УДК 004.032.26

А.М. СОКОЛОВ

ИССЛЕДОВАНИЕ УСКОРЕННОГО ПОИСКА БЛИЗКИХ ТЕКСТОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ С ПОМОЩЬЮ ВЕКТОРНЫХ ПРЕДСТАВЛЕНИЙ

Ключевые слова: *аппроксимация расстояния редактирования, приближенный поиск ближайшего соседа, нейросетевые информационные технологии, обнаружение спама, поиск дубликатов*.

1. ВВЕДЕНИЕ

Во многих прикладных задачах разных предметных областей существует необходимость сравнения длинных символьных последовательностей. Примерами таких областей являются веб-поиск, большие системы документооборота [1], генетика [2].

Для сравнения последовательностей необходимо задаться метрикой, которая должна быть адекватной задаче и эффективно вычислима. Первому условию часто удовлетворяет расстояние Левенштейна (классическое расстояние редактирования) [3]. Оно определяется 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 операции, происходящие при мутациях и эволюции генов, и как некоторые способы искажения текстов для несанкционированной рекламы в Интернет или искажения при оптическом распознавании текстов в системах документооборота.

Широко известен классический алгоритм вычисления расстояния Левенштейна, имеющий для строк длиной n сложность $O(n^2)$ [4]. Ввиду больших характерных размеров n, а также большого количества строк применение этого алгоритма в перечисленных выше областях весьма затруднительно. Поэтому требуется вычислительно эффективная оценка расстояния редактирования [5].

В работе [6] разработан метод оценки расстояния Левенштейна на основе вложения расстояния редактирования в векторное пространство, а также предложен алгоритм нахождения приближенных ближайших строк на основе модификации схемы локально-чувствительного хеширования (locality-sensitive hashing — LSH [7]), использующей p-стабильные распределения [8]. Однако предложенные методы требуют исследования в экспериментах и приложениях.

В настоящей статье описаны результаты численных экспериментов по проверке теоретических результатов предложенной схемы на искусственных данных. Рассматривается применение предложенного алгоритма приближенного нахождения ближайших строк к следующим актуальным практическим задачам фильтрации дубликатов (в поисковых машинах, системах документооборота и др.) и обнаружения спама, т.е. сообщений, как правило, рекламного характера, массово рассылаемых людям, не выразившим желание их получать (для веб-страниц, электронной почты и т.п.).

2. ВЛОЖЕНИЕ РАССТОЯНИЯ РЕДАКТИРОВАНИЯ

Опишем суть предложенного и обоснованного в работе [6] детерминированного и вероятностного вложения классического расстояния редактирования в векторное пространство, а также схемы поиска приближенных дубликатов символьных строк на основе предложенного вложения.

2.1. Детерминированная схема

Для некоторой символьной строки $x \in \Sigma^n$, а также параметра $q \in N$ назовем q-граммой подстроку длиной q. Назовем вектор вида $v_{n,q}(x) \in (N \cup \{0\})^{|\Sigma|}$ q-граммным вектором, где каждой q-грамме $\sigma \in \Sigma^q$ соответствует элемент вектора $(v_{n,q}(x))_\sigma \in N \cup \{0\}$, значение которого — число вхождений σ в x. Манхетенново (l_1) расстояние между такими векторами, т.е. сумма модулей разностей значений элементов векторов, называется q-граммным расстоянием [9] и обозначается $d_q(x, y)$.

Пусть заданы окно шириной w символов и два значения длины q-грамм — q_1 и q_2, $q_1 < q_2$. Строка x преобразовывается в вектор $v(x)$ конкатенацией всех q-граммных векторов

$$v_{i,w,q} = v_{w,q}(x[i, i+w-1]) \tag{1}$$

для $q = q_1, q_1+1, \ldots, q_2$ и $i = 1, \ldots, n-w+1$.

Расстоянием между такими векторами будем считать q-граммное расстояние, и на его основании определим новое расстояние между строками $x, y \in \Sigma^w$:

$$d^{\Sigma}_{w,q_1,q_2}(x, y) = \Sigma_{q=q_1,\ldots,q_2} d_q(x, y). \tag{2}$$

Обозначим $\Delta q = q_2 - q_1$. Для строк $x, y \in \Sigma^n$ определим с использованием (1) расстояние

$$D(x, y) = \Sigma_{i=1,\ldots,n-w+1} d^{\Sigma}_{w,q_1,q_2}(x[i, i+w-1], y[i, i+w-1]) / ((n-w+1)(\Delta q+1)). \tag{3}$$

Пусть $q_1 = 2w/3$, $\Delta q = \left\lfloor 1/2(-7 + (57 + 16(w-q_1))^{1/2}) \right\rfloor$, $Q = (\Delta q+1)(\Delta q+2)$, $t = w - \Delta q + 1$. В [6] показано, что

если $\mathrm{ed}(x, y) > k_2$, то $D(x, y) \geq Qt(k_2/(2(\Delta q+1)) - 2)/((n-w+1)(\Delta q+1)) = d_2$; (4)

если $\mathrm{ed}(x, y) \leq k_1$, то $D(x, y) \leq 2k_1[w^2 + (n+1)]/(n-w+1) = d_1$, (5)

где $\mathrm{ed}(x, y)$ — расстояние редактирования.

При условии $d_2 > d_1$ чем меньше разность между k_1 и k_2, тем точнее можно аппроксимировать $\mathrm{ed}(x, y)$ при известном $D(x, y)$. Положим $w = n^\gamma$. Тогда показатель роста w, обеспечивающий наибольшую точность аппроксимации, достигается при $\gamma = 0.5$. При этом $k_2 = \Omega(k_1 n^{1/2})$, время построения векторов $v(\cdot)$ равно $O(n^{3/2})$, а их размерность — $O(n^{5/4})$.

Полученные оценки времени построения и размерности получаемых векторов в детерминированной схеме слишком велики для эффективного использования на практике. Поэтому в [6] была предложена рандомизация этой схемы, применимая для задачи поиска приближенно ближайших соседей и менее требовательная к ресурсам.

2.2. Рандомизированная схема

Пусть вектор $v^i_{w,q_1,q_2}(x)$ является конкатенаций q-граммных (от q_1 до q_2) векторов (1) подстроки $x[i, i+w-1]$ длиной w, где i выбрано случайно и равновероятно из множества возможных позиций окна шириной w: $i = 1, \ldots, n-w+1$. Размерность вектора $v^i_{w,q_1,q_2}(x)$ равна $\Sigma_{q=q_1,\ldots,q_2} |\Sigma|^q$.

Пусть φ^i — случайный вектор такой же размерности, как и $\upsilon^i_{w,q_1,q_2}(x)$, с элементами, выбранными случайно из распределения Коши $p(x)=(\pi(1+x^2))^{-1}$. Построим для строки x хеш-вектор размерностью K: $h(x)=(h_1(x),h_2(x),\dots,h_K(x))$, где

$$h_i(x)=\left\lfloor((\upsilon^i_{w,q_1,q_2}(x),\varphi_i)+b_i)/r\right\rfloor, \tag{6}$$

r и b_i — действительные числа, b_i выбрано случайно и равновероятно из диапазона $[0,r]$.

Если определить шар $B(q,k)=\{x\in\Sigma^n\,|\,\mathrm{ed}(q,x)\le k\}$, то семейство $H=\{h:\Sigma^n\to X\}$ (где X — некоторое конечное или счетное множество значений) называется [7] (k_1,k_2,p_1,p_2)-чувствительным или кратко локально-чувствительным, если для любых $x,y\in\Sigma^n$ и любой независимо и равновероятно выбранной хеш-функции $h\in H$ выполняются условия:

$$\text{если } x\in B(y,k_1),\ \text{то } \mathrm{Prob}[h(x)=h(y)]>p_1, \tag{7}$$

$$\text{если } x\notin B(y,k_2),\ \text{то } \mathrm{Prob}[h(x)=h(y)]<p_2. \tag{8}$$

Для того чтобы семейство H позволяло отличить «близкие» строки от «дальних», необходимо выполнение условия $k_1<k_2$, т.е. близкая строка x должна находиться ближе к y, чем дальняя, при этом $p_2<p_1$, т.е. близкие строки должны вызывать коллизию хеш-функций с большей вероятностью, чем дальние.

В [6] доказывается, что (6) есть локально-чувствительной функцией (при $w=n^{2/3}$, $r=w$ и q_1, Δq, Q, t таких, как в детерминированной схеме) и на ее основе можно построить процедуру поиска ближайших строк, воспользовавшись следующей общей схемой из [9, 10].

Алгоритм поиска ближайшей строки с помощью LSH. Пусть имеется база P строк одинаковой длины n. Необходимо на запрос $q\in\Sigma^n$ возвратить приближенного ближайшего соседа — строку, находящуюся в шаре $B(q,k_2)$. Все строки $x\in P$ запоминаются в ячейках памяти следующим образом.

1. По формуле (6) для каждой строки x базы P генерируется L K-мерных случайных хеш-векторов $h^j(x)=(h^j_1(x),h^j_2(x),\dots,h^j_K(x))$, $j=1,\dots,L$.

2. Для каждого уникального хеш-вектора, полученного из всех строк базы $h^j(x)$, $j=1,\dots,L$, $x\in P$, создается ячейка памяти. Общее число ячеек $C\le L\cdot|P|$.

3. Каждая строка базы P ставится в соответствие тем ячейкам, хеш-векторы которых ею продуцируются.

Для нахождения приближенного ближайшего соседа к поступившей строке $q\in\Sigma^n$:

1) формируется L ее хеш-векторов;

2) просматриваются ячейки, хеш-векторы которых полностью совпадают со сформированными для q, соответствующие каждому хеш-вектору $h^j(q)=(h^j_1(q),h^j_2(q),\dots,h^j_K(q))$, $j=1,\dots,L$;

3) составляется список S содержащихся в просмотренных ячейках строк из базы P. Так как одна и та же строка базы P может встречаться в S несколько раз, то этот список можно представить в виде мультимножества.

Процедура заканчивается или по просмотру всех ячеек, соответствующих хеш-векторам $h^j(q)$, $j=1,\dots,L$, или когда размер списка S достигнет $2L$.

В [6] определяются значения вероятностей p_1 и p_2 и доказывается следующая теорема.

Теорема 1. При описанном построении векторов, значении $\rho = \log(p_1 / p_2)$ и при значениях

$$K = \log_{1/p_2} |P|, \tag{9}$$

$$L = |P|^{\rho} \tag{10}$$

алгоритм поиска выдаст в S с вероятностью большей, чем 1/2, строку y такую, что $\mathrm{ed}(x, y) \le k_2$, где $k_2 = O(zn^{1/3} \ln n)$, z — параметр, который влияет на значение k_2 и значения вероятностей p_1, p_2.

Для экспериментов, описываемых ниже, были приняты параметры $k_1 = 1$, $z = 1.01$. Описанное хеширование строки может быть также выполнено в виде нейросетевого распределенного представления [10].

Реализация поиска ближайшей строки с помощью LSH-леса. Для программной реализации описанной процедуры LSH здесь используется модификация описанной схемы LSH, называемая LSH-лес, которая позволяет находить ближайших соседей без обновления хеш-векторов при изменении размера базы P или параметра k_2 [11].

Для каждого $j = 1, \ldots, L$ все хеш-векторы $h^j(p)$ всех строк p базы P хранятся в виде отдельного префиксного дерева T_j глубиной до K уровней (корень имеет глубину 0, листья — глубину K). Узлы дерева соответствуют значениям элементов хеш-вектора и содержат ссылки на строки, хеш-векторы которых соответствуют пути от корня дерева к данному узлу. Листья деревьев соответствуют ячейкам в оригинальной схеме. Так, если в хеш-векторах двух строк совпадают первые k их элементов, то и первые k узлов на пути от корня дерева T_j до листьев, соответствующих этим строкам, также совпадают.

При поступлении запроса-строки q:

— формируются L его K-мерных хеш-векторов $h^j(q)$;

— для каждого хеш-вектора $h^j(p)$ в T_j находится узел, соответствующий $h^j(p)$ с наибольшим числом совпадающих первых элементов этих векторов;

— начиная от найденных выше узлов, все L деревьев синхронно просматриваются по направлению к корню и соответствующие указанным узлам строки p добавляются в результирующее мультимножество S строк;

— после того, как все строки, хеш-векторы которых совпадают на данном уровне, добавлены в S, повторяем процедуру для следующего (более высокого) уровня, пока не достигнем корня или $|S|$ не превысит $2L$.

В результате получаем мультимножество S строк (кандидатов на приближенного ближайшего соседа к q), 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.

Согласно теореме 5.1 из [11] S будет содержать с ненулевой константной вероятностью (аналогично теореме 1) приближенных ближайших соседей к запросу q.

3. ЧИСЛЕННОЕ ИССЛЕДОВАНИЕ ТЕОРЕТИЧЕСКИХ ОГРАНИЧЕНИЙ

3.1. Экспериментальная база RandomStrings

Для проверки теоретических результатов проведены эксперименты с детерминированной и вероятностной схемой на искусственно сгенерированных данных. Некоторая (случайная) строка — центр q случайно редактировалась с использованием операций вставки, удаления и замены. Полученные при таком редактировании строки составляли коллекцию строк. Поскольку общее число операций на-

много меньше длины строки, а посчитать расстояние редактирования не представляется возможным ввиду вычислительной сложности, то положим, что расстояние редактирования приблизительно равно сумме количеств искажений. Аналогичное допущение было принято в [12]. В дальнейшем будем ссылаться на этот набор строк как на RandomStrings.

3.2. Детерминированная схема

Проверка теоретических ограничений. Экспериментальное исследование того, попадает ли величина расстояния $D(x, y)$ из (3) в интервал (4) и (5), выполнялось для строк из базы RandomStrings. На рис. 1 приведены минимальные (крестики) и максимальные (нолики) экспериментальные значения величины $D(x, y)$ для каждого из значений расстояния редактирования $\mathrm{ed}\,(x, y)$ для $n = 5000$ и $n = 10000$. График становится пологим вблизи максимально возможного значения расстояния $D(x, y) = [2(w+1) - q_2 - q_1]$ (40 для $n = 5000$ и 58 для $n = 10000$), когда в каждом окне имеется $2(w - q + 1)$ различных q-грамм для $q = q_1, q_1 + 1, \ldots, q_2$. Теоретические значения верхней (4) и нижней (5) границ $D(x, y)$ изображены соответственно сплошной и пунктирной тонкими линиями. Видно, что экспериментальные данные соответствуют теоретическим оценкам.

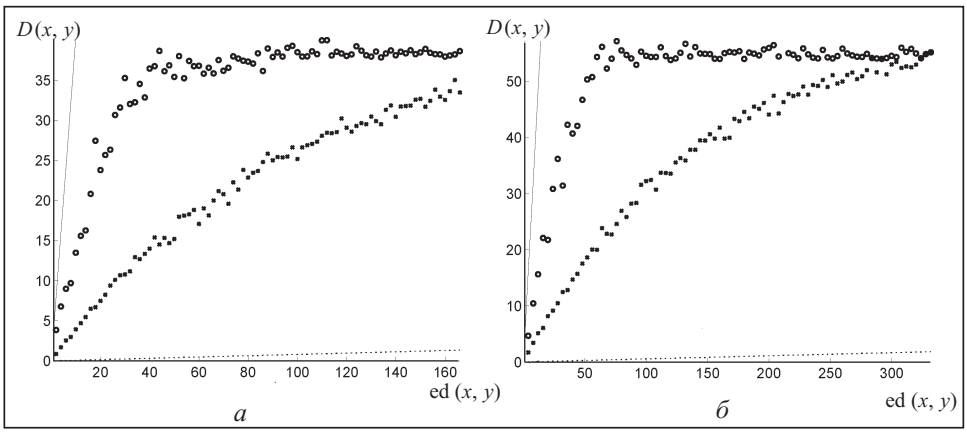

Рис. 1. Типичная зависимость расстояния $D(x, y)$ от ed (x, y) для $n = 5000$ (*а*) и $n = 10000$ (*б*) символов

3.3. Рандомизированная схема

Проверка теоретических ограничений. Как и для детерминированного варианта схемы, на искусственно сгенерированных строках разной длины для рандомизированного варианта экспериментально проверялось, попадают ли теоретически предсказанные значения вероятностей p_{col} коллизии хеш-функции (6) в пределы (7) и (8). На рис. 2 приведены экспериментально полученные вероятности совпадения значений хеш-функции (6) для строк длиной $n = 1000$ и $n = 3000$ вместе с теоретическими точками верхней (нолики) и нижней (крестики) границ. Видно, что эти экспериментальные данные также соответствуют теоретическим оценкам.

Выбор значения L и дополнительная фильтрация. Поскольку необходимые ресурсы растут линейно относительно L, на практике значения L из (9) оказываются слишком велики [11]. Однако при использовании меньших значений не гарантируется, что в возвращенное мультимножество S попадет хотя бы одна строка y такая, что $\mathrm{ed}(q, \mathrm{y}) \le k_2$. Следующие эксперименты выполнялись для проверки того, как влияет на «качество» мультимножества S выбор меньших (чем теоретические) значений L, а также дополнительная фильтрация мультимножества S, которой предположительно можно отсечь строки, где $\mathrm{ed}(q, y) > k_2$ (false positives).

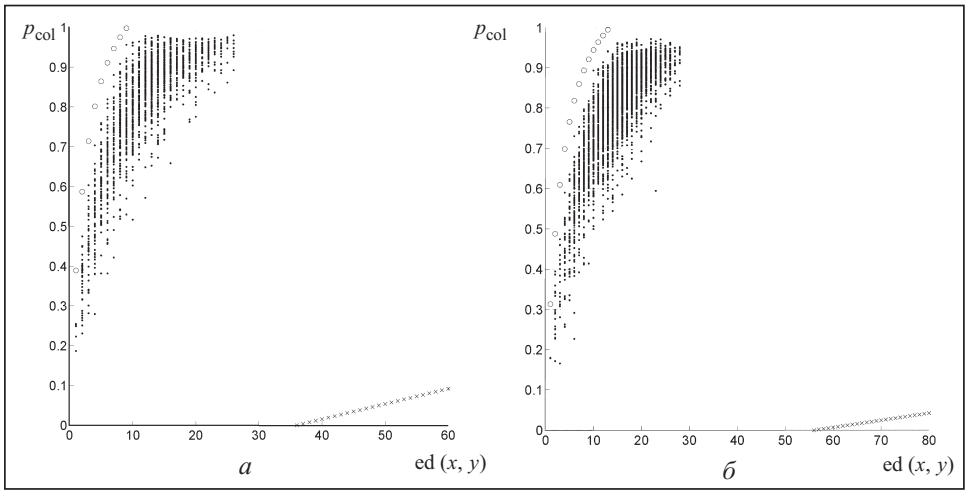

Рис. 2. Зависимость экспериментальных значений вероятности коллизии хеш-функций $h_i(x)$ и $h_i(y)$ от расстояния редактирования между строками длиной 1000 (*а*) и 3000 (*б*) символов

Проводились эксперименты следующими способами оценки качества мультиножества S.

А. Способ оценки по значению точности.

Б. Способ оценки по упорядоченности мультимножества S относительно известного эталона.

Эксперимент по оценке качества *S* согласно значению точности. Традиционно в системах поиска для исследования качества множества документов, возвращаемого на запрос, анализируется компромисс между количеством релевантных текстов (true positives) и количеством нерелевантных текстов (false positives). Для этого определяются величины точности ₱ (отношение количества релевантных запросу возвращенных документов к общему числу возвращенных документов) и полноты r (отношение количества релевантных запросу возвращенных документов к общему числу релевантных документов). Эти величины изображаются обычно на графиках зависимости точности от полноты [13].

В настоящей статье данный подход оказывается неинформативным, поскольку рассматриваемая задача поиска приближенного ближайшего соседа подразумевает нахождение одного соседа, а не множества соседей. Поэтому отношение (полнота) $|B(q,k_2)\cap P\cap S|/|B(q,k_2)\cap P|$ неинформативно.

Ситуация с неинформативностью полноты возникает и в оценке качества поисковых систем: для пользователя все множество возвращенных результатов не представляет интереса, он обычно ограничивается просмотром только первых страниц результатов [14]. В таких случаях вместо точности и полноты обычно используют характеристику точности на уровне n (precision at n / $P@n$) [15], вычисляемую как точность на ограниченном первыми n результатами множестве. В рассматриваемом случае этот уровень естественно ограничен размером мультимножества S и точность на уровне $|S|$ определяется как $|B(q,k_2)\cap P\cap S|/|S|$.

Чтобы не вносить смещения в оценку точности на уровне $|S|$ ввиду неодинакового количества строк внутри и вне шара $B(q,k_2)$ и различного количества строк на определенном расстоянии от q, мы отобрали из набора RandomStrings (разд. 3.1) 2200 строк длиной 1000 символов (при этом $k_2=126$) таким образом, что число строк, принадлежащих $B(q,k_2)$, составляло половину от общего числа строк и количество строк для каждого значения расстояния от центра q не превышало 10. Полученное множество P содержало строки, находящиеся от центра q на расстоянии редактирования от 10 до 248.

Множество P было запомнено в LSH-лесе с помощью рандомизированной схемы из п. 2.2. На вход процедуры поиска приближенного ближайшего соседа подавался запрос q. По полученному на выходе при каждом запросе множеству S ($|S| > 4L$) вычислялась точность на уровнях $0.5L$ (где L — четное), L, $2L, 3L, 4L$. Значения точностей усреднялись по 100 случайным независимым реализациям LSH-леса. Их значения и дисперсия $\sigma_{\text{р}}$ приведены в табл. 1.

Таблица 1

L	Значение точности на уровне $\|S\|$ для									
	$\|S\| = 0.5L$	$\sigma_{\text{р},0.5L}$	$\|S\| = L$	$\sigma_{\text{р},1L}$	$\|S\| = 2L$	$\sigma_{\text{р},2L}$	$\|S\| = 3L$	$\sigma_{\text{р},3L}$	$\|S\| = 4L$	$\sigma_{\text{р},4L}$
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

При малых значения 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 $P \setminus B(q, k_2)$ попасть в S, но в дальнейшем точность перестает существенно изменяться и 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, например, на теоретически рекомендованном значении $2L$.

Эксперимент по оценке упорядоченности мультимножеств. 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: $S' = \{\text{ed}(x, q) \mid x \in S\}$.

Для сравнения упорядоченных мультимножеств возьмем за основу обобщенное расстояние Кэндалла [16], которое позволяет сравнивать упорядоченные множества, вводя различные штрафы за нахождение элементов в разных относитель-

ных порядках в множествах M_1 и 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. каждый элемент входит соответственно в свое множество, то $u = p$.

Параметр p есть регулируемый штраф за ситуацию, когда первый (второй) элемент отсутствует в одном множестве, но присутствует в другом и наоборот. Для данных экспериментов использовалось $p = 1$.

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

$$D_K(M_1, M_2) = \Sigma_{i,j \in M_1 \cup M_2} u(i, j). \qquad (11)$$

Здесь изменен описанный алгоритм подсчета обобщенного расстояния Кэндалла с целью применения его к упорядоченным мультимножествам (значений расстояний редактирования от q до ближайших строк) следующим образом. Для каждого значения расстояния редактирования из $S'_1 \cup S'_2$, входящего хотя бы в одно из мультимножеств, каждое его j-е вхождение в оба мультимножества последовательно преобразуется 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 равен единице. 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: $S'_1 \to M_1 = \{1_1, 2_1, 3_1, 4_1, 4_2, 3_2, 5_1\}$, $S'_2 \to M_2 = \{1_1, 3_1, 3_2, 4_1, 4_2, 3_3, 4_3\}$. Множество элементов, полученное описанным способом из S', будем обозначать $M(S')$.

Таким образом, сведена задача сравнения упорядоченных мультимножеств S'_1 и S'_2 к задаче сравнения упорядоченных множеств $M(S'_1)$ и $M(S'_2)$.

Как и в эксперименте по вычислению точности на уровне $|S|$ (см. п. А) количество строк в S, возвращаемых процедурой LSH-лес и фигурирующее в описании алгоритма как $2L$, было переменным, т.е. алгоритм возвращал указанное количество строк, полученных в результате подъема по LSH-деревьям.

Исследование проведено на наборе строк RandomStrings (см. п. 3.1). Использовались те же 2200 строк длиной 1000 символов ($k_2 = 126$), что и в предыдущем эксперименте. Множество P было запомнено в LSH-лесе с помощью процедуры, рассмотренной в п. 2.2. На вход процедуры поиска приближенного ближайшего соседа подавался запрос q.

Обозначим X' упорядоченное по возрастанию мультимножество значений реальных расстояний редактирования от центра q до его ближайших соседей, а Y' — упорядоченное в ходе процедуры LSH-лес мультимножество значений расстояний редактирования от центра q до возвращенных строк в ходе процедуры LSH-лес. Вычислим вышеописанное расстояние $D_K(M(X'), M(Y'))$ из (11) при $|S'| = 50$ и $|S'| = 200$, где размер множества $M(X')$ ограничивался по значению $|M(Y')| = |S'|$.

Результаты усреднялись по 100 случайным независимым реализациям LSH-леса. На рис. 3 показана зависимость расстояния $D_K(M(X'), M(Y'))$ от количества деревьев L в лесе для различных ограничений на максимальный размер возвращенного множества S' при использовании следующих способов его дополнительной фильтрации:

1) all (без фильтрации);

2) = = max (только строки, совпавшие с запросом q на самом глубоком уровне для данного S);

3) = = half (строки, совпавшие на уровне от $\lfloor K_{\text{avg}} \rfloor$ до $\lceil K_{\text{avg}} \rceil$;

4) >= half (строки, совпавшие на уровне, большем или равном K_{avg}),

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

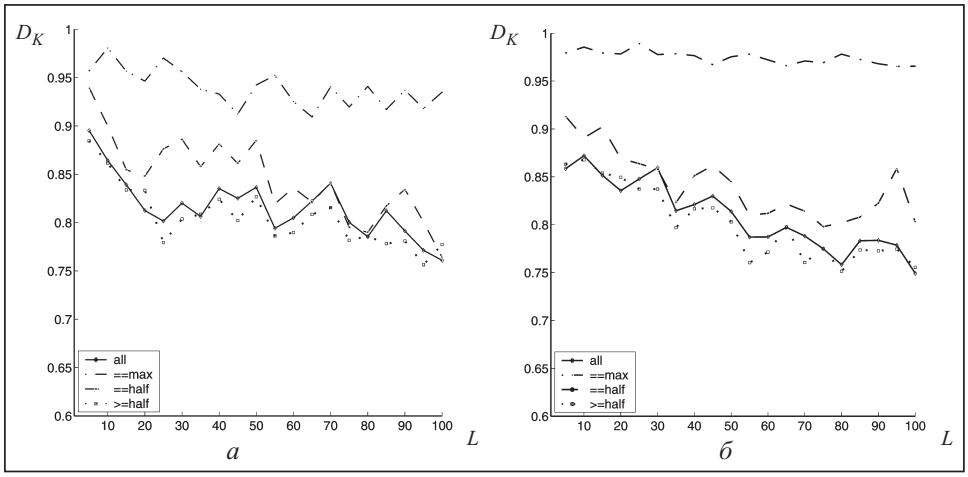

Рис. 3. Зависимость D_K от количества деревьев L для размеров мультимножества $|S'|= 50$ (*а*) и мультимножества $|S'|= 200$ (*б*)

Как видим, $D_K(M(X'), M(Y'))$ незначительно уменьшается при увеличении L, что можно объяснить увеличением $|S|$. Поэтому как и в предыдущем эксперименте можно сделать вывод о возможности фиксирования L на небольшом значении.

Способы фильтрации = = max и = = half имеют преимущество перед другими способами, подтверждая этим, что совпадение строк на более глубоких уровнях свидетельствует об их большем сходстве.

4. ЭКСПЕРИМЕНТЫ С ТЕКСТОВЫМИ КОЛЛЕКЦИЯМИ

Метод, рассмотренный в разд. 3, был применен в задаче поиска дубликатов в текстовых коллекциях, а также в задаче фильтрации спама в электронной почте. Использованный подход к решению этих задач основан на поиске приближенных дубликатов текстовых данных.

4.1. Задача поиска дубликатов в текстовых коллекциях

Операция поиска (приближенных) дубликатов текстовых документов требует эффективного выполнения в системах документооборота. Особенно востребованной эта операция становится в поисковых машинах Интернет. Документы, являющиеся приближенными дубликатами относительно один другого чрезвычайно распространены в Интернете. Яркими примерами являются 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 показа платной рекламы или перенаправления посетителей на другие сайты.

Описание коллекций. Поиск дубликатов осуществлялся в коллекциях текстов новостей Reuters-21578 [17] (21578 текстов длиной до 8316 символов) и учебных текстов British National Corpus [18] (4054 текстов диной до 2494232 символов).

Коллекция Reuters-21578 является стандартной коллекций для исследований в области обработки текстовой информации. Коллекция содержит тексты новостей агентства Reuters, среди которых много как полных, так и приближенных дубликатов (укороченные версии развернутых новостей, данные о котировках на биржах, отличающиеся датами и числами в тексте, и т.п.).

Коллекция учебных текстов British National Corpus (BNC) — большая коллекция текстов современного письменного и разговорного английского языка. Она представляет собой «золотой стандарт» и источник сведений о «правильном» английском языке (так, с помощью коллекции вычисляются вероятности префиксов, окончаний, слов, используемых в разных задачах). Теоретически дубликатов в BNC быть не должно, поскольку они искажают статистику правильного языка.

Методика поиска дубликатов. Для поиска дубликатов в текстовых коллекциях принимаем, что таковыми считаются те строки, у которых имеется хотя бы одно совпадение хеш-векторов длиной K. Находим дубликаты, число которых зависит от длины обрезки текста $n = 100, 150, 250, 500, 1000, 2000$ символах, а также количество дубликатов для текстов без обрезки (выравнивание по максимальной длине, условно обозначаемое $n = 0$). Используем следующие параметры схемы LSH: количество деревьев $L = 1, 5$; размерности хеш-векторов $K = 1, 2, 5, 10, 25, 50, 100, 150, 200$.

Использовалась предварительная фильтрация текстов, оставляющая только символы и цифры (C-функция isalnum()). Заголовки новостных текстов включались в текст, а прописные буквы заменялись строчными. Короткие тексты, длина которых меньше длины обрезки n, дополнялись до указанной длины одинаковым специальным символом в конце текста.

Определение количества найденных дубликатов в коллекциях. Найденное количество приближенных дубликатов в коллекциях Reuters-21578 и BNC в зависимости от значений K и n приведено соответственно в табл. 2 и 3 для $L = 1$ и $L = 5$. Использовались символы C-функции isalnum().

Таблица 2

K	Количество найденных приближенных дубликатов							
	$L = 1$							
	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	$L = 5$							
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 3

K	Количество найденных приближенных дубликатов							
	$L = 1$							
	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	$L = 5$							
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	–

Число приближенных дубликатов естественно уменьшается при увеличении K, стабилизируясь при больших K на значениях, приблизительно соответствующих количеству дубликатов, найденных методом, рассмотренным в работе [19] (320 дубликатов). Большое число приближенных дубликатов в случае, когда обрезки по фиксированной длине не используются (и их увеличение для эксперимента с символами isalnum() для $n = 2000$), объясняется большим сходством многих 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 $K \geq 10$ при визуальной проверке вызвано мелкими опечатками в текстах. При меньших значениях K эти опечатки могут не вызывать несовпадений хеш-векторов.

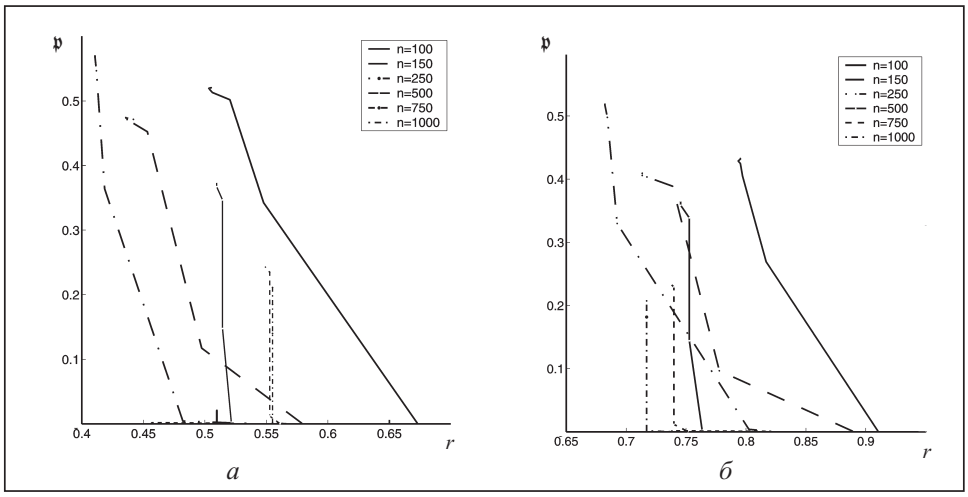

Рис. 4. Зависимость точности p от полноты r для значений sim = 0.95 (*а*) и sim = 0.99 (*б*) при $L = 5$

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

Сравнительные результаты поиска дубликатов. Результаты поиска дубликатов сравнивались с методом детерминированного вложения, описанным в [21]. За «золотой стандарт» были выбраны пары дубликатов как такой, на которой значение функции PERL String::Similarity (основанной на расстоянии редактирования) не меньше 0.85 (такой же подход применялся для создания коллекции дубликатов в [20]) .

Обозначим T_{sim} множество текстов со значением функции String::Similarity, бóльшим или равным sim. Принимая поочередно за «золотой стандарт» множества $T_{0.95}, T_{0.99}$, можно оценить качество работы нашего метода с помощью графиков точность–полнота путем изменения значения K (использовались значения $K = 5$, 10, 25, 50, 100, 150, 200). Как видно из рис. 4, при увеличении порога T_{sim} на значение функции String::Similarity полнота увеличивается, т.е. бóльшая доля «правильных» дубликатов попадает в мультимножество S, достигая единицы при $\text{sim} = 1$ (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.

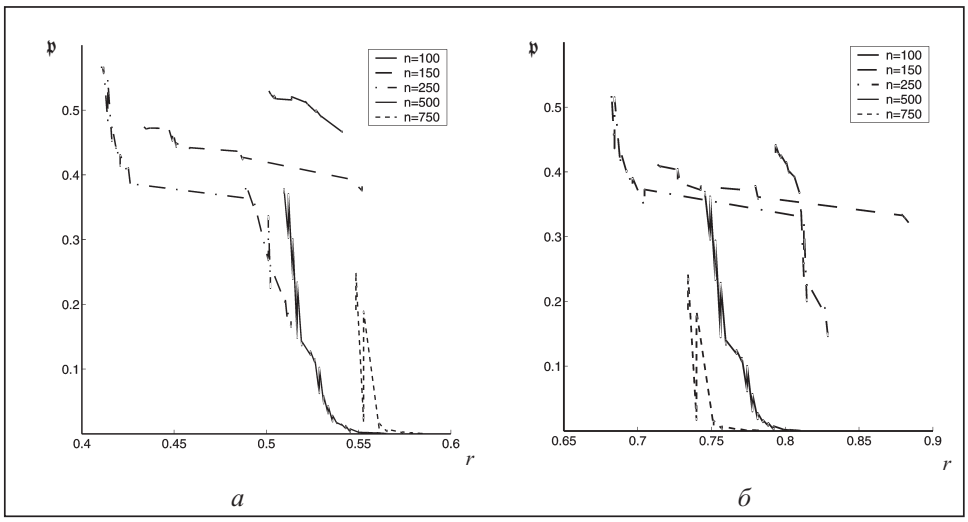

Рис. 5. Зависимость точности $\mathfrak{p}$ от полноты r для метода BY при sim = 0.95 (*а*) и sim = 0.99 (*б*)

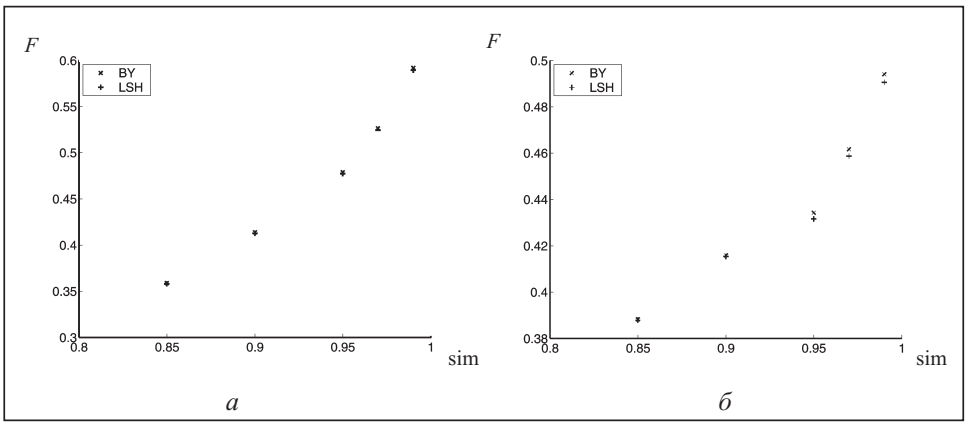

Рис. 6. Значения $F_{1\,\max}$ для $n = 250$ (*а*) и $n = 500$ (*б*)

Аналогичные графики были построены на рис. 5 для метода детерминированного вложения (BY), описанного в работе [21], 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 расстояния Хемминга между полученными векторами, определяющий, можно ли считать его дубликатом. Проверялись только тексты, где расстояние Хемминга не превышало 15% максимально возможного для данной длины n.

Для сравнения пар значений точность–полнота использовалась интегральная оценка F-measure с $\alpha = 1$, определяемая как $F_\alpha = (1+\alpha)r\wp / (\alpha\wp + r)$ [22], где r — полнота, $\wp$ — точность. Для каждой кривой, изображенной на рис. 4, 5, было подсчитано максимальное значение $F_{1\max}$, достигаемое на точках этой кривой. Полученные значения для $n = 250$ и $n = 500$ изображены на рис. 6. Точки (крестики), обозначенные в легенде BY, соответствуют алгоритму из [21], плюсики — алгоритму поиска с помощью LSH-леса.

Результаты показывают отсутствие существенных различий 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 в случае применения детерминированного метода BY из [21] — от часов до дней.

Исследовалось также качество поиска дубликатов на стандартной базе «Дубли Web-страниц коллекции РОМИП» [20] (предоставлена компанией «Яндекс»), содержащей список более 10 млн. пар веб-страниц, сходство между которыми по значению функции PERL String::Similarity не менее 0.85. Использовалась модификация метода поиска дубликатов, описанная выше (поиск дубликатов производился лишь среди документов, длина которых приблизительно равна длине запроса). Получены значения оценки F-measure до 0.88 в зависимости от порога на значение функции PERL String::Similarity и параметров K, L.

4.2. Задача оценки количества спама в коллекциях электронных писем

Обнаружение почтового спама является актуальной задачей, поскольку количество спама среди всей почты у среднего пользователя в 2005 году уже достигало 80–85% [23] и продолжает увеличиваться. Спам обнаруживают различными способами — в основном это проверки на специфические особенности спам-писем, такие как нахождение адреса адресанта в черном списке, применение разметки HTML в письме, подозрительные вложения. Используют также байесовскую классификацию [24].

Одной из распространенных спам-технологий, позволяющих преодолеть простейшие частотные фильтры, является внесение изменений в текст письма (например, специфическое написание слов, которые перестают быть точными копиями друг друга и искажают картину вероятностей слов или препятствуют предварительной обработке писем фильтрами [25]). Для борьбы с подобными технологиями некоторые исследователи предлагают обнаружение спама с использованием сравнения писем с ранее сохраненными [26]. Мы исследовали реализацию данной идеи на основе разработанных методов прямого сравнения текстовых строк-писем. Таким образом, мы оценили, какое количество спама может быть обнаружено всего лишь с помощью сравнения с ранее поступившими спам-письмами без применения специфических знаний и подробного анализа спам-технологий.

Описание использованных коллекций. Была использована тестовая база TREC 2006 Spam Track [27], широко применяемая разработчиками систем обнаружения спама для тестирования своих продуктов. Она содержит почтовые сообщения, размеченные экспертами как спам и не спам. Англоязычная часть базы TREC 2006 Spam Track содержит $|P| = 37822$ размеченных реальных почтовых сообщений объемом 189 Мб, из которых 24912 (66%) — спам.

Схема экспериментов и оценка эффективности. Коллекции в TREC и принятый способ оценки эффективности фильтров построены с учетом способа реального использования спам-фильтров конечными пользователями. Задача тестируемых фильтров — классифицировать подаваемые в хронологическом порядке сообщения и затем дообучиться после получения правильного класса для этого сообщения от эксперта. Этот процесс соответствует обычному порядку действий пользователя, когда он последовательно выбирает из входящих сообщений спам, позволяя, таким образом, 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 $sm\%$ (false positives) и проценту неправильно классифицированных неспам-сообщений $hm\%$ (false negatives). Изменением порога на значение score можно построить ROC-кривые (зависимость $sm\%$ от $hm\%$), по которым в TREC сравниваются различные алгоритмы.

Эксперименты и присвоение score. Предварительный фильтр оставлял в письмах одни лишь символы алфавита (C-функция isalpha()) и обрезал сообщение по размеру $n = 1000$. Подаваемые в хронологическом порядке сообщения из коллекций служили запросами для процедуры поиска приближенного ближайшего соседа с помощью LSH-леса. Значение L изменялось от 1 до 200. Значение K фиксировалось по формуле (9) LSH-схемы для заданных p_2, P (см. разд. 2).

На основании экспериментов, рассмотренных в разд. 3, выбраны два способа присвоения score письмам по уровню в LSH-лесе, которому принадлежат приближенные ближайшие соседи к входным письмам: по максимальному уровню $k_{\max} = K$ и по среднему уровню k_{avg}.

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

Результаты. На рис. 7 приведены ROC-кривые для способа назначения score по максимальному уровню (способ по среднему уровню дает подобные результаты). Как видим, при уровне $hm\% = 5 \div 10\%$ успешно обнаруживается приблизительно 80% спама для Spam Track 2006.

Экспериментально установлено, что реализовав идею обнаружения спама по приближенным дубликатам, можно отфильтровать значительную его часть, что позволяет судить о применимости данного подхода в больших почтовых серверах в качестве составной технологии (модуля) в более сложных системах. Чем больше централизован почтовый сервис и чем большее у него число пользователей, тем больший процент отфильтрованного спама можно получить.

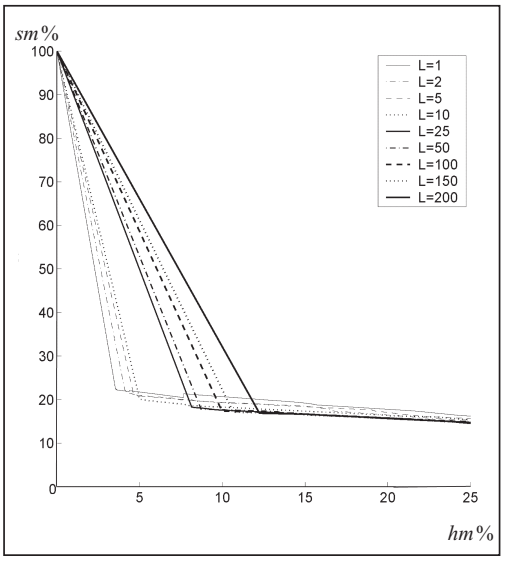

Рис. 7. Зависимость значения $sm\%$ от $hm\%$ для коллекции Spam Track 2006 для способа назначения score по максимальному уровню. Длина $n = 1000$

Из ROC-кривых на рис. 7 видно, что при $L = 1$ значения $hm\%$ часто получаются меньше, чем при $L > 1$. Это объясняется высокой степенью сходства части легаль-

ных писем со спамом, например ввиду использования html-формата, который оставлялся в письме, а не убирался и/или не анализировался его код, как это принято в реальных системах обнаружения спама.

5. ЗАКЛЮЧЕНИЕ

Проведенные эксперименты подтвердили эффективность метода вложения расстояния редактирования в векторное пространство, а также основанного на нем рандомизированного метода, как методов нахождения приближенно ближайших строк [6]. Показана возможность применения рандомизированного метода в реальных практических задачах поиска дубликатов и обнаружения спама. Учитывая при решении этих задач намеренное игнорирование информации о специфике предметной области, которая обязательно должна использоваться при решении реальных задач на практике, предложенный метод показал хорошие результаты в задаче оценки количества почтового спама и может применяться как составная часть в системах обнаружения спама.

Мы полагаем, что системы, основанные на обнаружении спама как приближенных копий, будут особенно эффективны в больших почтовых сервисах типа Gmail. Поскольку спам всегда направляется большому числу получателей, следует ожидать, что на почтовом сервисе обнаружить его намного легче, чем в рамках почтового ящика индивидуального пользователя, где 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 Vipul обнаружения спама, которая использует скетчи сообщений, отмеченных как спам участниками системы. В качестве таких скетчей можно рассмотреть предлагаемые нами хеш-векторы.

В дальнейшем предполагается применить подход к поиску приближенных дубликатов на основе предложенного метода вложения расстояния редактирования в других практических задачах, таких как поиск генов или анализ логов в компьютерных системах.

СПИСОК ЛИТЕРАТУРЫ

1. Brin S., Davis J., Garcia-Molina H. Copy detection mechanisms for digital documents // Proc. SIGMOD. — 1995. — P. 398–409
2. Gusfield D. Algorithms on strings trees and sequences. — Cambridge: Cambridge Univeristy Press, 1997. — 532 p.
3. Левенштейн В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов // Докл. АН СССР. — 1965. — **163**, вып. 4. — С. 845–848.
4. Винцюк Т.К. Распознавание слов устной речи методами динамического программирования // Кибернетика. — 1968. — № 1. — С. 81–88.
5. Indyk P. Open problems // Workshop on discrete metric spaces and their algorithmic applications / Ed. by Jiri Matousek. — Haifa, Israel, 2002.
6. Соколов А.М. Векторные представления для эффективного сравнения и поиска похожих строк // Кибернетика и системный анализ. — 2007. — № 4. — С. 18–38.
7. Indyk P., Motwani R. Approximate nearest neighbors: towards removing the curse of dimensionality // Proc. of 30th STOC. — 1998. — P. 604–613.
8. Locality-sensitive hashing scheme based on *p*-stable distributions / M. Datar, N. Immorlica, P. Indyk, V. Mirrokni // 20-th Symposium on Computational Geometry, 2004. — P. 253–262.
9. Ukkonen E. Approximate string-matching with *q*-grams and maximal matches // Theor. Comput. Sci. — 1992 — **92**, N 3. — P. 191–211.
10. Sokolov A. Nearest string by neural-like encoding // Proc. XI-th Conf. Knowledge-Dialogue-Solution. — Varna, Bulgaria, 2006 — P. 101–106.
11. Bawa M., Condie T., Ganesan P. LSH forest: self-tuning indexes for similarity search // Proc. of the 14th Conference on WWW. — New York: ACM Press, 2005. — P. 651–660.

12. Azenkot S., Chen T.-Y., Cormode G. An evaluation of the edit-distance-with-moves metric for comparing genetic sequences // DIMACS Technical Report 2005-39. — 2005.
13. Baeza-Yates R., Neto R. Modern Information Retrieval. — ACM Press Series/Addison Wesley, — New York, 1999 — 544 p.
14. Spink A., Bateman J., Jansen B.J. Searching the Web: Survey of EXCITE users // Internet Research: Electronic Networking Applications and Policy. — 1999. — **9**, N 4. — P. 117–128.
15. Hawking D., Voorhees E., Craswell N., Bailey P. Overview of the TREC8 Web Track // 8th Text REtrieval Conference. — Gaithersburg, 1999.
16. Fagin R., Kumar R., Sivakumar D. Comparing top *k* lists // SIAM J. on Discrete Mathematics, 2003. — P. 134–160.
17. Reuters-21578. — http://www.daviddlewis.com/resources/testcollections/reuters21578/
18. The British National Corpus. — http://www.natcorp.ox.ac.uk/
19. Sanderson M. Duplicate detection in the Reuters collection // Technical Report (TR-1997-5). — Department of Computing Science at the University of Glasgow. — Glasgow, UK, 1997.
20. Наборы данных конкурса «Интернет-Математика». — Яндекс. — http://company.yandex.ru/grant/datasets_description.xml. 2007
21. Approximating Edit Distance Efficiently / Z. Bar-Yossef, T. S. Jayram, R. Krauthgamer, R. Kumar // Proc. of the 45th IEEE Symposium on Foundations of Computer Science // IEEE, 2004 — P. 550–559.
22. Van Rijsbergen C.J. Information Retireval. — London: Butterworths, 1979 — 208 p.
23. Messaging anti-abuse working group. “Email Metrics Program: The Network Operators’ Perspective”. Report N 1 — 4th Quarter. — 2005.
24. Graham P. Plan for Spam. — http://www.paulgraham.com/stopspam.html, 2002.
25. Graham-Cumming J. The Spammers’ Compendium // Spam Conference at MIT, 2003. — http://www.jgc.org/tsc.html
26. Kolcz A., Chowdhury A., Alspector J. The impact of feature selection on signature-driven spam detection // Proc. of the 1st Conf. on Email and Anti-Spam, 2004. — http://www.ceas.cc/papers-2004/147.pdf
27. Cormack G.V. TREC 2006 Spam Track Overview // Proc. of the 15th Text REtrieval Conf. — NIST. — Gaithersburg, MD, 2006.

Поступила 09.08.2007