НАЦИОНАЛЬНАЯ АКАДЕМИЯ НАУК УКРАИНЫ

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

МЕЖДУНАРОДНЫЙ НАУЧНО-УЧЕБНЫЙ ЦЕНТР ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

На правах рукописи

СОКОЛОВ Артем Михайлович

УДК 004.8 + 004.032.26

МЕТОДЫ НЕЙРОСЕТЕВОГО РАСПРЕДЕЛЕННОГО ПРЕДСТАВЛЕНИЯ И ПОИСКА СХОДНЫХ СИМВОЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ В ЗАДАЧАХ КЛАССИФИКАЦИИ НА ОСНОВЕ РАССУЖДЕНИЙ ПО ПРИМЕРАМ

05.13.23 — системы и средства исскуственного интеллекта

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель
РАЧКОВСКИЙ Дмитрий Андреевич,
доктор технических наук

Киев — 2008

ОГЛАВЛЕНИЕ

Список условных обозначений 6

Введение 7

Раздел 1. Анализ современного состояния проблемы аппроксимации сходства и приближенного поиска последовательностей 14

1.1. Рассуждения на основе примеров 15

1.2. Распределенное векторное представление информации 17

1.3. Меры сходства последовательностей 21

1.3.1. Векторные меры 21

1.3.2. Частотные меры и меры на множествах 22

1.3.3. Расстояния редактирования 24

1.4. Методы аппроксимации и оценки сходства с помощью вложений пространств 26

1.4.1. Векторные метрики и меры сходства на множествах . . 27

1.4.2. Расстояния редактирования 29

1.5. Задача поиска приближенных ближайших соседей 32

1.5.1. Локально-чувствительное хеширование и распределенные представления 34

1.5.2. Решение задачи поиска приближенных ближайших соседей с помощью локально-чувствительного хеширования 35

1.6. Выводы по разделу 1 38

Раздел 2. Разработка и исследование детерминированного метода вложения пространства последовательностей с классической метрикой редактирования в манхеттеново пространство 40

2.1. Базовые обозначения и определения 40

2.2. Классификация структур графа де Брейна 41

2.2.1. Случай (q, w)-неповторяющихся строк 43
2.2.2. Случай (q, w)-повторяющихся строк 46
2.3. Нижнее и верхнее ограничение на расстояние редактирования . 49
2.3.1. Ограничения на расстояние редактирования в пределах одного интервала 49
2.3.2. Распространение ограничений на расстояние редактирования на несколько интервалов 58
2.4. Детерминированный метод вложения расстояния редактирования в ℓ_1 60
2.4.1. Нижняя оценка стоимости редактирования 60
2.4.2. Верхняя оценка на расстояние редактирования 64
2.4.3. Объединение оценок 65
2.5. Численное исследование детеминированного метода вложения 67
2.5.1. Экспериментальное исследование результатов лемм 2.7 и 2.8 67
2.5.2. Экспериментальная база RandomStrings 71
2.5.3. Проверка теоретических ограничений детерминированного вложения 72
2.6. Выводы по разделу 2 74

Раздел 3. Разработка распределенных методов вложения расстояния редактирования и эффективного поиска приближенных ближайших последовательностей 75
3.1. Рандомизированные методы формирования векторных представлений для поиска ближайших строк 75
3.1.1. Вероятностные варианты лемм раздела 2 76
3.1.2. Локально-чувствительная функция для расстояния редактирования 76
3.2. Решение задачи поиска приближенных ближайших последовательностей с помощью рандомизации детерминированного метода 78

3.3. Решение задачи поиска приближенных ближайших последовательностей с помощью разработанной локально-чувствительной функции 81
3.3.1. Анализ поиска приближенных ближайших последовательностей на основе предложенной локально-чувствительной функции 83
3.4. Распределенные представления последовательностей 84
3.4.1. Метод распределенного представления последовательностей без учета порядка элементов 84
3.4.2. Методы распределенного представления последовательностей с учетом порядка элементов 85
3.4.3. Прореживающее связывание и его связь с разработанной локально-чувствительной функцией 87
3.4.4. Тернаризация и бинаризация рандомизированных представлений на основе локально-чувствительной функции 90
3.5. Выводы по разделу 3 91

Раздел 4. Алгоритмическая реализация и численное исследование распределенного представления и поиска приближенных ближайших символьных последовательностей 93
4.1. Экспериментальное исследование вероятности коллизии разработанной локально-чувствительной функции 93
4.2. Поиск приближенных ближайших последовательностей с помощью LSH-леса 94
4.3. Методы работы с базами с разной длиной последовательностей 96
4.3.1. Дополнение последовательностей спецсимволами . . . 96
4.3.2. Разбиение последовательностей окнами 97
4.3.3. Кластеризация последовательностей по длине 97
4.4. Выбор параметров для обеспечения вычислительной эффективности и точности поиска 98

4.4.1. Экспериментальное исследование качества кандидатов на ближайшего соседа 98
4.4.2. Экспериментальное исследование порядка ближайших соседей 100
4.5. Выводы по разделу 4 104

Раздел 5. Реализация и применение разработанных методов распределенного представления и поиска последовательностей 106
5.1. Разработка программных средств 106
5.1.1. Программные библиотеки 106
5.1.2. Программный нейрокомпьютер SNC 108
5.1.3. Специализированные программные системы 111
5.2. Применение в задачах классификации 112
5.2.1. Исследование поиска дубликатов в текстовых и веб-коллекциях 112
5.2.2. Обнаружение спама в электронных сообщениях 122
5.2.3. Поиск генов и гиперчувствительных сайтов в последовательностях ДНК 127
5.2.4. Обнаружение вторжений в компьютерные системы . . . 136
5.3. Выводы по разделу 5 138

Выводы 141

Приложение А. Акты внедрения 144

Список использованной литературы 145

СПИСОК УСЛОВНЫХ ОБОЗНАЧЕНИЙ

Σ	–	алфавит символов
x, y	–	символьные последовательности, строки
$\|P\|$	–	количество элементов в множестве P
K	–	размерность распределенных векторов
L	–	количество деревьев
$Prob[A]$	–	вероятность события A
$N[A]$	–	число интервалов, где выполняется условие A
$[A]$	–	индикаторная функция (равна единице, если условие A выполняется, и нулю в остальных случаях)
$\langle Z \rangle$	–	операция контекстно-зависимого прореживания над вектором Z
(v_1, v_2)	–	скалярное произведение векторов v_1, v_2
АПНС	–	ассоциативно-проективная нейронная сеть
VSM	–	vector space model
CDT	–	context-dependent thinning (контекстно-зависимое прореживание)
CBR	–	case-based reasoning (рассуждения на основе примеров)
kNN	–	k Nearest Neighbors classifier (классификатор k ближайших соседей)
БС	–	ближайший сосед
LSH	–	locality-sensitive hashing (локально-чувствительное хеширование)
SNC	–	Software Neuro Computer (программный нейрокомпьютер)
БД	–	база данных

ВВЕДЕНИЕ

Актуальность темы. При решении широкого спектра задач поиска, классификации и распознавания качество анализа больших массивов данных существенно зависит от возможности учета не только наличия, но и последовательности информационных элементов. Примерами актуальных задач, требующих работы с последовательностями, являются поиск текстовых дубликатов в поисковых машинах, борьба с несанкционированными рассылками электронной почты (спамом), поиск генетических последовательностей, обнаружение вторжений в компьютерных системах для обеспечения информационной безопасности, распознавание и идентификация акустических сигналов и др. Для решения такого класса задач перспективно использование подхода, основанного на моделировании интеллектуальной деятельности человека и предполагающего реализацию механизма рассуждений на основе примеров.

При решении задач на основе примеров, в базе примеров запоминают данные с сопутствующей информацией, например, известными текстами, спамовыми сообщениями, размеченными генетическими последовательностями, «почерк» пользователей компьютерных систем с известными идентификаторами и др. Для новой входной информации система находит в базе один или несколько сходных запомненных примеров и принимает решения, делает прогнозы и выводы о входных данных, адаптируя к ним знания об имеющихся примерах. (J.G. Carbonell, K. Forbus, D. Gentner, J. Kolodner, C.K. Riesbeck, R.C. Shank, В.П. Гладун, Н.Г. Загоруйко, Д.А. Поспелов, А.И. Уемов и др.). Этап поиска сходных примеров является центральным в методе рассуждений на основе примеров. Для оценки сходства примеров, имеющих последовательную структуру, часто используют расстояние Левенштейна (классическое расстояние редактирования). Широко известен классический алгоритм вычисления расстояния Левенштейна, сложность которого квадратична относительно дли-

ны строк. Однако при больших характерных длинах и количестве примеров последовательностей в базах и во входных потоках непосредственное применение этого алгоритма требует больших вычислительных затрат.

Для повышения эффективности нахождения сходства примеров используют трансформации форм их представления. Ряд подходов к такой трансформации связан с моделированием нейросетевых механизмов структурно-функциональной организации мозга (П.И. Бидюк, Е.В. Бодянский, А.М. Касаткин, Л.М. Касаткина, Н.Н. Куссуль, А.М. Резник, А.А. Фролов, S. Amari, J. Hopfield, S. Grossberg, T. Kohonen, B. Widrow, D. Willshaw и др.) и распределенного представления информации в мозге (D. Hebb, G. Hinton, P. Kanerva, J. McClelland, G. Palm, T. Plate, J. Pollack, D. Rumelhart, P. Smolensky, D. Touretzky, Э.М. Куссуль, Д.А. Рачковский и др.). Распределенное представление – форма векторного представления информации, где каждый объект (символ, подпоследовательность, признак, физический объект, их совокупность и др.) представлен множеством элементов вектора, а отдельный элемент вектора может принадлежать представлениям разных объектов. Этот подход тесно связан с подходом вложений пространств (P. Indyk, R. Motwani, A. Broder, C. Sahinalp, G. Cormode, T. Batu, M. Charikar и др.).

Распределенные представления обеспечивают высокую информационную емкость, вычислительно эффективную оценку сходства векторными мерами, позволяют использовать известные методы обработки векторной информации. Однако предложенные подходы к нейросетевому распределенному представлению последовательностей нуждаются в теоретическом обосновании, должны быть также созданы и реализованы соответствующие методы, эффективность которых требуется исследовать в практических задачах.

Таким образом, в условиях роста объемов и сложности обрабатываемой информации, содержащей символьные последовательности (текстовая информация, Интернет, электронная почта, геномы, аудит-последовательности компьютерных сетей), существует практическая потребность в повышении эффективности обработки последовательностей на основе реализации рассужде-

ний по примерам с помощью распределенных представлений. В то же время уровень развития теоретической базы распределенного представления последовательностей недостаточен для эффективного решения прикладных задач, связанных с поиском сходных примеров, имеющих структуру последовательности элементов.

Это обусловливает актуальность научной задачи разработки методов нейросетевого распределенного представления последовательностей, а также их поиска и классификации, для эффективной оценки сходства и использования информации о последовательностях в системах искусственного интеллекта, применяющих модели рассуждений человека на основе примеров.

Практические аспекты работы требуют создания программных инструментальных средств и прикладных систем, реализующих и использующих разработанные методы, а также их исследования в приложениях.

Связь работы с научными программами, планами, темами. Работа выполнялась в рамках следующих НИР: «Разработка и исследование нейросетевых методов моделирования когнитивных процессов», № ГР 0101U002685 (2001-2003); «Дослідження та розроблення нових інтелектуальних інформаційних технологій на основі використання високоефективних нейромережевих методів та алгоритмів» № ГР 0102U002070 (2002-2006); «Разработка и исследование нейросетевых информационных технологий работы с базами знаний» № ГР 0104U003191 (2004-2006); «Создать опытные образцы нейрокомпьютеров новых поколений» № ГР 0101U006718 (2000-2001), «Разработать методы и создать способы интеллектуализации информационных технологий широкого использования» № ГР 0101U007953 (2001); «Створити засоби автоматичної обробки інформації із застосуванням міркувань за аналогіями», № ГР 0103U008280 (2003-2006), ДНТП «Образный компьютер»: «Розробка технології для створення систем смислової інтерпретації текстової інформації та смислового перекладу текстів з однієї мови на іншу» № ГР 0102U005512 (2002); «Разработать компьютерную технологию целенаправленной обработки текстовой и аудио информации» № ГР 0103U005770 (2003); »Разработать

интеллектуальные информационные технологии распознавания и идентификации аудио-видео-информации на основе нейросетевых технологий» № ГР 0104U008324 (2004).

Цель работы: Повышение эффективности оценки сходства информации с последовательной структурой для решения прикладных задач классификации и поиска за счет разработки и реализации методов и средств нейросетевого векторного распределенного представления последовательностей. Для достижения этой цели поставлены и решены следующие задачи:

1. Разработать методы векторного представления символьных последовательностей, сохраняющие сходство по расстоянию редактирования.
2. Разработать и теоретически проанализировать методы рандомизированного распределенного представления последовательной информации.
3. Разработать и исследовать методы поиска сходных символьных последовательностей с помощью распределенных представлений.
4. Разработать программные средства, реализующие предложенные методы представления, поиска и классификации приближенно ближайших последовательностей.
5. Экспериментально исследовать вычислительную эффективность и качество разработанных методов в задачах поиска и классификации информации с последовательной структурой на основе рассуждений по примерам.

Объект исследования: представление и обработка символьных последовательностей в системах поиска и классификации.

Предмет исследования: нейросетевые распределенные представления информации, методы их формирования и обработки, методы оценки сходства символьных строк, методы поиска и классификации сходных последовательностей.

Методы исследования. При разработке и исследовании методов формирования распределенных представлений, поиска и классификации последовательной информации использовались методы математического и имитаци-

онного моделирования, дискретной математики, теории вероятностей и математической статистики. Для экспериментальной проверки разработанных методов и программных систем применялись методы статистической обработки результатов массовых экспериментальных исследований. При разработке систем и средств реализации предложенных методов использовались методы системного и объектно-ориентированного анализа, проектирования и функционального программирования.

Научная новизна полученных результатов.

1. Разработан новый метод вложения пространства последовательностей с метрикой Левенштейна расстояния редактирования в векторное пространство с манхетенновой метрикой, отличающийся учетом подпоследовательностей переменной длины и обеспечивающий повышение точности аппроксимации расстояния редактирования.
2. Впервые предложена локально-чувствительная функция, отличающаяся учетом расстояния редактирования между последовательностями, которая продуцирует распределенное представление последовательностей, сохраняющее их сходство по расстоянию редактирования.
3. Впервые получены оценки затрат памяти, вычислительной сложности и точности аппроксимации расстояния редактирования с помощью полученных распределенных представлений за счет использования методов метрических вложений пространств и методов анализа рандомизированных алгоритмов.
4. Усовершенствованы методы поиска приближенных ближайших последовательностей за счет применения разработанных локально-чувствительных хеш-функций, что позволило обеспечить сублинейное относительно размера базы примеров время поиска сходных последовательностей
5. Получили дальнейшее развитие методы решения задач поиска и классификации последовательностей на базе рассуждений по примерам за счет использования разработанного метода хеширования и применения общей базовой операции поиска ближайших соседей в базах примеров-

последовательностей, имеющих различную длину.

Практическая значимость полученных результатов. На основе разработанных оригинальных методов представления и поиска последовательностей созданы новые программные средства для решения прикладных задач и реализации информационных технологий, связанных с обработкой символьных последовательностей.

- Программная объектно-ориентированная библиотека для представления и поиска последовательностей LSHLibrary, реализующая методы представления и поиска приближенных ближайших символьных последовательной.
- Программное средство TextInputTools, содержащее модули форматированного ввода для баз генетических последовательностей, электронных писем, ряда популярных текстовых корпусов, аудит-последовательностей UNIX систем.
- Програмные макеты DuplClassifier, EmailClassifier, NuclClassifier, SessionClassifier для поиска текстовых дубликатов, спама, кодирующих участков генетических последовательностей и классификации сессий пользователей UNIX-системы.
- Модули программного нейрокомпьютера SNC:
  - KNN, реализующий алгоритм поиска K ближайших соседей по заданной метрике;
  - VectorComparer – унифицированное средство сравнения векторов по множеству стандартных метрик и мер;
  - обрабатывающие блоки – форматирования, предобработки, кодирования и др.

Разработанное алгоритмическое и программное обеспечение используется в научных и практических целях, что подтверждается соответствующими актами: Министерства промышленной политики Украины (от 26.10.05), Института информатики АН Чешской Республики (от 21.11.2005), ТЕЛКО ЛИМИТЕД (от 17.10.2007).

Личный вклад соискателя. В работах, написанных в соавторстве и опубликованных в профильных изданиях, вклад соискателя состоит в следующем: [140] – метод идентификации пользователей с помощью нейронных сетей обратного распостранения ошибки. В [123] – концепция модулей форматированного ввода информации и классификации в нейрокомпьютере. В [107] – прореживающая схема распределенного представления последовательностей и идеи ее анализа. В [132] – поиск текстовой информации с использованием q-граммного представления. В [130] – модули форматированного ввода информации и классификации.

Апробация результатов диссертации. Результаты диссертационного исследования были доложены на: Intern. Joint Conf. on Neural Networks (USA, 2003), X, XI, XII Intern. Conf. «Knowledge-Dialogue-Solution» (Bulgaria, 2003, 2005, 2006); Международной конференции «Проблемы нейрокибернетики» (Ростов-на-Дону, 2002, 2005); Международном семинаре по индуктивному моделированию (Киев, 2005); Школе-семинаре «О проблемах образного мышления» (Жукин, 2005); семинарах «Проблемы нейрокомпьютеров и нейросетей» научного совета НАНУ по проблеме «Кибернетика» (Киев, ИПММС НАН Украины и МНУЦ ИТиС НАН Украины, 2001 – 2008) и «Образный компьютер» (Киев, МНУЦ ИТиС НАН Украины, 2008).

Публикации. Основные результаты диссертации изложены в 18 печатных работах, из них 11 – в профильных изданиях Украины и других стран, из них 6 – без соавторов.

РАЗДЕЛ 1

АНАЛИЗ СОВРЕМЕННОГО СОСТОЯНИЯ ПРОБЛЕМЫ АППРОКСИМАЦИИ СХОДСТВА И ПРИБЛИЖЕННОГО ПОИСКА ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Задачи обработки и извлечения полезной информации из больших массивов данных, для которых важна последовательность их компонентов, требует повышения эффективности и интеллектуализации соответствующих информационных технологий. Во многих прикладных задачах разных предметных областей существует необходимость сравнения длинных символьных последовательностей. Это поиск в Интернет, обнаружение плагиата, фильтрация дубликатов поисковыми машинами Интернет, системы документооборота. В компьютерных системах и сетях актуальной задачей является обнаружение вторжений, в частности, путем анализа сессий или потока сетевого трафика на предмет наличия аномалий в поведении пользователей. Современные спутники непрерывно генерируют и пересылают измерения, которые также необходимо сравнивать. Одной из самых распространенных современных задач, с которой генетики сталкиваются ежедневно, является поиск близких последовательностей нуклеотидов [77, 17, 144, 104, 79, 51].

Характерные длины последовательностей в этих областях изменяются от 10^2-10^5 (веб-страницы) до 10^9 (геномы), а количество последовательностей – от 10^1-10^3 (сессии пользователей) до 10^7-10^8 (индекс поисковых машин или база генетических последовательностей).

Сравнение и поиск усложняется в случае недоступности всей последовательности – память и быстродействие систем часто слишком ограничены, чтобы сохранить ее для дальнейшей обработки [54, 79].

Эффективным способом решения разного рода задач является модели-

рование интеллектуальной деятельности человека по рассуждению на основе примеров (п. 1.1). Центральным в этом подходе является поиск ближайшего известного примера ко входному объекту (запросу), поскольку на входной объект переносится известная информация о ближайшем примере (например, метка его класса при решении задачи классификации).

Для сравнения последовательностей-примеров в базе необходимо задаться метрикой (ряд метрик рассмотрен п. 1.3), которая должна быть адекватной задаче и эффективно вычислимой. Для сравнения и поиска ближайших среди большого количества длинных последовательностей необходимы эффективные методы. Для этого применяют «сокращенные» представления которые, требуя меньше ресурсов для хранения и обработки, чем исходная информация, сохраняют необходимую для приложений информацию. В п. 1.2 рассматриваются относящиеся к этому типу распределенные представления, развиваемые в области нейросетевого подхода к искусственному интеллекту, а в п. 1.4 – методы теории вложений, позволяющие аппроксимировать сложновычислимые метрики. В п. 1.5 рассмотрены подходы к приближенному поиску ближайших примеров. В п. 1.6 обосновано направление исследований диссертационной работы.

1.1. Рассуждения на основе примеров

В системах искусственного интеллекта решение класса задач, связанных с классификацией и поиском последовательностей, возможно с использованием подхода, основанного на моделировании интеллектуальной деятельности человека, а именно рассуждений на основе примеров или прецедентов (CBR – case-based reasoning) [69].

Направление рассуждений на основе примеров является альтернативой типичному для экспертных систем подходу, основанному на правилах [122]. Недостатками подходов, основанных на правилах, являются трудоемкость и высокая стоимость создания, ориентация на специфику предметных областей;

проблемы с обобщением; сложность последовательного поиска в больших системах и др. [89]. Применение CBR-подхода становится еще более актуальным в связи с ростом числа и объемов баз данных и знаний, где хранятся примеры ситуаций и решенных задач, обучающих выборок, и т.п. Разновидность этого подхода – метод ближайшего соседа [38] – широко используется в области классификации и распознавания образов.

При реализации подхода рассуждений по примерам в системах искусственного интеллекта формируется база прецедентов, где запоминаются описания индивидуальных или обобщенных примеров. Для новой входной ситуации система находит одну или несколько ближайших, сходных с ней ситуаций в базе, и принимает решения, делает выводы о входной ситуации, адаптируя к ней знания об известных примерах (рис. 1.1).

В основе этого подхода лежит поиск в базе ближайшего примера, который осуществляется путем вычисления некоторой меры сходства между представлениями примеров. При этом этап поиска сходных примеров является центральным в методе рассуждений на основе примеров, что обусловливает необходимость эффективных методов поиска сходных примеров, в частности тогда, когда примеры представляют из себя последовательности.

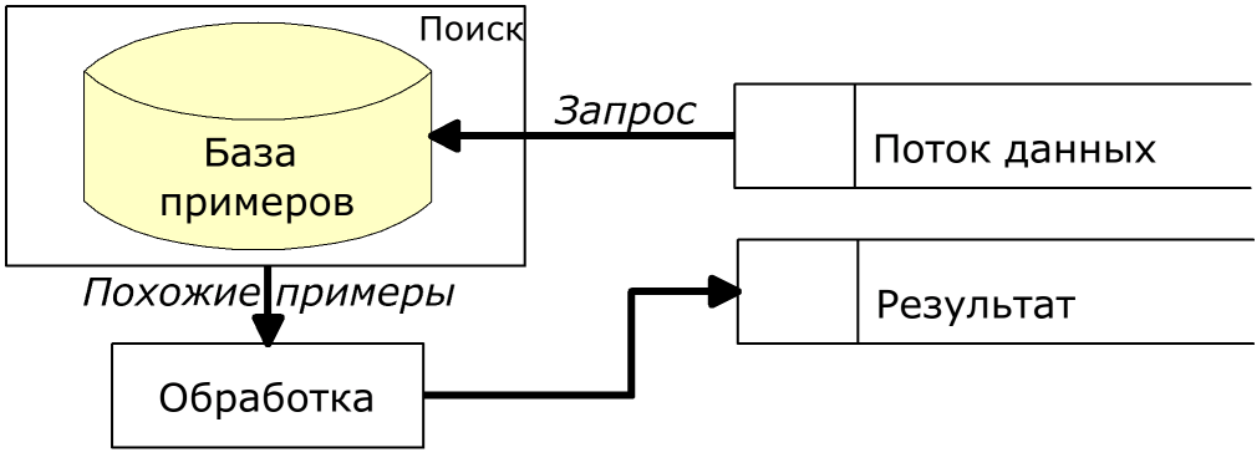


Рис. 1.1. Схема подхода к решению задач на основе рассуждений по примерам

1.2. Распределенное векторное представление информации

Моделирование представления информации в мозге является основой нейросетевого подхода к обработке информации. Применение рассуждений на примерах связано с проблемой, состоящей в том, что информация разного типа, модальности, степени сложности обычно представлена в разных форматах, что требует различных средств ее обработки и сильно влияет на эффективность метода в целом. Выбор представления информации определяет возможности и эффективность методов и алгоритмов, применяемых для решения поставленных задач [4, 127, 89].

Перспективными с точки зрения повышения эффективности обработки информации и интеграции информации различных типов являются распределенные представления. Распределенные представления информации (distributed representation) – форма векторного представления, где каждый объект (или единица информации) представлен многими элементами вектора, и каждый элемент вектора может участвовать в представлении многих объектов [111, 115]. В распределенных представлениях состояние отдельных элементов не интерпретируется в отрыве от значений состояний других элементов сети [111].

Нейросетевые распределенные представления обеспечивают параллельную обработку и поиск, хорошую обобщающую способность, поиск по неполным данным и т.п. Преимущества распределенных представлений [91, 111, 89, 138] состоят в:

1. эффективном использовании ресурсов (для представления информации используется ограниченное множество элементов, в то время как в локальных представлениях для представления нового объекта создается новый элемент);
2. помехоустойчивости за счет избыточности представления (удаление ограниченного числа элементов не оказывает влияния на восстановление информации);

3. явном представлении сходства объектов (по мерам сходства соответствующих им векторов);
4. параллельности алгоритмов обработки;
5. возможности использования широкого спектра методов поиска, классификации, регрессии и кластеризации, существующих для векторных представлений.

Сложности в использовании распределенных представлений для структурированной информации состоят в отсутствии очевидных механизмов их формирования для последовательностей или более сложных структур.

Относительно недавно появились модели распределенного представления, которые позволяют представлять и оперировать структурированной информацией (temporal synchrony [102], binary spatter-coding [64], holographic reduced representations [89], ассоциативно-проективные нейронные сети (АПНС) [91]).

В основе работы лежат распределенные представления, развиваемые в рамках парадигмы АПНС [72], истоки которой лежат в работах по моделированию процессов мышления, инициированных Н.М. Амосовым [4]. В АПНС любая информация представления многомерными (бинарными) разреженными псевдослучайными векторами. Под разреженностью понимается малое количество единиц, а под псевдослучайностью – случайность, но неизменность их расположения для представления одной и той же информации. Степень сходства объектов x и y оценивается по мерам сходства их векторных представлений.

Для представления непохожих объектов (различных символов или далеких числовых значений) часто используются независимые случайные векторы [92], каждый элемент такого вектора – независимая и одинаково распределенная случайная величина из $\{0, 1\}$. Для представления похожих объектов (близких строк или числовых значений) используются векторы с долей совпадающих единичных элементов, зависящим от степени сходства объектов. Предложены варианты распределенных представлений и в виде векторов с

вещественными элементами [88, 89]).

Представление структурированной информации в виде распределенных векторов требует адекватного учета информации о группировке элементов структуры. Для этого предложен ряд процедур «связывания» [92], которые основаны на избыточности распределенного представления информации.

Избыточное количество ненулевых элементов распределенного представления объекта позволяет удалять из представлений определенную долю единиц. Это позволяет использовать для формирования составных объектов прореженные представления их компонентов (частей), объединяемые поэлементными векторными операциями. Если при этом подмножество единиц отдельных векторов компонентов будет зависеть от всего набора компонентов, входящих в составной объект (например, подпоследовательностей), то таким образом будет сохранена информация о группировке компонент. Эта идея лежит в основе процедур связывания конъюнкцией и контекстно-зависимым прореживанием.

Для двух векторов X_k связывание поэлементной конъюнкцией осуществляется как $\langle Z \rangle = \wedge X_k, k = 1, \ldots, K$. Процедура связывания кодвекторов конъюнкцией имеет следующие свойства.

1. является детерминированной: применение процедуры к тому же входу дает одинаковый результирующий кодвектор.
2. каждый входной кодвектор представлен в выходном кодвекторе долей своих единиц (для бинарных, не ортогональных кодвекторов), таким образом, сохраняя сходство результата связывания и входных векторов.
3. средняя плотность итогового вектора зависит от числа входных векторов K и степени их перекрытия.
4. если два набора входных векторов похожи, их связанные векторы также похожи.
5. подмножества единиц входного вектора, остающиеся после связывания конъюнкцией, пересекаются тем больше, чем больше сходны векторы, которые поступают на вход процедуры связывания совместно с этим

вектором.

Аддитивная процедура *контекстно-зависимого прореживания* CDT (context-dependent thinning) [92] состоит в следующем. Формируется побитовая дизъюнкция связываемых векторов $Z = \bigvee_i v_i$, затем осуществляется контекстно-зависимое прореживание с помощью поэлементной конъюнкцией

$$\langle Z \rangle = \bigvee_{k=1,\ldots,K} (Z \wedge Z_k^*) = Z \wedge \bigvee_{k=1,\ldots,K} Z_k^*,$$

где Z_k^* – перестановка вектора Z. Для каждого k применяется своя случайная независимая перестановка, отличная от вектора Z. Число K – кратность прореживания, определяющая плотность результирующего вектора.

Связывание с помощью CDT [92]:

1. обеспечивает пропорциональное представительство компонентов (подмножество единиц вектора компонента, входящее в состав прореженного вектора, зависит от соотношения плотности данного вектора с векторами других компонентов);
2. сохраняет сходство часть-целое между составными объектами и компонентами, их образующими (в результирующий вектор входят подмножества единиц векторов соответствующих компонентов);
3. сохраняет сходство подмножеств составных объектов (в результирующий вектор входят подмножества единиц векторов соответствующих компонентов, поэтому векторы объектов, содержащие одинаковые компоненты, похожи);
4. обеспечивает структурное сходство составных объектов (состав подмножества единиц для вектора компонента зависит от других векторов, входящих в состав составного объекта)

На основе процедур прореживания можно формировать распределенные представления, учитывающие структуру, необходимые для создания вычислительно эффективных и качественно новых методов решения задач обработки структурированной информации [92, 91], в т.ч. последовательностей (п. 3.4.2). С другой стороны, существует связь между распределенными представлени-

ями и методами метрических вложений (п. 3.4.3), что позволяет получить теоретические оценки эффективности распределенных методов.

1.3. Меры сходства последовательностей

Обозначим Σ алфавит символов и две последовательности $x, y \in \Sigma^n$. Под элементами последовательностей $x_i \in x, y_i \in y$ будут пониматься или символы из алфавита Σ , или же (в зависимости от приложения) слова в текстах x, y. Существует ряд мер, определяющих тех или иным способом сходство последовательностей.

1.3.1. Векторные меры

Простейшим способом задать сходство между последовательностями x и y являются аналоги векторных метрик, примененные к последовательностям. Например, обобщенное расстояние Хэмминга

$$d_H(x, y) = \sum_{i=1}^{n} [x_i \neq y_i], \tag{1.1}$$

определяющее количество неодинаковых элементов в последовательности. Данная метрика не учитывает возможную разницу в количестве различных элементов и их порядок. Однако, несмотря на свою простоту, может применяться для сравнения генетических последовательностей [21]. Аналогично, для цели сравнения последовательностей могут применяться метрики пространств $\ell_p, p \geqslant 1$:

$$d_{l_p}(x, y) = ||x - y||_p = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{1.2}$$

или сравнение частотных моментов

$$F_k(x) = \sum_{i=1}^{n} x_i^k \tag{1.3}$$

(при $k = 0$ это количество различных элементов в последовательности, при $k = 1$ – длина последовательности и при $k = 2$ – т.н. показатель повторяемости

(repeat rate)), применяющиеся в базах данных, маркетинге и анализе сетевого трафика.

Преимуществом векторных метрик является простота, наличие готовых алгоритмов классификации, кластеризации и т.д., разработанных для векторов, а также возможные применения в потоковой модели и существование эффективных алгоритмов оценки (п. 1.4.1).

1.3.2. Частотные меры и меры на множествах

В области Information Retrieval (IR) популярны представления последовательностей (текстов) типа «bag-of-items», где последовательность (текст) x рассматривается как множество X входящих в нее символов или подпоследовательностей (слов). Основываются данные представления на очевидном наблюдении, что похожие (в частности, по смыслу) строки должны содержать много одинаковых или похожих частей.

Для последовательности x создается вектор $v(x)$, где элемент вектора $v_\sigma(x)$ равен числу вхождений элемента σ в x или основанной на $v_\sigma(x)$ величине – частоте элемента или статистической мере tf-idf [97]. Примером подобных векторных представлений является q-граммное расстояние [113, 63], определяемое как манхеттеново расстояние (ℓ_p-расстояние при $p = 1$, см. (1.2)) между соответствующими векторами

$$d_q(x, y) = \|v_q(x) - v_q(y)\|_{l_1} = \sum_{\sigma \in \Sigma^q} |v_\sigma(x) - v_\sigma(y)|, \tag{1.4}$$

где в качестве элементов последовательности выбраны все подстроки длины q строки x (q-граммы). Например, для слова $x =$ `'vacations'` и $q = 3$ множество $X = \{$`'vac'`, `'aca'`, `'cat'`, `'ati'`, `'tio'`, `'ion'`, `'ons'`$\}$.

Связанной с такими представлениями является методология векторных пространств (vector space models, VSM) [96]. В этой методологии слово может абстрактно рассматриваться как набор контекстов, в которых оно встречается. Каждый контекст представлен множеством слов («bag-of-words»), который является или документом, где встретилось слово, или ближайшей окрестностью

слова.

Каждому элементу контекста (тексту или слову) ставится в соответствие случайный вектор (с малым числом +1 и -1). Окончательное представление слова формируется суммированием этих векторов каждый раз, когда слово появляется в корпусе, и может быть проинтерпретировано как распределенное представление документа или термина (п. 1.2) или как рандомизированное уменьшение размерности (п. 1.4.1).

Помимо применения метрик, для оценки (семантического) сходства текстов x, y, представленных векторами «bag-of-items», применяется оценка сходства по величине косинуса [98] угла между соответствующими векторами $v(x), v(y)$:

$$cos(v(x), v(y)) = (v(x), v(y)) / \|v(x)\|_{l_1} \|v(y)\|_{l_1} . \tag{1.5}$$

В Интернете часто применяются меры *containment* C (степень вхождения одного документа x в y) и *resemblance* R (сходство документов x и y) между множествами X и Y, определяемые [61, 19] как

$$\begin{aligned} C(X,Y) &= |X \cap Y| / |X| \\ R(X,Y) &= |X \cap Y| / |X \cup Y|. \end{aligned} \tag{1.6}$$

При этом величина $1 - R(X,Y)$ является метрикой. Данный виды сходства применяется в Altavista [20], похожие алгоритмы использует Yandex [56], а судя по патенту [90], и Google.

Таким образом, задача оценки сходства между последовательностями в таких представлениях сводится к векторному случаю (п. 1.3.1) или к оценке сходства между множествами X, Y. В пользу частотных представлений говорят экспериментальные свидетельства, что учет только статистики наличия элементов в последовательностях позволяет сохранять сходство между текстами на уровне, достаточном для некоторых приложений [49, 124].

Недостатком подобных представлений является отсутствие учета порядка в последовательности (d_H и d_{l_p}) или же его учет лишь в той степени, насколько

он учитывается выбором элементов последовательности, из которых строятся соответствующие множества («bag-of-items»).

1.3.3. Расстояния редактирования

Во многих приложениях, работающих с последовательными данными, более адекватными являются расстояния редактирования. Наиболее известным является классическое расстояние редактирования Левенштейна [129], определяемое между символьными строками x и y как минимальное количество операций вставки, замены и удаления символов для преобразования x в y. Эти операции интерпретируются в задачах генетики как изменения, происходящие при мутациях и эволюции генов [51], в системах документооборота – как искажения при оптическом распознавании текстов, а также хорошо описывают некоторые способы искажения текстов для несанкционированной рекламы в Интернет [48] и обхода систем обнаружения вторжений в компьютерные системы [108].

Широко известен классический алгоритм вычисления расстояния Левенштейна [149, 119, 51, 150], имеющий для строк длиной n сложность $O(n^2)$ (п. 1.3.3). Ввиду больших характерных размеров n, большого количества строк и невозможности постоянного доступа к сравниваемым последовательностям в полном объеме, применение этого алгоритма в перечисленных выше областях затруднительно.

В обобщенном (взвешенном) варианте классического определения операциям редактирования (замене, вставке и удалению) присваиваются различные неотрицательные стоимости $c_{chg}(\sigma, \sigma')$ – замена символа σ на σ', $c_{ins}(\sigma)$ – вставка символа σ, $c_{del}(\sigma)$ – удаление символа σ. Для простоты примем, что эти операции удовлетворяют неравенству треугольника, иначе описанный ниже алгоритм сильно усложняется [150]. Преобразование $\tau : x \to y$ можно представить как последовательность операций редактирования $\tau = \tau_1, \ldots, \tau_N, \tau_j \in \{chg, ins, del\}$. Стоимостью некоторого преобразования $c(\tau)$ называется сумма входящих в него стоимостей каждой операции редактирования: $c(\tau) = \sum_j c_{\tau_j}$.

Под расстоянием редактирования $ed(x, y)$ понимается минимальная стоимость последовательности операций редактирования (замены, вставки и удаления), переводящая x в y:

$$ed(x, y) = \min_{\{\tau|\tau(x)=y\}} c(\tau). \tag{1.7}$$

В классическом варианте определения расстояния редактирования все стоимости равны единице независимо от символа σ.

Несмотря на то, что время вычисления $ed(x, y)$ полиномиально, существует необходимость ускорения его вычисления для длинных строк. На сегодня лучшим временем точного его вычисления – $O(\frac{n^2}{\log n})$ – обладает алгоритм из [76], что лишь немногим лучше оценки для классического алгоритма.

Отметим, что расстояние Хэмминга (1.1) является также расстоянием редактирования (определенной для строк одинаковой длины) с единственной допустимой операцией – заменой, и поэтому $d_H(x, y) \geqslant ed(x, y)$. Существуют другие варианты определения расстояния редактирования, отличающиеся множеством допустимых операций. Например, расстояние редактирования с дополнительной операцией транспозиции смежных символов [33] (частый вид ошибок при наборе с клавиатуры) и длина наибольшей общей подпоследовательности (LCS) (расстояние редактирования с разрешенными только вставками и удалениями), также имеющее квадратичную сложность вычисления [118]. Связанными задачами, основанными на взвешенном расстоянии редактирования, являются задачи поиска глобального [84] и локального [103] выравнивания в генетике, в том числе среди многих последовательностей [51]. Решения всех упомянутых вариантов основываются на динамическом программировании и поэтому полиномиально разрешимы (за квадратичное время).

Более общими и в некоторых приложениях (например, генетика) более адекватными являются расстояния редактирования, включающие помимо символьных операций также и операции над блоками символов. Так, операции копирования, перемещений и удаления (иногда только повторяющихся) блоков символов (а также иногда и изменение порядка символов в блоке), также являются и более общим описанием преобразований, часто встречающихся

при мутациях генов [51].

Первоначально вариант блочного расстояния редактирования (не являющегося метрикой) был предложен в [112]. Он представляет собой количество участков текста x, из которых может быть составлен текст y и вычисляется жадным алгоритмом. Варианты определения блочного расстояния редактирования включают LZ-расстояние (не метрика) и расстояние сжатия [31], а также расстояние редактирования с блочными перемещениями [30] и другие [75]. Точное вычисление общих вариантов расстояния редактирования с блочными операциями, как правило, NP полное [75, 101], в то же время они допускают очень эффективные аппроксимации [80, 31, 30, 27].

Значение расстояние редактирования с разрешенными блочными операциями (при условии включения классического набора символьных операций) является нижней границей на значение классического расстояния редактирования (1.7).

1.4. Методы аппроксимации и оценки сходства с помощью вложений пространств

Одним из способов преодоления недостатков трудновычислимых метрик и больших размерностей является изменение исходного пространства на более простое, где задача может быть решена легче. Под целевыми пространствами чаще всего понимаются векторные (или даже $\mathbb{R}^1$ [37]), желательно небольшой размерности и, если возможно, с простой метрикой типа ℓ_1 из (1.2), ℓ_∞ или хэммингoвой (1.1). Как и в случае использования векторных расстояний для сравнения последовательностей (п. 1.3), это объясняется тем, что в векторных пространствах существует множество алгоритмов для разных задач (k ближайших соседей, классификации, кластеризации и т.д.), а также тем, что для векторов меньших размерностей требуется меньше ресурсов. Кроме того, изменение исходного пространства может быть необходимо из-за невозможности точно посчитать исходную метрику (см. п. 1.3.3). Такое отображение

$X \xrightarrow{v} Y$ из исходного пространства в целевое называется *вложением* [57]. Если оба пространства метрические, то такое вложение называется *переключением метрик*, если метрика одинаковая, но уменьшается размерность – *уменьшением размерности*.

Вложения используются в различных прикладных задачах [57]. Текущие приложения включают экономную передачу и сжатие данных [31, 27], фильтрацию дубликатов в Интернет [19], сравнение и объединение записей в базах данных [28, 2, 24], сравнение XML-структур [43], генетику [21, 52]. Интуитивно, вложение должно «близкие» элементы в исходном пространстве отображать в «близкие» в целевом, «далекие» – в «далекие». Формально качество аппроксимации вкладываемой метрики оценивают с помощью понятия искажения. Если существует такое $c \geqslant 1$, что

$$\frac{1}{c} d'(v(x), v(y)) \leqslant d(x, y) \leqslant c \cdot d'(v(x), v(y)), \tag{1.8}$$

где d' и d – метрика соответственно целевого и исходного пространств, то оно называется *искажением вложения* [57].

Более слабым определением является понятие порогового (r_1, r_2, r'_1, r'_2)-вложения:

$$\begin{aligned} &\text{если } d(x, y) \leqslant r_1, \text{ то } d'(v(x), v(y)) \leqslant r'_1, \\ &\text{если } d(x, y) > r_2, \text{ то } d'(v(x), v(y)) > r'_2. \end{aligned} \tag{1.9}$$

Если (1.8) или (1.9) выполняются с некоторой вероятностью, то они называются *рандомизированными вложениями*.

1.4.1. Векторные метрики и меры сходства на множествах

Известным результатом является лемма Джонсона-Линденштрауса [62, 60, 34, 57], которая говорит о возможности вложения векторов из ℓ_2 размерности m в ℓ_2 размерности $m' < m$, при только лишь небольшой потере точности: существует рандомизированное вложение $\mathbf{A} : \ell_2^m \to \ell_2^{m'}$ с искажением $c = 1+\varepsilon$ и вероятностью ошибки $\delta = e^{\Omega(-m'/\varepsilon^2)}$. Вложение $\mathbf{A}$ линейно, и его матрица

содержит независимые и одинаково распределенные элементы из нормального распределения $N(0,1)$.

Таким образом, компоненты вектора-образа $v' \in l_2^{m'}$ получаются следующим образом:

$$v'_i = (\mathbf{A}v)_i = \sum_j a_{ij} v_j, \tag{1.10}$$

где v_j – компоненты исходного вектора $v \in l_2^m$, а $a_{ij} \sim N(0,1)$ – элементы матрицы $\mathbf{A}$. При уровне ошибки δ размерность целевого пространства составляет $m' = O(\frac{1}{\varepsilon^2}\log(\frac{1}{\delta}))$. Вместо распределения $N(0,1)$ можно выбирать элементы матрицы $\mathbf{A}$ из множеств $\{-1,1\}$ (равновероятно) или $\{-1,0,1\}$ ($p(1) = p(-1) = 1/6, p(0) = 2/3$) при неизменном искажении c [1].

Для пространства ℓ_1 показана невозможность подобного вложения [18] – т.е. уменьшение размерности для M точек в ℓ_1 с искажением c требует размерности порядка $M^{\Omega(1/c^2)}$ (см. также в [59] слабый вариант уменьшения размерности в ℓ_1). Однако для ℓ_1 существует [59] возможность создания векторов v' размерности $m' = O(\frac{1}{\varepsilon^2}\log(\frac{1}{\delta}))$ аналогично формуле (1.10), но с элементами a_{ij} из распределения Коши $p(x) = \frac{1}{\pi(1+x^2)}$. Тогда

$$(1-\varepsilon)d_{l_1}(v_1,v_2) \leqslant \text{median}(|\mathbf{A}v_1 - \mathbf{A}v_2|) \leqslant (1+\varepsilon)d_{l_1}(v_1,v_2).$$

Если вместо распределения Коши (1-стабильное распределение) взять другое p-стабильное распределение, то аналогичным способом можно добиться [29] аппроксимации ℓ_p метрик (1.2):

$$(1-\varepsilon)d_{l_p}(v_1,v_2) \leqslant B(p)\text{median}(|\mathbf{A}v_1 - \mathbf{A}v_2|) \leqslant (1+\varepsilon)d_{l_p}(v_1,v_2),$$

где $B(p)$ – определенный коэффициент.

Другие способы аппроксимации расстояния ℓ_1 приведены в [41], ℓ_2 и частотных моментов (1.3) – в [3].

Уменьшение размерности в случае метрики Хэмминга (1.1) возможно выполнить в смысле определения (1.9). Для этого в (1.10) операции сложения заменяются на сложение по модулю 2, а элементы бинарной матрицы выбираются из распределения Бернулли с вероятностью единицы, зафиксированной

на определенном малом значении. Можно показать [71], что существует такое $(r_1, r_1(1-\varepsilon), r', r'+1)$-вложение, что (1.9) выполняется с вероятностью $1-\delta$, размерность целевого хэммингова пространства при этом равна $m' = O(\frac{\log \frac{1}{\delta}}{\varepsilon^2})$.

Вложение меры, связанной с мерой сходства по косинусу (1.5) в хэммингово пространство возможно с помощью метода, предложенного в работе [23]. Из индикаторных функций $h(v, r) = [(v, r) \geqslant 0]$, где r – случайный m-мерный вектор с $r_i \sim N(0, 1)$, составляются бинарные векторы, которые могут быть использованы для поиска приближенно ближайших соседей в хэмминговом пространстве (п. 1.5.1).

Данные вложения и аппроксимации могут быть полезны для простейших способов учета последовательности элементов сравниваемых последовательностей векторными представлениями типа «bag-of-items» (п. 1.3.2).

Для меры resemblance сходства множеств (1.6) существует [19] рандомизированный способ оценки сходства путем своеобразного уменьшения размерности за счет сравнения множеств минимальных элементов случайной перестановки исходных множеств. При этом несмещенной оценкой $R(X, Y)$ из (1.6) является выражение $R(\min_s(\pi(X)), \min_s(\pi(Y))$, где π – случайная перестановка, определенная на $X \cap Y$, а $\min_s(\cdot)$ – множество из s минимальных элементов аргумента. Основываясь на таком подходе, в работе [60] предложен способ поиска приближенных ближайших по resemblance множеств (см. также п. 1.5.1).

1.4.2. Расстояния редактирования

Идея многих алгоритмов аппроксимации и вложения расстояния редактирования состоит в использовании того наблюдения, что близкие строки содержат много общих подстрок, часто значительно более коротких, чем сама строка. Такие методы часто называются q-граммными методами аппроксимации расстояния редактирования, исследование которых началось с работы [114], где доказана лемма Укконена, утверждающая, что для последовательностей $x, y \in \Sigma^n$ таких, что $ed(x, y) \leqslant k$, выполняется $d_q(x, y) \leqslant 2kq$, где $d_q(x, y)$ –

q-граммное расстояние (1.4).

Q-граммные методы основаны, в основном, на использовании числа входящих в строку q-грамм для оценки расстояния редактирования и широко используются для оценки верхней границы на расстояние редактирования в методах приближенного поиска строк, а также для фильтрации [83].

Вложение классического расстояния редактирования является известной открытой проблемой [57, 58].

В работе [5] доказано, что для классического расстояния редактирования искажение $c \geqslant 3/2$ для пространства ℓ_1. Однако, данный результат не конструктивен в том смысле, что не приведен алгоритма с таким искажением, и даже не доказана возможность его существования. В работе [66] показано, что $c = \Omega(\sqrt{\frac{\log n}{\log\log n}})$ и в работе [70] улучшено до $\Omega(\log n)$. Как и результат [5], данные результаты дают лишь теоретические пределы принципиально достижимой точности аппроксимации расстояния редактирования любыми алгоритмами, но не могут дать конкретного алгоритма построения таких вложений.

Используя определение вложения (1.9), переформулируем задачу поиска вложения как поиск таких отображения $v(\cdot)$ и величин d_1, d_2, k_1, k_2, что,

$$\text{если } ed(x,y) \geqslant k_1, \text{ то } d'(v(x),v(y)) \geqslant d_1, \tag{1.11}$$

$$\text{если } ed(x,y) \leqslant k_2, \text{ то } d'(v(x),v(y)) \leqslant d_2, \tag{1.12}$$

где d' – метрика целевого пространства.

В приложении к вложению расстояния редактирования задача поиска такого вложения называется k_1 vs. k_2 gapped edit distance problem (GEDP) [10]: получив на входе 2 строки длиной n, расстояние между которыми или меньше или равно k_1; или больше или равно k_2, решить какой из двух случаев имеет место. Чем меньше разница $(k_2 - k_1)$, тем точнее аппроксимируется расстояние редактирования. Например, классический алгоритм динамического программирования решает k vs. $k+1$ GEDP.

В работе [10] приводится q-граммный алгоритм, который для $k \leqslant \sqrt{n}$ решает k vs. $\Omega((k_1 n)^{2/3})$ GEDP используя объем памяти, независящий от n.

В [11] представлен алгоритм, строящий сокращенные представления строк для оценки расстояния редактирования за время $O(n^{\max(\alpha/2,2\alpha-1)})$. С их помощью решается $O(n^\alpha)$ vs. $\Omega(n)$ GEDP. Данный алгоритм не вкладывает строки в векторное пространство, а в качестве сокращенных представлений использует части строк, полученные случайным семплированием входной строки, а также использует наблюдение, что строки, находящиеся на малом расстоянии редактирования, имеют много похожих подстрок в близких позициях.

Наилучшим алгоритмом вложения в векторное пространство на сегодня является алгоритм вложения, приведенный в работе [86], с $c = 2^{O(\sqrt{\log n \log\log n})}$, что лучше, чем $O(n^\alpha)$ для любого $\alpha > 0$. Используя этот алгоритм, можно добиться решения k_1 vs. $k_1 2^{O(\sqrt{\log n \log\log n})}$ GEDP. Однако, требуемая при этом память квадратично растет от длины строк, что затрудняет его применение на практике.

В q-граммных методах утверждения о нижней границе вида (1.12) доказываются обычно следующим образом:

1. допускается, что $ed(x,y) > k_2$ и $d'(v(x),v(y)) < d_2$, где d_2 – некоторая величина;
2. используя значение d_2, находится для данного случая некоторая (не обязательно оптимальная) последовательность операций редактирования в виде определенного алгоритма и ее стоимость $\tilde{ed}(x,y)$;
3. подбираются параметры так, чтобы $\tilde{ed} < k_2$;
4. получается противоречие с $ed(x,y) > k_2$, поскольку по определению $ed(x,y) \leqslant \tilde{ed}(x,y)$.

Утверждения о верхней границе (1.11), как правило, доказываются применением леммы Укконена (п. 1.3.3, стр. 29).

В работе [10], на шаге 2, авторы ограничиваются рассмотрением операций редактирования только внутри одного интервала строки. Если две одинаковые q-граммы находятся рядом в двух строках с небольшим сдвигом, то строки редактируются в начале и в конце, чтобы «подогнать» их к друг другу.

Аппроксимация простейшего расстояния редактирования с одной опера-

цией – замены, т.е. расстояния Хэмминга (1.1), путем уменьшения размерности описана в п. 1.4.1.

В работе [30] рассматривается более общее определение расстояния редактирования с дополнительной операцией перемещения произвольных блоков и его вложение в манхетенново пространство с искажением $O(\log n \log^* n)$. В то же время, как указано в п. 1.3.3 точное вычисление этого расстояния является NP-полной задачей.

В работе [12] на основе методики Locally Consistent Partitioning [94] предложен метод аппроксимации расстояния редактирования с коэффициентом $\min\{n^{1/3+o(1)}, ed(x,y)^{1/2+o(1)}\}$ за практически линейное время $\tilde{O}(n)$, а также метод вложения пространства строк длиной n с метрикой редактирования в пространство строк $n/r, \forall r > 0$ с той же метрикой с искажением $c = \tilde{O}(r^{1+o(1)})$.

Таким образом, существует необходимость создания эффективных средств аппроксимации расстояния редактирования, поскольку простые векторные метрики хотя и позволяют эффективную аппроксимацию (см. п. 1.4.1), но не настолько адекватно отражают сходство последовательностей, как это требуется на практике.

1.5. Задача поиска приближенных ближайших соседей

Сначала рассмотрим задачу (точного) поиска ближайшего соседа (БС). Имеется множество X и набор объектов $P = \{x_1, \ldots, x_P | x_i \in X\}$ и входной запрос $y \in X$, тогда задача БС состоит в поиске хотя бы одного x^*, такого, что $\forall x \in P,\ ed(x,y) \geqslant ed(x^*,y)$.

Существует ряд проблем с поиском точного БС даже в векторных пространствах, получивших условное название «проклятие размерности». Для больших размерностей входного пространствасуществующие алгоритмы точного поиска ближайшего соседа сводятся к линейному поиску по P или требуют экспоненциально большой памяти [16], что часто не оправдано для применения на практике, даже в случае, когда метрика вычисляется просто (в случае

вложения в векторное пространство).

Однако, для многих приложений бывает достаточным поиск приближенного ближайшего соседа, что вызвало большое количество публикаций, связанных с разработкой таких алгоритмов. Дополнительным аргументом в пользу поиска приближенного соседа является то, что часто сами данные имеют некоторую точность, с которой они известны, поэтому применение точных алгоритмов не оправдано.

Задача поиска приближенного ближайшего соседа ε-БС состоит в поиске такого $x^* \in P$, что $\forall x' \in P,\ ed(x^*, y) \leqslant (1+\varepsilon) ed(x', y)$. Иначе говоря, необходимо найти строку, находящуюся от запроса не более чем в $(1+\varepsilon)$ раз дальше, чем его точный ближайший сосед x^*.

Определим шар радиуса k, состоящий из объектов, расстояния которых от центра шара t не превышает k: $S(t,k) = \{x : ed(x,s) \leqslant k\}$. Задача ε-БС сводится [60] к решению задачи (ε, k)-PLEB (Point Location in Equal Balls), которая формулируется как поиск алгоритма, который для любого запроса $y \in P$:

1. если существует $x \in P$, такое, что $y \in S(x,k)$, возвращает YES и любую из точек $x' \in P$, таких, что $y \in S(x', (1+\varepsilon)k)$;
2. если $y \notin S(x, (1+\varepsilon)k)$ для всех $x \in P$, возвращает NO;
3. если ближайшая точка $x \in P$, такая, что $k \leqslant ed(y,x) \leqslant (1+\varepsilon)k$, то возвращает или YES или NO.

Таким образом, можно решать задачу ε-БС с помощью решения задачи (ε, k)-PLEB. Задача (k_1, k_2)-PLEB является вариантом ε-PLEB задачи и формулируется как поиск алгоритма, который:

1. если существует $x \in P$, такое, что $y \in S(x, k_1)$, возвращает YES и любую из точек $x' \in P$, таких, что $y \in S(x', k_2)$;
2. в остальных случая возвращает NO.

1.5.1. Локально-чувствительное хеширование и распределенные представления

В работе [60] предложена схема локально-чувствительного хеширования (LSH) для решения задачи (k_1, k_2)-PLEB (вариант задачи (ε, k)-PLEB) и примененная там для расстояния Хэмминга и меры сходства resemblance (1.6). В [71] также была предложена похожая схема для расстояния Хэмминга.

Идея использования схемы LSH для поиска соседей состоит в том, чтобы, используя определенное количество некоторых хеш-функций из семейства H, добиться высокой вероятности коллизии (совпадения значений хеш-функций) между близко находящимися объектами, и низкой – для далеких объектов. Тогда, применяя такое же хеш-преобразование к запросу, мы проверяем, не равен ли вектор $g(x) = (h_1(x), h_2(x), \ldots, h_K(x))$, $h_i \in H$, составленный из хеш-значений функции, какому-нибудь из ранее посчитанных хеш-векторов объектов из P.

Генерируемые локально-чувствительными функциями векторы $g(x)$ можно рассматривать как распределенное векторное представление объекта x размерностью . Поскольку функции g после выбора из G зафиксированы, то распределенное представление одинаковых объектов будет одинаковым (псевдослучайным), а благодаря локальной чувствительности похожие объекты будут иметь похожие представления.

При совпадении найденный вектор $x \in P$ будет являться приближенным соседом к запросу. Насколько он будет отличаться от точного ближайшего соседа, зависит от свойств использованных хеш-функций.

Определение [60]: семейство функций $H = \{h : X \to U\}$ (где U – некоторое конечное или счетное множество значений) называется (k_1, k_2, p_I, p_{II})-чувствительным или просто локально-чувствительным, если для любых $x, y \in X$ и любой независимо и равновероятно выбранной хеш-функции $h \in H$ выполняется:

$$\begin{aligned} &\text{если } x \in S(y, k_1), \text{ то } Prob[h(x) = h(y)] > p_I, \\ &\text{если } x \notin S(y, k_2), \text{ то } Prob[h(x) = h(y)] < p_{II}. \end{aligned} \tag{1.13}$$

Для того, чтобы семейство H было «полезным», необходимо, чтобы выполнялось условие $k_1 < k_2$, т.е. «близкая» точка x должны находятся ближе к y, чем точки, считающиеся «дальними», и

$$p_{II} < p_I, \tag{1.14}$$

т.е. «близкие» точки должны вызывать коллизию хеш-функций с большей вероятностью, чем «дальние».

1.5.2. Решение задачи поиска приближенных ближайших соседей с помощью локально-чувствительного хеширования

Структуры данных и обработка запроса. На предварительном этапе подготовки структуры для приближенного поиска ближайшего соседа определяется семейство функций $G = \{g : X \to U^K\}$, где $g(x) = (h_1(x), \ldots, h_K(x))$, $h_i \in H$. Всего из семейства G выбирается случайно и равновероятно L штук функций $g_1, \ldots, g_L$. Создается также таблица, в ячейку которой заносятся строки x из базы P на основании значения хеш-вектора $g(x)$: каждый объект x попадает в ячейку с идентификатором, равным значению хеш-вектора, посчитанном на ней, а именно $(h_1(x), h_2(x), \ldots, h_K(x))$. Размер таблицы ограничен $O(|P|L)$, кроме того, еще необходимо хранить исходную базу – $O(n|P|)$. Построение таблицы завершено, когда каждая $x \in P$ занесена в соответствующую ячейку.

Процедура поиска. Для строки-запроса y вычисляются все хеш-векторы $g_i(y), i = 1, \ldots, L$ и проверяются соответствующие ячейки в таблице. Если проверяемая ячейка содержит объект $x^* \in S(y, k_2)$, мы возвращаем YES и x^*. Если после проверки $2L$ объектов не было найдено ни одного объекта из P в $S(y, k_2)$, возвращаем NO.

Условия корректной работы процедуры. В [60] доказывается, что можно добиться выполнения следующих условий с константной вероятностью (большей $1/2$), при которых описанная процедура поиска приближенного ближайшего соседа корректна:

1. если существует $x^* \in S(y, k_1)$, то должно выполняться $g_j(x) = g_j(x^*)$ для некоторого $j = 1, \ldots, L$;
2. количество коллизий хеш-функций с объектами вне $S(y, k_2)$ в L проверенных ячейках должно быть меньше $2L$:

$$\sum_{j=1}^{L} \left|(P - S(y; k_2)) \cap g_j^{-1}(g_j(y))\right| < 2L.$$

Увеличение размерности векторов (параметр K) используется для уменьшения вероятности коллизии на точках, лежащих вне $S(y, k_2)$. Увеличение количества копий (параметр L) необходимо для достижения константной вероятности ($> 1/2$) события 1. Это достигается это выбором значений K, L.

В теореме Индыка-Мотвани доказано, что, если H является (k_1, k_2, p_I, p_{II})-чувствительным семейством функций, и

$$K = \log_{\frac{1}{p_{II}}} |P| \tag{1.15}$$

$$L = |P|^{\rho}, \tag{1.16}$$

где

$$\rho = \ln(\frac{p_I}{p_{II}}), \tag{1.17}$$

тогда алгоритм решения (k_1, k_2)-PLEB задачи занимает $O(n|P| + |P|^{1+\rho})$ памяти, использует $O(|P|^{\rho})$ вычислений расстояния и $O(|P|^{\rho} \log_{\frac{1}{p_{II}}} |P|)$ вычислений хеш-функций [60]. Из $p_{II} < p_I$ и выражения для ρ следует, что время поиска сублинейно от $|P|$.

Для реализации описанной процедуры LSH можно использовать модификацию, называемую LSH-лес, которая позволяет находить ближайших соседей без обновления хеш-векторов при изменении размера базы P или параметров k_1, k_2 [13].

Основной сложностью в применении данного алгоритма решения задачи (ε, k)-PLEB в рассуждениях по примерам является нахождение семейства локально-чувствительных функций (продуцирующих распределенное представление), отражающих сходство с приемлемой степенью в данной прикладной области.

Так, в работе [35] для построения локально-чувствительного к ℓ_p-норме семейства хеш-функций предлагается использовать то свойство p-стабильных распределений [85], что линейные комбинации случайных величин ϕ_i из этого распределения распределены так же, как и одна случайная величина, умноженная на норму коэффициентов данной линейной комбинации [85]. Поскольку скалярное произведение линейно, то $(v_1, \bar{\phi}) - (v_2, \bar{\phi})$ распределено так же, как и $\|v_1 - v_2\|_{l_1} \bar{\phi}$. Поэтому, если поделить действительную ось на равные промежутки, то интуитивно понятно, что скалярные произведения векторов с близкой нормой на случайный вектор из соответствующего стабильного распределения будут попадать в одни и те же или близкие промежутки, и на этом основании можно построить локально-чувствительное семейство.

В случае $p = 1$ (1-стабильное распределение), для использования этого свойства хеш-функции задаются в виде

$$h(v) = \left\lfloor \frac{(v, \bar{\phi}) + b}{r} \right\rfloor, \tag{1.18}$$

где $r \in \mathbb{R}$ – некоторое число, b – равномерно распределенная случайная величина из $[0, r]$, а $\bar{\phi}$ – вектор элементов из 1-стабильного распределения Коши с плотностью $\frac{1}{\pi(1+x^2)}$. Можно показать [35], что для двух фиксированных векторов v_1, v_2, в зависимости от значения l_1-нормы разности векторов $c = \|v_1 - v_2\|_{l_1}$, вероятность коллизии хеш-фукции (1.18) равна

$$p(c) = \int_0^r \frac{1}{c} f(c)(1 - \frac{t}{c})dt, \tag{1.19}$$

где $f(\cdot)$ – функция плотности модуля случайной величины, распределенной по Коши. Функция $p(c)$ есть монотонно убывающая функция, поэтому, по определению (1.13), семейство функций (1.18) будет локально-чувствительным и их можно использовать в схеме LSH (п. 1.5.2).

Задача поиска (приближенных) ближайших соседей для последовательностей характеризуется теми же сложностями («проклятие размерности»), что и поиск соседей в векторном пространстве. А именно, для большинства наборов операций редактирования поиск ближайшей последовательности должен ис-

пользовать или экспоненциальную по длине последовательности память, или линейный поиск по базе (сравнение каждой $p \in P$ с запросом) [80].

В случае использования простейших векторных представлений для последовательностей существует множество алгоритмов, решающих задачу ε-БС в случае векторных пространств (например, работы [60, 71] для хэммингова расстояния, [35] для ℓ_1, [71] для ℓ_2). Так, для хэммингова пространства и пространства ℓ_1 предложены [60, 35] схемы на основе схемы LSH (п. 1.5.1).

В случае простейшего варианта метрики редактирования и наличия только операции замены, задача поиска приближенного ближайшего соседа сводится к случаю уменьшения размерности в хэмминговом пространстве (п. 1.4.1) или же может быть решена с помощью схемы LSH. В случае классической метрики редактирования не существует прямого построения локально-чувствительной функции, которая позволила бы применять метод LSH. Однако возможно его применение для векторных представлений, которые получаются в некоторых методах аппроксимации (блочного) расстояния редактирования с помощью вложений (п. 1.4.2).

Связанной задачей является задача pattern matching – приближенного поиска короткой сроки в длинной строке, для решения которой также может быть использована схема LSH для случая хэммингова расстояния между строками [8, 6].

1.6. Выводы по разделу 1

Для решения задач поиска и классификации, имеющих дело с последовательностями, возможно использование подхода на основе рассуждений по примерам. При этом актуальным является эффективный поиск близких примеров-последовательностей – базовая операция в данном подходе.

Для оценки сходства последовательностей широко используется классическое расстояние редактирования. Однако, в связи с ростом длин последовательностей, объемов данных в базах примеров и во входных потоках сложность

точного вычисления расстояния редактирования препятствует его эффективной оценке и поиску сходных последовательностей в базах примеров.

Существующие методы приближенного вычисления классического расстояния редактирования с помощью вложения пространства символьных последовательностей в векторное пространство, а также существующие методы поиска последовательностей обладают недостаточной точностью или же большой ресурсоемкостью.

Перспективными для быстрого нахождения сходства между объектами различной природы являются нейросетевые распределенные векторные представления, которые позволяют эффективное сравнение. Однако существующие подходы к распределенному представлению последовательностей нуждаются в теоретическом обосновании, должны быть также созданы и реализованы соответствующие методы, эффективность которых требуется исследовать в практических задачах. Для задачи теоретического обоснования распределенных представлений перспективно использование подходов метрических вложений.

Это обусловливает актуальность направленности диссертационной работы на разработку и исследование новых методов представления последовательностей, повышение эффективности решения задач классификации на их основе.

РАЗДЕЛ 2

РАЗРАБОТКА И ИССЛЕДОВАНИЕ ДЕТЕРМИНИРОВАННОГО МЕТОДА ВЛОЖЕНИЯ ПРОСТРАНСТВА ПОСЛЕДОВАТЕЛЬНОСТЕЙ С КЛАССИЧЕСКОЙ МЕТРИКОЙ РЕДАКТИРОВАНИЯ В МАНХЕТТЕНОВО ПРОСТРАНСТВО

Раздел посвящен разработке методов, направленных на повышение эффективности аппроксимации классического расстояния редактирования последовательностей путем его детерминированного вложения в векторное пространство. На основании математического аппарата графов де Брейна (п. 2.2) и оценок нижней и верхней границ на расстояние редактирования (п. 2.3) разработан и проанализирован метод детерминированного вложения (п. 2.4), улучшающий точность аппроксимации по сравнению с известным методом детерминированного вложения. Результаты анализа проверены путем численных экспериментов (п. 2.5).

2.1. Базовые обозначения и определения

Будем обозначать алфавит символов Σ, а множество строк длины $n \in \mathbb{N}$, заданных над Σ, как Σ^n. Символ, находящийся в позиции i строки x, будем обозначать $x[i]$. *Подстроку* $x[i]x[i+1]\dots x[j]$ строки x будем обозначать $x[i,j]$ и писать $x[i,j] \subseteq x$, а диапазоны позиций вида $[i,j]$ назовем *интервалами*. Подстроки вида $x[i,i+q-1], q \in \mathbb{N}$ называются *q-граммами* [114], а множество всех q-грамм строки – ее *q-спектром*. Длину строки x обозначим $|x|$.

Для $x \in \Sigma^n$ и $q \in \mathbb{N}$ *q-граммным вектором* строки будем называть вектор $v_{n,q}(x) \in (\mathbb{N} \cup \{0\})^{|\Sigma|}$, где каждой q-грамме $\sigma \in \Sigma^q$ соответствует элемент вектора $(v_{n,q}(x))_\sigma \in \mathbb{N} \cup \{0\}$, равный числу раз, которое σ встретилась в x: $(v_{n,q}(x))_\sigma = \sum_{i=1}^{n-q+1} [x[i,i+q-1] = \sigma]$. Когда это не вызывает неоднозначно-

стей, будем вместо $v_{n,q}(x)$ писать $v_q(x)$ или просто $v(x)$.

Для строк $x, y \in \Sigma^n$ *q-граммным расстоянием* $d_q(x, y)$, называется манхеттеново расстояние (ℓ_1-расстояние) между соответствующими q-граммными векторами $\|v_q(x) - v_q(y)\|_{l_1} = \sum_{\sigma \in \Sigma^q} |(v_q(x))_\sigma - (v_q(y))_\sigma|$ ([114], п. 1.3.2).

Последовательность операций редактирования, которые трансформируют строку x в y, будем называть *восстановлением* y по x.

2.2. Классификация структур графа де Брейна

Для некоторых $q_1, q_2 \in \mathbb{N}, q_2 > q_1$ обозначим $\Delta q = q_2 - q_1$, $q_m = \frac{q_1+q_2}{2}$, $Q = (\Delta q + 1)(\Delta q + 2)$, $t = w - q_2 - \Delta q$.

Графом де Брейна [36, 67, 100] $B(\Sigma; q)$ для алфавита Σ и параметра $q \geqslant 3$ называется направленный граф $G(V, E)$, множеству вершин V которого соответствуют все $(q-1)$-граммы, а множеству дуг $E \subset V \times V$ – все q-граммы в данном алфавите. Соответствующие дугам и вершинам q- и $(q-1)$-граммы назовем *метками*. При этом для символов $l_i \in \Sigma$ дуга с меткой $l_1 l_2 \dots l_q$ соединяет вершины с метками $l_1 l_2 \dots l_{q-1}$ и $l_2 l_3 \dots l_q$. Иначе говоря, вершины v и v' соединены направленой дугой тогда и только тогда, когда $\exists \sigma \in \Sigma, v' = (v \times |\Sigma| mod |\Sigma|^{q-1}) + \sigma$.

Каждой строке $x, |x| \geqslant q$ соответствует определенный *путь* π_x на графе де Брейна, состоящий из дуг, последовательно соединяющих вершины, метки которых есть последовательно входящие в строку $(q-1)$-граммы (рис. 2.2).

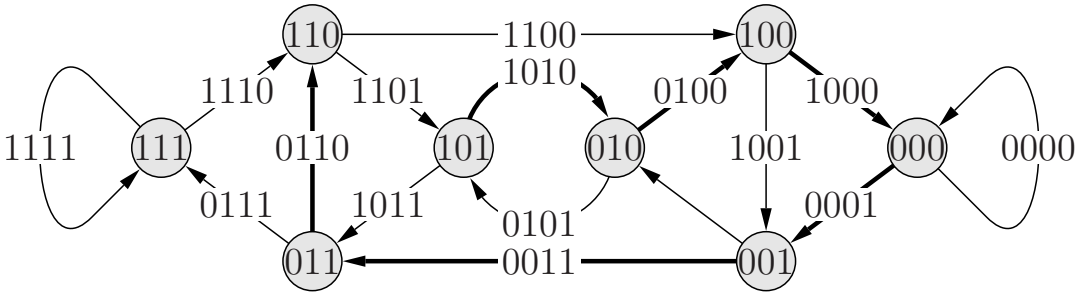


Рис. 2.1. Пример графа де Брейна $B(\{0, 1\}; 4)$. Путь π_x, соответствующий строке $x = 101000110$, выделен жирным

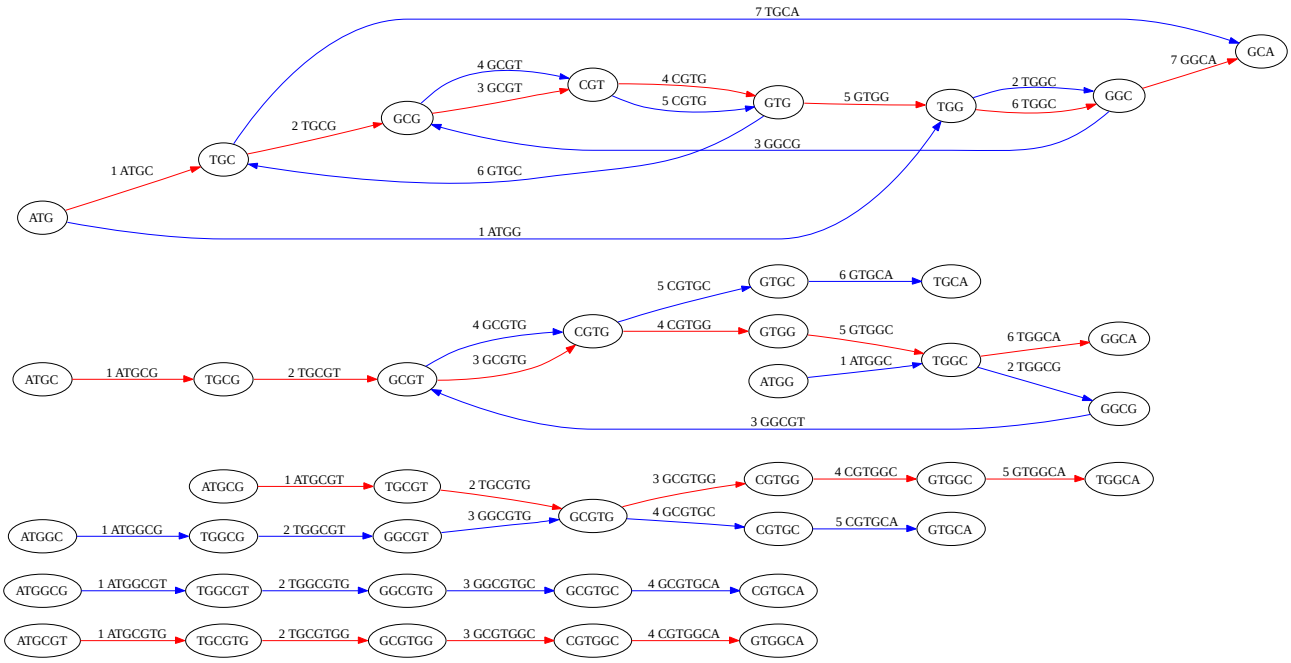


Рис. 2.2. Пример графов $B[\text{ATGCGTGGCA}, \text{ATGGCGTGCA}; q], q = 4, 5, 6$. Номер q-грамы в строке указан цифрой над соответствующей дугой

Подпутем π'_x пути π_x будем называть связную последовательность дуг π_x, соответствующую подстроке $x' \subseteq x$ и обозначать как $\pi'_x \subseteq \pi_x$. Путь π_x, состоящий из двух подпутей π'_x и π''_x, будем обозначать как $\pi_x = \pi'_x\pi''_x$.

Для $x, |x| \geqslant q$ обозначим $B[x; q]$ подграф графа де Брейна, т.е. пересечение графа де Брейна и дуг, входящих в строку, вершинами которого являются все $(q-1)$-граммы строки x (элементы $(q-1)$-спектра x), а дуги соответствуют ее q-граммам. Обозначим $B[x, y; q]$ граф, построенный на объединении $(q-1)$-спектров двух строк $x, y \in \Sigma^w$ одинаковой длины w: $B[x, y; q] = B[x; q] \cup B[y; q]$. При этом множество вершин объединенного графа есть объединение вершин исходных графов $V_{x,y} = V_x \cup V_y$, а множество дуг объединенного графа является мультимножеством, состоящим из всех дуг каждого из исходных графов (рис. 2.2).

Рассмотрим возможные локальные взаимные конфигурации путей π_x и π_y на $B[x, y; q]$, соответствующих строкам $x, y \in \Sigma^w$.

Строку $x, |x| \geqslant w$ назовем (q, w)-*неповторяющейся*, если в любом интервале из w символов все $w-q+1$ штук q-грамм различны. Строку $x \in \Sigma^w$, явля-

ющуюся (q, w)-неповторяющейся, будем называть просто q-*неповторяющейся*.

Рассмотрим два случая наличия повторов q-грамм:

Случай I. Обе строки $x, y \in \Sigma^w$ являются q-неповторяющимися.

Случай II. Хотя бы одна из строк $x, y \in \Sigma^w$ не является q-неповторяющейся.

2.2.1. Случай (q, w)-неповторяющихся строк

Сначала рассмотрим случай I. Если x и y содержат общую q-грамму с метками $l_1 \dots l_q$, то соответствующие им пути π_x и π_y на $B[x, y; q]$ будут проходить через одну и ту же дугу, соединяющую вершины с метками $l_1 \dots l_{q-1}$ и $l_2 \dots l_q$. *Левой* (вида -<) и *правой* (вида >-) *точками ветвления* назовем вершины, где пути π_x и π_y, соответственно, расходятся после общей дуги или сходятся перед общей дугой.

Например, вершина $l_2 \dots l_q$ будет правой точкой ветвления, если следующие за ней вершины, принадлежащие π_x и π_y, различны: $\ell_3 \dots l_q l_{q+1} \neq (l_3 \dots l_q l'_{q+1})$ или же один из путей закончился в вершине $l_2 \dots l_q$. Аналогично, вершина $l_1 \dots l_{q-1}$ – левая точка ветвления, если π_x и π_y достигают ее по разным дугам (или для только одного из путей вершина $l_1, \dots, l_{q-1}$ является начальной) и продолжаются дальше по общей дуге $l_2 \dots l_q$.

В рассматриваемом случае в $B[x, y; q], x \neq y$ можно выделить смежные участки вида >-<, -<, >-, т.е. содержащие как минимум одну общую дугу обоих путей π_x, π_y и ограниченные с одной или двух сторон точкой ветвления.

Для случая I введем понятия петли (полупетли), вилки (полувилки) и сдвига. Для каждого из двух путей совокупность левой точки ветвления, ближайшей к ней в порядке прохождения дуг правой точки ветвления, и дуг этого пути между ними (вида <_>) назовем *полупетлей* (на рис. 2.3 полупетлями являются участки обоих путей от вершины bcd до fgh, заключенные в пунктирные области, а вершины bcd и fgh являются, соответственно, правой и левой точками ветвления). Каждой полупетле соответствует также полупетля (возможно, нулевой длины), состоящая из дуг второго пути и соединяющая те же точки ветвления (тогда вместе эти полупетли имеют вид <=>).

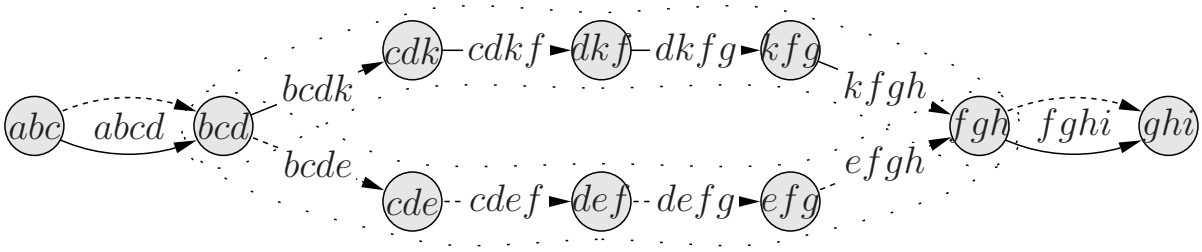


Рис. 2.3. Пример графа $B[abcdefghi, abcdkfghi; 4]$ и конфигурации «петля» для путей $\pi_{abcdefghi}$ (обозначен пунктирной линией) и $\pi_{abcdkfghi}$ (обозначен сплошной линией)

В случае конфигурации вида >-< (если левая и правая точки двух таких смежных конфигураций соединены полупетлей вида >-<_>-<) ей соответствует полупетля, соединяющая оставшиеся последнюю левую и следующую за ней первую правую точки ветвления. Такую пару соответствующих друг другу полупетель назовем *петлей*. По данному определению, как минимум одна из полупетель петли имеет ненулевою длину.

Граф $B[x, y; q]$ назовем *вилкой*, если существует такой общий для π_x, π_y подпуть $\pi_c, w > |\pi_c| > 0$, что $\pi_x = \pi'_x \pi_c \pi''_x$, а $\pi_y = \pi'_y \pi_c \pi''_y$, где для любых дуг $e \in \pi'_x \cup \pi''_x, e' \in \pi'_y \cup \pi''_y$ выполняется условие $e \neq e'$. Поскольку $|x| = |y|$, то $|\pi'_x| + |\pi''_x| = |\pi'_y| + |\pi''_y|$. Подпути $\pi'_x, \pi''_x, \pi'_y, \pi''_y$ левой или правой вилок, составленые из дуг одной строки, будем называть *полувилками*. *Сдвигом* назовем граф, являющийся частным случаем графа-вилки, где левая точка ветвления есть начальной вершиной для одного из путей и правая точка ветвления есть финальной вершиной для другого. Это соответствует ситуации, когда существуют такой общий для π_x и π_y подпуть $\pi_c, w \geqslant |\pi_c| > 0$, что $\pi_x = \pi'_x \pi_c$, а

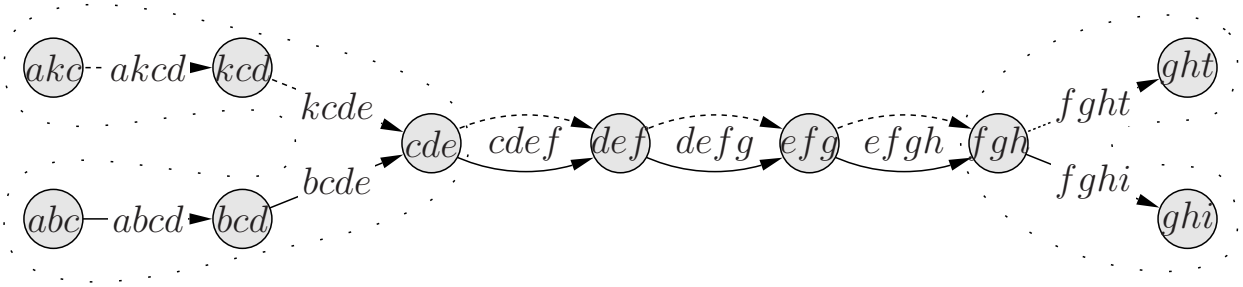


Рис. 2.4. Пример графа $B[abcdefghi, akcdefght; 4]$ и двух «вилок». Путь $\pi_{abcdefghi}$ обозначен сплошной линией и $\pi_{akcdefght}$ – пунктирной линией

$\pi_y = \pi_c\pi'_y$, где $\forall e \in \pi'_x, e' \in \pi'_y$ имеем $e \neq e'$. Подпути π'_x и π'_y в этом случае будем также называть полувилками. На рис. 2.2.1 вершины cde и fgh являются, соответственно, левой и правой точками ветвления. Правая вилка, обозначена пунктиром и состоит из правой точки ветвления fgh и дуг $fght$ и ghi, левая – из левой точки ветвления cde и дуг $akcd$, $kcde$, $abcd$ и $bcde$.

При такой конфигурации вилка, связанная с левой точкой ветвления, содержит начальную дугу одного из путей и не содержит дуги другого. А вилка, связанная с правой точкой ветвления, содержит последнюю дугу другого пути и не содержит дуги первого. При этом, одинаковые подстроки x', y', соответствующие пути π_c, сдвинуты относительно друг друга в x и y на величину $|\pi'_x| = |\pi'_y|$.

Если граф $B[x, y; q]$ является сдвигом, то будем говорить, что x является сдвигом y и наоборот.

На рис. 2.2.1 строка y есть сдвиг x. Путь π_x обозначен сплошной линией, путь π_y обозначен пунктирной линией. Подстроке $x' = cdefghi$, находящейся в обоих строках, со сдвигом, соответствует подпуть $\pi'_x = \pi'_y = (cde)(def)(fgh)(ghi)$. Вершины cde и ghi являются, соответственно, левой и правой точками ветвления, вилки (обозначеные пунктирными областями) которых содержат дуги только одного из путей – соответственно, $abcd$, $bcde$ и $ghik$, $hikt$.

Справедливы следующие утверждения о зависимостях числа общих дуг при наличии описанных конфигураций.

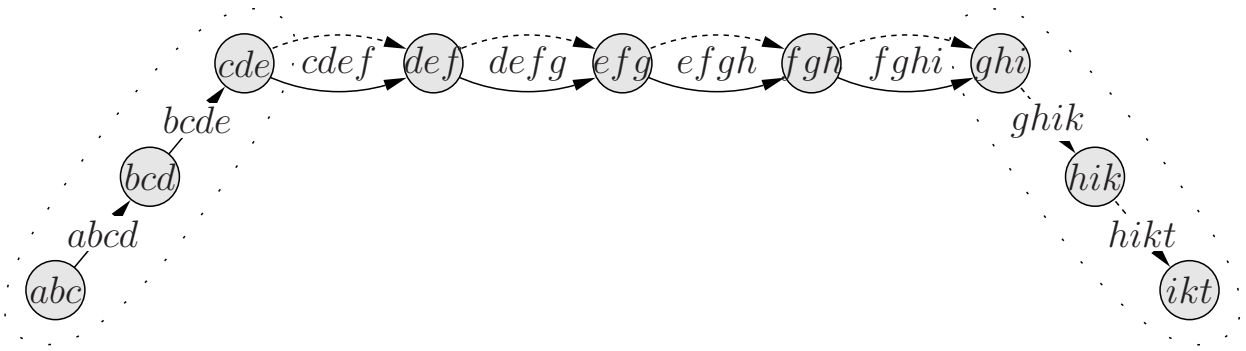


Рис. 2.5. Пример конфигурации сдвиг графа $B[x, y; 4]$ для $x = abcdefghi$, $y = cdefghikt$

Утверждение 2.1. *Если* $B[x, y; q]$ *является вилкой (сдвигом), то* $B[x, y; q+1]$ *также является вилкой (сдвигом) с тем же количеством дуг в полувилках.*

Утверждение 2.2. *Если* $B[x, y; q], x, y \in \Sigma^n$ *является вилкой с длиной общей части* $|\pi_c|$*, причем правая вилка состоит из двух полувилок длиной* $|\pi'_x| = s_1$ *и* $|\pi'_y| = s_2$*, а левая – с длиной* $|\pi''_x| = s_3$ *и* $|\pi''_y| = s_4$*, то для преобразования* x *в* y *достаточно выполнить не более* $\min(s_1, s_2) + \min(s_3, s_4)$ *операций замены и* $|s_2 - s_1| + |s_3 - s_4|$ *операций вставки или удаления. Так как* $s_3 = n - |\pi_c| - s_1, s_4 = n - |\pi_c| - s_2$*, то суммарно* $ed(x, y) \leqslant \max(s_1, s_2) + \max(s_3, s_4)$*. Заметим, что верхняя оценка также справедлива для* x, y*, граф которых не является вилкой, но* π_x *и* π_y *имеют некоторую общую часть* π_c*, а* π'_x *и* π'_y *(или* π''_x *и* π''_y*) могут частично совпадать.*

2.2.2. Случай (q, w)*-повторяющихся строк*

Случай II. Когда строки не являются (q, w)-неповторяющимися, у одной или обеих подстрок существует хотя бы один подпуть $\pi'_x \subseteq \pi_x$ ($\pi'_y \subseteq \pi_y$) на графе $B[x; q]$ ($B[y; q]$), соответствующий подстроке $x' \subseteq x$ ($y' \subseteq y$), имеющий самопересечение и потому являющийся циклом. Такой цикл имеет хотя бы одну вершину, встречающуюся более одного раза и, поэтому, как минимум одну повторяющуюся $(q-1)$-грамму. Заметим, что один и тот же цикл может соответствовать разным подстрокам x' (y') – для этого достаточно начать прохождение цикла с другой вершины. На рис. 2.2.2 путь π_x, соответствующий строке $x = abcderstcdefg$, обозначен сплошной линией, а путь π_y, соответствующий строке $x = oprstcderstvw$, обозначен пунктирной линией. Подстрока $x' = cderstcde$ есть вращение подстроки $y' = rstcderst$).

С рассматриваемым случаем II связано понятие вращения [114, 100]. Если записать $z \in \Sigma^l$ как последовательность $(q-1)$-грамм:

$$z = z[1, q-1]z[2, q]z[3, q] \dots z[i-1, i+q-3]z[i, i+q-2]z[i+1, i+q-1] \dots$$
$$\dots z[l-q+1, l-1]z[l-q+2, l],$$

и если $z[1, q-1] = z[l-q+2, l]$, то вращением называется операция, перево-

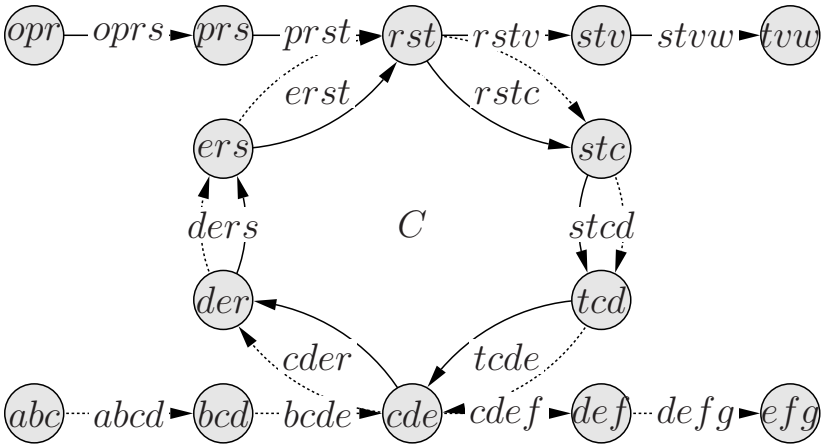


Рис. 2.6. Пример графа $B[x, y; 4]$ и общего цикла C, который каждый из путей проходит один раз

дящая z в строку z', такую что:

$$z' = z[i, i+q-1]z[i+1, i+q-1] \dots z[l-q+1, l-1]z[1, q-1]z[2, q]z[3, q+1] \\ \dots z[i-1, i+q-3]z[i, i+q-2]z[i, i+q-1],$$

где для некоторого $2 < i < l - q$ непустые подследовательности $(q-1)$-грамм $z[i+1, i+q-1] \dots z[l-q+1, l-1]$ и $z[2, q]x[3, q+1] \dots z[i-1, i+q-3]$ меняются местами. В этом случае говорят, что z является вращением z', и наоборот. Отметим, что вращением могут быть только строки, у которых совпадают первая и последняя $(q-1)$-граммы.

Если $|x'| = |y'|, x' \neq y'$, то при совпадении множеств дуг циклов, соответствующих x' и y', эти подстроки являются вращением друг друга.

Следующая лемма утверждает, что при достаточно большом q на $B[x; q]$, где x – (q, w)-неповторяющаяся строка, имеется не более, чем один цикл.

Лемма 2.1. *Для $x \in \Sigma^w, q > 2w/3$, если существует подстрока $x' \subseteq x$, такая, что $\pi_{x'}$ образует цикл C на $B[x; q]$, то одинаковые дуги вне цикла C отсутствуют.*

Доказательство. Пусть $f = |x'|$ и подстроки $x'' = x[i, i+q-2]$, $x''' = x[j, j+q-2], i < j$ являются парой максимально удаленных среди совпадающих $(q-1)$-грамм в $\pi_{x'}$. Поскольку $q > 2w/3$, то эти $(q-1)$-граммы пересекаются в как минимум $2\lceil 2w/3 \rceil - w$ символах и символы подстроки

$x[i, j-1]$ будут периодически повторяться подряд в x': если обозначить символы подстроки $x[i, j-1]$ как $b_1 b_2 \dots b_B$, то $x'[k] = b_{((k-1) \mod B)+1}, k = 1, \dots, f$.

Допустим, совпали две $(q-1)$-граммы $x[i', i'+q-2] = x[j', j'+q-2]$, $j' > i' > j$, т.е. $(q-1)$-граммы, выходящие своим правым концом за правую границу x'''. Поскольку $q > 2w/3$, то эти q-граммы полностью покрывают общий с x'' и x''' участок, содержащий как минимум одно вхождение всей последовательности символов $b_1, \dots, b_B$. А поскольку для подстроки $x[i', j'+q-2]$ справедливы аналогичные рассуждения про периодический характер (со своей повторяющейся последовательностью), то $\pi_{x[i,j+q-2]}$ и $\pi_{x[i',j'+q-2]}$ проходят те же дуги, что и цикл C. Аналогично рассматривается случай совпадения $(q-1)$-грамм до x''. □

Лемма 2.2. *Для $x, y \in \Sigma^w, q > 2w/3$, если первые (последние) вершины подпутей $\pi_{x'} \subseteq \pi_x, \pi_{y'} \subseteq \pi_y$, образующих общий цикл C, не совпадают, то общие дуги π_x и π_y перед (после) этими вершинами отсутствуют.*

Доказательство. Рассмотрим случай, когда подпути $\pi_{x'}, \pi_{y'}$, состоящие из дуг цикла C, оканчиваются в разных вершинах цикла. Пусть $x[i, i+q-1]$ и $y[j, j+q-1]$ – q-граммы, соответствующие последним дугам указанных подпутей в цикле C. Допустим, для $i' > i, j' > j$ совпали две q-граммы $\tilde{x} = x[i', i'+q-1] = y[j', j'+q-1] = \tilde{y}$. Так как $q > 2w/3$, то они содержат хотя бы одну последовательность символов $b_1, \dots, b_B$ из в доказательства леммы 2.1.

Пусть позиции правых концов $x[i, i+q-1]$ и $y[j, j+q-1]$ внутри, соответственно, $\tilde{x}$ и $\tilde{y}$ равны i'', j''. Поскольку подпути заканчиваются в разных вершинах, а $\tilde{x} = \tilde{y}$, то $i'' \neq j''$.

Допустим, $i'' < j''$. Из $\tilde{x} = \tilde{y}$ следует $\tilde{x}[i''+1, j''] = \tilde{y}[i''+1, j'']$, что эквивалентно $x[i+q, i+q-1+(j''-i'')] = y[j+q-(j''-i''), j+q-1]$. Поскольку и y', и x' составлены из периодически повторяющихся последовательностей символов $b_1, \dots, b_B$, то получаем, что, во-первых, x' заканчивается не q-граммой $x[i, i+q-1]$, а $x[i+(j''-i''), i+q-1+(j''-i'')]$, и, во-вторых, x' и y' заканчиваются в одной вершине, что противоречит условию леммы. □

Следующее утверждение говорит о характере поведения путей на графах

с циклами при изменении q.

Утверждение 2.3. *Если для $x \in \Sigma^w$ существует цикл на $B[x;q], q > 2w/3$, такой, что в нем присутствуют повторяющиеся дуги, то с каждым увеличением q на единицу длина участка пути между этими вершинами, проходящего более одного раза по одним и тем же дугам, сокращается. При этом общее количество уникальных дуг в цикле не изменяется. Если повторяющиеся дуги в цикле отсутствуют (т.е. каждая дуга цикла присутствует в π_x ровно один раз), то при следующем увеличении q на единицу цикл пропадает.*

Доказательство. По лемме 2.1 цикл единственный. Утверждение можно проверить, проследив на графе $B[x;q]$ с одним циклом изменение количества дуг в цикле и вне его при последовательном увеличении q. □

2.3. Нижнее и верхнее ограничение на расстояние редактирования

2.3.1. Ограничения на расстояние редактирования в пределах одного интервала

Необходимо найти такое определение расстояния между строками x и y и порог на его значение, чтобы для расстояний меньше этого порога (и при выполнении не зависящих от строк условий на определенные величины) граф $B[x,y;q]$ для случая I содержал бы сдвиг или вилку, и не содержал бы полупетель. Это позволило бы восстановить строки за небольшое число операций (чем меньше это число, тем точнее будет аппроксимация расстояния редактирования), используя утверждение 2.2. Не любое расстояние удовлетворяет этим требованиям: например, при использовании обычного q-граммного расстояния (ℓ_1-расстояние между q-грамными векторами) и при фиксированном q для любого допустимого значения q существуют строки x, y, где на графе $B[x,y;q]$ имеется петля. Тогда найти искомый порог не представляется возможным. Пример приведен на рис. 2.3.1, где для строк $x = abeeeegh$ (сплошная линия), $y = abeeeghi$ (пунктирная линия) и графов $B[x,y;3]$ и $B[x,y;4]$ значения q-граммных расстояний, соответственно, $d_3(x,y) = 2$ и $d_4(x,y) = 4$.

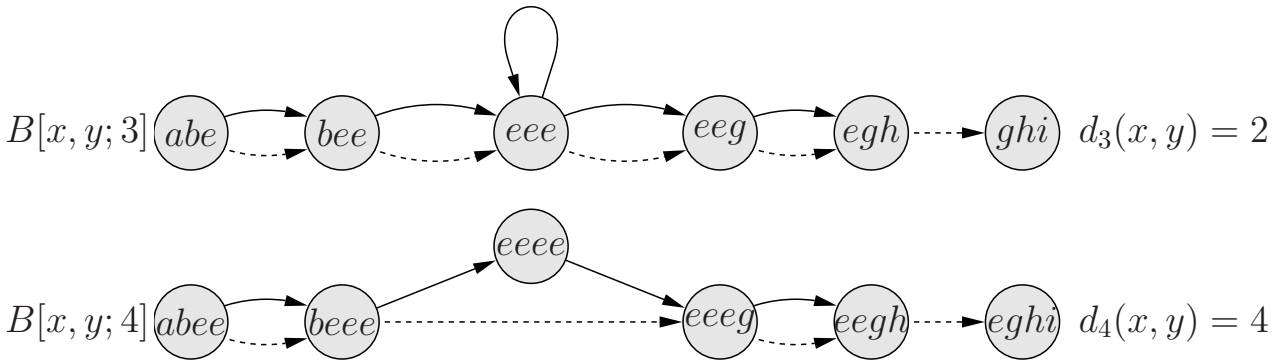


Рис. 2.7. Пример, иллюстрирующий неадекватность q-граммного расстояния для аппроксимации расстояния редактирования

В случае I, когда строки являются (q, w)-неповторяющимися, при увеличении q для полупетель и вилок существуют отличия зависимостей числа различных q-грамм. Поэтому для определения различия между полупетлей и вилкой предлагаем использовать следующее расстояние (основанное на обычном q-граммном расстоянии d_q):

$$d^{\Sigma}_{w,q_1,q_2}(x, y) = \sum_{q=q_1}^{q_2} d_q(x, y), \tag{2.1}$$

определенное для строк одинаковой длины w и фиксированных значений q_1, q_2 длины q-грамм. Покажем, что с его помощью можно определить наличие вилки или сдвига, т.е. возможность восстановления пары строк за небольшое число операций редактирования. Случай II наличия повторов q-грамм потребует отдельного рассмотрения (лемма 2.4).

Обозначим $\Delta q = q_2 - q_1$, $Q = (\Delta q + 1)(\Delta q + 2)$. Будем называть пару строк x, y *плохой*, если неравенство $d^{\Sigma}_{w,q_1,q_2}(x, y) < Q$ не выполняется, и *хорошей* в противном случае. Покажем, что для хорошей пары строк x, y граф $B[x, y; q_2]$ является вилкой (или сдвигом).

В следующей лемме рассматривается ситуация, когда для определенных значений q_1, q_2 $(q_2 > q_1)$ и w строки x, y являются (q_1, w)-неповторяющимися и найдем необходимые условия существования петли на графе $B[x, y; q_2]$ (при этом пара x, y будет плохой). Тогда невыполнение этих условий будет достаточным условием наличия сдвига или вилки при отсутствии общих циклов.

Лемма 2.3. *Пусть* $x, y \in \Sigma^w$ (q_1, w)*-неповторяющиеся,* $w \geqslant q_2 > q_1 \geqslant 3$*,*

$$\Delta q \leqslant \left\lfloor \frac{w - q_1 + 1}{2} \right\rfloor, \tag{2.2}$$

$$Q \leqslant 4(w - q_2 + 1), \tag{2.3}$$

$$d^{\Sigma}_{w,q_1,q_2}(x, y) < Q, \tag{2.4}$$

тогда $ed(x, y) < 2(\Delta q + 1)$.

Доказательство. Для любой полупетли в графе $B[x, y; q]$, имеющей как минимум по одной общей дуге, находящихся до и после, соответственно, левой и правой точек ветвления, увеличение q на единицу приводит к увеличению на единицу количества дуг в каждой из полупетель, участвующих в формировании петли от каждой строки.

На рис. 2.8 изображено два графа $B[x, y; q]$ и $B[x, y; q + 1]$, где на графе $B[x, y; q]$ имеется петля и левая вилка. При увеличении q на единицу (переходе к $B[x, y; q + 1]$) узлами $B[x, y; q + 1]$ становятся дуги $B[x, y; q]$, что обозначено пунктирными узлами на дугах $B[x, y; q]$. Количество дуг в каждой из полупетель увеличивается на единицу по сравнению с $B[x, y; q]$, в то время как количество дуг, формирующих полувилки, не изменяется.

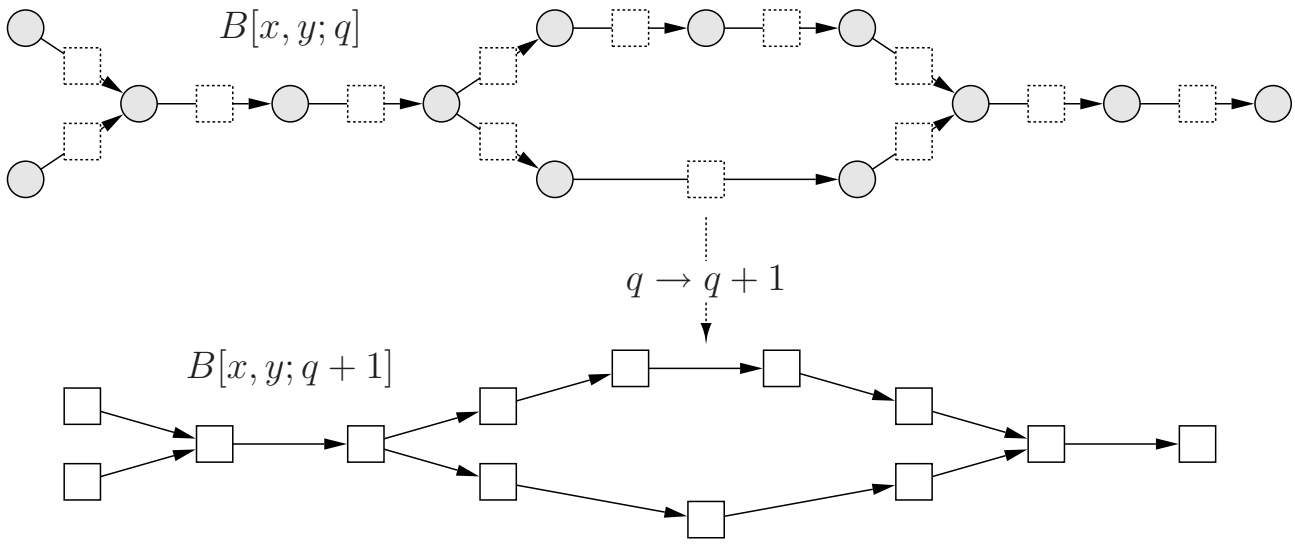


Рис. 2.8. Иллюстрация изменения количества различных дуг в зависимости от конфигурации путей на графе де Брейна

Поскольку по определению полупетель в $B[x,y;q]$ дуги одной полупетли не совпадают с дугами из соответствующей ей полупетли другой строки, $x[i,i+q-1] \neq y[i,i+q-1]$, для всех i таких, что $x[i,i+q-1]$ и $y[i,i+q-1]$ принадлежат своим полупетлям, то тем более не будет совпадений дуг петли в $B[x,y;q+1]$, т.к. метки дуг в $B[x,y;q+1]$ содержат метки дуг $B[x,y;q]$ как подстроки: $x[i,i+q-1] \to x[i,i+q] = x[i,i+q-1]x[i+q]$. Поэтому для строк x,y, содержащих хотя бы одну петлю, окруженную общими дугами, спрведливо, что общее ℓ_1-расстояние между $v_{q+1}(x)$ и $v_{q+1}(y)$ увеличивается как минимум на 2 по сравнению с ℓ_1-расстоянием для предыдущего значения параметра q: $d_{q+1}(x,y) \geqslant d_q(x,y)+2$.

Если же $B[x,y;q]$ является вилкой, то по утверждению 2.1 при увеличении q на единицу они сохранятся и $d_q(x,y)$ не изменится. Общее число дуг при этом уменьшается на единицу.

Поскольку при значении Δq большем, чем половина количества имеющихся в графе $B[x,y;q_1]$ дуг, ни одна полупетля не сохраняется, для сохранения полупетель наложим дополнительное условие $\Delta q \leqslant \lfloor (w-q_1+1)/2 \rfloor$ (2.2).

Учитывая, что минимальное q-граммное расстояние между различными строками равно двум, и при условии, что правая и левая точки ветвления петли не становятся, соответственно, первой или последней вершиной путей в $B[x,y;q]$, при $q < q_2$ (что обеспечивается условием (2.3)), получаем

$$\begin{aligned} d^{\Sigma}_{w,q_1,q_2}(x,y) &\geqslant \sum_{q=q_1}^{q_2} (d_{w,q_1}(x,y) + 2(q-q_1)) \\ &\geqslant 2(q_2-q_1+1) + (q_2-q_1)(q_2-q_1+1) \\ &= (q_2-q_1+1)(q_2-q_1+2) = (\Delta q+1)(\Delta q+2) = Q. \end{aligned} \tag{2.5}$$

Обратно, если для двух строк $x,y \in \Sigma^w$, $d^{\Sigma}_{w,q_1,q_2}(x,y) < Q$, то пути на $B[x,y;q_2]$ не образуют петлю. Тогда соответствующие им пути на графе $B[x,y;q]$ образуют вилку (одностороннюю или двухстороннюю) при некотором $q' \in [q_1,\ldots,q_2]$ и далее при $q > q'$.

Для утверждения, что при $d^{\Sigma}_{w,q_1,q_2}(x,y) < Q$ не возникает конфигурация

вида $=$ (отсутствие совпадающих вершин в путях π_y и π_x) вместо конфигураций сдвига или вилки, должно выполняться условие $|\pi_c| \geqslant 1$, содержащееся в определении вилки и сдвига, т.е. наличие как минимум одной общей дуги для $q = q_2 - 1$:

$$d_{q_2-1}(x, y) \leqslant 2(w - (q_2 - 1) + 1) - 2 = 2(w - q_2 + 1).$$

Если же $d_{q_2-1}(x, y) > 2(w - q_2 + 1)$ (т.е. нет ни одной общей дуги в $B[x, y; q_2 - 1]$), то

$$\begin{aligned} d^{\Sigma}_{w,q_1,q_2}(x, y) &= \sum_{q=q_1}^{q_2} \|v_{w,q}(x) - v_{w,q}(y)\|_{l_1} = \sum_{q=q_1}^{q_2-1} d_q(x, y) + d_{w,q_2}(x, y) \\ &> \sum_{q=q_1}^{q_2-2} 2(w - q + 1 - (q_2 - q)) + 2(w - q_2 + 1) \\ &= 2(w - q_2 + 1)(\Delta q + 1) > 4(w - q_2 + 1). \end{aligned} \tag{2.6}$$

Следовательно, если $d^{\Sigma}_{w,q_1,q_2}(x, y) \leqslant 4(w - q_2 + 1)$, то $d_{q_2-1}(x, y) \leqslant 2(w - q_2 + 1)$ и существует хотя бы одна общая вершина при $q = q_2$.

Таким образом, ввиду наличия петель на графе $B[x, y; q]$ для $q = q_1, \ldots, q_2$ с необходимостью следует $d^{\Sigma}_{w,q_1,q_2}(x, y) \geqslant Q$. Но, так как по условию (2.4) $d^{\Sigma}_{w,q_1,q_2}(x, y) < Q$, то, следовательно, петель на $B[x, y; q_2]$ быть не может, т.е. $B[x, y; q_2]$ является вилкой (сдвигом), либо пути на $B[x, y; q_2]$ имеют конфигурацию вида $=$. Последнее исключается согласно условию (2.3), как показано выше. Тогда граф $B[x, y; q_2]$ является вилкой (сдвигом).

По утверждению 2.2 для восстановления вилки при фиксированном q необходимо затратить не более $\max(s_1, s_2) + \max(s_3, s_4) \leqslant s_1 + s_2 + s_3 + s_4 = d_q(x, y)$ операций редактирования. Поскольку заранее не известно, при каком значении $q = q_1, \ldots, q_2$ образовалась вилка, то можно лишь утверждать, что $ed(x, y) \leqslant d_{q_2}(x, y)$.

Хотя отсюда уже и следует, что $ed(x, y) < Q$, т.к. $d_{q_2}(x, y) \leqslant d^{\Sigma}_{w,q_1,q_2}(x, y) < Q$, усилим это неравенство. При q_2 петли не может быть, т.к. $d^{\Sigma}_{w,q_1,q_2}(x, y) < Q$. Поэтому обозначим как $q^*, q_1 \leqslant q^* < q_2$ последнее значение q, при котором

еще имеется петля. Тогда $d^{\Sigma}_{w,q_1,q^*}(x,y) \geqslant (q^* - q_1 + 1)(q^* - q_1 + 2)$ и

$$\begin{aligned} Q > d^{\Sigma}_{w,q_1,q_2}(x,y) &= d^{\Sigma}_{w,q_1,q^*}(x,y) + (q_2 - q^*)d_{q^*}(x,y) \\ &\geqslant (q^* - q_1 + 1)(q^* - q_1 + 2) + (q_2 - q^*)d_{q^*}(x,y) \\ &\geqslant (q^* - q_1 + 1)(q^* - q_1 + 2) + (q_2 - q^*)d_{q_2}(x,y), \text{ т.е.} \end{aligned}$$

$$\begin{aligned} ed(x,y) \leqslant d_{q_2}(x,y) &< \frac{(Q - (q^* - q_1 + 1)(q^* - q_1 + 2))}{q_2 - q^*} \\ &= q_2 + q^* - 2q_1 + 3 \leqslant 2(\Delta q + 1). \end{aligned}$$

□

Рассмотрим теперь случай II наличия повторов q-грамм. Нас будет интересовать зависимость количества общих дуг на $B[x,y;q]$ у двух путей π_x и π_y от q. Отметим, что от случая I будут отличаться только те ситуации наличия циклов, где присутствует совпадение дуг одного пути с дугами другого пути (при этом совпавшая дуга в хотя бы одном цикле встречается больше одного раза) и где уменьшение количества дуг в утверждении 2.3 не компенсирует увеличение расстояния $d_q(x,y)$.

Поэтому ограничимся рассмотрением случаев, когда цикл, содержащийся в пути π_x и цикл, содержащийся в π_y, одинаковы при $q = q_1$, но могут отличаться как длины подпутей $\pi_{x'} \subseteq \pi_x$ и $\pi_{y'} \subseteq \pi_y$, принадлежащих циклу, так и вершины входа и выхода из него.

Лемма 2.4. *Пусть при $q_1 > 2w/3$ для строк $x, y \in \Sigma^w$ существуют такие подстроки $x' \subseteq x, y' \subseteq y, x' \neq y'$, что на графе $B[x,y;q]$ пути $\pi_{x'}$ и $\pi_{y'}$ состоят из дуг, принадлежащих общему циклу C. Пусть также $d^{\Sigma}_{w,q_1,q_2}(x,y) \leqslant Q$. Тогда $ed(x,y) < 2(\Delta q + 1)$.*

Доказательство. Допустим от противного, что $ed(x,y) \geqslant 2(\Delta q + 1)$. Покажем, что в этом случае $d^{\Sigma}_{w,q_1,q_2}(x,y) > Q$. Найдем минимально возможное значение $d^{\Sigma}_{w,q_1,q_2}(x,y)$ при заданных условиях леммы. Поскольку характер изменения $d_q(x,y)$ при наличии циклов может быть весьма сложным и разнообразным, а получение этой зависимости в замкнутом виде громоздко, для выяснения множества параметров, при которых может реализовываться минимум $d^{\Sigma}_{w,q_1,q_2}(x,y)$, воспользуемся следующими качественными соображениями,

основанными на сравнении зависимостей значений $d_q(x, y)$ от характера прохождения цикла обоими путями π_x, π_y.

Пусть меньшее число дуг цикла C между вершинами, соответствующими начальным вершинам (первым $(q_1 - 1)$-граммам) путей $\pi_{x'} \subseteq \pi_x$ и $\pi_{y'} \subseteq \pi_y$, равно s. Пусть L_x и L_y – длины подпутей, соответственно $\pi_{x'}$ и $\pi_{y'}$, принадлежащих циклу C.

Численно можно построить зависимости $d_q(x', y')$ от L_x и L_y для диапазона изменения $s = 0, \ldots, \lfloor L/2 \rfloor$, где L – количество дуг в цикле C (L постоянно, пока цикл существует, по утверждению 2.3). Если длина только одного из путей становится меньше L, то $d(x', y') \geqslant |L_x - L_y|$. На рис. 2.9 построена зависимость величины $d(x, y)$ от L_x, L_y для значений $s = 0, \ldots, \lfloor L/2 \rfloor$ с учетом приведенных соображений. Видно, что минимальные значения $d_q(x, y)$ достигаются при $L_x = L_y$ вне зависимости от s.

Поведение $d_q(x', y')$ в случае наличия одного цикла полностью определяется указанными параметрами s, L, L_x, L_y, пока $L_x > L$, $L_y > L$, поэтому можно рассматривать $d^{\Sigma}_{w,q_1,q_2}(x', y')$ как функцию от s, L, L_x, L_y. При увеличении длины q-граммы (рис. 2.9) $d_q(x', y')$ изменяется вдоль линий постоянного значения величины $|L_x - L_y|$ с уменьшением L_x и L_y на единицу. Из характера поверхностей видно, что если $L_x < L$ и $L_y < L$, то циклы в обоих графах $B[x; q]$ и $B[y; q]$ пропадают, и дальнейшее поведение $d_q(x', y')$ при увеличении q описывается леммой 2.3.

По лемме 2.2 совпадения дуг разных путей вне цикла возможны, если совпадают первые или последние вершины путей $\pi_{x'}$ и $\pi_{y'}$. Вклад в $d_q(x, y)$ дуг путей вне цикла будет не меньше $|L_x - L_y|$ при $L_x \geqslant L, L_y \geqslant L$ (что реализуется при $s = 0$ или совпадении последних вершин $\pi_{x'}$ и $\pi_{y'}$). Поэтому $d_q(x, y) \geqslant d_q(x', y') + |L_x - L_y|$.

Можно показать, что сумма вдоль такой линии ряда идущих подряд значений $d_q(x', y')$ для $s > 0$ будет не меньше, чем для $s = 0$ для этого же значения $|L_x - L_y|$ и в том же диапазоне суммирования $q_1, \ldots, q_2$.

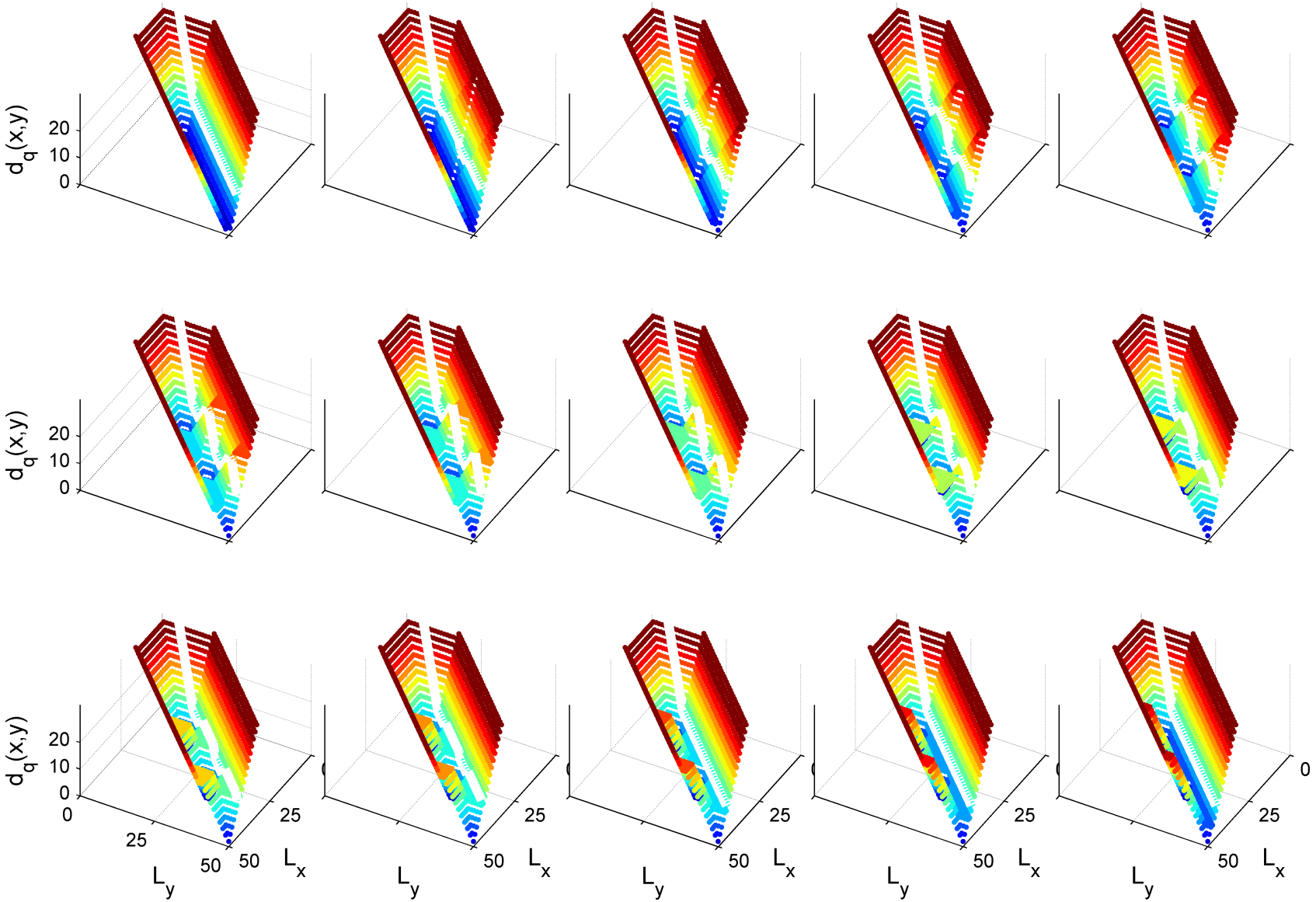


Рис. 2.9. Зависимости $d_q(x', y')$ от L_x и L_y для $s = 0, \ldots, L$, где $L = 17$. Для иллюстрации белой линией изображено множество точек, где $L_y = L_x + 5$

Тогда, если $(s', L'_x, L'_y) = arg\min_{s,L_x,L_y} d^{\Sigma}_{w,q_1,q_2}(s, L_x, L_y)$, то $d^{\Sigma}_{w,q_1,q_2}(s', L'_x, L'_y) \geqslant d^{\Sigma}_{w,q_1,q_2}(0, L'_x, L'_y)$. Поэтому рассмотрим только ситуацию $s = 0$, когда первые вершины подпутей $\pi_{x'}$ и $\pi_{y'}$ совпадают. Случай, когда при этом также $|L_x - L_y| = 0$, рассматривать не будем, т.к. он сводится к отсутствию цикла.

Учитывая, что согласно лемме 2.2 для рассматриваемых конфигураций можно применить утверждение 2.2, получаем, что для $|L_x - L_y| = \max(s_1, s_2) = \max(s_3, s_4)$ должно выполняться $|L_x - L_y| \geqslant \Delta q+1$. Иначе получим $ed(x, y) \leqslant \max(s_1, s_2) + \max(s_3, s_4) < 2(\Delta q + 1)$, что противоречит предположению в начале данного доказательства. Отсюда

$$d^{\Sigma}_{q_1,q_2,w}(x, y) \geqslant \sum_{q=q_1}^{q_2} 2|L_x - L_y| > 2(\Delta q + 1)^2 > (\Delta q + 1)(\Delta q + 2) = Q. \quad \square$$

Объединим лемму 2.3 с леммой 2.4 и сформулируем лемму 2.5.

Лемма 2.5. *Пусть* $x, y \in \Sigma^w$, $w \geqslant 6$, $w \geqslant q_2 > q_1 \geqslant 2w/3$ *и*

$$\Delta q \leqslant \lfloor (w - q_1 + 1)/2 \rfloor, \tag{2.7}$$

$$Q \leqslant 4(w - q_2 + 1), \tag{2.8}$$

Если $Q > d^{\Sigma}_{w,q_1,q_2}(x, y)$, *то* $ed(x, y) < 2(\Delta q + 1)$.

Доказательство. Из неравенств 2.7 и $q_1 \geqslant 2w/3$ следует $\Delta q \leqslant w/6$, поэтому значения Δq могут быть больше или равными единице при $w \geqslant 6$. Применяем в случае I (q_1, w)-неповторяющихся строк x, y лемму 2.3, в случае II – лемму 2.4. $\square$

Мы провели экспериментальную проверку леммы для тех значений параметра w, которые позволяли сделать полный перебор строк на стандартном PC с 1Гб памяти для всех удовлетворяющих условиям леммы значений q_1, q_2 из диапазона $[3, w]$. Для бинарного алфавита полный перебор 2^w строк был выполнен для значений $w = 4, \ldots, 15$ (это заняло около 50 часов). Для тернарного алфавита перебор 3^w строк был выполнен для значений $w = 7, \ldots, 10$ (около 90 часов). Во всех экспериментах не было найдено пары строк, которые бы опровергали утверждение леммы.

2.3.2. Распространение ограничений на расстояние редактирования на несколько интервалов

Введем понятие *хорошего* и *плохого* интервала аналогично понятию хорошей или плохой пары строк. Хорошим интервалом назовем такой интервал вида $[i, j]$, что для пары ограничиваемых им в x и y строк $x[i, j]$ и $y[i, j]$ выполняется $d^{\Sigma}_{w,q_1,q_2}(x[i, j], y[i, j]) < Q$ (выражение (2.4) при условиях леммы 2.5).

Основной идей, позволяющей нам улучшить предыдущие результаты, является возможность восстановления участков, которые можно рассматривать как сдвиг, малым числом операций, что уточняет нижнюю границу (k_2) в задаче GEDP (п. 1.4.2).

Ранее [10, 86] возможность восстановления сдвигов не рассматривалась и восстановление происходило по-q-граммно [10], или по-интервально, при выполнении определенных условий [86]. Поэтому, если, например, в одной строке имелся более длинный, чем длина интервала или q-граммы участок, входящий в другую строку в другой позиции, он восстанавливался по частям. Это приводило к завышению оценки стоимости редактирования и увеличению количества, необходимых для достижения нужной вероятности успеха, итераций вероятностных алгоритмов, основанных на таких вложениях. Отличие разработанного подхода от работы [10] в том, что мы соотносим длинные участки (не покрываемые одним интервалом), совпадающие в обеих строках, то есть те участки, где не применялись бы операции редактирования при преобразовании строк в друг друга. Эти участки могут находиться со сдвигом относительно друг друга. Тогда длинные (превышающие как q, так и w) совпадающие участки могут быть восстановлены за небольшое количество операций в их начале и конце при определенных соотношениях параметров q, w и других.

Далее рассмотрим интервал вида $[it'+1, it'+w]$, где i – натуральное число из определенного диапазона, т.е. интервалы, взятые с шагом t'. Следующая лемма говорит, что возможно «пакетное» восстановление строк, содержащих последовательно идущие хорошие интервалы, для $t' < t$, где $t = w - \Delta q + 1$.

Лемма 2.6. *Пусть* $x, y \in \Sigma^m$, $\mathbb{N} \ni t'$ *делит* $(m-w)$ *и* $t' < t$. *При выполнении условий леммы 2.5, если для всех* $i = 0, \ldots, \frac{m-w}{t'}$ *выполняется* $d^{\Sigma}_{w,q_1,q_2}(x[it'+1, it'+w], y[it'+1, it'+w]) < Q$, *то* $ed(x,y) < 2(\Delta q + 1)$.

Доказательство. Из леммы 2.3 следует, что при отсутствии совпадающих q_1-грамм для каждой пары удовлетворяющих условию леммы 2.5 строк $x' = x[it'+1, it'+w]$ и $y' = y[it'+1, it'+w]$ пути $\pi_{x'}$ и $\pi_{y'}$ на $B[x, y; q_2]$ образуют вилку или сдвиг с общим участком пути, состоящим как минимум из $w - q_2 + 1 - 2\Delta q$ дуг (условие (2.3)) и $ed(x', y') < 2(\Delta q + 1)$. Если на интервале $[it'+1, it'+w-1]$ имеются повторяющиеся q_1-граммы в $\pi_{x'}$ и/или в $\pi_{y'}$, то из леммы 2.4 следует, что $x[it'+1, it'+w], y[it'+1, it'+w]$ можно интерпретировать также как сдвиг или вилку (у соответствующих путей есть общий центральный участок дуг длиной как минимум $w - q_2 + 1 - 2\Delta q$, различные дуги могут быть только по краям интервала) и что также $ed(x', y') < 2(\Delta q + 1)$.

Рассмотрим два соседних интервала $[it'+1, it'+w]$ и $[(i+1)t', (i+1)t'+w]$. Поскольку $t' \leqslant t - 1 = w - \Delta q$, то они пересекаются не менее, чем Δq символами, что равно максимально возможному размеру полувилки у подпутей, ограниченных хорошими интервалами шириной w.

Правая вилка (сдвиг) на графе $B[x[it'+1, it'+w], y[it'+1, it'+w]; q_2]$ и левая вилка (сдвиг) на $B[x[(i+1)t', (i+1)t'+w], y[(i+1)t', (i+1)t'+w]; q_2]$, принадлежащие пересечению интервалов, состоят из одних и тех же дуг. Поэтому вариант, когда указанные графы являются вилками с непустыми полувилками (соответственно, правой и левой вилок) или сдвигами на разную величину или в разную сторону, исключается. Следовательно, интервал $[it'+1, (i+1)t'+w]$ весь является сдвигом или вилкой, причем размер сдвига или полувилок так же ограничен сверху Δq, как и у исходных интервалов. Следовательно, $ed(x[it'+1, (i+1)t'+w], y[it'+1, (i+1)t'+w]) \leqslant 2(\Delta q + 1)$.

Поскольку такие же рассуждения можно распространить на остальные интервалы, то $ed(x[1, m], y[1, m]) < 2(\Delta q + 1)$. □

Следствие 2.1. *Последовательность (при* $t' = 1$*) из смежных плохих интервалов, окруженная не менее чем одним хорошим интервалом с каждой*

стороны, состоит из, как минимум, t интервалов.

Доказательство. Пусть $t \geqslant 1$, иначе тривиально. Допустим, такая последовательность состоит из $t' \leqslant t$ плохих интервалов, тогда по лемме 2.6 все интервалы между ними должны быть хорошими. □

2.4. Детерминированный метод вложения расстояния редактирования в ℓ_1

В этом подразделе на основании утверждений, полученных в подразделах 2.2 и 2.3, получены временные, ресурсные и точностные характеристики детерминированного вложения.

В предлагаемом детерминированном методе вложения расстояния редактирования, входная строка $x \in \Sigma^n$ преобразовывается в вектор $v(x)$ конкатенацией всех q-граммных векторов $v_{i,w,q} = v_{w,q}(x[i, i + w - 1])$, для $q = q_1, \ldots, q_2$ и $i = 1, \ldots, n - w + 1$. В качестве расстояния между векторами принимается манхетенново (ℓ_1) расстояние.

Для строк $x, y \in \Sigma^n$ определим, с использованием введенного в (2.1) расстояния $d^{\Sigma}_{w,q_1,q_2}(x, y)$, расстояние

$$D(x, y) = \frac{\sum_{i=1}^{n-w+1} d^{\Sigma}_{w,q_1,q_2}(x[i, i + w - 1], y[i, i + w - 1])}{(n - w + 1)(\Delta q + 1)}, \tag{2.9}$$

и с использованием доказанных в предыдущем подразделе утверждений покажем, насколько хорошо аппроксимирует расстояние редактирования предлагаемый метод вложения.

2.4.1. Нижняя оценка стоимости редактирования

Будем находить ограничение снизу на количество плохих интервалов при $ed(x, y) > k_2$, где k_2 – параметр метода. Пусть количество плохих интервалов фиксировано, обозначим его N. Найдем сначала ограничение сверху на стоимость редактирования по всем возможным расположениям N интервалов, используя следующий алгоритм редактирования (восстановления) строк.

Будем последовательно переходить от интервала к интервалу, смещаясь на один символ. Допустим, строка восстановлена до позиции $j-1$. Если последовательно все интервалы, начиная с j-й позиции и до интервала в $(j+r)$-й позиции, хорошие, то они восстанавливаются за не более, чем $2(\Delta q+1)$ операций, используя результат леммы 2.6 для $t'=1$. Все интервалы в позициях, затронутых этим восстановлением (т.е. начинающиеся между позициями j и $j+r$), назовем *накрытыми*. Если следующий, интервал, начинающийся в j-й позиции, плохой, используем одну операцию редактирования, замещая $x[j]$ на $y[j]$ и восстанавливая таким образом один символ, и далее переходим к следующему интервалу, начинающемуся в $(j+1)$-й позиции.

Псевдокод изображен на рис. 2.10. Функция `Good`(x,y) возвращает `true`, если выполняются условия леммы 2.5 и $d^{\Sigma}_{w,q_1,q_2}(x,y)<Q$ и `false` в других случаях. Функция `EditShift`(x,y) выравнивает строки $x,\ y$, являющиеся сдвигом друг друга, затрачивая на это не более $2(\Delta q+1)$ операций редактирования (лемма 2.5). Функция `Replace`(a,b) заменяет символ a на символ b, используя одну операцию замены. Отметим, что нам не нужно в действительности использовать этот алгоритм, нас интересует только оценка числа операций редактирования.

Лемма 2.7. *Обозначим $S=\left\lfloor\frac{n-1}{w}\right\rfloor$. Восстановление строки y из x с помощью описанного алгоритма потребует не более операций, чем*

$$2(\Delta q+1)\max(N,2(\Delta q+1)\cdot\min(S,N)+\min[N,n-1-(w-1)\cdot \\ \cdot\min(S,N)])<2(\Delta q+1)(N+1). \tag{2.10}$$

Доказательство. Найдем расположение плохих интервалов, которое максимизирует стоимость редактирования по описанному алгоритму на рис. 2.10. По определению расстояния редактирования, это верхняя оценка $ed(x,y)$.

Максимальная стоимость редактирования будет достигаться при максимальном количестве возникновений ситуации, описанной в лемме 2.6. Поэтому расположение плохих интервалов, на котором достигается максимум стоимости редактирования, будет иметь вид повторяющихся серий вида +-===,

input : $x, y \in \Sigma^n$

output: edit distance upper estimate

$j \leftarrow 0$;

edit_dist $\leftarrow 0$ **while** $j \leqslant n - w + 1$ **do**

  if `Good(`$x[j, j + w - 1]$, $y[j, j + w - 1]$`)` **then**

    $r \leftarrow 0$;

    while `Good(`$x[j + r + 1, j + r + w]$, $y[j + r + 1, j + r + w]$`)` **do**

      $r \leftarrow r + 1$;

    end

    `EditShift` ($x[j, j + r + w - 1]$, $y[j, j + r + w - 1]$);

    edit_dist $\leftarrow$ edit_dist $+2(\Delta q + 1)$;

    $j \leftarrow j + w + r$;

  end

  else

    `Replace` ($y[j]$, $x[j]$);

    edit_dist $\leftarrow$ edit_dist $+1$;

    $j \leftarrow j + 1$;

  end

end

return edit_dist ;

Рис. 2.10. Алгоритм восстановления строк x, y

где плюсом (+) обозначен хороший интервал, минусом (-) – плохой, а знаком равенства (=) – накрытый хороший (достаточно только одного минуса, следующего за плюсом, для добавления $2(\Delta q + 1)$ операций в общую стоимость). Кроме того, в искомой конфигурации последний интервал будет хорошим, т.к. это дает дополнительные $2(\Delta q + 1)$ операций редактирования к суммарной стоимости.

Таким образом, общая стоимость редактирования всех конфигураций вида +-=== есть $2(\Delta q + 1)\min(S, N)$ (стоимость восстановления таких конфигураций, умноженная на максимальное их число). Остается или $n - w \cdot \min(S, N) - 1$

плохих интервалов (минус единица здесь из-за последнего, всегда хорошего, интервала), или $N - \min(S, N)$ – в зависимости от того, какое из этих выражений имеет меньшую величину.

Проанализируем, какие значения может принимать выражение (2.10). Если $S \geqslant N$, тогда

1. если $N < n-1-(w-1)N$, то $N < n-1/w$ и с учетом $N \leqslant \lfloor (n-1)/w \rfloor = S$ и $\lfloor n-1/w \rfloor < n-1/w$, получаем

$$ed \leqslant (2(\Delta q+1)-1)N + N + 2(\Delta q+1) = 2(\Delta q+1)(N+1);$$

2. $N > n-1-(w-1)N$ быть не может, т.к. тогда $N > n-1/w = S$, что противоречит $S \geqslant N$;
3. если $N = n-1-(w-1)N$, то $N = n-1/w = S$, отсюда

$$ed \leqslant (2(\Delta q+1)-1)N + n-1-(w-1)N + 2(\Delta q+1) = 2(\Delta q+1)(N+1).$$

Таким образом, из 1)–3), получаем, что если $S \geqslant N$, то

$$ed \leqslant 2(\Delta q+1)(N+1).$$

Если $S < N$, тогда

1. если $N < n-1-(w-1)S$, то, подставляя $S < N$,

$$ed \leqslant (2(\Delta q+1)-1)S + N + 2(\Delta q+1) < 2NQ + 2(\Delta q+1) = 2(\Delta q+1)(N+1);$$

2. если $N > n-1-(w-1)S$, то т.к. $S < N$, получаем $N > n-1/w \geqslant S$;

$$\begin{aligned} ed &< S(2(\Delta q+1)-w) + n - 1 + 2(\Delta q+1) < \\ &< N(2(\Delta q+1)-w) + n - 1 + 2(\Delta q+1) = \\ &= 2(\Delta q+1)(N+1) + n - 1 - wN < 2(\Delta q+1)(N+1); \end{aligned}$$

3. если $N = n-1-(w-1)S$, подставляя $S < N$, то как и в 2) получаем $N > n-1/w$ и

$$\begin{aligned} ed <& (2(\Delta q+1)-1)S + N + 2(\Delta q+1) = 2(\Delta q+1)(S+1) + N - S = \\ =& S(2(\Delta q+1)-1) + 2(\Delta q+1) + N < 2(\Delta q+1)(N+1). \end{aligned}$$

Итак, и в случае $S < N$, и в случае $S \geqslant N$, учитывая очевидное неравенство $N + 2(\Delta q + 1) < 2(\Delta q + 1)(N + 1)$, получаем верхнюю оценку $ed(x, y) < 2(\Delta q + 1)(N + 1)$. □

Следствие 2.2. *Пусть* $ed(x, y) > k_2$, $q_1 > 2w/3$, *а также выполняются условия леммы 2.5, тогда* $N > \frac{k_2}{2(\Delta q+1)} - 1$.

Доказательство. Допустим от противного, что число плохих интервалов $N \leqslant \frac{k_2}{2(\Delta q+1)} - 1$. Из леммы 2.7 следует, что $ed(x, y) < 2(\Delta q + 1)(N + 1) \leqslant k_2$. Получаем противоречие с предположением $ed(x, y) > k_2$. □

Учитывая следствие 2.1, которое говорит о группировке плохих интервалов в кластеры по не менее $t = w - q_2 - \Delta q$ подряд идущих интервалов, можно доказать следующую лемму, которая аналогична лемме 2.7.

Лемма 2.8. *Пусть число плохих интервалов для строк* x, y *длиной* n *фиксировано и равно* N. *Тогда*

$$ed(x, y) \leqslant 2(\Delta q + 1)(\left\lceil \frac{N}{t} \right\rceil + 1). \tag{2.11}$$

Экспериментальная иллюстрация лемм 2.7 и 2.8 приведена в п. 2.5.1.

Аналогично следствию 2.2, с учетом леммы 2.8 получаем следующее следствие.

Следствие 2.3. *Пусть* $ed(x, y) > k_2$, *тогда* $N > t(\frac{k_2}{2(\Delta q+1)} - 2)$.

Доказательство. Из $k_2 < ed(x, y) \leqslant 2(\Delta q + 1)(\lceil \frac{N}{t} \rceil + 1)$ получаем, что $\frac{N}{t} + 1 > \lceil \frac{N}{t} \rceil > \frac{k_2}{2(\Delta q+1)} - 1$, следовательно $N > t(\frac{k_2}{2(\Delta q+1)} - 2)$. □

2.4.2. Верхняя оценка на расстояние редактирования

Для $x, y \in \Sigma^w$ назовем q-грамму $x[i, i + q - 1]$ *k-хорошей*, если найдется такая q-грамма $y[i, i+q-1]$, что $|j - i| \leqslant k$, т.е. q-грамма в строке y, сдвинутая не более, чем на k символов. Если такой q-граммы в y нет, то назовем $x[i, i + q - 1]$ *k-плохой*.

Интервал $[i, i+q-1]$ назовем *k-хорошим*, если $B[x[i, i+q-1], y[i, i+q-1]; q]$ является сдвигом и число k-хороших q-грамм в обеих строках больше или равно $w - q + 1 - k$, откуда $d_q(x[i, i + q - 1], y[i, i + q - 1]) \leqslant 2k$. И *$k$-плохим* в противном случае.

Пусть $N[R]$ – количество интервалов $[i, i+w-1]$, где выполняется некоторое условие R, налагаемое на эти интервалы.

Лемма 2.9. *Для* $x, y \in \Sigma^n$ *и* $q_1 < q_2 \leqslant w \leqslant \frac{n+1}{k_1+1}$, *если* $ed(x,y) \leqslant k_1$, *то*

$$N[d^{\Sigma}_{w,q_1,q_2}(x[i,i+w-1], y[i,i+w-1]) \leqslant 2k_1(\Delta q+1)] > n - w(k_1+1) + 1.$$

Доказательство. Подсчитаем минимальное количество k_1-хороших интервалов при $ed(x,y) \leqslant k_1$ согласно лемме Укконена (п. 1.3.3, стр. 29). Всего имеется $n-w+1$ интервалов. Поскольку каждая операция редактирования может изменить максимум w интервалов, то общее количество k_1-плохих интервалов не превышает $k_1 w$. Оставшиеся $n-w+1-k_1 w$ интервалов будут k_1-хорошими, поскольку могут сдвинуться максимум на k_1 символов. Для k_1-хорошего интервала $[i, i+w-1]$ $d^{\Sigma}_{w,q_1,q_2}(x[i,i+w-1], y[i,i+w-1]) = \sum_{q=q_1}^{q_2} d_q(x[i,i+w-1]), y[i,i+w-1]) < 2k_1(\Delta q+1)$. □

2.4.3. Объединение оценок

С использованием следствия 2.3 и леммы 2.9 можно определить верхнюю и нижнюю границу на $D(x,y)$ при, соответственно, $ed(x,y) \leqslant k_1$ и $ed(x,y) > k_2$. Положим для этого $w = n^{\gamma}, \gamma \leqslant 1 - \frac{\ln(k_1+1)}{\ln n}$.

Обозначим множество позиций, где начинаются плохие интервалы, как $I_B = \{i = 1, \ldots, n-w+1 \mid d^{\Sigma}_{w,q_1,q_2}(x[i,i+w-1], y[i,i+w-1]) \geqslant Q\}$, и множество позиций, где начинаются хорошие интервалы, как $I_G = \{i = 1, \ldots, n-w+1 \mid d^{\Sigma}_{w,q_1,q_2}(x[i,i+w-1], y[i,i+w-1]) < Q\}$. Тогда, учитывая следствие 2.2, получаем

$$\begin{aligned}(n-w+1)(\Delta q+1)D(x,y) = \\ = \left(\sum_{i\in I_B} + \sum_{i\in I_G}\right) \sum_{q=q_1}^{q_2} d_q(x[i,i+w-1], y[i,i+w-1]) \geqslant \\ \geqslant NQ + 0 \geqslant Q(\frac{k_2}{2(\Delta q+1)} - 1).\end{aligned}$$

Учитывая группировку плохих интервалов (следствие 2.3), получаем

$$D(x,y) \geqslant \frac{Qt(\frac{k_2}{2(\Delta q+1)} - 2)}{(n-w+1)(\Delta q+1)} = d_2. \qquad (2.12)$$

Обозначим множество позиций, где начинаются k_1-плохие интервалы, как $I_B^{k_1} = \{i = 1, \ldots, n-w+1 \mid \exists\, j \in [i-k_1, \ldots, k_1+i],\ y[j, j+q-1] = x[i, i+q-1]\}$, а где начинаются k_1-хорошие – как $I_G^{k_1} = \{i = 1, \ldots, n-w+1 \mid \nexists j \in [i-k_1, \ldots, k_1+i],\ y[j, j+q-1] = x[i, i+q-1]\}$.

$$
\begin{aligned}
&(n-w+1)(\Delta q+1)D(x,y) = \\
&\qquad = \left(\sum_{i\in I_B^{k_1}} + \sum_{i\in I_G^{k_1}}\right) \sum_{q=q_1}^{q_2} d_q(x[i, i+w-1], y[i, i+w-1]) \leqslant \\
&\qquad \leqslant (\Delta q+1)[\left|I_B^{k_1}\right|(2w+2-q_2-q_1) + ((n-w+1) - \left|I_B^{k_1}\right|)2k_1] \leqslant \\
&\qquad \leqslant 2k_1[w(w+1-\frac{q_1+q_2}{2}-k_1) + (n-w+1)] < \\
&\qquad < 2k_1[w^2+(n+1)](\Delta q+1), \text{ откуда} \\
&\qquad D(x,y) \leqslant \frac{2k_1[w^2+(n+1)]}{n-w+1} = d_1. \qquad (2.13)
\end{aligned}
$$

Таким образом, доказана следующая теорема

Теорема 2.1. *Пусть* $w \geqslant 6$, $k_1 \geqslant 1$, $q_1 = 2w/3$, $n > w(k_1+1)+1$, $\Delta q = \frac{1}{2}(-7+\sqrt{57+16(w-q_1)})$, $Q = (\Delta q+1)(\Delta q+2)$, $t = w-\Delta q+1$, *тогда*

$$\textit{если } ed(x,y) \leqslant k_1, \textit{ то } D(x,y) \leqslant \frac{2k_1[w^2+(n+1)]}{n-w+1} = d_1, \qquad (2.14)$$

$$\textit{если } ed(x,y) > k_2, \textit{ то } D(x,y) \geqslant \frac{Qt(\frac{k_2}{2(\Delta q+1)}-2)}{(n-w+1)(\Delta q+1)} = d_2. \qquad (2.15)$$

Построение векторов $v(\cdot)$ с помощью префиксного дерева потребует порядка $O(n(q_2+w))$ операций. Соответственно, общая оценки времени построения равна $O(n^{1+\gamma})$.

Размерность вектора $v(\cdot)$ при условии фиксированого алфавита определяется количеством уровней префиксного дерева $\Delta q = O(n^{\gamma/2})$ и количеством узлов на каждом уровне – $O(n)$, поэтому итоговая размерность имеет порядок всего $O(n^{1+\gamma/2})$. Такой же порядок имеет и время вычисления расстояния между векторами.

Из (2.12) и (2.13) следует, что для обеспечения $d_2 > d_1$ необходимо, чтобы $k_2 = \Omega(k_1(n^{\gamma}+n^{1-\gamma}))$. Отсюда, оптимальным значением будет $\gamma = 1/2$,

при этом $k_2 = \Omega(k_1 n^{1/2})$ и время построения векторов $v(\cdot)$ – $O(n^{3/2})$, а их размерность – $O(n^{5/4})$.

2.5. Численное исследование детеминированного метода вложения

В данном подразделе проводится проверка некоторых из доказанных лемм с помощью численных экпериментов.

2.5.1. Экспериментальное исследование результатов лемм 2.7 и 2.8

Для экспериментальной проверки и иллюстрации доказательства лемм 2.7 и 2.8 сведем задачу к задаче нахождения разметки в $(\max, +)$-задаче на цепочечной структуре (см., например, [150]) с $2(w+1)$ состояниями.

Пусть K, $|K| = 2(w+1)$ – это следующее множество состояний:

1. $\boxed{+}$ – хороший интервал,
2. $\boxed{-}$ – плохой интервал,
3. $\boxed{+_i^c}$ – накрытый хорошой интервал, $i = 1, \ldots, w$,
4. $\boxed{-_i^c}$ – накрытый плохой интервал, $i = 1, \ldots, w$.

Построим граф соседства [150] следующим образом. Каждому интервалу в строке подставим в соответствие вершину, состояние или метка которой может принимать значения из описанного множества. Индексами $i = 1, \ldots, w-1$ указывается порядковый номер накрытого состояния внутри участка, восстанавливаемого за $2(\Delta q + 1)$ операций по лемме 2.6.

Вершины, соответствующие соседним интервалам, соединим дугами, вес которых определяется функциями переходов $g(t, t')$ между состояниями $t, t' \in K$, которые определяют разрешенные переходы и их стоимость. Функции переходов $g(t, t')$ определяются из алгоритма восстановления 2.10 и задаются следующим образом:

- $g(+, -) = -\infty$ – запрещен, т.к. после хорошего интервала начинаются накрытые состояния,
- $g(-, +) = 1$ – одна простая замена при нахождении в плохом интервале,
- $g(+, +) = 0$ – переход без стоимости,

- $g(-,-) = 1$ – одна замена, аналогично случаю $g(-,+)$,
- $g(+_i^c, -_j^c) = \begin{cases} 0, & j = i+1 \text{ разрешается переход на следующее по} \\ -\infty, & j \neq i+1 \text{ счету накрытое состояние,} \end{cases}$
- $g(+_i^c, +_j^c) = \begin{cases} 0, & j = i+1 \text{ – разрешается переход на следующее по} \\ -\infty, & j \neq i+1 \text{ счету накрытое состояние,} \end{cases}$
- $g(-_i^c, -_j^c) = \begin{cases} 0, & j = i+1 \text{ – разрешается переход на следующее по} \\ -\infty, & j \neq i+1 \text{ счету состояние,} \end{cases}$
- $g(-_i^c, +_j^c) = \begin{cases} 0, & j = i+1 \text{ – разрешается переход на следующее по} \\ -\infty, & j \neq i+1 \text{ счету накрытое состояние,} \end{cases}$
- $g(-_i^c, -) = \begin{cases} 0, & j = w-1 \text{ – выход из накрытого плохого состояния} \\ -\infty, & j \neq w-1 \text{ возможен только в } w\text{-м состоянии,} \end{cases}$
- $g(-_i^c, +) = \begin{cases} 0, & j = w-1, \\ -\infty & , j \neq w-1 \end{cases}$ – аналогично случаю $g(-_i^c, -)$,
- $g(-, -_i^c) = -\infty$ – входа в накрытое состояние из плохого интервала нет,
- $g(+, -_i^c) = \begin{cases} 2(\Delta q + 1), & i = 1, \\ -\infty, & i \neq 1 \end{cases}$ – при первом входе в накрытое состояние (всегда плохое) платим полную стоимость,
- $g(+_i^c, -) = \begin{cases} 0 & , i = w-1, \\ -\infty & , i \neq w-1 \end{cases}$ – аналогично случаю $g(-_i^c, -)$,
- $g(+_i^c, +) = \begin{cases} 0 & , i = w-1, \\ -\infty & , i \neq w-1 \end{cases}$ – аналогично случаю $g(-_i^c, -)$,
- $g(+, +_i^c) = -\infty$ – хороший интервал не накрывает другой хороший, т.к. они могут быть восстановлены вместе,
- $g(-, +_i^c) = -\infty$ – плохой интервал не накрывает ничего.

Участок получаемого графа приведен на рис. 2.11. Дуги нулевой стоимости обозначены пунктиром, запрещенные дуги, стоимость которых равна $-\infty$,

не показаны.

Разметкой графа назовем присвоение каждой вершине метки из множества K. Будем искать разметку $\bar{k}^* = \arg\max_{\bar{k},|k|\_=N} \sum_{i=2,\dots,n} g(k_{i-1}, k_i)$ (тут $|k|\_$ – количество плохих интервалов, как накрытых, так и нет).

Поскольку структура полученного графа – цепочка, задача $(\max, +)$ без ограничения $|k|\_ = N$ решается с помощью динамического программирования за время $O(nw^2)$ [150]. За время $O(n(wN)^2)$ находится максимальный путь, содержащий фиксированное число N плохих интервалов (условие $|k|\_ = N$). Это достигается введением дополнительных копий состояний на каждом шаге i, для каждого из возможных $0, \dots, N$ текущих значений количества плохих интервалов $|k_1 \dots k_i|\_$, которые к этому шагу содержатся в $k_1 \dots k_i$ (на рис. 2.11 не показаны).

Экспериментально полученные разметки для разных значений w приведены в таблице 2.12 при $N = \left|\bar{k}\right|\_ = 10, 20$ и длине последовательности $\left|\bar{k}\right| = 68$. Для данного примера $2(\Delta q + 1) = 10$. Значение $4(\Delta q + 1)(N + 1)$ при $N = 10, 20$ соответственно равно 220 и 420. Приведенная последовательность $\bar{k}^*$ хороших $(+)$ и плохих $(-)$ интервалов каждой строке максимизирует расстояние редактирования по алгоритму 2.10 для изменяющейся ширины интервала $w = 2, \dots, 68$ (первый столбец). Накрытые состояния $(+)$ и $(-)$ обозначены, соответственно, $(=)$ и $(\_)$. Реальное количество операций редактирования, возвращаемое алгоритмом на рис. 2.10, и значение по формуле (2.10) приведены, соответственно, в предпоследнем и последнем столбцах.

Аналогичные разметки, с учетом группировки плохих интервалов (лемма 2.8), приведены в таблице 2.13 для тех же значений N, w. Минимальная длина последовательсности из плохих интервалов $t = 3$ и $t = 5$. Значение $4(\Delta q + 1)(\lceil \frac{N}{t} \rceil + 1)$ при $N = 10$, $t = 3$ и $N = 20$, $t = 5$ равно 100. Реальное количество операций редактирования, возвращаемое алгоритмом на рис. 2.10, приведено в последнем столбце.

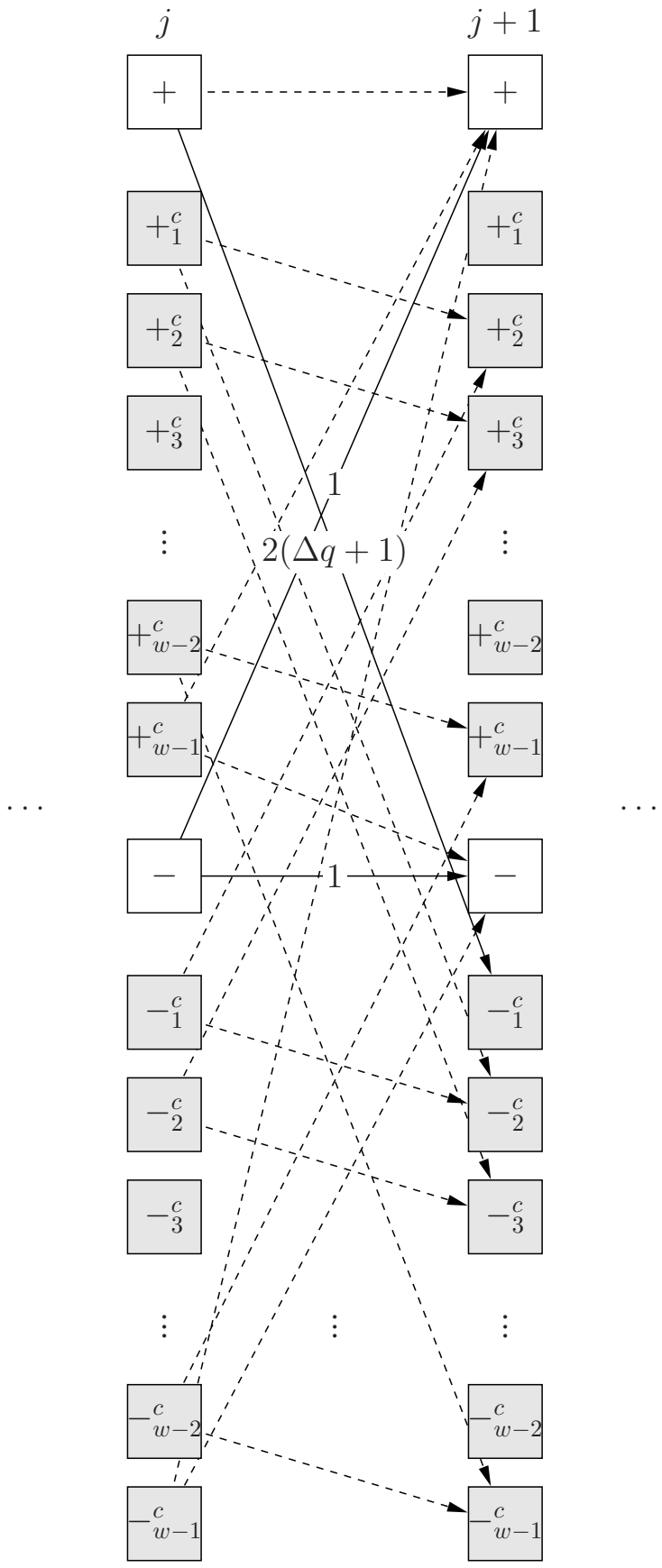


Рис. 2.11. Пример участка графа соседства и разрешенные переходы между состояниями j-й и $j+1$-й вершин

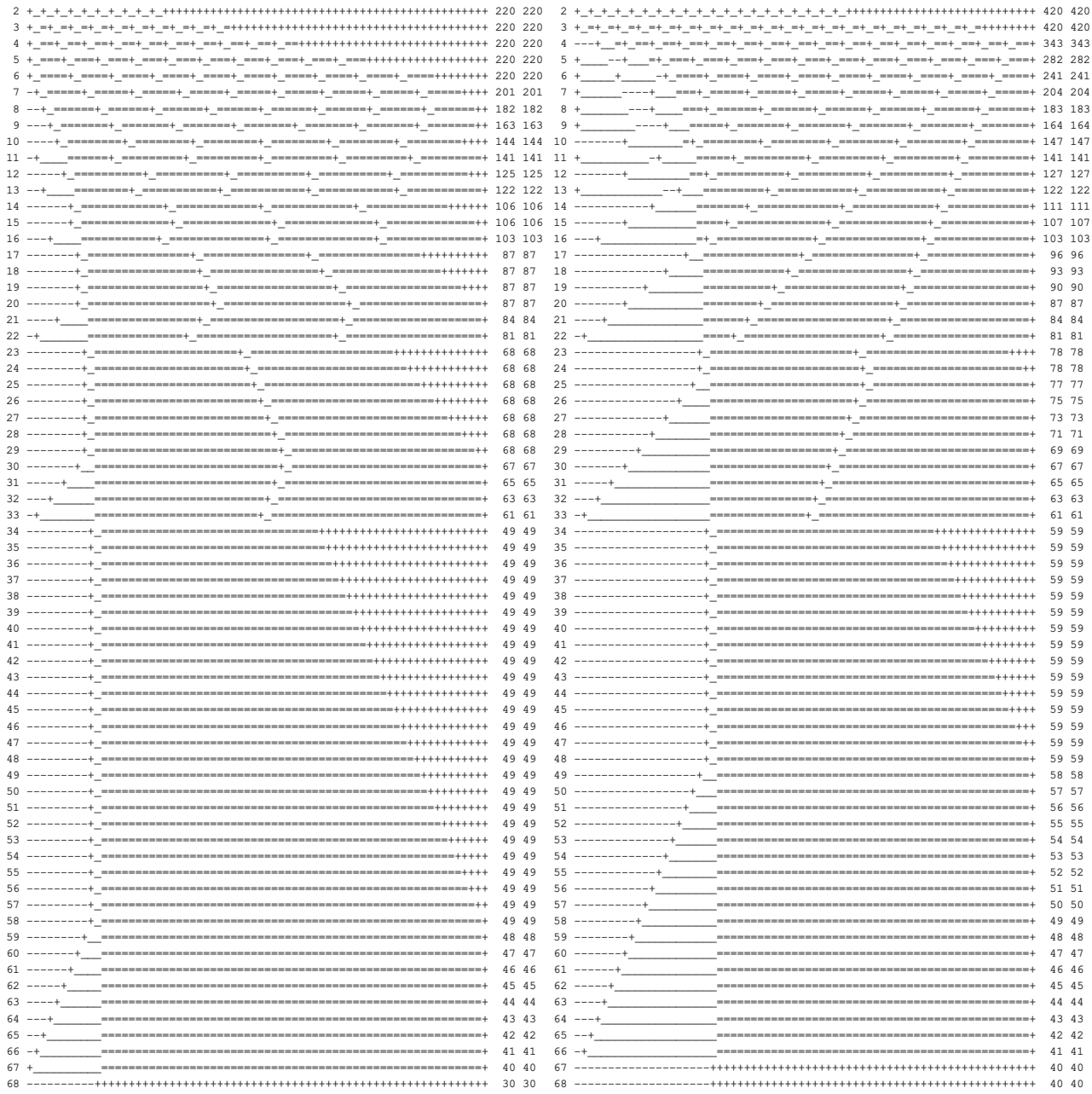


Рис. 2.12. Результат решения $(\max, +)$ задачи

2.5.2. Экспериментальная база RandomStrings

Для некоторого набора значений длин n некоторая (случайная) строка «центр» q случайно редактировалась с использованием операций вставки, удаления и замены. Полученные при таком редактировании строки составляли коллекцию строк P. Поскольку общее число операций принято намного меньшим длины строки, а посчитать точно расстояние редактирование не представляется возможным из-за вычислительной сложности, положим, что расстояние редактирования от центра q до $y \in P$ примерно равно сумме количеств ис-

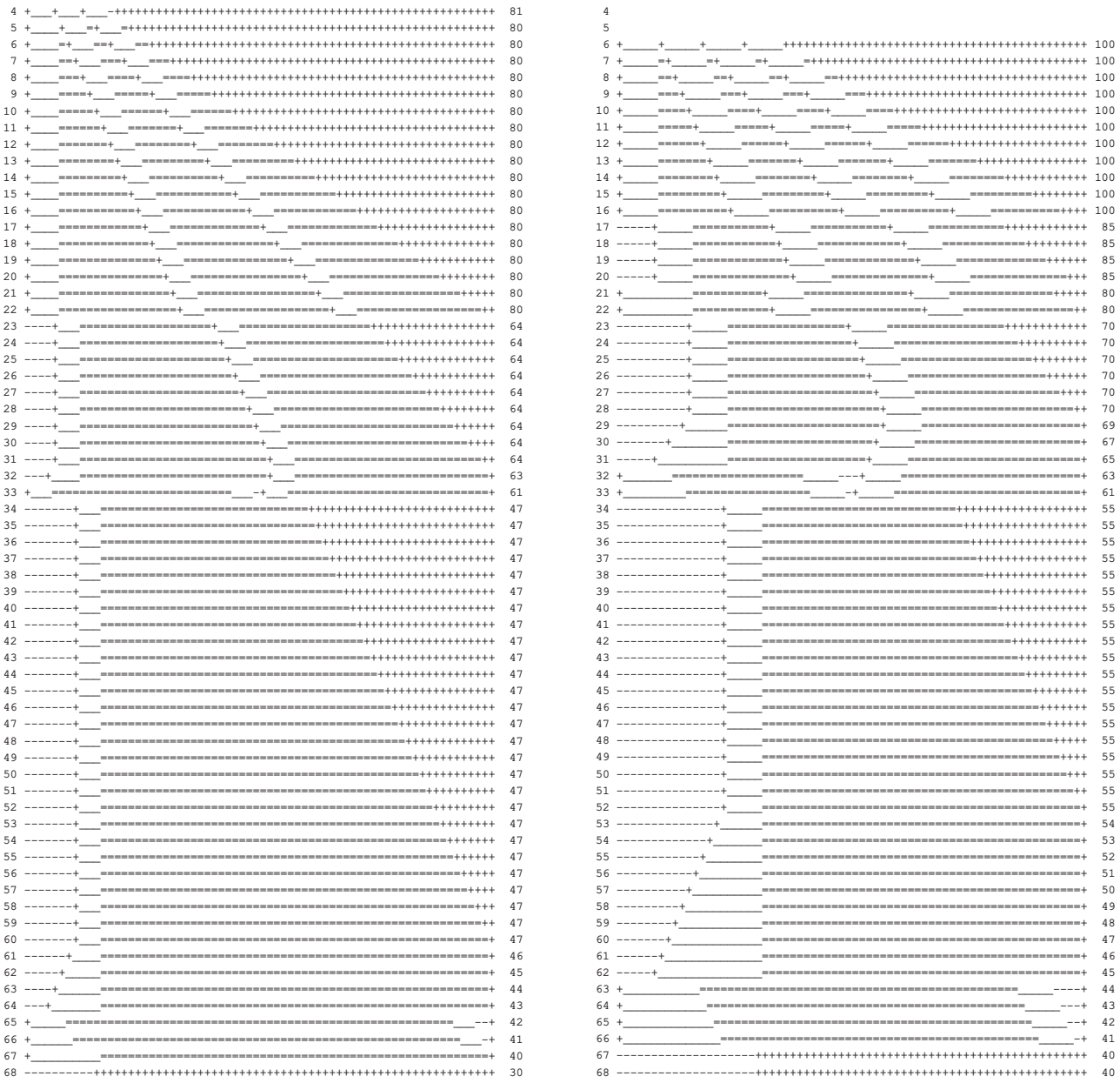
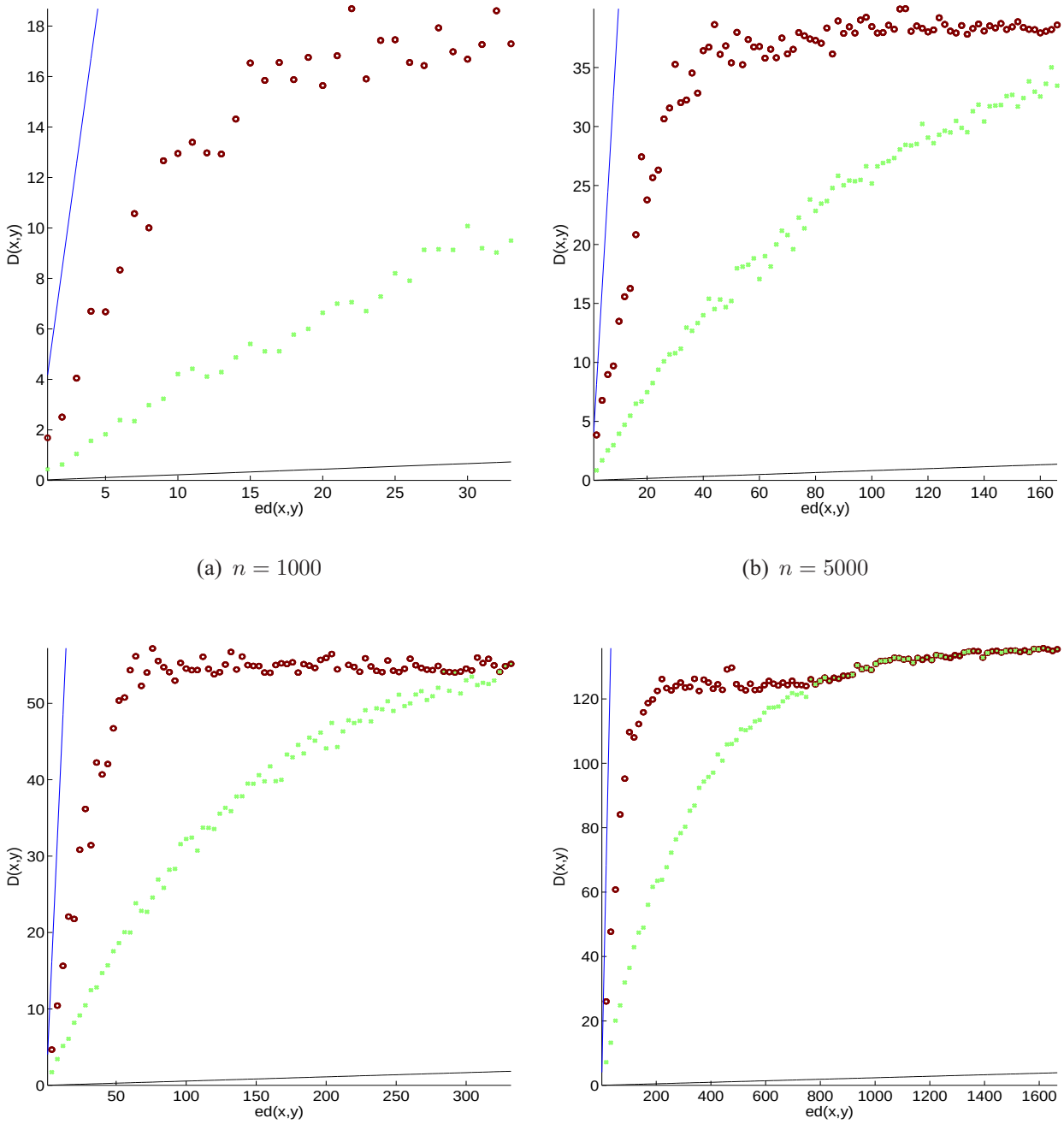


Рис. 2.13. Результат решения $(\max, +)$ задачи с учетом группировки плохих интервалов

кажений, проведенных над q для получения y. Аналогичное допущение было принято в [7]. В дальнейшем мы будет ссылаться на этот набор строк как на RandomStrings.

2.5.3. Проверка теоретических ограничений детерминированного вложения

Экспериментальное исследование того, попадает ли величина расстояния $D(x, y)$ (2.9) в ограничения (2.12) и (2.13), выполнялось для строк из базы Ran-



(a) $n = 1000$ (b) $n = 5000$

(c) $n = 10000$ (d) $n = 50000$

Рис. 2.14. Типичная зависимость расстояния $D(x, y)$ (2.9) от $ed(x, y)$

domStrings. Переписка с авторами работ [10, 11] говорит, что они не проводили подобную экспериментальную проверку своих алгоритмов. Единственной нам известной проверкой на практике алгоритма вложения (блочного) расстояния редактирования ([31, 30, 27], п. 1.3.3) в векторное пространство является [7].

На рис. 2.5.2 приведены минимальные (крестики) и максимальные (нолики) экспериментальные значения $D(x, y)$ для каждого из значений расстояния редактирования $ed(x, y)$ для двух значений $n = 5000$ и $n = 10000$. Теоретиче-

ские значения верхней (2.13) и нижней границ (2.12) на значения расстояния изображены, соответственно, сплошной и пунктирной тонкими линиями. Из рис. 2.5.2 видно, что экспериментальные данные с большим запасом соответствуют теоретическим оценкам (2.12) и (2.13).

Выполаживание графика происходит вблизи максимально возможного значения расстояния $D(x,y) = 2(w+1) - q_2 - q_1$ (что равно $19, 42, 60, 136$ для соответственно $n = 1000, 5000, 10000, 50000$), когда в каждом окне имеется $2(w-q+1)$ различных q-грамм для $q = q_1, \ldots, q_2$.

2.6. Выводы по разделу 2

1. Разработанный q-граммный детерминированный метод аппроксимации расстояния редактирования путем вложения пространства строк длины n с метрикой редактирования в пространство ℓ_1, за счет использования q-грамм переменной длины и восстановления в смежных интервалах улучшает качество аппроксимации по сравнению с известными методам вложения расстояния редактирования в пространство ℓ_1.
2. Разработанный метод анализа разработанного вложения на основе математического аппарата графов де Брейна позволил оценить точность аппроксимации расстояния редактирования, вычислительную сложность и затраты памяти. Показано, что разработанный метод вложения дает верхнюю границу интервала k_2 аппроксимации $k_2 = O(k_1\sqrt{n})$, размер используемой памяти $O(n^{5/4})$ и время построения векторов $O(n^{3/2})$ по сравнению с известными методам вложения расстояния редактирования в пространство ℓ_1.
3. Численные экспериментов показали соответствие диапазона значений манхетеннова расстояния между векторными представлениями итогового пространства $D(x,y)$ теоретически полученным ограничениям на этот диапазон.

РАЗДЕЛ 3

РАЗРАБОТКА РАСПРЕДЕЛЕННЫХ МЕТОДОВ ВЛОЖЕНИЯ РАССТОЯНИЯ РЕДАКТИРОВАНИЯ И ЭФФЕКТИВНОГО ПОИСКА ПРИБЛИЖЕННЫХ БЛИЖАЙШИХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

На основе разработанного в разделе 2 метода детерминированного вложения пространства последовательностей с классической метрикой редактирования в пространство ℓ_1 в данном разделе разработана локально-чувствительная функция для расстояния редактирования (п. 3.1.2), а также разработаны два рандомизированных варианта вложения (п. 3.3 и п. 3.2). Они продуцируют распределенные представления, которые позволяют сократить время поиска близких последовательностей и требуемую память. На основе полученных распределенных представлений разработаны методы поиска приближенных ближайших последовательностей. Получены аналитические оценки эффективности предложенных методов представления (п. 3.1.2) и поиска (п. 3.3.1). Показана связь полученного в пункте 3.3 распределенного представления с предложенным распределенным прореживающим представлением последовательностей (п. 3.4.3).

3.1. Рандомизированные методы формирования векторных представлений для поиска ближайших строк

Применение детерминированной версии (раздел 2) вложения пространства (Σ^n, ed) в ℓ_1 затруднено по причине больших размерностей получаемых векторов. Кроме того, потенциально большие размеры баз последовательностей затрудняют поиск в них ближайших соседей. Разработаем рандомизированные версии детерминированного метода, которые могут использоваться для практических приложений в задаче поиска ближайшего соседа.

3.1.1. Вероятностные варианты лемм раздела 2

Доказательство следующих следствий – вероятностных вариантов леммы 2.9 и следствия 2.3 – тривиально, и получается при случайном выборе значения i из диапазона $1, \ldots, n-w+1$.

Следствие 3.1. *Если* $ed(x,y) \leqslant k_1$, $q_1 < q_2 \leqslant w \leqslant \frac{n+1}{k_1+1}$, *то*

$$Prob[d^{\Sigma}_{w,q_1,q_2}(x[i,i+w-1], y[i,i+w-1]) \leqslant 2k_1(\Delta q+1)] > 1 - \frac{k_1 w}{n-w+1}.$$

Доказательство. Общее число интервалов в последовательности $n-w+1$. Утверждение следствия получается, поделив правую часть выражения из леммы 2.9 на общее число интервалов. □

Следствие 3.2. *Если* $ed(x,y) > k_2$, *то*

$$Prob[d^{\Sigma}_{w,q_1,q_2}(x[i,i+w-1], y[i,i+w-1]) \geqslant Q] > \frac{t(\frac{k_2}{2(\Delta q+1)} - 2)}{n-w+1}.$$

Доказательство. Утверждение следствия также получается, поделив правую часть выражения из следствия 2.3 на общее число интервалов. □

3.1.2. Локально-чувствительная функция для расстояния редактирования

Предложим новую локально-чувствительную хеш-функцию для расстояния редактирования на основе разработанного метода детерминированного вложения из раздела 2 и схемы хеширования для ℓ_1 на основе 1-стабильного распределения (п. 1.5.2, стр. 36) и определения (1.18).

Возьмем случайный вектор $v^i_{w,q_1,q_2}(x)$, получаемый случайным и независимым выбором окна шириной w в строке x (см. следствия 3.2 и 3.1). Сгенерируем свой вектор ϕ_i, элементы которого взяты из распределения Коши. Хеш-функции $h'_i(x)$, – элементы вектора $h'(x) = (h'_1(x), h'_2(x), \ldots, h'_1(x))$, – определим как

$$h'_i(x) = \left\lfloor \frac{(v^i_{w,q_1,q_2}(x), \phi_i) + b_i}{r} \right\rfloor, \tag{3.1}$$

где b_i, r – параметры из определения (1.18).

Получается трехступенчатая процедура хеширования последовательностей: получаем прореженный q-граммный вектор путем семплирования случайным окном шириной w, умножаем скалярно на случайный вектор, распределенный по Коши, и наконец применяем к полученому произведению отображение (1.18).

Докажем, что данная функция является локально-чувствительной и может использоваться в схеме LSH (п. 1.5.1 и 3.3.1).

Для определения значений p_I, p_{II} (см. (1.13)) необходимо выяснить распределение величин $h'_i(x)$ и $h'_i(y)$. Пусть $p(d|k)$ – вероятность того, что расстояние $d^{\Sigma}_{w,q_1,q_2}(x,y) = d$ при условии $ed(x,y) = k$, а $Prob[h'(x) = h'(y)|c] = p(c)$, где c – целочисленное значение $d^{\Sigma}_{w,q_1,q_2}(x,y)$.

Обозначим

$$d_1 = 2k_1(\Delta q + 1), \quad d_2 = Q, \quad p_1 = 1 - \frac{k_1 w}{n - w + 1}, \quad p_2 = \frac{t(\frac{k_2}{2(\Delta q+1)} - 2)}{n - w + 1}.$$

Тогда для семейства новых хеш-функций (3.1) получаем

$$\begin{aligned} &Prob[h'_i(x) = h'_i(y)|ed(x,y) \leqslant k_1] \geqslant \\ &\geqslant \sum_{c \leqslant d_1} p(c|ed(x,y) \leqslant k_1)p(c) \geqslant p(d_1) \sum_{c \leqslant d_1} p(c|ed(x,y) \leqslant k_1) = \\ &= p(d_1)p(c \leqslant d_1|ed(x,y) \leqslant k_1) \geqslant p(d_1)p_1 = p_I, \end{aligned} \tag{3.2}$$

$$\begin{aligned} &Prob[h'_i(x) = h'_i(y)|ed(x,y) > k_2] \leqslant \\ &\leqslant p(d_2) \sum_{c \geqslant d_2} p(c|ed(x,y) > k_2) + \sum_{c < d_2} p(c|ed(x,y) > k_2)p(c) \leqslant \\ &\leqslant p(d_2)p(c \geqslant d_2|ed(x,y) > k_2) + \sum_{c < d_2} p(c|ed(x,y) > k_2)p(0) = \\ &= p(d_2)p_2 + (1 - p_2)p(0) \leqslant 1 - p_2(1 - p(d_2)) = p_{II}. \end{aligned} \tag{3.3}$$

Предложенная локально-чувствительная функция генерирует распределенное представление последовательности – вектор, элементы которого есть значения независимо и равновероятно выбранных функций вида (3.1), зафиксированных и использующихся для получения распределенного представления всех последовательностей.

3.2. Решение задачи поиска приближенных ближайших последовательностей с помощью рандомизации детерминированного метода

Для решения задачи поиска приближенных ближайших последовательностей разработаем еще один метод, использующий схему из [71], но отличающийся использованием рандомизации метода детерминированного вложения из раздела 2.

Пусть зафиксирована позиция i из диапазона $1, \ldots, n-w+1$ и соответствующий ему интервал $[i, i+w-1]$. Если позиция i выбрана случайно и равновероятно (с возвращением) среди всех $n-w+1$ возможных значений, то выражение (2.9) является матожиданием величины $\frac{d^{\Sigma}_{w,q_1,q_2}(x[i,i+w-1],y[i,i+w-1])}{\Delta q+1}$. Определим вектор $v_i(x)$ как конкатенацию q-граммных векторов $v_q(x)$ этого интервала для значений $q = q_1, \ldots, q_2$.

Выберем случайно, независимо и равновероятно (с возвращением) U значений $i_f,\ f = 1, \ldots, U$ (обозначим их множество I_U) и конкатенируем соответствующие им $v_{i_f}(x)$ в один вектор $\tilde{v}(x)$. Конкатенацию всех $x[i_f, i_f+w-1]$ обозначим $x(I_U)$. Обозначим $\tilde{D}(x,y) = \frac{\|\tilde{v}(x)-\tilde{v}(y)\|_{l_1}}{(\Delta q+1)U}$.

Пусть строки $p_0, p_a, p_b \in P$ такие, что $ed(p_0,p_a) \leqslant k_1$, $ed(p_0,p_b) > k_2$.Тогда

$$Prob[\tilde{D}(p_0,p_a) > d_1+\varepsilon] < e^{-2U\varepsilon^2}, \quad Prob[\tilde{D}(p_0,p_b) < d_2-\varepsilon] < e^{-2U\varepsilon^2}.$$

Для доказательства используем идею леммы 2 из [71]. Пусть матожидание $d^{\Sigma}(p_0,p_a)$ равно $M(d^{\Sigma}(p_0,p_a))$. Норму $\|\tilde{v}(p_0)-\tilde{v}(p_a)\|_{l_1}$ можно рассматривать как сумму независимых и равновероятно распределенных величин

$$\|v_{i_u}(p_0)-v_{i_u}(p_a)\|_{l_1}, u = 1, \ldots, U.$$

Для любых строк $p, p' \in \Sigma^n$ значения нормы разницы векторов $v_{i_u}(p)$ и $v_{i_u}(p')$ ограничены:

$$0 \leqslant \|v_{i_u}(p)-v_{i_u}(p')\|_{l_1} \leqslant \sum_{q=q_1}^{q_2} \frac{2(w-q+1)}{\Delta q+1} = (2w+1-2q_m).$$

Применяя неравенство Хеффдинга [55] для ограниченных случайных величин, получаем:

$$\begin{aligned} P[\tilde{D}_U(p_0,p_a) > d_1+\varepsilon] &= P[\|\tilde{v}(p_0)-\tilde{v}(p_a)\|_{l_1} > d_1+\varepsilon] \\ &\leqslant P[\|\tilde{v}(p_0)-\tilde{v}(p_a)\|_{l_1} > M(d^{\Sigma}(p_0,p_a))+\varepsilon] \\ &\leqslant e^{-\frac{2U\varepsilon^2}{(2w+1-q_m)}} < e^{-\frac{2U\varepsilon^2}{w}}. \end{aligned} \tag{3.4}$$

Аналогично доказывается, что $P[\tilde{D}_U(p_0,p_a) < d_2-\varepsilon] < e^{-\frac{2U\varepsilon^2}{w}}$.

Создадим структуру S, состоящую из n структур S_k, $k=1,\ldots,n$, по одной на каждое из возможных значений расстояний редактирования между строками длины n. Каждая из S_k состоит из M структур $F_1,\ldots,F_M$. Каждая структура $F_m,\ m=1,\ldots,M$ содержит U значений позиций $i_{m1},\ldots,i_{mU}$ (указывающих на начало соответствующих интервалов $I_{i_{mu}}=[i_{mu}, i_{mu}+w-1]$), выбранных случайно и равновероятно, с возвращением, из диапазона $[1,\ldots,n-w+1]$, и таблицу Λ_m, содержащую $|\Sigma|^n$ ячеек (по числу возможных строк длиной n). Содержимое интервала $I_{i_{mu}}$ в структуре F_m для строки x обозначим как $x[I_{i_{mu}}]$. Конкатенацию всех $v_{w,q_1,q_2}(x[I_{i_{mu}}]), u=1,\ldots,U$ обозначим $v_m(x)$. Ячейка таблицы Λ_m, соответствующая $z \in |\Sigma|^{wU}$, содержит строку $p \in P$, если $\tilde{D}(p,z) \leqslant d_1$, если такая p существует, иначе ячейка пуста. Структура S при простейшей реализации будет занимать объем памяти порядка $O(nM(U+\log P|\Sigma|^{wU})+nP)$ и может быть построена за время $O(nMP(wU+|\Sigma|^{wU}))$.

Объем таблиц Λ_m можно уменьшить до $|\Sigma|^{wU}$, индексируя ее строками, получающимися конкатенацией U интервалов. Далее, для принятия решения о внесении строки в ячейку нам необходимы q-спектры подстрок $z[1+(u-1)w, uw]$ при $q=q_1,\ldots,q_2$. Поэтому можно сэкономить место под хранение таблиц Λ_m, объединив ячейки с совпадающими q-спектрами интервалов $z[1+$

$(u-1)w, uw]$ для всех $q = q_1, \ldots, q_2$. В этом случае можно обозначить

$$\tilde{D}_m(p;z) = \sum_{u=1}^{U} d^{\Sigma}_{w,q_1,q_2}(p[I_{i_{mu}}], z[1+(u-1)w, uw]) = \sum_{u=1}^{U} \|\tilde{v}(p[I_{i_{mu}}]) - \tilde{v}(z[1+(u-1)w, uw])\|_{l_1}. \quad (3.5)$$

И ячейка таблицы Λ_m, соответствующая $z \in |\Sigma|^{wU}$, будет содержать строку $p \in P$, если $\tilde{D}_m(p;z) \leqslant d_1$.

Для строки-запроса $p_0 \in \Sigma^n$ будем говорить, что *структура F_m ошибается на P*, если существует $p_1 \in P$, такая, что $ed(p_0, p_1) \leqslant k_1$ (или строка p_2 с $ed(p_0, p_2) \geqslant k_2$) и $\tilde{D}_m(p_0, p_1) > d_1$ ($\tilde{D}_m(p_0, p_2) < d_2$). Если существует k такое, что более, чем $\mu M / \log n$ структур F_m в S_k ошибаются на P, то будем говорить, что *структура S ошибается на p_0*. Будем говорить, что *структура S ошибается*, если существует $p_0 \in \Sigma^n$, такое, что S ошибается на p_0.

Для любого $\delta > 0$, если положить $M = (n\log_2(\Sigma) + \log n - \log\delta)\ln n/\mu$ и $F = 0.5\varepsilon^{-2}\ln(2eP\ln n/\mu)$, то для любого запроса p_0 вероятность, что структура S ошибается на p_0, есть максимум $\delta 2^{-n}$. Для доказательства воспользуемся схемой доказательства из [71] (теорема 4). Для любого i, $F_j \in S_k$ и $p_a \in P$ вероятность, что $\tilde{D}(p_0, p_a)$ будет вне нужного интервала, есть максимум $e^{-\frac{2T\varepsilon^2}{w}} = \frac{\mu}{2eP\ln n}$.

Суммируя по P строкам в базе, получаем, что вероятность, что F_j не работает, есть максимум $\frac{\mu}{2e\ln n}$. Тогда среднее число структур F_j, которые не работают, есть $\frac{M\mu}{2e\ln n}$. Пусть $\xi_j = 1$, если F_j не работает, и $\xi_j = 0$ иначе. Количество не работающих F_j будет $\sum_{j=1}^{M}\xi_j$. Найдем вероятность, что не работает более чем $\frac{M\mu}{\ln n}$ структур. По одной из форм неравенства Чернова

$$Prob[\sum_{j=1}^{M}\xi_j > \frac{M\mu}{2e\ln n} + \frac{M\mu}{2e\ln n}] = Prob[\sum_{j=1}^{M}\xi_j > M(\sum_{j=1}^{M}\xi_j) + \frac{M\mu}{2e\ln n}] \leqslant (\frac{e^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}})^{\frac{M\mu}{2e\ln n}} < 2^{-\frac{M\mu}{2e\ln n}} = \frac{\delta}{n2^n}. \quad (3.6)$$

Просуммировав по всем структурам $1, \ldots, n$, завершаем доказательство. Таким образом, S на всей P ошибается с вероятностью δ.

Процедура поиска. Допустим, S не ошибается (это происходит с вероятностью $1-\delta$). Используем бинарный поиск для нахождения ближайшего соседа. Выберем произвольно начальное k и выберем случайно и равновероятно одну из F_m в S_k. Подсчитаем значение $p_0(I_{F_m})$ и проверим соответствующую ячейку. Если ячейка не пуста, переходим и проверяем аналогично S_{k-1}, иначе переходим к S_{k+1}. По лемме 6 из [71] вероятность, что выбранное F_i ошибается на p_0, меньше или равна μ.

Если все S_k не ошибаются на p_0, тогда расстояние от p', возвращенной алгоритмом поиска, отличается не более, чем в $1+\varepsilon$ раз от расстояния от p_0 до настоящего ближашего соседа. Для доказательства воспользуемся идеей доказательства леммы 7 из [71]. Пусть $k_{min} = min_{p \in P} ed(p, p_0)$. Если $k < k_{min}/(1+\varepsilon)$, то все S_k не сработают на p_0. С другой стороны, все $S_k, k \geqslant k_{min}$ работают на p_0. Следовательно, процедура бинарного поиска завершится на таком k, что $k_{min}/(1+\varepsilon) \leqslant k \leqslant k_{min}$. При этом возвращенная строка p' такая, что $\tilde{D}(p_0, p') \leqslant d_1$. Так как для любой $p \in P,\ ed(p, p_0) > k(1+\varepsilon)$, $\tilde{D}(p_0, p) \geqslant d_2(k)$. Поскольку $d_2 > d_1$, то $\tilde{D}(p_0, p) > d_1$, и $ed(p, p_0) \leqslant (1+\varepsilon)k < (1+\varepsilon)k_{min}$. Согласно теореме 8 [71], получаем, что приведенная процедура поиска находит с высокой вероятностью $(1+\varepsilon)$-ближайшего соседа, для любого $\varepsilon > 0$: если S не ошибается, то для любого запроса p_0 алгоритм бинарного поиска находит $(1+\varepsilon)$-приближенного соседа с вероятностью $1-\mu$ за время $O(\varepsilon^{-2} n(\log|P| + \log\log n - \log\mu)\log n)$.

3.3. Решение задачи поиска приближенных ближайших последовательностей с помощью разработанной локально-чувствительной функции

Для решения задачи ε-БС была применена общая схема решения задачи (ε, k)-PLEB (п. 1.5.1) и разработанная локально-чувствительная функция (п. 3.1.2) для расстояния редактирования.

Предварительно найдем асимптотические значения (1.19) при $r \gg c$ и

при $r \ll c$:

1. Для $r \ll c$ имеем $\tan^{-1}(x) \simeq x - \frac{x^3}{3} + O(x^3)$, поэтому

$$p(c) = \frac{1}{\pi}(2\tan^{-1}(\frac{r}{c}) - \frac{c}{r}\ln(1 + (\frac{r}{c})^2)) < \frac{1}{\pi}(2\frac{r}{c} - \ln e^{\frac{r}{c}}) = \frac{1}{\pi}\frac{r}{c}. \quad (3.7)$$

2. Для $r \gg c$, поскольку $\tan(x) \to \frac{\pi}{2}$ при $x \to \infty$, имеем

$$p(c) = \frac{1}{\pi}(2\tan^{-1}\frac{r}{c} - \frac{c}{r}\ln(1 + (\frac{r}{c})^2)) < 1 - \frac{2}{\pi}\frac{c}{r}\ln\frac{r}{c}. \quad (3.8)$$

Для семейства новых хеш-функций (3.1) получено в (3.3) и (3.2), что

$$Prob[h'_i(x) = h'_i(y) | ed(x,y) \leqslant k_1] \geqslant p(d_1)p_1 = p_I, \quad (3.9)$$

$$Prob[h'_i(x) = h'_i(y) | ed(x,y) > k_2] \leqslant 1 - p_2(1 - p(d_2)) = p_{II}. \quad (3.10)$$

Чтобы такое семейство было локально-чувствительным (см. (1.14)), должно выполняться

$$1 - p_2(1 - p(d_2)) < p(d_1)p_1. \quad (3.11)$$

В свою очередь, для его выполнения необходимо, чтобы

$$k_2 > 2(\Delta q + 1)(2 + \frac{n - w + 1}{t(1 - p(d_2))}(1 - p(d_1) + \frac{wk_1p(d_1)}{n - w + 1}). \quad (3.12)$$

Из этого выражения найдем асимптотическое поведение k_2 и, следовательно, точность аппроксимации расстояния редактирования в зависимости от n. Для этого положим $w = n^\gamma,\ 0 < \gamma < 1$, и $r = n^\mu,\ \mu > 0$. Из условия 2.3 леммы 2.3 $\Delta q = \Theta(\sqrt{w})$ поэтому

$$d_1 = 2k_1(\Delta q + 1) = \Theta(k_1 n^{\gamma/2}), \quad d_2 = Q = \Theta(n^\gamma), \quad t = w - q_2 - \Delta q = \Theta(n^\gamma).$$

При $n \to \infty$, используя (1.19), (3.7), (3.8), найдем оценки $1 - p(d_1)$ и $1 - p(d_2)$ в зависимости от соотношения параметров $\mu,\ \gamma$:

1. Если $\gamma = \mu$, то $r = \omega(d_1),\ r = \Theta(d_2)$, то

$$1 - p(d_2) = const, \quad 1 - p(d_1) > \frac{2}{\pi}\frac{d_1}{r}\ln(\frac{r}{d_1}).$$

2. Если $\mu > \gamma$, то $r = \omega(d_1)$ и $r = \omega(d_2)$, то

$$1 - p(d_1) > \frac{2}{\pi}\frac{d_1}{r}\ln(\frac{r}{d_1}), \quad 1 - p(d_2) > \frac{2}{\pi}\frac{d_2}{r}\ln(\frac{r}{d_2}).$$

3. Если $\gamma/2 < \mu < \gamma$, то $r = \omega(d_1)$ и $r = o(d_2)$, то

$$1 - p(d_2) > 1 - \frac{1}{\pi}(\frac{r}{d_2}), \quad 1 - p(d_1) > \frac{2}{\pi}(\frac{d_1}{r}\ln(\frac{r}{d_1})).$$

4. Если $\mu = \gamma/2$, то $r = \omega(d_1)$ и $r = o(d_2)$, то

$$1 - p(d_2) > 1 - \frac{1}{\pi}(\frac{r}{d_2}), \quad 1 - p(d_1) = const.$$

Для удовлетворения (3.11), подставим полученные выражения в (3.12) и найдем выражение для k_2 для каждого случая

1. Если $\gamma = \mu$, то $k_2 = \Omega(k_1(n^{1-\gamma}\ln n + n^{\gamma/2})$. При $\gamma > 2/3$, $n^{\gamma/2}$ доминирует $n^{1-\gamma}\ln n$ и k_2 будет иметь асимптотику $\Omega(n^{\gamma/2}), \gamma > 2/3$. При $\gamma \leqslant 2/3$, $n^{1-\gamma}\ln n$ доминирует $n^{\gamma/2}$ и k_2 имеет асимптотику $\Omega(n^{1-\gamma}\ln n), \gamma \leqslant 2/3$. Отсюда, оптимальным значением будет $\gamma = 2/3$, и $k_2 = \Omega(k_1 n^{1/3}\ln n)$.
2. Если $\mu > \gamma$, то $k_2 = \Omega(n^{\gamma/2} + k_1[n^{1-\gamma} + n^{\mu}/\ln n]))$. Поскольку $\mu > \gamma$, то $\frac{n^{\mu}}{\ln n} = \Omega(n^{\gamma})$ и k_2 в этом случае растет быстрее, чем в варианте с $\mu = \gamma$.
3. Если $\gamma/2 < \mu < \gamma$, то $k_2 = \Omega(k_1(n^{1-\mu}(\mu - \gamma/2)\ln n + n^{\gamma/2})$. Поскольку $\mu < \gamma$, то $n^{1-\mu}\ln n = \Omega(n^{1-\gamma}\ln n)$, что является более плохой оценкой, чем оценка в варианте с $\mu = \gamma$.
4. Если $\mu = \gamma/2$, то $k_2 = \Omega(n)$, что также хуже, чем вариант $\mu = \gamma$.

Итак, оптимальным значением γ будет $\gamma = 2/3$, при котором

$$k_2^* = \Omega(k_1 n^{1/3}\ln n).$$

3.3.1. Анализ поиска приближенных ближайших последовательностей на основе предложенной локально-чувствительной функции

Положим k_2 из (3.12) равным

$$k_2 = 2(\Delta q + 1)(2 + z\frac{n - w + 1}{t(1 - p(d_2))}(1 - p(d_1) + \frac{wk_1 p(d_1)}{n - w + 1})$$

для $\mathbb{R} \ni z > 1$. Это повлияет на значение и асимптотику p_2, поскольку его значение зависит от k_2. Оценив в таком случае величину $\rho = \frac{\ln(p_1 p(d_1))}{\ln(1-p_2(k_2)(1-p(d_2)))} = \frac{\ln(1-A)}{\ln(1-zA)}$, где $A = 1 - p_1 p(d_1)$, аналогично оценке ρ для случая хэммингова расстояния [60, 46] получаем $\rho = O(\frac{1}{1+z})$ при условии $\ln |P| > p_1 p(d_1)$.

Таким образом, согласно теореме Индыка-Мотвани (п. 1.5.2, стр. 36), метод приближенного поиска ближайшей строки, построенный на предложенном в подразделе 3.1 варианте LSH на 1-стабильном распределении, возвратит с вероятностью, большей $1/2$, строку из P, которая принадлежит шару $S(q, O(zk_1 n^{1/3} \ln n))$, затратив на это порядка $O(|P|^{1/(1+z)})$ операций.

Поскольку схема имеет константную вероятность ошибки, повторив описанную процедуру LSH паралельно и независимо порядка $O(\ln \frac{1}{\alpha})$ раз, мы можем добиться вероятности успеха хотя бы в одном запуске процедуры не менее $1 - \alpha$ для любого заданого уровня ошибки $\alpha < 1$.

3.4. Распределенные представления последовательностей

Покажем связь методов распределенного представления символьных последовательностей на основе рандомизированных вложений, разработанных в п. 3.1 и п. 3.3, и разработанных на основе нейросетевых подходов (п. 1.2 и п. 3.4.1-3.4.3).

3.4.1. Метод распределенного представления последовательностей без учета порядка элементов

Поскольку символы алфавита Σ различны, будет представлять их (и/или q-граммы) случайно и независимо сгенерированными и затем фиксированными векторами (п. 1.2). Каждой q-грамме t ($q \geqslant 1$) поставим в соответствие случайный независимый вектор $v(t)$. Вектор всей последовательности создается путем суммирования или дизъюнкции векторов q-грамм. Таким образом, векторы последовательностей, состоящих из большого количества одинаковых элементов, получат близкие представления.

То есть, вектор $v(x)$ для строки $x = x_1, \ldots, x_n$ определяется как $v(x) =$

$\sum_{i=1,\ldots,n} r(x_i)$, где $r(x)$ есть случайный вектор, соответствующий элементу x. Выражение для $v(x)$ можно переписать как

$$v(x) = \sum_{i=1,\ldots,n} r(x_i) = \sum_{\sigma \in \Sigma} \sum_{i=1,\ldots,n} I[x_i = \sigma] = \sum_{\sigma \in \Sigma} r(\sigma) n_x(\sigma), \tag{3.13}$$

где $n_x(\sigma)$ – количество вхождений символа σ в s.

Выражение (3.13) эквивалетно проекции вектора-представления типа «bag-of-items» (п. 1.3.2), с элементами векторов – q-граммами или символами, – $n_\sigma^T(\sigma) = (n_1(\sigma), \ldots, n_{|\Sigma|}(\sigma))$ путем умножения его на случайную матрицу $(d \times |\Sigma|)$ со столбцами $r(\sigma)$. Таким образом, данное представление может рассматриваться как отображение из (например, VSM, п. 1.3.2) пространства размерности $|\Sigma|$ в $\mathbb{R}^d$, где d может быть сделано существенно меньше, чем число всех возможных элементов $|\Sigma|$.

Если оба пространства евклидовы, то, применяя лемму Джонсона-Линденштрауса (п. 1.4.1, стр. 27) и выбирая $r(x)$ из нормального распределения (или из определенного бинарного или тернарного распределения [1]), можно получить, что для некоторого желаемого искажения $0 < \varepsilon < 1$, возможно уменьшить размерность, оставляя $\|v(x) - v(y)\|_{l_1}$ в интервале $(1 \pm \varepsilon) \|v_q(x) - v_q(y)\|_{l_1}$ с большой вероятностью.

Таким образом, используя лемму Укконена (п. 1.3.3, стр. 29), для двух строк с расстоянием редактирования меньше k эвклидово расстояния между соответствующими векторами не превышает $(1 + \varepsilon)qk$. Это дает верхнюю границу на расстояние между векторами и может использоваться с целью фильтрации строк или документов по расстоянию редактирования. При этом распределенные представления в виде векторов малой размерности используются вместо разреженных q-граммных векторов большой размерности.

3.4.2. Методы распределенного представления последовательностей с учетом порядка элементов

В следующих двух предложенных методах информация о позиции элемента в последовательности объединяется с представлением самого элемента

путем модификации представления элемента позиционно-зависимым способом. Векторы могут быть бинарными, при этом суммирование заменяется на дизъюнкцию.

3.4.2.1. Позиционное связывание с помощью перестановок

Для сдвигового представления [109] распределенные представления элементов последовательности модифицируются путем циклического сдвига (или, в общем случае, путем случайной перестановки) на число элементов вектора, зависящее от позиции элемента последовательности, а представление всей последовательности x формируется дизъюнкцией распределенных представлений ее элементов – q-грамм:

$$v(x) = \bigvee_{i=1}^{|s|} v(x[i, i+q-1]) >> i,$$

где $\vee$ – побитовая дизъюнкция, а $X >> y$ обозначает вектор X, сдвинутый циклически на y элементов. В общем случае перестановок

$$v(x) = \bigvee_{i=1}^{|x|} \pi_i(v(x[i, i+q-1])),$$

где π_i – случайные и независимо выбранные перестановки.

Рассмотрим, например, строки 'abc', 'abd', 'cba'. Каждый символ представим случайным вектором: $v(a), v(b), v(c)$ и т.д. Тогда вектор последовательностей формируются следующим образом:

$$v(abc) = (v(a) >> 1) \vee (v(b) >> 2) \vee (v(b) >> 3),$$
$$v(bca) = (v(b) >> 1) \vee (v(c) >> 2) \vee (v(a) >> 3).$$

Пересечение полученных векторов для строк, не содержащих одинаковых элементов в одинаковых позициях, будет в среднем пренебрежимо мало, при условии, что случайные векторы достаточно разрежены, а строки не слишком длинные. Недостатком приведенной процедуры является то, что она не учитывает информацию об идентичных символах в различным позициях.

3.4.2.2. Позиционное связывание с помощью представлений позиций

В данном методе дополнительно к векторам подстрок генерируются векторы для позиций. Для того, чтобы представить подстроку в конкретной позиции, вектор подстроки и вектор позиции *связываются*:

$$v(x) = \sum_{i=1}^{|x|} Z(v(x[i, i+q+1]), v(i)),$$

где Z – некоторый оператор связывания.

В бинарном случае связывание осуществляется с помощью побитовой конъюнкции и называется прореживанием: $Z = \bigwedge$ и соответствует связыванию поэлементной конъюнкцией (п. 1.2).

Таким образом, в вектор символа – элемента последовательности (например, символа или q-граммы) привносится («впечатывается») информация о его позиции в последовательности. То есть в распределенном представлении теперь есть информация и о символе, и о его позиции и оно имеет значительное перекрытие с вектором и символа, и позиции.

В отличии от сдвигового метода (п. 3.4.2.1), данный вид связывания учитывает сходство идентичных элементов в разных позициях. Распределенное представление позиций может быть коррелированным для близких порядковых номеров. Чтобы сформировать распределенное представление всей последовательности, векторы связывания $v(x)$ символ-позиция объединяются побитовой дизъюнкцией или суммированием.

3.4.3. Прореживающее связывание и его связь с разработанной локально-чувствительной функцией

Установим аналогию между прореживающим распределенным представлением и подходами к аппроксимации расстояния редактирования.

Формирование i-го элемента выходного вектора $v(x)$ размерности d при прореживающем представлении (п. 3.4.2.2) можно записать таким образом:

$$v_i(x) = \sum_{k=1,\dots,n} c_{ki} r_{ix[k]} = \sum_{k=1,\dots,n} c_{ki} \sum_{\sigma \in \Sigma} r_{i\sigma}[x_k = \sigma] = \sum_{\sigma \in \Sigma} \sum_{k=1,\dots,n} r_{i\sigma} l_{\sigma k} c_{ki}. \quad (3.14)$$

Рассмотрим часть произведения $X = L(x) \cdot C$ (рис. 3.1). Если бы матрица C содержала в своих элементах только единицы, то это произведение давало бы столбцы векторных представлений строки x (таких, как, например, q-граммные). Но, поскольку C содержит и нули, в X учитываются подстроки не во всех позициях. Таким образом, столбцы результирующей матрицы X можно считать «неполными» векторными представлениями (q-граммными представлениями, если Σ есть множество всех q-грамм) входной строки x. Столбцы матрицы C играют роль бинарных масок, отмечающих позиции в s, откуда символы суммируются для формирования конкретного q-граммного вектора. Например, j-ый столбец C маскирует s для получения вектора с i-м элементом, равным $\sum_{k=1,\ldots,n} l_k c_{ki}$.

Таким образом, в отличии от обычных q-граммных представлений, которые пренебрегают информацией о порядке q-грамм в последовательности, здесь ее возможно учесть с помощью векторов позиций.

Выше было отмечено, что векторы позиций (строки матрицы C) могут формироваться так, чтобы векторы соседних позиций отличались на малое расстояние Хэмминга и чтобы это расстояние увеличивалось с увеличением расстояния между позициями. Некоторые методы построения таких векторов для порядковых чисел предложены в [139].

Разные столбцы играют роль независимых сэмплеров, как и в методе сублинейной аппроксимации расстояния редактирования в работе [11] (п. 1.4.2, стр. 30).

Рассмотрим умножение X на матрицу R. Каждый i-й элемент выходного вектора $v(x)$ является скалярным поризведением вектора r_i на соответствующий «прореженный» q-граммный вектор. Такая операция (проекция на случайное направление) уже присутствовала в описании аналогии между неупорядоченным распределенным представлением строк с помощью присваивания случайных векторов их подстрокам (q-граммам) (п. 3.4.1). Разница в том, что там каждый из q-граммных векторов проецировался на все случайные направления, тогда как тут различные q-граммные векторы проецируются на соб-

$$K \times K \left\{ \begin{pmatrix} \mathbf{1} & 0 & 3 & \dots & 2 \\ 0 & \mathbf{-2} & 4 & \dots & 0 \\ 2 & -1 & \mathbf{0} & \dots & 1 \\ 1 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 5 & 1 & 0 & \dots & \mathbf{3} \end{pmatrix} \right. =$$

$$\underbrace{\left\lfloor \underbrace{\begin{pmatrix} 0.38 & -1.3 & 0.74 & 0.47 \\ -1.1 & -1.1 & -0.2 & 1.73 \\ -0.9 & 5.43 & 1.99 & -1.4 \\ \dots & \dots & \dots & \dots \\ -1.0 & 0.34 & 10.2 & 0.69 \end{pmatrix}}_{\mathbf{R},\ K \times |\Sigma|} \cdot \underbrace{\begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}}_{\mathbf{L},\ |\bigcup_{q=q_1}^{q_2} \Sigma^q| \times n} \cdot \underbrace{\begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}}_{\mathbf{C},\ n \times K} \right\rfloor}$$

столбцы$\sim$по Коши

индикаторная матрица

столбцы – случайные окна шириной w

Рис. 3.1. Пример произведения матриц (3.14) с указанием размерностей и способов получения матриц

ственные случайные направления. Однако, заметим, что из-за наличия общих единиц между столбцами матрицы C части маскированного (прореженного) q-граммного вектора будут проецироваться на различные направления.

Из выражения (3.14) следует:

$$v_i(x) = \sum_{\sigma \in \Sigma} \sum_{k=1,\dots,n} c_{ki} r_{i\sigma} l_{\sigma k} = \sum_{k=1,\dots,n} R(C(k)) l_k,$$

где $R(C(k))$ – матрица R с только теми строками, которые соответствуют тем векторам позиций, которые имеют единицы в позициии k. Таким образом, каждый вектор l_k проецируется на соответствующее случайное подпространство,

определяемое теми C_j, которые накрывают позицию k. Чем дальше находятся идентичные символы (или q-граммы) друг от друга, тем меньше общих случайных направлений они имеют, и, таким образом, тем дальше друг от друга будут находиться их проекции.

Исходя из анализа в п. 3.1, векторы r_i должны выбираться градуальными и распределенными по Коши, векторы позиций – бинарными (строки матрицы, составленной из столбцов, где единицы указывают положения, накрытые случайно выбранными окнами ширины w).

Таким образом, мы получили согласующуюся с нейросетевыми концепциями (п. 3.4.2.2) схему формирования распределенных представлений последовательностей, а также возможность сублинейного поиска приближенно ближайших последовательнотсей путем применения схемы локально-чувствительного хеширования (п. 1.5.1) за счет использования в ней разработанной функции (1.18).

3.4.4. Тернаризация и бинаризация рандомизированных представлений на основе локально-чувствительной функции

Помимо выбора параметра r (3.1), исходя из минимизации отношения p_I/p_{II} (cf. [35]) и значения $\rho = \ln(p_I/p_{II})$, и следовательно, ускорения приближенного поиска ближайшей строки (см. выражение для L (1.16)), можно выбирать его, исходя из удобства представления и работы с получаемыми векторами. Так, работать с тернарными (или даже бинарными) векторами предпочтительнее, чем с целочисленными, из-за меньших требований к памяти и более быстрых реализаций алгоритмов. Также (разреженные) бинарные и тернарные вектора используются в моделях распределенной обработки информации, таких как АПНС [91], или для поиска текстов [132, 131]. Поэтому представляет интерес получение на выходе тернарных или бинарных векторов.

Рассмотрим, как выбор значений параметров влияет на множество значений, принимаемых хеш-функцией (1.18). Хеш-функция (1.18) будет принимать тернарные значения $\{-1, 0, 1\}$, если $-1 < \left\lfloor \frac{(v,\bar{\phi})+b}{r} \right\rfloor < 2$, и отсюда

$-r < (v, \bar{\phi}) < r$. Интегрируя выражение $(\pi(1+x^2))^{-1}$ в пределах от $-r/\left\|v\right\|_{l_1}$ до $r/\left\|v\right\|_{l_1}$, получаем, что вероятность получения тернарных значений в (1.18) равна $\frac{2}{\pi} arctan(\frac{r}{\left\|v\right\|_{l_1}})$.

Плотность нулевых элементов в векторах определяется величиной

$$\frac{1}{\pi}\left(arctan\left(\frac{r}{\left\|v\right\|_{l_1}} - \frac{\left\|v\right\|_{l_1}}{2r}\ln(1+(\frac{r}{\left\|v\right\|_{l_1}})^2)\right)\right)$$

и увеличивается с увеличением параметра r.

Полученные тернарные векторы могут быть бинаризованы пороговой операцией [132]. Также, для бинаризации значений векторов разработанной локально-чувствительной функции может использоваться метод бинаризации из [23]. Эффективность отражения сходства строк с помощью таких тернарных или бинарных распределенных представлений следует исследовать на практике.

Материалы данного раздела изложены в следующих публикациях: [107, 109, 105, 106, 145, 146].

3.5. Выводы по разделу 3

1. На основе разработанного детерминированного метода вложения пространства последовательностей с классической метрикой редактирования в пространство ℓ_1 предложено новое семейство локально-чувствительных функций для классического расстояния редактирования, продуцирующих нейросетевые распределенные представления последовательностей путем использования рандомизации векторных представлений и связывания элементов последовательности с их позициями. Это обеспечивает унификацию формата представления, малую ресурсоемкость, сохранение сходства последовательностей и возможность использования мер сходства векторных представлений для оценки сходства последовательностей.
2. На основе разработанной локально-чувствительной функции предложен метод приближенного поиска ближайших строк по расстоянию редак-

тирования, который имеет большую вычислительную эффективность, чем применение векторных детерминированных представлений раздела 2. Выведены теоретические оценки эффективности предложенных методов поиска.

3. На основе нейросетевого распределенного представления информации предложены два метода (сдвиговый и прореживающий) распределенного представления последовательностей, учитывающие порядок компонентов последовательностей за счет связывания векторов позиций и элементов. Показано, что прореживающее распределенное представление может рассматриваться как рандомизированное вложение расстояния редактирования на основе локально-чувствительных функций.
4. Показана возможность получения тернарных и бинарных векторов путем выбора параметров локально-чувствительного хеширования, что может быть полезно для систем, использующих распределенное представление информации.

РАЗДЕЛ 4

АЛГОРИТМИЧЕСКАЯ РЕАЛИЗАЦИЯ И ЧИСЛЕННОЕ ИССЛЕДОВАНИЕ РАСПРЕДЕЛЕННОГО ПРЕДСТАВЛЕНИЯ И ПОИСКА ПРИБЛИЖЕННЫХ БЛИЖАЙШИХ СИМВОЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

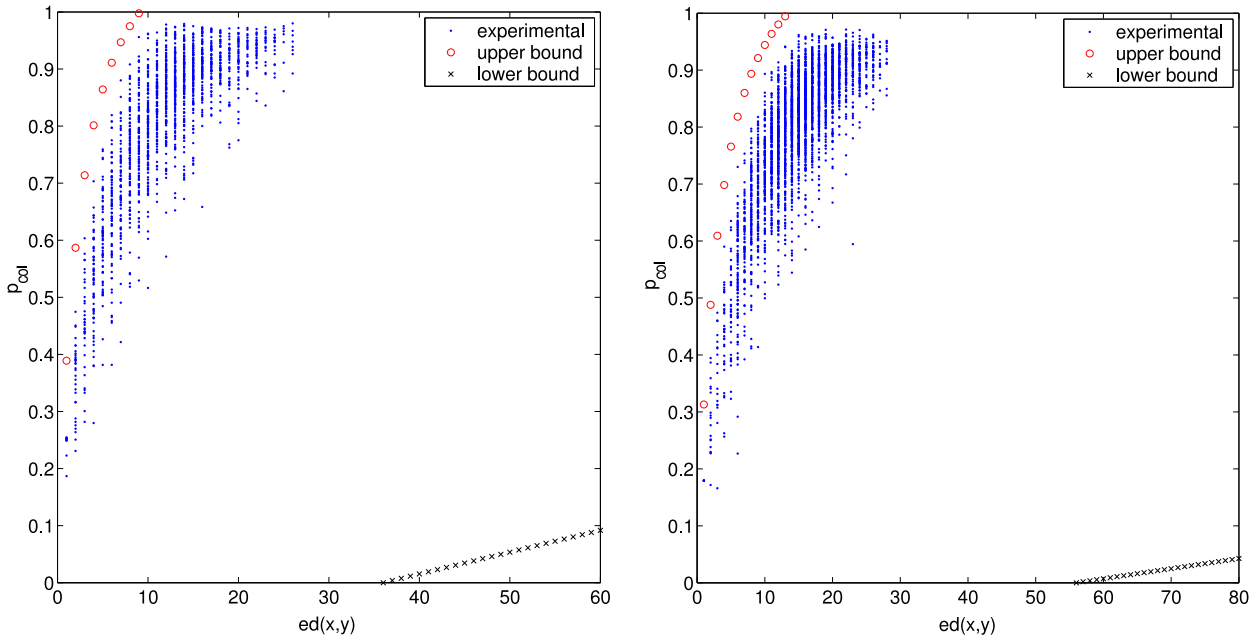
Раздел посвящен экспериментальному численному исследованию метода распределенного представления символьных последовательностей на основе рандомизированного вложения классического расстояния редактирования, разработанного в разделе 3. Приведена алгоритмическая реализация (п. 4.2) метода поиска приближенных ближайших последовательностей с помощью локально-чувствительного хеширования LSH. Предложены методы работы с базами последовательностей разной длины (п. 4.3). Проведены эксперименты со схемой на основе 1-стабильного распределения на искусственно сгенерированных данных для проверки соответствия эмпирической вероятности совпадения значений хеш-функции теоретически полученным ограничениям (п. 4.1). Проведены экспериментальные исследования влияния выбора параметров поиска приближенных ближайших последовательностей на эффективность и точность поиска, а также упорядоченность получаемых кандидатов на приближенных соседей (п. 4.4).

4.1. Экспериментальное исследование вероятности коллизии разработанной локально-чувствительной функции

Экспериментальное исследование соответствия вероятности совпадения (коллизии) $p_{col} = Prob[h(x) = h(y) \mid ed(x, y)]$ значений хеш-функции (3.1)и теоретических оценок (3.3) и (3.2) проводились на базе RandomStrings (п. 2.5.2).

На рис. 4.1 приведены экспериментально полученные вероятности совпа-

дения (коллизии) для строк длиной $n = 1000$ и $n = 3000$ вместе с теоретическими точками верхней (нолики) и нижней (крестики) границ. Результаты показывают, что экспериментальные данные соответствуют полученным теоретически ограничениям (3.3) и (3.2).



(a) $n = 1000$ (b) $n = 3000$

Рис. 4.1. Экспериментальные значения вероятностей коллизии хеш-функций $h_i(x)$ (3.1)

4.2. Поиск приближенных ближайших последовательностей с помощью LSH-леса

Для реализации разработанной процедуры LSH (п. 3.3) при поиске приближенных ближайших соседей использована ее модификация, называемая LSH-лес (п. 1.5.1) и позволяющая пополнять базу примеров последовательностей P и изменять параметры k_1, k_2 без обновления хеш-векторов [13].

Для каждого $j = 1, ..., L$ все -мерные хеш-векторы $h_j(x)$ всех строк $x \in P$ хранятся в виде отдельного префиксного дерева T_j (рис. 4.2) глубиной до K уровней (корень имеет глубину 0, листья – K). Узлы дерева соответствуют значениям элементов хеш-вектора и содержат ссылки на строки, хеш-векторы

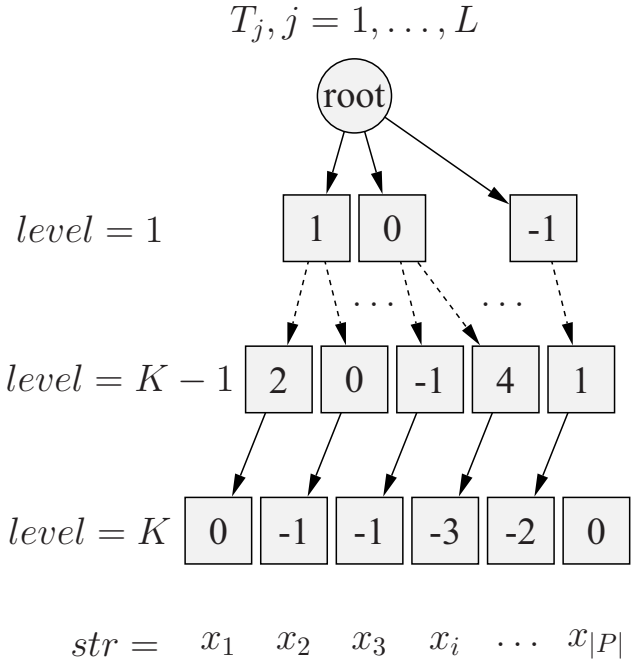


Рис. 4.2. Дерево T_j LSH-леса

которых соответствуют пути от корня дерева к данному узлу. Листья деревьев соответствуют ячейкам в схеме LSH (п. 1.5.2).

Если в хеш-векторах двух строк совпадают первые k элементов хеш-векторов, то и первые k узлов на пути от корня дерева T_j до листьев, соответствующих этим строкам, также совпадают.

При поступлении запроса y:

1. формируются L его K-мерных хеш-векторов $h_j(y)$;
2. для каждого хеш-вектора $h_j(y)$ в T_j находится узел, соответствующий $h_j(x)$ с наибольшим числом совпадающих первых элементов этих векторов;
3. для найденных на шаге 2 узлов самого глубокого уровня, для всех L деревьев соответствующие узлам этого уровня строки x добавляются в результирующее мультимножество S;
4. после того, как все строки, хеш-векторы которых совпадают на данном уровне, добавлены в S, все L деревьев синхронно просматриваются на следующем (более «высоком») уровне. Процедура повторяется, пока не достигнем корня или $|S|$ не превысит $2L$.

В результате получаем мультимножество S строк (кандидатов на приближен-

ного ближайшего соседа к y), упорядоченное в порядке убывания глубины узлов дерева, до которых совпадали первые элементы хеш-векторов соответствующей строки и запроса. Будем далее под уровнем строки из S понимать глубину последнего узла дерева, на котором еще совпадали первые элементы хеш-векторов запроса и этой строки.

Исследование влияния размера мультимножества S на результаты поиска рассмотрено в п. 4.4.

Согласно результатам [13], мультимножество S будет содержать с ненулевой константной вероятностью приближенных ближайших соседей к запросу. Параметры леса при этом выбираются, исходя из теоремы Индыка-Мотвани (п. 1.5.2, стр. 36). Выбор параметров на практике исследован в п. 4.4.

4.3. Методы работы с базами с разной длиной последовательностей

Для поиска приближенных ближайших последовательностей в базах, содержащих последовательности различной длины, предлагаются следующие способы применения метода поиска приближенных ближайших последовательностей.

4.3.1. Дополнение последовательностей спецсимволами

Для приложений, где отличие длин строк в P не слишком большое, простейшим способом сведения к одинаковой длины является дополнение строк одинаковым специальным символом $\$ \notin \Sigma$ до максимальной длины n, представленной в P: $n = \max_{p \in P} |p|$. Каждая строка $p \in P$ преобразуется в $p' = p\$^{n-|p|}$. Поскольку для любой пары строк $p_1, p_2 \in P$ выполняется $ed(p_1, p_2) = ed(p'_1, p'_2)$, то поиск приближенно ближайших соседей выполняется в базе дополненных строк $P' = p' = p\$^{n-|p|} | p \in P$ и, если p' является приближенным соседом к q', то соответствующая строка p является приближенным соседом к q.

4.3.2. Разбиение последовательностей окнами

Для приложений с длинными последовательностями предлагается применить разбиение строк из P, а также запроса q, скользящим окном фиксированной ширины n. После этого поиск выполняется для каждого окна запроса вида $q[i, i+n-1], i = 1, \ldots, |q| - n + 1$ в базе $P' = \{p[i, i+n-1] \mid i = 1, \ldots, |p| - n + 1,\ p \in P\}$.

4.3.3. Кластеризация последовательностей по длине

Поскольку минимальное значение расстояния редактирования между строками длины n и m составляет $|m - n|$, а похожие строки, как правило, мало отличаются по длине , то поиск соседей имеет смысл выполнять только для ближайшей окрестности (кластера). Поэтому для поиска приближенно ближайших соседей в базах строк с большим разнообразием значений длин предлагается использовать следующую схему кластеризации базы по длине.

Зададимся параметром ε, определяющим ширину кластеров следующим образом: для фиксированной длины n (центр кластера) окружающим кластером считаются все строки, длины которых содержатся в интервале $I_n = [\lfloor (1-\varepsilon)n \rfloor, \lceil (1+\varepsilon)n \rceil]$.

Значения центров n можно выбирать по следующему алгоритму:

$$n_0 = \min_{p \in P, |p| \neq 0} |p|,$$
$$n_{i+1} = \lceil (1+\varepsilon) n_i \rceil.$$

Таким образом, каждое значение длины строки n накрывается двумя интервалами, что исключает пропуск возможных соседей для строк-запросов с длинами, лежащими на границах кластеров.

Запрос $p, |p| = n$ выполнялся для двух LSH-лесов, базы P_1 и P_2 которых состоят из строк, длины которых попадали в оба интервала, которые «накрывают» значение длины запроса. Длины строк и запроса в каждом интервале выравнивались добавлением специальных символов в конце строки до значения максимальной длины в каждом интервале, как в методе п. 4.3.1.

Разработанные методы и процедуры использованы в приложениях раздела 5.

4.4. Выбор параметров для обеспечения вычислительной эффективности и точности поиска

На практике, так как необходимые для поиска ресурсы растут линейно от L (1.16), рекомендуемые теорией значения L часто оказываются слишком велики. Однако, при использовании меньших значений L теория (п. 1.5.1) не гарантирует, что в возвращенное мультимножество строк S попадет как минимум одна строка y из $S(q, k_2)$. Следующие эксперименты выполнялись для проверки того, как на «качество» мультимножества S влияет

1. выбор меньших, чем теоретические, значений L и
2. дополнительная фильтрация S, которой предположительно можно отсечь строки $y \notin S(q, k_2)$ (false positives).

Качество S оценивалось

1. по значению точности (п. 4.4.1);
2. по упорядоченности S относительно известного эталона (п. 4.4.2).

4.4.1. Экспериментальное исследование качества кандидатов на ближайшего соседа

Традиционно в системах поиска текстовой информации для оценки качества множества документов, возвращаемого в ответ на запрос, анализируется компромисс зависимости между количеством найденных релевантных текстов (true positives) и количеством найденных нерелевантных текстов (false positives). Для этого определяются величины точности (отношение количества релевантных запросу возвращенных документов к общему числу возвращенных документов) и полноты (отношение количества релевантных запросу возвращенных документов к общему числу релевантных документов) и изображают их на графиках зависимости точности от полноты [9].

При решении задачи поиска приближенного ближайшего соседа подразумевается нахождение одного соседа, а не множества соседей. Поэтому отношение $|S(q, k_2) \cap P \cap S|/|S(q, k_2) \cap P|$ (полнота) не информативно. Зачастую полнота не информативна и при оценке качества поисковых систем. Для пользователя все множество возвращенных результатов не интересно, он обычно ограничивается просмотром только первых страниц результатов [110]. В таких случаях вместо точности и полноты обычно используют характеристику «точность на уровне n» («precision at n /P@n») [53], вычисляемую как точность на ограниченном первыми n результатами множестве. В нашем случае этот уровень естественно ограничен размером множества S и точность на уровне $|S|$ определяется как $|S(q, k_2) \cap P \cap S|/|S|$.

Чтобы не вносить смещение в оценку точности на уровне $|S|$ из-за неодинакового количества строк внутри и вне шара $S(q, k_2)$ и разного количества строк на определенном расстоянии от q, было сделано следующее. Мы отобрали из набора RandomStrings (п. 2.5.2) 2200 строк длиной 1000 символов (при этом $k_2 = 126$) таким образом, что число строк, принадлежащих $S(q, k_2)$, составляло половину от общего числа строк, и количество строк на каждом расстоянии от центра q не превышало 10. Полученное множество P содержало строки, находящиеся от q на расстоянии редактирования от 10 до 248.

Множество P было запомнено в LSH-лесе с помощью процедуры из п. 4.2. На вход процедуры поиска приближенно ближайшего соседа подавался запрос q. По полученному на выходе при каждом запросе множеству S ($|S| > 4L$) вычислялась точность на уровнях $0.5L, L, 2L, 3L, 4L$. Значения точностей усреднялись по 100 случайным независимым реализациям LSH-леса. Их значения и дисперсия приведены в таблице 4.4.1.

При малых значениях L наблюдается максимальная точность, что является особенностью процедуры LSH-лес, возвращающей множество строк S, упорядоченное по глубине уровня. При увеличении значения L вначале точность падает, что объясняется бóльшими шансами у строк из P $S(q, k_2)$ попасть в S, но в дальнейшем точность перестает существенно изменятся, и

Таблица 4.1

Точность на уровне $|S|$ в зависимости от значения L и $|S|$

L	$\|S\| = 0.5L$, где $\|S\|$ – целое	σ_p	$\|S\| = L$	σ_p	$\|S\| = 2L$	σ_p	$\|S\| = 3L$	σ_p	$\|S\| = 4L$	σ_p
1			0.950	0.048	0.945	0.029	0.927	0.025	0.893	0.031
2	0.930	0.065	0.895	0.056	0.853	0.054	0.825	0.032	0.810	0.028
3			0.885	0.050	0.816	0.035	0.784	0.030	0.770	0.025
4	0.855	0.071	0.842	0.041	0.810	0.031	0.777	0.023	0.757	0.021
5			0.824	0.039	0.786	0.021	0.759	0.014	0.735	0.014
6	0.853	0.052	0.797	0.040	0.782	0.025	0.760	0.019	0.730	0.014
7			0.778	0.031	0.768	0.018	0.743	0.013	0.721	0.010
8	0.846	0.038	0.791	0.024	0.755	0.016	0.729	0.013	0.724	0.010
9			0.759	0.024	0.741	0.013	0.717	0.011	0.697	0.009
10	0.860	0.030	0.787	0.024	0.749	0.015	0.722	0.009	0.689	0.008
12	0.807	0.035	0.776	0.021	0.740	0.013	0.712	0.009	0.688	0.007
14	0.821	0.028	0.770	0.018	0.740	0.010	0.716	0.007	0.682	0.006
16	0.809	0.021	0.746	0.013	0.721	0.010	0.691	0.007	0.673	0.005
18	0.845	0.019	0.765	0.014	0.719	0.008	0.686	0.005	0.665	0.004
20	0.811	0.024	0.762	0.014	0.708	0.006	0.682	0.005	0.658	0.004

одновременно уменьшается дисперсия ее значений, что позволяет говорить о стабилизации значений точности. Аналогичный эффект наблюдается при увеличении $|S|$. Таким образом, на практике достаточно ограничиться значением L, равным нескольким десяткам (например, 30 или 40). Это позволяет получить приемлемый уровень точности и сэкономить ресурсы. Значение $|S|$ можно зафиксировать на значении $2L$ (п. 1.5.2).

4.4.2. Экспериментальное исследование порядка ближайших соседей

Исследуем, насколько порядок строк в S соответствует «реальному» упорядочению этих же строк по расстоянию редактирования до q. Обозначим S' мультимножество значений расстояния редактирования строк из S до q: $S' = \{ed(x, q) \mid x \in S\}$.

Для сравнения упорядоченных мультимножеств за основу было принято обобщенное расстояние Кендалла [40], которое позволяет сравнивать упорядоченные множества, рассматривая различные штрафы за нахождение элементов в разном относительном порядке в двух множествах M_1, M_2. Так, если два элемента $i, j \in M_1 \cup M_2$ входят в сравниваемые множества под индексами i_1, i_2, j_1, j_2, то штраф u вычисляется по следующим правилам:

1. если $i_1 < i_2, j_1 < j_2$ $(i_1 > i_2, j_1 > j_2)$, то $u = 0$;
2. если $i_1 < i_2, j_1 > j_2$ $(i_1 > i_2, j_1 < j_2)$, то $u = 1$;
3. если i_2 (i_1) не определено, т.е. соответствующий элемент не входит во второе (первое) множество, а $j_1 < j_2$ $(j_1 > j_2)$, то $u = 0$;
4. если j_2 (j_1) не определено, т.е. соответствующий элемент не входит во второе (первое) множество, а $i_1 < i_2$ $(i_1 > i_2)$, то $u = 0$;
5. если i_1, j_2 (i_2, j_1) не определено, т.е. соответствующие элементы входит каждый в свое множество, то $u = p$.

Параметр p есть регулируемая характеристика «степени пессимистичности» ситуации, когда один из элементов отсутствует в одном множестве, но присутствует в другом, и наоборот. Для наших экспериментов использовалось $p = 1$.

Расстояние $D_K(M_1, M_2)$ между множествами является суммой штрафов всех пар элементов $M_1 \cup M_2$:

$$D_K(M_1, M_2) = \sum_{i,j \in M_1 \cup M_2} u(i, j). \tag{4.1}$$

Мы изменили описанный алгоритм подсчета обобщенного расстояния Кендалла для применения к упорядоченным мультимножествам (значений расстояний редактирования от q до ближайших строк) следующим образом. Для каждого из значений расстояния редактирования из $S'_1 \cup S'_2$, входящего хотя бы в одно из мультимножеств, каждое его j-ое вхождение в оба мультимножества последовательно преобразуются в уникальный абстрактный элемент нового множества, условно обозначаемый символом s_j, где индекс соответствует порядковому номеру, под которым он входит в данное множество. В частности, если элемент входит в одно их множеств только один раз, то его индекс в нем 1.

Например, из множеств $S'_1 = \{1, 2, 3, 4, 4, 3, 5\}$, $S'_2 = \{1, 3, 3, 4, 4, 3, 4\}$, получаем следующие множества элементов: $S'_1 \to M_1 = \{1_1, 2_1, 3_1, 4_1, 4_2, 3_2, 5_1\}$, $S'_2 \to M_2 = \{1_1, 3_1, 3_2, 4_1, 4_2, 3_3, 4_3\}$. Полученное описанным способом из S' множество элементов будем обозначать $M(S')$. Таким образом, мы свели задачу сравнения упорядоченных мультимножеств S'_1 и S'_2 к задаче сравнения упорядоченных множеств M_1, M_2.

Как и в эксперименте в пункте 4.4.1, количество строк в S, возвращаемых процедурой LSH-лес, было переменным. То есть алгоритм возвращал множество из $2L$ строк, полученных в результате этапа подъема по LSH-деревьям (п. 4.2), при изменяющемся L.

Исследование проведено на наборе строк RandomStrings (п. 2.5.2). Мы использовали те же 2200 строк длиной 1000 символов, что и в предыдущем эксперименте п. 4.4.1. Множество P было запомнено в LSH-лесе с помощью процедуры LSH-лес (п. 4.2). На вход процедуры поиска приближенно ближайшего соседа подавался запрос q.

Обозначим как X упорядоченное по возрастанию мультимножество значений реальных расстояний редактирования от центра q до его ближайших соседей. Обозначим как Y упорядоченное в ходе процедуры LSH-лес мультимножество значений расстояний редактирования от центра q до возвращенных строк в ходе процедуры LSH-лес. Расстояние $D_K(M(X), M(Y))$ (4.1) вычислялось при $|S| = 10, 50, 100, 200, 300, 500$, где размер множества $M(X)$ ограничивался по значению $|M(Y)| = |S|$. Результаты усреднялись по 100 случайным независимым реализациям LSH-леса. Зависимость $D_K(M(X), M(Y))$ от количества деревьев L в лесе изображена на рис. 4.3 для различных ограничений на максимальный размер возвращенного множества S и для следующих способов его дополнительной фильтрации:

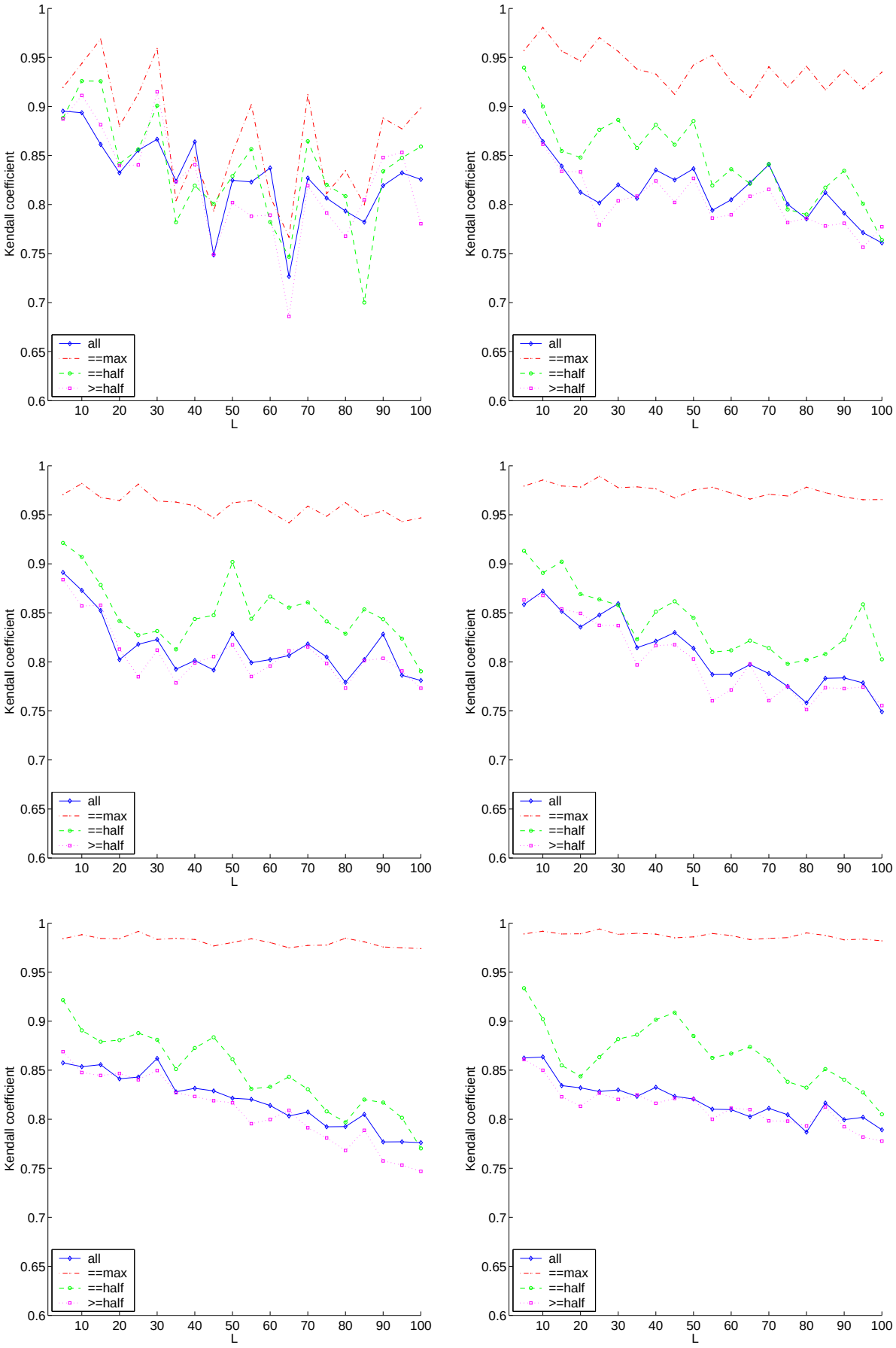


Рис. 4.3. Зависимость $D_K(M(X), M(Y))$ от L для $|S| = 10, 50, 100, 200, 300, 500$

1. «all» (без фильтрации);
2. «==max» (только строки, совпавшие с запросом q на самом глубоком для данного S уровня);
3. «==half» (строки, совпавшие на уровне от $\lfloor K_{avg} \rfloor$ до $\lceil K_{avg} \rceil$;
4. «>=half» (строки, совпавшие на уровне большем или равном K_{avg});

где K_{avg} – среднее значение уровня среди возвращенных строк.

Из рис. 4.3 видно, что $D_K(M(X), M(Y))$ незначительно уменьшается при увеличении L, что можно объяснить увеличением $|S|$. Поэтому, как и в эксперименте (п. 4.4.1), можно сделать вывод о возможности фиксирования L на небольшом значении.

Способы фильтрации «==max» и «==half» «выигрывают» у остальных способов (по значению D_K), подтверждая то, что совпадение строк на более глубоких уровнях свидетельствует об их большем сходстве.

Результаты данного раздела отражены в публикациях [146, 147].

4.5. Выводы по разделу 4

1. Разработанная алгоритмическая реализация запоминания векторных представлений базы примеров за счет использования LSH-леса позволяет осуществлять поиск приближенных ближайших последовательностей, обеспечивая эффективность счет использования структуры деревьев, и позволяющая не хешировать заново все последовательности базы при изменении параметров поиска и размера базы.
2. Разработанные методы поиска приближенных ближайших символьных последовательностей для последовательностей разной длины (дополнением спецсимволами, разбиением на окна, кластеризацией по длине) позволяют осуществлять поиск последовательностей в реальных базах.
3. Экспериментальное исследование на модельных данных показало соответствие диапазона эмпирической вероятности коллизии разработанных локально-чувствительных хеш-функции теоретически полученным огра-

ничениям.

4. Экспериментальное исследование на модельных данных показало возможность применения на практике небольшого (несколько десятков) количества деревьев в процедуре LSH-лес и размера мультимножества кандидатов на приближенных соседей, при сохранении точности поиска. Дополнительная фильтрация последовательностей-кандидатов на приближенных ближайших соседей по их уровню в LSH-лесе при небольшом количества деревьев сохраняет соответствие упорядоченности множества кандидатов реальному упорядочению по расстоянию редактирования.

РАЗДЕЛ 5

РЕАЛИЗАЦИЯ И ПРИМЕНЕНИЕ РАЗРАБОТАННЫХ МЕТОДОВ РАСПРЕДЕЛЕННОГО ПРЕДСТАВЛЕНИЯ И ПОИСКА ПОСЛЕДОВАТЕЛЬНОСТЕЙ

В данном разделе рассматриваются разработанные программные средства, в которые реализованы методы и алгоритмы распределенного представления символьных последовательностей, их сравнения и поиска сходных (разд. 3, п. 3.3). Также рассмотрено их применение в следующих актуальных практических задачах классификации: обнаружение дубликатов (п. 5.2.1), обнаружения спама (п. 5.2.2), классификация участков ДНК (п. 5.2.3), а также в задаче классификации сессий пользователей с целью обнаружения аномалий в компьютерных системах (п. 5.2.4).

5.1. Разработка программных средств

Разработаны программные средства, которые использовались для решения прикладных задач (п. 5.2), а также переданы в другие организации (см. акты внедрения в Приложении А).

5.1.1. Программные библиотеки

Были разработаны следующие программные объектно-ориентированные библиотеки.

1. Библиотека форматированного ввода TextInputTools – содержит модули форматированного ввода для баз генетических последовательностей, электронных писем, ряда популярных текстовых корпусов, аудит-последовательностей UNIX-систем.
2. Библиотечный модуль VectorComparer – унифицированное средство срав-

нения векторов по стандартным метрикам и мерам на основе шаблонных классов, что позволяет его использование и адаптацию под разные типы данных – векторы с элементами разных типов, квазивекторные данные (например, деревья) и др.

3. Библиотека LSHLibrary – реализует методы представления и поиска приближенных ближайших символьных последовательной (разд. 3).

   Спроектирована для работы с последовательностями произвольной символьной природы (текстовые последовательности, генетические последовательности, последовательности команд и т.д.).

   Основные классы библиотеки LSHLibrary:

   - `tree_encoder` – класс, отвечающий за формирование хеш-функций по входным строкам. В реализации псевдослучайной генерации векторов по q-грамме используется кодирование q-граммы по алгоритму хеширования Карпа-Рабина [65], результат которого инициализирует генератор псевдослучайных чисел;
   - `lsh_tree` – класс, реализующий заполнение, хранение, и запросы к LSH-дереву (п. 4.2);
   - `lsh_tree_node` – класс узла LSH-дерева, обеспечивающий хранение значения хеш-функции (3.1) (п. 3.1.2);
   - `lsh_forest` – класс, обеспечивающий создание, управление и логику обработки запросов к лесу – набору LSH-деревьев;
   - `lsh_forest_collector` – класс, обеспечивающий создание и управление несколькими LSH-лесам, а также обработку запросов к ним.

Все библиотеки реализованы в виде шаблонных классов для C++ без использования системно-зависимых функций, что обеспечивает их мультиплатформенность. Основой программных средств является объектно-ориентированная библиотека классов C++ CommonLib, содержащая классы векторов различных видов, методы генерации случайных величин и векторов из различных распределений, другие часто используемых компоненты. Библиотеки

могут быть использованы для решения широкого круга как исследовательских, так и практических задач.

5.1.2. Программный нейрокомпьютер SNC

Модульный программный нейрокомпьютер SNC разработан в отделе нейросетевых информационных технологий Международного научно-учебного центра информационных технологий и систем НАН и МОН Украины. SNC принадлежит к нейрокомпьютерам общего назначения [73, 125, 133, 137], предназначенным для решения широкого круга задач. Система имеет модульную архитектуру и позволяет визуально конструировать конфигурации обработки данных, используя расширяемый набор обрабатывающих блоков.

Возможность реализации произвольных алгоритмов обработки информации обеспечивается средствами визуального конструирования потоков данных и потоков управления, а также поддержкой встроенного языка JavaScript. Разделение потоков данных и управления с возможностью их визуального конфигурирования не реализовано в других существующих системах.

Функциональность нейропакета наращивается путем реализации новых обрабатывающих блоков: форматов и обрабатывающих блоков. Универсальный механизм сохранения результатов в БД позволяет сохранять в удобном виде все необходимые данные выполняемого проекта: параметры конфигурации, результаты работы, и т.д. Информация может использоваться для дальнейшей обработки и формирования отчетов.

В основу архитектуры SNC положена технология COM [121], которая позволила стандартизировать программные модули и облегчить включение в систему новых модулей. Ядром системы (рис. 5.1) является модуль Configuration Manager, который управляет процессом выполнения конфигурации и координирует совместную работу дополнительных модулей.

Существует два типа дополнительных модулей: форматы и обрабатывающие блоки. Модули форматы реализуют чтение и запись внешних данных, например генетических последовательностей из формата FASTA, аудит-логов

из формата BSM или ACCT и т.п. Модули обрабатывающих блоков реализуют различные алгоритмы обработки данных – например, обучение и классификацию на основе персептрона, алгоритм ближайшего соседа, метод опорных векторов и др. Поддержка новых форматов и реализация новых алгоритмов осуществляется сравнительно несложно благодаря стандартизированным COM интерфейсам и шаблонным классам, реализующим необходимую минимальную функциональность.

Сохранение параметров и результатов выполнения проектов в базе данных обеспечивается модулем Storage. При запуске конфигурации в базе данных создается иерархическая схема данных проекта. В неё помещаются схема данных конфигурации проекта и схемы данных, поддерживаемых обрабатывающими блоками проекта.

Для создания конфигураций разработана графическая оболочка Graph Shell (рис. 5.2), которая позволяет визуально в режиме САПР настраивать обрабатывающие блоки, а также потоки данных и управления.

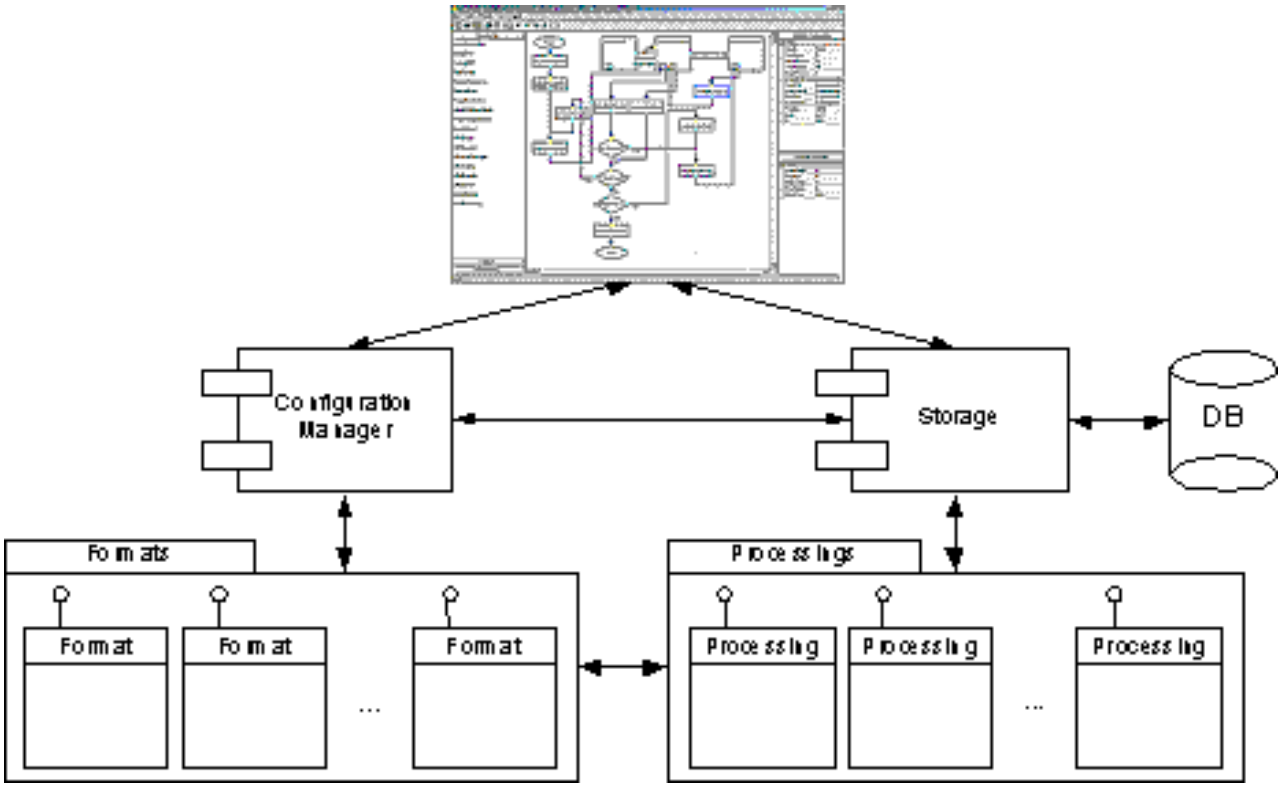


Рис. 5.1. Архитектура программного нейрокомпьютера SNC

В рамках программного нейрокомпьютера реализовано большое количество реализованных блоков форматов и обрабатывающих блоков, что позволяет легко строить различные алгоритмы обучения и классификации. На рис. 5.2 приведен пример реализации одного из проектов.

Результаты работы сохраняются в файле и базе данных с помощью механизмов модуля Storage, а также могут быть визуализированы при выполнении проекта в графической оболочке (см. рис. 5.3).

Программный нейрокомпьютер SNC представляет собой эффективное средство для создания нейросетевых технологий решения разноплановых исследовательских, образовательных и прикладных задач. Гибкая системная архитектура, большое количество реализованных методов и алгоритмов в сочетании со средствами визуального построения алгоритмов обработки позволяют конфигурировать систему для решения требуемых задач.

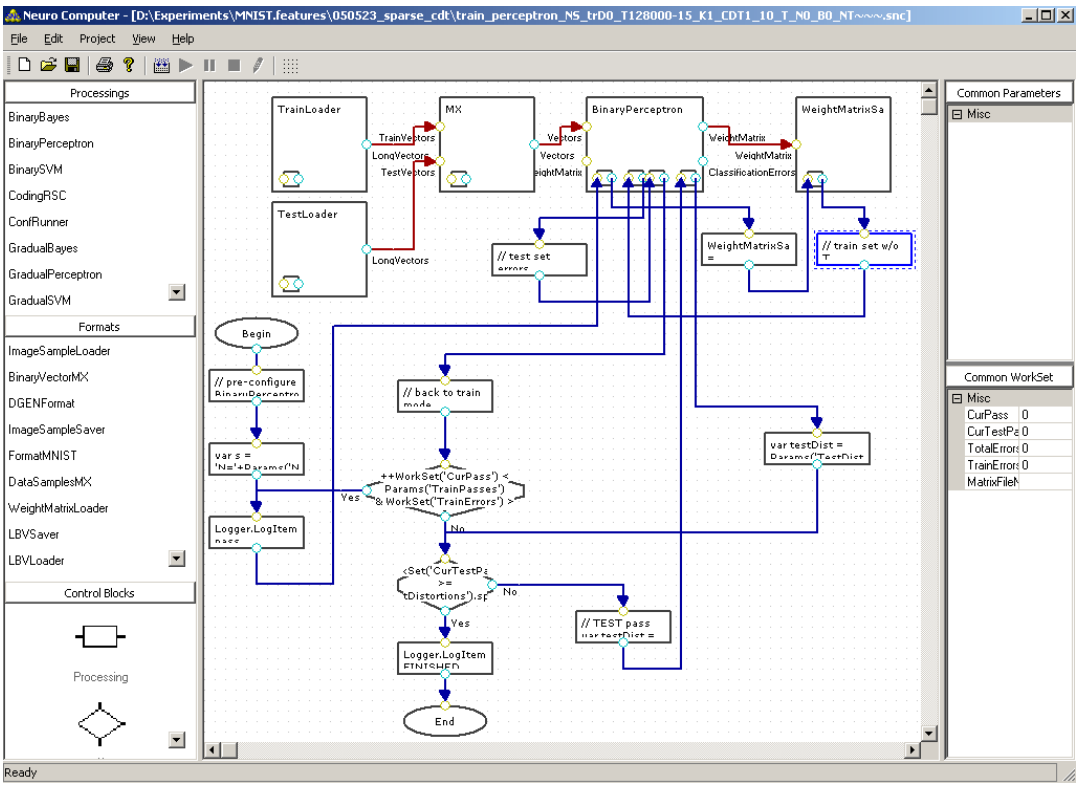


Рис. 5.2. Графическая оболочка программного нейрокомпьютера

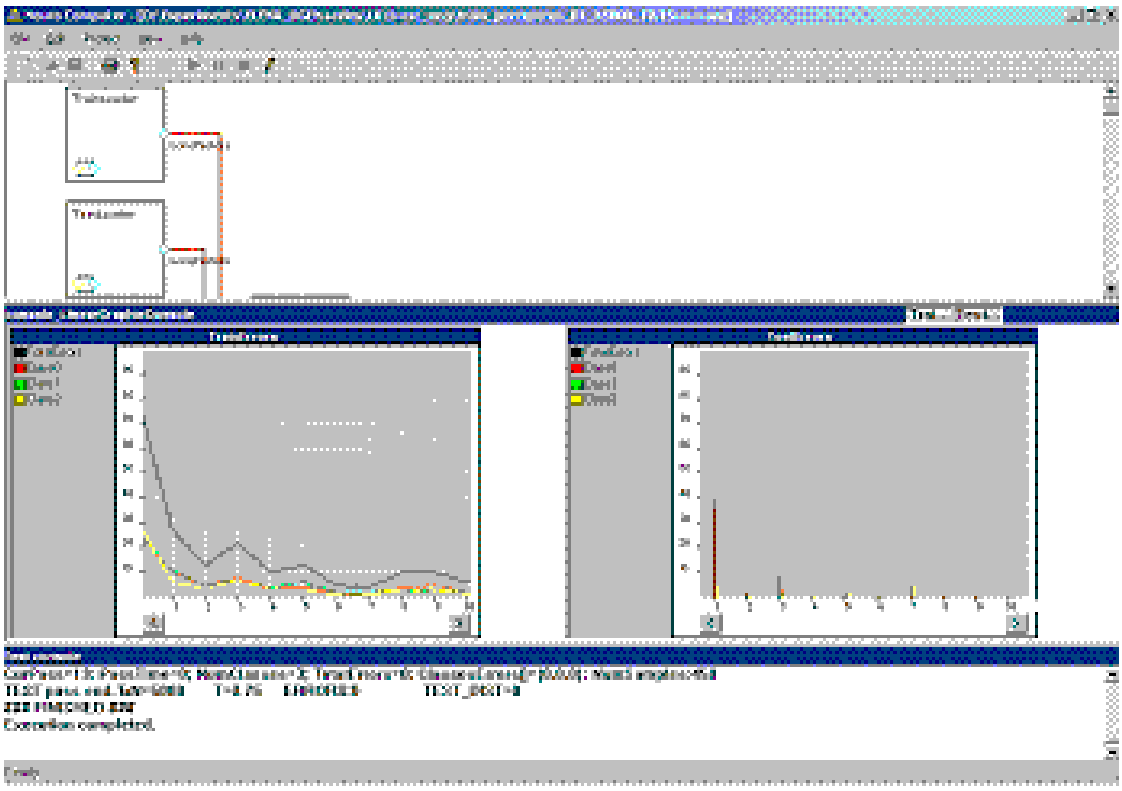


Рис. 5.3. Графическая оболочка SNC в режиме выполнения конфигурации

5.1.3. Специализированные программные системы

На основе разработанных библиотек для поиска текстовых дубликатов, спама, кодирующих участков генетических последовательностей и классификации сессий пользователей UNIX-системы были разработаны программные макеты DuplClassifier (поиск приближенных тектовых дубликатов и дубликатов веб-документов), EmailClassifier (классификация электронной почты), NuclClassifier (классификация нуклеотидов генетических последовательностей), SessionClassifier (классификация пользовательских сессий в компьютерной системе).

Программные средства реализованы с использованием языка C++ и с интерфейсом командной строки, что обеспечивает мультиплатформенность.

5.2. Применение в задачах классификации

5.2.1. Исследование поиска дубликатов в текстовых и веб-коллекциях

Метод п. 3.3 был исследован в задаче поиска дубликатов в текстовых базах. Поиск (приближенных) дубликатов текстовых документов является операцией, которую необходимо эффективно выполнять в системах документооборота. Особенно критичной эта операция становится в поисковых машинах Интернет [17]. Документы, являющиеся приближенными дубликатами друг друга, чрезвычайно распространены в Интернет. Яркими примерами являются различные сборники FAQ, справочники по языкам программирования, командам операционных систем. Кроме того, некоторые технологии привлечения посетителей на сайты предусматривают наполнение «поддельных» страниц частями текстов легальных страниц с целью привлечения случайных посетителей для показа платной рекламы или перенаправления на другие сайты [48].

5.2.1.1. Базы данных для экспериментального исследования

Поиск дубликатов осуществлялся в следующих базах новостных текстов Reuters-21578 [74], учебных текстов British National Corpus [15], характеристики которых приведены в табл. 5.1, а также на базе РОМИП [148].

Reuters-21578 является стандартной базой для исследований в области обработки текстовой информации. Содержит тексты новостей агентства Reuters, среди которых много как приближенных (укороченные версии развернутых новостей, данные о котировках на биржах, отличающиеся датами и числами в

Таблица 5.1

Характеристики использованных текстовых баз (при применении С-функции `isalpha()`)

Коллекция	Максимальная длина текстов	Минимальная длина текстов	Медиана длины текстов	Всего текстов
Reuters-21578	8316	13	519	21578
BNC	2494232	106	98250	4054
РОМИП	1719491	0	1212	780266

тексте, и т.п.), так и полных дубликатов.

British National Corpus (BNC) – большая база текстов современного письменного и разговорного английского языка. Представляет собой «золотой стандарт» и источник сведений о «правильном» английском языке (например, по ней вычисляются вероятности префиксов, окончаний, слов, для использования в разных задачах). «Теоретически» дубликатов в BNC быть не должно, поскольку дубликаты портят статистику «правильного» языка.

Использовалась предварительная фильтрация текстов баз Reuters-21578 и BNC, оставляющая только символы и цифры (C-функция `isalnum()`) или дополнительно еще знаки пунктации ((C-функция `ispunct()`)). Заголовки новостных текстов включались в текст, а прописные буквы заменялись на строчные. Короткие тексты, длина которых меньше длины обрезки n, дополнялись до указанной длины одинаковым специальным символом в конце.

Качество поиска дубликатов для веб-документов исследовалось на стандартной базе «Дубли Web-страниц коллекции РОМИП»[1] [134], содержащей список более 10 млн. пар веб-страниц, входящих в базу РОМИП [148], сходство между которыми по значению функции `String::Similarity` не менее 0.85. База РОМИП является случайной выборкой 3% сайтов из домена `narod.ru`, что составляет 0.12%-0.30% от размера Рунета. Общий размер базы составляет 1.5Гб в архиве и включает около 800 тысяч веб-документов с более чем 23000 сайтов. База содержит множество типов сайтов, однако в ней больше мусора, чем в среднем по русской части Интернет, популярны типовые шаблоны оформления страниц, нет ряда категорий сайтов (например, корпоративных), а граф ссылок отличается от Веб [135].

Предварительная обработка базы включала удаление html-разметки, символов пунктуации, лишних пробелов, заглавные буквы заменялись маленькими, документы, состоящие из менее, чем 20 слов удалялись – в итоге документы набора содержали лишь символы латинского и кириллического алфавитов, а также цифры.

[1] Коллекция предоставлена компанией «Яндекс» [136].

5.2.1.2. Методика поиска дубликатов

Использовался метод поиска приближенно ближайших строк, описанный в п. 4.3.3. При поиске дубликатов последовательно проходятся все тексты коллекции, которые и служат запросами к LSH-лесу, где запомнены их распределенные представления. Запрос выполнялся для двух лесов, базы P_1 и P_2 которых состояли из текстов, длины которых попадали в оба интервала, «накрывающие» значение длины запроса (п. 4.3.3). Длины текстов (и запроса) в каждом интервале выравнивались добавлением специальных символов в конце текста до значения максимальной длины в каждом интервале (п. 4.3.1).

Был проведен ряд экспериментов для значений $L = 1, 2, 5, 10$, $K_{max} = 1, 5, 10, 25$. Дубликатами считались строки, совпавшие на максимальном уровне дерева. Если рекомендуемое теорией значение K (см. выражение (1.15)) для данного кластера было меньше, чем K_{max}, то для сигнализации дубликата было достаточно совпадения на уровне K.

Для поиска дубликатов в Reuters-21578 и BNC было принято, что дубликатами будут считаться тексты, у которых имеется хотя бы одно совпадение хеш-векторов длиной K. Находилось количество дубликатов в зависимости от длины обрезки текста до $n = 100, 150, 250, 500, 1000, 2000$ символов и без обрезки (выравнивание по максимальной длине, условно обозначаемое $n = 0$). Использовались следующие параметры схемы LSH: количество деревьев $L = 1, 5$; размерности хеш-векторов $K = 1, 2, 5, 10, 25, 50, 100, 150, 200$.

5.2.1.3. Результаты

Найденное количество приближенных дубликатов в зависимости от значений K и n, для $L = 1, 5$ приведено для Reuters-21578 в табл. 5.2 (использовалась C-функция `isalpha()`) и в табл. 5.3 (для предварительной фильтрации использовалась C-функция `isalpha()+ispunct()`, а для BNC – соответственно в табл. 5.4 и 5.5

Число приближенных дубликатов ожидаемо уменьшается при увеличении K, стабилизируясь при больших K на значениях, примерно соответствующих количеству дубликатов, найденных в работе [99] другим методом (320 дуб-

Таблица 5.2

Количество найденных приближенных дубликатов в коллекции Reuters-21578 в зависимости от значений n и K, при $L = 1$ и $L = 5$ и использовании C-функции `isalnum()`

K / $L = 1$	$n = 100$	$n = 150$	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 2000$	$n = 0$
1	21347	21334	21281	21224	21206	21213	21275	21458
2	20310	20183	19761	19312	19259	19271	19898	21442
5	8715	7780	5670	5450	7507	10244	15595	20725
10	409	379	806	2124	3273	4721	11109	19199
25	371	350	329	553	1504	1875	4280	17533
50	371	349	325	376	990	1444	3409	15227
100	370	346	322	305	268	271	584	4960
150	370	346	322	305	267	262	479	4823
200	369	346	322	305	267	261	264	2748

K / $L = 5$	$n = 100$	$n = 150$	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 2000$	$n = 0$
1	20709	20637	19326	21207	20860	21015	21034	21575
2	19531	19185	19439	19676	19094	20157	20384	21453
5	15513	14532	11858	9593	10393	12436	16536	20658
10	973	1251	2418	4552	6442	8407	14815	20664
25	689	629	615	1832	3168	3769	7238	17401
50	674	600	523	672	1459	2245	4266	14372
100	666	595	513	487	462	505	1592	6497
150	662	592	513	474	437	447	1253	5625
200	661	591	511	467	429	422	591	3499

Таблица 5.3

Количество найденных приближенных дубликатов в коллекции Reuters-21578 в зависимости от значений n и K, при $L = 1$ и $L = 5$ и использовании `isalnum()` и `ispunct()`

K / $L = 1$	$n = 100$	$n = 150$	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 2000$	$n = 0$
1	21354	21324	21268	21234	21218	21198	21237	21445
2	20277	20119	19748	19350	19231	19214	19844	21429
5	8601	7635	5376	5317	6971	9783	15457	20742
10	403	361	499	1812	2983	4330	10787	19279
25	372	344	318	363	1062	1643	4047	17755
50	369	342	315	312	538	1158	3048	15702
100	369	342	315	298	264	255	469	4920
150	367	342	315	298	263	255	344	4787
200	367	342	315	298	263	254	242	2161

K / $L = 5$	$n = 100$	$n = 150$	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 2000$	$n = 0$
1	21115	18540	18485	20085	19897	16501	21340	21576
2	20267	19719	20079	19248	19915	19625	20670	21453
5	15515	14647	11682	8998	9701	11786	15932	20870
10	1456	877	1629	3580	5523	7836	14648	20614
25	765	610	520	1256	2324	3066	6355	17276
50	743	593	490	472	797	1530	3262	14254
100	721	587	486	435	405	410	943	5806
150	713	588	487	436	397	394	640	4930
200	706	585	486	435	394	384	402	2768

ликатов). Большое число приближенных дубликатов в случае без использования обрезки по фиксированной длине (и их увеличение для эксперимента с `isalnum()` для $n = 2000$) объясняется большим сходством большинства строк, так как к ним добавлялись одинаковые символы для выравнивания длины. При увеличении L количество дубликатов также увеличивается, поскольку увеличивается вероятность совпадения K элементов хотя бы у одного дерева. При «визуальной» проверке, однако, некоторые дубликаты оказались точными. Уменьшение количества дубликатов при увеличении длины обрезки для $K \geqslant 10$ при визуальной проверке показало, что оно вызвано мелкими опечатками в текстах. При меньших значениях K эти опечатки могли не вызывать несовпадений хеш-векторов.

Таблица 5.4

Количество найденных приближенных дубликатов в коллекции BNC в зависимости от значений n и K, при $L = 1$ и $L = 5$ и использовании C-функции `isalnum()`

K $L = 1$	$n = 100$	$n = 150$	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 2000$	$n = 0$
1	3943	3933	3915	3902	3888	3857	3832	4027
2	3545	3470	3346	3203	3044	2941	2619	4027
5	895	668	297	84	39	31	15	3523
10	10	9	9	8	8	9	9	3447
25	9	9	9	8	8	8	7	3403
50	9	9	9	8	8	8	7	2421
100	9	9	9	8	8	8	7	1263
150	9	9	9	8	8	8	7	1263
200	9	9	9	8	8	8	7	246

K $L = 5$	$n = 100$	$n = 150$	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 2000$	$n = 0$
1	3587	3867	3782	3680	3543	3706	3645	4052
2	3618	3716	3613	3374	3503	3489	3349	4037
5	2187	1855	1020	378	168	81	37	4008
10	12	10	9	8	8	10	18	3997
25	9	9	9	8	8	8	9	3493
50	9	9	9	8	8	8	7	2430
100	9	9	9	8	8	8	7	1738
150	9	9	9	8	8	8	7	-
200	9	9	9	8	8	8	7	-

Таблица 5.5

Количество найденных приближенных дубликатов в коллекции BNC в зависимости от значений n и K, при $L = 1$ и $L = 5$ и использовании `isalnum()` и `ispunct()`

K $L = 1$	$n = 100$	$n = 150$	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 2000$	$n = 0$
1	3939	3928	3921	3902	3884	3876	3833	4028
2	3567	3476	3349	3204	3027	2953	2635	4027
5	921	706	293	79	35	25	14	3559
10	10	10	9	8	7	7	7	3487
25	9	9	9	8	7	7	6	3447
50	9	9	9	8	7	7	6	2413
100	9	9	9	8	7	7	6	1267
150	9	9	9	8	7	7	6	1267
200	9	9	9	8	7	7	6	236

K $L = 5$	$n = 100$	$n = 150$	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 2000$	$n = 0$
1	3962	3805	3752	3376	3684	3637	3705	4052
2	3426	3627	3582	3568	3454	3415	3342	4034
5	2203	1915	1067	355	147	71	34	4009
10	13	13	9	8	7	8	17	3997
25	9	9	9	8	7	7	6	3498
50	9	9	9	8	7	7	6	2426
100	9	9	9	8	7	7	6	1751
150	9	9	9	8	7	7	6	-
200	9	9	9	8	7	7	6	-

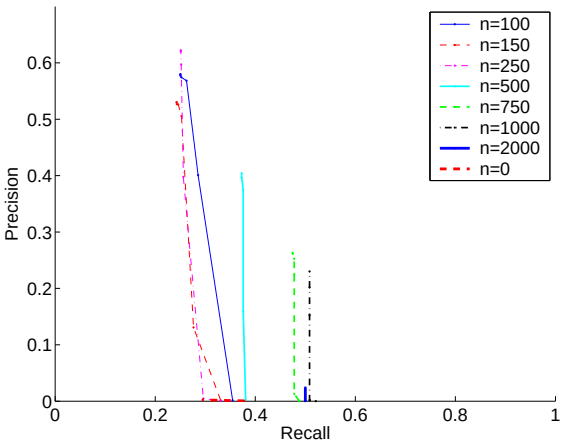
Время поиска всех дубликатов равно времени обхода всех листьев в LSH-дереве и не превышало 0.2 секунды на стандартном PC AMD Athlon XP 2600 с 1.5Гб памяти.

Результаты поиска сравнивались с результатами метода детерминированного вложения из [10]. За «золотой стандарт» было выбрано определение пары дубликатов, как такой, на которой значение функции языка PERL `String::Similarity` (основанной на алгоритме Майерса [81] вычисления расстояния редактирования) не меньше 0.85 (такой же подход применялся для создания коллекции дубликатов в [134]).

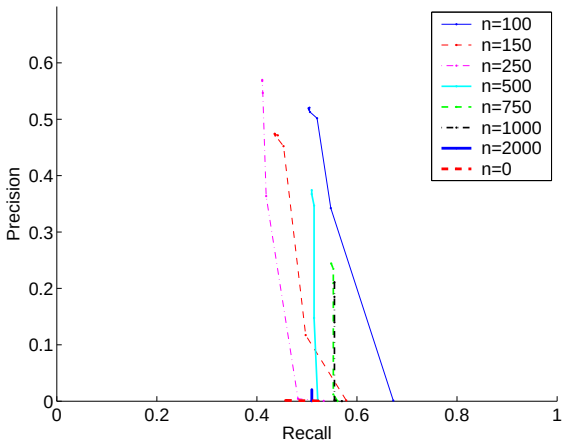
Обозначим как T_{sim} множество пар текстов со значением функции `String::Similarity`, большим или равным sim. Принимая поочередно за «на-

стоящие дубликаты» множества $T_{0.85}, T_{0.95}, T_{0.97}, T_{0.99}$, качество работы нашего метода оценивалось с помощью графиков точность-полнота путем изменения значения K (использовались $K = 5, 10, 25, 50, 100, 150, 200$). Результаты приведены на рис. 5.4.

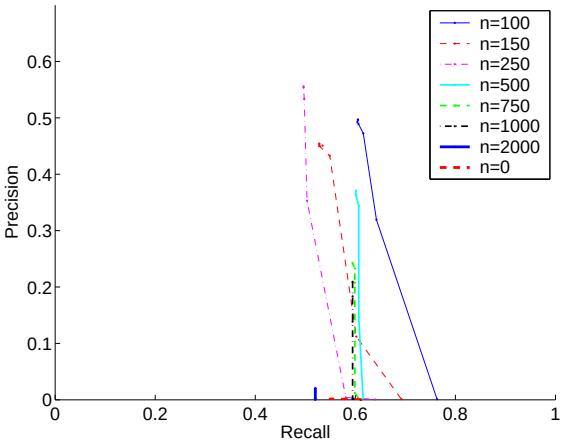
Как видно, при увеличении порога sim на значение функции `String::Similarity` полнота увеличивается, т.е. все большая доля «правильных» дубликатов попадает в мультимножество $|S|$, достигая единицы при $sim = 1$ (т.е. когда дубликатами считаются только полные дубликаты). Уменьшение точности при увеличении длины обрезки n объясняется более частым «срабатыванием» метода на большем количестве специальных символов, с помощью которых выравнивалась длина текстов.



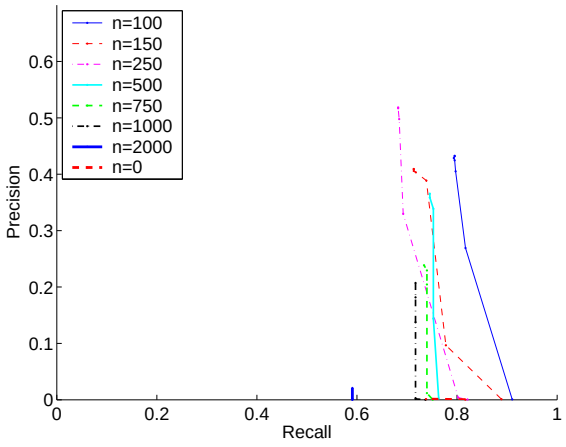
(a) $sim = 0.85$



(b) $sim = 0.95$



(c) $sim = 0.97$

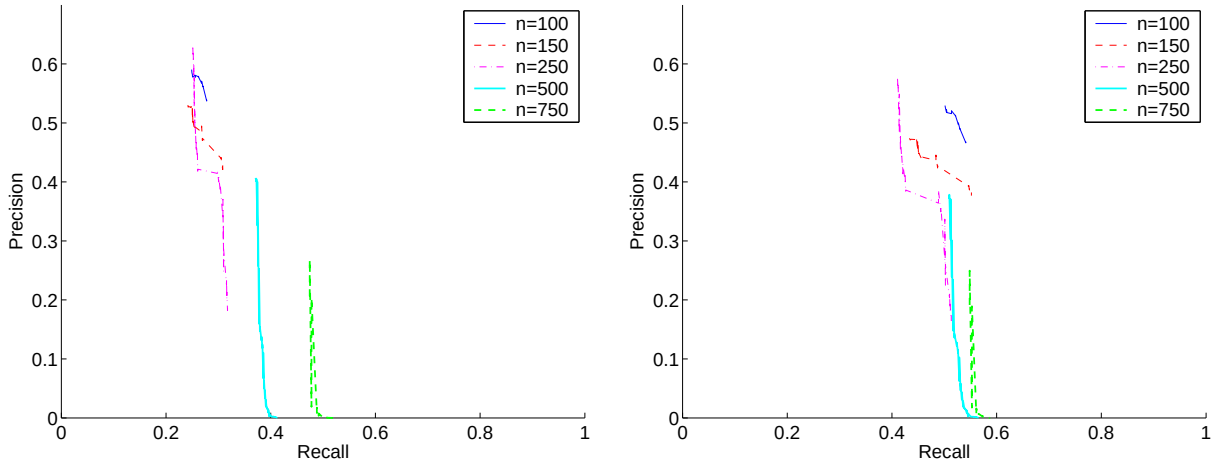


(d) $sim = 0.99$

Рис. 5.4. Зависимость точности от полноты поиска, полученная разработанными методами для значений $sim = 0.85, 0.95, 0.97, 0.99$ при $L = 5$

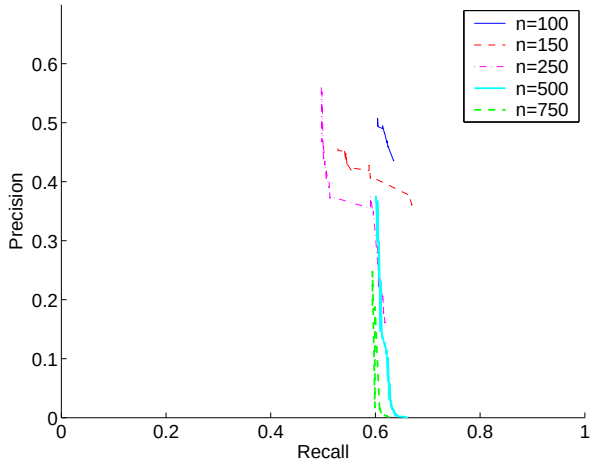
Аналогичные графики (рис. 5.5) были построены для метода детерминированного вложения, описанного в работе [10] (BY), для длины обрезок $n = 100, 150, 250, 500, 750$ (для бóльших значений n не удалось получить результатов за приемлемое время). Для построения графиков изменялся порог на расстояние Хемминга между полученными векторами, указывающий, что считать дубликатом. Проверялись только тексты, где расстояние Хемминга не превышало 15% от максимально возможного для данной длины n.

Для сравнения пар значений точность-полнота использовалась интегральная оценка – F-мера с $\alpha = 1$, определяемая как $F_\alpha = (1+\alpha)rp/(\alpha p + r)$ [116], где r – полнота, p – точность. Для каждой из кривых, изображенных на рис. 5.4

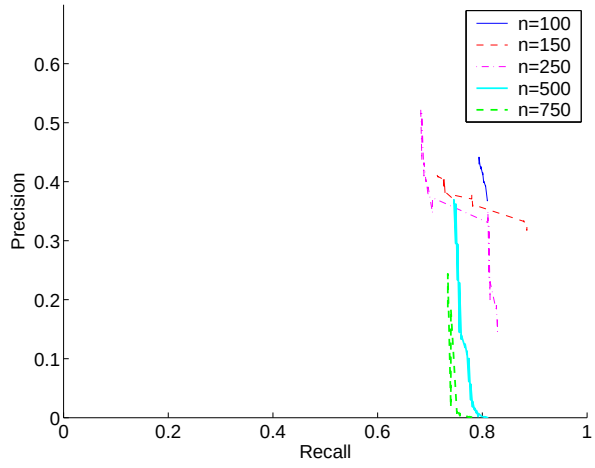


(a) $sim = 0.85$

(b) $sim = 0.95$



(c) $sim = 0.97$

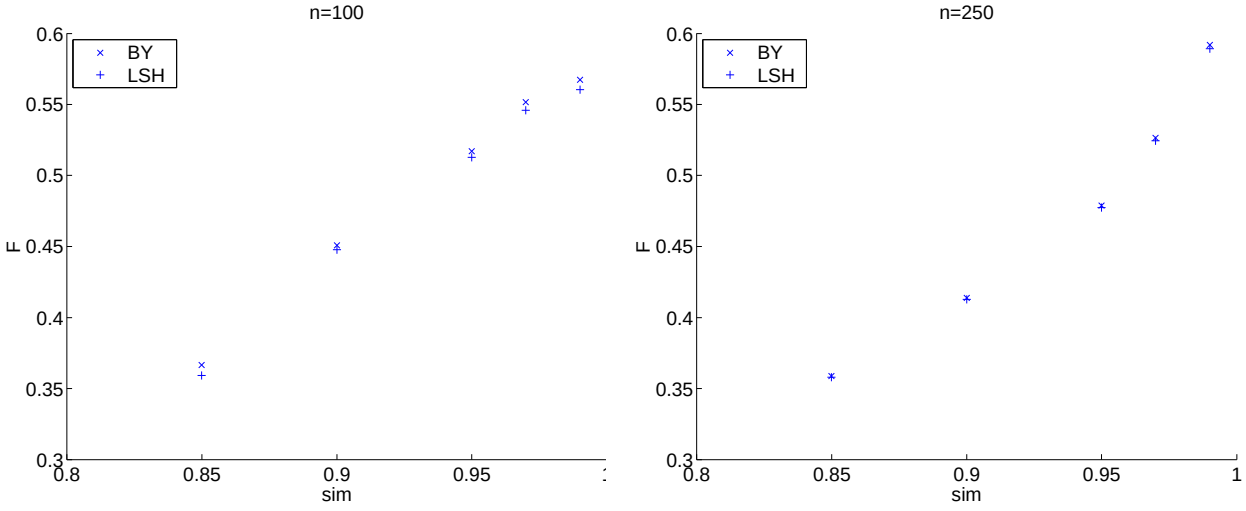


(d) $sim = 0.99$

Рис. 5.5. Зависимость точности от полноты поиска, полученная методом Бар-Йозефа для значений $sim = 0.85, 0.95, 0.97, 0.99$ при $L = 5$

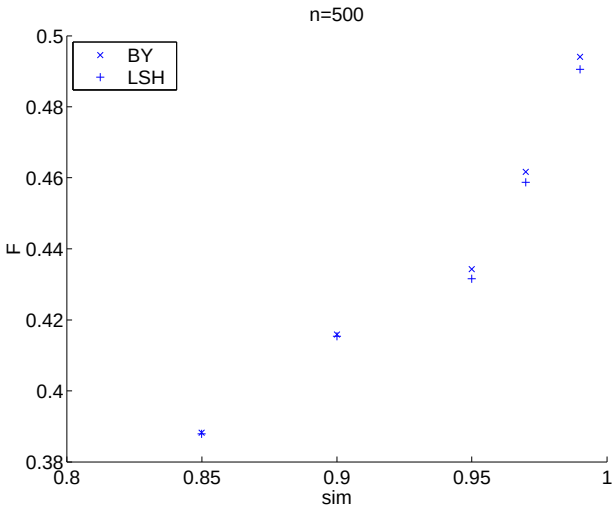
и 5.5, было подсчитано максимальное значение F_{1max}, достигаемое на точках этих кривых, которое для $n = 100, 250, 500, 750$ изображено на рис. 5.6.

Результаты показывают отсутствие существенных отличий в качестве между двумя сравниваемыми методами относительно «золотого стандарта», однако время решения задачи существенно отличается. Порядок времени поиска дубликатов на всей коллекции с учетом построения деревьев в методе, основанном на применении LSH-леса, составляет минуты, тогда как применении детерминированного метода Бар-Йозефа (BY) из [10] – от часов до дней.

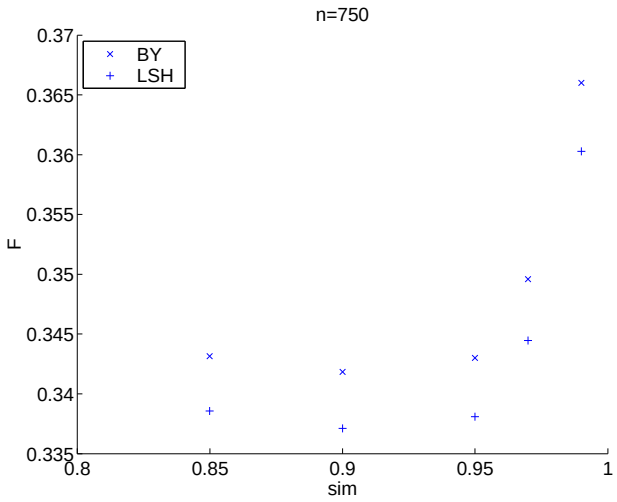


(a) $n = 100$

(b) $n = 250$



(c) $n = 500$



(d) $n = 750$

Рис. 5.6. Значения F_{1max} для $n = 100, 250, 500, 750$. Точки (кресты), обозначенные в легенде как BY, соответствуют методу Бар-Йозефа, плюсы – разработанному поиска с помощью LSH-леса

Таблица 5.6

Результаты поиска дубликатов в базе РОМИП для $K = 5$

sim	L	ε	дубликатов		точность	полнота	F_1
			найденых	из них действительных			
0.85	1	0.005	4985427	3830904	0.768	0.376	0.505
		0.01	5139157	3948794	0.768	0.387	0.515
		0.05	5248143	3972392	0.757	0.389	0.514
		0.1	5156946	3777558	0.733	0.370	0.492
		0.2	5347664	3942282	0.737	0.386	0.507
		0.3	6568104	4988581	0.760	0.489	0.595
	2	0.005	6420701	5237682	0.816	0.513	0.630
		0.01	5271275	4047322	0.768	0.397	0.523
		0.05	5412778	4093720	0.756	0.401	0.524
		0.1	5875172	4455863	0.758	0.437	0.554
		0.2	6469246	4902982	0.758	0.481	0.588
		0.3	6084588	4066070	0.668	0.399	0.499
	5	0.005	6892482	5632544	0.817	0.552	0.659
		0.01	6087416	4779835	0.785	0.469	0.587
		0.05	7451718	5839771	0.784	0.572	0.662
		0.1	6685008	4787036	0.716	0.469	0.567
		0.2	8157607	5770751	0.707	0.566	0.629
		0.3	10637716	5582452	0.525	0.547	0.536
	10	0.005	6788055	5419816	0.798	0.531	0.638
		0.01	6994610	5569043	0.796	0.546	0.648
		0.05	7583605	5697767	0.751	0.559	0.641
		0.1	8320132	5783167	0.695	0.567	0.624
		0.2	10031992	6153251	0.613	0.603	0.608
		0.3	12748666	5981496	0.469	0.586	0.521
0.90	1	0.005	4985427	3828085	0.768	0.472	0.584
		0.01	5139157	3945978	0.768	0.486	0.595
		0.05	5248143	3968774	0.756	0.489	0.594
		0.1	5156946	3773058	0.732	0.465	0.569
		0.2	5347664	3933411	0.736	0.485	0.584
		0.3	6568104	4977487	0.758	0.613	0.678
	2	0.005	6420701	5233834	0.815	0.645	0.720
		0.01	5271275	4041883	0.767	0.498	0.604
		0.05	5412778	4089536	0.756	0.504	0.605
		0.1	5875172	4444390	0.756	0.548	0.635
		0.2	6469246	4889954	0.756	0.603	0.671
		0.3	6084588	4048459	0.665	0.499	0.570
	5	0.005	6892482	5625450	0.816	0.693	0.750
		0.01	6087416	4772726	0.784	0.588	0.672
		0.05	7451718	5828493	0.782	0.718	0.749
		0.1	6685008	4770168	0.714	0.588	0.645
		0.2	8157607	5747202	0.705	0.708	0.706
		0.3	10637716	5561177	0.523	0.685	0.593
	10	0.005	6788055	5410453	0.797	0.667	0.726
		0.01	6994610	5560190	0.795	0.685	0.736
		0.05	7583605	5682726	0.749	0.700	0.724
		0.1	8320132	5763061	0.693	0.710	0.701
		0.2	10031992	6121904	0.610	0.754	0.675
		0.3	12748666	5950054	0.467	0.733	0.570
0.95	1	0.005	4985427	3819310	0.766	0.567	0.652
		0.01	5139157	3937318	0.766	0.585	0.663
		0.05	5248143	3944431	0.752	0.586	0.658
		0.1	5156946	3728857	0.723	0.554	0.627
		0.2	5347664	3895280	0.728	0.578	0.645
		0.3	6568104	4936204	0.752	0.733	0.742
	2	0.005	6420701	5218269	0.813	0.775	0.793
		0.01	5271275	4021834	0.763	0.597	0.670
		0.05	5412778	4063162	0.751	0.603	0.669
		0.1	5875172	4389525	0.747	0.652	0.696
		0.2	6469246	4827277	0.746	0.717	0.731
		0.3	6084588	3994770	0.657	0.593	0.623
	5	0.005	6892482	5596560	0.812	0.831	0.821
		0.01	6087416	4731374	0.777	0.702	0.738
		0.05	7451718	5782190	0.776	0.858	0.815
		0.1	6685008	4721373	0.706	0.701	0.704
		0.2	8157607	5647986	0.692	0.838	0.758
		0.3	10637716	5490671	0.516	0.815	0.632
	10	0.005	6788055	5374122	0.792	0.798	0.795
		0.01	6994610	5526002	0.790	0.820	0.805
		0.05	7583605	5587554	0.737	0.829	0.780
		0.1	8320132	5649969	0.679	0.839	0.751
		0.2	10031992	5951911	0.593	0.884	0.710
		0.3	12748666	5831647	0.457	0.866	0.599
1.00	1	0.005	4985427	2879287	0.578	0.934	0.714
		0.01	5139157	2876899	0.560	0.933	0.700
		0.05	5248143	2893800	0.551	0.938	0.695
		0.1	5156946	2896820	0.562	0.939	0.703
		0.2	5347664	2850398	0.533	0.924	0.676
		0.3	6568104	2899630	0.441	0.940	0.601
	2	0.005	6420701	2947543	0.459	0.956	0.620
		0.01	5271275	2866858	0.544	0.930	0.686
		0.05	5412778	2883717	0.533	0.935	0.679
		0.1	5875172	2951605	0.502	0.957	0.659
		0.2	6469246	2962112	0.458	0.961	0.620
		0.3	6084588	2944994	0.484	0.955	0.642
	5	0.005	6892482	2969869	0.431	0.963	0.595
		0.01	6087416	2952613	0.485	0.957	0.644
		0.05	7451718	3031310	0.407	0.983	0.575
		0.1	6685008	2965239	0.444	0.962	0.607
		0.2	8157607	3004195	0.368	0.974	0.534
		0.3	10637716	2974400	0.280	0.965	0.434
	10	0.005	6788055	2979851	0.439	0.966	0.604
		0.01	6994610	3027923	0.433	0.982	0.601
		0.05	7583605	2993147	0.395	0.971	0.561
		0.1	8320132	2994701	0.360	0.971	0.525
		0.2	10031992	3037549	0.303	0.985	0.463
		0.3	12748666	3026745	0.237	0.981	0.382

Результаты поиска дубликатов в базе РОМИП представлены в таблицах 5.6 и 5.7, где приведены значения точности, полноты и F-меры для значений порога на значение $sim = 0.85, 0.90, 0.95, 1.00$; значений $\varepsilon = 0.005, 0.01, 0.05, 0.1, 0.2, 0.3$ и $L = 1, 2, 5, 10$, $K = 5, 10, 25, 50, 100, 150$.

В работе [128] максимальное значение F-меры, полученное с помощью

Таблица 5.7

Результаты поиска дубликатов в базе РОМИП для $K = 10$

sim	L	ε	дубликатов		точность	полнота	F_1
			найденых	из них действительных			
0.85	1	0.005	3908801	3641183	0.932	0.357	0.516
		0.01	3899047	3603716	0.924	0.353	0.511
		0.05	4765682	3608695	0.757	0.354	0.482
		0.1	4779527	3594699	0.752	0.352	0.480
		0.2	4271999	3105982	0.727	0.304	0.429
		0.3	4209123	3044360	0.723	0.298	0.423
	2	0.005	3967556	3699197	0.932	0.363	0.522
		0.01	3349999	3048948	0.910	0.299	0.450
		0.05	6252955	5088332	0.814	0.499	0.618
		0.1	4965640	3796809	0.765	0.372	0.501
		0.2	5264461	4084909	0.776	0.400	0.528
		0.3	5220525	3960655	0.759	0.388	0.514
	5	0.005	5315658	4927579	0.927	0.483	0.635
		0.01	4059110	3722767	0.917	0.365	0.522
		0.05	4996663	3821504	0.765	0.375	0.503
		0.1	6373136	5185962	0.814	0.508	0.626
		0.2	5784287	4579077	0.792	0.449	0.573
		0.3	5521622	4338048	0.786	0.425	0.552
0.90	1	0.005	3908801	3638909	0.931	0.448	0.605
		0.01	3899047	3602530	0.924	0.444	0.600
		0.05	4765682	3607510	0.757	0.445	0.560
		0.1	4779527	3592388	0.752	0.443	0.557
		0.2	4271999	3101496	0.726	0.382	0.501
		0.3	4209123	3041467	0.723	0.375	0.494
	2	0.005	3967556	3697191	0.932	0.456	0.612
		0.01	3349999	3046697	0.909	0.375	0.531
		0.05	6252955	5085666	0.813	0.627	0.708
		0.1	4965640	3792318	0.764	0.467	0.580
		0.2	5264461	4080445	0.775	0.503	0.610
		0.3	5220525	3955228	0.758	0.487	0.593
	5	0.005	5315658	4924911	0.926	0.607	0.733
		0.01	4059110	3719249	0.916	0.458	0.611
		0.05	4996663	3818905	0.764	0.471	0.582
		0.1	6373136	5179700	0.813	0.638	0.715
		0.2	5784287	4570818	0.790	0.563	0.658
		0.3	5521622	4329089	0.784	0.533	0.635
0.95	1	0.005	3908801	3632656	0.929	0.539	0.683
		0.01	3899047	3597985	0.923	0.534	0.677
		0.05	4765682	3599723	0.755	0.534	0.626
		0.1	4779527	3580533	0.749	0.532	0.622
		0.2	4271999	3068607	0.718	0.456	0.558
		0.3	4209123	3032952	0.721	0.450	0.554
	2	0.005	3967556	3691524	0.930	0.548	0.690
		0.01	3349999	3038125	0.907	0.451	0.602
		0.05	6252955	5075246	0.812	0.753	0.781
		0.1	4965640	3762915	0.758	0.559	0.643
		0.2	5264461	4067244	0.773	0.604	0.678
		0.3	5220525	3929345	0.753	0.583	0.657
	5	0.005	5315658	4915249	0.925	0.730	0.816
		0.01	4059110	3709104	0.914	0.551	0.687
		0.05	4996663	3804695	0.761	0.565	0.649
		0.1	6373136	5135753	0.806	0.762	0.784
		0.2	5784287	4521985	0.782	0.671	0.722
		0.3	5521622	4289446	0.777	0.637	0.700
1.00	1	0.005	3908801	2854854	0.730	0.926	0.817
		0.01	3899047	2825887	0.725	0.916	0.809
		0.05	4765682	2807878	0.589	0.911	0.715
		0.1	4779527	2811872	0.588	0.912	0.715
		0.2	4271999	2861305	0.670	0.928	0.778
		0.3	4209123	2847887	0.677	0.923	0.781
	2	0.005	3967556	2847639	0.718	0.923	0.808
		0.01	3349999	2834088	0.846	0.919	0.881
		0.05	6252955	2857195	0.457	0.926	0.612
		0.1	4965640	2847827	0.574	0.923	0.708
		0.2	5264461	2825232	0.537	0.916	0.677
		0.3	5220525	2851299	0.546	0.925	0.687
	5	0.005	5315658	2899309	0.545	0.940	0.690
		0.01	4059110	2882741	0.710	0.935	0.807
		0.05	4996663	2869433	0.574	0.930	0.710
		0.1	6373136	2871701	0.451	0.931	0.607
		0.2	5784287	3024541	0.523	0.981	0.682
		0.3	5521622	2890528	0.523	0.937	0.672

подхода на основе поиска замкнутых множеств признаков, на части коллекции РОМИП достигало 0.49. В работе [124] на основании методов, использующих статистическую информацию (спецсимволы, длины элементов, наборы популярных слов) для порогов $0.85, 0.90, 0.95, 1.00$ были получены значения F-меры, соответственно, $0.66, 0.75, 0.82, 0.89$. Как видно из табл. 5.6, 5.7, метод поиска дубликатов на основании поиска приближенно ближайших строк показывает результаты, превышающие полученные в [128, 124]. Также отметим, что, в отличие от экспоненциального по $|P|$ метода [128], исследовавшего лишь до 50% всей базы, вычислительная сложность предложенного метода позволила исследовать его на полной коллекции РОМИП.

5.2.2. Обнаружение спама в электронных сообщениях

Обнаружение почтового спама (сообщений, как правило, рекламного характера, массово рассылаемых людям, не выразившим желание их получать) также является актуальной задачей, поскольку количество спама среди всей почты у среднего пользователя в 2005 году уже достигало 80-85% [39] и увеличивается. Спам обнаруживают различными способами – в основном, это проверки на специфические особенности спам-писем, такие как нахождение адреса адресанта в черном списке, экстенсивное применение разметки HTML в письме, подозрительные (графические) вложения, а также использование байесовской классификации [87] на основе вероятностей специфических слов.

Одной из распространенных спам-технологий, позволяющих спамерам избежать простейших частотных фильтров, является внесение изменений в текст письма, таких как специфическое написание слов в письмах, которые перестают быть точными копиями друг друга, а также искажают картину вероятностей слов или препятствуют предварительной обработке писем фильтрами [48]. Для борьбы с подобными технологиями некоторые исследователи [142, 68, 141] предлагают использовать сравнение писем с ранее сохраненными, поскольку спам по своей природе имеет массовый характер, а рассылаемая реклама одинакова для тысяч получателей.

Мы применили этот подход на основе примеров, чтобы оценить, какое количество спама может быть обнаружено только лишь с помощью сравнения с ранее поступившими (спам) письмами, без применения специфических знаний и подробного анализа спам-технологий.

5.2.2.1. Базы данных для экспериментального исследования

Были использованы тестовые базы, широко применяющиеся разработчиками систем обнаружения спама для тестирования своих продуктов: TREC 2005 Spam Track [26] и TREC 2006 Spam Track [25]. Обе базы содержат почтовые сообщения, размеченные экспертами на два класса: «спам» и «не спам». TREC 2005 Spam Track содержит $|P| = 92189$ реальных почтовых сообщений объемом 0.8 Гб, из которых 52790 (57%) спамовые. Англоязычная часть TREC 2006 Spam Track содержит $|P| = 37822$ реальных почтовых сообщений, объемом в 189 Мб, из которых 24912 (66%) – спам.

5.2.2.2. Схема экспериментов и оценки эффективности

Коллекции TREC и принятый способ оценки эффективности фильтров на них построены с учетом способа реального использования спам-фильтров у конечных пользователей. Задача тестируемых фильтров – классифицировать подаваемые в хронологическом порядке сообщения и, затем, дообучиться после получения правильной метки для этого сообщения от эксперта. Этот процесс соответствует обычному порядку действий пользователя, когда он каждый раз помечает во входящих сообщениях спам, что позволяет более точно настроить фильтр.

По условиям TREC, каждому письму должен быть присвоен параметр – степень «спамности» письма (score), по которому оно классифицируется: сообщения, получившие score больше определенного порога, считаются спамом, остальные – обычными письмами. Оценка качества работы фильтра проводится по двум основным параметрам – проценту неправильно классифицированных спам-сообщений $sm\%$, т.е. false positives, и проценту неправильно классифицированных неспам-сообщений $hm\%$, т.е. false negatives. Изменением порога на значение score можно построить ROC-кривые (зависимость $sm\%$

от $hm\%$), по которым в TREC сравниваются различные алгоритмы.

Предварительный фильтр оставлял в письмах одни лишь символы алфавита (C-функция `isalpha()`) и обрезал сообщение по размеру $n = 1000$. Подаваемые в хронологическом порядке сообщения служили запросами для процедуры поиска приближенного ближайшего соседа с помощью LSH-леса. Значение L изменялось от 1 до 200. Значение K фиксировалось по формуле (1.15).

На основании экспериментов в пункте 4.4.2 было выбрано два способа присвоения score письмам по уровню в LSH-лесе, к которому принадлежат приближенные ближайшие соседи ко входным письмам. А именно

1. по максимальному уровню $k_{max} = K$;
2. по среднему уровню k_{avg}.

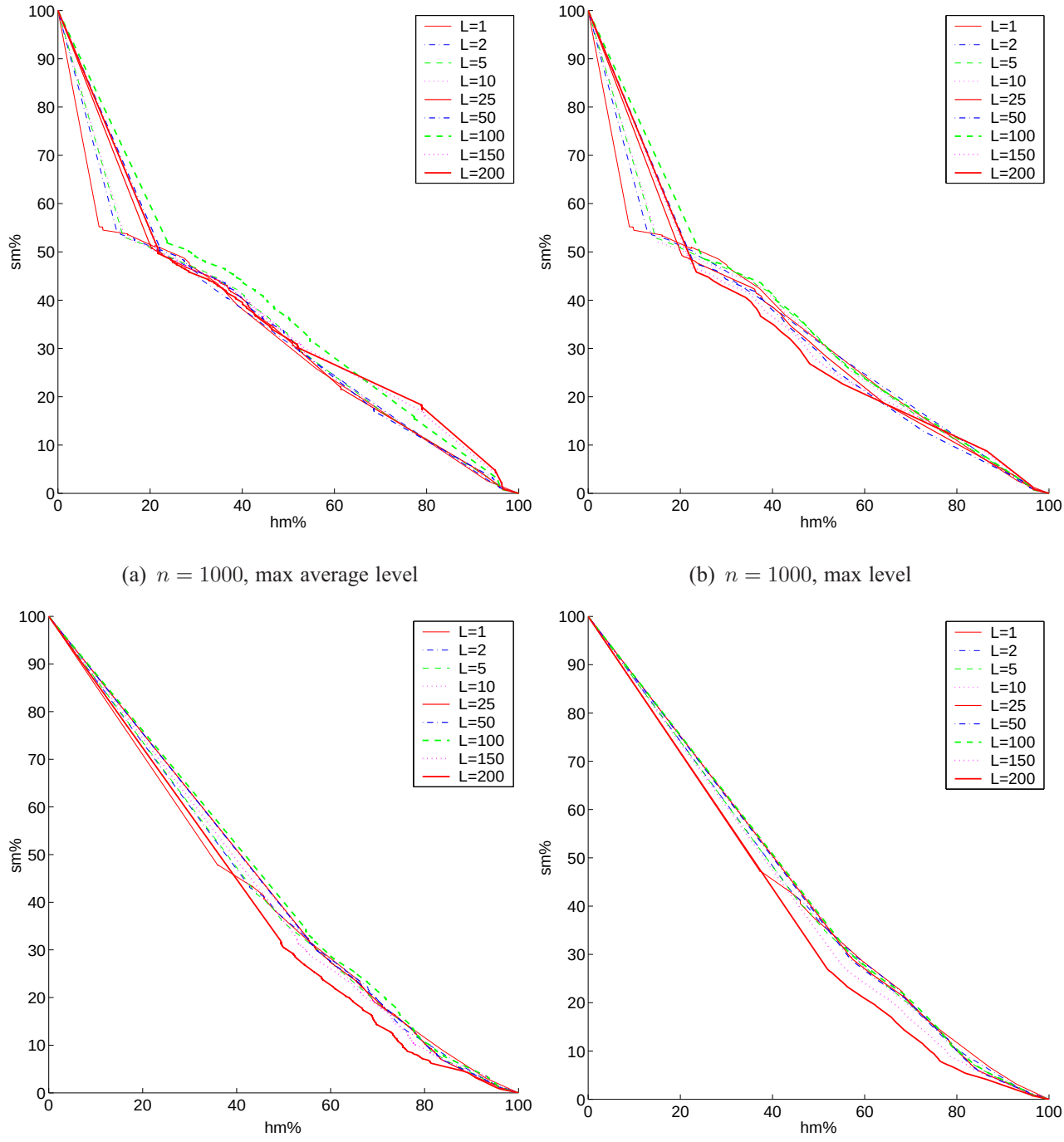
Если сообщение на самом деле имело в коллекции экспертную метку «спам», то оно добавлялось в множество P и на классификацию подавалось следующее сообщение.

5.2.2.3. Результаты

Полученные ROC-кривые изображены на рис. 5.7 и 5.8, откуда видно, что

- для Spam Track 2005 при уровне hm%=10% успешно обнаруживается только 50% спама для $n = 1000$, а при уровне $n = 5000$ спам почти не обнаруживается (практические случайная классификация);
- для Spam Track 2006 результаты значительно выше: при уровне $hm\%$=5-10% успешно обнаруживается примерно 80% спама. Возможно, это связано с большим объемом базы.

Из ROC-кривых на рис. 5.8 видно, что при $L = 1$ значения $hm\%$ зачастую меньше, чем при $L > 1$. Это можно объяснить высокой степенью сходства части легальных писем с спамом, например, из-за использования html-формата, который оставлялся в теле письма, а не убирался и/или не анализировался его код, как это принято в реальных системах обнаружения спама. Для $L = 1$ ROC-кривые для разных значений K приведены на рис. 5.9, откуда видно, что увеличение K приводит к более точной классификации легальных писем.



(a) $n = 1000$, max average level

(b) $n = 1000$, max level

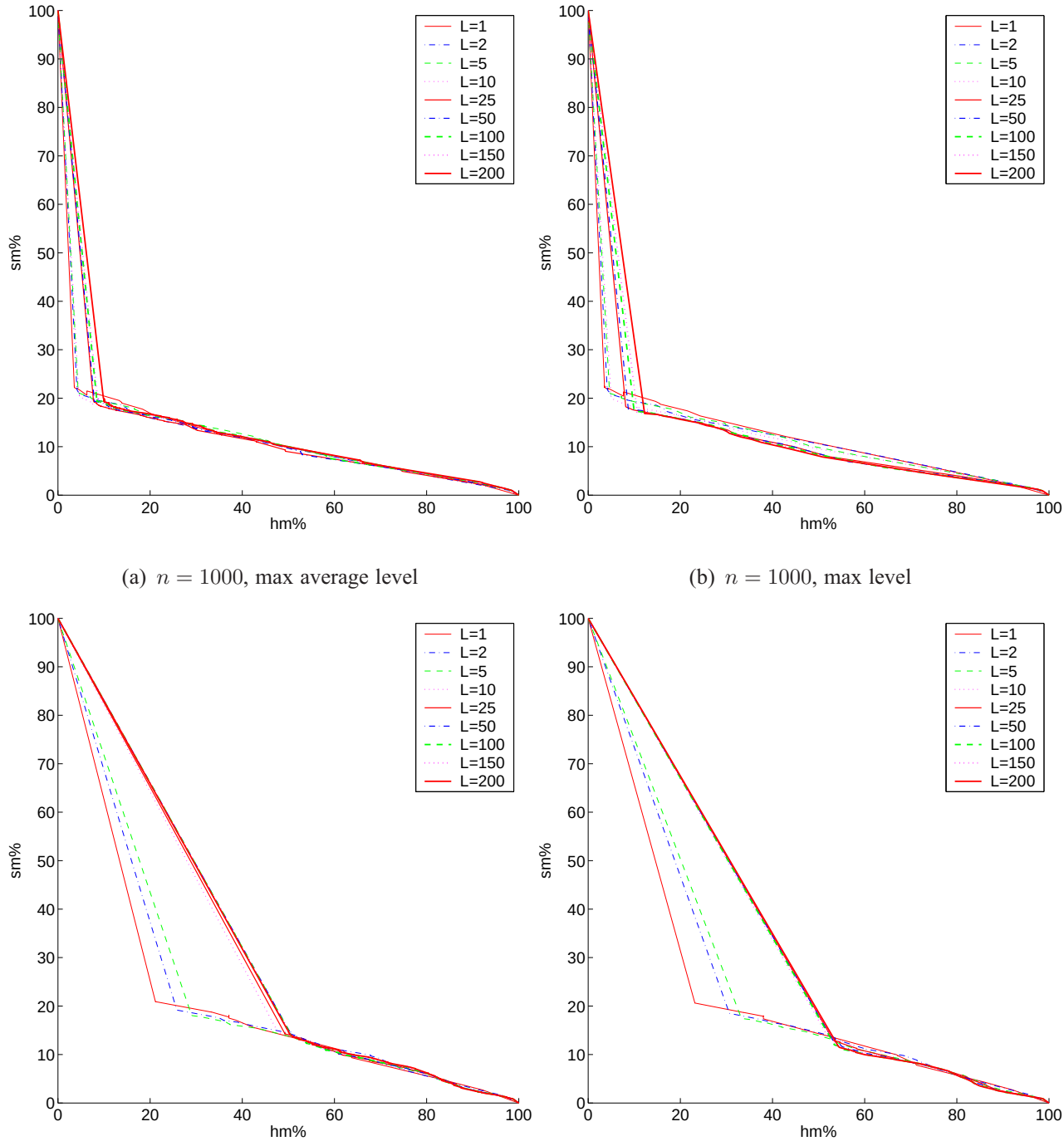
(c) $n = 5000$, max average level

(d) $n = 5000$, max level

Рис. 5.7. Зависимость $sm\%$ от $hm\%$ для коллекции Spam Track 2005 для двух способов присваивания score – по среднему и по максимальному уровню, для $n = 1000$ и $n = 5000$

Возможно, при этом «улавливаются» их более тонкие отличия от спама.

Из проведенных экспериментов видно, что, реализовав идею обнаружения спама по приближенным дубликатам, можно отфильтровать значительную его часть, что позволяет говорить о применимости данного подхода для боль-



(a) $n = 1000$, max average level

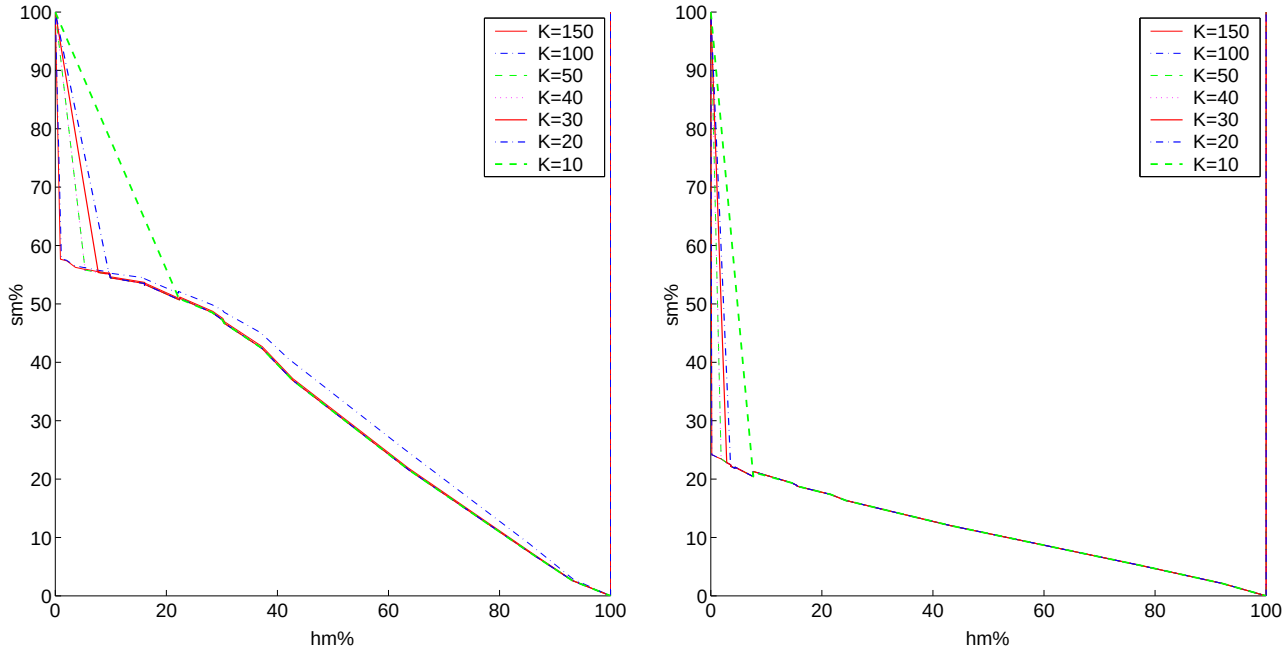
(b) $n = 1000$, max level

(c) $n = 5000$, max average level

(d) $n = 5000$, max level

Рис. 5.8. Зависимость $sm\%$ от $hm\%$ для коллекции Spam Track 2006 для двух способов присваивания score – по среднему и по максимальному уровню, для $n = 1000$ и $n = 5000$

ших почтовых серверов в качестве одного из модулей системы обнаружения спама. Чем централизованнее почтовый сервис и чем большее у него число пользователей, тем больший процент отфильтрованного спама можно ожидать. Так, можно ожидать, что системы, основанные на обнаружении спама как при-



(a) Spam Track 2005 (b) Spam Track 2006

Рис. 5.9. ROC-кривая для разных значений K при $L = 1$ для коллекций Spam Track 2005 и Spam Track 2006

ближенных копий, будут особенно эффективны в больших почтовых сервисах типа Gmail [47]. Поскольку спам всегда направляется огромному числу получателей, на почтовом сервисе обнаружить его намного легче, чем в рамках почтового ящика индивидуального пользователя, куда похожий спам приходит в гораздо более скромных масштабах. Для индивидуальных пользователей подобный подход можно применять, перейдя к кооперативному обнаружению спама. В качестве примера можно привести распределенную систему обнаружения спама Vipul's Razor [117], которая использует сокращенные представления сообщений, отмеченных как спам участниками системы. В качестве таких сокращенных представлений можно рассмотреть использование разработанных распределенных представлений – хеш-векторов.

5.2.3. Поиск генов и гиперчувствительных сайтов в последовательностях ДНК

Поиск новых генов – это фундаментальная задача вычислительной биологии, состоящая в алгоритмизированном поиске биологически функциональных

участков генетических последовательностей. Генами называются определенные подпоследовательности нуклеиновых баз (нуклеотидов) дезоксирибонуклеиновой кислоты (ДНК), которые в процессе транскрипции кодируют 20 различных аминокислот, использующихся для создания протеинов. Гены высших организмов (эукариотов) состоят из несмежных участков – экзонов (по 140 нуклеотидов в среднем). Между экзонами находятся гораздо более длинные некодирующие, «мусорные» последовательности, не относящиеся к генам – интроны [120], у человека суммарно составляющие до 99% длины всего генома [95].

Биологи все мира каждый год генерируют около 10 миллиардов мегабайт данных, а объем базы генов GenBank [44] удваивается каждые 15 месяцев [14]. Поэтому поиск генов невозможен без эффективных вычислительных инструментов анализа большого количества длинных последовательностей. Сходство (гомологичность) генетических последовательностей исследуемых организмов с известными позволяет находить новые гены, а также указывает на их эволюционное родство и функцию. Это обусловливает актуальность эффективного поиска гомологичных генетических последовательностей.

Другой важной задачей является поиск гомологичных некодирующих участков в локус-контролирующих областях (locus control region, LCR) бета-глобина, которые управляют транскрипцией в локусе. Известно, что последовательности бета-глобина содержат хорошо сохранившиеся в процессе эволюции области, называемые гиперчувствительными сайтами ДНКазы [42, 21], что позволяет идентифицировать эти области в бета-глобине исследуемых организмов.

Нуклеотиды можно представить, как алфавит из четырех символов A, G, T и C, а цепи молекулы ДНК – как последовательности таких символов. Степень отличия двух цепей может быть определена как расстояние редактирования между последовательностями составляющих их нуклеотидов.

Для поиска экзонов будем использовать подход рассуждений на основе примеров, где роль примеров играют экзоны последовательностей обучающей выборки. По таким примерам ищутся экзоны в тестовых последовательно-

стях (п. 5.2.3.2), либо участки гиперчувствительных сайтов одного организма (п. 5.2.3.4), по которым необходимо найти такие же в бета-глобине другого организма. Для повышения эффективности поиска используются описанные в разделе 3 методы.

5.2.3.1. Базы данных для экспериментального исследования

Для первого эксперимента по поиску экзонов обучающей базой был выбран набор HMR195, использованный в [93] для тестирования программ обнаружения генов (195 последовательностей с 948 экзонами, средняя длина последовательностей 7096). Тестовой выборкой служила база последовательностей Burset-Guigo из [22] (570 последовательностей, 2649 экзонов). В обеих базах экзоны размечены.

Для второго эксперимента по поиску коротких некодирубщих участков ДНК использовалась LCR-последовательность бета-глобина мыши (длиной 12 Кб, GenBank Z13985) и человека (первые 20 Кб GenBank U01317) из [21]. В качестве обучающей выборки был принят бета-глобин мыши, тестовой – человека.

5.2.3.2. Методика классификации нуклеотидов для поиска экзонов

Цель первого эксперимента – оценить эффективность предложенного метода поиска сходных строк в задаче поиска экзонов в генетических последовательностях позвоночных по известным примерам экзонов других организмов.

Поиск экзонов осуществлялся путем распознавания принадлежности экзону отдельных нуклеотидов. Каждому i-му нуклеотиду всех тестовых последовательностей ставился в соответствие счетчик c_i, изначально инициализированный нулем. Для каждого экзона g_i из обучающей последовательности принималось $P = \{g_i\}$ и строился LSH-лес. Далее все тестовые последовательности y «нарезались» скользящим окном шириной $|g_i|$ (ссылка). Подстроки вида $y[i, i+|g_i|-1]$ поочередно служили запросом к построенному LSH-лесу. Для всех подстрок, совпавших с экзоном g_i на уровне, большем или равном K, с помощью классического алгоритма вычислялось расстояние редактирования. Для тех подстрок, расстояние редактирование которых было минимально

в пределах данной тестовой последовательности, увеличивалось на единицу значение счетчиков, соответствующих нуклеотидам, принадлежащим данным подстрокам. Решение о том, что считать нуклеотидом, принадлежащим экзону, определялось порогом t, отсекающим малые значения c_i.

Чтобы избежать влияния того, что краевые нуклеотиды покрываются меньшим числом окон, предварительно все последовательности из тестовой выборки дополнялись в начале и в конце спецсимволом N (в количестве 2454 символа, что равно длине самого длинного экзона из набора HMR195).

Как интегральная оценка качества поиска экзонов применялась «приближенная корреляция» (approximate correlation, AC) [22], определяемая как

$$AC = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TP}{TP+FP} + \frac{TN}{TN+FP} + \frac{TN}{TN+FN}\right) - 1, \qquad (5.1)$$

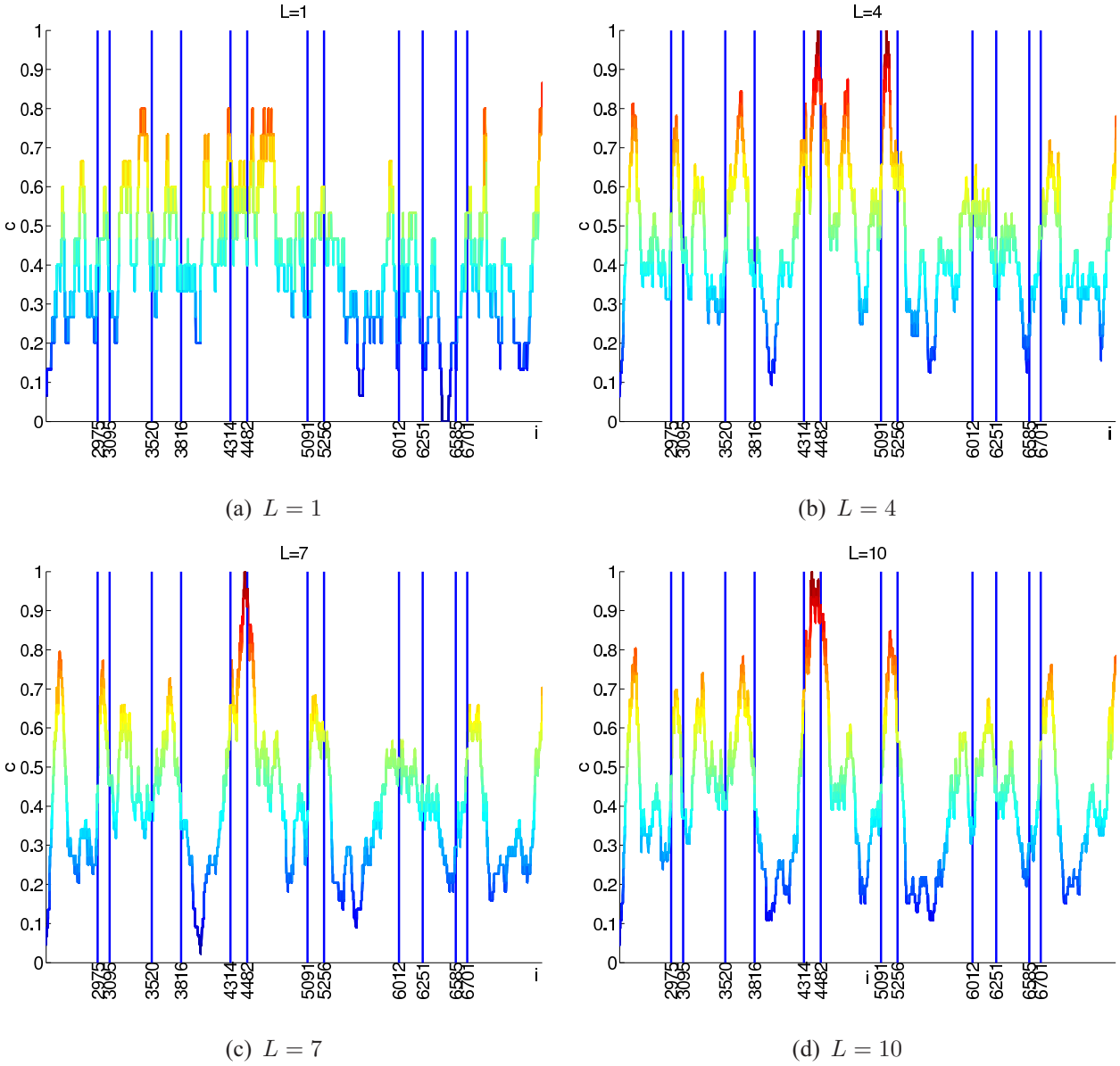
где TP – true positives, TN – true negatives, FP – false positives, FN – false negatives.

На рис. 5.10 изображены графики нормализованных значений c_i, а также реальные границы экзонов для последовательности ACU08131 из набора Burset-Guigo для значений $K = 5$ и $L = 1, 4, 7, 10$. Как видно из рисунков, увеличение L приводит к увеличению разницы между значениями c_i для интронов и экзонов, пики становятся более узкими.

5.2.3.3. Сравнительные результаты экспериментального исследования поиска экзонов

Предложенный метод сравнивался с подходом на основе классического алгоритма редактирования [32]. В последнем лучшие значения AC (5.1), усредненного по всем тестовым последовательностям (AC_{avg}), достигали 0.49. Однако, полученная нами оценка времени работы такого алгоритма на исследованной базе составляет около 6 лет на однопроцессорном компьютере Athlon XP 2600+.

В табл. 5.8 для $K = 4, \ldots, 9$ и $L = 1, \ldots, 10$ приведены полученные описанным методом значения $AC_{max} = \max_t AC_{avg}(t)$, значение AC_{avg} (и его дисперсия σ^2_{AC}), усредненные по всем тестовым последовательностям, а также



(a) $L = 1$ (b) $L = 4$

(c) $L = 7$ (d) $L = 10$

Рис. 5.10. Графики нормализованных значений c_i, а также реальные границы экзонов для последовательности ACU08131 для $K = 5$ и $L = 1, 4, 7, 10$

суммарное время, потраченное на поиск кандидатов с помощью предложенного метода и на проверку полученных кандидатов с помощью классического алгоритма. Как видно из табл. 5.8, метод на основе поиска приближенных ближайших строк с помощью процедуры LSH-лес показывает сравнимые результаты с методом из [32] для небольших значений K, но за гораздо меньшее практически в 750 раз время. Улучшение результатов с одновременным увеличением времени с уменьшением K свидетельствует о бóльшем количестве потенциальных кандидатов, проверяемых с помощью классического алгорит-

Таблица 5.8

Результаты поиска экзонов в наборе Burset-Guigo для $K = 4, \ldots, 9$

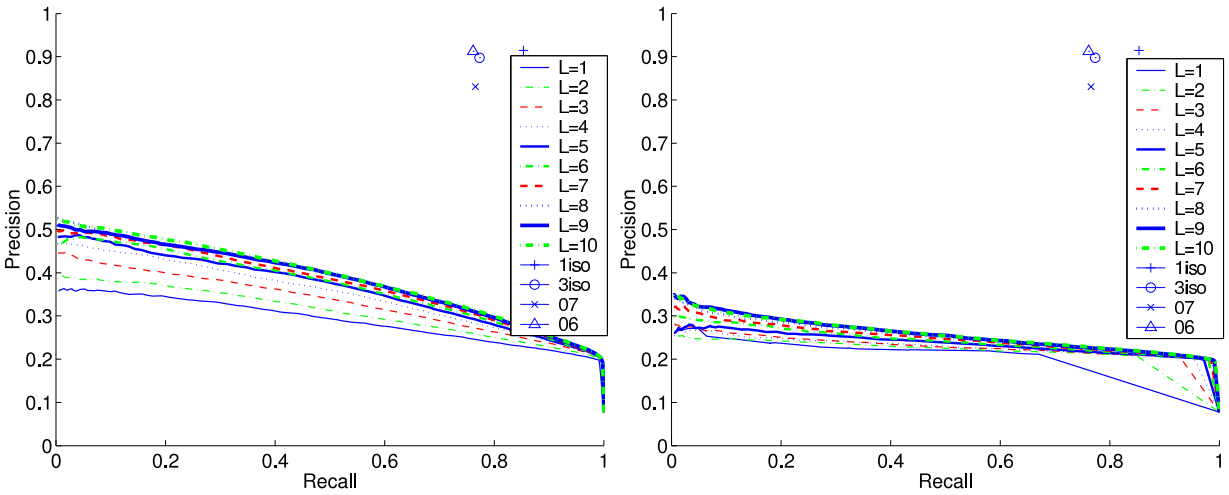
K	L	AC_{max}	AC_{avg}	σ^2_{AC}	F_{max}	время, час
4	1	0.440	0.353	0.0125	0.438	20.5
	2	0.452	0.365	0.0123	0.453	39.5
	3	0.463	0.373	0.0124	0.467	56.4
	4	0.468	0.377	0.0121	0.468	28.7
	5	0.469	0.382	0.0118	0.471	61.7
	6	0.470	0.384	0.0116	0.474	41.9
	7	0.470	0.384	0.0115	0.473	62.0
	8	0.473	0.385	0.0116	0.477	71.0
	9	0.471	0.385	0.0116	0.479	80.1
	10	0.474	0.385	0.0117	0.479	69.8
5	1	0.408	0.290	0.0153	0.379	5.5
	2	0.419	0.312	0.0150	0.394	11.2
	3	0.431	0.333	0.0140	0.413	22.7
	4	0.438	0.347	0.0133	0.429	17.5
	5	0.442	0.355	0.0129	0.440	21.7
	6	0.448	0.357	0.0132	0.443	33.9
	7	0.449	0.361	0.0128	0.449	39.2
	8	0.452	0.365	0.0125	0.454	34.0
	9	0.455	0.365	0.0128	0.456	37.9
	10	0.458	0.368	0.0126	0.458	42.2
6	1	0.394	0.228	0.0146	0.345	8.8
	2	0.403	0.255	0.0162	0.355	16.6
	3	0.408	0.278	0.0157	0.366	15.0
	4	0.411	0.287	0.0158	0.371	25.9
	5	0.413	0.299	0.0153	0.378	25.4
	6	0.415	0.306	0.0145	0.379	29.9
	7	0.420	0.315	0.0141	0.392	26.3
	8	0.423	0.323	0.0136	0.400	29.8
	9	0.426	0.326	0.0138	0.402	33.2
	10	0.428	0.330	0.0138	0.408	37.1

K	L	AC_{max}	AC_{avg}	σ^2_{AC}	F_{max}	время, час
7	1	0.308	0.165	0.0094	0.322	4.9
	2	0.364	0.184	0.0131	0.334	9.2
	3	0.385	0.206	0.0145	0.337	23.9
	4	0.391	0.212	0.0161	0.337	25.0
	5	0.393	0.226	0.0166	0.339	24.7
	6	0.395	0.237	0.0162	0.342	29.3
	7	0.397	0.246	0.0160	0.346	25.6
	8	0.398	0.255	0.0157	0.348	28.9
	9	0.400	0.261	0.0156	0.349	32.2
	10	0.402	0.263	0.0160	0.352	35.9
8	1	0.212	0.129	0.0062	0.279	8.3
	2	0.264	0.144	0.0077	0.306	15.7
	3	0.305	0.159	0.0094	0.319	18.8
	4	0.334	0.172	0.0106	0.327	18.0
	5	0.354	0.184	0.0117	0.332	24.5
	6	0.367	0.188	0.0126	0.332	28.9
	7	0.376	0.193	0.0137	0.334	25.2
	8	0.381	0.198	0.0143	0.336	28.6
	9	0.384	0.199	0.0152	0.336	31.9
	10	0.389	0.206	0.0151	0.337	35.6
9	1	0.149	0.100	0.0043	0.213	8.3
	2	0.176	0.108	0.0051	0.247	15.8
	3	0.199	0.112	0.0060	0.267	18.8
	4	0.235	0.128	0.0067	0.288	14.7
	5	0.256	0.140	0.0072	0.299	18.3
	6	0.270	0.142	0.0079	0.305	21.6
	7	0.284	0.142	0.0089	0.310	25.3
	8	0.291	0.143	0.0095	0.313	28.6
	9	0.304	0.145	0.0102	0.317	31.9
	10	0.312	0.151	0.0103	0.319	35.6

ма, что позволяет не пропустить действительных ближайших соседей. При увеличении L при фиксированном K результаты улучшаются, поскольку повышается вероятность для действительного соседа попасть в мультимножество кандидатов $|S|$.

Результаты работы предложенного метода поиска генов сравнивались с результатами программы для поиска генов GeneID 1.3.8 [50, 45]. Алгоритм программы основан на использовании ряда разработанных эвристик в комбинации с марковскими цепями со специально подобранными параметрами для разных типов организмов, а также правилами, выведенными экспертами. Для сравнительного анализа использованы следующие наборы параметров, предоставляемые с программой GeneID 1.3.8, – human1iso, human3iso, human.070606 и human.061209.

На рис. 5.11 сплошными линиями изображены графики изменения точности (отношение количества правильно классифицированных нуклеотидов к



(a) $K = 5$ (b) $K = 7$

Рис. 5.11. Графики точность-полнота для $K = 5, 7$ и $L = 1, \ldots, 10$, полученные разработанным методом, а также результаты GeneID для разных наборов параметров («+» – human1iso, «o» – human3iso, «x» – human.070606, «Δ» – human.061209)

общему числу нуклеотидов, классифицированных как кодирующие) от полноты (отношение количества правильно классифицированных нуклеотидов к общему числу кодирующих нуклеотидов) для метода на основе LSH-леса, полученные изменением порога t на значение c_i и усреднением по всем последовательностями тестовой выборки. Маркерами на рисунке изображены результаты GeneID. Как видно из рис. 5.11, метод, основанный на процедуре LSH и не использующий никаких специфических экспертных знаний о природе тестируемых последовательностей, тем не менее, правильно распознает принадлежность 25-50% нуклеотидов, находящихся в экзонах.

В ходе эксперимента было замечено, что короткие экзоны распознаются гораздо хуже, чем длинные, что может объяснять не очень высокий по сравнению с GeneID общий результат. Более высокие результаты работы программы GeneID также объясняются ее высокой специализацией и настройкой на конкретный тип организмов.

5.2.3.4. Методика классификации для поиска коротких некодирующих последовательностей в ДНК

Помимо поиска генов, исследовался поиск коротких гомологичных участков также в некодирующих участках генетических последовательностей для значений n, при которых не выполняется $\rho < 1$ (1.17).

Цель эксперимента – исследование эффективности поиска близких коротких последовательностей нуклеотидов, а именно гиперчувствительных сайтов в бета-глобине родственных организмов.

Обозначим обучающую выборку x_m, тестовую – h_m. Все подстроки длиной n обучающей выборки составляли базу P, на которой строился LSH-лес. Создавался также массив счетчиков $c, |c| = |h_m|$, изначально инициализированный нулями. При проходе последовательности h_m скользящим окном шириной n с помощью LSH-процедуры находились кандидаты на приближенных ближайших соседей к запросам $q_i = h[i, i + n - 1], i = 1, \ldots, |h_m|$. Если среди строк-кандидатов с глубиной совпадения K, возвращенных LSH- процедурой на запрос q_i, была хотя бы одна подстрока, принадлежащая шару $S(q, e)$ (это проверялось с помощью классического алгоритма вычисления расстояния редактирования), то значения $c(j)$ для всех $j = i, \ldots, i + n - 1$ увеличивались на единицу. Проводилось усреднение по 100 независимым запускам процедуры поиска с разными инициализациями генератора случайных чисел.

5.2.3.5. Сравнительные результаты поиска гиперчувствительных сайтов

Реальное количество подстрок длиной 50, находящихся на расстоянии редактирования не больше $e = 3$ в исследованных последовательностях, равно 19. В табл. 5.9 приведены результаты поиска. В колонке c приведено (усредненное по 100 независимым запускам) количество найденных подстрок, в колонке $\sigma^2(c)$ – дисперсия этого числа, в двух остальных колонках, соответственно, среднее время поиска и затраты памяти.

Используя программу из [82], все 19 подстрок были найдены за 0.4 мин. (на машине Athlon XP 2600+ под оболочкой Cygwin). Как видно из табл. 2, результаты, полученные с помощью метода, основанного на процедуре LSH-

Таблица 5.9

Пример результатов поиска подстрок длиной $n = 50$, находящихся на расстоянии редактирования не больше 3 в последовательностях бета-глобина мыши и человека для $K = 7, 8$

	с	$\sigma^2(c)$	время, мин	память, Мб	с	$\sigma^2(c)$	время, мин	память, Мб
$L\backslash K$	7				8			
1	0.3	0.5	0.01	2.3	0.3	0.6	0.01	2.7
2	0.6	1.1	0.01	3.4	0.7	1	0.01	4.1
3	1	1.7	0.01	4.4	0.9	1.4	0.01	5.4
4	1.6	2.5	0.01	5.4	1.4	2.1	0.01	6.8
5	2	3.4	0.02	6.4	1.7	2.2	0.02	8.2
6	2.4	3.6	0.02	7.4	2.1	3.1	0.02	9.6
7	2.8	3.8	0.02	8.5	3	0	0.02	10.9
8	3.2	4.3	0.03	9.5	2.6	3.2	0.03	12.3
9	3.5	5	0.03	10.5	2.9	3.4	0.03	13.7
10	3.9	5.7	0.03	11.5	3.2	3.7	0.03	15.1
20	7.2	6.1	0.07	21.7	5.8	7.1	0.07	28.8
30	9.9	7.3	0.11	31.9	8.2	7.4	0.1	42.6
40	12.1	7.6	0.15	42.1	10.2	6.1	0.14	56.3
50	13.6	5.6	0.19	52.2	11.6	5.7	0.18	70.1
100	17.4	1.7	0.4	103	16.4	2.8	0.36	138.7
150	18.3	0.7	0.61	153.7	17.9	1	0.55	207.3
200	18.6	0.4	0.84	204.3	18.4	0.7	0.74	275.8
300	18.7	0.3	1.23	305.8	18.7	0.3	1.18	412.9

лес, в среднем уступают результатам [82], что может объясняться слишком короткими строками ($n = 50$), тогда как минимальная длина подстрок, при которой $\rho < 1$ (1.17) равна $n = 100$.

5.2.4. Обнаружение вторжений в компьютерные системы

Обнаружение хакерских атак является актуальной задачей и, одновременно, сложной, так как поток данных, генерируемых аудит-системами, имеет огромный объем – до гигабайтов в день. Системы обнаружения вторжений предназначены для обнаружения попыток несанкционированного доступа в компьютерную систему. Такие системы должны противостоять атакам, даже если злоумышленник с точки зрения соблюдения прав доступа имел необходимые полномочия на свои действия. Одним из подходов к обнаружению атак является создание профиля «нормальной» активности пользователя, а любая активность, не подпадающая под принятое понимание «нормальности», считается опасной. Такие системы обнаружения вторжений называются системами обнаружения аномалий. Некоторые существующие подходы к построению систем обнаружения аномалий рассмотрены в [144, 143].

Одним из направлений обнаружения вторжений есть их обнаружение рассуждениями на основе примеров, где в качестве базы примеров используют аудит-файлы (логи) пользователей. Используются подходы, оперирующие в каждый момент времени ограниченными участками сессии (окнами). Наиболее востребованы системы обнаружения аноманий в виде online-систем, которым, очевидно, не доступны будущие команды, а старые логи, как правило, ежедневно или даже ежечасно архивируются, поскольку на их хранение необходимы ресурсы.

5.2.4.1. Классификация сессий пользователей для обнаружения вторжений с помощью рассуждений по примерам

Мы применили подход на основе рассуждений по примерам для классификации сессий пользователей для обнаружения вторжений в компьютерных системах.

Цель эксперимента – проверить, насколько эффективной может быть классификация принадлежности пользовательских сессий (последовательностей системных команд) на основе предложенного метода поиска близких строк.

5.2.4.2. Базы данных для экспериментального исследования

Эксперименты проводились на данных, полученных с UNIX-сервера физико-технического института НТУ Украины «КПИ» [104, 126]. Механизмами аудита ОС FreeBSD отслеживались процессы, запускавшиеся от имени зарегистрированных в системе пользователей, на протяжении 671 дня (с июня 2001 г. по декабрь 2003 г., с перерывами). Всего были получены данные для 717 пользователей, выполнивших более 23 млн. команд.

В качестве обучающей выборки были приняты все сессии, выполненные за 2001-2002 год (всего 403 дня), в качестве тестовой – сессии 2003 года (268 дней). Роль обучающей выборки здесь играет множество подпоследовательностей логов сессий. В отличие от задачи поиска генов, где кодирующие экзоны редки и разделены длинными интронами, все команды в логе несут определенную семантическую нагрузку. Поэтому в данной задаче проводилась «нарезка» и обучающего, и тестового логов на окна одинаковой длины (п. 4.3.2).

5.2.4.3. Методика классификации сессий

Обучающий лог x пользовательской сессии разбивался на пересекающиеся окна вида $x[i, i+n-1], i = 1, \ldots, |x|-n+1$ фиксированной длины n, распределенные представления которых сохраняются для каждого пользователя u в отдельном LSH-лесе F_u. На этапе тестирования создается массив счетчиков C_u для каждого пользователя u. Каждое окно $y[j, j+n-1], j = 1, \ldots, |y|-n+1$ тестового лога у пользователя u^* является запросом в F_u для всех присутствовавших в обучающей сессии пользователей u. Обозначим $l_{max}(u, y[j, j+n-1])$ максимальное значение уровня строк, возращенных в ходе процедуры LSH-лес для леса F_u при запросе $y[j, j+n-1]$. Если $l_{max}(u, y[j, j+n-1]) = K$, то значение счетчика C_u увеличивается на единицу. Пусть U – множество пользователей, таких, что C_u имеет максимальное значение для всех пользователей: $U = arg\max_u(C_u)$. Если $u^* \in U$, то считалось, что пользователь определен правильно. Иначе, пользователь определен неправильно и имеет место аномалия.

Таблица 5.10

Доля правильно классифицированных сессий для значений ширины окна $n = 10, 20, 40$ и параметров $K = 5, 7$, $L = 1, 5, 10$

n	K	L		
		1	5	10
10	5	0.470	0.984	0.997
	7	0.431	0.945	0.987
20	5	0.440	0.942	0.983
	7	0.403	0.741	0.942
40	5	0.354	0.848	0.967
	7	0.230	0.390	0.836

5.2.4.4. Результаты

Доля правильно классифицированных сессий для разных значений ширины окна $n = 10, 20, 30, 40$ и параметров $K = 5, 7$, $L = 1, 5, 10$ представлена в табл. 5.10. Видно, что при увеличении L достигается точность классификации практически 100%. Затраты времени в среднем составляют от 3 до 11 сек. (в зависимости от параметров) на проверку одной сессии. Таким образом, предложенный метод является перспективным в качестве предварительной онлайн-обработки логов.

Результаты данного раздела отражены в публикациях [78, 108, 123, 144, 130, 146, 147].

5.3. Выводы по разделу 5

1. Разработаны базовые программные библиотеки, которые реализуют оригинальные методы распределенного представления последовательностей, поиска приближенных ближайших последовательностей и классификации в прикладных задачах:
   - библиотечный модуль VectorComparer – унифицированное средство

сравнения векторов по множеству стандартных метрик и мер;

- мультиплатформенная программная библиотека LSH, реализующая разработанные методы представления и методы поиска приближенных ближайших символьных последовательной произвольной природы;
- библиотека форматированного ввода TextInputTools, содержащая модули форматированного ввода данных для баз генетических последовательностей, электронных писем, ряда популярных текстовых баз, аудит-последовательностей UNIX-систем.

Модули дают основу для использования в системах искусственного интеллекта, применяющих поиск символьных последовательностей для классификации в прикладных задачах.

2. Разработаны специализированные программные системы DuplClassifier, EmailClassifier, NuclClassifier, SessionClassifier для поиска текстовых дубликатов, спама, кодирующих участков генетических последовательностей и классификации сессий пользователей UNIX-системы. Созданные программные системы позволили проверить эффективность заложенных в них методов представления и поиска приближенных ближайших для решения задач классификации по примерам.
3. Разработанны модули и подсистемы программного нейрокомпьютера SNC (Software NeuroComputer) , который является средством визуальной разработки, реализации и использования нейросетевых технологий обработки информации, задания потоков данных, алгоритмов их обработки. Использование средств визуального конфигурирования, средств сохранения результатов в базе и последующей обработки значительно упростили исследование методов векторного представления последовательностей в прикладных задачах классификации.
4. Созданные программные библиотеки, системы и средства подтвердили эффективность заложенных в них методов распределенного представления последовательностей для решения задач поиска и классификации по

примерам.

5. Разработанные методы поиска сходных последовательностей за счет использования распределенных представлений и локально-чувствительного хеширования обеспечивают поиск последовательностей разной длины в реальных базах данных и решение прикладных задач поиска дубликатов и спама на основе рассуждений по примерам. При поиске дубликатов в базе РОМИП результат улучшен на 20-40%, на базе Reuters-21578 – на уровне известных. Перспективность применения методов для обнаружения спама в крупных почтовых серверах показана на примере оценки количества спама в коллекциях электронных писем TREC Spam Track 2006, где обнаружено до 80% спама при уровне неправильно классифицированных легальных сообщений 5-10%.
6. Разработанные методы представления и поиска последовательностей обеспечивают решение прикладных задач классификации участков ДНК и обнаружения вторжений, что подтверждает эффективность использования рассуждений на основе примеров для обработки последовательностей в реальных базах данных. В задаче классификации участков ДНК поиск экзонов ускорен в 750 раз при сохранении качества на уровне известных результатов в этой области, использующих подход на основе рассуждений по примерам. Разработанный метод поиска последовательностей может применяться при более широкой области значений параметров, чем следует из теоретического анализа, что экспериментально показано на примере задачи поиска некодирующих участков бета-глобина при обработке коротких строк. Метод перспективен для применения в реальных системах обнаружения вторжений, что подтверждается результатом классификации аудит-последовательностей компьютерных систем, где получена точность классификации на уровне более 90%.

ВЫВОДЫ

Совокупность полученных в диссертации результатов обеспечивает решение актуальной научной задачи разработки методов нейросетевого распределенного представления последовательностей, а также их поиска и классификации для эффективной оценки сходства и использования информации о последовательностях в системах искусственного интеллекта, применяющих модели рассуждений человека на основе примеров. Разработаны, аналитически исследованы, а также программно реализованы методы распределенного представления и поиска последовательностей. Эффективность разработанных методов подтверждена экспериментальными исследованиями на тестовых и реальных данных при решении задач поиска сходных последовательностей и классификации информации различного рода (тексты, ДНК, аудит-последовательности).

По результатам проведенного исследования сделаны следующие выводы:

1. Разработанный метод векторного представления обеспечивает сохранение сходства данных с последовательной структурой по расстоянию редактирования и возможность анализа с помощью теории метрических вложений, что позволяет оценивать характеристики поиска и классификации последовательностей в прикладных задачах.
2. Предложенное векторное представление последовательностей обеспечивает более высокую точность аппроксимации расстояния редактирования по сравнению с известными результатами, что показано аналитически с помощью теории метрических вложений и путем численных экспериментов на искусственных данных.
3. Разработанные, проанализированные и реализованные методы распределенного представления последовательностей за счет использования локально-чувствительного хеширования обеспечивают малую ресурсоемкость и сублинейное относительно размера базы примеров время по-

иска приближенно ближайших последовательностей. Экспериментальное исследование качества поиска на искусственных данных показало достаточность использования на практике меньших, чем определенных аналитически, значений параметров метода, что позволяет уменьшить ресурсоемкость поиска.

4. Предложенный метод нейросетевого распределенного представления последовательностей, использующий рандомизацию векторных представлений и связывание элементов последовательности с их позициями, обеспечивает унификацию формата представления и возможность использования мер сходства векторных представлений для оценки сходства последовательностей.
5. Разработанные методы поиска сходных последовательностей с помощью кластеризации по длине последовательностей и выравнивания длины, за счет использования распределенных представлений и локально-чувствительного хеширования обеспечивают поиск последовательностей разной длины в реальных базах данных и решение прикладных задач поиска дубликатов и спама на основе рассуждений по примерам. Эффективность и практическая значимость методов подтверждена сравнением полученных результатов с известными. Так, при поиске дубликатов в базе РОМИП результат улучшен на 20-40%, на базе Reuters-21578 - на уровне известных. Перспективность применения методов для обнаружения спама в крупных почтовых серверах показана на примере оценки количества спама в коллекциях электронных писем TREC Spam Track 2006 и 2005, где обнаружено до 80% спама при уровне неправильно классифицированных легальных сообщений 5-10%.
6. Разработанные методы представления и поиска последовательностей обеспечивают решение прикладных задач классификации участков ДНК и обнаружения вторжений, что подтверждает эффективность использования рассуждений на основе примеров для обработки последовательностей в реальных базах данных. В задаче классификации участков ДНК поиск

экзонов ускорен в 750 раз при сохранении качества на уровне известных результатов в этой области, использующих подход на основе рассуждений по примерам. Разработанный метод поиска последовательностей может применяться при более широкой области значений параметров, чем следует из теоретического анализа, что экспериментально показано на примере задачи поиска некодирующих участков бета-глобина при обработке коротких строк. Метод перспективен для применения в реальных системах обнаружения вторжений, что подтверждается результатом классификации аудит-последовательностей компьютерных систем, где получена точность классификации на уровне более 90%.

7. Разработанные методы представления и поиска приближенных ближайших последовательностей, реализованные в виде программных средств, могут быть использованы в качестве компонентов информационных технологий, либо как самостоятельные модули, в системах классификации и поиска последовательностей. Практическая значимость разработок подтверждается 3 актами внедрения.

ПРИЛОЖЕНИЕ А
АКТЫ ВНЕДРЕНИЯ

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] *Achlioptas D.* Database-friendly random projections: Johnson-lindenstrauss with binary coins / D. Achlioptas // *J. Comput. Syst. Sci.*, 2003. — V. 66, № 4. — P. 671–687.

[2] *Alon N.* Tracking join and self-join sizes in limited storage / N. Alon, P. B. Gibbons, Y. Matias, M. Szegedy // Proc. of the 18th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. — New York, NY, USA: ACM Press, 1999. — P. 10–20.

[3] *Alon N.* The space complexity of approximating the frequency moments / N. Alon, Y. Matias, M. Szegedy // *J. of Computer and System Sciences*, 1999. — № 58. — P. 137–147.

[4] *Amosov N.* Modelling of Thinking and the Mind / N. Amosov. — New York: Spartan Books, 1967. — 304 p.

[5] *Andoni A.* Lower bounds for embedding edit distance into normed spaces / A. Andoni, M. Deza, A. Gupta, P. Indyk, S. Raskhodnikova // Proc. of the 14th annual ACM-SIAM symposium on Discrete algorithms. — Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003. — P. 523–526.

[6] *Andoni A.* Efficient algorithms for substring near neighbor problem / A. Andoni, P. Indyk // Proc. of the 17th annual ACM-SIAM symposium on Discrete algorithms. — New York, NY, USA: ACM, 2006. — P. 1203–1212.

[7] *Azenkot S.* An evaluation of the edit-distance-with-moves similarity metric for comparing genetic sequences: Tech. Rep. 2005-39 / S. Azenkot, T.-Y. Chen, G. Cormode: Center for Discrete Mathematics and Theoretical Computer Science, DIMACS, 2005. — 18 p.

[8] *Badoiu M.* Fast approximate pattern matching with few indels via embeddings / M. Badoiu, P. Indyk // Proc. of the 15th annual ACM-SIAM sympo-

sium on Discrete Algorithms. — Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2004. — P. 651–652.

[9] *Baeza-Yates R.* Modern Information Retrieval / R. Baeza-Yates, B. Ribeiro-Neto. — Addison Wesley, 1999. — 544 p.

[10] *Bar-Yossef Z.* Approximating edit distance efficiently / Z. Bar-Yossef, T. S. Jayram, R. Krauthgamer, R. Kumar // Proc. of the 45th Annual IEEE Symposium on Foundations of Computer Science. — Washington, DC, USA: IEEE Computer Society, 2004. — P. 550–559.

[11] *Batu T.* A sublinear algorithm for weakly approximating edit distance / T. Batu, F. Ergün, J. Kilian, A. Magen, S. Raskhodnikova, R. Rubinfeld, R. Sami // Proc. of the 35th annual ACM symposium on Theory of computing. — New York, USA: ACM, 2003. — P. 316–324.

[12] *Batu T.* Oblivious string embeddings and edit distance approximations / T. Batu, F. Ergun, C. Sahinalp // Proc. of the 17th annual ACM-SIAM symposium on Discrete Algorithms. — New York, USA: ACM, 2006. — P. 792–801.

[13] *Bawa M.* LSH forest: self-tuning indexes for similarity search / M. Bawa, T. Condie, P. Ganesan // Proc. of the 14th Int. Conf. on WWW. — New York, NY, USA: ACM, 2005. — P. 651–660.

[14] *Benson D.* Genbank / D. Benson, I. Karsch-Mizrachi, D. Lipman, J. Ostell, B. A. Rapp, D. L. Wheeler // *Nucleic Acids Research*, 2000. — V. 28, № 1. — P. 15–18.

[15] BNC. — The British National Corpus. — *http://www.natcorp.ox.ac.uk/*.

[16] *Borodin A.* Lower bounds for high dimensional nearest neighbor search and related problems / A. Borodin, R. Ostrovsky, Y. Rabani // Proc. of the 31-st annual ACM symposium on Theory of computing. — New York, NY, USA: ACM, 1999. — P. 312–321.

[17] *Brin S.* Copy detection mechanisms for digital documents / S. Brin, J. Davis, H. García-Molina // Proc. of the ACM SIGMOD international conference on Management of data. — New York, NY, USA: ACM, 1995. — P. 398–409.

[18] *Brinkman B.* On the impossibility of dimension reduction in l1 / B. Brinkman, M. Charikar // *J. ACM*, 2005. — V. 52, № 5. — P. 766–788.

[19] *Broder A.* On the resemblance and containment of documents / A. Broder // Proc. of the Conf. on Compression and Complexity of Sequences. — Washington, DC, USA: IEEE Computer Society, 1997. — P. 21–29.

[20] *Broder A. Z.* Syntactic clustering of the web / A. Z. Broder, S. C. Glassman, M. S. Manasse, G. Zweig // Proc. of the 6th Int. Conf. on WWW. — 1997. — P. 1157–1166.

[21] *Buhler J.* Efficient large-scale sequence comparison by locality-sensitive hashing / J. Buhler // *Bioinformatics*, 2001. — V. 17, № 5. — P. 168–173.

[22] *Burset M.* Evaluation of gene structure prediction programs / M. Burset, R. Guigó // *Genomics*, 1996. — V. 34. — P. 353–367.

[23] *Charikar M. S.* Similarity estimation techniques from rounding algorithms / M. S. Charikar // Proc. of the 34th annual ACM symposium on Theory of Computing. — New York, NY, USA: ACM, 2002. — P. 380–388.

[24] *Chaudhuri S.* Robust and efficient fuzzy match for online data cleaning / S. Chaudhuri, K. Ganjam, V. Ganti, R. Motwani // SIGMOD '03: Proc. of the 2003 ACM SIGMOD Int. Conf. on Management of data. — New York, NY, USA: ACM Press, 2003. — P. 313–324.

[25] *Cormack G.* TREC 2006 spam track overview / G. Cormack // Proc. of the 15th Text REtrieval Conf. — Gaithersburg, MD: NIST, 2006. — *http://trec.nist.gov/pubs/trec15/papers/SPAM06.OVERVIEW.pdf* .

[26] *Cormack G.* TREC 2005 spam track overview / G. Cormack, T. Lynam // Proc. of the 14th Text REtrieval Conf. / Ed. by E. M. Voorhees, L. P. Buckland. — Gaithersburg, MD: NIST, 2005. — *http:trec.nist.gov/pubs/trec14/papers/SPAM.OVERVIEW.pdf* .

[27] *Cormode G.* Sequence Distance Embeddings: Ph.D. thesis / University of Warwick. — 2003. — 174 p.

[28] *Cormode G.* Comparing data streams using hamming norms (how to zero in) / G. Cormode, M. Datar, P. Indyk, S. Muthukrishnan // *IEEE Transactions on*

Knowledge and Data Engineering, 2003. — V. 15, № 3. — P. 529–540.

[29] *Cormode G.* Fast mining of tabular data via approximate distance computations / G. Cormode, P. Indyk, N. Koudas, S. Muthukrishnan // Int. Conf. on Data Engineering. — 2002. — P. 605–616.

[30] *Cormode G.* The string edit distance matching problem with moves / G. Cormode, S. Muthukrishnan // Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms. — Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002. — P. 667–676.

[31] *Cormode G.* Communication complexity of document exchange / G. Cormode, M. Paterson, S. C. Sahinalp, U. Vishkin // Proc. of the 11th annual ACM-SIAM symposium on Discrete algorithms. — Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000. — P. 197–206.

[32] *Costello E.* A case-based approach to gene finding / E. Costello, D. C. Wilson // Proc. Workshop CBR Health Sci. — 2003. — P. 19–28.

[33] *Damerau F. J.* A technique for computer detection and correction of spelling errors / F. J. Damerau // *Commun. ACM*, 1964. — V. 7, № 3. — P. 171–176.

[34] *Dasgupta S.* An elementary proof of the johnson-lindenstrauss lemma: Tech. Rep. TR-99-006 / S. Dasgupta, A. Gupta. — Berkeley, CA, USA: U.C. Berkeley, 1999. — 6 p.

[35] *Datar M.* Locality-sensitive hashing scheme based on p-stable distributions / M. Datar, N. Immorlica, P. Indyk, V. S. Mirrokni // Proc. of the 20th annual symposium on Computational geometry. — New York, NY, USA: ACM, 2004. — P. 253–262.

[36] *de Bruijn N.* A combinatorial problem / N. de Bruijn // Koninklijke Nederlandsche Akademie van Wetenschappen. — V. 49. — 1946. — P. 758–764.

[37] *Dhamdhere K.* Approximation algorithms for minimizing average distortion / K. Dhamdhere, A. Gupta, R. Ravi // STACS. — 2003. — P. 234–245.

[38] *Duda R.* Pattern Classification / R. Duda, P. Hart, D. Stork. — 2nd ed. edition. — New York: John Wiley & Sons, 2000. — 680 p.

[39] Email metrics program: The network operators' perspective: Tech. Rep.

1 - 4th Quarter: Messaging Anti-Abuse Working Group, 2005. — 3 p. — *http://www.maawg.org/about/FINAL_4Q2005_Metrics_Report.pdf*.

[40] *Fagin R.* Comparing top k lists / R. Fagin, R. Kumar, D. Sivakumar // Proc. of the 14th annual ACM-SIAM symposium on Discrete algorithms. — Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003. — P. 28–36.

[41] *Feigenbaum J.* An approximate L1-difference algorithm for massive data streams / J. Feigenbaum, S. Kannan, M. Strauss, M. Viswanathan // Proc. of the 40th Annual Symposium on Foundations of Computer Science. — IEEE Computer Society, 1999. — P. 501–511.

[42] *Forrester W.* Evidence for a locus activation region: the formation of developmentally stable hypersensitive sites in globin-expressing hybrids / W. Forrester, S. Takegawa, T. Papayannopoulou, G. Stamatoyannopoulos, M. Groudine // *Nucl. Acids Res.*, 1987. — V. 24, № 15. — P. 10159–10177.

[43] *Garofalakis M.* Correlating XML data streams using tree-edit distance embeddings / M. Garofalakis, A. Kumar // Proc. of the 22nd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. — ACM Press, 2003. — P. 143–154.

[44] Genbank. — National Center for Biotechnology Information, National Institutes of Health. — *ftp://ftp.ncbi.nih.gov/genbank/*.

[45] Geneid. — Genome BioInformatics Research Lab, Institut Municipal d'Investigació Mèdica. — *http://genome.imim.es/software/geneid/*.

[46] *Gionis A.* Similarity search in high dimensions via hashing / A. Gionis, P. Indyk, R. Motwani // Proc. of the 25th Int. Conf. on Very Large Data Bases. — San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. — P. 518–529.

[47] Gmail. — Google Mail: Google Inc. — *http://www.gmail.com*.

[48] *Graham-Cummings J.* The spammers' compendium / J. Graham-Cummings // Proc. of the Spam Conf. — 2003. — P. 1–17. — *http://www.jgc.org/tsc.html*.

[49] *Grossman D. A.* Information Retrieval: Algorithms and Heuristics /

D. A. Grossman, O. Frieder. — Norwell, MA, USA: Kluwer Academic Publishers, 1998. — 276 p.

[50] *Guigó R.* Prediction of gene structure / R. Guigó, S. Knudsen, N. Drake, T. F. Smith // *J. of Molecular Biology*, 1992. — V. 226. — P. 141–157.

[51] *Gusfield D.* Algorithms on strings, trees, and sequences: computer science and computational biology / D. Gusfield. — New York, NY, USA: Cambridge University Press, 1997. — 554 p.

[52] *Halperin E.* Detecting protein sequence conservation via metric embeddings / E. Halperin, J. Buhler, R. Karp, R. Krauthgamer, B. Westover // *Bioinformatics*, 2003. — V. 1, № 1. — P. 1–8.

[53] *Hawking D.* Overview of the TREC8 web track / D. Hawking, E. Voorhees, N. Craswell, P. Bailey // Proc. of the 8th Text REtrieval Conf. — Gaithersburg: 1999. — P. 1–24.

[54] *Henzinger M. R.* Computing on data streams / M. R. Henzinger, P. Raghavan, S. Rajagopalan // *External memory algorithms*, 1999. — P. 107–118.

[55] *Hoeffding W.* Probability inequalities for sums of bounded random variables / W. Hoeffding // *J. of American Statistical Association*, 1963. — V. 58, №301. — P. 13–30.

[56] *Ilyinsky S.* An efficient method to detect duplicates of web documents with the use of inverted index / S. Ilyinsky, M. Kuzmin, A. Melkov, I. Segalovich // Proc. 11th Int. Conf. on WWW. — 2002. — *http://www2002.org/CDROM/poster/187/*.

[57] *Indyk P.* Algorithmic aspects of geometric embeddings / P. Indyk // Proc. of the 42nd Annual IEEE Symposium on Foundations of Computer Science. — Washington, DC, USA: IEEE Computer Society, 2001. — 10 p.

[58] *Indyk P.* Open problems / P. Indyk // Workshop on Discrete Metric Spaces and their Algorithmic Applications / Ed. by J. Matoušek. — Haifa: 2002. — *http://kam.mff.cuni.cz/~matousek/metrop.ps.gz*.

[59] *Indyk P.* Stable distributions, pseudorandom generators, embeddings, and data stream computation / P. Indyk // *J. ACM*, 2006. — V. 53, № 3. — P. 307–323.

[60] *Indyk P.* Approximate nearest neighbors: towards removing the curse of dimensionality / P. Indyk, R. Motwani // Proc. of the 30th annual ACM symposium on Theory of computing. — New York, NY, USA: ACM Press, 1998. — P. 604–613.

[61] *Jaccard P.* Étude comparative de la distribution florale dans une portion des alpes et des jura / P. Jaccard // *Bulletin del la Société Vaudoise des Sciences Naturelles*, 1901. — № 37. — P. 547–579.

[62] *Johnson W.* Extensions of Lipschitz maps into a Hilbert space / W. Johnson, J. Lindenstrauss // *Contemp. Math.*, 1984. — № 26. — P. 189–206.

[63] *Jokinen P.* Two algorithms for approximate string matching in static texts (extended abstract) / P. Jokinen, E. Ukkonen // Proc. of the 16th Int. Symposium on Mathematical Foundations of Computer Science / Ed. by A. Tarlecki. — Berlin, Heidelberg: Springer, 1991. — P. 240–248.

[64] *Kanerva P.* Binary spatter-coding of ordered K-tuples / P. Kanerva // Proc. of Int. Conf. on Artificial Neural Networks. — Berlin: Springer, 1996. — P. 869–873.

[65] *Karp R. M.* Efficient randomized pattern-matching algorithms / R. M. Karp, M. O. Rabin // *IBM J. Research and Development*, 1987. — V. 31, № 2. — P. 249–260.

[66] *Khot S.* Nonembeddability theorems via fourier analysis / S. Khot, A. Naor // Proc. of the 46th Annual IEEE Symposium on Foundations of Computer Science. — Washington, USA: IEEE Computer Society, 2005. — P. 101–112.

[67] *Knuth D. E.* Seminumerical Algorithms / D. E. Knuth. — Second edition. — Reading, Massachusetts: Addison-Wesley, 1981. — V. 2 of *The Art of Computer Programming*. — 688 p.

[68] *Kolcz A.* The impact of feature selection on signature-driven spam detection / A. Kolcz, A. Chowdhury, J. Alspector // Proc. of the 1st Conf. on Email and Anti-Spam. — Mountain View, CA, USA: 2004. — *http://www.ceas.cc/papers-2004/147.pdf* .

[69] *Kolodner J.* Case-based Reasoning / J. Kolodner. — San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1993. — 668 p.

[70] *Krauthgamer R.* Improved lower bounds for embeddings into L1 / R. Krauthgamer, Y. Rabani // Proc. of the 17th annual ACM-SIAM symposium on Discrete algorithm. — NY, USA: ACM, 2006. — P. 1010–1017.

[71] *Kushilevitz E.* Efficient search for approximate nearest neighbor in high dimensional spaces / E. Kushilevitz, R. Ostrovsky, Y. Rabani // Proc. of 30th annual ACM symposium on Theory of computing. — 1998. — P. 614–623.

[72] *Kussul E.* Associative-projective neural networks: Architecture, implementation, applications / E. Kussul, D. Rachkovskij, T. Baidyk // 4th Int. Conf. Neural Networks & their Applications. — 1991. — P. 463–476.

[73] *Kussul M.* A visual solution to modular neural network system development / M. Kussul, A. Riznyk, E. Sadovaya, A. Sitchov, T.-Q. Chen // Int. Joint Conf. on Neural Networks. — V. 1. — 2002. — P. 749–754.

[74] *Lewis D.* Reuters-21578 collection / D. Lewis. — *http://www.daviddlewis.com/resources/testcollections/reuters21578*.

[75] *Lopresti D.* Block edit models for approximate string matching / D. Lopresti, A. Tomkins // *Theor. Comput. Sci.*, 1997. — V. 181, № 1. — P. 159–179.

[76] *Masek W. J.* A faster algorithm computing string edit distances / W. J. Masek, M. Paterson // *J. Comput. Syst. Sci.*, 1980. — V. 20, № 1. — P. 18–31.

[77] *Maurer H.* Plagiarism - a survey / H. Maurer, F. Kappe, B. Zaka // *J. of Universal Computer Science*, 2006. — V. 12, № 8. — P. 1050–1084.

[78] *Misuno I.* SNC: The software neurocomputer with modular architecture / I. Misuno, D. Rachkovskij, E. Revunova, A. Sokolov // Междунар. конф. "Проблемы нейрокибернетики". — Т. 2. — Ростов-на-Дону, Россия: 2002. — С. 109–113.

[79] *Muthukrishnan S.* Data Streams: Algorithms and Applications / S. Muthukrishnan. — *http://www.cs.rutgers.edu/~muthu/stream-1-1.ps*.

[80] *Muthukrishnan S.* Approximate nearest neighbors and sequence comparison with block operations / S. Muthukrishnan, S. C. Sahinalp // Proc. of the 32nd

annual ACM symposium on Theory of computing. — New York, NY, USA: ACM, 2000. — P. 416–424.

[81] *Myers E. W.* An O(ND) difference algorithm and its variations / E. W. Myers // *Algorithmica*, 1986. — V. 1, № 2. — P. 251–266.

[82] *Narayanan M.* Gapped local similarity search with provable guarantees / M. Narayanan, R. M. Karp // Algorithms in Bioinformatics, 4th Int. Workshop / Ed. by I. Jonassen, J. Kim. — V. 3240 of *Lecture Notes in Computer Science*. — Bergen, Norway: Springer, 2004. — P. 74–86.

[83] *Navarro G.* A guided tour to approximate string matching / G. Navarro // *ACM Computing Surveys*, 2001. — V. 33, № 1. — P. 31–88.

[84] *Needleman S. B.* A general method applicable to the search for similarities in the amino acid sequence of two proteins / S. B. Needleman, C. D. Wunsch // *J. of Molecular Biology*, 1970. — V. 48, № 3. — P. 443–453.

[85] *Nolan J. P.* Stable Distributions - Models for Heavy Tailed Data / J. P. Nolan. — Boston: Birkhäuser, 2007. — 352 p.

[86] *Ostrovsky R.* Low distortion embeddings for edit distance / R. Ostrovsky, Y. Rabani // Proc. of the 37th annual ACM symposium on Theory of computing. — New York, NY, USA: ACM Press, 2005. — P. 218–224.

[87] *Paul G.* Plan for spam / G. Paul // 2002. — *http://www.paulgraham.com/stopspam.html*.

[88] *Plate T.* Holographic reduced representations / T. Plate // *IEEE Transactions on Neural Networks*, 1995. — V. 6, № 3. — P. 623–641.

[89] *Plate T.* Holographic Reduced Representation: Distributed Representation for Cognitive Structures / T. Plate. — CSLI Publications, 2003. — 300 p.

[90] *Pugh W.* Detecting duplicate and near-duplicate files / W. Pugh, M. R. Henzinger // 2003. — United States Patent 6,658,423, granted on Dec 2, 2003.

[91] *Rachkovskij D.* Representation and processing of structures with binary sparse distributed codes / D. Rachkovskij // *IEEE Transactions on Knowledge and Data Engineering*, 2001. — V. 13, № 2. — P. 261–276.

[92] *Rachkovskij D. A.* Binding and normalization of binary sparse distributed representations by context-dependent thinning / D. A. Rachkovskij, E. M. Kussul // *Neural Computation*, 2001. — V. 13, № 2. — P. 411–452.

[93] *Rogic S.* Evaluation of gene-finding programs on mammalian sequences / S. Rogic, A. K. Mackworth, F. B. Ouellette // *Genome Res*, 2001. — V. 11, № 5. — P. 817–832.

[94] *Sahinalp S. C.* Symmetry breaking for suffix tree construction / S. C. Sahinalp, U. Vishkin // Proc. of the 26th annual ACM symposium on Theory of computing. — New York, NY, USA: ACM, 1994. — P. 300–309.

[95] *Sakharkar M. K.* Distributions of exons and introns in the human genome / M. K. Sakharkar, V. T. K. Chow, P. Kangueane // *In Silico Biology*, 2004. — V. 4, № 4. — P. 387–393.

[96] *Salton G.* Automatic Text Processing - The Transformation, Analysis, and Retrieval of Information by Computer / G. Salton. — Addison-Wesley, 1988. — 543 p.

[97] *Salton G.* Term-weighting approaches in automatic text retrieval / G. Salton, C. Buckley // *Inf. Process. Manage.*, 1988. — V. 24, № 5. — P. 513–523.

[98] *Salton G.* A vector space model for automatic indexing / G. Salton, A. Wong, C. S. Yang // *Commun. ACM*, 1975. — V. 18, № 11. — P. 613–620.

[99] *Sanderson M.* Duplicate detection in the Reuters collection: Tech. Rep. TR-1997-5 / M. Sanderson: Department of Computing Science, University of Glasgow, 1997. — 11 p.

[100] *Shamir R.* Lecture Notes in Analysis of Gene Expression Data, DNA Chips and Gene Networks: Sequencing by Hybridization / R. Shamir. — *http://cs.tau.ac.il/~rshamir/ge/04/scribes/lec02.pdf* .

[101] *Shapira D.* Generalized edit distance with move operations / D. Shapira, J. A. Storer // 13th Symposium on Combinatorial Pattern Matching. — 2002. — P. 85–98.

[102] *Shastri L.* From simple associations to systematic reasoning: Connectionist representation of rules, variables, and dynamic bindings using temporal syn-

chrony / L. Shastri, V. Ajjanagadde // *Behavioral and Brain Sciences*, 1993. — V. 16, № 3. — P. 417–494.

[103] *Smith T.* Identification of common molecular subsequences / T. Smith, M. Waterman // *J. of Molecular Biology*, 1981. — V. 147, № 1. — P. 195–197.

[104] *Sokolov A.* An adaptive detection of anomalies in user's behavior / A. Sokolov // Proc. of the Int. Joint Conf. on Neural Networks. — V. 4. — Portland, Oregon,US: 2003. — P. 2443–2447.

[105] *Sokolov A.* Nearest string by neural-like encoding / A. Sokolov // Proc. of 12th Int. Conf. Knowledge-Dialogue-Solution. — Varna, Bulgaria: FOI BG, 2006. — P. 101–106.

[106] *Sokolov A.* Searching for nearest strings with neural-like string embedding / A. Sokolov // *Information Theories and Applications*, 2007. — V. 14, № 3. — P. 294–299.

[107] *Sokolov A.* Approaches to sequence similarity representation / A. Sokolov, D. Rachkovkij // *Information Theories and Applications*, 2005. — V. 13, № 3. — P. 272–278.

[108] *Sokolov A.* On handling replay attacks in intrusion detection systems / A. Sokolov, D. Rachkovskij // *Information Theories & Applications*, 2003. — V. 10, № 3. — P. 341–348.

[109] *Sokolov A.* Some approaches to distributed encoding of sequences / A. Sokolov, D. Rachkovskij // Proc. of 11th Int. Conf. Knowledge-Dialogue-Solution. — V. 2. — Varna, Bulgaria: FOI BG, 2005. — P. 522–528.

[110] *Spink A.* Searching the web: a survey of excite users / A. Spink, J. Bateman, B. J. Jansen // *Internet Research: Electronic Networking Applications and Policy*, 1999. — V. 9, № 2. — P. 117–128.

[111] *Thorpe S.* Localized versus distributed representations / S. Thorpe // Handbook of Brain Theory and Neural Networks / Ed. by M. A. Arbib. — Cambridge, MA: MIT Press, 1995. — P. 549–552.

[112] *Tichy W.* The string-to-string correction problem with block moves / W. Tichy // *ACM Trans. Comput. Syst.*, 1984. — V. 2, № 4. — P. 309–321.

[113] *Ukkonen E.* On approximate string matching / E. Ukkonen // Proc. Int. Conf. on Foundations of Comp. Theory. — V. 158. — Springer-Verlag, Lecture Notes on Comp. Sci., 1983. — P. 487–495.

[114] *Ukkonen E.* Approximate string-matching with q-grams and maximal matches / E. Ukkonen // *Theoretical Computer Science*, 1992. — V. 92, № 1. — P. 191–211.

[115] *van Gelder T.* Distributed Versus Local Representation / T. van Gelder. — New York: The MIT Press, 1999. — P. 235–237.

[116] *van Rijsbergen C. J.* Information Retrieval, 2nd edition / C. J. van Rijsbergen. — London: Butterworths, 1979. — 208 p.

[117] Vipul's razor. — Sourceforge. — *http://razor.sourceforge.net/*.

[118] *Wagner R.* An extension of the string-to-string correction problem / R. Wagner, R. Lowrance // *J. of the ACM*, 1975. — V. 22, № 2. — P. 177–183.

[119] *Wagner R. A.* The string-to-string correction problem / R. A. Wagner, M. J. Fischer // *J. of the ACM*, 1974. — V. 21, № 1. — P. 168–173.

[120] *Zhang M. Q.* Computational prediction of eukaryotic protein-coding genes / M. Q. Zhang // *Nat. Rev. Genet.*, 2002. — V. 3, № 9. — P. 698–709.

[121] *Бокс Д.* Сущность технологии COM / Д. Бокс. — Санкт-Петербург: Питер, 2001. — 400 с.

[122] Веб-коллекция Narod.Ru. — Российский семинар по оценке методов информационного поиска. — *http://romip.ru/ru/collections/narod.html*.

[123] *Винцюк Т.* Распознавание устной речи методами динамического программирования / Т. Винцюк // *Кибернетика*, 1968. — № 1. — С. 81–88.

[124] *Гриценко В.* Концепция и архитектура программного нейрокомпьютера SNC / В. Гриценко, И. Мисуно, Д. Рачковский, Е. Ревунова, С. Слипченко, А. Соколов // *Управляющие системы и машины*, 2004. — № 3. — С. 3–14.

[125] *Джексон П.* Введение в экспертные системы / П. Джексон. — Издательский дом «Вильямс», 2001. — 624 с.

[126] *Косинов Д.* Использование статистической информации при выявлении схожих документов / Д. Косинов // Интернет-математика 2007: сб. работ участников конкурса науч. проектов по информ. поиску / Под ред. П. Браславский. — Екатеринбург: Изд-во Урал. ун-та, 2007. — С. 84–90.

[127] *Круглов В.* Искусственные нейронные сети. Теория и практика / В. Круглов, В. Борисов. — М.: Горячая линия, 2002. — 382 с.

[128] *Куссуль Н.* Адаптивное обнаружение аномалий в поведении пользователей компьютерных систем с помощью марковских цепей переменного порядка. Часть 2. Методы обнаружения аномалий и результаты экспериментов / Н. Куссуль, А. Соколов // *Проблемы управления и информатики*, 2003. — № 4. — С. 83–88.

[129] *Куссуль Э. М.* Ассоциативные нейроподобные структуры / Э. М. Куссуль. — Киев: Наукова думка, 1992. — 144 с.

[130] *Кузнецов С.* Порождение кластеров документов дубликатов: подход, основанный на поиске частых замкнутых множеств / С. Кузнецов, Д. Игнатов, С. Объедков, М. Самохин // Сб. работ стипендиатов. — Яндекс, 2005. — *http://company.yandex.ru/grant/2005/07_Kuznetsov_102820.pdf* .

[131] *Левенштейн В. И.* Двоичные коды с исправлением выпадений, вставок и замещений символов / В. И. Левенштейн // *Докл. АН СССР*. — Т. 163, № 4. — С. 845–848.

[132] *Мисуно И.* Модульный программный нейрокомпьютер SNC: реализация и применение / И. Мисуно, Д. Рачковский, Е. Ревунова, С. Слипченко, А. Соколов, А. Тетерюк // *Управляющие системы и машины*, 2005. — № 2. — С. 74–85.

[133] *Мисуно И. С.* Обработка текстовой информации с помощью векторных представлений / И. С. Мисуно, Д. А. Рачковский, С. В. Слипченко, А. М. Соколов // Международный семинар по индуктивному моделированию. — Киев: 2005. — С. 230–236.

[134] *Мисуно И. С.* Поиск текстовой информации с помощью векторных представлений / И. С. Мисуно, Д. А. Рачковский, С. В. Слипченко, А. М. Со-

колов // *Проблемы программирования*, 2005. — № 4. — С. 50–67.

[135] *Морозов А.* Нейрокомпьютеры и нейротехнологии накануне нового старта / А. Морозов, В. Клименко, А. Резник // *Управляющие системы и машины*, 1997. — № 1-2. — С. 1–7.

[136] Наборы данных конкурса «Интернет-Математика» // 2007. — *http://company.yandex.ru/grant/datasets_description.xml*.

[137] *Некрестьянов И.* Оценка систем информационного поиска / И. Некрестьянов // Курс лекций «Алгоритмы для Интернет» / Под ред. Ю. Лифшиц. — ИТМО, 2006.

[138] *Рачковский Д.* Концепция и методы нейросетевого распределенного представления информации в задачах ИИ / Д. Рачковский, И. Мисуно, Е. Ревунова, С. Слипченко, А. Соколов // Междунар. конф. "Проблемы нейрокибернетики". — Т. 2. — Ростов-на-Дону, Россия: 2005. — С. 30–33.

[139] *Рачковский Д.* Разреженное бинарное распределенное кодирование скалярных величин / Д. Рачковский, С. Слипченко, Э. Куссуль, Т. Байдык // *Проблемы управления и информатики*, 2005. — № 3. — С. 89–102.

[140] *Резник А.* Нейросетевая идентификация пользователей компьютерных систем / А. Резник, Н. Куссуль, А. Соколов // *Кибернетика и вычислительная техника*, 1999. — Т. 123. — С. 70–79.

[141] *Рiзник О.* Багатофункціональний нейрокомп’ютер NeuroLand / О. Рiзник, Е. Калина, О. Садова, О. Дехтяренко, О. Сичов // *Математичні машини і системи*, 2003. — № 1. — С. 36–45.

[142] *Сегалович И.* Некоторые автоматические методы детектирования спама, доступные большим почтовым системам / И. Сегалович // 2004. — *http://company.yandex.ru/articles/antispam.xml*.

[143] *Сегалович И.* Принципы и технические методы работы с незапрашиваемой корреспонденцией / И. Сегалович, Д. Тейблюм, А. Дилевский // 2004. — *http://company.yandex.ru/articles/spamooborona.html*.

[144] *Соколов А.* Обнаружение аномалий с помощью марковских цепей переменного порядка / А. Соколов // *Исскуственный интеллект*, 2002. —

№ 4. — С. 74–83.

[145] *Соколов А.* Современные модели обнаружения аномалий в компьютерных системах / А. Соколов // *Управляющие Системы и Машины*, 2004. — № 5. — С. 67–73.

[146] *Соколов А.* Векторные представления для эффективного сравнения и поиска похожих строк / А. Соколов // *Кибернетика и системный анализ*, 2007. — № 4. — С. 18–38.

[147] *Соколов А.* Исследование ускоренного поиска близких текстовых последовательностей с помощью векторных представлений / А. Соколов // *Кибернетика и системный анализ*, 2008. — № 4. — С. 32–47.

[148] *Соколов А.* Рандомизированное вложение расстояния редактирования в задачах поиска генов и обнаружения вторжений / А. Соколов // *Системные технологии*, 2008. — № 2. — С. 126–139.

[149] *Шлезингер М.* Десять лекций по статистическому и структурному распознаванию / М. Шлезингер, В. Главач. — Наукова думка, 2004. — 535 с.

[150] Яндекс. — Yandex Inc., 2008. — 2008. — *http://company.yandex.ru*.