемые слова языка, включая их словоформы (всего порядка 3 млн слов для русского языка),

поэтому целесообразно существенно уменьшить количество сравнений.

В рассматриваемой задаче предлагается использовать методы, основанные на индексировании всех подстрок длины q, называемых q-граммами [30]. Для строки s множество $G_q(s)$ всех q-грамм, $q < |s|$ содержит все подстроки длины q строки s, дополненной слева и справа специальным символом «*» (всего их $|s|-q+3$). Например, для строки «слово» множество 2-грамм будет равно {«*с», «сл», «ло», «ов», «во», «о*»}. Обозначим через $s[g]$ число вхождений подстроки g в строку s. Близость $\widehat{\rho}$ двух строк s_1 и s_2 определяется как

$$\hat{\rho}(s_1, s_2) = \sum_{g \in G_q(s_1) \cup G_q(s_2)} |s_1[g] - s_2[g]|.$$

Для всех слов словаря $\mathcal{W}$ строится q-граммный инвертированный индекс (отображение $\bigcup\limits_{w \in \mathcal{W}} G_q(w) \to \mathcal{W}$), позволяющий для любой q-граммы за константное время получить список слов, в которых она содержится. Процесс поиска всех ближайших строк строки s состоит из двух шагов. Сначала производится поиск каждой q-граммы данной строки в инвертированном индексе, и извлекаются строки, содержащие данную q-грамму. Затем для всех найденных строк вычисляется близость $\hat{\rho}$ со строкой s. Имея некоторый порог $\hat{\tau}$ можно выбрать множество строк $W_s = \{w \in \mathcal{W} : \hat{\rho}(s, w) < \hat{\tau}\}$.

После этого для каждого слова из $w \in W_s$ производится вычисление расстояние Левенштейна $\rho(s, w)$. Таким образом, для каждого слова s в ячейке таблицы получим множество слов $Q_s = \{w \in W_s : \rho(s, w) < \tau\}$ из словаря $\mathcal{W}$. Каждому слову $q \in Q_t$ соответствует расстояние Левенштейна $\rho(s, q) < \tau$.

Поиск всех ключей в ячейке описания. Рассмотрим задачу поиска всех ключей в ячейке описания c. Пусть $text(c) = x$ состоит из слов $x_1, \ldots, x_n$. Семейство всех подстрок строки x, включающее всевозможные комбинации

близких слов, содержит не более f^n подстрок, где $f = \max\limits_{i=\overline{1,n}} |\tilde{X}_i|$. Рассмотрим вопрос о сложности построения множества $\mathcal{X}_\mathcal{D}$.

Утверждение 2.1. *Сложность построения множества $\mathcal{X}_\mathcal{D}$ с помощью полного перебора равна $O(n^2 f^n |\mathcal{D}|)$, где $f = \max\limits_{i=1,\ldots,n} |\tilde{X}_i|$.*

Доказательство. Общее количество подстрок в строке $(x_1, \ldots, x_n)$ равно $\frac{n(n+1)}{2}$, что соответствует $O(n^2)$. Так как каждое слово может быть заменено на любой из f синонимов, то общее количество элементов множества $\mathcal{X}$ равно $O(n^2 f^n)$. Если осуществлять поиск каждой строки, сравнивая её со всеми элементами множества $\mathcal{D}$, получим оценку общего числа сравнений $O(n^2 f^n |\mathcal{D}|)$.

■

Таким образом, целесообразно использовать эффективные структуры для поиска.

Суффиксные деревья. Суффиксные деревья [43] являются одной из самых эффективных структур данных для поиска подстроки в строке с предварительной обработкой. Они позволяют осуществлять поиск за линейное время по длине запроса.

Рассмотрим произвольную строку s в алфавите Σ. Суффиксом строки s называется любая подстрока вида $s[i \ldots |s|]$, начинающаяся с позиции $i \in \overline{1, |s|}$ и заканчивающаяся последним символом. Над всеми суффиксами строки можно построить дерево (рис. 2.9) суффиксов, в котором реализуется взаимно-однозначное соответствие между всеми путями, начинающимися из корня, и суффиксами строки s. Каждому ребру дерева при этом соответствует некоторая метка, которая является подстрокой исходной строки. Для того чтобы определить, содержится ли строка p в строке s, необходимо определить, существует ли путь от корня в суффиксном дереве, построенном по

строке s, такой, что конкатенация меток дуг равняется строке p. Очевидно, что сложность такой операции линейна по длине строки p.

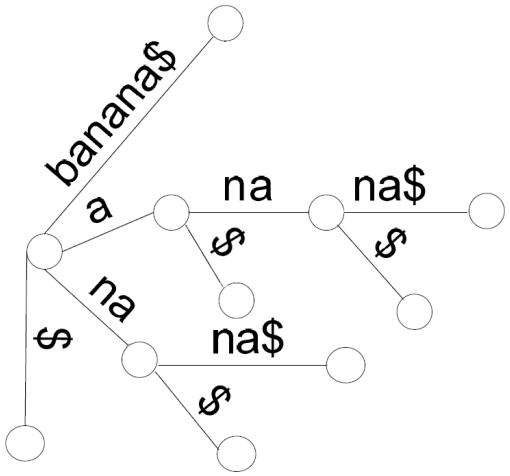


Рис. 2.9. Пример суффиксного дерева для строки «banana$».

Известны линейные алгоритмы построения суффиксных деревьев [29, 38]. Обобщенное суффиксное дерево (ОСД [2]) представляет собой суффиксное дерево, построенное для набора строк. Как показано в [2], ОСД можно построить, если дополнить каждую строку уникальным маркером конца, конкатенировать все строки и построить суффиксное дерево для получившейся строки. В [1] подробно описаны вопросы практической применимости и реализации ОСД.

В рассматриваемой задаче поиска всех ключей предлагается использовать ОСД не над строками, а над последовательностями слов. Если каждому слову $t \in \mathcal{W}$ поставить в соответствие уникальный номер $i \in 1, \ldots, |\mathcal{W}|$, то исходная строка может рассматриваться как строка в алфавите $\Sigma_{\mathcal{W}} = \{1, 2, \ldots, |\mathcal{W}|\}$. Таким образом все словесные описания ключей $D_1, \ldots, D_{|\mathcal{K}|}$ представляются в алфавите $\Sigma_{\mathcal{W}}$. ОСД, построенное над всеми словесными описаниями, позволит осуществлять поиск по всему множеству $\mathcal{D}$ за время $O(n)$, где n — число слов в строке-запросе, что позволяет избавиться от линейной зависимости от размера словаря $\mathcal{D}$. Таким образом, верно следующее

утверждение.

Утверждение 2.2. *При использовании суффиксных деревьев сложность построения множества* $\mathcal{X}_{\mathcal{D}}$ *равна* $O(n^2 f^n)$, *где* $f = \max\limits_{i=1,\ldots,n} |\tilde{X}_i|$.

Добавление новых ключей Описанный алгоритм поиска всех ключей в тексте ячеек описания не будет находить те ключи, текстовых описаний которых нет в множестве $\mathcal{D}$. Так как заполнение этого множества заранее может потребовать значительных затрат, предлагается процедура его динамического пополнения.

Все ненайденные фрагменты текста ячеек описания запоминаются и предлагаются для анализа пользователю в порядке частоты их встречаемости в разных ячейках описания и в разных таблицах. Пользователь имеет возможность использовать каждый фрагмент целиком или разделить его на части и отнести его либо к существующему ключу, добавив в соответствующее множество D_i, либо создать новый ключ, инициализировав множество его словесных описаний новым фрагментом.

2.2.2.7. Извлечение периодов времени

Рассмотрим статистическую таблицу $T = \langle G^{M \times N}, C_V, C_K, \mathfrak{R} \rangle$. Одним из определяющих компонентов статистического показателя, который строится на основе ячейки данных $c \in C_V$, является интервал времени $d = [t_1, t_2]$, где t_1 и t_2 — границы интервала. При этом левая часть интервала может быть не задана.

Границы интервала задаются в виде астрономических дат. Даты состоят из трёх компонентов: номера дня месяца, номера месяца и года, т. е. $t_i = (d_i, m_i, y_i), i = 1, 2$. Таким образом, задача определения периода времени мо-

жет быть поставлена как задача определения каждого из этих компонентов для левой и правой границы интервала d. Каждый компонент может задаваться неявно, в текстовой форме, в любой из ячеек множества $\mathfrak{R}(c)$ или в названии таблицы.

Для извлечения дат применяется система шаблонов, основанная на известных стандартах представления дат. В таблице 2.1 представлены типичные элементы этих шаблонов.

Таблица 2.1. Шаблоны для извлечения информации о дате

Шаблон	Допустимые значения
DD	$01, 02, \ldots, 31$
MM	$01, 02, \ldots, 12$
MMM	янв, фев, мар, ..., дек
MMMM	январь, февраль, март, ..., декабрь
YYYY	$0001 - 2100$

Каждому элементу ставится в соответствие множество допустимых значений. Эти элементы используются в полных шаблонах дат, часть из которых представлена в таблице 2.2. Множество полных шаблонов доступно для редактирования пользователю.

Таблица 2.2. Шаблоны полных дат

Шаблон	Комментарий
DD.MM.YYYY	ГОСТ Р 6.30-2003
YYYY-MM-DD	ISO 8601
MM/DD/YYYY	применяется в США и Канаде

Перечисленные в таблицах 2.1 и 2.2 шаблоны используются для описания

текстовых шаблонов дат. Этот список также доступен для редактирования пользователем системы. Шаблон полной даты определяется в нём словом DATE. Каждому шаблону ставится в соответствие полный набор из шести компонент обеих дат интервала. По умолчанию считается, что дата определяет правую границу интервала d. Однако на конец или начало может быть явное указание, например «на конец года» или «на начало интервала». Приведём примеры распространённых шаблонов:

- «В YYYY году»;
- «На конец MMMM», «На начало YYYY»;
- «С DATE по DATE»;
- «В DATE», «До DATE»;
- «В YYYY/YY учебном году».

Алгоритм извлечения элементов дат (Алгоритм 3) извлекает все элементы из всех ячеек таблицы, а при наличии конфликтов использует те элементы, которые ближе по некоторому расстоянию, заданному на сетке $G^{M\times N}$.

Очевидно, что этот алгоритм не обеспечивает извлечение всех элементов дат, если их нет в тексте ячеек описания таблицы. В этом случае предлагается заполнять пропуски в соответствии со следующим упорядоченным набором правил:

1) если не задан год, остановиться. Данную таблицу необходимо показать пользователю для проверки;
2) если не задан месяц, считать его равным декабрю для правого интервала, и январю — для левого;

Алгоритм 3 ExtractDate

Вход: T_s — статистическая таблица, c — ячейка данных, Θ — множество шаблонов;

Выход: $(d_1, m_1, y_1, d_2, m_2, y_2)$ — элементы интервала времени;

$\mathbf{e}^* = (\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing)$ — начальные значения для $(d_1, m_1, y_1, d_2, m_2, y_2)$

$\rho_i^* = \infty, i = \overline{1,6}$

для каждого шаблона $\theta \in \Theta$

для каждой ячейки $s \in \mathfrak{R}(c)$

$\mathbf{e} = ExtractTemplate(\theta, s)$ — применение шаблона

$\rho_i = Distance(s, c)$

для $i = \overline{1,6}$

если $e_i \neq \varnothing$ **и** $\rho*_i > \rho_i$ **то**

$e*_i = e_i; \rho*_i = \rho_i$

вернуть $(d_1, m_1, y_1, d_2, m_2, y_2)$

3) если не задан день, считать его равным последнему дню месяца для правого интервала и первому дню — для левого.

2.2.2.8. Извлечение единиц измерения

Единица измерения $U_s \subset \mathcal{K}$ является одним из компонентов статистического показателя $s = (K_s, U_s, d_s, v_s)$.

Обозначим через $U_e \subset \mathcal{K}, U_m \subset \mathcal{K}$ множества ключей, определяющие смысл единиц измерения. Первое множество содержит все единицы измерения, такие как «штука», «рубль», «км» и пр., а второе множество — все их модификаторы и мультипликаторы. Например, в U_m должны содержаться значения «тысяча», «десяток» и т.п.

Пусть для некоторой статистической таблицы T известно отображение $\mathfrak{R} : C_V \to 2^{C_K}$, которое каждой ячейке данных ставит в соответствие множество ячеек описания, и отображение $\mathfrak{F} : C_K \to 2^{\mathcal{K}}$, ставящее в соответствие каждой ячейке описания множество ключей.

Рассмотрим ячейку $a_1 \in C_V$, тогда $\mathfrak{R}(a_1) = \{b_1, \ldots, b_p\}, b_i \in C_K$ — множество ячеек описания, соответствующих ячейке a_1. Каждая ячейка b_i может содержать ключи из U_e и U_m. Обозначим через $U_e^{b_i} = \{\mathfrak{F}(b_i) \cap U_e\}$ множество ключей, являющихся единицами измерения в каждой из ячеек $b_1, \ldots, b_p$. Аналогичным образом определим U_m. Найдя все элементы единиц измерения в ячейках описания, можно собрать их вместе: $U_s = \bigcup\limits_{i=1}^{p} \left(U_e^{b_i} \cup U_m^{b_i}\right)$.

Полученное множество полностью определяет единицу измерения статистического показателя s.

2.3. Алгоритмы динамического обучения

При динамическом обучении объекты появляются по одному, причём каждый объект обрабатывается только один раз. Динамическое обучение состоит из последовательности чередующихся шагов классификации и дообучения. На стадии классификации очередного объекта алгоритму не известно значение целевого признака, однако непосредственно после классификации оно ему передаётся. Источником этого ответа могут служить действия эксперта. Это значение используется для модификации параметров алгоритма (дообучения).

Будем рассматривать объекты x_i как векторы в признаковом пространстве. Всё множество объектов обозначим через $X = \{x_1, ..., x_l\}$, а множество признаков — через $\mathcal{F} = \{f_1, ..., f_M\}$. Будем считать, что $\big(f_1(x), ..., f_n(x)\big) \subseteq \mathbb{R}^n$. Если множество значений признака конечно $|\{f_j(x)|x \in X\}| < \infty$, то f_j — номинальный признак, иначе — числовой. Множество классов обозначим через Y. В нашем случае $|Y| < \infty$. Пусть дана обучающая выборка $X^l = \{(x_i, y_i)_{i=1}^l\}$, где l — количество пар объект-ответ.

Определим важное свойство обучающей выборки, которое является одним из условий корректности алгоритма динамического обучения.

Определение 5. *Обучающая выборка X^l является непротиворечивой, если* $\nexists x_i, x_j : i \neq j$ *и* $x_i = x_j \wedge y_i \neq y_j$.

2.3.1. Известные алгоритмы динамического обучения

Среди известных алгоритмов динамического обучения для оценки качества работы и удовлетворения требованию корректности были выбраны три алгоритма: k ближайших соседей, индукции правил по множеству образцов (Rule Induction from a Set of Exemplars, RISE) [16] и ITI (Incremental Tree

Induction) [39, 40]. Выбор этих алгоритмов обусловлен тем, что они основаны на запоминании обучающей выборки. Это может обеспечить выполнение требования корректности.

Для проверки условия корректности и доли ошибок были проведены эксперименты на реальных данных для задачи распознавания типа ячейки. Обучающая выборка состояла из 20 разнообразных таблиц, в каждой из которых было около 100 ячеек. Каждый эксперимент состоял из 10 запусков, перед каждым из которых происходило случайное перемешивание таблиц, а затем ячеек в них. В каждом запуске небольшая часть исходной выборки использовалась для начального обучения алгоритма. Остальные объекты подавались алгоритмам последовательно. После классификации очередного объекта алгоритму давался правильный ответ и происходило его дообучение. Затем проверялась классификация всех ранее классифицированных объектов, чтобы удостовериться, что требование корректности выполнено.

Алгоритм k ближайших соседей основан на полном запоминании всей выборки. Обучение алгоритма заключается в добавлении нового объекта в список известных. При классификации происходит поиск k ближайших объектов и выбирается доминирующий класс. В качестве меры близости использовалось евклидово расстояние без взвешивания признаков. Очевидным недостатком данного алгоритма является скорость его работы. Результат эксперимента для этого алгоритма представлен на Рис. 2.10. Данный алгоритм позволяет получить неплохое значение ошибки на новых объектах и не делает ошибок на старых объектах.

Алгоритм RISE строит правила из объектов обучающей выборки. Классификация заключается в поиске ближайшего правила и использовании его

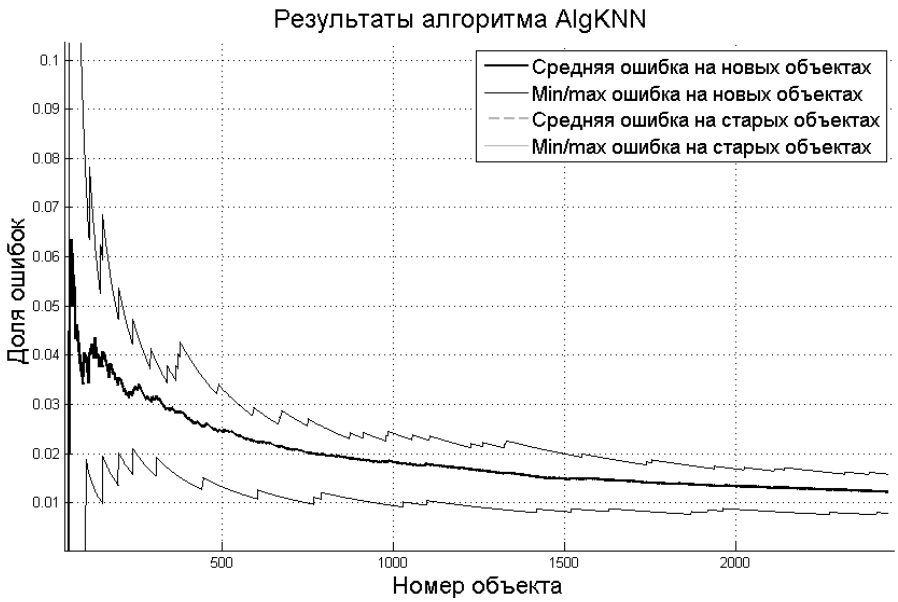


Рис. 2.10. Результат работы алгоритма KNN.

класса в качестве ответа с помощью функции расстояния между правилом R, заданным в виде конъюнкции предикатов $r_1 \wedge \cdots \wedge r_N$, и объектом $E = (e_1, \ldots, e_N)$:

$$\Delta(R, E) = \sum_{i=1}^{N} \delta^2(i),$$

где $\delta(i)$ — функция расстояния по i-му признаку:

$$\delta(i) = \begin{cases} 0, & r_i = true; \\ \text{orddist}(i), & i\text{-й признак номинальный}; \\ \text{numdist}(i), & i\text{-й признак числовой}. \end{cases}$$

Предикатом для номинального признака является индикатор принадлежности некоторому подмножеству допустимых значений S, а для числового — индикатор принадлежности отрезку $[a, b]$. Функция $\text{orddist}(i)$ принимает значения 0 или 1 в зависимости от принадлежности множеству S, а функция $\text{numdist}(i)$ равна расстоянию до отрезка $[a, b]$. Сначала из каждого объекта $e = (e_1, \ldots, e_N)$ обучающей выборки порождаются правило, состоящее из предикатов вида $r_i(x) = [x_i = e_i]$, если i-й признак номинальный и $r_i(x) = [x_i \in$

$[e_i - \epsilon, e_i + \epsilon]]$, $\epsilon > 0$, если i-й признак числовой. Далее правила обобщаются до ближайшего с точки зрения расстояния $\Delta(R, E)$ непокрытого объекта. Для каждого номинального признака в предикат добавляется значение этого признака на объекте E, а для числового — отрезок предиката расширяется таким образом, чтобы значение признака объекта E попадало в интервал. После этого происходит удаление тех правил, которые полностью поглащаются хотя бы одним другим правилом.

На Рис. 2.11 представлен результат эксперимента для алгоритма RISE. Видно, что он делает ошибки на старых объектах, поэтому его использование в системе затруднительно.

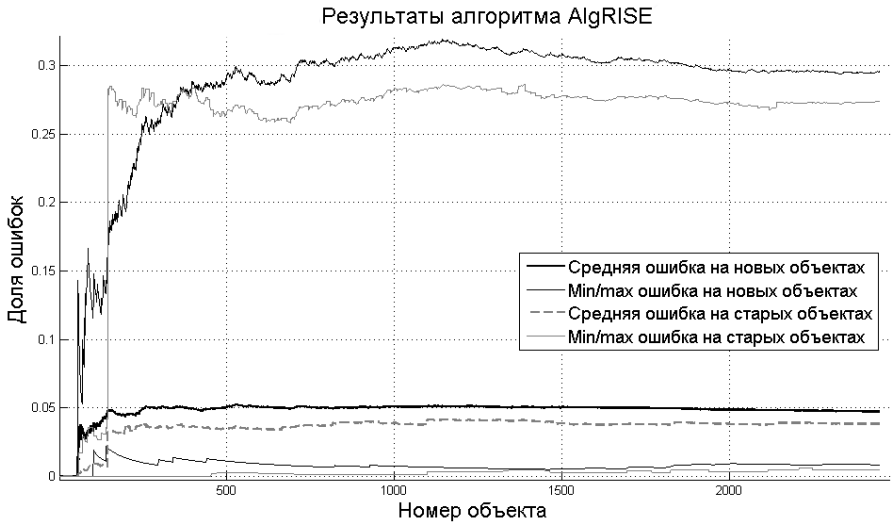


Рис. 2.11. Результат работы алгоритма RISE.

Алгоритм ITI строит бинарное решающее дерево. В узлах дерева хранятся простые предикаты для одного признака. Для номинальных признаков — это проверка на равенство одному из его значений, а для числовых — неравенство вида $[x_i < a]$. Более подробно алгоритм описывается в разделах 2.3.2 и 2.3.3.

Результаты эксперимента с алгоритмом ITI представлены на Рис. 2.12. На графике видно, что значение ошибки на новых объектах монотонно пада-

ет, а график ошибки на старых объектах совпадает с осью абсцисс.

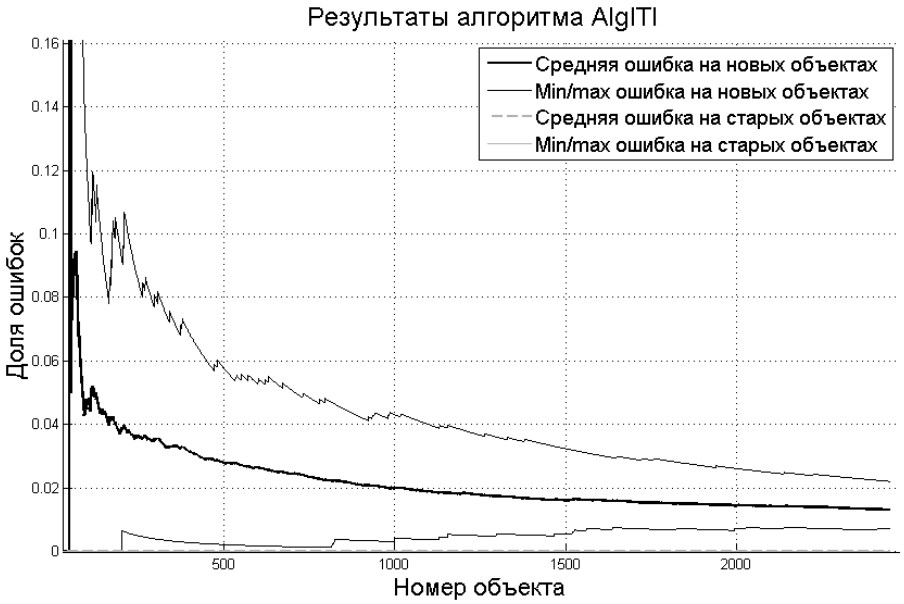


Рис. 2.12. Результат работы алгоритма ITI.

На Рис. 2.13 приведены графики ошибок на новых объектах для всех алгоритмов. Худший результат показал алгоритм RISE — среднее значение ошибки стабилизировалось на уровне 5%.

Значение ошибки на новых объектах у алгоритмов kNN и ITI было примерно одинаковым, на уровне 2%. При этом алгоритм ITI показал хорошую скорость работы. Этот алгоритм был выбран в качестве базового для дальнейших исследований.

2.3.2. Решающие деревья

Решающее дерево — алгоритм классификации $X \to Y$, который представим в виде множества вершин $T = \{\nu_i\}$, корень дерева обозначим через ν_0 (рис. 2.14). Каждой узловой вершине ν соответствует $(L_\nu, R_\nu, \beta_\nu)$, где L_ν, R_ν — ссылки на левое и правое поддеревья, а β_ν является логическим условием. Будем рассматривать $\beta_\nu(x) = [f_j(x) < a]$ в случае числового признака, и $\beta_\nu(x) = [f_j(x) = a]$ — в случае номинального. Каждой вершине ν

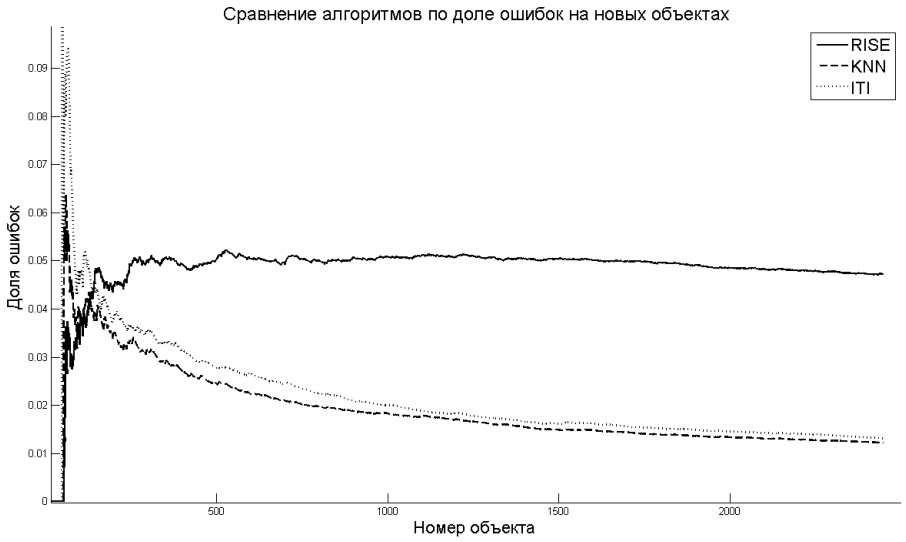


Рис. 2.13. Сравнение алгоритмов по числу ошибок на новых объектах.

соответствует метка класса.

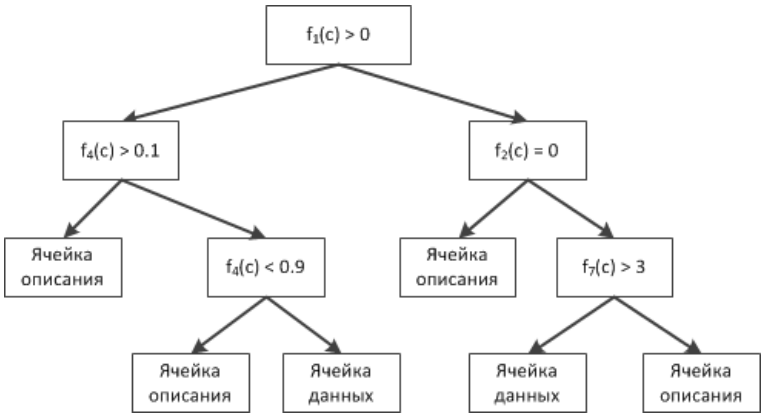


Рис. 2.14. Пример простого дерева.

Классификация осуществляется путём последовательной проверки условий в узлах дерева, начиная от корня дерева. Если $\beta_\nu(x) = 0$, то $\nu = L_\nu$, иначе — $\nu = R_\nu$.

Обучение алгоритма состоит в рекурсивном разбиении выборки на две части по наиболее информативному логическому условию (Алгоритм 4). На каждом этапе выбор наилучшего условия осуществляется перебором всевозможных предикатов, то есть перебором всевозможных разбиений выборки и вычислением информативности каждого из них. Информативность может

быть вычислена с помощью энтропийного критерия информативности, индекса Джини или других эвристических критериев [19].

Алгоритм 4 TrainTree

Вход: $X^l = \{(x_i, y_i)_{i=1}^l\}$ — множество пар объект-ответ;

Выход: ν — корень дерева;

если в X^l представлены объекты как минимум 2 классов **то**

$\beta =$ наилучшего предикат, разделяющий X^l

$L := TrainTree(\{x \in X^l | \beta(x) = 0\})$; (обучаем левое поддерево)

$R := TrainTree(\{x \in X^l | \beta(x) = 1\})$; (обучаем правое поддерево)

вернуть $\nu = \langle L, R, \beta \rangle$;

иначе

вернуть $\nu = \langle y \rangle$, y — класс объектов, представленных в X^l;

Пусть имеется подвыборка $A = \{(a_i, b_i)\} \subseteq X^l, 1 \leqslant i \leqslant |A|$ и её разбиение на два непересекающихся подмножества A_1, A_2, $A_1 \cap A_2 = \varnothing$. Обозначим относительное число объектов класса y через $p_y(A)$:

$$p_y(A) = \frac{1}{|A|} \sum_{i=1}^{|A|} [b_i = y].$$

Информативность на основе индекса Джини (Gini index)

$$I_{Gini} = \frac{|A_1|}{|A|} g(A_1) + \frac{|A_2|}{|A|} g(A_2),$$

$$g(A) = 1 - \sum_{y \in Y} p_y^2(A).$$

Информационный выигрыш (information gain)

$$I_{Gain} = H(A) - \frac{|A_1|}{|A|}H(A_1) + \frac{|A_2|}{|A|}H(A_2);$$
$$H(X) = -\sum_{y \in Y} \frac{p_y(X)}{|X|} \log_2 \frac{p_y(X)}{|X|}.$$

2.3.3. Инкрементное построение дерева решений

Алгоритм Incremental Tree Induction (ITI, [39, 40]) предполагает инкрементное построение бинарного дерева решений. В узловой вершине ν инкрементного дерева хранится пятёрка $\langle L_\nu, R_\nu, \beta_\nu, X_\nu, s_\nu \rangle$, где L_ν, R_ν, β_ν, как и в обычном дереве решений, левая, правая вершина и предикат соответственно, $X_\nu \subset X^l$ — список объектов, прошедших через узел ν, s_ν — состояние узла. Если через него прошли новые объекты, то $s_\nu = 1$, иначе $s_\nu = 0$.

Перед началом инкрементного обучения производится обучение на части выборки $X_0 \subset X^l$. Далее начинается этап инкрементного обучения. После классификации очередного объекта x происходит его встраивание в дерево (Алгоритм 5). Начиная от корневого узла происходит последовательный переход к дочерним узлам в соответствии с предикатами, пока очередной узел не окажется листом.

В каждом посещённом узле ν устанавливается состояние $s_\nu = 1$ и к списку объектов добавляется новый объект x. В случае если класс нового объекта совпадает с классом, установленным в листе то он добавляется к списку объектов. Если не совпадает — происходит поиск наилучшего разбиения объектов, хранимых в листе, и он превращается в узел, для которого строятся потомки. Таким образом, в корне дерева хранится информация обо всей выборке; на следующем уровне в каждом узле или листе хранится информация, которая в совокупности даёт информацию обо всей выборке, и т. д.

Алгоритм 5 AddToTree: добавление объекта в дерево

Вход: T, x, y — обученное дерево, объект для классификации, правильный ответ;

Выход: T — обновленное дерево;

$\nu := \nu_0$;

пока ν — узловая вершина

$X_\nu := X_\nu \cup x$;

$s_\nu := 1$;

если $\beta_\nu(x) = 0$ **то**

$\nu := L_\nu$; (переход влево)

иначе

$\nu := R_\nu$; (переход вправо)

если $y = c_\nu$ **то**

$X_\nu := X_\nu \cup x$;

иначе

$Z := X_\nu \cup x$; (список объектов в узле)

$\beta_\nu =$ наилучший предикат для Z;

$L_\nu := TrainTree(\{z \in Z | \beta_\nu(z) = 0\})$; (обучаем левое поддерево)

$R_\nu := TrainTree(\{z \in Z | \beta_\nu(z) = 1\})$; (обучаем правое поддерево)

$\nu := \langle L, R, \beta, X_\nu, s_\nu \rangle$;

вернуть T

Лемма 2.3. *Решающее дерево, построенное на основе алгоритма 5 по непротиворечивой выборке, является корректным алгоритмом классификации.*

Доказательство. Алгоритм 5 оставляет неизменными условия в узлах дерева. При добавлении нового объекта x он проходит по дереву от корня к листьям в соответствии с предикатами, установленными в узлах. Если он по-

падает в лист ν с объектами другого класса, то ν превращается в узел. По построению, условие для нового узла будет выбрано так, что одному из листьев будет соответствовать множество объектов X_ν, а второму — добавляемый объект x с соответствующими метками классов. Таким образом, результат классификации объектов X_ν не изменится, поэтому полученное дерево является корректным алгоритмом классификации. ■

В ходе работы алгоритма получается дерево, отличное от того, которое было бы построено по тому же множеству объектов, причём такое дерево давало бы лучший результат. Для обеспечения наилучшего качества при добавлении нового объекта можно было бы перестраивать дерево заново, однако это вычислительно не эффективно. В [40] описывается способ инкрементного преобразования дерева, который заключается в проверке того, что во всех узлах установлены предикаты, имеющие наилучшую информативность. Если это не так, то происходит поиск нового условия и транспозиция дерева, описанная в Алгоритме 6. В [40] предлагается выполнять эти операции после добавления каждого нового объекта. Однако поиск лучшего предиката имеет высокую вычислительную сложность — его выполнение занимает до 90% времени работы всего алгоритма.

Исходя из этого, на практике операция транспозиции выполняется не после добавления каждого нового объекта в дерево, а с некоторым периодом.

Алгоритм 6 ITI.Transpose

Вход: ν, β — обученное поддерево, новый предикат;
Выход: ν — обновленное поддерево;
если $L_\nu = \varnothing$ OR (L_ν — лист) AND $R_\nu = \varnothing$ OR (R_ν — лист) **то**
$\beta_\nu := \beta$; (оба поддерева листы, пустые, либо одно пустое, одно лист)
$\nu := ReAddObjects(\nu, \nu)$; (перераспределяем объекты в узле согласно новому условию)
вернуть ν;
если $L_\nu = \varnothing$ OR (L_ν — лист) **то**
$R_\nu := ITI.Transpose(R_\nu, \beta)$; (только левое поддерево лист или пустое)
$\nu := R_\nu$;
$\nu := ReAddObjects(L_\nu, \nu)$;
вернуть ν;
если $R_\nu = \varnothing$ OR (R_ν — лист) **то**
$L_\nu := ITI.Transpose(L_\nu, \beta)$; (только правое поддерево лист или пустое)

$\nu := L_\nu$;
$\nu := ReAddObjects(R_\nu, \nu)$;
вернуть ν;
$R_\nu := ITI.Transpose(R_\nu, \beta)$; (выполняем транспозицию в поддеревьях)
$L_\nu := ITI.Transpose(L_\nu, \beta)$;
$\alpha_\nu := \beta_\nu$; («поднимаем» предикат β из дочерних поддеревьев, а в них устанавливаем старый предикат из ν)
$\beta_\nu := \beta_{L_\nu}$;
$\beta_{L_\nu} := \alpha_\nu$;
$\beta_{R_\nu} := \alpha_\nu$;
$\gamma := R_{L_\nu}$; (меняем местами 2 соответствующих подподдерева)
$R_{L_\nu} := L_{R_\nu}$;
$L_{R_\nu} := \gamma$;
$RestoreInformationByChildren(L_\nu)$; (восстанавливаем информацию в дочерних узлах на основе информации в их обновленных поддеревьях)
$RestoreInformationByChildren(R_\nu)$;
вернуть ν;

На разных задачах можно наблюдать разный эффект от использования транспозиции. Можно заключить, что наличие транспозиции наиболее значимо для задач со сложной геометрией классов: несколько компонент связности, вложенность друг в друга, пересечения, или наличие сложной границы. Это

объясняется тем, что чаще всего начальная обучающая выборка X_0 определяет геометрию классов задачи не полностью.

2.4. Алгоритм RIF

Основные затраты при построении инкрементного дерева и выполнении транспозиции приходятся на поиск наилучшего предиката, то есть перебора разбиений. Одним из способов существенного сокращения этих затрат является сокращение числа признаков, используемых при переборе.

Алгоритм 7 RIF.Train: обучение композиции

Вход: $X^l = \{(x_i, y_i)_{i=1}^l\}, K, M$ — множество пар объект-ответ, число деревьев, число используемых признаков;

Выход: $RIF = \{\langle RITree_i, M_i\rangle\}_{i=1}^K$ — множество пар дерево-признаковое подпространство;

для $i = 1, \ldots, K$

$M_i :=$ случайное подмножество признаков из M элементов;

$RITree_i := TrainTree(X^l, M_i)$;

Алгоритм Random Forest [13] представляет собой композицию решающих деревьев. Он основывается на двух принципах: бэггинге и методе случайных подпространств. Применительно к решающим деревьям это означает, что каждое новое дерево строится по новой случайной выборке с повторениями (полученной из исходной) и по случайному поднабору признаков.

В работе предлагается новый алгоритм (Алгоритмы 7, 8) построения случайного инкрементного леса деревьев (Random Incremental Forest, RIF), результатом работы которого является композиция случайных инкрементных деревьев Random Incremental Tree (RIT).

Алгоритм 8 RIF.Update: дообучение композиции новым объектом

Вход: $RIF = \{\langle RITree_i, M_i\rangle\}_{i=1}^{K}, x, y$ — обученная композиция, новый объект и правильный ответ;

Выход: $RIF = \{\langle RITree_i, M_i\rangle\}_{i=1}^{K}$ — обновлённая композиция;

для $i = 1, \ldots, K$

$RITree_i := AddToTree(RITree_i, x, y)$; — обновление каждого дерева, простое внедрение объекта;

вернуть RIF;

RIF можно представить в виде множества пар — дерево, набор признаков, $RIF = \{\langle RITree_i, M_i\rangle\}_{i=1}^{p}$, где $RITree_i$ — инкрементное дерево, построенное по случайному набору признаков $M_i \subseteq \mathcal{F}$.

2.4.1. Обучение

По начальной обучающей выборке строится композиция решающих деревьев по аналогии с Random Forest. Для построения каждого дерева используется одна и та же исходная выборка без перемешивания с повторениями, от принципа бэггинга приходится отказаться для обеспечения корректности композиции.

Для каждого дерева случайно генерируется поднабор из $M \leqslant |\mathcal{F}|$ признаков. Каждое дерево строится по стандартному алгоритму построения дерева решений с небольшим изменением. При поиске наилучшего разбиения выбирается случайный признак из поднабора признаков. Таким образом проверяется только один признак.

2.4.2. Классификация и внедрение нового объекта

Объект даётся на классификацию всем деревьям в композиции и относится к тому классу, за который проголосовало большее число деревьев. Внедрение нового объекта происходит во все деревья композиции.

Аналогично алгоритму Incremental Tree Induction, информация об объекте сохраняется в узлах по мере прохождения его по дереву, и если он оказывается в листе, содержащем объекты другого класса, происходит превращение листа в узел (Алгоритм 8).

В некоторых случаях по тому набору признаков, который закреплён для каждого дерева, невозможно построить разбиение для некоторого множества объектов — все объекты на этом наборе признаков принимают одинаковые значения. В таком случае необходимо выполнить перебор всех признаков и для такого узла сделать исключение — признак в найденном разбиении не будет входить в набор, закреплённый для дерева. Однако такое случается довольно редко и, как правило, возможно лишь когда в подвыборке находится очень малое число объектов, то есть такое исключение не оказывает значительного влияния на быстродействие.

Тот факт, что каждое дерево в композиции является простым инкрементным деревом, позволяет нам доказать следующую лемму.

Лемма 2.4. *Композиция случайных инкрементных деревьев, построенная по непротиворечивой выборке, является корректным алгоритмом классификации.*

Доказательство. В соответствии с леммой 2.3 каждое инкрементное дерево в композиции является корректным. Поэтому во время классификации объекта обучающей выборки все деревья дадут одинаковый результат, соответству-

ющий условию корректности. Таким образом, вся композиция представляет собой корректный алгоритм. ■

2.4.3. Отбор деревьев

Так как набор признаков в каждом дереве случаен, получающиеся в ходе инкрементного обучения деревья могут сильно отличаться по качеству. В случае длинных обучающих выборок можно осуществить отбор деревьев по некоторому критерию качества.

Для сбора статистики каждое дерево композиции дополняется двумя показателями: τ — число объектов, которые были классифицированы деревом, и μ — число ошибок дерева. Эти показатели обновляются после классификации и внедрения каждого нового объекта.

2.4.3.1. Процедура отбора деревьев

Введём несколько параметров.

K_0 — начальное число деревьев, стоит выбирать разумно большим, в зависимости от темпа отбора и размеров задачи.

W — число худших деревьев, которые будут удалены.

B — число лучших деревьев, на основе которых будут строиться новые деревья. При построении нового дерева признаки случайно выбираются из поднабора признаков, на которых построены лучшие деревья. Будем осуществлять взвешивание признаков — если некоторый признак оказался в признаковых наборах нескольких деревьев, то его вес и соответственно вероятность его выбора в набор для нового дерева увеличивается.

ΔK — число новых деревьев, которые будут построены на основе признакового подпространства лучших деревьев.

Алгоритм 9 RIFTS.SelectTrees: Процедура отбора деревьев

Вход: $RIFTS = \{\langle RITree_i, M_i\rangle\}_{i=1}^{K}, X^l, W, B, \Delta K, K_{min}, \tau_0$ — обученная композиция, вся доступная обучающая выборка, параметры отбора деревьев;

Выход: $RIFTS = \{\langle RITree_i, M_i\rangle\}_{i=1}^{M}$ — обновлённая композиция;

для всех деревьев с номерами $i = 1, \ldots, K : \tau_i \geqslant \tau_0$

вычислить качество q_i;

Отсортировать деревья по убыванию $q_i, i = 1, \ldots, K$;

Удалить $min(W, K - K_{min})$ последних деревьев;

$A = \bigcup\limits_{i=1,\ldots,B} M_i$; — признаки лучших деревьев;

для $j = 1, \ldots, \Delta K$

$M_j =$ случайные M признаков из A;

$RIT_j =$TrainTree(X^l, M_j);

$RIFTS.Add(RIT_j)$; — добавление нового дерева в композицию;

вернуть $RIFTS$;

Процесс отбора деревьев представляет собой последовательность следующих операций (Алгоритм 9). Вычисляется «качество» дерева согласно некоторому критерию, выбирается W худших деревьев, затем они удаляются, строятся ΔK новых деревьев на основе лучших B деревьев. Причем построение новых деревьев происходит по всей доступной на момент построения выборке (начальная обучающая выборка и объекты, добавленные в процессе динамического обучения). Параметры τ, μ у построенных деревьев инициализируются нулями. Процедура отбора деревьев выполняется с заданным периодом.

Значение $W - \Delta K$ характеризует темп отбора, значение B характеризует «ширину» отбора. Чем меньше B, тем меньше набор признаков для выбора

при построении новых деревьев.

Кроме того, имеет смысл определить минимальное «время жизни» τ_0 для деревьев и минимальное число деревьев K_{min}. Слишком «молодые» деревья не участвуют в процедуре отбора, так как их статистика малоинформативна. Процедура отбора не выполняется и в том случае, если число деревьев достигло минимума, то есть существует риск потери качества классификации.

Рассмотрим вопрос о влиянии процедуры отбора деревьев на корректность композиции деревьев.

Теорема 1. *Композиция деревьев, полученная в результате работы процедуры отбора деревьев и построенная по непротиворечивой выборке, является корректным алгоритмом классификации.*

Доказательство. По лемме 2.4 композиция инкрементных деревьев является корректным алгоритмом. Новые инкрементные деревья будут также корректны по лемме 2.3. Таким образом, полученная после отбора деревьев композиция будет состоять из корректны деревьев, из чего следует корректность всей композиции. ■

2.4.3.2. Оценка качества дерева

Так как деревья строятся в разные моменты и не ошибаются на своих обучающих выборках, для их сравнения недостаточно учитывать только величину μ (очевидно, построенные позже деревья всегда будут иметь значение μ меньше), нужно учитывать и величину τ.

Деревья можно сравнивать по относительному числу ошибок (величина $\frac{\tau}{\mu}$). Рассмотрим два дерева, которые имеют одинаковую относительную ошибку μ, но одно из деревьев имеет значительно большую величину τ. Такое дерево кажется более надежным, но это никак не отражается в таком

критерии. Можно выводить различные эвристические формулы с участием величин τ и μ, но адекватность критериев будет, как правило, зависеть от размеров задачи. Далее будет рассмотрен универсальный статистический критерий, который не зависит от размеров задачи, разумно отдает предпочтение деревьям с большим τ.

Статистический критерий качества дерева. Пусть классификация объекта — случайная величина η из распределения Бернулли с параметром p.

$$\eta \sim Ber(p) = \begin{cases} 1, & \text{ошибка, с вероятностью } p \\ 0, & \text{успех, с вероятностью } p \end{cases}$$

Пусть имеется n испытаний и m ошибок. Тогда по теореме Муавра-Лапласа сумма случайных величин имеет асимптотически нормальное распределение с параметрами np, $np(1-p)$, обозначим $(1-p) = q$:

$$\sum_{i=0}^{n} \eta_i \sim N(np, npq) \Rightarrow \frac{\sum_{i=0}^{n} \eta_i - np}{\sqrt{npq}} \sim N(0,1).$$

Рассмотрим доверительный интервал с уровнем доверия α:

$$P\left(-\Phi_{\frac{1-\alpha}{2}} \leqslant \frac{\sum_{i=0}^{n} \eta_i - np}{\sqrt{npq}} \leqslant \Phi_{\frac{1+\alpha}{2}}\right) = \alpha \Rightarrow -\Phi_{\frac{1-\alpha}{2}} < \frac{\sum_{i=0}^{n} \eta_i - np}{\sqrt{npq}}.$$

Здесь Φ_α — квантиль стандартного нормального распределения порядка α. Заменим величину $\sqrt{pq}$ исправленной выборочной дисперсией S_1. Для корректности такой замены нам необходима состоятельность и несмещенность, чем мы и пользуемся. Получаем

$$-\Phi_{\frac{1-\alpha}{2}} < \frac{\sum_{i=0}^{n} \eta_i - np}{\sqrt{n} S_1} \Rightarrow p < \frac{\sum_{i=0}^{n} \eta_i + \Phi_{\frac{1-\alpha}{2}} \sqrt{n} S_1}{n}.$$

Заметим, что $\sum_{i=0}^{n} \eta_i = m$. Тогда получим

$$p < \frac{m}{n} + \frac{\Phi_{\frac{1-\alpha}{2}} \sqrt{n} S_1}{n}.$$

Таким образом, получена оценка вероятности ошибки через доверительный интервал уровня доверия α. Второе слагаемое позволяет при одинаковых относительных ошибках отдавать предпочтение дереву с большим n, как более надежному. Вычислим критерий качества в явном виде через m и n. Посчитаем выборочную дисперсию без учета исправления, отличие лишь в коэффициенте $\frac{n-1}{n}$ и большого качественного смысла оно не несет. Тогда имеем

$$p = \frac{m}{n} + \frac{\Phi_{\frac{1-\alpha}{2}} \sqrt{n} S_0}{n}.$$

С учётом

$$S_0^2 = \bar{\mathbb{E}}(\eta_i - \bar{\mathbb{E}}\eta_i)^2 = \bar{\mathbb{E}}(\eta_i - \frac{m}{n})^2 =$$
$$= \frac{\sum_{i=1}^{n} [\eta_i = 1](\eta_i - \frac{m}{n})^2 + \sum_{i=1}^{n} [\eta_i = 0](\eta_i - \frac{m}{n})^2}{n} =$$
$$= \frac{m(n-m)^2 + (n-m)m^2}{n^3},$$

получаем

$$\frac{\Phi_{\frac{1-\alpha}{2}} \sqrt{n} S_0}{n} = \frac{\Phi_{\frac{1-\alpha}{2}} \sqrt{n} \sqrt{\frac{m(n-m)^2 + (n-m)m^2}{n^3}}}{n} =$$
$$\frac{\Phi_{\frac{1-\alpha}{2}} \sqrt{nm(n-m)}}{n^2} = \Phi_{\frac{1-\alpha}{2}} \sqrt{\frac{m(n-m)}{n^3}}.$$

В итоге

$$p = \frac{m}{n} + \Phi_{\frac{1-\alpha}{2}} \sqrt{\frac{m(n-m)}{n^3}}.$$

Полученное выражение для p используется в качестве критерия качества дерева (при $n = \tau$, $m = \mu$).

2.5. Основные выводы

1. Решение задачи извлечения статистических показателей из таблиц сведено к последовательному решению задач распознавания типа ячеек, распознавания суперстрок, распознавания вложенных ячеек, извлечения названия таблицы из окружающего текста, поиска содержимого ячеек и названия таблицы в словарях, извлечения периода времени, извлечения единиц измерения и построения множества статистических показателей.

2. Первые три задачи являются задачами классификации, и для их решения предлагается использовать методы обучения по прецедентам. Объектами в этих задачах являются соответственно: ячейки, строки и некоторые пары соседних ячеек таблиц. Предлагаемые признаковые пространства учитывают физическое расположение и размер ячеек и строк, особенности текстового содержимого ячеек и наличия специальных символов. Для корректного решения задачи классификации типа ячеек поставлена задача нахождения оптимального покрытия ячеек блоками данных и описания, а также предложен алгоритм её решения.

3. Задача поиска ключей в ячейках описания ставится как задача минимизации функционала, оценивающего покрытие исходного текста ячейки множеством непересекающихся словесных описаний ключей с учётом пропуска слов и оценики их близости. Для поиска словесных описаний предлагается использовать обобщённые суффиксные деревья. Доказано утверждение о том, что это позволяет сделать поиск независимым от размера множества всех словесных описаний ключей.

4. Для задач извлечения названия таблицы и построения периода времени

предлагаются эвристические алгоритмы, основанные на анализе HTML кода и использовании шаблонов дат.

5. Проведённое сравнение некоторых известных динамических алгоритмов на задаче распознавания типа ячеек показало целесообразность использования алгоритма инкрементного построения решающих деревьев ITI. Доказывается корректность алгоритма на непротиворечивой обучающей выборке. Для снижения эффекта переобучения в нём применяется операция транспозиции, требующая значительных вычислительных затрат.

6. Предложен новый метод инкрементного построения композиции случайных инкрементных деревьев. Каждое дерево строится на подмножестве признаков и условие ветвления выбирается только для одного случайного признака.

7. Предложена параметрическая процедура отбора деревьев в композиции. Для оценки качества дерева используется верхняя оценка вероятности ошибки дерева.

Глава 3

Система поиска статистических данных

3.1. Архитектура системы извлечения данных

Система извлечения данных состоит из нескольких основных компонентов:

- модуля предварительной обработки таблиц;
- модуля генерации признакового описания объектов;
- модуля классификации и динамического обучения;
- интерфейса оператора;
- модуля построения статистических показателей;
- базы данных извлечённых статистических показателей.

Общая схема процесса обработки таблиц показана на рис. 3.1. В начале обработки происходит генерация признаков по всем задачам классификации (тип ячеек, суперстроки, вложенные ячейки), затем выполняется классификация для распознавания логической структуры таблиц и происходит извлечение статистических показателей. После этого может быть выполнена верификация результатов экспертами. В результате появляются новые объекты обучающей выборки, на основе которых происходит дообучение алгоритмов.

Рис. 3.1. Процесс динамического обучения

3.1.1. Предварительная обработка таблиц

Так как исходными данными для системы являются таблицы, полученные из интернета, необходимо произвести их очистку от стилевого оформления и особенностей вёрстки. Для этого осуществляется разбор (parsing) всех таблиц на HTML странице. Разбор осуществляется на основе набора предопределённых регулярных выражений.

Для последующего анализа выбираются таблицы, которые удовлетворяют следующим требованиям:

- не содержат в тексте ячеек других таблиц;
- доля ячеек, содержащих числовое значение, составляет более половины.

В результате каждая таблица преобразуется в множество координат ячеек (r_1, c_1, r_2, c_2) и соответствующих им текстовым значениям.

Для хранения таблиц используется Javascript Object Notation (JSON). Результат разбора таблицы сохраняется в специальный объект, а затем сериализуется в виде JSON в отдельный файл. Этот объект содержит три поля: список ячеек, число строк и число столбцов в таблице. Каждый элемент списка ячеек также является объектом и состоит из исходного текста ячейки, текста ячейки после фильтрации HTML тегов и замены символов юникод, а также четырёх координат.

3.1.2. Генерация признаков

Для генерации признаков реализован базовый класс, позволяющий быстро добавлять новые задачи классификации, если появится такая потребность (рис. 3.2).

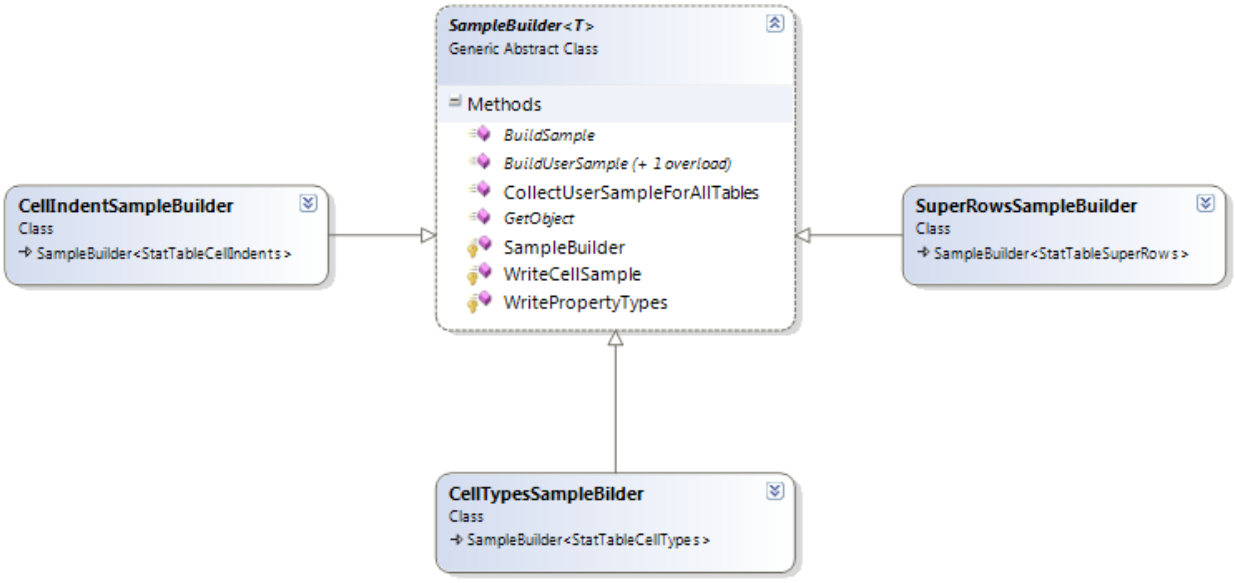


Рис. 3.2. Классы для генерации признаков

Некоторые признаки в задачах основываются на анализе слов в тексте

ячеек описания. При анализе содержимого строки нельзя полагаться на правильную расстановку пробельных символов и других синтаксических разделителей. Для корректного разделения на слова используется специальная процедура, учитывающая символьный контекст и позволяющая корректно определять десятичную запись чисел, числовых диапазонов, и слова, написанные через дефис.

3.1.3. Динамическое обучение и классификация

Для каждой задачи классификации имеется база объектов выборки, которая хранится в текстовых файлах. Каждый объект может иметь известный ответ (метку класса), либо пропуск, означающий, что объект ещё не был классифицирован. Кроме того, каждому объекту поставлен в соответствие идентификатор таблицы, по которой он был сгенерирован. После добавления новой таблицы генерация признаков пополняет файлы с выборкой. Так как разные таблицы могут порождать пересекающиеся множества объектов, всегда хранится только один из этих объектов, а все его дубликаты представляются в виде ссылки на оригинальный объект.

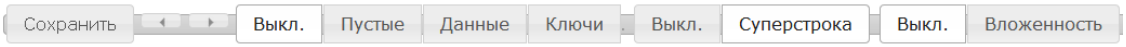
Алгоритмы динамического обучения ITI и RIF реализованы на языке MATLAB. Древовидная структура деревьев определяется на матрицах, на работу с которыми нацелен этот язык программирования. Особенность этих алгоритмов построения инкрементных деревьев состоит в том, что каждый узел и лист дерева хранит множество объектов, которые прошли через него во время обучения. Эта информация хранится в виде списков номеров объектов в исходной выборке. Необходимая часть выборки загружается в оперативную память по требованию, когда нужно перестроить дерево или определить самый информативный предикат для разделения подмножества объектов.

3.1.4. Интерфейс оператора

	1990	1995	1996	1997	1998	1999
	Городское население					
Российская Федерация	35,7	46,0	48,7	49,2	52,1	55,9
Северный район	32,0	44,5	48,2	47,7	51,5	59,0
Республика Карелия	30,6	48,3	50,5	50,3	54,6	58,1
Республика Коми	36,3	47,9	53,4	55,2	56,7	64,9
Архангельская область	27,0	41,7	43,2	44,9	50,0	62,7

Рис. 3.3. Интерфейс разметки структуры таблицы

В качестве одного из компонентов предлагаемой технологии был реализован интерфейс оператора, позволяющий верифицировать результаты автоматической классификации. Интерфейс представляет из себя веб-приложение, реализованное на основе технологии Microsoft ASP.NET MVC с использованием библиотеки jQuery. Оно позволяет пользователям просматривать результаты обработки статистических таблиц и выполнять действия по их разметке.



	1995	2000	2005	2007	2008
	Человек				
Умершие в возрасте до 1 года от всех причин	24840	19286	16073	14858	14436
в том числе:					
от некоторых инфекционных и паразитарных болезней	1741	1162	729	601	511
от болезней органов дыхания	3319	2073	1212	1101	1026
от болезней органов пищеварения	148	117	103	100	72
от врожденных аномалий (пороков развития), деформаций и хромосомных нарушений	5725	4460	3948	3610	3486
от отдельных состояний, возникающих в перинатальном периоде	10767	8517	7195	6809	6650
от внешних причин смерти	1385	1222	1109	1012	999
	На 10 000 родившихся живыми				
Умершие в возрасте до 1 года от всех причин	181,2	153,3	109,7	93,6	85,2
из них:					
от некоторых инфекционных и паразитарных болезней	12,7	9,2	5,0	3,8	3,0

Рис. 3.4. Разметка вложенных ячеек

На рис. 3.3 и 3.4 приведены примеры отображения таблиц из коллекции Росстата в веб интерфейсе оператора. Оранжевым цветом отмечаются

ячейки описания, синим — ячейки данных, серым — неинформативные, а зелёным — суперстроки. Вложенные ячейки обозначаются стрелками влево и вправо. В приведённом примере на рис. 3.3 первый ряд ячеек данных отображается с ошибкой. Для исправления ошибки классификации типа ячеек оператор может нажать на кнопку «Пустые», «Данные» и «Ключи», и выбрать область ячеек двумя кликами: по верхнему левому углу и по нижнему правому. Аналогичным образом, с помощью кнопки «Суперстроки», оператор имеет возможность переключить статус сроки. Переключения статуса вложенности (наличие/отсутствие стрелки влево или вправо) изменяется нажатием по ячейке, когда включён режим «Вложенность».

Основной целью оператора при работе с каждой таблицей является достижение правильности её разметки. После нажатия на кнопку «Сохранить» происходит передача действий оператора на сервер, где выполняется генерация признакового описания объектов для этой таблицы.

3.1.5. База данных

Для хранения извлечённых из таблиц статистических показателей разработана база данных, схема которой представлена на рис. 3.5. Для каждой таблицы перечислены поля и используемые типы данных. Линиями, связывающими таблицы между собой, обозначены внешние ключи.

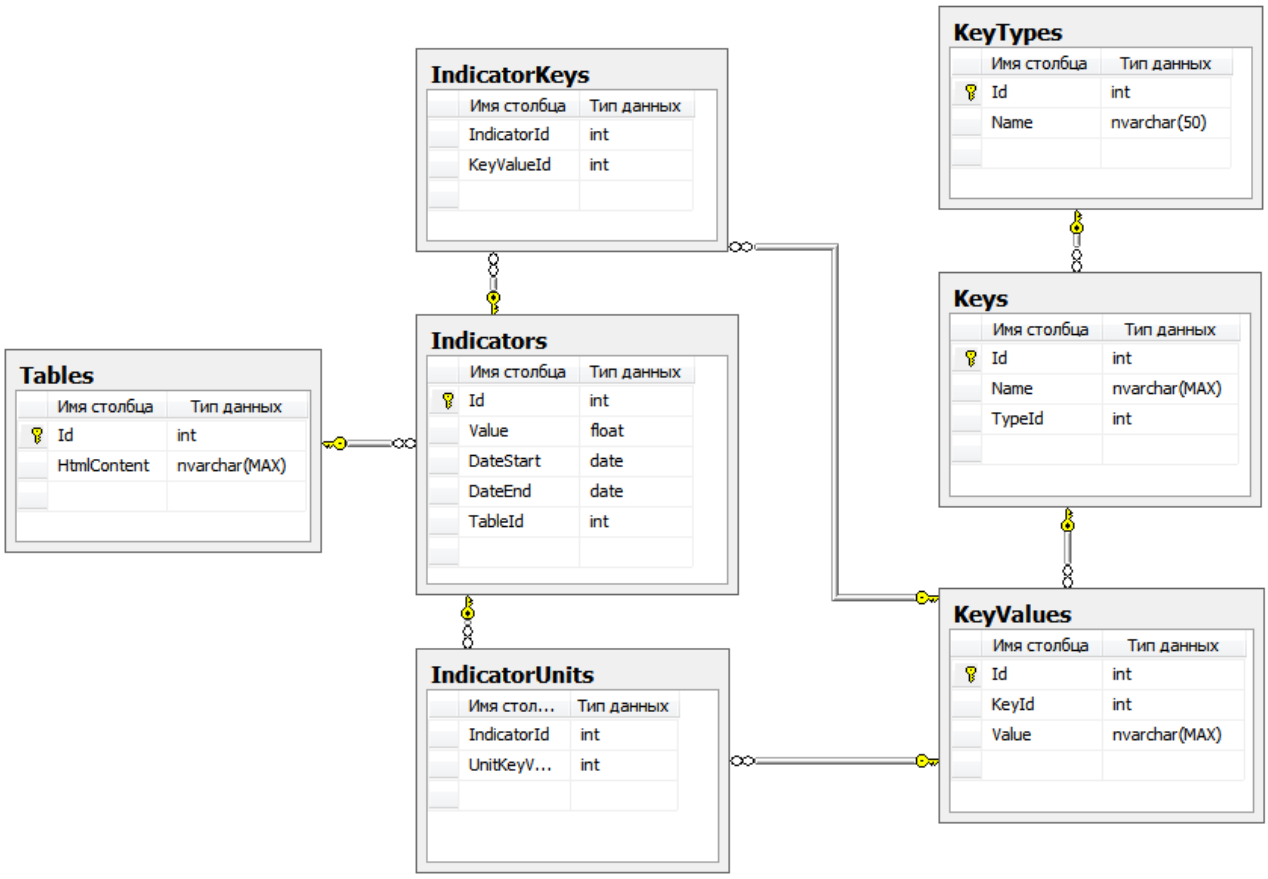


Рис. 3.5. Схема базы данных для хранения извлечённых статистических показателей

В предлагаемой схеме 7 таблиц. Таблица Tables содержит html код статистических таблиц. Каждой таблице присваивается уникальный идентификатор. Таблица KeyTypes предназначена для хранения названий типов ключей. Туда могут включаться и названия общероссийских классификаторов ОКП, ОКАТО, ОКВЭД и пр. Таблица Keys содержит список известных системе ключей, каждому ключу присвоен уникальный идентификатор и его тип. Таблица KeyValues содержит словесные описания ключей, то есть множества D_i, описанные в разделе 2.2.2.6.

Сами статистические показатели хранятся в таблице Indicators. Каждому статистическому показателю поставлен в соответствие уникальный идентификатор, числовое значение, интервал дат и идентификатор таблицы, из которой он был получен. Связанные с каждым статистическим показателем

значения ключей и единиц измерения хранятся в таблицах IndicatorKeys и IndicatorUnits соответственно.

3.2. Эксперименты

Для оценки качества полученного алгоритма использовались задачи из репозитория UCI [11], перечисленные в таблице 3.1, и задачи классификации реальных таблиц из коллекций Росстата.

Таблица 3.1. Задачи из репозитория UCI.

Название	Число объектов	Число признаков
Ionosphere	351	35
WBC	699	6
Car	1728	6
Vehicle	940	18
German	1000	24
Heart	345	13
Liver	345	7
Abalone	4177	7
Landsat	6435	36
Nursery	12960	8
Optical	5620	64
Pen	10992	16
Yeast	1484	8

Инкрементное обучение запускалось несколько раз на одной выборке с её случайным перемешиванием. Сначала алгоритм обучался на небольшой

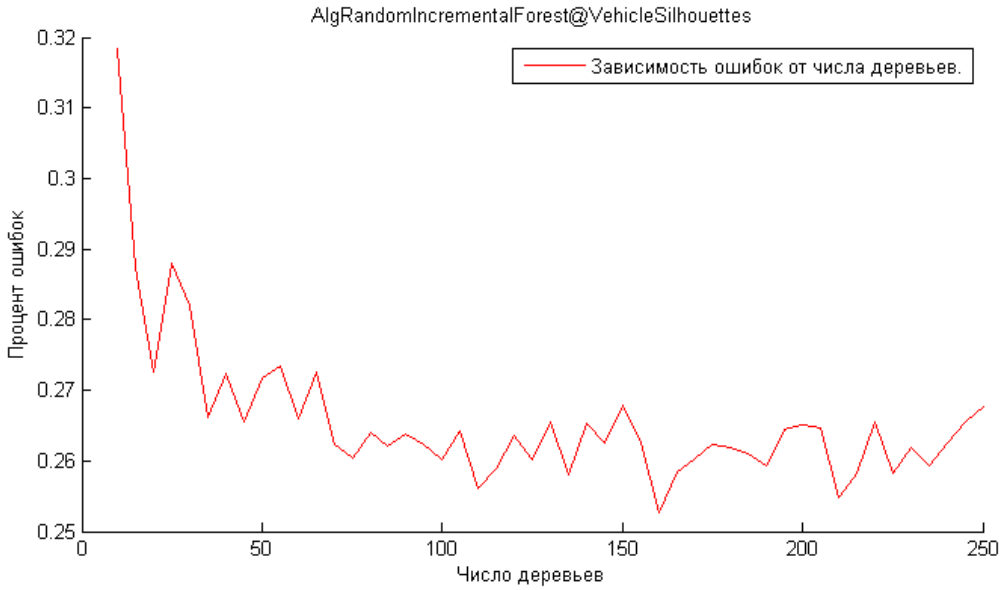


Рис. 3.6. Зависимость ошибок от числа деревьев алгоритма RIF на задаче Vehicle.

части обучающей выборки. Затем на классификацию подавался очередной объект. Если результат классификации был ошибочным, показатель числа онлайн ошибок увеличивался на единицу. После этого объект подавался на дообучение. По усреднённому значению онлайн ошибок строились кривые обучения вместе с доверительными интервалами. Значение на кривой вычислялось как средняя величина по 20 повторениям с перемешиванием обучающей выборки. В качестве критерия информативности разбиения использовался индекс Джини.

3.2.1. Оптимальное число деревьев

Рассмотрим некоторые задачи и проанализируем зависимость качества от числа деревьев и сравним с результатами, полученными на этих же задачах, алгоритма ITI с транспозицией и без неё.

Период транспозиции выбирался равным 50 или 100 в зависимости от размеров задачи. Эксперименты проводились на тех же задачах, целью было выявить зависимости ошибок алгоритма от числа деревьев. В таблице 3.2

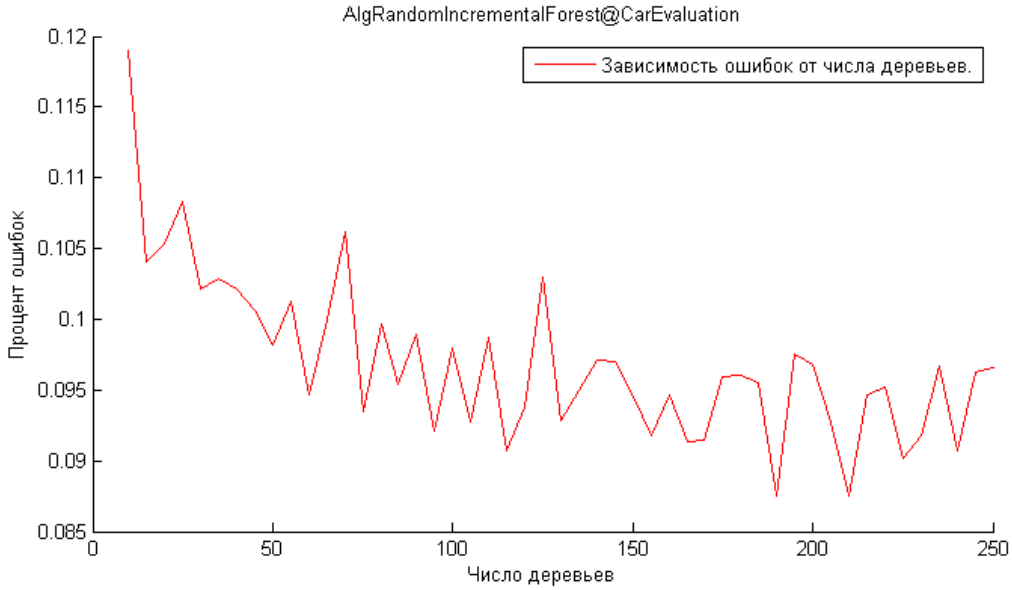


Рис. 3.7. Зависимость ошибок от числа деревьев алгоритма RIF на задаче Car.

приведены найденные наилучшие пары ошибка-число деревьев для каждой задачи. Эксперименты проводились для значений числа деревьев начиная с 10 и до 250 с шагом 5.

Задача	Число деревьев RIF	Ошибка RIF	Ошибка ITI с транспозицией	Ошибка ITI без транспозиции
Ionosphere	85	**7.6**	12.8	12.6
WBC	140	**2.9**	7.7	7.7
Car	130	9.3	**7.3**	14.5
Vehicle	230	**25.5**	30.4	35.2
German	100	**27.4**	34.2	34.8
Heart	75	**21.3**	27.9	30.1
Liver	200	**36.6**	45.4	45.2
Yeast	115	**47**	52.1	55.3

Таблица 3.2. Задачи из репозитория UCI.

Почти на всех задачах заметен выигрыш в качестве алгоритма RIF по отношению к ITI, на задачах Ionosphere и WBC алгоритм RIF работает значительно лучше.

3.2.2. Сравнение композиции и среднего дерева

Теперь проверим, насколько композиция случайных деревьев лучше, чем среднестатистическое случайное дерево в композиции. В экспериментах композиции состояли из 50 деревьев.

Таблица 3.3. Сравнение процента ошибок композиции и среднестатистического дерева

Задача	Ошибка RIF	Средняя ошибка по отдельным деревьям
Ionosphere	8.3	19
WBC	3.2	8.1
Car	10.2	21.3
Vehicle	26.5	43
German	27.7	37
Heart	23.4	36.2
Liver	40.4	45
Abalone	76.9	82.5
Landsat	11.4	25.3
Nursery	6.4	22
Optical	12	56.4
Pen	4.1	34
Yeast	48.8	63.7

Из результатов эксперимента, представленных в таблице 3.3 видно, что

среднестатистическое случайное дерево работает гораздо хуже чем композиция почти на всех задачах. Особенно выделяются задачи Nursery, Optical и Pen.

3.2.3. Отбор деревьев

Эксперименты проводились только на достаточно больших задачах. Композиция без отбора RIF состояла из 50 деревьев.

Таблица 3.4. Результаты отбора деревьев

Задача	Параметры отбора RIFTS (K_0,B,W,ΔK,K_{min})	Ошибка RIFTS	Ошибка RIF (без отбора)
Car	(160,10,10,7,50)	8.1	10.2
Nursery	(200,8,10,7,50)	4.3	6.4
Optical	(160,8,10,7,50)	13.2	12
Pen	(160,8,10,7,50)	9.9	4.1

Результаты эксперимента описаны в таблице 3.4. Видно, что отбор деревьев даёт улучшения в работе алгоритма не всегда, более того, на задачу Pen отбор оказал отрицательное влияние.

3.2.4. Задачи распознавания структуры таблиц

Алгоритмы RIF и ITI сравнивались на реальной выборке из 600 таблиц Росстата из коллекций «Регионы России, 2008» и «Финансы России, 2010». По каждой таблице генерировалась выборка для каждой из задач, после этого все выборки объединялись в общую выборку. Сначала алгоритм обучался на выборке из 50 объектов, после чего запускалось динамическое обучение.

Все эксперименты проводились 20 раз со случайно перемешанной выборкой. Производилось перемешивание номеров таблиц, а не объектов. Это позволяет смоделировать другие последовательности поступления таблиц в систему. Композиция RIF состояла из 50 деревьев. Кривые обучения алгоритмов (графики зависимости доли ошибок от числа обучающих объектов) представлены на рис. 3.8, 3.9, 3.10, доверительные интервалы приведены в таблице 3.5.

Таблица 3.5. Доверительные интервалы частоты ошибок алгоритмов (в процентах) на последнем объекте.

	ITI			RIF		
	CT	SR	CI	CT	SR	CI
Мин.	0,05	0,02	0,6	0,04	0,02	1,3
Сред.	0,08	0,04	1,7	0,07	0,03	2,8
Макс.	0,13	0,08	2,5	0,11	0,05	4,4

Задача распознавания типа ячеек (CT) представлена 28 624 объектами, 8 числовыми признаками. Оба алгоритма справились с задачей достаточно успешно, ошибка на последнем объекте составила менее 0,1%. На рис. 3.8 видно, что алгоритм RIF обучается немного быстрее, показывая меньший процент ошибок в начале выборки. Следует заметить, что доверительный интервал ошибки у RIF меньше, чем у ITI.

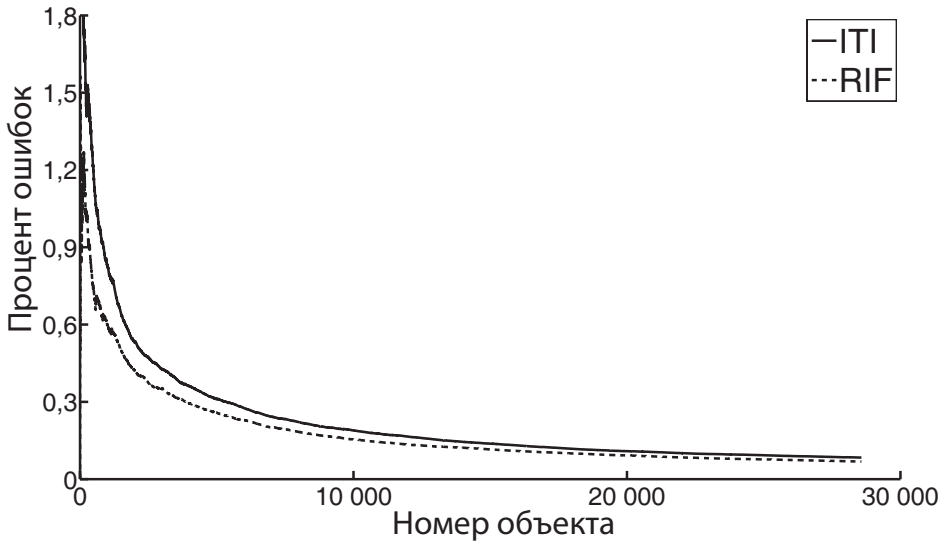


Рис. 3.8. Задача распознавания типа ячеек (CT).

Задача распознавания суперстрок (SR) представлена 10 217 объектами, 5 числовыми признаками. Результат эксперимента представлен на рис. 3.9. Ошибка на последнем объекте составила менее 0,1% (RIF — 0,03%, ITI — 0,05%). Однако по сравнению с задачей CT, алгоритм RIF медленнее обучается на этой задаче, к концу обучающей выборки показывает немного меньшее значение ошибки.

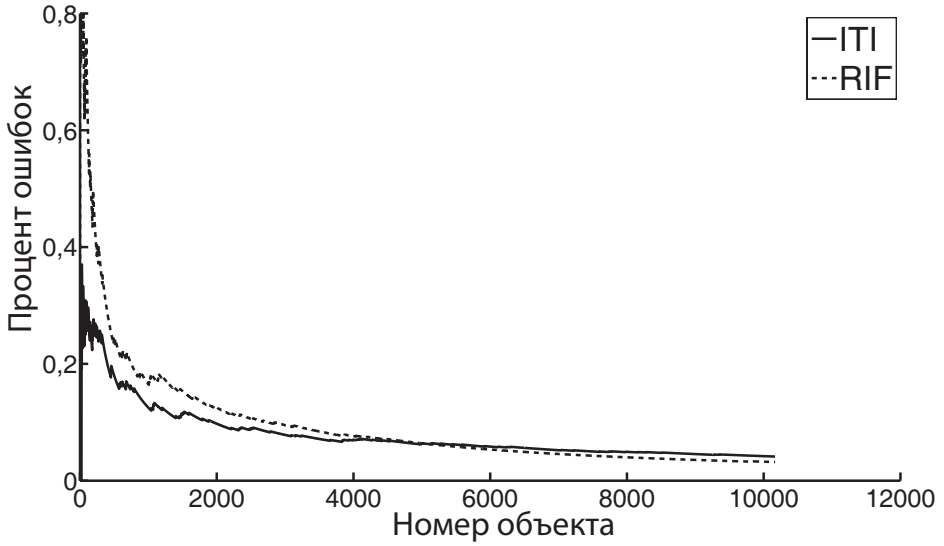


Рис. 3.9. Задача классификации суперстрок (SR).

Задача распознавания вложенных ячеек (CI) представлена 370 объектами, 4 номинальными и 7 числовыми признаками. ITI справился с задачей лучше — 1,7% ошибок против 2,8% ошибок у RIF на последнем объекте.

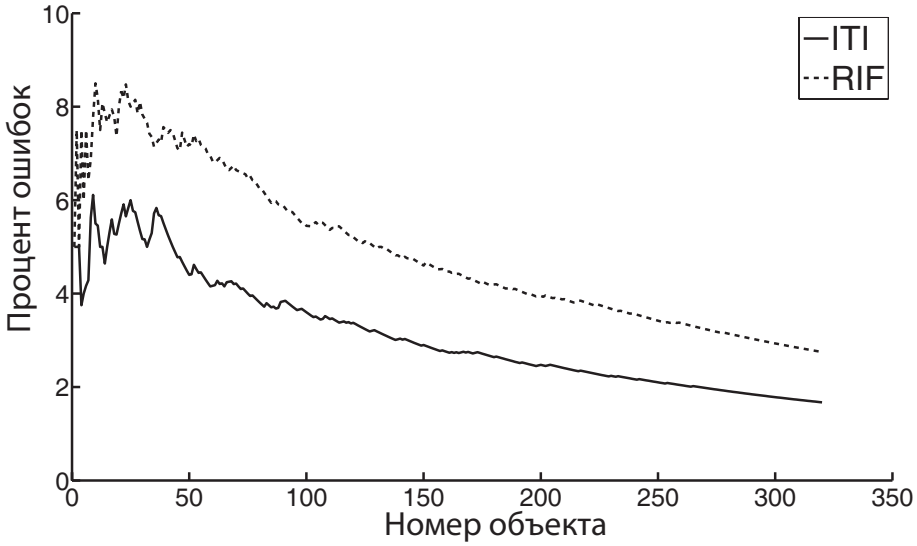


Рис. 3.10. Задача классификации вложенных ячеек (CI).

3.2.5. Вычислительная эффективность

Сравним теперь вычислительную эффективность алгоритмов RIF и ITI на задаче распознавания типа ячеек. При проведении экспериментов на этой задаче сохранялось время классификации и обучения алгоритма на всех новых объектах. В качестве результата использовалось среднее время обработки одного объекта среди всех запусков.

На рис. 3.11 показан график зависимости времени в секундах, затраченного на обучение, в зависимости от номера объекта. Видно, что алгоритм ITI значительно уступает новому алгоритму RIF. Скачки с периодом 5000 на кривой алгоритма ITI связаны с выполнением операции транспозиции.

График времени, затраченного на классификацию объектов, показан на рис. 3.12. Так как алгоритм RIF состоит из нескольких деревьев, время классификации у этого алгоритма больше, чем у ITI. Стоит отметить, что оно

составляет 6-8 миллисекунд и с увеличением выборки рост времени замедляется.

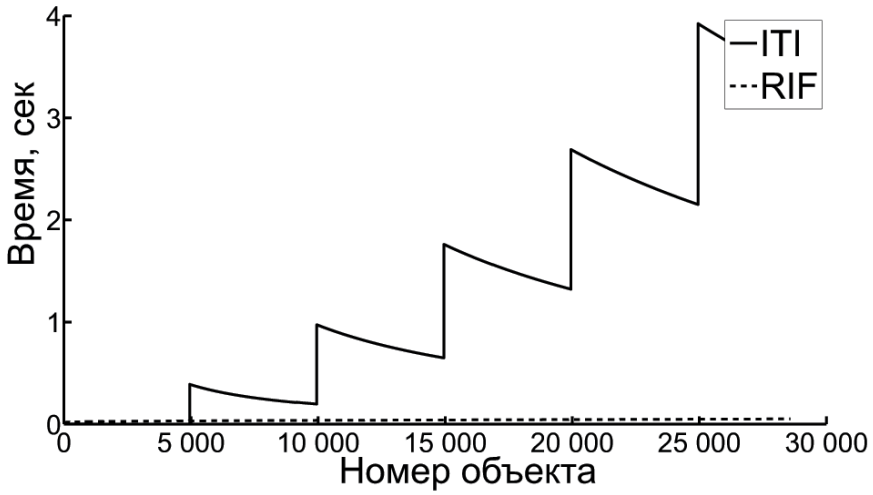


Рис. 3.11. Сравнение скорости обучения на задаче распознавания типа ячеек.

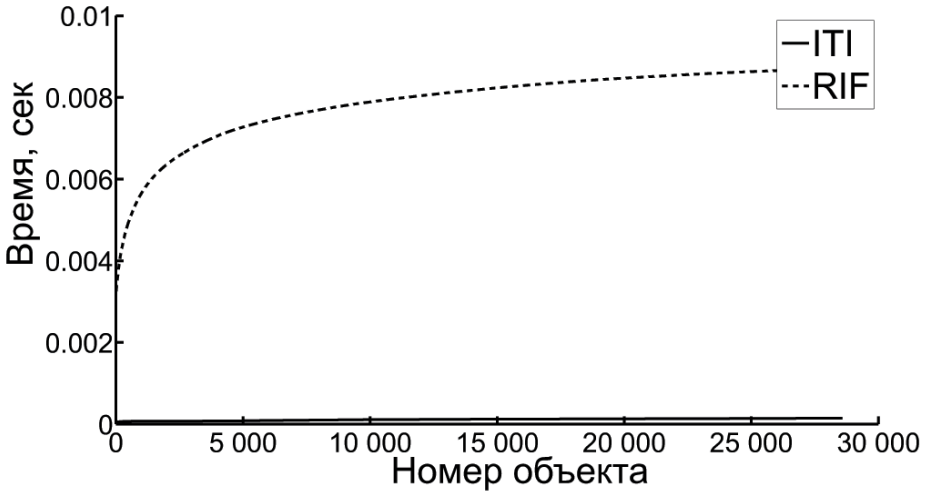


Рис. 3.12. Сравнение скорости классификации за задаче распознавания типа ячеек.

3.3. Основные выводы

1. Основными модулями системы извлечения статистической информации являются модули подготовки данных, генерации признаков, динамического обучения и классификации, а также интерфейс оператора. Опи-

санные в главе особенности реализации этих модулей позволяют в полной мере применить методологию адаптивного извлечения информации, предложенную в предыдущих главах.

2. Проведённые вычислительные эксперименты показали, что новый алгоритм RIF для построения случайного инкрементного леса с отбором деревьев работает лучше на 7 задачах UCI из 8 по сравнению с алгоритмом ITI с транспозицией и без.

3. Доля ошибок на задачах классификации при извлечении статистических показателей для алгоритма RIF составила $0,07\%$ на задаче распознавания типа ячеек, $0,03\%$ на задаче распознавания суперстрок и $2,8\%$ для задачи распознавания вложенных ячеек.

4. Композиция случайных деревьев RIF работает значительно быстрее, чем ITI с малыми периодами транспозиции и/или на больших задачах. Это увеличивает круг практических задач, которые могут быть решены с помощью нового алгоритма.

Заключение

В работе рассмотрена проблема извлечения информации из статистических таблиц, представленных в текстовом виде. Актуальность этой проблемы обусловлена необходимостью разработки поисковой системы статистической информации. Для решения рассматриваемой проблемы предложена методология, основанная на применении адаптивных корректных методов, используемых при решении задач распознавания.

В работе описаны основные свойства статистических таблиц. Информацию предлагается извлекать в виде статистических показателей — множества числовых значений с их полными смысловыми описаниями. Приведена формальная постановка задачи извлечения статистических показателей из статистических таблиц и предложен метод её решения. Основной идеей метода является сведение задачи к последовательности задач распознавания. Для определения логической структуры таблиц приведены постановки соответствующих задач классификации, описаны их признаковые пространства. Разработана процедура чтения таблицы, т. е. поиска всех ячеек описания, соответствующих каждой ячейке данных. Предложен метод нечёткого поиска всех ключей в тексте ячеек описания с учётом ошибок и опечаток. Описан способ эффективной реализации процедуры поиска с использованием обобщённых суффиксных деревьев.

Для решения задач классификации используются алгоритмы динамического обучения, которые позволяют уменьшить трудозатраты при пополнении выборки и снизить вычислительные затраты на обучение. Исследуются корректные инкрементные методы обучения. В работе предложен новый эффективный алгоритм динамического обучения RIF, который строит композицию решающих деревьев. Доказана теорема о корректности алгоритма на

непротиворечивой обучающей выборке.

Проведены сравнительные эксперименты известного алгоритма инкрементного построения решающего дерева ITI и предложенного алгоритма RIF на некоторых задачах из репозитория UCI. Эксперименты показали превосходство качества работы нового алгоритма на большинстве задач. Эксперименты на реальной выборке статистических таблиц показали качество классификации на уровне 96–99% для разных задач.

Литература

[1] *Андрианов И. А.* Анализ и разработка способов индексирования текстов на основе обобщённых и неплотных суффиксных деревьев: Дис... канд. техн. наук: 05.13.11 / Вологодский государственный университет. — Вологда, 2005. — 158 с.

[2] *Гасфилд Д.* Строки, деревья и последовательности в алгоритмах: Информатика и вычислительная биология. — БХВ-Петербург, 2003.

[3] Интегрированная база данных по социально-демографической статистике: ресурс и пользовательские сервисы для поддержки гуманитарных исследований / А. В. Богомолова, О. И. Карасев, П. Ю. Кудинов и др. // Труды XI Всероссийской объединенной конференции «Интернет и современное общество» IMS-–2008 (Санкт-Петербург, 28–30 октября 2008 года). — СПб.: Факультет филологии и искусств СПбГУ, 2008. — С. 58–59.

[4] *Кудинов П. Ю.* Задача распознавания статистических таблиц // Доклады 14-й Всероссийской конференции «Математические методы распознавания образов» ММРО-2009. — М.: МАКС Пресс, 2009. — С. 552–555.

[5] *Кудинов П. Ю.* Об одном подходе к организации системы распознавания таблиц, содержащих статистические данные // Материалы XVI Международной конференции студентов, аспирантов и молодых учёных «Ломоносов–2009». — М.: Издательский отдел факультета ВМиК МГУ; МАКС Пресс, 2009. — С. 44.

[6] *Кудинов П. Ю., Полежаев В. А.* Динамическое обучение распознаванию статистических таблиц // Доклады 8-й Международной конференции «Интеллектуализация обработки информации» ИОИ–2010 (Республ-

лика Кипр, г. Пафос, 17–24 октября 2010). — М.: МАКС Пресс, 2010. — С. 512–515.

[7] *Кудинов П. Ю., Полежаев В. А.* Инкрементное обучение деревьев решений в задаче распознавания структуры статистических таблиц // Доклады 15-й Всероссийской конференции «Математические методы распознавания образов» ММРО–2011. — М.: МАКС Пресс, 2011. — С. 593–596.

[8] *Кудинов П. Ю., Полежаев В. А.* Композиция случайных инкрементных деревьев и восстановление структуры таблиц // ***Бизнес-информатика***. — 2011. — № 4(18). — С. 39–46.

[9] *Левенштейн В. И.* Двоичные коды с исправлением выпадений, вставок и замещений символов // *Доклады Академий Наук СССР*. — 1965. — Т. 163, № 4. — С. 845–848.

[10] *Шигаров А. О.* Технология извлечения табличной информации из электронных документов различных форматов: Дис... канд. техн. наук: 05.25.05 / Ин-т вычисл. технологий СО РАН. — Иркутск, 2009. — 143 с.

[11] *A. Asuncion D. N.* UCI machine learning repository. — 2007. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[12] *Amano A., Asada N.* Graph grammar based analysis system of complex table form document // International Conference on Document Analysis and Recognition. — 2003. — Pp. 916–920.

[13] *Breiman L., Schapire E.* Random forests // Machine Learning. — 2001. — Pp. 5–32.

[14] *Cohen W. W.* Fast effective rule induction // In Proceedings of the

Twelfth International Conference on Machine Learning. — Morgan Kaufmann, 1995. — Pp. 115–123.

[15] *Cohen W. W., Singer Y.* A simple, fast, and effective rule learner // In Proceedings of the Sixteenth National Conference on Artificial Intelligence. — AAAI Press, 1999. — Pp. 335–342.

[16] *Domingos P.* Rule induction and instance-based learning: A unified approach // International Joint Conference on Artificial Intelligence. — 1995. — Pp. 1226–1232.

[17] *Douglas S., Hurst M., Quinn D.* Using natural language processing for identifying and interpreting tables in plain text. — 1995.

[18] *Embley D. W., Tao C., Liddle S. W.* Automating the extraction of data from html tables with unknown structure // *Data and Knowledge Engineering.* — 2005. — Vol. 54.

[19] *Furnkranz J., Flach P. A.* Roc 'n' rule learning - towards a better understanding of covering algorithms // Machine Learning. — 2005. — Pp. 39–77.

[20] *Hori O., Doermann D.* Consensus-based table form recognition // In 3th International Conference on Document Analysis and Recognition. IEEE Computer Society. — 1995. — Pp. 218–221.

[21] *hsi Chen H., chung Tsai S., he Tsai J.* Mining tables from large scale html texts // In Proceedings of the 18th International Conference on Computational Linguistics (COLING'00). — 2000. — Pp. 166–172.

[22] *Hulley G., Marwala T.* Evolving classifiers: Methods for incremental learning // *ArXiv e-prints.* — 2007.

[23] *Hurst M. F.* The Interpretation of Tables in Texts: Ph.d. thesis / The University of Edinburgh. — Edinburgh, 2000. — 325 pp.

[24] *Kudinov P. Y.* Extracting statistics indicators from tables of basic structure // ***Pattern Recognition and Image Analysis***. — 2011. — Vol. 21, no. 4. — Pp. 630–636.

[25] *Landau G. M., Vishkin U.* Efficient parallel and serial approximate string matching. — 1986.

[26] Learn++: An incremental learning algorithm for supervised neural networks / R. Polikar, L. Udpa, S. Udpa et al. // *IEEE Transactions on System, Man and Cybernetics (C), Special Issue on Knowledge Management*. — 2001. — Vol. 31. — Pp. 497–508.

[27] *Lee S.-W. X.* Table structure extraction from form documents based on gradient-wavelet scheme // *Lecture Notes in Computer Science*. — 1999. — Vol. 1655/1999. — Pp. 240–254.

[28] *Lopresti D. P., Nagy G.* A tabular survey of automated table processing // Graphics Recognition. — 1999. — Pp. 93–120.

[29] *McCreight E. M.* A space-economical suffix tree construction algorithm // *Journal of the Association for Computing Machinery*. — 1976. — no. 23. — Pp. 262–272.

[30] *Navarro G.* A guided tour to approximate string matching // *ACM Computing Surveys*. — 1999. — Vol. 33. — P. 2001.

[31] *Nielson H., Barrett W.* Consensus-based table form recognition // In 7th

International Conference on Document Analysis and Recognition (ICDAR 2003. — 2003. — Pp. 906–910.

[32] *Pyreddy P., Croft W. B.* Tintin: A system for retrieval in text tables // In proceedings of the second ACM International Conference on Digital Libraries. — 1997. — Pp. 193–200.

[33] *Silva A. C. E., Jorge A. M., Torgo L.* Design of an end-to-end method to extract information from tables // *International Journal on Document Analysis and Recognition.* — 2006. — Vol. 8. — Pp. 144–171.

[34] Table form document analysis based on the document structure grammar / A. Amano, N. Asada, M. Mukunoki, M. Aoyama // *International Journal on Document Analysis and Recognition.* — 2006. — Vol. 8. — Pp. 201–213.

[35] Table-processing paradigms: a research survey / D. W. Embley, M. Hurst, D. P. Lopresti, G. Nagy // *International Journal on Document Analysis and Recognition.* — 2006. — Vol. 8. — Pp. 66–86.

[36] *Tengli A., Yang Y., Ma N. L.* Learning table extraction from examples // Proceedings of the 20th international conference on Computational Linguistics. — COLING '04. — Stroudsburg, PA, USA: Association for Computational Linguistics, 2004.

[37] *Ukkonen E.* Approximate string matching over suffix trees // Proceedings of the 4th annual symposium on Combinatorial Pattern Matching, number 684 in Lecture Notes in Computer Science. — Springer-Verlag, 1993. — Pp. 228–242.

[38] *Ukkonen E.* On-line construction of suffix trees. — 1995.

[39] *Utgoff P. E.* An improved algorithm for incremental induction of decision trees. — 1994.

[40] *Utgoff P. E., Berkman N. C., Clouse J. A.* Decision tree induction based on efficient tree restructuring // *Machine Learning.* — 1997. — no. 29. — Pp. 5–44.

[41] *Wang Y., Hu J.* A machine learning based approach for table detection on the web // In Proceedings of the 11th Int'l Conf. on World Wide Web (WWW'02. — ACM Press, 2002. — Pp. 242–250.

[42] *Wang Y., Phillips I. T., Haralick R. M.* Table structure understanding and its performance evaluation // *Pattern Recognition.* — 2004. — Vol. 37. — P. 1479 – 1497.

[43] *Weiner P.* Linear pattern matching algorithms // IEEE 14th Ann.Symp. on Switching and Automata Theory. — 1973. — P. 1–11.

[44] *Yoshida M., Torisawa K.* A method to integrate tables of the world wide web // In Proceedings of the International Workshop on Web Document Analysis (WDA 2001. — 2001. — Pp. 31–34.

[45] *Zanibbi R.* Language for Specifying and Comparing Table Recognition Strategies: Ph.D. thesis / Queen's University Kingston. — 2004.

[46] *Zanibbi R., Blostein D., Cordy J. R.* A survey of table recognition: Models, observations, transformations, and inferences // *International Journal on Document Analysis and Recognition.* — 2003.